eLML - eLesson Markup Language

Responsible persons:

Joël Fisler

(Overall)

Susanne Bleisch

(Content)

Table Of Content

eLML - eLesson Markup Language

The *eLesson Markup Language (eLML)* [1] is an testopen source XML framework for creating structured eLessons using XML. For easier lesson authoring eLML we offer the web-based WYSIWYG **Firedocs eLML Editor** and to create eLML template layouts withouth any XSLT-knowledge you can use our new **Template Builder**. Once you created your eLML-lesson you can transform it into many different output-formats like **IMS Content Package or SCORM**, various **HTML-templates**, **eBooks (ePub format)**, **PDF**, **Office-Document (ODF)** and many more **listed under "Output Formats"**. Check out the **eLML-tools page** for more useful software and check the **download page** for the latest eLML release. The eLML project is **hosted at Sourceforge** and offers all the regular tools (CVS, bugtracker, forum etc.) that you might already be familiar with when working with Sourceforge.

Only pictures can be viewed in this version! For Flash, animations, movies etc. see online version. Only screenshots of animations will be displayed. [link]

The latest eLML news...

Feel free to include the eLML news using our **RSS feed**.

Only pictures can be viewed in this version! For Flash, animations, movies etc. see online version. Only screenshots of animations will be displayed. [link]

Prizes for eLML:

eLML received a **Learning Impact Award** silver medal at the **IMS Learning Impact Conference** held from May 17th to May 20th 2010 in Long Beach, California.

Joël Fisler (eLML project) thanks Rob Abel (CEO IMS) for the IMS Award

In September 2009 eLML von the **Swiss Open Source Award** in the category "Education".

[1] eLML, the eLesson Markup Language, is an XML framework developed by the GITTA project. The Swiss eLearning project GITTA started working with XML in 2001 but it was only after the official ending of the project in 2004 that its XML structure was released as an open source project under the name of eLML. For more information read the implementation chapter or visit www.eLML.org.

Renata Sevcikova and Immo Wille holding the Swiss Open Source Award Certificate in Winterthur

In July 2008 eLML won the **CATCon Gold Award** at the ISPRS conference in Beijing.

Joël Fisler and Susanne Bleisch holding the ISPRS Catcon 5 Award in Beijing, China

Furthermore eLML was one of the 19 finalists running for the **Medida-Prix 2008**! In the end **eLML did not win but GITTA**, the project behind eLML and created with eLML, won the second prize! Congratulations!

Austrian Minister Johannes Hahn and GITTA team members Monika Niederhuber and Joël Fisler with the €25'000 prize in Vienna, Austria

More Information about eLML

The history behind eLML

Screenshot of a GITTA lesson

Since eLML originally emerged from the *GITTA project* [2], let us begin with a short introduction about GITTA and its purpose:

"*GITTA was funded by the SVC* [3]*, a program initiated by the Swiss Confederation. In order to achieve a truly integrated Virtual Campus of relevant players in GIST* [4] *education in Switzerland, the GITTA consortium, made up of 10 partners spread throughout the country, covers a wide variety of disciplines and specifically integrates contributions of partners from universities, federal institutes of technology, and universities of applied sciences within a multilingual distribution (German, French and Italian).* " (Fisler 2006)

In comparison to other Swiss Virtual Campus (SVC) projects, GITTA spent a lot of time evaluating the available technologies. When they started the project in 2001, they were not satisfied with the existing eLearning software that was available in the market. Most of them were expensive proprietary systems that let you put in some information (mostly as HTML code) but offered few possibilities to design your content or even export it to other platforms. These systems were not deemed sustainable by the project leader. Since the goal of GITTA was to integrate eLearning into the regular curriculum of the participating universities, they could not build the project on proprietary software.

---

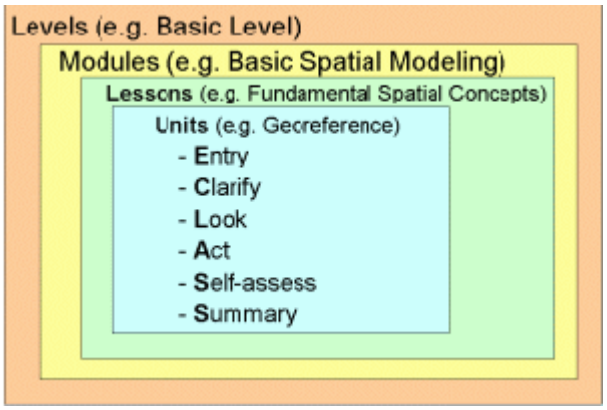
[2] GITTA is a Swiss eLearning project about GIS and it is the abbreviation for Geographic Information Technology Training Alliance. For more information about GITTA have a look at www.gitta.info.

[3] *SVC, the Swiss Virtual Campus, was founded in 1999 after a decision of the Swiss Parliament that over 50 Million Swiss Francs should be used to build up eLearning projects at Swiss universities. In the first project phase out of about 200 project drafts a total of 50 projects were accepted and supported. GITTA was one of them. For more information have a look at the SVC website.*

[4] *GIST is the abbreviation for Geographic Information Systems Technology.*

After months of evaluation the solution was to use XML to store the content and open source software (Apache Cocoon was used back then) to serve it. Based on a selected **pedagogical concept** a DTD (later an XML Schema) was developed. It described exactly what a GITTA lesson could or should contain and in which order. After nearly three years of using and improving the GITTA structure a decision was taken to make both the XML Schema and the corresponding XSL files to create both HTML (XSLT) and PDF (XSL-FO) freely available. EduTech, whose mandate was to provide technological support for SVC projects, supported the idea both financially and through valuable input. The published XML Schema with the corresponding XSL files were created again from scratch based on the GITTA DTD that had been successfully used for the last three years but leaving out the drawbacks and bugs. The result was named eLML and is presented in detail on the **structure page**.
Check the **contact page** to see who is or has been behind the development of eLML.

ECLASS - The pedagogical concept behind eLML



The ECLASS model used on unit level in eLML

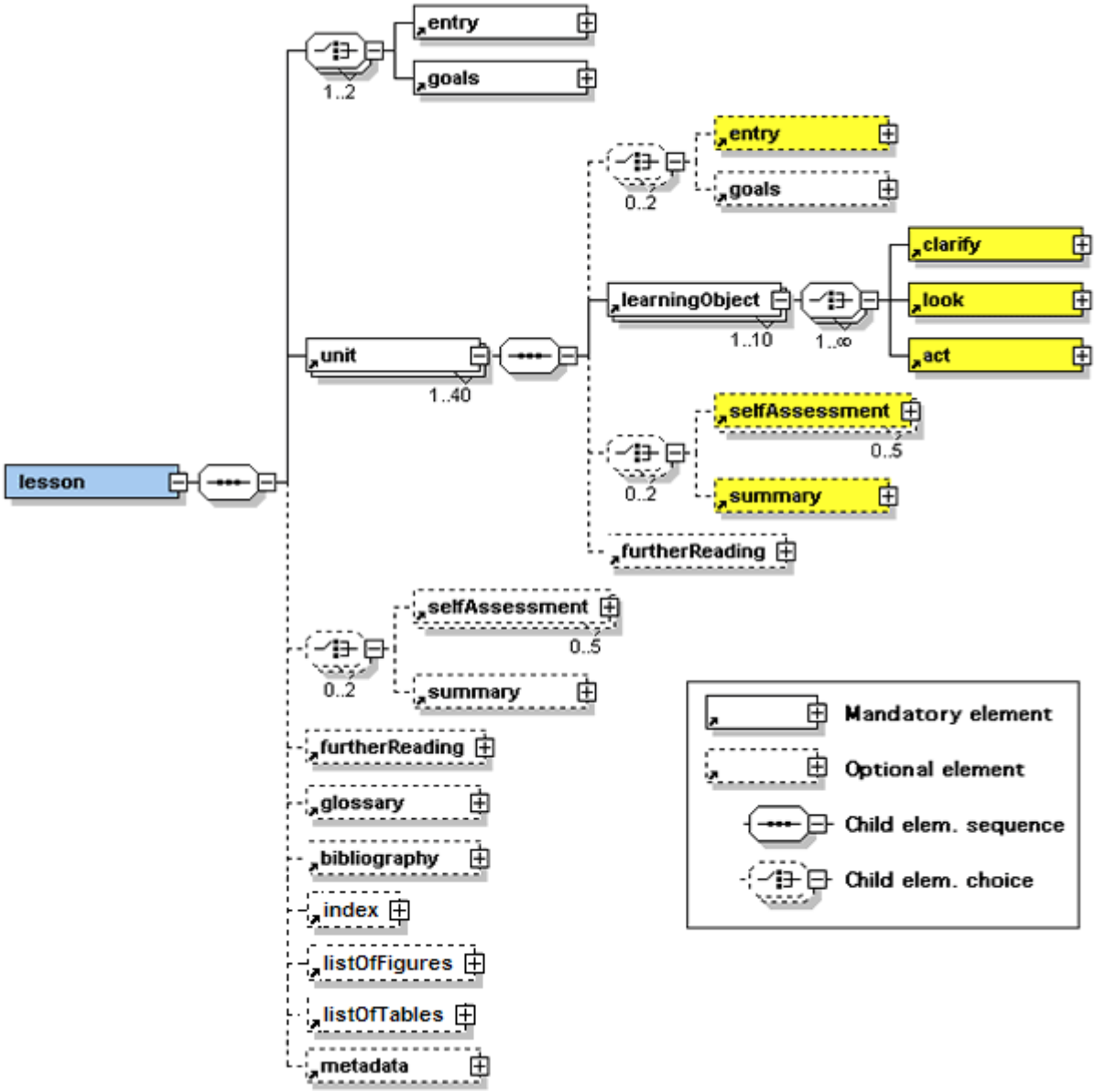
Diverse efforts have focused on research into finding an adequate didactic structure. Currently *eLML* is based on the *ECLASS* [5] structure (Gerson 2000). We adapted the model according to our needs as described below:

- Entry refers to the introductory statements made before each single lecture unit in a class. An example for an entry could be: What is to be discussed? Why is this topic being introduced? Originally the first E stood for "explain" which is also what the introduction does.
- Clarify represents the core of what is being taught in a unit and its key concepts. In this section, the reading of facts is inevitable. Module related concepts are conveyed. In our case, Data Presentation, a short example is shown to help students visualize the problem.
- Look allows students to review examples or samples of a model that will be taught. It defines the important aspects of the unit through illustrations, animations, videos, white board activities etc.
- Act is to encourage the student to practice what he or she has just been taught. It should be an important integral part of the online learning course, as it actively engages the student.
- Summary is a new point added to M. Gersons structure. It should sum up a unit and point out the main facts shown in this unit. It should contain what was learned and possibly also further expectations. In (2000) the second S stood for share, meaning group exercises. In the self-developed learning structure eLML, we introduced the Summary as second S and used only one exercise object, the self-assessment.

[5] ECLASS is based on Steven Gersons Guide to develop online courses. It is an abbreviation for the terms E = Entry; C = Clarify; L = Look; A = Act; S = Self-Assessment; S = Summary. Described in detail in the concept chapter.

The XML structure of eLML in detail

The **described ECLASS model** was mapped to an *XML* [6] structure and named *eLML*, the eLesson Markup Language. In the following illustration the blue "lesson" element is the root element and the yellow elements are the ones after which the ECLASS model was named:



The first three levels of the eLML structure in detail

The elements listed here are described in detail in the **top level elements** page. The subelements that can be used to format your content are described in the **content elements chapter**.

[6] XML, the eXtensible Markup Language, is a standard of the World Wide Web Consortium (W3C). XML documents use elements (tags) known from other markup languages like HTML. Using XSL transformations XML files can be transformed into other formats like XHTML or PDF. Many common used languages are based on XML: XHTML, SVG, GML, RSS, MathML etc. For detailed information about XML visit the W3C or read Wikipedia's explanation.

Interactive content is mainly created using Flash and *SVG* [7]. To prepare the content for common Internet browsers, the XML files are transformed using *XSLT* [8] provided by eLML. The final user interface is determined by easy-to-maintain eLML layout templates (mostly created using the **eLML Template Builder**) and served both as XHTML and as PDFor other **output formats**. Most authors use *CVS* [9] to store and update the lessons on the central **eLML repository server**.
Thanks to the use of standards like *XML*, *XSLT* and *SVG* eLML lessons can be viewed with any web browser (see image below) on any platform and are totally software independent. But because *eLML* supports both the *IMS Content Package* [10] and as of 2006 the *SCORM* [11] standard, the content can easily be imported into any modern *Learning Management System* [12] like *OLAT* [13], Blackboard or Moodle. Please refer to the **output formats chapter** for more technical background information.

[7] SVG, the Scalable Vector Graphics, is a standard of the World Wide Web Consortium (W3C). It is an open, XML-based format to describe graphics and animations and can be used as an alternative to the proprietary Adobe Illustrator and Macromedia Flash formats. To view SVG within a browser use either a modern browser (like Firefox or Apple's Safari) that has native support for SVG or download a plugin. For detailed information about SVG visit the W3C or read Wikipedia's explanation.

[8] XSLT, the XSL Transformations, is part of the Extensible Stylesheet Language (XSL) family and is a standard of the World Wide Web Consortium (W3C). XSLT files are used to transform XML files into other formats like HTML or formatting objects (FO) for generating PDF files. For detailed information about XSL visit the W3C or read Wikipedia's explanation.

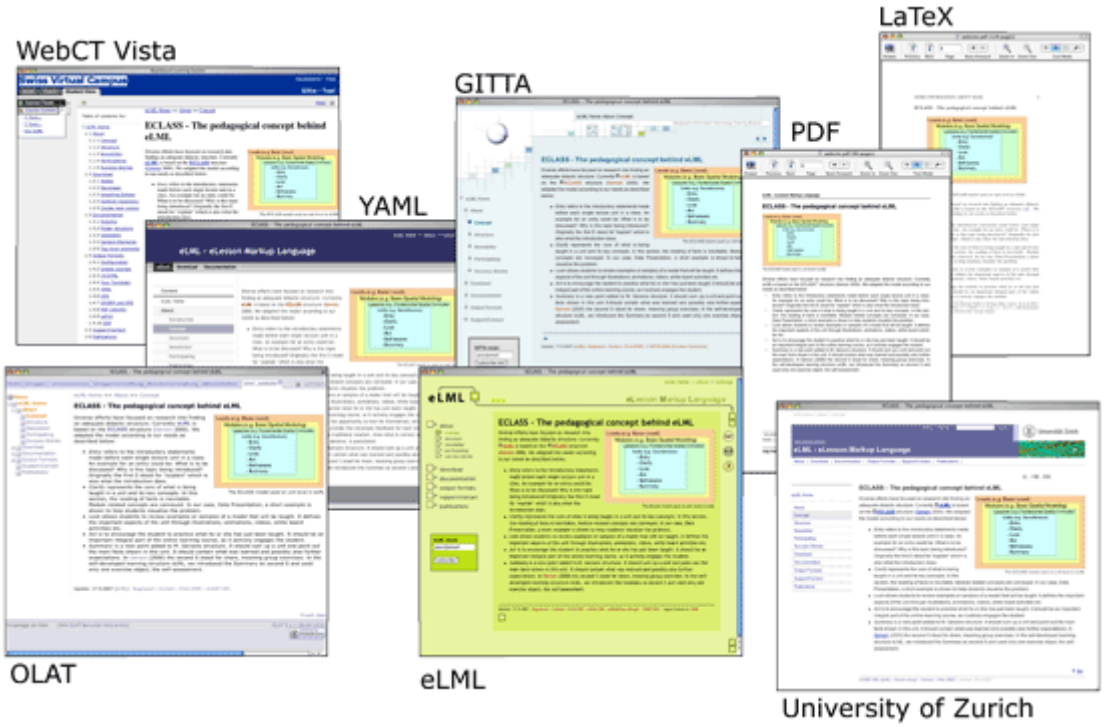
[9] CVS, the Concurrent Versions System, is the most widely used tool for controlling different versions of a source code and for a group of programmers to work simultaneously on a source code. Before working with a file, a user needs to do a "checkout" of the file from a "repository" stored on the project server. When writing updates back to the repository (called "committing"), CVS checks issues like access privileges, actual status of code and if no other group member meanwhile altered the code, CVS writes it back to the repository. By doing a "update" all project group members get the latest version of the code. CVS of course stores information about who altered which part of the code and automatically stores different versions of the code. Therefore using CVS it is possible to always reconstruct a former state of the code. eLML, and therefore also GITTA, uses CVS to store the XML code, images and multimedia elements of a lesson.

[10] The IMS Global Learning Consortium (usually known as IMS) is a non-profit standards organization concerned with establishing interoperability for learning systems and learning content and the enterprise integration of these capabilities. Their mission is to "support the adoption and use of learning technology worldwide". Some famous IMS standards are the CP (Content Package) standard used to import/export of content, the "Learning Resource Meta-data Specification" (LOM) or the QTI standard for question and test interoperability.

[11] The Shareable Content Object Reference Model (SCORM) is a standard for web-based eLearning. It defines how the individual instruction elements are combined at a technical level and sets conditions for the software needed for using the content. SCORM is distributed by the Advanced Distributed Learning (ADL) Initiative, a US organization under the Department of Defense (DoD).

[12] A Learning Management System (or LMS) is a software package, usually on a large scale (that scale is decreasing rapidly), that enables the management and delivery of learning content and resources to students. Most LMS systems are web-based to facilitate "anytime, anywhere" access to learning content and administration. Some famous open source LMS are OLAT and Moodle, famous commercial LMS are WebCT and Blackboard.

[13] The development of the open source LMS OLAT (Online Learning And Training) started at the University of Zurich in 1999 and won the Medida Prix for best eLearning software in 2000. Today OLAT is already in its fifth version and is the strategic platform of the University of Zurich. Besides Zurich other universities like Bern, Sachsen (Germany) etc. are using OLAT as their main LMS. More information and download of the software can be found on the OLAT website.



One GITTA lesson shown in different versions/designs using eLML layout templates

Newsletter

We offer four mailing lists to which you can subscribe (free):

- **Newsletter**: Get news about eLML. *Low traffic* (max. once a month) moderated news list. Just use the form below the menu to subscribe to this list!
- **Users** (Authors & developers): A *low traffic* non-moderated list about technical discussions between eLML users or developers. Only available to registered Sourceforge members. **Please subscribe to this list if you are working with eLML!** We announce and discuss new features, tests, hints etc. via this list.
- **CVS Messages**: A *medium traffic* mailing list where all CVS commit messages are automatically sent to. Postings on this list are not allowed. For developers.
- **Tracker Messages**: A *medium traffic* mailing list where all Tracker (Bug, Support, Feature Request) messages are automatically sent to. Postings on this list are not allowed. For developers.

Besides these two newsletters, Sourceforge offers other tools to which you may subscribe:

- **eLML News**: Published both on the main **www.eLML.org** and **Sourceforge Homepage**. Each article does have a comment button, in case you would like to clarify or ask something regarding the issue.
- **Monitor new file releases**: On the **eLML Sourceforge page** click under "Download", the "Monitor" newsletter icon to automatically stay informed about new eLML releases. Only available to registered Sourceforge members.
- **RSS feeds**: You are welcome to include some of our **RSS feeds** on your website!

How can I participate or contribute?

If you are looking for an interesting topic for your thesis or diploma work, consider one of the following projects:

New output formats:

1. IMS Common Cartridge: The new **IMS CC** format for course export/import into an LMS supports content, quizzes (QTI), tools (LTI) and discussion forums. This project would have to evaluate first on how the CC standard is supported and if new features like forum-support have to be introduced into eLML and then a XSLT-based Common Cartridge converter would have to be developed.
2. SCORM 2004: Many modern LMS's offer the possibility to import **SCORM 2004** content and therefore it would be important for the eLML to not only offer SCORM 1.2 but also the 2004 format as an output format. But since eLML is more focused on linear navigation and does not offer sequencing, it is imperative to first developed a concept about how SCORM 2004 is supported in eLML before developing a converter.
3. Slides version: Export an eLML lesson as slides. What format? E.g. **Open Office Impact** or HTML-based **S5** or a PDF version?
4. eLML Flash client/output: Either a transformation from XML to Flash (is there a XML format behind flash?) or a kind of Flash player to import eLML lessons transform them into flash files. These Flash files could then be distributed as standalone lessons and that run platform-independent.
5. PDFXML Converter: **Adobe's Mars project** (still experimental) offers the possibility to create PDFs based on XML without using FO and an FO parser. A transformation file to create PDFXML code out of an eLML lesson might be very useful.

New eLML tools development:

1. Metadata Editor: Although the **eLML structure** does include metadata, the **Firedocs eLML editor** does not offer a WYSIWYG GUI yet to enter eLML metadata. At the University of Zurich we have developed a webbased metadata editor based on **OLA**T for the **edulap** project. Maybe this tool could serve as a basis for creating an eLML metadata editor? Preferably the editor should be based on the Firefox XUL format and work together with the exiting **Firedocs eLML editor**.
2. Client to transform and manage eLML lessons with features like: Transformation of lessons into various formats, preview, versioning, project management, project navigation, import/export etc. Similar to the **Lenya Add-On** but as a standalone software. Preferably based on Firefox and with **Firedocs eLML editor** included? To be defined.
3. **OpenOffice plugin**: As a results of a student's thesis we do have a beta version of an eLML offline editor working with OpenOffice. The plugin is programmed in Java and supports most eLML elements plus an export feature. We are looking for someone to finalize the plugin and release it as an open source tool via the eLML project page.
4. Default layout/themes: Why not include some nice layout templates in eLML? If you have some flair for graphics and XSLT/CSS knowledge you could create some nice default templates for eLML (like **Kubrik** or **K2** in **Wordpress** etc.) either using the **Template Builder** or using XSLT.
5. EndNote Interface: People using EndNote should have an easy interface to use their bibliography with an eLML lesson.
6. Tool to create non-linear navigation (e.g. topic/concept maps) similar to e.g. ShowMeDo's **Learning Paths**.

GSoC 2008 is history!

The following eLML projects were successfully realized during summer 2008 and integrated into the eLML project:

- **Template Builder** by Thomas Linowsky.
- **DocBook converter** by Alberto Sanz.

Success Stories: Projects using eLML

The eLML website itself was also created using eLML

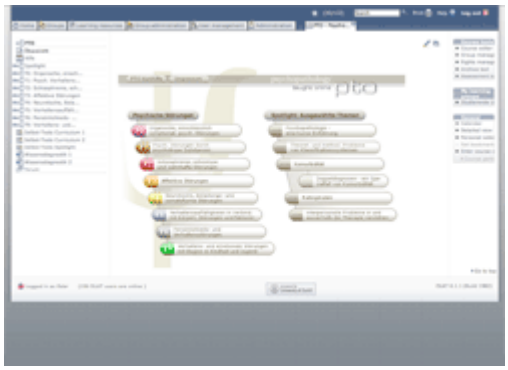
Who is using eLML? Since it's open source we can't really track who is using eLML. But we would like to list some projects that have been using eLML for the last years and that also helped improve eLML with valuable input or manpower. If your project is using eLML and you would like to be listed here feel free to **contact us**.

GITTA - Geographic Information Technology Training Alliance

GITTA Screenshot (click for large view)

GITTA is a Swiss eLearning project for Geographic Information Science and Technology. Nearly 50 lessons or case studies are available for free passive use. GITTA uses the XML based framework eLML, so there is no commercial or proprietary software needed for showing the contents.
Website: **www.gitta.info**
Contact: **Prof. Dr. Robert Weibel**
License: Non-Commercial use under the Creative Commons Licence (Attribution-NonCommercial-Share-Alike)

PTO - Psychopathology Taught Online



PTO Screenshot (click for large view)

PTO is an eLearning course which facilitates the understanding of mental disorders. It covers mainly the phenomenology of mental disorders, but also other important facets of psychopathology such as diagnostic challenges or information on rating scales. PTO is held in German and designed to be implemented in various blended learning scenarios, but can also be used as a stand-alone learning module. Additionally, PTO introduces an innovative adaptive learning system, which compares the individual cognitive learning structures of the students with expert structures, resulting in automated learning recommendations and exercises. PTO's first official release in summer term 2007 was used by 350 students at four different universities in Switzerland and Austria. Up to now, PTO is regularly used at the Universities of Zurich, Bern, Fribourg, Basel, Salzburg and the LMU Munich with a total number of more than 1300 users. PTO reached the finals and won the "Publikumspreis" of the **Medida-Prix 2007** as well as the "Springer E-Learning Award 2008".
Website: **www.pto.uzh.ch**
Contact: **Roland Streule**
License: Free access for all partner universities. For Commercial Licensing (individual users, institutions) please contact Roland Streule.

EyeTeach

EyeTeach Screenshot (click for large view)

EyeTeach is an elearning project by the eye clinic (**Augenklinik**) of the University Hospital Zurich. Students can learn about the basics of the eye, ophtalmic diseases and how to treat them. EyeTeach has been integrated into the OLAT, the learning management of the University of Zurich (UZH), and is therefore only accessible for UZH-students.
Website: **www.eyeteach.org**

Contact: **Anne Jansen**
License: Only accessible for University of Zurich students.

ALI Swahili

GLOPP Screenshot (click for large view)

ALI Swahili creates a teaching module in Swahili as part of the curriculum of African linguistics at the University of Zurich, using for this purpose a combination of multimedia CDROM and internet technology. By exploiting the didactic potential of a multimedia learning environment, ALI Swahili makes the acquisition of one of the world's pre-eminent non-European languages more effective, more flexible and, as a consequence, more attractive to students.
Website: **Only old version is online (not new eLML-version)!**
Contact: **Thomas Bearth**
License: Open for partners, ask Prof. Bearth for details.

GLOPP - Globalisation and Livelihood Options of People living in Poverty

GLOPP Screenshot (click for large view)

GLOPP is an interdisciplinary modular course consisting of 18 blocks (each 5-6 hours worktime), developed by UZH, UNIBE and EPFL. The course thematically focuses on poor people’s livelihoods and contributions to the improvement of their situation. Blocks are integrated into different seminars. Students work through text, illustrations, animations, (collaborative) exercises and self-tests to prepare better for the presence phases. GLOPP was completed at the end of 2007.
Website: **www.glopp.ch**
Contact: **Norman Backhaus**
License: Free access for partners (on LMS OLAT and Moodle). Open Content with some exceptions

GI - Geschprächsanalyse Interaktiv

GI Screenshot (click for large view)

Mit dem e-Learning-Projekt "gi - Gesprächsanalyse interaktiv" wird das Ziel verfolgt, eine innovative Vermittlungsform zu etablieren, die spezifisch auf die methodologischen Anforderungen der Gesprächsanalyse zugeschnitten ist. Nach dem Prinzip des kollaborativen forschenden Lernens werden den Studierenden in "gi" die wesentlichen Komponenten einer gesprächsanalytischen Untersuchung in Form angeleiteter Gruppenarbeit und internetbasierten Selbstlernens vermittelt.
Website: **OLAT Server**
Contact: **Katrin Lindemann**
License: Check **this page** for details!

GISMA

GISMA Screenshot (click for large view)

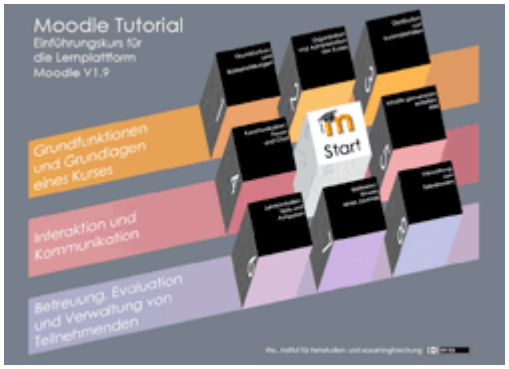
GISMA is a Faculty of Geography Marburg eLearning project to derive competence building in Geographic Information Science. 8+1 lessons are available for free use. It is the first complete Blended Learning resource to provide a specific Elearning environment to support bachelor of geography students. The didactic approach is based on the problem based learning concept.
GISMA uses the XML based framework eLML, so there is no commercial or proprietary software needed for showing the contents. additionally it integrates parts of GITTA courses merged with new developed material
Website: **gisbsc.www.gis-ma.org**
Contact: **Christoph Reudenbach**
License: Creative Commons Licence by-nc-sa 3.0

MESOSworld

MESOSworld Screenshot (click for large view)

MESOSworld is an elearning project developed by the universities of Berne, Fribourg and Zurich. Students in the first semesters get a theoretical overview over fundamental methodology in social sciences (statistics etc.). Apart from the theory MESOSworld contains also exercises, learning controls and case studies.
Website: **www.mesosworld.ch**
Contact: **Roland Streule**
License: Free access for all partner universities. For Commercial Licensing (individual users, institutions) please contact Roland Streule.

Moodle Tutorial



Moodle Tutorial Screenshot (click for large view)

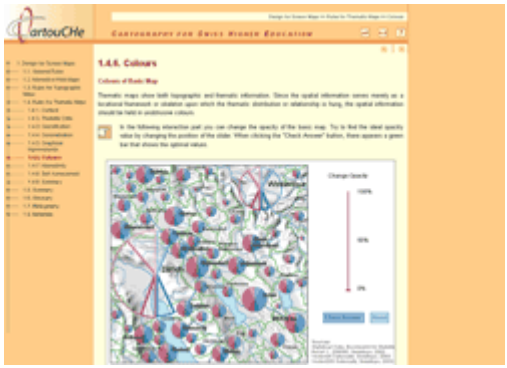
The Moodle Tutorial is an introduction into the Learning Management System Moodle created by the Institute for Research in Open-, Distance- and eLearning in Brig. The tutorial is a self-learning courses with video instructions to help beginners getting to know and using Moodle.
Website: **moodle.ifel.ch/tutorial**
Contact: **Roman von Wartburg**
License: Creative Commons «Attribution/Share Alike» 2.5 Switzerland

OLAT - Online Learning And Training

OLAT Screenshot (click for large view)

OLAT is an open source learning management system developed originally at the University of Zurich but now used throught the world and translated into 17 languages. Not only the OLAT website and all its manuals and documentation were created with eLML but also a lot of content on the main OLAT server itself was developed using eLML.
Website: **www.olat.org**
Contact: **Immo Wille**
License: Open Source (Apache 2 licence)

Cartouche - Cartography for Swiss Higher Education



Cartouche Screenshot (click for large view)

Cartouche is a new eLearning project and Internet course, imparting knowledge about the multimedia and Internet cartography to advanced students. The course consists of the 3 modules "Multimedia Cartography", "Location Based Services" and "3D Applications". The modules were developed by the participating universities ETH Zurich, University of Zurich and University of Applied Sciences Northwestern Switzerland. Each module consists of a fix number of lessons, each lesson has a maximum of 6 units (presenting 1 learning object). Since the course structure and even the pedagogical concept is modular, eLML as highly flexible and extensible technique was used to transform the concept to the web.
Website: **www.e-cartouche.ch**
Contact: **Samuel Wiesmann**
License: internal use, 2 modules will be published under the Creative Commons License in the next months

ELT@RWI - E-Learning Templates der Rechtswissenschatlichen Fakultät

ELT@RWI Screenshot (click for large view)

The factulty of law at the University of Zurich creates all of its e-learning content using eLML. Topics cover legal history, family law and (in German) Verfassungs- und Obligationenrecht.
Website: **www.elt.uzh.ch**
Contact: **Lukas Stähli**
License: Mostly internal (ask Lukas Stähli for details)

SOREL - Swiss ORL E-Learning

SOREL Screenshot (click for large view)

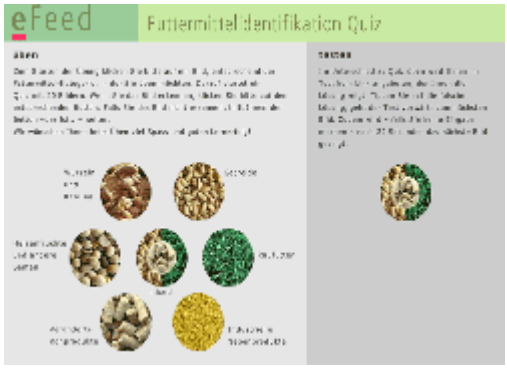
SOREL (Swiss ORL E-Learning) provides interuniversitary, countrywide unified e-Learning modules containing the full knowledge base in the field of Otorhinolaryngology needed by the undergraduate student to become a general practitioner. All modules will be available in german and french. SOREL has sucessfully started in 2008 and is continually enhancing its content. Parts of SOREL are already in productive use (module Ear). At a later stage, SOREL contents will also be examinable.
SOREL is a project of the department of Otorhinolaryngology, Head and Neck Surgery of the University Hospital Z¸rich in collaboration with all university clinics (ORL) in Switzerland.
Website: not public
Contact: **René Holzreuter**
License: internal use

FOIS - Foundations of Information Systems

FOIS Screenshot (click for large view)

FOIS provides twelve self-directed e-learning modules on an introductory level in the field of information systems. The modules cover topics like e-business and computer networks or specific applications and systems for enterprise resource planning, supply chain management or decision-support. Each module is self-contained and can be employed complementary in face-to-face classes according to the blended learning approach.
A given learning path and a clear structure, various examples and exercises, numerous interactions and self-assessments support the student's learning process. FOIS modules have been implemented since 2004 and, in the meantime, are used by more than 3000 students at several institutions in higher education in Switzerland and Germany.
Website: **www.fois.ch**
Contact: **Stefanie Hauske**
License: Free access for all project partners. The modules will be published under the Creative Commons License in the next months.

eFeed

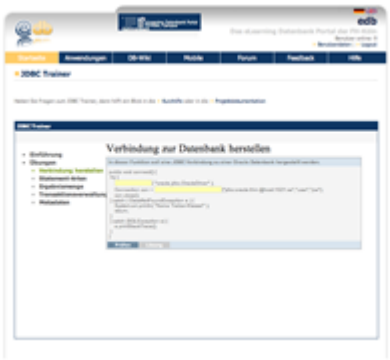


eFeed Screenshot (click for large view)

eFeed is an eLearning project that offers students from veterinary medicine and agricultural sciences flexible learning modules about animal feed nutrition. The learning process is supported by visualisation of feed stuff and exercises with a focus on enhancing problem solving skills. Self-tests are offered to evaluate the individual learning process. Furthermore the Swiss feed database (Leading House: Swiss Federal Institute of Technology Zurich) provides students with actual data of feed stuffs.
Website: **integrated in OLAT LMS**
Contact: **René Schegg**

License: Open Content for Universities and Universities of Applied Sciences

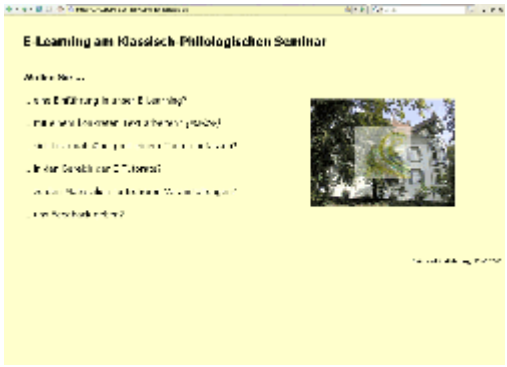
JDBC-Trainer



JDBC-Trainer Screenshot (click for large view)

The JDBC-Trainer is one of the training units developed at the University of Applied Sciences in Koeln, Germany. This module was created by a student using eLML.
Website: **JDBC-Trainer** (Registration necessary)
Contact: **Heide Faeskorn-Woyke**
License: ©FH-Köln (for details see Website)

Introduction to Classical Philology



Project Screenshot (click for large view)

The «Einführung in die Klassische Philologie» provides access to the most important subjects in Classical Philology, complementary to courses taught regularly at the University of Zurich. The course is held in German on three levels and offers modules such as «Textkritik und -edition», «Stilistik», or «Wissenschaftliche Arbeiten». At the moment, the modules originally designed for and with OLAT are being edited with eLML.
Website: **under construction**
Contact: **Dominique Stehli**
License: Internal use at present (University of Zurich)

Download eLML: Difference between the stable and the developer release

eLML is available in two versions:

1. **Stable Version**: A *ZIP* [14] file for immediate download with the latest stable eLML release.
2. **Developer Version**: Install eLML via *CVS* (for developers working with eLML)

We are using Eclipse in our examples, since this is an open source tool available for many platforms. Please **read the Eclipse instructions** if you have never worked with Eclipse before. If you are an author using (or planning to use) the elml.uzh.ch content server: Please read the **content server chapter** carefully and be sure to know the basics of *CVS*.
When you have finished installing eLML you can continue with **creating your first testlesson** and looking at our extensive **eLML documentation**. Dont miss the **output formats section** with instructions on how to transform your content into different formats and check the **tools** available for eLML. For bugfixes, questions, feature requests etc. please consult the **eLML support**.

[14] ZIP is a compression algorithm widely used on most of todays operating systems. A ZIP archive can contain multiple files, folders and subfolders. Both SCORM and IMS content packages are actually ZIP files that must contain a valid imsmanifest.xml file on their root level. With MacOS X you can right-mouse-click files or folders in the finder and choose "Create archive of ..." from the context menu. On Windows you have to install a tool like WinZIP to create ZIP archives.

Installing the eLML Stable Release

To get the latest stable eLML release as a *ZIP* archive (including the XML Schema, XSLT files, templates, manuals and demo lessons) click on the green "Download eLML" button on the main **eLML Sourceforge Site**. The binary release is meant for non-developers who would like to have a quick look at eLML and maybe use it for their project or for authors who want to work with a stable release of eLML without regular updates. For active eLML users and developers we recommend the **developer version**.
You have the possibility to automatically get an email when the eLML team releases a new stable binary version. While logged in into Sourceforge click the "Monitor" symbol in the list of releases and you'll be notified about updates.

Installing the eLML Stable Release in Eclipse
If you want to work with the software development tool Eclipse (**read instructions**) it's fairly easy to install eLML:

1. **Download eLML 8 Stable Release**
2. Unzip the ZIP file and put the "eLML_v8" folder somewhere on your harddisk. This is your eLML-folder, you are ready to create your first lesson with **Firedocs** and transform it with **EasyELML**!
3. Continue here only if you work with **Eclipse**! Startup Eclipse and choose the "eLML_v8" folder as your workspace.
4. Close the Eclipse "entry desktop" by clicking on the big arrow on the top right corner.
5. Now you have an empty workspace. To get your files choose "File->New->Project..." and a window opens.
6. Go to "General" and double-click "Project".
7. Enter the project name "core" and click "Finish".
8. Repeat "File->New->Project...->General->Project" and enter "elml" as project name.
9. Repeat "File->New->Project...->General->Project" and enter "gitta" as project name.
10. That's it. You have three projects now and all should have content in it. Continue with an example on **how to create an eLML lesson**.

After successfully having installed the **stable** release of eLML you should have at least two folders on your harddisk: The "core" folder and at least one project folder (e.g. the "gitta" or "elml" folder that are part of the stable version). For more information about the exact folder structure of eLML, check the **folder structure chapter** of the documentation.

Installing the eLML Developer Release

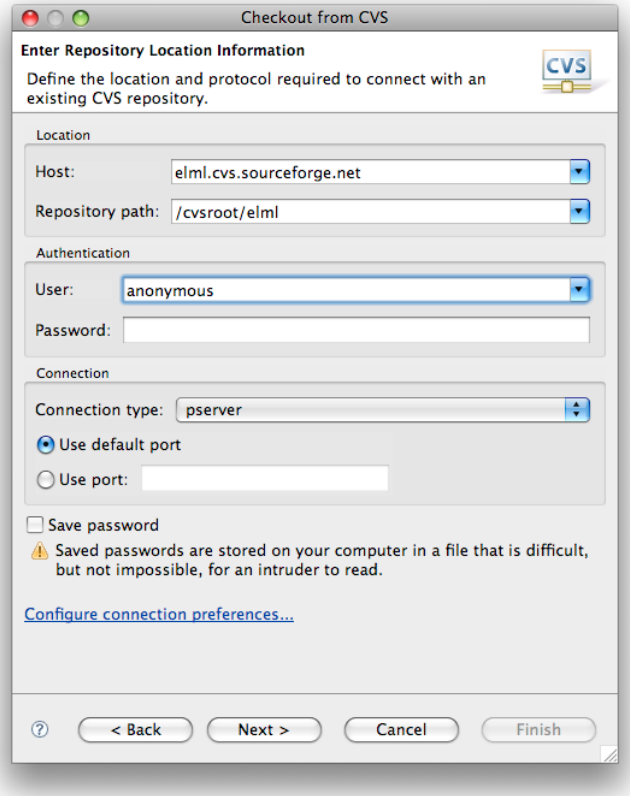
This manual will give you an introduction to using the eLML developer release from the *CVS* repository. Please note that this approach is meant for authors working with eLML and wanting to update the eLML core files regularly or commit updates themselves. For non-developers we recommend **downloading and installing the stable version**. Please do read the **Installing Eclipse** chapter first, if you are not familiar with the Eclipse tool. Of course you are free to use any other available CVS tool.

Connecting to Sourceforge and checking out the eLML core files

Before starting, you should open a free **Sourceforge Account** and ask to become an **eLML project member**. Otherwise you will not be able to do an authenticated checkout (but you can still use the anonymous "pserver").

1. In Eclipse choose `File:New:Project...` and the Eclipse "Project wizard" will open.
2. Double click on "CVS" and the option "Projects from CVS" will become available. Double click it to create a new CVS project.
3. Now enter the information as listed below:
   - Host: `elml.cvs.sourceforge.net`
   - Repository Path: `/cvsroot/elml`
   - Username: Your Username or `anonymous`
   - Password: Your Password or *empty*
   - Connection type: Use "extssh" for authenticated checkour or "pserver" for anonymous checkout

   If you plan to commit changes, please **contact the eLML administrator**. Your Sourceforge username will be added to the project and you will be able to do a checkout using "extssh" using your Username/Password as displayed on the screenshot below (95% of all users will not need this though).
4. Click "Next" and choose "Use an existing module" in the next window.
5. Eclipse will now connect to Sourceforge and show you a list of available modules to download. Choose and mark the "core" and the "elml" module by shift-clicking on it. Click "finish" and you're done.

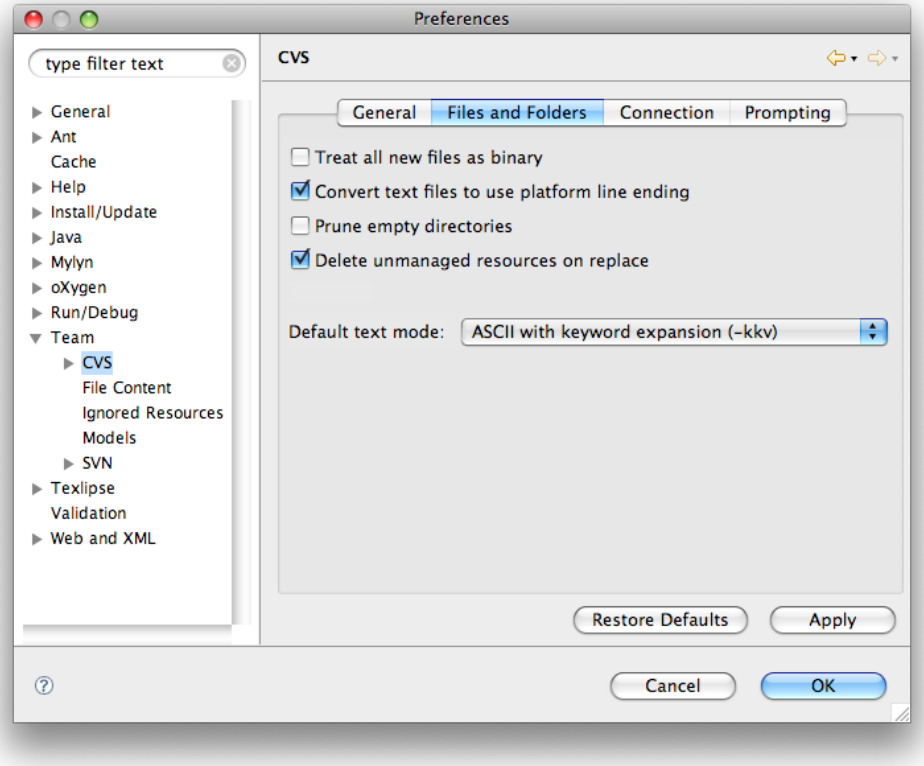


Eclipse screenshot: The CVS checkout window

Now you should have a new project called "core" and a second project called "elml" in your Eclipse workspace with the server name "elml.cvs.sourceforge.net" to the right of the project name. The "core" module contains all required files for working with eLML. The "elml" project folder contains the website (including the manual) elml.org. For more information about the exact folder structure of eLML, check the **folder structure chapter** of the documentation.

Empty Folders: By default CVS does not show empty folders. Therefore, if you e.g. checkout a new repository that is still empty, you will have to tell Eclipse to show empty folders. You can do this within the preferences under `Window:Preferences:Team:CVS:Files and Folders` where you should uncheck "Prune empty directories".

If you plan to work with the eLML content server, continue **reading the next chapter**. If you plan to create your own project, refer to our extensive **eLML documentation**



Eclipse screenshot: Uncheck 'Prune empty directories' in the preferences to see empty folders in CVS

Installing Eclipse and oXygen

Why Eclipse?

In this manual we will explain how developers can work with Eclipse and either oXygen or XMLSpy Editor used as Eclipse plugin. Eclipse is used for CVS operations and file management. Eclipse is available for Mac, Windows or Unix and therefore a good solution for a group of authors working together. You are free to use other CVS and project management software.

Since Eclipse is an open source project, you can get it for free from the **eclipse.org website**. **oXygen**, the XML editor of choice (for us), is a commercial software and is based on Java (like Eclipse) and therefore also available for Mac OSX, Windows, Linux and different Unix flavors. Additionally, it is also available as an integrated Eclipse plugin, which makes it easier for authors using eLML to have all their working files in one place and working with one single tool. The platform-independency of Eclipse and oXygen makes it for us the no. 1 choice when working with XML and CVS. But also **Altovas XMLSpy** is available as Eclipse plugin but only on Windows platforms. So here is a short tutorial on how to install the software on your computer:

1. **Download Eclipse** and install it on your harddisk. Choose the classic Eclipse release since it contains all the tools we need.
2. Start Eclipse. You will be asked to define a workspace where all your files will be stored. We recommend creating a new folder and naming it e.g. "eclipse_elml_workspace". Now lets install the oXygen plugin.
3. Open the Menu "Help" and go to "Install New Software". In the field "Work with:" enter the following URL **http://www.oxygenxml.com/InstData/Eclipse/site.xml** and hit the enter key. (If the above URL does not work, check the **oXygen Download page** for more info)
4. Eclipse will look for the available software now. In the result window select the check box for the oXygen XML Editor and press the "Next" button two times.
5. After accepting the license and clicking the "finish" button oXygen starts downloading and installing. When finished, you must do a restart of Eclipse. Now you can open an XML file and oXygen will be loaded into Eclipse. Please note that this is a commercial software so you will have to enter a (trial) license key when oXygen opens the first time.
6. If you would also like to install the XMLSpy plugin, please go to **Altovas Eclipse page** and download the Installer. Follow the instructions of the installer.

The elml.uzh.ch content repository server for storing lessons

Connecting to elml.uzh.ch and checking out the eLML project files

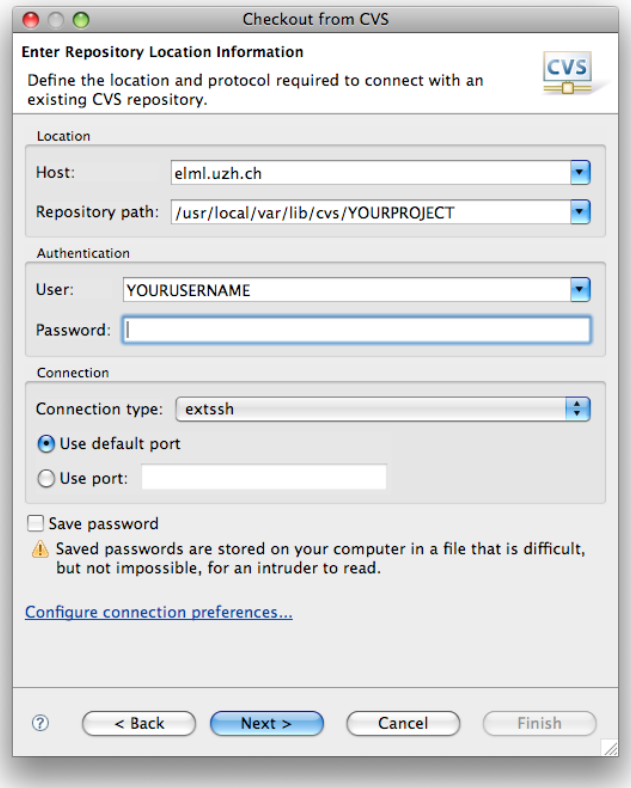
> This chapter is only meant for authors whose projects already reside on the elml.uzh.ch server or who plan to put it on this server (maintained and administered by the **University of Zurich**). Please note: Access to this server is limited to UZH-members but if you want you can download the **make4eLML serverscripts** and create a similar CVS-repository server yourself!

What is the advantage of using the elml.uzh.ch content repository?

- Server hosted and maintained by the IT Services UZH
- Backup nightly by the IT Services UZH
- Professional versioning software (CVS)
- The whole authors team can write and update lessons
- The **make4eLML** serverscript regularly checks if a projects lesson are valid (and sends a mail if not)
- Possibility to transform a lesson automatically into the online, PDF, ODF and LaTeX format plus IMS and SCORM ZIPs with every commit (saved modficiation of a lesson) and upload the results to your server or choice (see **make4eLML** for more information)

If you choose to use our content repository here's how: Besides the "core" and "elml" project within your Eclipse workspace you will need a new project folder containing all your project specific stuff (content, layout templates, config files etc.). It will be named after your project and can be stored and

checked out from the *CVS* server elml.uzh.ch. Contact the administrator of the eLML content server (via **elml@id.uzh.ch**) for a new repository (YOURPROJECT) an account and access to the server (YOURUSER-



NAME).

Eclipse: CVS checkout from content server using extssh.

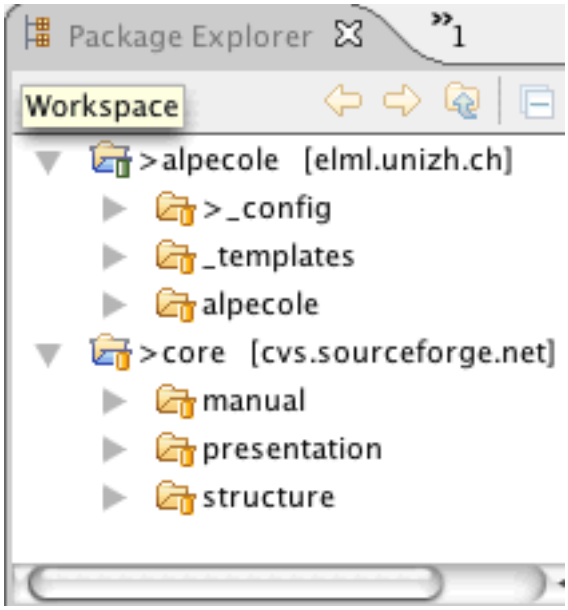
You will have to send the administrator your public ssh key (What is a key? **Tell me more!**) and he will send you back a username. Dont start below before you haven't got your username!
If you got your projects cvs repository (YOURPROJECT) and your username (YOURUSERNAME) for the content repository server, here is how you can checkout e.g. the GITTA project folder:

1. First you need an empty project folder named after your project: In Eclipse choose `File:New:Project...` and the Eclipse "Project wizard" will open. Here choose General:Project and click "Next". You can enter the name of your project (gitta, cartouche, pto, efeed etc.) in the next window and click "Finish".
2. Now again choose `File:New:Project...` and the Eclipse "Project wizard" will open again. This time open or double click "CVS" and choose the option "Projects from CVS" that will become available. Double click it to create a new CVS project.
3. Eclipse will show you a list of stored repositories to choose from (e.g. the Sourceforge repository you just used to checkout the core, if you are using the **developer release**!). Since you are using the content repository server for the first time you will have to choose the "Create a new repository location" radio button to create a new checkout of a repository.
4. Now enter the information as listed below:
   Host: `elml.uzh.ch`
   Repository Path: `/usr/local/var/lib/cvs/YOURPROJECT`
   Username: *YOURUSERNAME* (as told you by the administrator)

Password: *leave empty* (you will use key login - see below)
Connection type: Use "extssh"

5. If this is the first time you access this server you have to accept the server "fingerprint" now. Afterwards Eclipse will ask you for the password of your ssh-key. If you generated a key with a password enter it now. Else just click "Next".
6. In the next window you can click the second radio button to view a list of modules you can checkout*. Choose the folder "_config" and "_templates" (without them eLML won't work) AS WELL AS the lessons you want to check out using command-click. Choose "Next" (not "Finish") because you are not finished yet :-)
7. In the next window choose the last option "Check out into an existing project" and click "Next".
8. Now you will see the project folder you've just created (named after your project) as target folder in the following window. Choose it and now you can click "Finished" because you're done :-)

* What if the list does not appear? Restart Eclipse. Still nothing? In this case Eclipse didn't recognize your ssh-key. Please make sure that within the Eclipse settings the correct path AND name of your private key is set and maybe restart Eclipse. **Read more...**



Eclipse screenshot: Example of 'package explorer' view

Now you can start working with eLML as described in the **eLML user manual**. You Eclipse workspace should now look like the picture to your right (instead of "Alpecole" there will be a folder with your project name). Please read the paragraph about **displaying empty folders in Eclipse** if you installed your content repository and nothing is displayed.

How can I add new lessons to the repository?
WARNING: This is a tricky one and if anyone knows a simpler method to include a new lesson, please let us know! The main problem is Eclipse only allows to add new modules (=eLML lessons) into a CVS repository if they are a single Eclipse project. But since we are not using one project per lesson (a lesson is in cvs terms called "module"), we cannot directly add new lessons to the repository and have to do a workaround as described below:

1. First you need an empty project folder named after your lesson: In Eclipse choose `File:New:Project...` and the Eclipse "Project wizard" will open. Here choose General:Project and click "Next". You can enter the label name of your lesson in the next window and click "Finish".
2. Now you can create the subfolders (e.g. "de" or "en" and within this folder "text" and "image" etc.) and create a new XML file within the "text" folder. You can do this by doing a right-mouse-click on the "text" folder and by choosing `File:New:XML File`. Please remember to name the XML file like this: `lessonlabel.xml` where *lessonlabel* stands for the actual label of your lesson.
3. The XML file must start with the "lesson" root element and contain the schema reference, the lessonlabel and title attribute etc. as described in the **create a new lesson** chapter.
4. Now you can start creating the lesson with entry, goals, unit etc. elements and validate it as described in the **validation chapter**.
5. Once you're finished you can make a right-mouse-click on the project/lesson and choose: `Team:Share Project...`
6. In the following dialog choose the CVS repository where the lesson belongs to and click next. The following steps are pretty similar to the steps described above (always click the default button and you're OK :-)
7. We have now submitted the new lesson but within your workspace the lesson is at the wrong place. Therefore you have to do a new checkout from within your project folder. Make a right-mouse-click on your actual project folder (not your lesson you've just created) and choose `Import...`
8. Now you can choose your lesson and do a checkout into your project folder (refer to point 6 and 7 in the list above!).
9. After you have successfully included your lesson within your project you can delete the standalone lesson project folder we created in step 1. The lesson is now included into your CVS repository as a unique module.

If you're a command line geek you can add a new lesson to the repository with the command:
cvs -d :ext:*YOURUSERNAME*@elml.uzh.ch:/usr/local/var/lib/cvs/*YOURPROJECT* import -m "Some comment" *lessonlabel* vendor tag start

Appendix: Key generation in Eclipse

Please note that our content server elml.uzh.ch only accepts key authentication. **If you plan to use the eLML content server read this chapter carefully**. You can skip it if you are already familiar with key authentication.

I don't know about keys and authentication, tell me more about it!

For authentication purposes usually a password is used. This is not very secure since passwords can be hacked while transmitted from your computer to the server. Therefore using keys is a more secure solution. The way it works is that you generate a key pair with a public and a private key that match each other. The private key always (!) remains on your computer. The public key is distributed to the servers where you would like to have access. As soon as you login on that server using your username, the server knows the public key that belongs to that username. The server then sends the public key to your computer to check if it matches your private key. If they match, you get access. This way no passwords or sensitive data is transmitted since the private key always remains on your computer (and the private key should also be password protected itself!).

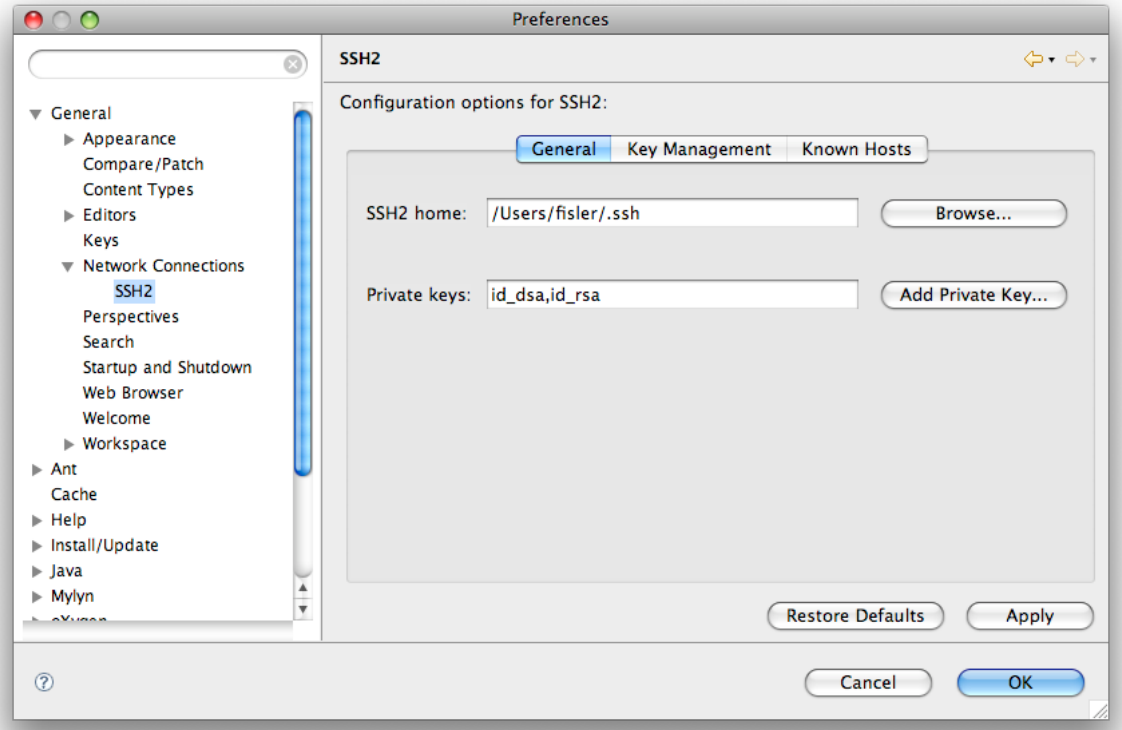
Generating a key in Eclipse

1. Choose "Preferences..." from the "Eclipse" (or "Windows", depending on OS) Menu. Now look for "General:Network Connections:SSH2". By clicking on the "Key Management" Tab you are able to create and manage keys.
2. Click on "Generate DSA Key" and a new DSA key (RSA is an older method not recommended using) is generated. Enter a useful comment to remember what you created this key for (E.g. "eLML Content Server for YOURPROJECTNAME" or something similar) and enter a password for this key.
3. Click "Save Private Key" to save your public and private key. Usually this is done in the ".ssh" directory of your home directory and usually it is named "id_dsa" but both can be chosen as wished. Just be sure that your application (here Eclipse) knows where to look for the keys (see below). Give a useful name to your key and after saving Eclipse generated both a public (with the ".pub" extension) and a private key. **Set your permissions correctly** so that no one but you has privileges to read your private key!!! This is usually done using the command `chmod 700 YOURKEY` within the .ssh directory.
4. Click "Apply" and "OK" and you're done with this step.
5. Please note: If you need to email someone your public key and your .ssh directory is hidden (e.g. on OSX or Linux/Unix but NOT on Windows), you cannot directly attach the public key to your email from within the mail client. To do so please issue the following command with a terminal application: `cp ~/.ssh/id_dsa.pub ~/id_dsa.pub` (now your public key named "id_dsa.pub" should be visible within your home directory).

The next step can be omitted if you've just installed Eclipse. The default settings should then still apply but feel free to check it:

Telling Eclipse where to look for keys:

1. Choose "Preferences..." from the "Windows" Menu. Now look for "General:Network Connections:SSH2". Go to the "General" Tab if you're not already there.
2. Define here where your ".ssh" directory is (normally this is "~/.ssh") and the names of your keys (usually "id_dsa" for DSA keys). Please note that in the "Private keys" field you can enter as many keys as you like, separated by commas. If they are all in the SSH directory defined right above, entering just the name is fine, else enter the whole path (or just use the "Browse" button and Eclipse does it for you :-)
3. Click "Apply" and "OK" and you're done with this step.



Eclipse screenshot: Key management in Eclipse.

Using your key to authenticate at Sourceforge

Sourceforge offers both password and key authentication. If you generated your key as described above and prefer to work with keys instead of entering your password, you have to upload your public key to Sourceforge. Within your **Account Options** on the Sourceforge website you will find a link to add and edit SSH Keys for your account. Please refer to the **Sourceforge Documentation** for more information.

How can I create a new eLesson with eLML and use it?

This section aims at providing a rudimentary guide for the creation of an eLesson with the eLML structure. It is both a technical guide for creating a new project and shows how an eLesson may be structured from the content point of view.

The technical stuff: How to create a new lesson

Please note that if you downloaded the **stable version**, the MyProject-files are already part of the package. This means that you have to do step 1 to 3 only if you work with the **developer release**.

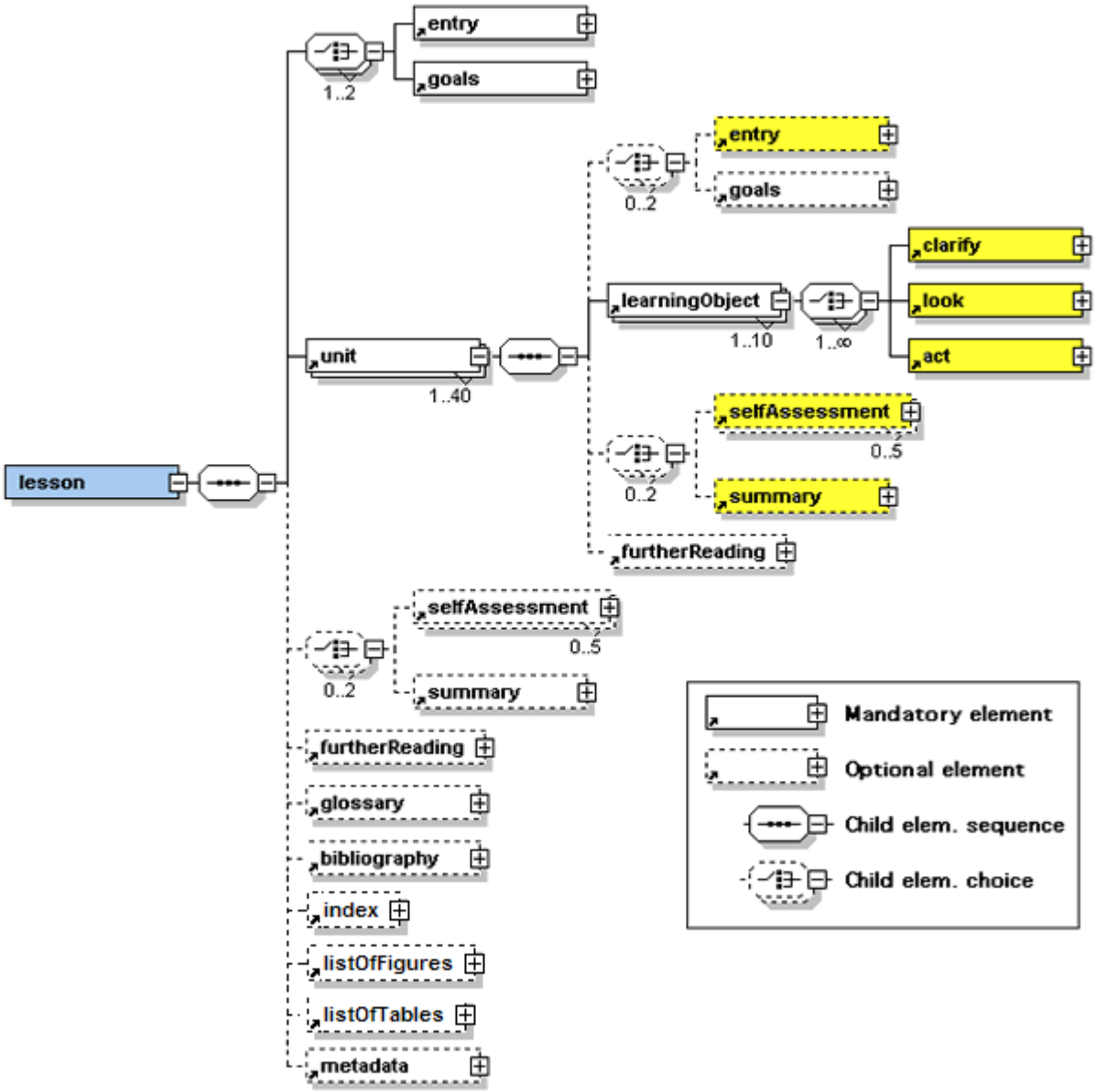
1. In your Eclipse workspace (where the "core" folder resides) create your project folder named "MyProject".
2. **Download the MyProject.zip file** (*ZIP* file that includes the basic folder structure for a new project) and unzip it.
3. Import (choose from context menu in Eclipse) or drag&drop the three folders "_config", "_templates" and "MyLesson" within your project "MyProject".
4. Rename the folder "MyLesson" and the file "MyLesson/en/text/MyLesson.xml" using your lesson label. Please note that the lesson folder should be named the same as your lessons label attribute (lowercase, less than 11 digits, no special signs etc.). (To rename use the context menu while clicking on the folder or file)
5. Rename the language folder "en" to "de", if your lesson is in German, to "fr" if your lesson is in French etc. Use the **ISO 639 two-digit (Alpha-2) standard** to define your language.
6. Open the "_config" folder and adapt the two files according to your needs. The **config.xml** is used to define your transformation (XSLT) parameters and the **validate.xsd** is needed for customization of the eLML XML Schema. You have to add the lesson label you created in step 4 to your validate.xsd file to the "PredefinedLabelsType" element, else your new lesson wont be valid.
7. If you want to create multiple lessons, we suggest to duplicate (many times, if needed) the "MyLesson" folder and rename it.

If you prefer to create your own XML file, make sure that the lesson root element contains the following attributes:
<lesson label="lessonlabel" title="My Lesson Title" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.elml.ch ../../../_config/validate.xsd" xmlns="http://www.elml.ch">
From this point on your XML editor should show you interactively the available eLML elements and you should be able to transform your lessons as shown in the **transformation examples**.

The content: First thoughts about an eLML eLesson

Assuming that you already have a topic about which you want to create a lesson and you know what your students shall be able to learn (goals/objectives), you should first think about the amount of material covered by this topic. It is recommended that an eLesson represent in size about 1 to 2 face-to-face lectures. The student should need at most 2 hours to work through the material provided. Therefore, you will have to decide if your topic is about right in size, if you need to extend it or if it might be sensible to split it into two or more separate lessons (each of them, for example, highlighting one specific aspect of the topic).

Then, a lesson can be substructured into up to 40 units which should all be about the same size. Additionally, you'll have to think about the structure of the unit. A unit is structured according to the ECLASS scheme and can consist of up to 10 learning objects. A single learning object deals with one small aspect of the topic in a threefold way. Ideally, it teaches this aspect with a clarify, a look and an act part. It is up to you in which sequence you use these parts and to decide if it is necessary to use one part twice.
Another important part of this first phase of creating an eLML eLesson is to think about short and meaningful titles that each lesson, unit and learning object should have.
What is described above is not always that easy in practice. Have a look at the following figure which shows the structure of a lesson. You may want to create a simple text file which has several subheadings representing the different parts of lesson. In this text file you can write your first thoughts and notes about the lesson and units and outline what (more or less) the units should contain.



A detailed view of the eLML structure

To give an actual example, let us assume we want to create a lesson which gives an overview about the Virtual Reality Modelling Language **VRML**. In about two hours time we might be able to give a short introduction to VRML and show and maybe explain a simple VRML scene. The text file could be structured as follows (please note that the content written here are first thoughts and might need revision at a later stage):

Lessontitle:	**First Glance At VRML**
Entry:	why was this lesson created, what will it teach
Goals:	- students know what VRML is and where it originates from - students have looked at a simple application of VRML and understand it
Unit1 – Title:	Overview and Background of VRML
Entry:	introduction to this unit
Goals:	- students know… (do not repeat lesson goals but refine them at unit level or skip them)
LearningObject:	Title: What is VRML?
Clarify:	explain what it is, where it can be used, advantages, disadvantages (Remark: This is all combined in one clarify because this is only an overview lesson, we could also create a more detailed lesson called 'What is VRML?' and then have these topics as separate learningObjects or even separate units.)
Act/Look:	install VRML Viewer, look at different examples of VRML use (Remark: This example shows that it is not always easy to distinguish exactly between the three

	parts clarify, look and act. Here, the parts act and look are combined in one part (later called either act or look). This is a possible and valid solution.)
LearningObject:	Title: Where does VRML originate from?
Look:	A figure showing the history and development process of VRML
Clarify:	Further explanations not included in the figure
SelfAssessment:	Some questions about the content of the unit, students can optionally share their findings in a discussion board
Summary:	summarise
Unit2 – Title:	Simple Application of VRML
...	
SelfAssessment:	Flash quiz with questions about this lesson
Summary:	Summarise and conclude the lesson, eventually give some outlook for further use of VRML
Glossary:	Make notes of all terms in this lesson that will need to be explained in the glossary.
Bibliography:	Here one writes whatever resources one used to create the lesson. Try to do it while creating the lesson, so you won't have to think about it at the end.

Example of the structure of a lesson about VRML

This might, hopefully, give you some idea how you could start structuring and creating an eLML eLesson.

Tips and Tricks for creating good eLessons

Some of the following Tips and Tricks apply to eLearning in general while others are specific to the eLML structure. More information about eLearning in general can be found on various websites and in books. Some of them are listed in the reference section of this documentation.

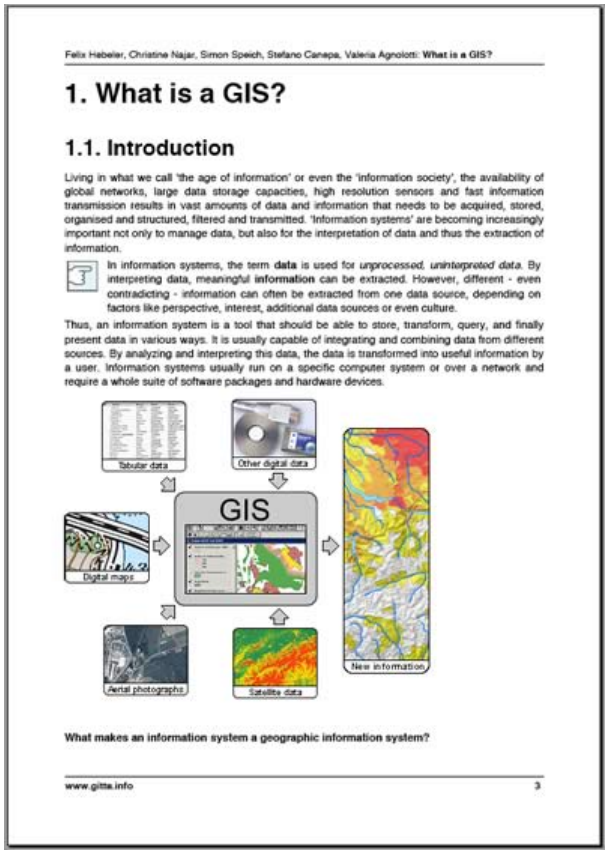
- Structure your lesson quite in detail before you start writing actual content using the XML structure. The structure above might even contain ready-made sequences of text and graphics before you start writing in XML. This is recommended as it is easy to loose the overview and good structure of the lesson when starting to dig into details too early.
- Having short and meaningful titles for lessons, units and learning objects helps you and the student.
- Try not to copy a textbook. Think about look, act and self assessment parts that enable the student to interact with the material and to learn more deeply. This part is very important in eLearning and should also allow the students to collaborate, share their work and build a community of learners.

- Look at the XML structure as a help and guide and not as a straitjacket. The structure should help you create good eLearning material and not hinder you. With some experience you will figure out the structure is much less constraining than you first thought. Actually, it is possible to develop material for many different learning scenarios within this structure. Let your imagination run wild.
- Write short, clear and meaningful sentences and paragraphs. The students learn the material on the screen and have no direct means to ask if they cannot understand the content.
- Use the explanation of XML elements in **Content elements chapter** to understand what each element means, how it is used and how it is displayed in the standard layout.
- Create detailed metadata for your lesson. This helps you when looking later at the same lesson and gives tutors the needed information when assembling courses from different lessons.
- The bibliography section allows you to list all the resources you have used to create the lesson. Keep the further reading section short and list only the resources that are and especially recommended for reading by the students.

Presenting the final lesson to your students

The simplest way to use lessons generated with eLML is to transform them into a set of standalone XHTML files. Using the **Template Builder** and the included XSLT files you can generate both one and multiple output XHTML pages or a PDF version (or other **output formats**) of your lesson. In the latter interactive elements like Flash animations are not visible. These files can be stored on a regular web server or put an a CD and distributed to your students. An individual layout can be created as **described in this manual**. An example of a standalone GITTA lesson is shown below:

Screenshot of a GITTA lesson as a standalone XHTML page.



Screenshot of the same GITTA lesson as PDF document.

You can also integrate your lessons within a *learning management system (LMS)* like OLAT, Moodle or WebCT. You will have to create either an *IMS Content Package* or *SCORM module* as described in the **IMS/ SCORM chapter**.

Documentation: Background information about eLML

The documentation of the eLesson Markup Language provides the explanations needed to work with the XML framework *eLML* and the associated files and structures. The documentation is constantly updated (see the footer for the last modified date) and reflects the status of the developer release. The stable release always contains a manual for the corresponding version within the *ZIP* file. You will need at least the following steps to use eLML:

1. **Download eLML**
2. **Create a new eLML lesson**
3. **Validate an eLML lesson**
4. **Transform an eLML lesson**
5. **Create IMS or SCORM packages**

A full overview about all the topics covered within this documentation:
This documentation **does not** provide information about:

- History/Background of eLML: Read the **about section** or visit the *GITTA* website.
- XML, XML Schema, XSLT etc. introduction: Read the explanation in the **glossary** and visit the links published there for more information.
- Pedagogical information/Didactics: This is a technical manual. Read the **pedagogical concept section** for more information.

Navigating on the eLML website using shortcut keys
If you choose to use the **HTML5 transformation**, eLML offers built-in support for quickly accessing parts of your lessons using shortcuts on your keyboard. Currently the following keys are supported in all eLML lessons (of course only if the according feature like glossary, index etc. is used in the lesson):

- Arrow left/right: Navigate to next or previous page
- 1..9: Jump to unit 1 to 9
- Ctrl + h: Home (first page of lesson)
- Ctrl + g: Glossary
- Ctrl + i: Index
- Ctrl + b: Bibliography
- Ctrl + c: Contact

The structure of the eLML XML Schema

eLML uses W3C XML Schema to define all the elements and attributes that are allowed within an eLML eLearning lesson. To some extent the content of elements and attributes is also defined (e.g. enumerations). eLML consists of three XML "core" schemas named elml.xsd, biblio_harvard.xsd and metadata_elml.xsd completed with the project-specific validate.xsd (the latter is not in the structure folder but in the _config folder of your project - see chapter **about the folder structure**). Referencing the validate.xsd from a XML document allows for validation of the document against the eLML structure defined in the XML Schema.

> Please note that the exact definition of the eLML Schema structure is found the "manual" folder within the eLML "core" folder. This extensive automatically generated reference file is updated with every eLML version and always reflects the current eLML Schema definitions.

The elml.xsd file

The elml.xsd file is the heart of the eLML structure. All the main elements are defined within this XML Schema. Some explanations about this schema can be found in the schema itself when opened in a simple text editor or with an XML editor. In order to be part of the "eLML community" it is absolutely imperative that everyone is using the same XML Schema. XML documents that can be validated against the standard defined in the eLML XML Schema will be able to share XSLT files, scripts and tools developed for eLML. To validate an eLML lesson XML file use the validate.xsd file described below (it includes all the following files).

The biblio_harvard.xsd file

The elml.xsd file includes the biblio_harvard.xsd file. In the latter all the bibliography elements are defined according to the Harvard standard. The creation of this XML Schema was roughly guided by the referencing guide of the Bournemouth University (SCILS 2004). If you want to use another referencing style you will have to create a new XML Schema defining the needed elements. Call this new XML Schema "biblio_yourstylename.xsd" and save into the structure folder. To replace the biblio_harvard.xsl file you will need to change the following line in the validate.xsd

<xs:include schemaLocation="biblio_harvard.xsd"/>

with

<xs:include schemaLocation="biblio_yourstylename.xsd"/>.

Note that then you will also need to change the layout files to include the new bibliography elements. Besides the Harvard style eLML also offers transformation files for the APA bibliographic style (American Psychological Association 2007). Please note for both XSLT files (APA and Harvard) the same biblio_harvard.xsd schema file is used (both standards use more or less the same elements).

The metadata_elml.xsd file

The elml.xsd file includes the metadata_elml.xsd file. In the latter all the metadata elements are defined. Those metadata elements were adapted from GITTA and do not directly comply with any official metadata standard such as LOM (**Learning Object Metadata by IMS**). But the eLML XSLT file contains a template to create an IMS Metadata compatible LOM file! This is actually required for the IMS Content Package manifest file. If you want to use another set of metadata (e.g. an official metadata standard) then you will have to take the XSD file of the selected metadata standard and include it in the validate.xsd file replacing the current reference

to the metadata_elml.xsd (see below). Save the XSD file of the selected metadata standard into the structure folder or in a subfolder (to be created) of the structure folder. To replace the metadata_elml.xsl file you will need to change the following line in the validate.xsd
<xs:include schemaLocation="metadata_elml.xsd"/>
with
<xs:include schemaLocation="pathIfNeeded/filenameOfYourStandard.xsd"/>.
Note that then you will also need to change the layout files to include the new metadata elements.

The validate.xsd file
Even though it is recommended that all use the unchanged elml.xsd file there are some possibilities to adapt the structure to the needs of your project. Most of these possibilities are united in the validate.xsd file (located in the_config folder of your project not in the core structure folder). Like the name says this is a configuration file in which some changes for the needs of your own project can be made. The parameters that can be changed are:

- the levels of the lessons
- the module names (shortcuts)
- the icons used for the layout and their format
- the institutions and departments
- the lesson labels
- the language shortcuts available
- the copyright statement
- etc.

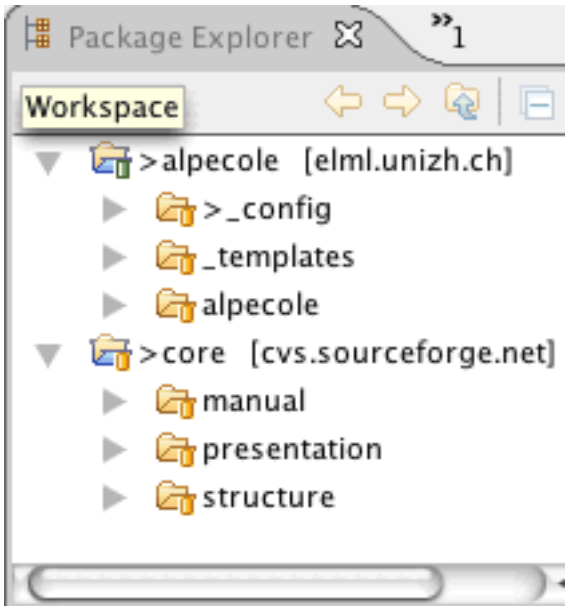
We used to have fix values like "_blank" or "_top" also for the "target" attribute of the **link element**. But this was not very convenient because users want to name the target (window) according to their needs. Therefore since eLML 5 we changed this and now the value entered is not checked. If you still want to have just a list of values allowed for the "target" attribute you need to replace the following line 8 in the validate.xsd:
<xs:include schemaLocation="../../core/structure/elml.xsd"/>
with this statement (basically replace a xs:include with a xs:redefine):
<xs:redefine schemaLocation="../../core/structure/elml.xsd">
<xs:simpleType name="targetType">
<xs:restriction base="elml:targetType">
<xs:enumeration value="_blank"/>
<xs:enumeration value="_top"/>
<xs:enumeration value="yourtargetname"/>
</xs:restriction>
</xs:simpleType>
</xs:redefine>
Please note that in XHTML 1.1 Strict the target-attribute is not allowed and thus if you choose to **generate XHTML 1.1 Strict code** you should not use the target-attribute of the link element because eLML will simply ignore it.

The config.xsd file

The config.xsd file defines the structure of the "config.xml" configuration file which lies in the _config folder of your project. eLML uses this file while transforming lessons into different formats. You can find detailed information about **configuring the eLML output transformation here**. An exact schema reference for the config.xsd file can also be found in the core/manual folder.

The eLML folder structure

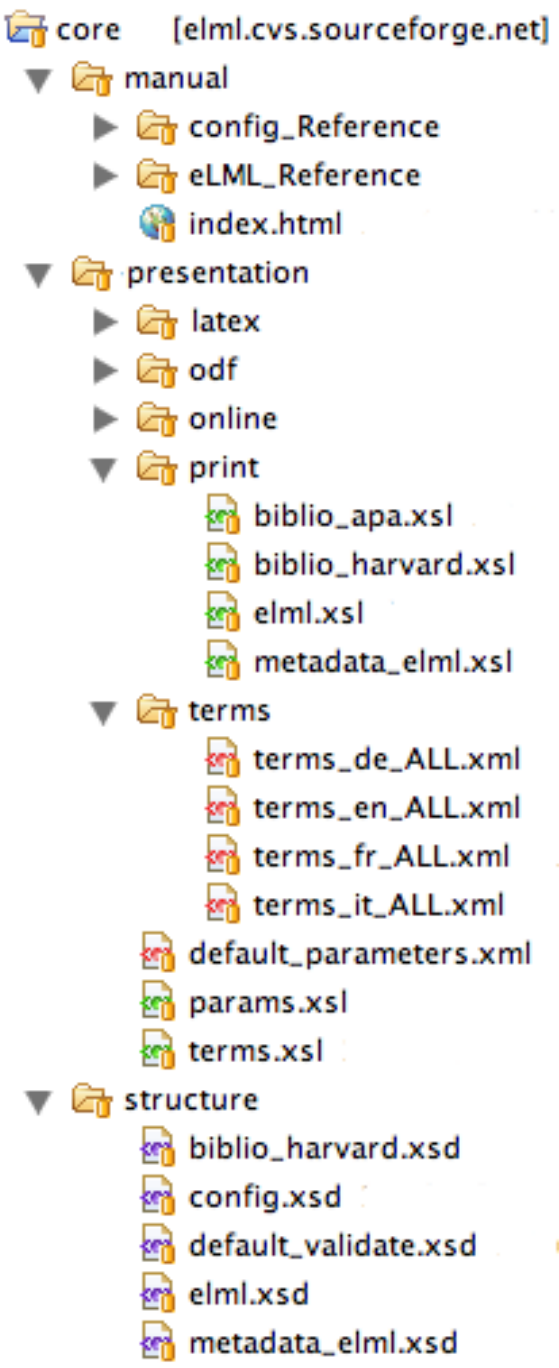
The core folder structure



Eclipse 'package explorer' view

Within eLML you always have a "core" folder, which contains general XML Schema and XSLT transformation files used by all project members, and one or more project folders (on the same hierarchical level) that contain your lessons (XML file) and project specific configuration and layout template data. If your project is called "Alpecole", the basic structure will look like shown in the screenshot to the left. You can have more than one project folder but you will always have only one "core" folder.
Within the eLML "core" folder there are three folders with the following subfolders:

- `structure`: The eLML XML Schema XSD files, the “heart” of eLML! See the **last chapter**.
- `presentation`: Here you will find the XSL files to generate different output formats of your lesson (see the separate **output formats chapter** for examples and tutorials). A technical overview about the files within the "presentation" folder can be found below.
- `manual`: An automatically generated reference for all eLML elements and attributes available both for the eLML and the config schema.
- `tools`: Contains binary and/or sources of all the tools available from eLML. See the **tools chapter**.



The core folder structure

Now let's have a closer look at how the "presentation" folder is built up:

- `online`: XSLT files needed to convert your XML file to XHTML and create content packages for *LMS* import. **Read more...**
- `print`: XSLT files needed to convert XML to XSL Formatting Objects to create a print PDF version. **Read more...**

- `epub`: XSLT files needed to convert XML to eBooks in the *ePub format* [15]. **Read more...**
- `latex`: XSLT files needed to convert XML to *LaTeX* [16]. **Read more...**
- `odf`: XSLT files needed to convert XML to *ODF* [17]. **Read more...**
- `docbook`: XSLT files needed to convert XML to *DocBooks* [18]. **Read more...**
- `default_parameters.xml`: This file is used if you don't have a **configuration file** defined in your project.
- `params.xsl`: Defines parameters used for the transformations.
- `terms` folder: Contains the translation XML files for general terms in German (de), English (en), French (fr) and Italian (it). Feel free to contribute more translations to the eLML project :-)
- `terms.xsl`: This file uses the folder "terms" and is responsible for creating the term keys in the according language (see below).

The `terms.xsl` file is responsible for creating the term keys in the according language. If a key is defined in a projects **configuration file** then it takes its value from there. This way each project can define how terms like bibliography or further reading are translated. Please refer to the config file of the "elml" or "gitta" project for examples. If no personalization is used the terms are taken from the according XML file within the core/presentation/terms folder. Currently we provide translations for German, English, French and Italian. If a certain language is not available the processor automatically uses the English term as "fallback". The actual language of translation has not to be defined, eLML knows it from the folder structure/folder name.

The project(s) folder structure

Now that you know how the "core" folder is built up, you will have to know how to create your own project folder. In the screenshot above you already saw the "alpecole" project. Typically your project folder has short, lowercase name and resides on the same level as the "core" folder. It contains at least the following three subfolders:

1. `_config`: Contains the following two configuration files:

---

[15] ePub is an XML based format for electronic books or eBooks. This format is used e.g. bei Apples's iPad. The standards definition can be found on openebook.org. More information about creating eBooks with eLML can be found here.

[16] This is how Wikipedia defines LaTeX: LaTeX is a document markup language and document preparation system for the TeX typesetting program. It is widely used by mathematicians, scientists, and scholars in academia and the commercial world, and by others as a primary or intermediate format (e.g. translating DocBook and other XML-based formats to PDF) because of the quality of typesetting achievable by TeX. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies. See the eLML to LaTeX chapter for more information.

[17] The Open Document Format (ODF) is an open source standard for office documents (text, spreadsheets, presentations etc.). It is used e.g. by OpenOffice or StarOffice and other similar open source tools. Since Microsoft Office 2007 started its own (only partially open...) XML format called "OpenXML" there is a "war of formats" going on because each standard tries to become THE standard for office documents. eLML offers a ODF converter because the team thinks that ODF is the better standard and deserves support from the open source community. Read this comparison to know more about the differences between ODF and OpenXML. There are many tools around to convert one format into another one and good office tool even support both by default so don't worry :-)

[18] DocBook is a semantic markup language for technical documentation. It was originally intended for writing technical documents related to computer hardware and software but it can be used for any other sort of documentation.

- `_config/config.xml`: The **configuration file** allows you to define the pagebreak level, the contact email address, the home URL and much more. You can also define your own translations for general terms (see above). It is used automatically when transforming XML files.
- `_config/validate.xsd`: The **validate.xsd** file should also be opened and adapted to your own needs. It is used as the main file for validating your lesson!

2. `_templates`: As the name suggests, it contains the layout templates for your project. There should be at least one template preferably with the same name as your project folder. If you use the **Template Builder** to create an eLML layout template, then you should choose this folder to store your projects and files.
   - `_templates/`*`yourtemplatename`*`/online.xsl`: Here you can define your own templates to create a unique layout. **Read more...**
   - `_templates/`*`yourtemplatename`*`/elml.css` and `elml_print.css`: These CSS files are used for viewing and printing of XHTML versions of a lesson. eLML totally relies on CSS, so you will have a lot of customization options just by adapting the elml.css file alone (without creating customized templates). Read the **eLML CSS guide** for more information.
   - `_templates/`*`yourtemplatename`*`/elml.js`: Contains JavaScripts used by eLML elements.
3. *lessons*: On the same level you have your eLML lessons. Each lesson has its own folder that should be named just like the lesson label you will have to define. A "lesson label" is a unique identifier (ID) for each lesson, should be lowercase only and have less than 25 digits with no special characters in it.

Validating your eLML lesson

To understand how validation in eLML works you must understand that the eLML XML Schema consists of two parts:

1. General part: This static part of the XML Schema is located in the "core" folder and should not be altered.
2. Project-specific part: This dynamic part is located within your projects "_config" folder (called "validate.xsd") and can be customized to your project's needs.

To validate you will always use your projects "_config/validate.xsd" file and never the "core" files since the "validate.xsd" file automatically imports the "core" files. For more information read the **schema chapter**.

Validate an existing eLML lesson
Please note: The examples below refers to the GITTA lesson "Introduction to Database Systems" that is included in the **stable version**. You will find the lesson XML file here:
gitta/IntroToDBS/en/text/IntroToDBS.xml

1. Open the "Introduction to Database Systems" lesson XML file in oXygen or XMLSpy.
2. Click on the "validate" button (and you should get a "Document is valid" message).

Validate a new eLML lesson
If you created a new eLML lesson and want to validate it you must first reference the eLML XML Schema within the lesson. Here is how you do this in Eclipse/oXygen:

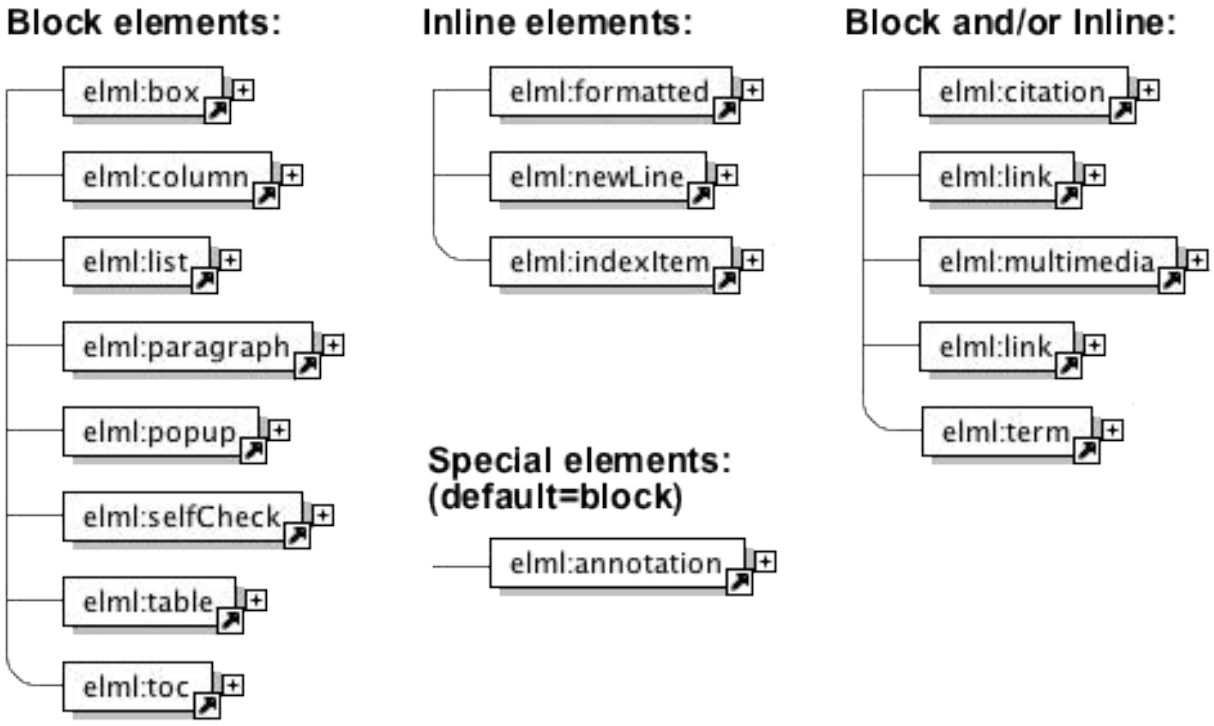
1. Create a new lesson file. **Refer to the manual** for more information about this step.
2. In Eclipse save the new lesson XML file using the "New..."-wizard, enter the file name and choose the location where to save your XML file. Please **refer to the eLML folder structure** and store it within your lessons "text" folder.
3. Click "Next..." (not "Finish"!) and you will be able to reference an XML Schema.
4. Choose the "XML Schema" tab and click on right folder icon to select your projects "validate.xsd" file.
5. The menu "Document root" in the same window should now become an active pulldown menu. Choose the "lesson" element as root tag.
6. The namespace should also be filled in automatically and the prefix should stay empty as we want to work with the default namespace (you can also enter "elml" as Prefix if you prefer working this way).
7. We recommend to check the second checkbox "Add first Choice particle" but not the first one.
8. Now you can click "Finish" and your lesson XML file with the correct schema declaration will be created.

Generally speaking, add the following attributes to your "lesson" root element:
<lesson label="lessonlabel" title="My Lesson Title" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.elml.ch ../../../_config/validate.xsd">

Validate your configuration file
You can use a configuration file to control the transformation process in eLML. This config.xml file can also be validated using a separate schema. Please **refer to the configuration page** for more information.

The eLML content elements

Read the **concept** and **structure** page in the **"about" chapter** to understand the eLML structure and the pedagogical concept behind this structure. Please also read the **eLML CSS manual** because most of both the structural and content elements can be adapted using CSS. For more information **about attributes go to the next page**. The following and last page of this chapter will explain what **top level elements** (lesson, unit, learningObject, glossary, bibliography etc.) are and what they offer.



The available eLML content elements listed according to their display possibilities

The 'annotation' element (special - default=block)

The "annotation" element is meant for containing additional information like links, images, more text etc. about a certain topic. It is a special element because its representation has to be defined by a project or else you won't see any difference to a normal paragraph. This means, by default, the content of an annotation element will be displayed as a normal paragraph as you can see below. You have to create a special (e.g. two column) layout and define that one column contains the content and the second column contains all the annotations to really be able to use the potential of this element. As you can see in the following example, the annotation element is transformed as an DIV tag in HTML with the CSS class called "annotation". Check out the source view of the following paragraph:

This is an annotation element (displayed as paragraph if nothing special defined but can be displayed in a separate column or special window as in this example).

To see one implementation of the annoation element just hoover over the above button and you should see the annotation in a seperate floating window on your left (**check out the UZH layout version** to see this effect in another layout). A second possibility especially when using YAML would be to include the annotation content within a separate third column.

You can include any content you like into the annotation: text, tables, lists, images, movies etc.

The 'box' element (block only)

This box-title is always within the box

> The <box>. element is presented using a DIV tag. The exact layout is defined in the CSS file. Please note that boxes can have titles and icons (like the remark icon used in this example). A box can contain nearly all content elements, with the exception of the "column" element that can only be used on the "top" level of the content elements (right after a structural element like "clarify", "entry" etc.).

Of course you don't need to use all this fancy stuff (note the XML representation below):

> This is a simple box.

<box>This is a simple box.</box>

The 'citation' element (block and inline)

This part describes the different uses of the element <citation>. Please note that this element is one of the eLML elements than can be used as either block, if used as sibling of other block elements, or inline representation, if used nested within another content element! Using the "yearOnly" attribute you can make sure that only the year (and page number) are printed out and the name of the author is not (see example 2 and 4). In this case you will have to write the authors name yourself within the text (if you want it to appear). Here is what the citation element looks like (in example 3 and 4 it would of course be empty and contain no text):

```
<citation bibIDRef="webist2006" pageNr="15">XML is easy to learn.</
citation>
```

Citation used inline

1. The most common use is to include and reference a direct quote. In this case the citation element contains the cited text and the page number where the quote appears should be added using the "pageNr" attribute (optional):
   ... it is stated that "*XML is easy to learn*" (Fisler et al. 2006, p. 15) . This means that ...
2. If you want to cite a direct quote but state the name of the author leading to the citation, then you must use the "yearOnly" attribute and write out the authors name in the text:
   In their book Fisler et al. state that "*XML is easy to learn*" (2006, p. 15) . Therefore, ...
3. The citation element might also be used to paraphrase a work and reference the literary source. In the example the citation element contains no text and is an empty element:
   … often it is written that XML would be easy to learn (Fisler et al. 2006, p. 15). Therefore, ...
4. Sometimes you may want to state the name of the author in the text leading to the citation and have just the publication year in brackets. To do this use an empty citation element included the attribute "yearOnly" set to "yes" and make sure that your write the authors name because eLML will print out only the year:
   In their book Fisler et al. (2006, p. 15) state that XML would be easy to learn. Therefore, ...

Citation used as block

5. If a quote is longer than a couple of words, we may wish to display it as separate paragraph:
"*XML is easy to learn.*" (Fisler et al. 2006, p. 15)

6. Citation as empty block element: This case is currently not displayed in eLML because it doesn't make sense. When using citation as a block element you should always include a quoted text.
Please note: If using the citation element as a block it can contain the attribute icon set to remark, question or important etc. If the representation attribute is set to inline the icon attribute is ignored. Like almost all elements, <citation> can contain the attribute "label" which allows to label it and to link to it from any place within the lesson.

Citation: HTML code Remarks
In older versions of eLML the citation element was displayed mostly in italic font using the `<i>` tag. With eLML 6 and the possibility to create **more elaborate XHTML 1.1 Strict HTML code**, the citation element is transformed into the HTML tags `<cite>`, `<blockquote>` and `<q>`, depending on the representation mode. See the **XHTML 1.1 chapter** for more information.

The 'column' element (block only)
The <column> element can either be used on the "top" level of the content elements (right after a structural element like "clarify", "entry" etc.) or within a box or popup element. You have the possibility to use a two- or three-column layout to better display your content. Within a column you can use all of the content elements with the exception of the column element itself. You also have the possibility to horizontally and vertically align the content of your columns. You can see an example of a column below (where the "list" element is presented) or on the **elml.org** entry page: The news flashes are in one column, the boxes on the right are in a second column. For examples of the align/valign attributes scroll to the end of this page.

The 'formatted', the 'indexItem' and the 'newLine' element (inline only)
The <formatted>, <indexItem> and the <newLine> elements can only be used "inline". That means, they can only be used within paragraphs, boxes etc. The element has only two attributes, one is the "cssClass" attribute to apply unique CSS classes and the other one is the "style" attribute, whose values are listed in the table below:

Tags allowed within simple text	
<formatted>	**formatted 'bold'** ~~formatted 'crossed Out'~~ *formatted 'italic'* formatted 'lower Case' formatted 'underlined' formatted 'upper Case' `formatted 'code'` formatted $_{\text{'subscript'}}$ formatted $^{\text{'superscript'}}$ Formatted elements can also be nested. For example: **This text is bold and $_{\text{subscript}}$.**
<newLine>	<newLine> creates a line break. The attribute 'space' controls the size of the spacing to the next paragraph (values: short, long).

<indexItem>	The <indexItem> element is used to mark special terms that should be listed in the index at the end of the lesson. To create an index you must enter an empty <index> element between the <bibliography> and the <metadata> element of your lesson. An indexItem is usually displayed in italic but its representation can be defined in the CSS file.

The available 'style' attribute values of the 'formatted' element

The 'link' element (block and inline)

Another special type of element is the <link> element. It can also be used inline (nested) or as a paragraph. Please note that a "download link table" will be created if you have more than one link used as a block element right next to each other (siblings). Using it and its attributes appropriately gives several possibilities to link within the lesson or to external resources:

1. External link (inline): To link to another website using an URL you should use the `uri` attribute. Use the `target` attribute if you are using frames (read **here** if you want predefined values for the target-attribute).
2. External link (as paragraph): As above but instead of using the link element within e.g. a paragraph you use it as a sibling of the paragraph element.
3. Internal link within lesson: Use the `targetLabel` attribute to e.g. link to the **Firedocs eLML Editor chapter**. Note that the their needs to be a matching `label` attribute somewhere in the lesson, else the lesson will not be valid. Firedocs will present a pulldown menu of all possible values.
4. Link to another lesson: Use the `targetLesson` attribute combined with the `targetLesson-Lang` attribute (fix values are de, en, fr and it) if the lesson is written in a different language. This will link to the entry page of the lesson.
5. Link to a specific chapter within another lesson: First link to the other lesson as described above and then use the `targetLabel` attribute to another label. This works only if the label is a unit, learningObject, selfAssessment or summary in the other lesson. Please note that this will generate an invalid lesson since the label was not found within the original lesson. You'll have to live with that :-)
6. Download resource (inline): Use the `uri` attribute with relative paths to point to the file. You can also add more attribute like `size`, `type` or `legend`. Example: **this nice paper**
7. Download table with many resource: Use many link elements one after another as block/paragraph elements and you'll get a download/link table like this:

Download demo movie	1GB	MPG	a short moview
eLML Website			Click on this link
Weird sword sound		mp3	might take a while to download this
How to hack eLML?	3 MB	pdf	you need the acrobat viewer to view this

If a link is presented as block the attribute icon can be used to set one of the defined icons (important, remark or question).

Download PDF version of eLML website	5.5 MB	PDF	might take a while to download this

Using the `role` attribute you can define if a certain link is only visible for tutors or also for students.

The 'list' element (block only)
Below a two-column layout using the <column> element and in each column there is an example of the <list> element. The "listStyle" attribute is used to define if you want an ordered or unordered list:

- an item
- another item
- yet another item

1. first item
2. second item
3. third item

Two examples of a list with a title. The first list has nested another list in it, the second example uses the icon attribute and a bibliographical reference:

Title of this list

1. some text
2. another list
    - also text
    - an image could also be used
3. some text

Title of this list

1. first item
2. second item

(Fisler et al. 2006)

The 'multimedia' element (block and inline)
The <multimedia> element is used for everything that is not text. The <multimedia> element can be used inline (e.g. within a paragraph) or as a block (e.g. next to a list, table or paragraph element). It is important to understand the difference: If an image is used inline and aligned left (default) or right, the text is floating around the image. If an image is used as block element, the space left and/or right of the image is left empty. See the examples below and the extensive examples at the end of this page for more information about alignment. The second important thing to know is that these definitions are set within the CSS file. If you don't add the definition for the ".multimedia_*" classes in your elml.css file, there is no floating at all (see the **CSS chapter**). On the other hand: If you don't use a customized layout but work with the plain layout of the elml.xsl file, the most basic CSS class definitions are added automatically.
Here is an overview of the items a multimedia element can contain:
In the online version eLML will transform most of the multimedia elements (except images) using the `<object>` tag plus the deprecated `<embed>` tag. If you dont want eLML to create the oldschool `<embed>` tag, you can deactivate that in the **configuration file**!

- Picture (GIF, JPG, PNG)
- Flash
- Quicktime Video
- MPEG Video
- MP3
- RealOne (Audio/Video)
- SVG
- Java Applet
- VRML
- X3D
- MathML

- HTML or PHP
- etc.

This list will grow as needs grow. Here are some examples of multimedia elements and their XML representation:

Test image with legend and bib-ref (Bleisch et al. 2005)

<multimedia src="../image/test.jpg" type="jpeg" align="right" width="200" units="pixels" legend="Test image with legend and bib-ref" bibIDRef="delfi2005">
The element multimedia also allows to set the attribute icon or to have thumbnails. In this case a small image (url defined in the attribute thumbnail) is shown in the lesson and the original image opens in a new window when clicking on the small image. This is especially useful for bigger flash animations. eLML also features a special "zoom-script" called "lightwindow"! Look at the following example and **read the tutorial** if you want to use this script.

Test animation with legend and thumbnail

<multimedia src="../multimedia/testAnimation.swf" type="flash" align="center" width="200" units="pixels" thumbnail="../multimedia/testAnimation.gif" legend="Test animation with legend and thumbnail">
An interaction included in the lesson (without thumbnail) could look like this.

Only pictures can be viewed in this version! For Flash, animations, movies etc. see online version. Only screenshots of animations will be displayed. [link]

<multimedia src="../multimedia/testInteraction.swf" type="flash" align="left" bibIDRef="webist2006" height="200" width="300" units="pixels" legend="Test animation with legend and bibliography reference">
You should always use the "width" and "height" attributes with movies or interactions. Some browser do have problem showing a movie in the correct size if the width and height are missing! You can see an example in the code above. Use "pixels" as unit since the "percent" option can also cause problems with movies.

A short MP3-Soundfile

Only pictures can be viewed in this version! For Flash, animations, movies etc. see online version. Only screenshots of animations will be displayed. [link]

MPEG Video of a rain radar

Only pictures can be viewed in this version! For Flash, animations, movies etc. see online version. Only screenshots of animations will be displayed. [link]

Including SVG or MathML in your lesson

Both MathML and *SVG* XML languages that most modern browser can display directly without the need for a plugin (Please note that Safari does not display MathML directly...). If you want to use MathML or SVG in a eLML lesson you can either reference a file or directly write code in your lesson.

Example 1: Referencing an external SVG (or MathML) file. Please note that any content within the multimedia element is ignored if the "src" attribute is used:

<multimedia src="http://www.croczilla.com/bits_and_pieces/svg/samples/polygons2/polygons2.xml" type="svg" />

Example 2: Including SVG (or MathML) XML code directly into your lesson file. Please note that you must declare the SVG namespace correctly!

```
<multimedia type="svg">
<svg xmlns="http://www.w3.org/2000/svg" height="300">
<g transform="scale(0.8) translate(-200)">
<polygon fill="blue" points="350,75 379,161 469,161 397,215 423,301 350,250 277,301 303,215 231,161 321,161"/>
<polygon fill="red" points="850,75 958,137.5 958,262.5 850,325 742,262.6 742,137.5"/>
</g>
</svg>
</multimedia>
```

If SVG or MathML are used then the file ending must be .xhtml and not .html! eLML will automatically use the correct file ending if it detects either SVG or MathML in a lesson. Both SVG and MathML can also be used when generating a PDF using XSL-FO. Refer to the chapter **Create a PDF version of your lesson using XSL-FO** for more information.

The 'paragraph' element (block only)

The <paragraph> element is used to write normal paragraphs. It offers a lot of attributes to e.g. make it visible to tutors only, to be used only in the print (PDF) or online (HTML) version, to define a cssClass, a title or an icon.

A title of a paragraph
This paragraph uses both the "title" and the "icon" attribute. You can enter text in a paragraph but you can also enter the following elements: citation, formatted, indexItem, link, multimedia, newLine or term.

The 'popup' element (block only)
A <popup> element allows an author to write a question and to hide the answer. The student then has to click on the question to see the answer. The answer can include multimedia elements, tables etc.

How long would it take to hike to Ulan Baator?

I have no idea but this gives you an impression on how the <popup> element works. The display is defined in the elml.css file and is usually pretty similar to the <box> element. But it can be displayed totally differently. In the print version of a lesson the solution is always visible, since there is no possibility to "click" on a printed document.
By the way: The script that triggers the opening and closing of the box is stored in the elml.js file that should be included in a layout template.

The 'selfCheck' element (block only)
The <selfCheck> element gives authors the possibility to add selfCheck questions (control questions) along with their content. This enables eLML authors to quickly add questions to their content without prior knowledge of Flash or JavaScript. eLML selfCheck questions don't support any *LMS* functionality and therefore can't save the test results of students.

Since the selfCheck element is meant to be used for short quizzes and self-tests we decided that the selfCheck element can only be used within the following structure elments: act and selfAssessment! All the other elements will not allow the use of selfCheck as child element.

The following three types of questions are supported by eLML:

- **single choice** (one correct answer)
- **multiple choice** (more than one correct answer)
- **fill-in-the-blanks** (text with missing words that have to be completed)

The selfCheck element gives you some additional features:

- **shuffle:** single choice and multiple choice questions support the shuffle attribute which shuffles the answers on page reload.
- **feedback:** single choice and multiple choice questions support the feedback attribute which can be added to every answer. The feedback will be shown as a tooltip on the right hand side of the answer after the user tried to answer the question.
- **solution:** all question types support the solution element which can be used to give solution tips or sample solutions. The solution will be shown if the user gives the correct answer or clicks the "Solution" button.
- **synonyms:** the fill-in-the-blanks question type supports adding of synonyms for any specific gap to extend the possible correct answers.
- **images:** all question types support the use of the <multimedia> element to include images and other multimedia content in your questions and answers.

- **custom CSS:** all question types can be customized with CSS.

Question 1 (single choice)

Do you really think you know the answer to the question?

❍ Yes, I do!

❍ No, I'm not really sure.

Question 2 (multiple choice)

Which question types are supported by eLML?

❑ fill-in-the-blanks

❑ single choice

❑ compare

❑ multiple choice

Question 3 (fill-in-the-blanks)

The following text has a few missing words you should be able to complete all by yourself.

The following _____ has a few missing _____ you should be able to complete all by yourself.

The 'table' element (block only)

You can see an example of a <table> element above where the "formatted" and other inline elements are presented. The table below presents the possible attributes of the "table" element:

Example of a title for a 100% width table

Attribute:	**Used for:**
title	Inserts a title above the table
icon	Can be used for displaying an icon in front of the table.
width, height, units	Define the width and height of the table in pixels or percents.
bibIDRef	If a table is taken from a book etc. enter the bibliographic reference here.
align/valign	Used to horizontally and vertically align the content of a table cell. Have a look at the examples at the end of this page.
cssClass	All tables have the "table" CSS class assigned to them. With this attribute you can assign your own CSS class to a table.
legend	Enter a legend for the table. The bibliography reference is also part of the legend, if the attribute is filled out.
role	Create tables only visible for tutors using this attribute.

visible	Create tables only visible in the print or in the online version of a lesson using this attribute.

This legend explains what the table is all about

Each table contains "tablerow" elements which themselves contain "tableheading" and "tabledata" element for each table cell (the former is used for the heading of a table and is displayed in bold). Each tableheading and tabledata element also offer most of the above elements plus the "rowspan" and "colspan" attributes for spanning over rows and columns.

The 'term' element (block and inline)

The <term> element is used to reference glossary terms. If you want to talk about *Cascading Style Sheets* [19] or about *XSLT* and you entered the definition of this term in the <glossary> element/part of the lesson, you can reference it within a text. The definition of the term is then displayed as a mouseover element. Per default the mouseover contains only text and HTML-code, images, movies etc. are only displayed in the glossary itself. If you want to have HTML-code, images etc. also in the mouseover, set the variable `$glossary-MousoverWithHTML` in the **configuration file** to "yes". The script eLML uses for this effect is **Walter Zorns Tooltip JavaScript**. This script is highly customizable (color, font, size, behaviour etc.). Read the **detailed instructions** for more information.

If you want to have a glossary <term> included in a lesson not only as a reference to the glossary but as paragraph in itself you can use the element <term> as a block element and the term and its definition is directly included within the lesson. An example:

eLML:

> eLML, the eLesson Markup Language, is an XML framework developed by the GITTA project. The Swiss eLearning project GITTA started working with XML in 2001 but it was only after the official ending of the project in 2004 that its XML structure was released as an open source project under the name of eLML. For more information read the implementation chapter or visit **www.eLML.org**. (Fisler et al. 2005)

The table of content 'toc' element (block only)

The <toc> element is used on every page of this documentation at the beginning to show a table of contents of the actual page. The <toc> elements offers two attributes:

1. **scope**: Defines the scope of the inserted table of content. Can either be "lessons" (only if a course with **multiple lessons** is transformed), "lesson", "unit" or "learningObject". In the latter case the titles of each clarify, look or act element within the actual learningObject are shown, a level of depth that the normal eLML navigation does not offer! This option was used within this documentation.
2. **recurse**: If set to "yes", the table of content will be shown with two levels of depth (e.g. lessons and units or units and learningObjects etc.). Default is "no" meaning that only the actual scope selected is shown.

[19] Cascading Style Sheets (CSS): A stylesheet language used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document. CSS is a W3C Standard.

All about alignment

This last section is not about a specific element but about an attribute: The "align" attribute of the multimedia, the tableheading/tabledata and the columnLeft/columnRight element and the vertical alignment attribute "valign" used for tables and columns.

multimedia element: inline representation with align=left (default)

This is the test legend

text text

multimedia element: inline representation with align=center

This is the test legend

text text
As you can see, it is not possible in HTML to float around both sides!

multimedia element: inline representation with align=right

This is the test legend

text text
Now let's have a look at the exact same code with the only difference that the multimedia elements are not nested within the paragraph but outside of it and therefore displayed as paragraph:

multimedia element: block representation with align=left (default)

text text

This is the test legend

multimedia element: block representation with align=center

text text

This is the test legend

multimedia element: block representation with align=right

text text

This is the test legend

table/column element and their align/valign attribute values

text example	image example
 *Test with align=left*	text text text text
 *Test with align=center*	text text text text
	text text text text

Test with align=right	
 *Test with valign=top*	text text text text
 *Test with valign=middle*	text text text text
 *Test with valign=bottom*	text text text text

The most common attributes in eLML

In eLML nearly every element does have a specific set of attributes you can use. The attributes are usually specific for an element and thus are described in the chapter about the **top level elements** or the **content elements**. But certain attributes are available for nearly all the elements in eLML and these general attributes are described on this page.

label and labelRef Attribute
The "Label" is the ID that you set for an element. It can have up to 25 characters, numbers or underlines (special characters are not allowed). It is mandatory for the root element "lesson" and its child element "unit". But you can set a label for nearly all the elements in eLML. The label is used to generate filenames or to create anchors within a file. In the online version the label is transformed into an "id" attribute and thus allows the users to reference the object. The following elements will make use of the label-attribute:

- **Link element**: With "targetLabel" you can jump to another place within your lesson or with "targetLesson" you can jump to another lesson. Both use the label-attribute as input value.
- Referencing/highlighting other parts of the lesson using labelRef-attribute (see below)
- **Configuration file**: When creating modules or defining optional units in the config file you'll need the according label

Many **eLML content elements** do also offer a labelRef-attribute. Using labelRef you can reference another part of the lesson and thus make sure they belong together. This could be used for example for highlighing (move your cursor/mouse over the following text): "Understanding the label-attribute is imperative before using the labelRef-attribute". The first paragraph that defines the label-attribute should highlight dark green, when you move your curser over the last sentence. To enable this simple java script in eLML define a color for the $use_labelReferences parameter listed in the "online" section of your **configuration file**.
But this is just one example of how you can use the labelRef attribute. You can of course build your own scripts or use-cases around this features. The basic meaning of the attribute is that you can reference or "connect" one part of your lesson (a character, a sentance, a whole paragraph, a list, a list item etc.) to another part of your lesson that you have defined with the label-attribute.

title and navTitle Attribute
For all the elements that appear in the navigation (lesson, unit, learningObject etc.) or that could potentially appear on one single HTML page, you have to set a title. Other so-called **top-level-elements** like summary, selfAssessment or furtherReading do have default titles that appear if you dont define a title yourself. You can override these default titles in the **configuration file**, by the way. All these elements offer you the possibility to set an optional navTitle-attribute used for the navigation (or the manifest-file, if you create SCORM or IMS CP packages) only. Usually the title-attribute is a longer title displayed on top of a page and the navTitle is a shorter version of the title used in the navigation on the left side.
Then there are **content elements** which offer you the possibilty to set a title displayed within a page. These titles dont appear in any navigation and thus they do not offer a navTitle-attribute.

icon Attribute
Most **content elements** offer an icon attribute. Using this attribute you can mark certain paragraphs, boxes or lists as "important", "remark" or "question". eLML then puts the according icon on the left of the paragraph or box. The idea of this icon is to help visualize for the student "important" (or "question" etc.) paragraphs but if you want you can also write an XSLT function that summarizes all the "important" paragraphs on one page.

The icon-values can be customized, you dont have to use the default values important, remark etc. To customize the icon-values open your **validate.xsd schema file**, search for "icon" and replace the value list with your own values.
eLML expects the icons to be in the folder *yourproject*/_templates/*yourtemplate*/icons/. For every icon-value there has do be an according file in the "icons" folder, so if you have an "important"-icon then eLML expects an "important.png" file in the "icons" folder. PNG is the default format but if you choose to use JPEGs or GIFs you can define the icon file format in the **configuration file**.

Who can see what? role and visible attribute
In eLML you can create a students- and a tutor-version of a lesson. In the students version certain meta-information or solutions is hidden from the user. In the tutor-version this information is displayed. Using the "role" attribute you can mark the following elements to be only visible for tutors:

- unit: Write a whole unit only visible for tutors
- lObjectives: If you have certain goals that you want to hide from the student, mark them role=tutor and only tutors will see those learning objectives
- content elements that can be visible for tutors only: table, paragraph and link

But you can not only customize your content based on your target audiance, you can also customize it based on the target format. Using the "visible" attribute eLML allows you to mark which content is only visible in the online-version or in the print-version, in none of those or in both (default). Here is an overview about what will be displayed in which output format when you are using the visible-attribute:

- visible=online: XHTML, IMS CP, SCORM, YAML and of course if you use your own online layout
- visible=print: XSL-FO/PDF, LaTeX, ODF, DocBook and ePub/eBooks

As a rule of thumb: All the formats that allow playing movies and flash animations are considered "online" and the other formats are considered "print". So since eBooks dont allow containing flash files, **ePub format** is considered "print".

Using the glossary, bibliography, index, list of figures etc.

Besides eLMLs *ECLASS* based lesson-unit-learningObject structure (see the **structure page**) each eLML lessons has some optional top level elements that can be used. These elements are listed below and its purpose and use will be explained here.

Glossary
The `<glossary>` contains all the definition of important terms used within the lesson. It is usually used together with the `<term>` element (see **content elements**) which displays the glossary's term definition as a mouseover as can be seen on this page in the first sentence. To display a terms definition as a mouseover eLML uses **Walter Zorns** JavaScript Tooltip which can be customized concerning colors, fonts, size, behaviour and more (**read the instructions** for more information). A click on the term brings the user directly to the glossary page.
The glossary itself is listed at the end of the lesson. It contains all terms used within the lesson and its definition. The definition can contain all kinds of eLML content elements (multimedia, links etc.). The glossary element offers only one attribute:

- **visible**: Use "all" (default) to display the glossary in all version, "online" or "print" if the glossary should only be visible in the HTML or PDF (both **XSL-FO** and **LaTeX**) version and "none" if the glossary itself should not be listed at all in the navigation. Please note that the glossary HTML page is generated in all cases. This means that you can e.g. put the value to "print" and in the online version create a glossary button that links to the glossary page but does not list the glossary in the online versions navigation. This has been done in the eLML website (click button on your right A-Z)!

An example can be seen here: Compare the implementation in the **LaTeX PDF**, the **XSL-FO PDF** and the **online version**

Bibliography
The eLML `<bibliography>` contains all your references used anywhere in the lesson. Each entry has a unique bibID and you can use this ID within your lesson to reference to your entry. The most common use is certainly done using the `<citation>` element. The citation element offers you various kinds of citations as **described here**. But also the "multimedia" element, the "table" element or other elements allow you to reference to a bibID. And: the furtherReading element used on lesson and on unit level also works with a reference to a bibID. The following attributes are allowed:

- visible: same as above under glossary. **See here** for more information.
- sorting: you bibliography (as also the furtherReading element) can be sorted either by year or by author or it can be grouped together by year or by type. Of course you can also turn the sorting off and leave the order as it is in the XML file.

An example can be seen here: Compare the implementation in the **LaTeX PDF**, the **XSL-FO PDF** and the **online version**

Index
The `<index>` element creates an index based on the `<indexItem>` elements it finds within the document. It also offers the "visible" attribute described under "glossary".
An example can be seen here: Compare the implementation in the **LaTeX PDF**, the **XSL-FO PDF** and the **online version**

List of Figures

The `<listOfFigures>` element will display a list of all multimedia elements (images, flashes etc.) used in the lesson together with a bibliography reference if there is one. This usually makes more sense in the print version (where the page number is shown) than in the online version. It also offers the "visible" attribute described under "glossary".

An example can be seen here: Compare the implementation in the **LaTeX PDF**, the **XSL-FO PDF** and the **online version**

List of Tables

The `<listOfTables>` element will display a list of all tables used in the lesson together with a bibliography reference if there is one. This usually makes more sense in the print version (where the page number is shown) than in the online version. Please note that in eLML you should not use the `table` element for content only and to layout your page. eLML offers the `column` element if you need to align your content. If you don't follow this rule the list of table will not be usable. It also offers the "visible" attribute described under "glossary".

An example can be seen here: Compare the implementation in the **LaTeX PDF**, the **XSL-FO PDF** and the **online version**

Metadata

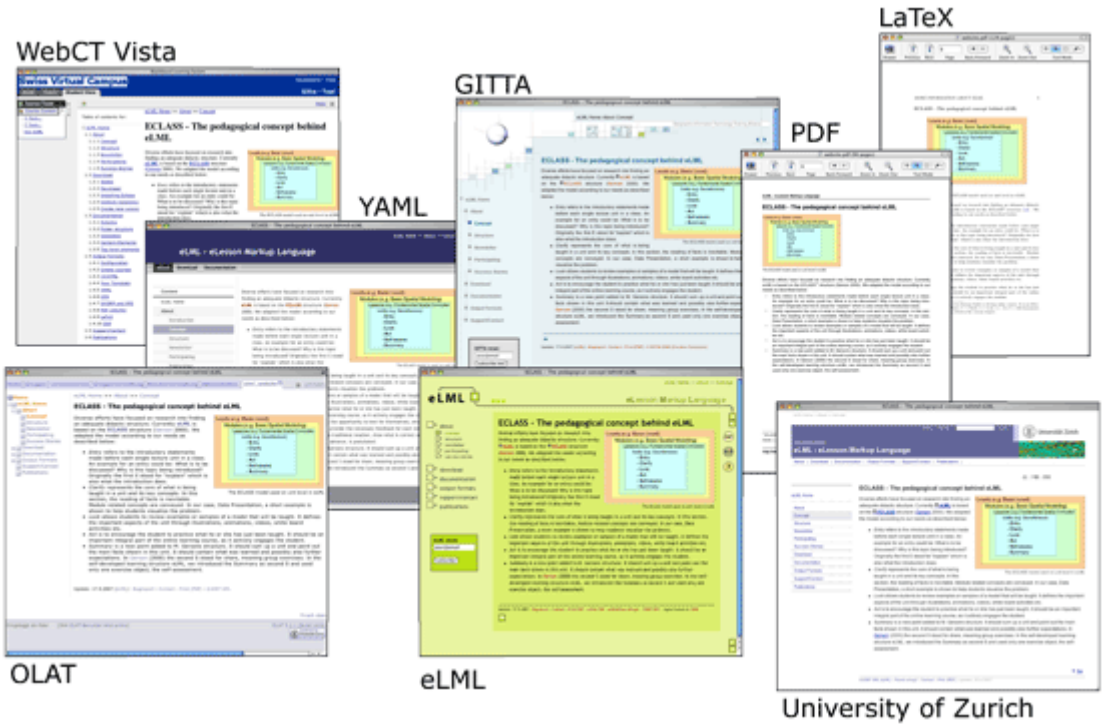
The `<listOfTables>` section contains all kinds of meta information about the authors of the lesson, the length or difficulty etc. Please try to always fill out this self-explanatory part as it helps keep your lessons structure and sustainable. Although the "metadata" element offers a "visible" attribute it can only be listed in the online version. The implementation for the print version is not finished yet. You can use the "role" attribute to define whether it should be visible for tutors only or also for students (or not at all).

Check the metadata for the eLML website **here**.

Output Formats: Transforming your XML file into various formats

XML view of an eLML lesson (click to enlarge)

So you **downloaded** and **installed** eLML, **created a new lesson**, read everything about the **eLML schema** and how to **validate a lesson**. Your lesson in eclipse probably looks similar to the eclipse screenshot (click on image for larger view) of the eLML website - this website was also created using eLML - and now you wonder: What can I do with that XML file? Which formats can I transform it to? Was the effort of separating content from design really worth it? The answer is of course YES and this chapter will show you how to transform your XML file into different formats. Please check back regularly because we are updating our transformation scenarios constantly.
Here is the up-to-date list of formats eLML lessons can be transformed into:



One GITTA lesson shown in different versions/designs using eLML layout templates

Customize your transformation process

While transforming a lesson into any of the formats described in this chapter, eLML always reads the projects configuration file and uses its parameters to create e.g. a student or a tutor version, to display or hide the navigation, to define the position of the page break etc. Therefore it is important to understand how the eLML configuration file is used before starting to transform a lesson. If a project does not have a config.xml file eLML just uses the default values from the eLML core which might lead to unwanted results.

Create a new config.xml file

The location where eLML is looking for a config.xml file is always:

*projectfolder/*_config/config.xml

So the config.xml file is basically in the same folder as the validate.xsd file which you need to validate a lesson! The root element is called "config" and it should contain the configuration schema definition as follows:

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.elml.ch ../../core/structure/config.xsd">

The parameters in the config.xml file

The root element of the config.xml file is "config" as described above. Subelements of it are general, online, print, latex*, modules* and terms* (* means optional). The subelements contain several parameters that allow influencing the transformation. They are explained in detail in the table below. First let's look at an example of a configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
    <general>
        <contact>elml@id.uzh.ch</contact>
        <server>http://www.elml.org</server>
        <role>student</role>
        <pagebreak_level>lo</pagebreak_level>
        <chapter_numeration>no</chapter_numeration>
        <manifest_type>ims</manifest_type>
        <optional_units>
            <labelname>none</labelname>
        </optional_units>
    </general>
    <online>
        <bugtracker>https//jira.elml.org</bugtracker>
        <use_navigation>yes</use_navigation>
        <use_labelReferences>green</use_labelReferences>
        <use_embed>yes</use_embed>
        <html_version>5</html_version>
        <css_framework>yaml</css_framework>
        <icon_filetype>gif</icon_filetype>
        <lightwindow>yes</lightwindow>
        <glossaryMousoverWithHTML>no</glossaryMousoverWithHTML>
    </online>
    <print>
        <display_links>no</display_links>
        <hyphenation>yes</hyphenation>
        <fop_version>0.9</fop_version>
        <pageheight>29.7cm</pageheight>
        <pagewidth>21cm</pagewidth>
        <fontsize>11pt</fontsize>
        <lineheight>15pt</lineheight>
        <fontweighttitle>bold</fontweighttitle>
        <converter_pixel_mm>0.2646</converter_pixel_mm>
    </print>
    <modules>
        <course authors="Joel Fisler" subnavigation="yes" title="eLML Website and Help">
            <labelname>website</labelname>
            <labelname>help</labelname>
        </course>
    </modules>
    <terms>
        <msg name="name_bugtracker" lang="en">Bugreport</msg>
    </terms>
</config>
```

Screenshot of a config.xml file

The section "general" contains all parameters used in every kind of transformation:

element	values	description
contact	an email address	Enter the contact email that students will use to contact the tutor of the course.

server	an url	Enter the URL to your server's home page.
role	"tutor" or "student"	Some eLML element offer the "role" attribute. With this attribute you can add content (like solutions, information about setting up an exercise etc.) that only the tutor should see. In the configuration file you can then define if you want your transformation to be a student or a tutor version. Default is the student version with less information shown.
pagebreak_level	"oneoutput", "lesson", "unit" or "lo"	"oneoutput" transforms all lessons into one single file (html, pdf,...) "lesson" transforms the current lesson into one single file (for the transformation of a single lesson the output of "oneoutput" and "lesson" is the same) "unit" starts for each unit a new file (html) or a new page (pdf, latex,...) "lo" starts for each learningObject a new file (html) or a new page (pdf, latex,..)
chapter_numeration	"yes" or "no"	Switch on ("yes") or off ("no") the automatic numeration of the chapter headings (in the headings and in the table of content).
manifest_type	"ims", "scorm" or "both"	Defines what type of standardised imsmanifest.xml file will be produced. See section **Creating SCORM and IMS Content Package for LMS import** for more information.
optional_units	labelname of units	Units listed here are marked specially so that the student knows that those units are optional learning material.

Overview configuration file: Elements of the section 'general'

The section "online" controls how the XHTML version is generated:

element	values	description
bugtracker	url	Enter an URL to your bugtracker form here. Don't use the element if you don't have a bugtracker in your project
use_navigation	"yes" or "no"	Switch on ("yes") or off ("no") to display or hide the navigation.
use_labelReferences	"yes", "no" (default) or Color Code	With this element you can enable the creation of a highlighting-JavaScript when you are **using the labelRef attribute**. Interpreted are the values 'no' (no highlighting), 'yes' (highlighting with the default color red) or a colour value such as "#FEF456" or "green" (highlighting with the specified colour value).
use_embed	"yes" (default) or "no"	The HTML tag "embed" is officially deprecated but older browser still use it. With this element you can choose wether you want to disable or enable the generation of the embed tag in eLML (eLML will generate the object tag in both cases!).
html_version	"1.0" (default), "1.1" or "5"	Setting this tag does far more than generate older XHTML 1.0 Transitional code, newer XHTML 1.1 Strict code or latest HTML5 code. Read the manual for more information about **XHTML 1.1** and **HTML5** in eLML!
css_framework	"yaml" or "none"	It is possible to use the **CSS Framework YAML** by defining the value 'yaml'. If you dont know what this is leave it to 'none'.
icon_filetype	"png", "jpg" or "gif"	Define the types of Icons you are using. Either gif, png or jpg.

lightwindow	"yes" or "no"	Would you like to use the **lightwindow script** to show large images? Works with the multimedia element if thumbnails are used.
glossaryMousoverWithHTML	"yes" or "no"	Choose "no" if you want your glossary mouseovers to contain text only, choose "yes" if the mouseovers can also contain images, links etc.

Overview configuration file: Elements of the section 'online'

The section "print" is used to define parameters for the *XSL-FO*/PDF transformation:

element	**values**	**description**
display_links	"yes" or "no"	Links are also clickable in the PDF version. Should additionally the link URI be written in brackets after the link text? Choose "yes" or "no".
hyphenation	"yes" or "no"	Do you want to active hyphenation in your PDF document?
fop_version	"0.2", "0.9", "0.95" or "1"	Choose which FOP version you are using.
pageheight		The height of your page (e.g. "29.7cm" for A4).
pagewidth		The width of your page (e.g. "21cm" for A4).
fontsize		Your default font size (e.g. "11pt").
lineheight		Your default line height (e.g. "15pt").
fontweighttitle		Your default font weight for titles (e.g. "bold").
converter_pixel_mm	0.2646	The conversion factor from pixel to mm (do NOT change, should be "0.2646").

Overview configuration file: Elements of the section 'print'

The LaTeX parameters allow the control of the generated *LaTeX* file (a second technology besides XSLFO to create a PDF version):

element	values	description
documentclass	"article" or "book"	Defines the type of document (in LaTeX called "document class") generated.

Overview configuration file: Elements of the section 'latex'

The following section "modules" allows the transformation of several lessons into one course. Please read the chapter **how to create course** to see some examples and a tutorial on how to transform multiple lessons into one course.

element	values	description
course	subelements labelname with valid lesson labels	List the labels of all lessons which build together a course
attribute subnavigation of element course	"on" or "off"	Switch on or off the subnavigation.
attribute title of element course	some title	Define the title of your course.
attribute authors of element course	author names	List the most important authors of your course (they will be shown e.g. on the first page of the pdf. All authors will still be visible in the metadata of each lesson.

Overview configuration file: Elements of the section 'modules'

In the section "terms" you can override predefined terms like glossary, bibliography or introduction etc.:

element	values	description
msg with attributes name and lang	term in the language specified in the attribute lang	Enter your own term for specific predefined terms.

Overview configuration file: Elements of the section 'terms'

Creating courses that contain multiple lessons

Until eLML 3 only one lesson (or technically spoken: one XML file) could be transformed at the same time. Since eLML 4 it is possible to transform multiple lessons into one course no matter what the output format is. So technically spoken you can transform multiple XML files into one PDF file or one content package (*ZIP* file). All you have to do is define a so called "course" in your configuration file. Here is how it works:

1. Be sure that you have a **valid configuration file**.
2. Create a child element of the root node called "modules".
3. The "modules" element can hold as many "course" child elements as you like. Each course will later be transformed into one PDF file or content package. The "course" element offers various attributes like a title for your course, a list of authors responsible for it or the possibility to show or hide the lessons subnavigation.
4. To define which lessons are part of a course you add "labelname" child elements to it. Each "labelname" element will hold exactly one label name of a lesson (in XPath terms: the attribute value /lesson/@label). Make sure that you don't mistype the label, else the lesson will not be recognized!
5. Now transform one of the lessons part of the course (You don't have to start with the first lesson, eLML will recognize the order by itself).

Before transforming a lesson eLML checks if the label name of the actual lesson is part of a course defined in the projects config.xml file. If not, the XML file is transformed alone. If it finds a course that contains the lesson label, eLML starts by transforming the first lesson listed in the course and will continue until it reaches the last lesson label. This works for every available format and of course eLML will also create a continuous numeration if you activated the numeration of chapters. The following example shows you how a course is defined in your config.xml. This course contains the four *GITTA* database lessons IntroToDBS, DBSysConcept, LogicModelin and RelQueryLang (in this order). To see how this course looks in the PDF version click on the PDF navigation screenshot on your right and download the database course as a PDF file.

```
<modules>
   <course subnavigation="no" title="Databases" authors="Susanne Bleisch">
      <labelname>IntroToDBS</labelname>
      <labelname>DBSysConcept</labelname>
      <labelname>LogicModelin</labelname>
      <labelname>RelQueryLang</labelname>
   </course>
</modules>
```

Example of a course definition in the config.xml file: this database course contains four lessons

Restrictions:

- All lessons part of a course must be written in the same language!
- The **folder structure of eLML** has to be used strictly applied!

The most important transformation: From XML to (X)HTML

Since we are talking about eLearning and eLessons the transformation into HTML is probably the most import one. In eLML you can choose to create the latest **HTML5**, **strict XHTML 1.1** or the older XHTML 1.0 transitional code (default). **Described below** is how you choose your HTML version and how you create a plain HTML transformation of your lesson. But you most probably want more than just plain white XHTML pages so when you have read this page continue with the following chapter about **custom layouts**!

- **Create your own online templates**
- **Use the YAML CSS framework for your layout**
- **Create IMS or SCORM content packages and use them within your LMS**
- **Adapt the CSS file to get full control over the display of the eLML elements**

So usually you will work with one or more of the options below when creating your final online version of a lesson or a whole course. If you still just would like to create a simple XHTML version (e.g. to check your content) here's how:

Transform a lesson without a layout template (the 'plain' version)

1. Open your lesson or the "Introduction to Database Systems" lesson XML file in oXygen or XMLSpy.
2. In oXygen click on the "Apply transformation scenario" button. Since you didn't yet define a "Transformation Scenario" you will need to do this now. It's pretty straightforward but refer to the oXygen manual if you don't know how to do it. It is important that you choose an XSLT 2 processor like Saxon 8 or later in your transformation scenario, with Saxon 6.5 it wont work! In XMLSpy just click the "XSL" button.
3. Choose the file `../../../../core/presentation/online/elml.xsl` file as input XSLT file (in oXygen you also have to define an output folder: enter e.g. `tmp/output.txt` but it doesn't really matter since the exact paths for storing files are part of the XSLT 2.0 files anyway).
4. Open the file `gitta/IntroToDBS/en/index.html` with any web browser (in XMLSpy the files are opened automatically).

Difference between XHTML 1.0 Transitional and XHTML 1.1 Strict in eLML
Since eLML 6 you can choose if you want to either generate the older XHTML 1.0 Transitional or the newer and more advanced XHTML 1.1 Stric version when generating HTML code. Please note that there are no differences between the strict versions of XHTML 1.0 and 1.1 that are affecting eLML. All the differences described here have to do either with the change from transitional to strict XHTML code or with the choice made by the eLML authors to create "better" XHTML code. Keep that in mind when you are reading the list below.

How to create XHTML 1.1 Strict code?
By default eLML generates XHTML 1.0 Transitional HTML code. If you want to change that open your **configuration file** and set the $html_version parameter to 1.1. Please refert to the **configuration chapter** if you want to know more about how to configure eLML transformations.

What is different if I choose to create XHTML 1.1 Strict code?
The differences are listed below. As mentioned above those changes have nothing to do with XHTML 1.0 or 1.1 but mainly with the fact that you are switching from transitional to strict XHTMl code. Plus the eLML authors have added some other changes in the XHTML 1.1 version in order to create better HTML code. Refer to the **eLML manual** if you don't know what the elements listed below are used for. Here is an overview:

- A lot of the presentation is moved from using HTML attributes like width, height etc. to using CSS (see next item as example or **read this article**)
- The eLML attributes "width" and "height" used e.g. in the multimedia element are transformed using `<span>` with CSS style attribute instead of using HTML attributes width/height
- The content model is more strict: HTML elements like `body`, `form` or `blockquote` can only contain block elements. Other elements like `span` or `img` have do be descendents of a block element.
- All HTML documents are created using the "xml:lang" attribute to tell the browser what language to expect. Please note that this attribute is the only real difference between XHMTL 1.0 and 1.1 since its not available yet in version 1.0!
- `<citation>`: In the XHTML 1.1 version eLML generates a purely semantic approach using the HTML elements `<cite>`, `<blockquote>` and `<q>` plus the according attributes. **See here** how this looks in your browser! In the older XHTML 1.0 version eLML uses the span- and i-element plus CSS code to display citations.
- `<term>`: Creates `<dfn>` HTML element
- The "target" attributes in HTML links is not allowed in strict XHTML. Thus the eLML `<link>` element's "target" attribute is ignored. This does of course not apply to "targetLabel" or "targetLesson" attributes!
- `<formatted style="bold">`: Creates `<strong>` instead using `<b>` element
- `<formatted style="italic">`: Creates `<em>` instead using `<i>` element
- `<formatted style="crossedOut">`: Creates `<del>` instead using `<span>` with CSS style attribute
- In strict XHTML `<br clear="all" />` is not allowed, only `<br />`. Be aware of this when generating XHTML 1.1 and using the **YAML framwork** (can cause layout problems although only in certain rare cases)

Difference between XHTML 1.1 and HTML 5 in eLML
Since eLML 7 you can choose if you want to generate XHTML 1.0 Transitional, XHTML 1.1 Stric or HTML5 when generating HTML code. Please note that HTML5 includes all the changes listed above for XHTML 1.1 with one exception: The "target" attribute of the "link" element is allowed again in HTML5.

How to create HTML5 code?
By default eLML generates XHTML 1.0 Transitional HTML code. If you want to change that open your **configuration file** and set the $html_version parameter to 5. Please refert to the **configuration chapter** if you want to know more about how to configure eLML transformations.

What is different if I choose to create HTML5 code?
HTML 5 code is more modern and works better on modern browsers. It contains all the changes of **XHTML 1.1** plus some additional ones listed here:

- Navigate through your lesson using shortcut keys like **arrow keys and others listed here**.
- Semantic Web: The new semantic elements section, chapter, navigation, header and footer are generated to structure the code or a page.
- New HTML elements "audio" and "video" are used for multimedia formats QuickTime, MPEG and MP3.
- New element "time" is used to mark the creation date of the lesson.
- And of course: new document type "html" is used.

- More things will be added as the new HTML5 standard is evolving...

Custom Layouts: Create your own templates

Use the Template Builder!

> In summer 2008 Thomas Linowsky created the **Template Builder** as part of the **Google Summer of Code** project. This new tool creates eLML layout templates using a webbased WYSIWYG frontend. It basically makes the manual on this page obsolete. Nevertheless we recommend reading this page since you will need to understand how eLML layout templates work before using the Template Builder. Furthermore you will need this manual if you want to make some fine-tuning of your layout or create layout templates that are totally different from the grid offered by the Template Builder.

eLML website in University of Zurich layout (click image)

In eLML you can create as many layout templates as you like. As an example have a look at the screenshot on your right: This is the eLML website with the exact same content but transformed with the UZH layout template (**view it online here!**)! Interested in creating your own template? eLML layout templates are *XSLT* based and therefore you will need at least some basic XSLT knowledge. To understand how the layout templates in eLML work you must understand the concept of XSLT templates and how the precedence rule works. In XSLT you define so called "template" elements that match a certain eLML element and tells the parser how to transform it. These templates e.g. define how the citation, the term or the selfCheck eLML elements (for a full list see the **content elements chapter**) will look in the online, the print or any other version of the lesson. Now let's say you defined two times within your XSLT file how e.g. the citation element has to be transformed, what will the parser do? It will take the last (not the first!) template it encounters and apply it. To create a personal layout template we use exactly this technique: First we import the core's `elml.xsl` file which contains XSLT templates for all existing eLML elements, then we write our own XSLT templates for the elements we want to transform differently. This works both for the so called "named templates" and the "matcher templates" in XSLT.
The first thing you need to do when creating your own template is to create the templates folder with an empty XSLT file in it:

1. **Download the MyProject.zip file** containing an eLML layout template called "plain" within the "_templates" folder. Copy the whole "_templates" folder into your project folder. It should be on the same level as your lesson folder and the "_config" folder containing the **configuration files**.
2. If you want you can rename the folder (dont use empty space or special character when renaming the folder). Use this name also as the $layout parameter described later!
3. You will see some XSLT files already included within the layout template. We usually name the templates files according to their purpose: online.xsl, print.xsl, odf.xsl etc.
4. The template folder must also contain the files `elml.css`, `elml_print.css` and `elml.js`. Do you want to customize the CSS file? Read more about **CSS in eLML** and the **YAML support in eLML**.
5. Now open your "online.xsl" file responsible for the HTML layout.

Let's look at how the very simple "plain" layout template looks like (Your version might contain more stuff between 3 and 4: This code is needed to display the next/back-buttons within your layout and it is not described here):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:elml="http://www.elml.ch" version="2.0" xmlns:xs=
 1 <xsl:import href="../../../core/presentation/online/elml.xsl"/>
   <!-- ***** Parameter definitions  *****-->
   <!--The name of this layout (=folder name of template folder!) -->
 2 <xsl:param name="layout" select="'plain'"/>
   <!-- ***** Template definitions  ***** -->
 3 <xsl:template name="elml:LayoutBody">
       <body>
         4 <xsl:call-template name="elml:navigation"/>
           <a name="top"/>
         5 <xsl:call-template name="elml:LayoutBodyContent"/>
           <hr/>
         6 <xsl:call-template name="elml:footer"/>
       </body>
   </xsl:template>
</xsl:stylesheet>
```

The very basic layout template to create the 'plain' version

To understand how this layout template works look at the red numbers:

1. The first element after the root element is the `xsl:import` element. It imports the core's elml.xsl file and makes sure that your template will contain the default transformation information for all existing eLML elements.
2. Every template needs the definition of a parameter called "layout". This parameter will hold the name of the layout you create and must be named just like the folder that contains your template files (the subfolder of the "_templates" folder). Your template would already work with only these two XSLT elements but it would not make sense since it would not be any different than the default transformation.
3. The `xsl:template` element named "LayoutBody" contains the HTML element <BODY> with all its content (see 4 to 6). You could e.g. add your own header information here or create a two column layout etc.

4. The first statement of the "LayoutBody" XSL template is a call statement to include the navigation. The core files contain a named template called "navigation" which will insert the navigation as an unordered list. For formatting you can use the CSS file and create a customized navigation.
5. After an HTML anchor named "top" the template contains a second call statment which will include the whole content of the page.
6. Preceded by a line (HR tag) the third call statement includes the footer of the page.

This example shows you how a very simple eLML layout template could look. Please refer to the list below for a complete list of named templates you could include with the `xsl:call-template` element.

Transform a lesson with your custom layout template

1. Open the "Introduction to Database Systems" lesson or your own XML file in oXygen or XMLSpy.
2. Use your own template or the following XSLT file for transformation: `gitta/_templates/gitta/online.xsl`
3. Open the file `gitta/IntroToDBS/en/index.html` with any web browser (in XMLSpy the files are opened automatically). You should now see the lesson in the blueish "gitta" layout or your own layout.

The named templates available from the core files

The eLML core files contain many named templates you can use within your own layout templates. They will include stuff like the navigation, a forward/back link, the footer etc. It is recommended to use the existing named templates instead of programming your own. The eLML team will ensure that these templates always work so even with possible future schema changes you don't have to do any debugging if you use the provided templates. Here's a short overview, for a detailed view please open the `elml.xsl` file directly and have a look at the content.

- `elml:LayoutHead`: Defines the HTML tag <HEAD> and its content. You should not rewrite this template if possible.
- `elml:LayoutBody`: Defines the HTML tag <BODY> and its content. This is one of the main (maybe even the only one!) templates you have to overwrite as shown in the example above.
- `elml:LayoutBodySkiplinks`: Includes the hidden skip links used by screen readers and other special browsers e.g. for blind people. You should always call this template directly after the <BODY> tag (Unfortunately this wasn't done in the example above).
- `elml:LayoutBodyContent`: As shown in the example above a call of the elml:LayoutBodyContent template will include all the content for the actual page being processed. Depending on the pagebreak_level used this named template will include the content of the whole lesson, of each unit or of a single learningObject.
- `elml:next_file` and `elml:prev_file`: These named templates will output the filename of the following or preceding page. You can use them within HTML anchor tag as value for the "href" attribute. Use the `xsl:attribute` element to do this.
- `elml:ims_metadata`: A named template that will output all the metadata available according to the IMS LOM standard. You probably want to output this metadata in a single page using the xsl:result-document element.
- `elml:navigation`: Includes the whole navigation as an unordered list. Each <ul> and <li> contains CSS classes and IDs and therefore can be customized by changing the CSS file. We strongly recommend to use this navigation template and customize its view via CSS instead of creating your own navigation XSL template!

- `elml:path_full`: Inserts a breadcrumb navigation (can also be customized via CSS).
- `elml:footer`: Either use this template to include a footer (as in the example above) or copy the whole XSL template to your own file and adapt it.
- `elml:LayoutTooltipScriptConfig`: If you want to customize the way your term tooltips look (color, font, position etc.) you need to copy this named template located in the `scripts_styles.xsl` file into your own XSLT file and alter the parameters!

Customizing the tooltip script

To display a definition as a mouseover window the **term element** in *eLML* uses **Walter Zorns** tooltip script. This Java Script is highly customizable and should satisfy the needs of most projects. If you want to change the default color, font, position or behaviour of the tooltip script you will only have to add one template to your online.xsl file:

1. Open the file `/core/presentation/online/scripts_styles.xsl`
2. Look for a template named `elml:LayoutTooltipScriptConfig` and copy the whole template definition
3. Paste this xsl:template to your online.xsl (anywhere) and adapt the parameters to your needs (check **Walters documentation** with its list of all parameters and its values)
4. Transform your lesson again and the script should reflect your changes

Of course you can use the script for other stuff too. As an example check the **implementation of the annotation tag** on this wesbsite. By the way, if you dont like the glossary term icon used in eLML you can either replace it or you can hide by adding the following line to your elml.css file:
.term_icon {visibility: hidden;position: absolute; top: 0px; left: 0px}

Using the lightwindow script for image-previews

You may have noticed that we used a script for zooming into images and screenshots. Click on the screenshot thumbnail on this page (in the upper right corner) and you'll see that the image "zooms" and gets bigger. This effect uses the "lightwindow" JavaScript from **Stickmanslab**. How can you implement the script in your eLML-lesson? Here's a short tutorial:

1. Open your **configuration file** and set the lightwindow-parameter under "online" to "yes". Now eLML generates all the necessary links and classes.
2. Download the **Lightwindow-Script** and extract the ZIP file. You will have a "lightwindow" folder now on your desktop. Please move the "lightwindow" folder into your `_templates/mytemplate/` folder.
3. Create `multimedia` elements with the "thumbnail" attribute pointing to the clickable small version of the image and the "src" attribute pointing the large version of the image or to the movie, HTML-page etc. This object will open in your screen in full resolution (does not have to be an image). Use the "width" and "height" attributes to define the display size of the thumbnail! The width/height of the full size object is defined automatically by the script.
4. Open the file `elml.css` (which should also be in your templates folder) and at the beginning of the file add the line `@import url(lightwindow/css/lightwindow.css);`
5. Transform your lesson and you should see the script in effect when clicking on any thumbnail image.

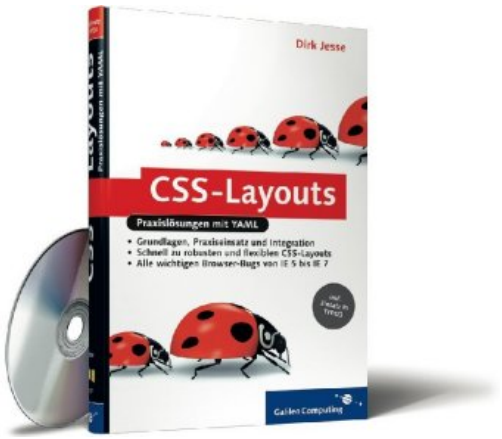
As an alternative you can use the `link` element (for this purpose the link must be used as inline and not as block element, e.g. within a paragraph). Add the @cssClass attribute with the value "lightwindow" and your link will be opened in a lightwindow frame and zoomed to maximum size. Example:

```
<paragraph>This is a very interesting <link cssClass="lightwindow"
 uri="http://www.google.ch">google link</link> that you might al-
ready know!</paragraph>
```

Lightwindow options: The lightwindow script offers many options **described in detail in the manual**. We made a minor "hack" of the multimedia element if you want to use these options: You can enter these options into the multimedia element as text (not attribute!) and eLML will use everything you enter as the "params" attribute of lightwindow. Example of a flash shown in a window of size 320x240:

```
<multimedia type="flash" src="../multimedia/yourflash.swf" thumb-
nail="../im-
age/yourthumnail.png">lightwindow_width=320,lightwindow_height=240</
multimedia>
```

Custom Layouts: Using the YAML CSS framework



If you ever tried to design a sophisticated website using CSS techniques only, you know what we are talking about: To let your content and your navigation elements look great on every browser and every operating system is an almost impossible task. Even if modern web browsers support the CSS standards quite well, there are still many differences and browser bugs around. Even if you are a CSS crack you always have to keep your combination of browser hacks and workarounds up to date. With every new browser version you will have to check whether everything is still working as it should. Since YAML does this job very nicely we not only rebuilt the elml.org website based on YAML but we also integrated YAML support into eLML.

YAML (Yet Another Multicolumn Layout) is a CSS framework that takes care of all the CSS hassle mentioned above. It provides a flexible structure to build your custom design on. Even "out of the box" the design looks pleasing and the content is accessible for all users. YAML is open source, well documented and supported by a growing user and developer community.

The YAML CSS book (recommendable)

You might consider using eLML with YAML if you:

- Plan to provide your eLML content standalone, that is outside a Learning Management System (LMS) or CMS.
- Want to develop a custom CSS-based eLML design without having to start from zero.
- Need to integrate your eLML content in CMS. There are already several **YAML templates** for integration in popular CMS available.

How to use YAML in eLML?

Screenshot of YAML layout example (click image)

If you want to use YAML, here is how you do it:

1. Download the latest **YAML-release** and unzip it. Within the folder you will find documentation, examples, tools and a folder called "yaml".
2. Copy only this folder "yaml" into your projects "_template" folder. Please note that the "yaml" folder should not be placed within a specific layout template folder but one level higher within the "_templates" folder. This way if you have many different layout templates they can all use the same YAML-CSS-Files.
3. Create your own layout template **as described on this website**. You can create your layout template as you wish but your "elml.css" file should start with the following two lines:
```
@import url(../yaml/core/base.css);
@import url(../yaml/screen/content_default.css);
.hidecol2 #col3 { margin-left: 25%; margin-right: 0; }
.hidecol2 #col3_content{ padding-right: 20px; }
.hideboth #col1, .hideboth #col2, .hidecol1 #col1, .hidecol2
 #col2 { display:none; }
```
4. In your projects **configuration file** under "online" add an element "css_framework" with the value "yaml".
5. Now you can open your lesson.xml file and transform it using your layout templates XSL file. Your content should be visible in your layout and using the YAML-CSS-Framework.

If you create your own template we recommend using the **YAML Builder** to create your HTML code. YAML expects a specific HTML structure and some specific CSS class names in order to work correctly and the YAML Builder helps you to create this HTML code. You should now **create your own template** named "elml:LayoutBody" and copy the XHTML code in it. Now you need template calls to insert the content (`elml:LayoutBodyContent`), the navigation (`elml:navigation`) or the footer (`elml:footer`) at the places you want them to appear. Check the **preceding chapter** to get more information about this step.

If you are interested in YAML and would like to use it for your project send a **support request**. If you would like to participate in the discussion about eLML and YAML, feel free to send your comments via the **eLML Users Mailing list**.

Specials

Columns in YAML

Using YAML means using the classes and elements provided by the YAML framework. One of the elements provided by YAML is the **Subtemplates** features for two- or three-colum layout within one of the main YAML columns. The according element in eLML would be the "column" element. Now in eLML absolute values for column widths are allowed, in YAML - due to the way columns are implemented - this is not always possible. Therefore please consider the following points when using YAML and the eLML element "column":

- Try to use width in percent-values instead of absolute values in pixels. Check the **YAML subtemplates page** for the possible values. eLML will automatically use the closest available YAML-ratio (e.g. if you use 20% width for columnLeft, eLML will use the 25% YAML-class).
- Absolute width-values in pixels are only applied if there is no columnMiddle and if the width is applied to the columnRight element.
- If columnMiddle element is used, all the three columns (left, middle and right) will have a width of 33%.
- If only columnLeft and columnRight are used but both without with then both columns will have a width of 50%.

Further comments

- When creating content packages with YAML you might have problems with scrollbars appearing in your *LMS*. Please add the following line to your elml.css at the end of the file:
  ```
  html { margin-bottom:0; }
  ```

The eLML CSS Guide

Some general remarks about eLML and CSS

Since the start of the project in 2004 eLML has been building on *CSS (Cascading Style Sheets)*. With the launch of eLML 3.0 in April 2006 the "core" files have changed to fully support CSS and even use custom CSS classes defined by the projects themselves. All the CSS definitions are made in the `elml.css` file located within the corresponding templates folder. In eLML there are structural elements like "clarify", "look" and "act" parts of a learningObject and there are content elements like "table", "box" or "paragraph". The latter can be overridden by the @cssClass attribute of these elements. The available list of @cssClass attribute values is defined within the _config/validate.xsd file of the project and should not be changed by the author. It is imperative that the project defines as a whole if they want to use custom CSS classes and if so, how these classes will be named. Then the designer can do the implementation of these classes. But remember that each eLML element already is assigned to a CSS class (usually named the same as the element name). So customized CSS classes would only make sense if e. g. you want two or three different representations of a <box> element. But if all your boxes should look the same, you can just use the "box" CSS class and define it as you like. Here is an example:
<box>Hello World!</box>
within your eLML XML file will be transformed in your XHTML document as follows:
Hello World!
and the elml.css file contains the definition of this class:
.box {background-color:#BADE17; background-position:0 100%; padding:0 1em 1em; border:solid 2px #747566;}
Which will look like this on the eLML website:

Hello World!

In this example with only two words the box looks a bit strange. Therefore let's say you define your own CSS class called "fancylooking" and you use it for your box:
<box cssClass="fancylooking">Hello World!</box>
the code of your XHTML document would look like this:
Hello World!
We now define a totally different kind of box display for the CSS class "fancylooking":
.fancylooking {background-color:red; padding:1em; border:solid 2px black; margin-bottom:2em; width: 100px;}
Which this time will look like this on the eLML website:

Hello World!

All other "box" elements without the @cssClass attribute will still all be assigned to the "box" CSS class. The same is true for most eLML elements: If no @cssClass attribute is set the elements name is used as CSS class name. Have a **look at the source code**!

Content elements that can be overridden by the cssClass attribute (mostly content elements):
The following elements will all have the possibility to define a "cssClass" attribute to override the default CSS class. If this attribute isn't used, the element's name is used as a CSS class in the final XHTML document (as illustrated above).

- box: Used for the DIV tag that defines the box. The title is also inside this box.
- citation: A citation is always surrounded by a SPAN tag of that class. If a citation is shown inline or as paragraph, it is defined by the transformation using display:inline or display:block respectively. Within the citation text itself the citation is in italics using the I tag. The link to the bibliography has the CSS class "bibLink" assigned.
- column: Defines the TABLE tag that is used for a two or three column layout. Each column (TD tag) can have its own class. If not, the default CSS classes are named after the element (so: columnLeft, columnMiddle and columnRight)
- columnLeft, columnMiddle and columnRight: The TD tag that is used to display a two or three column (including columnMiddle) layout.
- item, itemAlt: Defines the LI list item of a list. If no @cssClass is defined, the LI classes are named alternating "itemAlt" and "item".
- link_table: If a link is displayed as paragraph, a TABLE with this CSS class is used (e.g. to generate download link tables).
- link: Used for the anchor A tag in both the block and the inline representation. If a block representation is used, the table that displays the download links has the CSS class "link_table".
- list: Defines either an OL or a UL tag depending on the @listStyle attribute of the list element.
- multimedia: The multimedia element itself (IMG, OBJECT or APPLET tag) gets this CSS class.
- paragraph: The P tag of a paragraph uses this CSS class.
- popup, popupTitle: The popupTitle CSS class is used for the sentence the student clicks on to view the solution (usually the question defined in the @title attribute of the popup together with a default "click here" text in brackets). If an icon is used, the icon is also part of the popupTitle class and can be clicked. The popup CSS class, on the other hand, is used for the box that opens afterwards containing the solution (defined with a DIV tag). It is usually similar to the box class but it can be defined according to personal taste using the @cssClass attribute.
- table: Defines the TABLE tag used by the table element.
- tabledata, tableheading: Defines the TD tag, and respectively, the TH table cell of a content table.
- tablerow, tablerowAlt: Defines the TR tag within a content table. If no @cssClass is defined the CSS classes are defined alternating: tablerow and tablerowAlt.
- term: The term class is only used for the inline representation of the term element (usually an anchor A tag). For the block representation, the "glossary" CSS class is used, as with the glossary itself.

eLML elements that cannot be overridden (mostly structural elements)

- clarify, look, act: Within a learning object each clarify/look/act element is nested within a DIV tag belonging to the CSS class named accordingly.
- glossary: Defines the glossary term definitions. Usually a DL tag. Also used for term elements displayed as block within the text.
- goals: Defines the goals list. Either a TABLE or a UL tag depending on the @presentation attribute. Each item (learning objective) uses the CSS class "lObjective" for the TR (@presentation=table) or LI (@presentation=list) tag.

- lObjective, lObjectiveAlt: Either the LI tag of a list or the TR tag of a table depending on how the goals @presentation attribute is defined. Alternating classes are used!
- icon: When an icon is used both the IMG and the TD tag that contains the icon image are defined as CSS class "icon".
- multimedia_*: Multimedia elements are all surrounded by a DIV tag that has the multimedia_container CSS class assigned to. Things like text-align, float, clear, padding, display=block or display=inline etc. are defined directly with the transformation!
- index_list: Defines the index list. Usually a UL tag.
- index: An indexed word with the text is surrounded by a SPAN tag of this class.
- legend: Defines the legend of a multimedia element or table. In HTML a SPAN tag with display=block (for correct display below images etc.) is used!
- tutor: CSS class used for things that are only visible to tutors. Can be P, LI, TD etc. tag in HTML representation.
- footer: The footer is usually displayed smaller and is defined by this CSS class.
- glossaryTooltip: Defines the box with the glossary description when the user moves the mouse over a term.
- metadata_table: Defines the table used within the metadata section to represent data.

The bibliography listings by default uses HTML tags for bold and italic but these tags can also be customized as following:

- bibliography: Defines the bibliography list. Usually a UL tag.
- furtherReading: Defines the further reading section. Usually a UL tag.
- bibLink: Used for the anchor A tag that links the author's name within a citation/reference to the according bibliography item at the end of the lesson.
- bibAuthor: The author's name is usually bold (B tag).
- bibTitle: The title of the resource (book, paper etc.). Defined with a I tag and therefore usually italic
- bibCommentFurther: Using a SPAN tag this class defines how a text added to a further reading resource is presented. Usually italic.
- bibCommentSource: Using a SPAN tag this class defines how a text added to a bibliography reading resource is presented. Usually italic.

A closer look at the navigation

The default navigation created by eLML is a special case since it does not only use CSS classes but also CSS IDs to identify the exact position of the navigation item.

As can be seen in the **source code of this documentation file**, the navigation of eLML is built up by lists (UL tag) and list items (LI tag) that are then defined in the elml.css file. These two HTML tags belong to the following CSS classes:

- UL tag: Always belongs to the "navigation" class. Depending on the level, the UL tag also has a CSS defined ID called "nav_lesson", "nav_unit" or "nav_lo"!
- LI tag: All links to pages use the CSS class "navigationLink", only the active page navigation item uses the "navigationActual" CSS class to allow special highlighting (and is NOT a link).

The whole eLML website navigation is built up only by defining the corresponding fonts, list-stlye-image etc. within the elml.css file. Have a look at the **source code** of this page and at the **elml.css file** for more information :-)

Creating SCORM and IMS Content Package for LMS import

Using eLML lessons with a Learning Management System (LMS)

A very convenient way of using eLML lessons is to integrate them into an open source or commercial *Learning Management System* like *OLAT* or WebCT. This is done using the *IMS Content Package* or *SCORM* standard. A content package is basically a *ZIP* archive with all your XHTML files, images, flashes etc. and with a XML file called imsmanifest.xml at the root level. The imsmanifest.xml contains both the metadata about the lesson (according to the IMS Metadata standard) and references to the different files.

Does it work with my LMS?

We did successfully test either the SCORM or the IMS Content Package with the following learning management systems:

- OLAT (see video below)
- Moodle
- WebCT 4/6/Vista (see video below)
- Ilias
- Dokeos
- ATutor

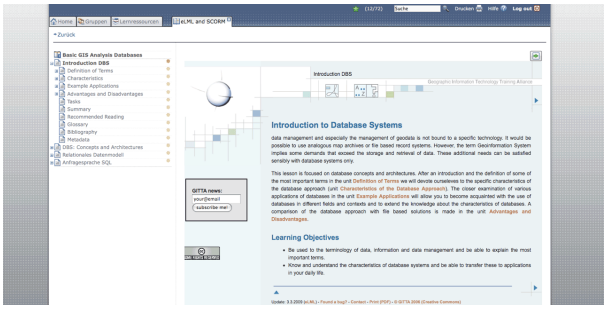
Please let us know if you imported eLML lessons into other learning management systems or **contact us** if you have problems.

Can I test it with my LMS?

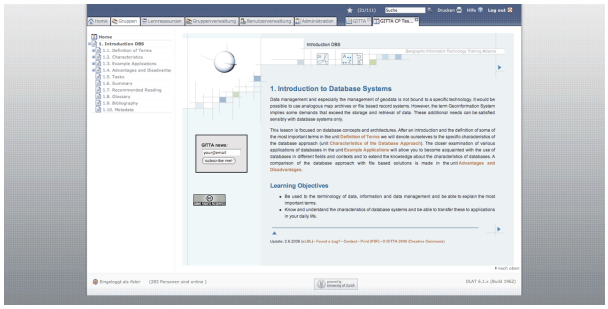
Of course! We did prepare some test-lessons you can download here:

GITTA test-lesson	1MB	IMS CP DBS Introduction
GITTA test-lesson	1MB	SCORM DBS Introduction

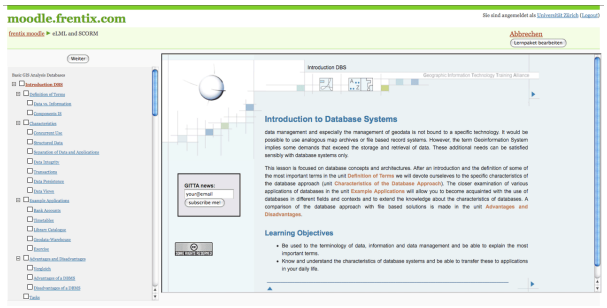
Screenshots of a GITTA lesson created with eLML, transformed into SCORM or IMS CP format and imported into different LMS:

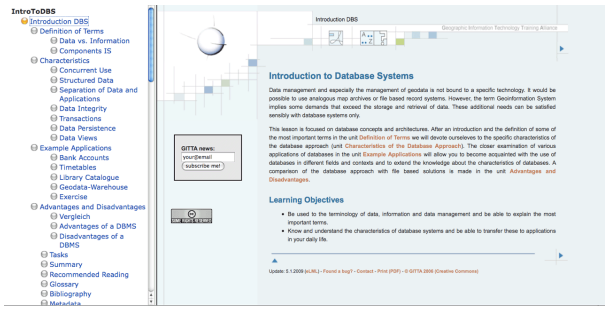


eLML SCORM package within OLAT

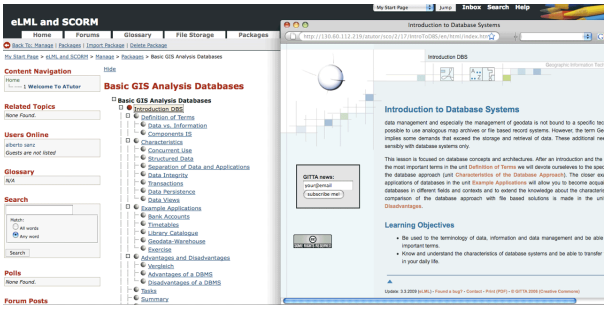


eLML IMS Content Package (CP) within OLAT

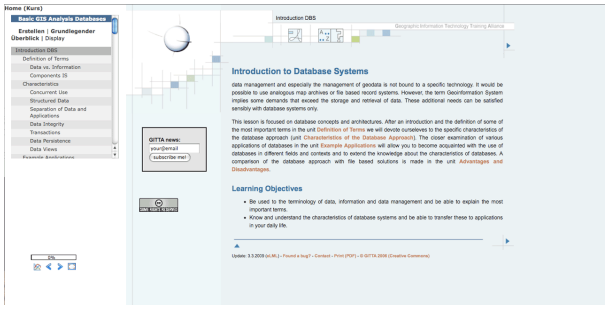




eLML SCORM package within Moodle



eLML SCORM package within Ilias



eLML SCORM package within ATutor

eLML SCORM package within Dokeos

How to create a SCORM or IMS CP in eLML?

To generate a content package of your lessons please follow these steps:

1. Be sure that your lesson is valid and that the transformation of standalone XHTML pages works.
2. Set the "pagebreak_level" parameter to either 'unit' or 'lo' to generate multiple output pages.
3. Set the "use_navigation" parameter to 'no' since you want the LMS to generate your navigation and not have it included into the XHTML files. Read the **configuration chapter** for more information about the transformation parameters you can set.
4. Set the "manifest_type" parameter to either 'ims' or 'scorm' depending on the type of content package you want to create. Please note that if you use 'both' the imsmanifest.xml files are stored into two separate folders and you will manually have to move one of the to the root folder. There is no way eLML can create for you automatically both the IMS CP and the SCORM package.
5. Use a simple layout like the "plain" layout since the LMS will be responsible for the layout. You can also use the **Template Builder** to create a simple layout that look good within your LMS. If you transform your lesson using the elml.xsl file directly, the plain layout is used by default.
6. Go into the eLML project folder and use the command: "`zip -r lessonname.zip .`" (This step can be done on OS X by using the context menu and choose "Create archive of ...". On Windows there are similar shortcuts to create ZIP archives.)
7. Upload this ZIP file into your LMS and it will be recognized as a content package.

Please note that in point 6 it is important that you do the zipping within your project folder and not outside of it. This way it is guaranteed that the imsmanifest.xml file is at the root level of the ZIP archive. Else it would be under yourpoject/imsmanifest.xml in the ZIP archive and would not be recognized by the LMS.

Videos showing how to import eLML lessons into OLAT or WebCT

The exact installation procedures to import an IMS or SCORM package into your learning management system (LMS) is described in detail within your LMS manual. To give you an idea, we provide three short installation screenshot movies for the open source LMS *OLAT* and for the commercial WebCT platform:

Import IMS CP into OLAT	2.4MB	QuickTime	OLAT 4.2 Demo
Import SCORM into OLAT	1.4MB	QuickTime	OLAT 4.2 Demo
Import IMS CP into WebCT CE 4.1	2.7MB	QuickTime	WebCT CE 4.1 Demo

Import SCORM into WebCT CE 6	5.5MB	Quick-Time	WebCT CE 6 Demo
Import SCORM into WebCT Vista	3.9MB	Quick-Time	WebCT Vista Demo

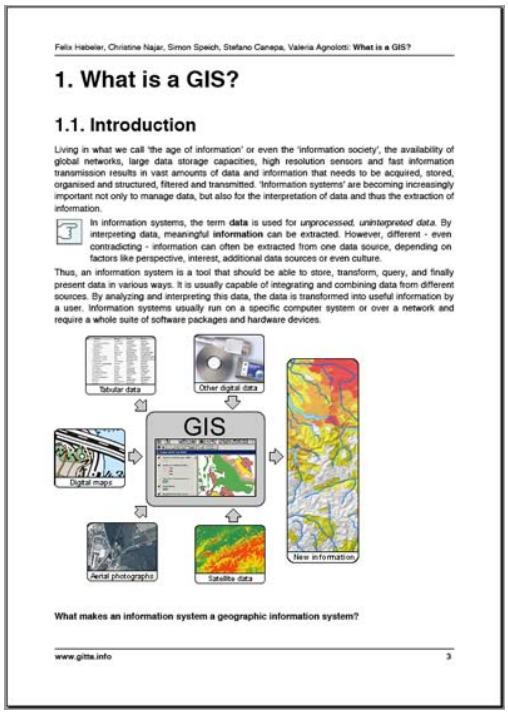
Please note that when importing a SCORM module into WebCT, the resulting error can be ignored. It appears because we referenced the SCORM schema by absolute URL (http://...) and din not use relative paths. It works perfectly.

SCORM and user tracking

eLML supports basic user tracking functionality of the SCORM standard such as setting the completed status and tracking the session time. Tracking the session time requires the call of a javascript function on every body onunload event. If you are using your own online transformation stylesheet instead of the default online tranformation stylesheet provided in the elml core folder, you may change the body tag in your online transformation stylesheet to:

```
<body>
<xsl:if test="$manifest_type='scorm'">
<xsl:attribute name="onunload"> <xsl:value-of>finish()</xsl:value-
of> </xsl:attribute>
</xsl:if>
...
```

Create a PDF version of your lesson using XSL-FO



Example of a PDF version (click image to download PDF)

In eLML you have two possibilities to create a PDF version of your lesson: Via XSL-FO or via LaTeX. The first method is described below, the latter method is described in the **LaTeX chapter**. Both methods have their advantages and drawbacks. For the XSL-FO approach (using the open source Apache FOP to process the FO file) the following drawbacks are known:

- Both the Apache Formatting Object Parser (FOP) version 0.25 and 0.9 do not yet support the full range of XSL-FO commands which sometimes leads to layout errors. But since 2010 FOP 1.0 is out and you should make sure that the XML editor you are working with uses this latest release. If it doesnt, **intall FOP 1.0 yourself**.
- The layout of the PDF file is defined via XSL-FO commands and cannot be changed easily.
- The XSL-FO technique is generally spoken more complex and less known than LaTeX. But if you're fine with the included eLML transformation file (see **PDF version of eLML website** as an example) you don't have to care about this point anyway. Just go through the four steps below and you're done.

How to create a PDF version of your lesson:

1. Open the "Introduction to Database Systems" lesson XML file in oXygen or XMLSpy.
2. Have a look at the "print" section of your projects **configuration file**. With the $display_links parameter you can tell the parser if the full URL of links used in your lesson should be printed or not (they are clickable links in both cases but that of course only works if you look at your PDF on your computer). Furthermore you can define if **hyphenation** should be used, which Apache FOP version you are working with, etc.
3. This time use the following XSLT file for transformation:
   `../../../../core/presentation/print/elml.xsl`

4. In your oXygen transformation scenario click the "Perform FO Processing" mark in the middle tab and enter a "real" output path. For example you can enter `${cfd}/${cfn}.pdf` and oXygen stores the PDF file in the same folder as your XML file.

Use the **Template Builder** to customize the look of your PDF file! If you want to adapt the look of your PDF file even more you will need to **create your own eLML layout template** for the FO-transformation. But beware: FO is not for the faint-hearted :-)
If you are using *SVG* within your lessons you can use it directly within your PDF file. In Adobe Flash or Illustrator you can export your graphics as SVG. Add this SVG file to your lesson.xml file using the "multimedia" element. You might want to set the "visible" attribute to "print" to make sure that this file is only used within the print-version. Then transform your lesson as described above into an FO file and use the Adobe FO parser to transform the file into a PDF document.

Install Apache FOP 1.0 (with hyphenation) yourself

1. Go to the Apache FOP website and **download the latest stable binary release** of FOP 1.0
2. To be able to use hyphenation go to the **OFFO-project** (Objects For Formatting Objects) and **download the latest stable** release plus the "offo-hyphenation-utf8" files.
3. After unzipping the files move the "fop-hyph.jar" file into the "build" folder (where the fop.jar file resides) and move the whole "hyph" folder (with all the different language hyphenation patterns) into the main FOP-folder where the "build"-folder is also located.
4. Now Apache FOP is ready to be used in command line mode with e.g. the following command:
```
/path-to-fop-folder/fop  /path-to-workspace/project/lesson/en/text/lesson.fo  /path-to-workspace/project/lesson/en/text/lesson.pdf
```

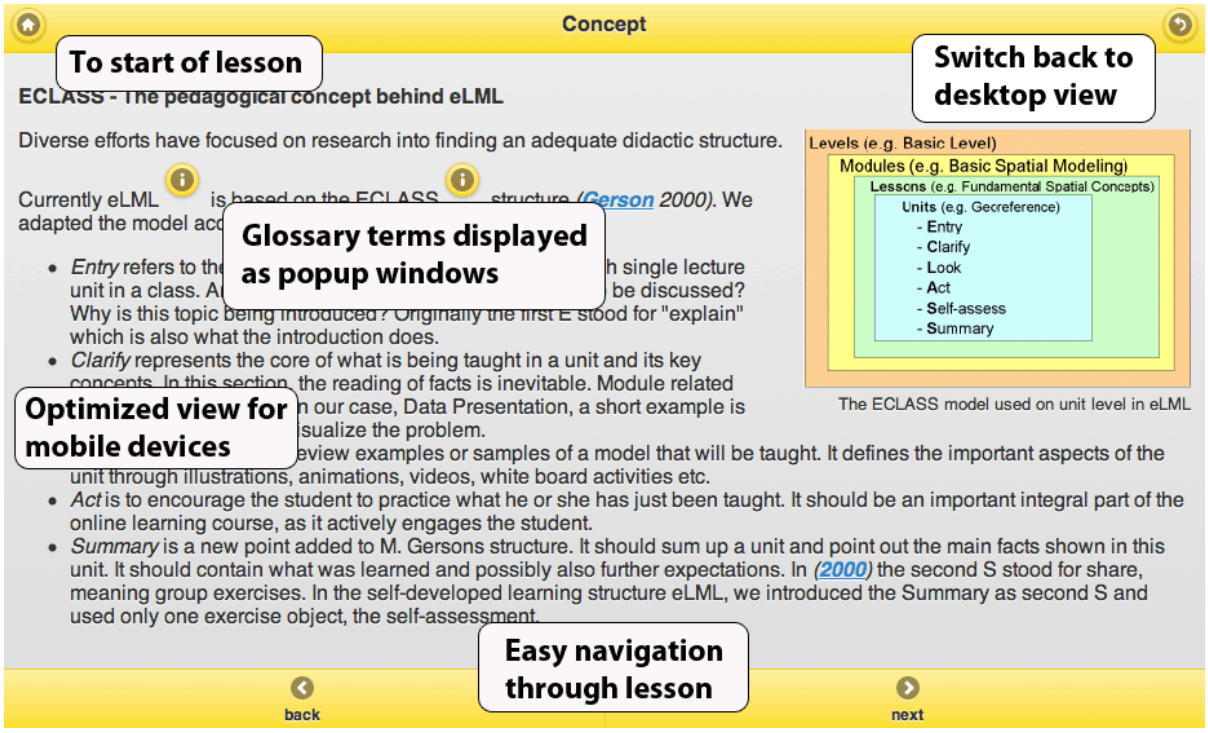
Including SVG and MathML in your PDF
Apache FOP supports the rendering of both SVG-images and MathML-formulas within your PDF, but you will need two additional libraries:

- For SVG: **Apache Batik**
- For MathML: **JEuclid**

If you are using an XML editor like Oxygen, chances are good that Apache FOP is already installed with the two libraries above. Otherwise you will have to follow the according manuals to download and install the libraries. Usually all you have to do is download the library and move it into the "bin" folder of your Apache FOP installation.
For more information visit the **FOP installation guide**.

Mobile view of your eLesson

The mobile view is a special layout optimized for smartphones and handheld devices. It is based on **jQuery Mobile** and works with most operating systems or devices.



Screenshot of mobile view

Features

- Layout based on **jQuery Mobile** 1 final
- Supports iOS (iPhone, iPad), Android, Blackberry, Windows Phone, Kindle and many other browsers
- Includes buttons to navigate (back/next), go to home page or to switch back to regular desktop view
- Glossary terms displayed as popups

How to create a mobile version?

1. Make sure that you first create a normal HTML-version (either **plain layout** or using **your own template**). This will be the desktop-version that a mobile user can always switch back to. Make sure you are using **HTML5** for better compatibility!
2. Now transform your lesson using the following XSLT template file: `core/presentation/mobile/elml.xsl`
3. Open the file `myproject/mylesson/mylanguage/index.html` in a smartphone and you are redirected to the mobile version. Try it out here: **http://www.elml.org/website/en/**.

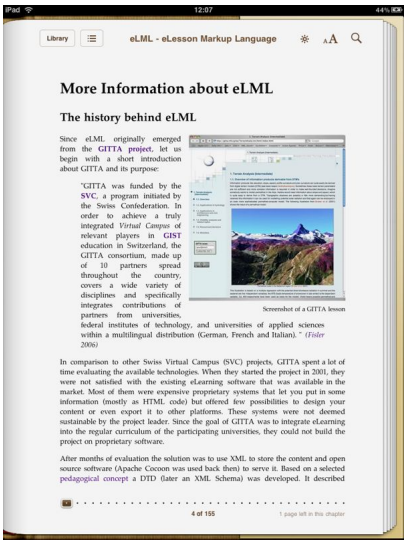
Important Comments and Remarks

- Please create the desktop version first and then create the mobile version. Only then the starting page of your lesson (index.html - see above) will contain a script that redirects mobile users to the mobile version!

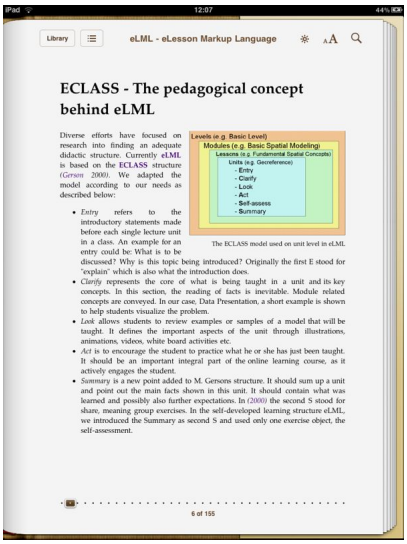
- Within a learningObject all subelements clarify, look and act will be transformed as a "tabbed navigation" element. Please use the "title" attribute to give those tabs a name!
- **Glossary terms** are displayed as popup windows in the mobile version. See screenshot above for an example!
- The back-button in the top right brings the user always back to the desktop version of the current page (follow instructions above to make sure that you have a desktop version!).
- In certain browsers the selfCheck element does not work properly. There seems to be a javascript conflict but we didnt find a solution. Please **contact us** if you find a bugfix or be patient until we found one :-) Sorry about that...

Creating eBooks with eLML using ePub Format

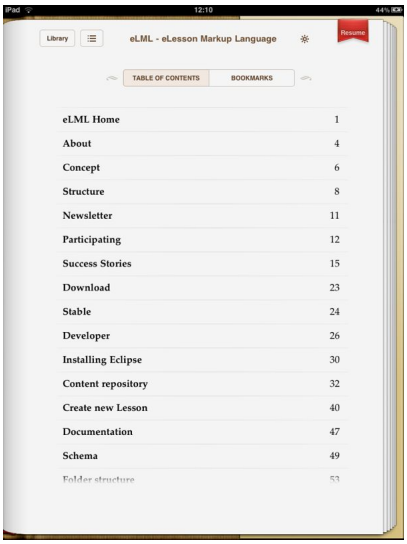
The *ePub-Format* is an XML-based packaging format for eBooks, like **SCORM or IMS CP** but with a differnt manifest file. It is currently supported by most popular eBook readers (except for Amazon Kindle!) and it became even more popular with Apple's announcement to support it on its iPad. Therefore in 2010 the eLML team decided to create an ePub-converter for eLML.



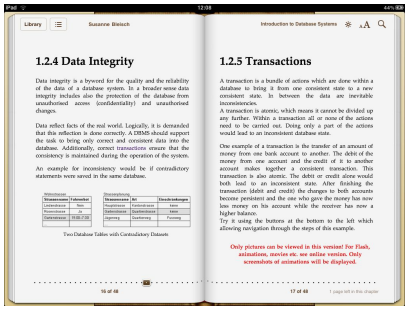
GITTA lesson as eBook on a iPad



eLML website as eBook on a iPad



iPad-eBook table of content



The iPad allows also a two-page view

If you are interested in the ePub standard, you should check out the **ePub-Wikipedia entry** and then head over to **openebook.org** for the full standard specification. For an easy introduction into creating eBooks we recommend having a look at **this ePub-tutorial**. The ePub-standard is based on XHTML 1.1 Strict and therefore the same **configuration settings** as when you would transform your lesson into plain HTML apply. Some settings like the **HTML-Version** or the prohibition of JavaScripts like **lightwindow** are already set in the transformation file and cannot be changed (thus if you change those values in the **configuration file**, eLML will simply ignore it). Please note that the default layout is white background with black default font. If you want, you can also **create your custom layout** similar as you do it with your online version.

How to create an ePub file?

1. Open your lesson or the "Introduction to Database Systems" lesson XML file in oXygen (or XMLSpy).
2. In oXygen click on the "Apply transformation scenario" button. Since you didn't yet define a "Transformation Scenario" for ePub-transformation you will need to do this now.

3. Choose the file `../../../../core/presentation/epub/elml.xsl` file as "XSL URL" XSLT file.
4. Zip your whole folder myproject/mylesson/language/, make sure that the folders "html", "images", "multimedia", "META-INF", the mimetype file etc. are in the root of your ZIP file and name it "mylesson.epub"! If you want to make it super-correct, create the ZIP file with "mimetype" as first file (see example below) but most eBook-readers dont care about this.

Example using Shell/Terminal-Commands (Step 4 from list above):
To create an eBook of the whole eLML website I use the following terminal commands after having transformed the file:

```
cd elml/website/en/
zip elmlwebsite.epub mimetype
zip -r elmlwebsite.epub content.opf toc.ncx META-INF/ html/ image/
 multimedia/
```

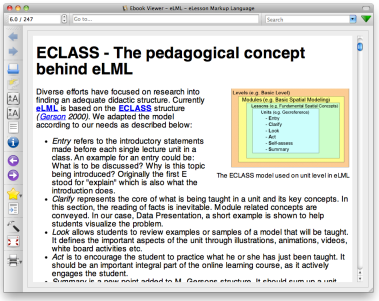
As mentioned above, you can also ignore the two-step zipping because most browser dont care (only **epubcheck** will regonize the difference) and simply ZIP using the following command:

```
zip -r elmlwebsite.epub *
```

Compatibility comments
Please considering the following comments when creating eBooks:

- Currently only JPG, PNG, GIF and SVG are supported. Any other media files (movies, animations etc.) are not yet supported by eBook-readers like the iPad and are therefore ignored.
- The ePub-format is based on **XHTML 1.1 Strict code** and will not work with the older XHTML 1.0 Transition code.
- In ePub no Java Script and no forms are allowed: The scripts for **glossary terms**, **label-references**, **lightwindow** and **selfCheck-assessments** are therefore disabled in ePub.
- We recommend using creating many single HTMl-pages and thus setting the $pagebreak_level=lo in your **configuration file**.
- Columns cause problems in iPads and other readers. Therefore by default they are disabled and columns are displayed one after another. If you want to enable them, set $display_columns to "yes" in your XSLT file.
- If you want to customize the way your eBook looks, create your own templates, similar to the way you created **your own online template**.
- Only a single lesson can be transformed into an eBook, the **creation of courses** is not yet supported (but will be in future releses)

eBook-readers and software



The eLML-website-eBook viewed in Calibre

eBooks can be viewed with any regular computer or with special devices like eBook-readers or smartphones. Here is an overview:

- **eBook-Readers (Hardware)**: Apple iPad, Amazon Kindle, Sony Reader, ...
- Smartphones with eBook-reader software: Apple iPhone, Android-Phones, Blackberry, Palm, ...
- Webbased eBook-readers: **EPUBReader-Firefox-Plugin**, **Bookworm**, **BookGlutton**
- Mac OSX Software: Calibre, Reader Library, Stanza, Adobe Digital Editions, ...
- Windows: Adobe Digital Editions, Calibre, Stanza, Mobipocket, ...
- Linux: Calibre, FBReader, ...

More Hard- and Software for eBooks are popping up every week. You might want to **check this list** or consult Wikipedia or Google for up-to-date information.

From XML to PDF via LaTeX

LaTeX

Before reading this chapter have a look at **the LaTeX (PDF) version of the eLML website**. For more information about LaTeX: here's the link to the **official LaTeX website**.

LaTeX is a typesetting document markup language and provides a set of macros essentially based on TeX which was founded 1977 by Donald Knuth. The idea of TeX (and LaTeX) was to focus the author's work onto the content, without the necessity to deal with the visual representation of his writing. Originally thought as an environment for scientists to typeset complicated formulas on a non-layouting system, it grew to an platform independent and perfectly scaling instrument for publishing journals and books.

LaTeX is able to handle list of figures and tables dynamically, gives support for footnotes and bibliographic citations and generates table of contents and indexes automatically. There are a lot of packages (macros) available to extend the basic functionality of LaTeX, all of them searchable in an internet archive called **CTAN**.

LaTeX can be typesetted using any vanilla text editor. The .tex document the gets processed by the LaTeX/TeX system to an intermediary output file format called DVI ("DeVice Independent" file format). From there it the author produce files in PostScript or PDF format. TeX/LaTeX systems exists for many platforms including MS Windows, Linux, Mac OS.

BibTeX

BibTeX is used to organize the bibliography of the lesson in LaTeX and to produce the correct linking between citation and bibliography entry. This is achieved by extracting every citation in the document and associating it to the according entry in the bibliography database (ie. the .bib file). The .bib file is automatically generated during the XML->LaTeX transformation based on the entries under the bibliography element.

Installation of the TeX System

Windows

- Download **proTeXt** from **here** and follow the installation wizard's instructions.

Mac OS X

- Download **MacTeX** from **here** as a Installation image.
- The package manager will install the TeX utility programs under /Applications/TeX and the unix binaries under /usr/texbin (will be added to the PATH environment variable).

If you are working with Eclipse we recommend using a plugin like **Texlipse**.

Transformation from XML to LaTeX

1. Open the "Introduction to Database Systems" lesson XML file in oXygen or XMLSpy.
2. Have a look at the "latex" section of your projects **configuration file**. At the moment you can only define the $documentclass parameter (choose 'article' or 'book') but more parameters might follow.
3. Choose the file `../../../../core/presentation/latex/elml.xsl` file as input XSLT file (in oXygen you also have to define an output folder: enter e.g. `output.txt` but it doesn't really matter since the exact paths for storing files are part of the XSLT 2.0 files anyway).

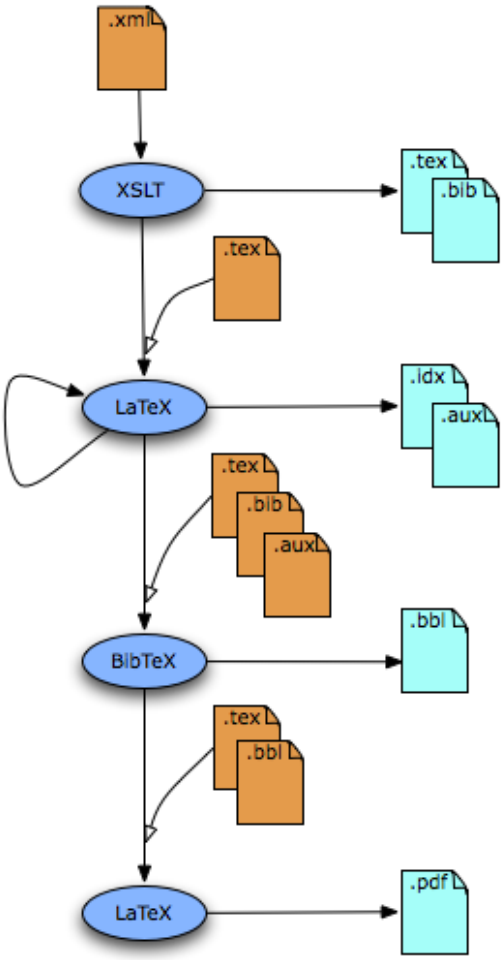
Two output files will be created: *./latex/<lesson label>.tex*, the LaTeX file and *./latex/<lesson label>.bib*, the bibliography file.

Typesetting the tex file

- Open the generated .tex Document in proTeXt/MacTeX, select the correct input format (e.g. LaTeX), typeset it to the preferred output format (e.g. pdflatex); to achieve a correct table of contents and figure/table listing it's recommended to typeset the same document several (e.g. 2 to 3) times.

Building the bibliography index

- After the first LaTeX processing an auxiliary file *<lesson label>.aux* is generated. A subsequent processing of the .tex file in BibTeX builds a bibliography index based on citation marks found in the .aux file and bibliography entries found in the .bib file. The index will be located at the end of the .tex file.
- After BibTeX-ing the .tex file it's necessary to typeset it once again in LaTeX to generate a PDF with a correct bibliography index.



The workflow in eLML from XML via XSLT to LaTeX

Known limitations

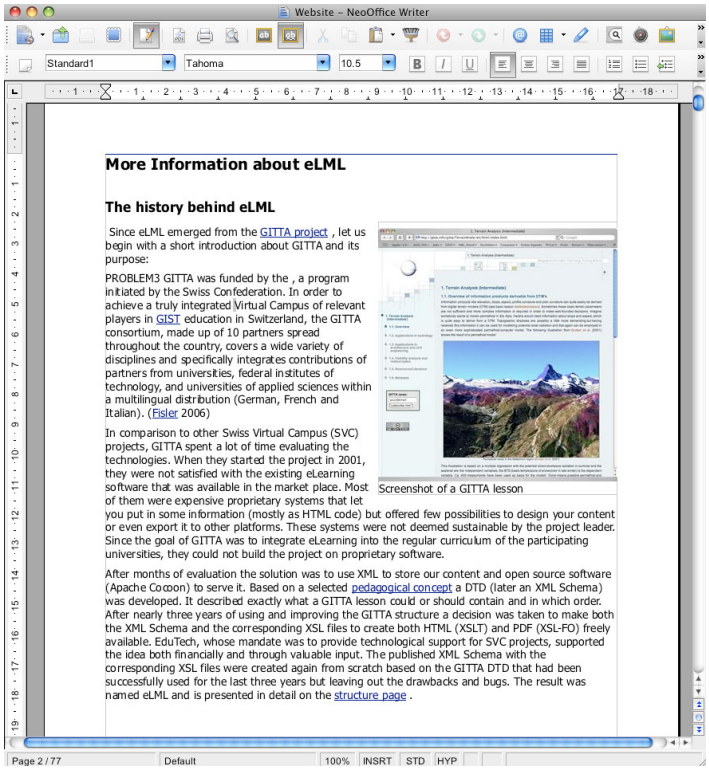
- Images: LaTeX supports PDF, JPG and PNG as inline image formats. GIF isn't supported; these images must first be converted to a supported file format, e.g. by bulk converting them using **GraficConverter**
- Tables: In contrast to tables in html output format, tables in LaTeX format are much more difficult to transform due to the fundamental differences of these technologies in handling tables. It is often impossible to guess the right cell widths for LaTeX tables in the actual context. Furthermore tables width an empty first row or vertical colspans won't transform correctly.

Open Document Format (ODF): Create an office document

 Alberto Sanz has developed an *Open Document Format (ODF)* converter for eLML. An ODF document can be opened both in **OpenOffice** and in Microsoft Office 2007 and 2003 (using a **converter**). The main purpose of this is to allow the transformation of one or multiple eLML lessons into an ODF document. We have tried to include as much multimedia formats as possible and we worked with default styles. For any questions you might have or suggestions, please contact the **eLML support (Alberto Sanz)**.
Because image files in ODF documents do need an information about their size, we suggest that you fill in the values for width and height within your eLML lesson before transforming to ODF.

How to create a ODF version of your lesson

1. Open your eLML lesson or the "Introduction to Database Systems" lesson in Oxygen or XMLSpy.
2. Choose `../../../../core/presentation/odf/elml.xsl` as input XSLT path for the transformation.
3. Once the transformation process is completed then a directory `yourproject/yourlesson/en/odf` is created.
4. Now you have to move every image file of your lesson into `yourproject/yourlesson/en/odf/Pictures`. Please be careful to insert only image files into this folder and not other subdirectories.
5. You are almost all set; only a zip-file of all files and directories within `yourproject/yourlesson/en/odf` must be created. Make sure that you create the ZIP file from within the "odf" folder and that the files and folders are on root-level or the ZIP archive.
6. Rename the created zip-file with the suffix `.odt` and double-click it.
7. Multimedia files such as video and sound are saved relatively to the position of the ODF document within your project folder structure. At the moment there is no possibility to save video or sound directly within the ODF document.



A Screenshot of a eLML-Lesson in ODF

Want to see this eLML website as an ODF-Document? **Download the website.odf file here**!

Two different possibilities to modify the default ODF Styles

The most popular Open Office Applications have a build-in import assistent for ODF Styles. If you want to import styles from an ODT template or an ODT file, apply the following steps:

1. Open the created ODT file in an Open Office Application.
2. Choose Format/Styles and Formatting from the main menue.
3. Click on the little file image on the right in the popup window.
4. Then choose the option "Load Styles..."
5. Select the ODT file or ODT template from wich you want to import the styles.

To overwrite some of the most important default style values in the XSLT file directly please follow this steps:

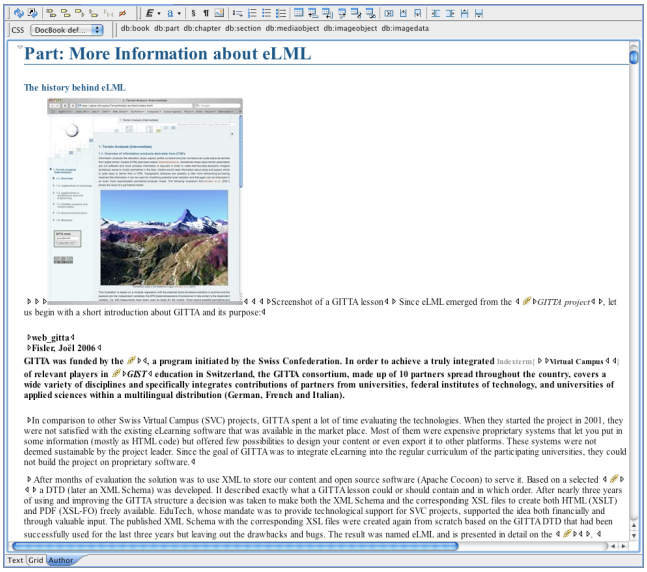
1. **Download the MyProject.zip file** containing an eLML layout template called "plain" within the "_templates" folder. Read the **"Create your own template"** section for more information about customizing eLML layout templates.
2. Open the odf.xsl file and adapt the default values for pagebreak_level, text-align, hyphenation, paragraph_orphans, Headings-size, Headings-margins, etc.
3. Use this file as input XSLT file for the ODF Transformation.

DocBook: Transform your lesson into a DocBook file

As part of the **"Google Summer of Code"** program a *DocBook* transformation file was created by Alberto Sanz. Using this transformation file you can transform your eLML lesson into a valid **DocBook 5.0 document**. DocBook is a semantic markup language for technical documentation that is in widespread use. The DocBook converter is still under development. **Contact Alberto Sanz** if you are using it and have some feedback.

How to create a DocBook version of your lesson

1. Open your eLML lesson or the "Introduction to Database Systems" lesson in oXygen or XMLSpy.
2. Choose the file `../../../../core/presentation/docbook/elml.xsl` as input XSLT file for the transformation.
3. Once the transformation process is completed then a directory `gitta/IntroToDBS/en/docbook` is created.
4. Open the new created file `yourproject/yourlesson/en/docbook/docbook.xml` in a XML or DocBook editor of your choice. In Oxygen you can enter the WYSIWYG-docbook-editor modus by changing from Text to Author view at the left bottom of the application window.



Screenshoot of Visual - a WYSIWYG DocBook Editor

It is important to notice that DocBook documents do not describe what their contents look like, but rather the meaning of those contents. It is up to an external processing tool or application to decide what it should look like.
Therefore to open a DocBook document you can use either an XML editor or one of the commercial or open-source WYSIWYG-Editors. One of the WYSIWYG-Editors that we have tested its called **Visual** and it comes with the oXygen-xml-editor.

Want to see this eLML website as DocBook-Document? **Download the docbook.xml file here**!

Further Links

- **The DoocBook 5.0 Manual**
- **A DocBook Validator**

- **Visual**: DocBook Editor in Oxygen

Tools for eLML

What good is a great framework like eLML if you need to do everything "by hand" and messing around with XML and XSLT? It remains a framework for the experts who know how to work with an XML editor and how to program XSL transformations. Therefore the University of Zurich and other started to create tools for eLML back in 2006 but it took over two years until 2008 the first useable tools were published (all under an open source licence). We divide between tools for editing, presenting or managing eLML lessons.

Edit your lessons with easy-to-use editors

- **Firedocs eLML Editor**: A Mozilla Firefox plugin to edit your eLML lessons in a WYSIWYG-like environment.
- **Open Office Plugin**: Create lessons within Open Office (development discontinued!)

Transformation tool to create various output formats

- **EasyELML**: Transform your eLML lesson into HTML, PDF, ePub and other formats with one mouseclick.

Graphical tools for creating eLML layout templates

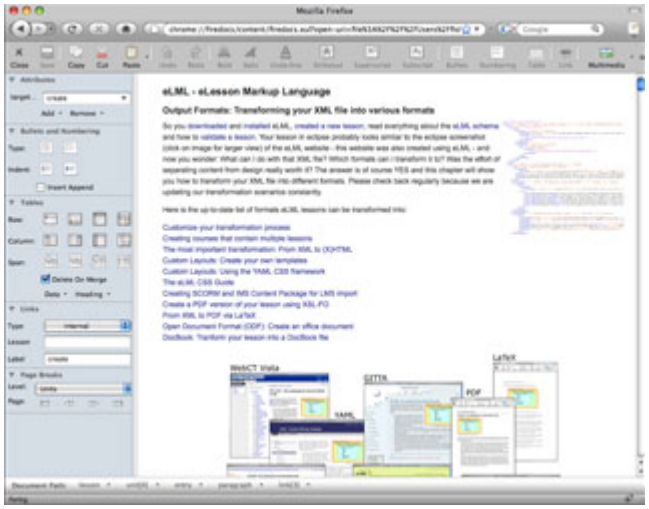
- **Template Builder**: Create eLML layout templates using a simple webbrowser. No XSLT-knowledge needed!

Repository: Storing, managing and versioning

- **AddOn for Lenya and UniCMS**: Import, manage, export and publish your lessons within Apache Lenya CMS or the UniCMS
- **make4eLML Serverscripts**: A set of shell scripts that validate and transform lessons committed e.g. via CVS.

Firedocs eLML Editor

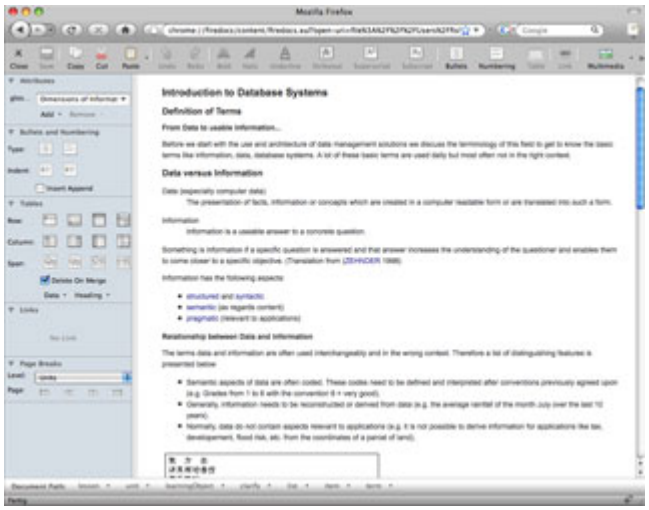
Introduction



The **University of Zurich** has been trying to build an editor for eLML for a long time (nearly since the beginning of the eLML project). After two unsuccessful approaches a third approach based on the Mozilla Firefox Plugin technology led to success. In summer 2008 the first beta version of the *Firedocs eLML Editor* [20] was released.

So what is **Firedocs**? Firedocs is a webbased XML editor for Mozilla Firefox that the University of Zurich has developed for both its Content Management System **UniCMS** and for eLML, the eLesson Markup Language. Both the UniCMS and eLML are XML-based strategic tools of the University of Zurich and needed an easy to use editor. The Firedocs project has now become an autonomous open source project but it offers extensions for both eLML and UniCMS. The editor provided on this website is already compiled containing the eLML extensions you will need to create and edit eLML lessons.

Features



Click on screenshot for large view

- Full featured xml-editor for eLML lessons
- Schema-directed editing and validation. This makes creating invalid lessons almost impossible.
- Works with large lessons: Pagebreak-level can be set to unit or lesson.
- Both standalone (lessons stored on your harddisk) and online (in conjunction with UniCMS repository) editing mode
- Multi-platform
- Auto-completion for term (glossary), citation (bibliography) and link (internal targets) element
- Full support for images, movies and other linked inline media
- Auto save

[20] Firedocs is a Mozilla Firefox plugin with special extensions for eLML. More information about Firedocs can be found on www.firedocs.org. The special "eLML edition" of the XML editor can only be downloaded through the eLML website.

- **Add-on Development**: Easily scriptable/extendable by using javascript/xul

For a full list of the Firedocs-features have a look at the **Firedocs-website**. Please note that the eLML-specific features are not listed there.

System Requirements

Firedocs is Mozilla Firefox plugin and therefore runs on every **platform supported by Mozilla Firefox**. Check the following things before installing:

- Firefox 4 or above!
- Java 1.6 (Java SE 6 Update 10) or above
- Java-Plugin installed and activated in Firefox (under "Extras:add-ons:Plugins")
- JavaScript and Cookies activated in Firefox (under "Settings")

Download and Installation

Firedocs is included in the tools-folder of the **stable** release in the eLML-core. But you can also download the ZIP file directly (see below). The installation of the "Firedocs eLML Editor" is quick and easy. If you ever installed a Mozilla Firefox plugin you know the procedure:

1. Get the **Firedocs eLML Editor**
2. Drag and drop the downloaded file `firedocs-elml-x.xpi` on a Firefox window and click "install".
3. Restart Mozilla Firefox
4. You will notice a new icon (notepad with pen) to the right of the browser bar. Click it and choose " File --> Open..." to open your existing eLML lessons XML file (or use the GITTA demolesson included in the stable release).

Documentation

There is no specific documentation for the "Firedocs eLML Editor" available yet. But you might want to have a look at the following Firedocs pages:

- **Firedocs-Editor Dokumentation** (German)
- **Technical documentation** (English): API specifications, Add-on development, sample code etc.
- **Was tun bei Problemen?** German troubleshooting page

Support

First of all check the "System Requirements" above (crucial!) and for further help the German **Was tun bei Problemen?** page. This page contains tutorials on how to upgrade your system if the system requirements are not fulfilled. You can also find these tutorials (e.g. how to upgrade Java version etc.) via Google in English or other languages. If that does not help: Check the **eLML support page** for more information about how to proceed. Do you want to have some specific features added to Firedocs? **Contact Thomas Comiotto**, the Firedocs developer for commercial services.

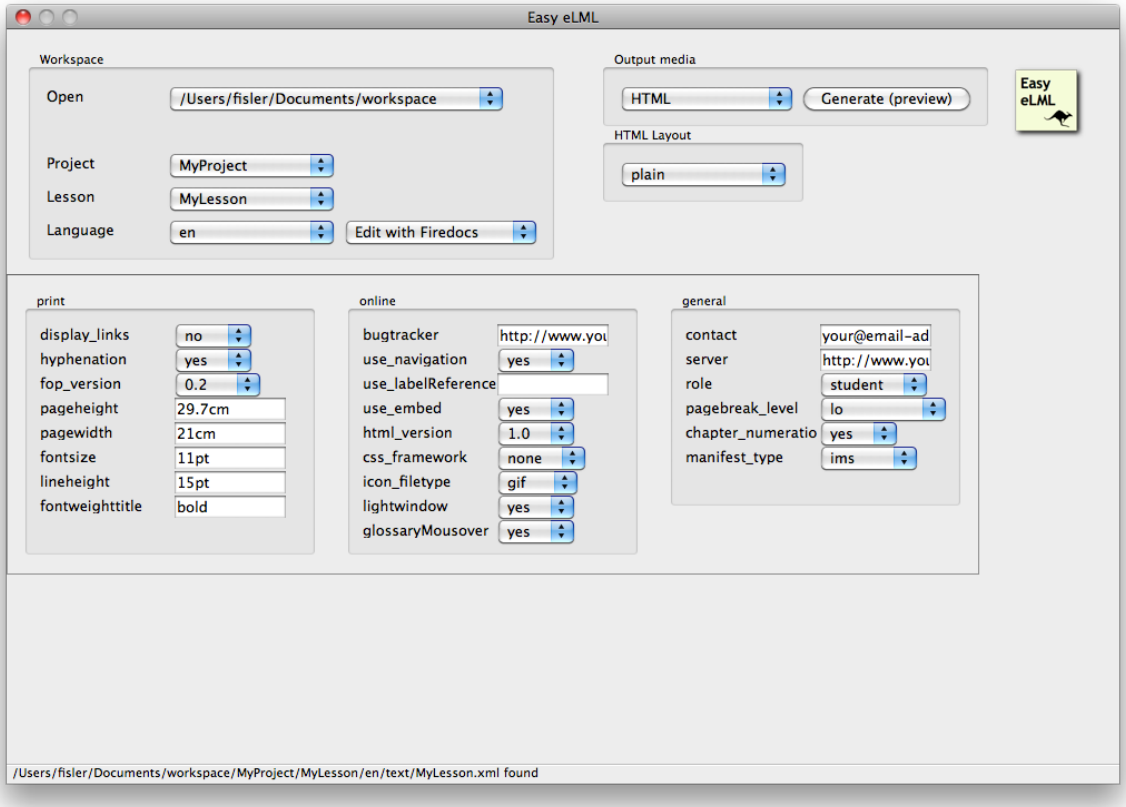
If Firedocs does not start up ("freezes" during startup) you should try to **deactivate the Next Generation Plugin API**. Go to "Extras:Add-ons:Extensions" and click on the Firedocs Workspace Settings to deactive it! Another thing you can try is to clear the XSLT and Schema Cache. You can find these buttons also in the Firedocs settings in the tab "Editor".

If Firedocs messes up your encoding in your XML file, then you should make sure that you have set Firefox to UTF8 encoding. Open the Firefox Settings, to the the tab "Content". Click on "Advances" und "Fonts and Colors" and there choose the character encoding "Unicode (UTF-8)". Now it should work!
Found a bug? No tool is bug-free, we know that. If you found a bug, please **submit it using our bugreport**. Make sure that your bug is not already submitted and that (if it isn't) you choose the category "Firedocs eLML Editor" in the bugreport-form. You can also contact us on the eLML support address: **elml@id.uzh.ch**.

Easy eLML - Lesson Converter

Introduction

Easy eLML provides an easy (sic!) to use crossplatform compatible interface to the various tools, XML parsers, XSLT transformers, which serve behind the scenes of eLML to produce an e-learning lesson. The main objective of Easy eLML is to make the installation and setup process of eLML for end users as straightforward as possible. So among other things, it's not mandatory anymore to have eclipse and/or an XML editor installed to produce a valid e-learning lesson out of a XML file. Instead open source editions of different parsers/transformers are batteries included. Easy eLML remembers user decisions and workspace locations, and allows a centralized management of the various other helpers in the eLML framework as for example "Firedocs Editor". Easy eLML should therefore not be seen as yet another tool for eLML but as an inegrative platform for the existing toolset.



Screenshot of the Easy eLML menu - click for large version

Features

- Generates the various output formats (html, pdf, scorm/ims cp, epub) of an existing eLML lesson per one button click. No need to install any third party software.
- Integration of Firedocs eLML Editor and Text Editor. No need to search for specific lesson on the file System.
- Bookmarks of recent work spaces. No need to remember their locations.
- Build a eLML lesson from scratch. Downloads all needed files and data from the offical repository and let's you start with a brand new eLML lesson right ahead.

- Adjust all print, online and general settings of an eLML lesson in a handy gui.

System Requirements
Easy eLML has been tested on: MacOS X (10.4-10.6), Win XP and Windows 7.

Download and Installation
Easy eLML is included in the tools-folder of the **stable** release in the eLML-core. But you can also download the latest version **directly from sourceforge**!

Installation on MacOS X

1. Double-Click the image file.
2. Move the containing application to some place on your hard drive.

Installation on Windows

1. Double-Click the installer file "EasyElmlInstaller.exe" and follow the instructions.

Documentation
In the download package we have included a "ReadMe" file that contains a quickstart guide and some additional information.

Support
Contact us via the **eLML support page** any questions, concerns or found a bug. You can also **contact the developer Michael Ziege** directly if you have a specific question.

Template Builder

Introduction

As part of the **"Google Summer of Code"** program a Template Builder was created by Thomas Linowsky. The Template Builder is a tool that creates templates in a Web environment (browser) for eLML without you having to deal with XSLT or CSS. These templates are needed to transform your eLML lessons into one of the available **output formats**. Thomas Linowsky is currently working a newly rebuilt version of his Template Builder. **Contact him directly** if you have questions about that.

Features

- Create eLML layout templates in an easy to use WYSIWYG environment
- Multi-platform: runs on most operating systems and browsers (see below)
- Supports both HTML and PDF layout templates
- Preview your lesson directly within the Template Builder
- Manage (save/open) different layout template projects
- Multi-language (at the moment: English and German)

System Requirements

The tool was tested using the following browsers and OS:

- Firefox 2 on Windows XP
- Internet Explorer 7 on Windows XP
- Firefox 2 on Mac OSX
- Firefox 3 on Mac OSX
- Safari 3 on Mac OSX
- Firefox 3 on Linux (Ubuntu 8.04)

The Template Builder requires **YAML installed within your "_templates" folder** (at least version 2).

Download

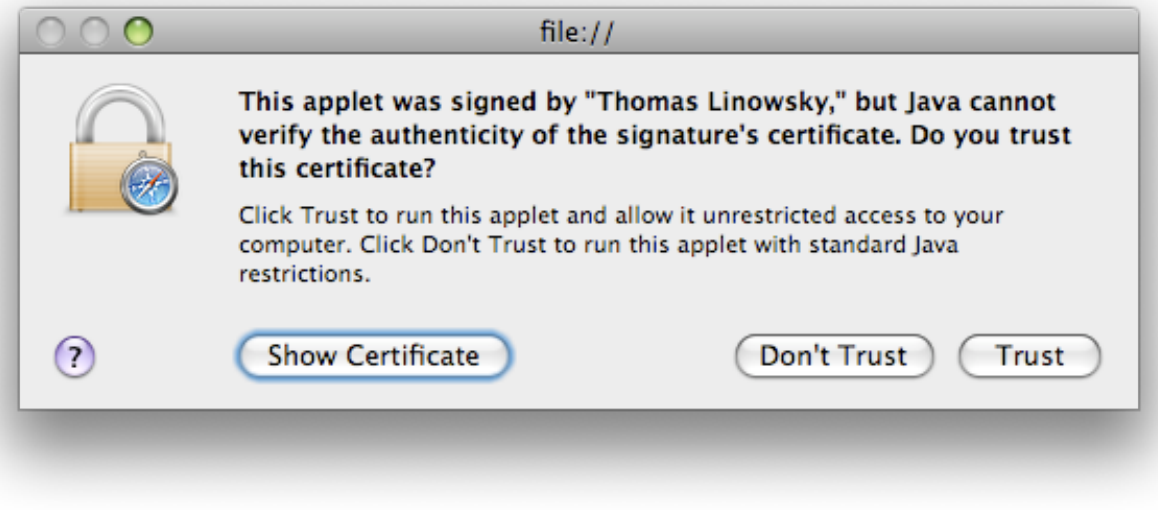
We offer both a stable and a developer release (same as with eLML itself). Both versions are included in the tools-folder of both the **stable** and the **developer** release the eLML-core. You will also have to download YAML since the Template Builder is based on the YAML CSS-framework. If you want you can also download them directly:

1. **Download Template Builder**
2. **Download YAML** and put only the folder called "yaml" into your "_templates" folder. Read more about **YAML and eLML here**.

Installation

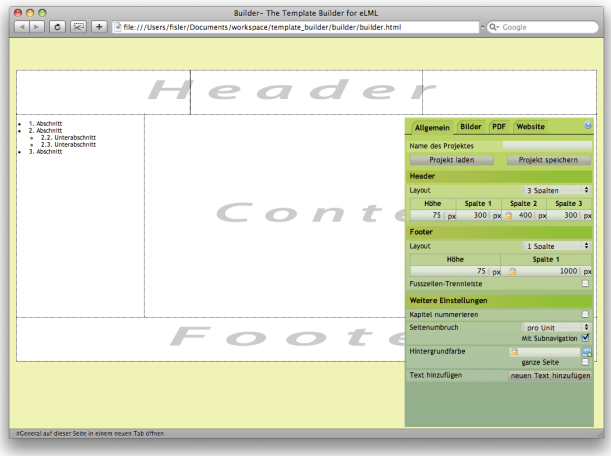
Once you downloaded and uzipped the Template Builder you should store the `template_builder` folder within your workspace. It does not really matter where the folder is located on your harddisk but we recommend to put it within your workspace where your "core" and project folder is located.
The tool can be accessed through the HTML-Page "`template_builder/builder/builder.html`". When you open the page you will be asked if this page is confidential and if you would like to trust it. Please click on "trust" otherwise you will not be able to save your templates, and in certain

cases, you might not be able to even start the tool. You can also click ok and ignore the note informing you that the application might be slow (if this warning should appear, depends on your hardware). In spite of that, the Builder should function properly.



Trust the certificate to be able to use the Template Builder

Documentation



Screenshot of the Template Builder shown in Safari Webbrowser

The Template Builder consists of two main parts:

1. The preview page that represents the appearance of the final layout.
2. The control section called "builder" where you can modify the settings.

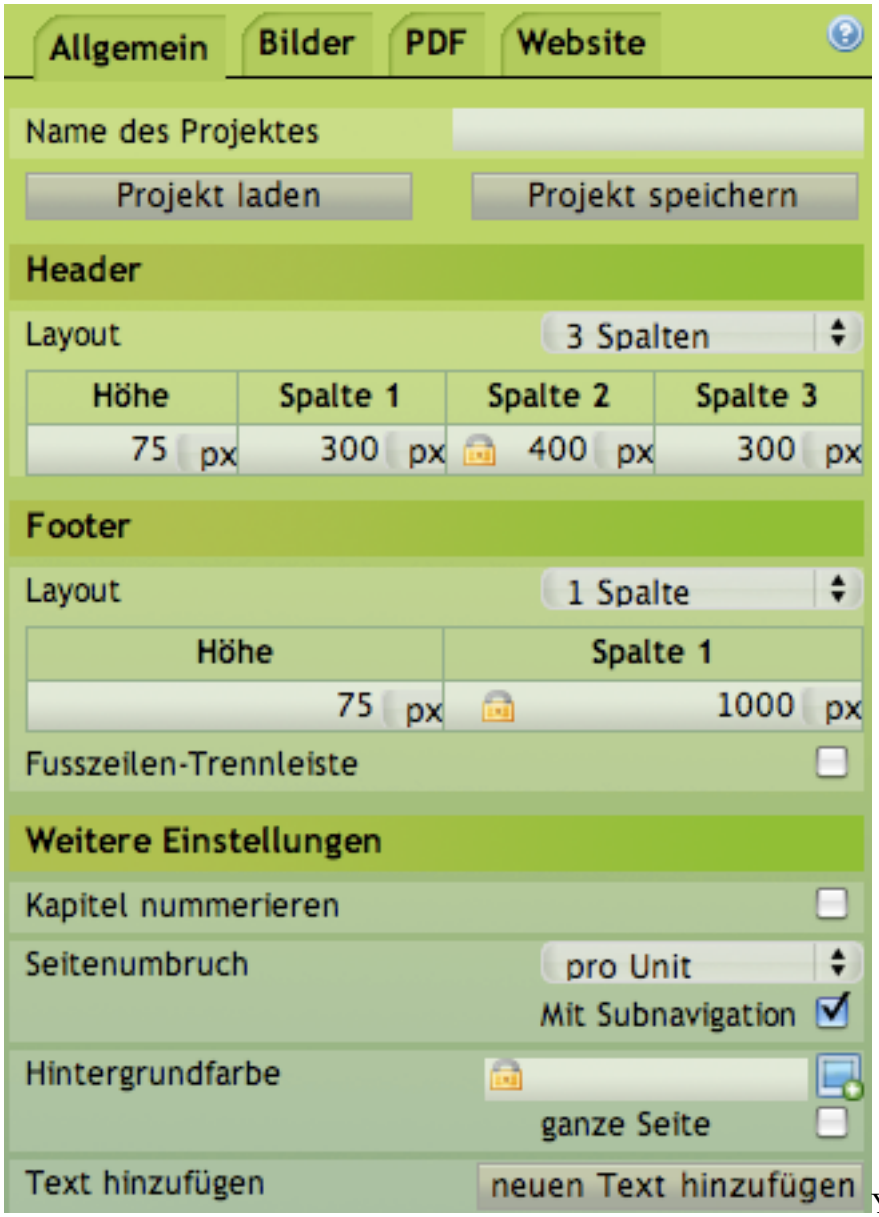
The Preview Page

The background can also be seen in the preview page. The default background is the same as the eLML-layout but you can always modify it. The theree main parts header, content and footer are visible by default. If you dont need a header or footer you can select 0 in the height settings under the “General” tab. The default settings of the area containing these three parts are:

- Width: 1000 pixel
- Height: 768 pixel

The height of the final layout will be adjusted to the content automatically later on.

The Builder



You can move around the "builder" window according to your needs. The control section consists of four tabs:

Tab General

In the "General" tab you can modify the general settings of the layout template:

- *Project settings*: In the field “Name of the project” you can write a name for your template. Please be careful: Templates with the same name will be overwritten! With a click on “Save project” you can save your project and later load it using "Load project" to continue editing.

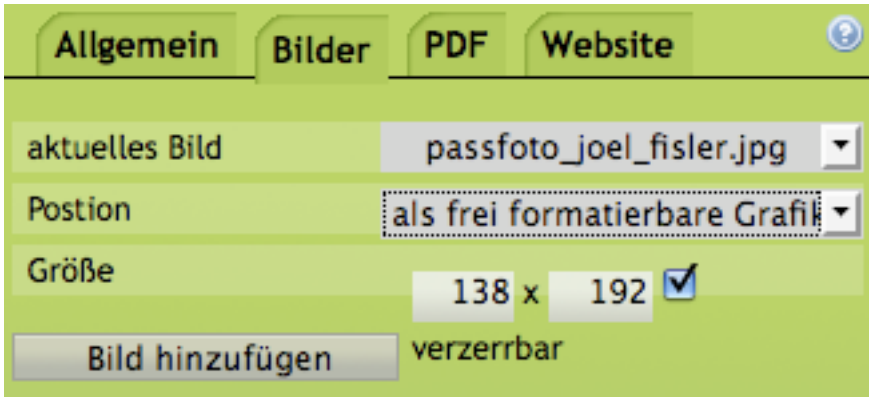
- *Header and Footer settings*: Here you can set up the number of columns of the header as well as adjust the heights and widths. The width of the column can also be adjusted using the dashed lines in the header. By modifying the number of columns all the adjustments in the header will be lost; therefore, you are advised to first set up the number of columns before you proceed with the rest of the settings. Columns that cannot be directly adjusted are write-protected (lock symbol). If you would like to change these, then you must change the other columns first.
- *Additional settings*: Here you can adjust settings such as the chapter numeration or pagebreal level (overriding the **configuration file**!). In the field “Background color” you will notice the name of the actual element you have chosen by clicking on the preview page. That allows you to select elements on the preview page where you want to modify the background color. The checkbox “Whole page” marks the whole page but not the background. First you have to select an element by clicking on it in the preview window. Then click on the icon at the side of the field “Background color” and a table with colors will open. A click on one of these colors will change the background color for the selected element. If you click on the “Add text” icon a editable text element will appear on the preview page. This element is represented in the final layout always relative to the clicked position on the preview-page. The generated element can be modified in terms of the width and length thereby changing its font size.

Access the other tabs by clicking on the tab-name (Pictures, PDF or Website).

Tab Pictures

In this tab you can insert pictures, modify their sizes and move them accordingly.

Actual image: By clicking on the picture the settings will be loaded. The file name in the field “Actual image” illustrates the picture that is being selected.



Insert picture: By clicking on the button you can select the picture you would like to insert. Once the picture has been loaded, you will be able to see on the left side of the window of the builder the actual size of the picture. Now you can drag the picture in the preview element of your choice in order to insert it as the background. The chosen element will appear as it is being dragged with a blue background color.

Background picture: Pictures that are inserted through drag and drop will be displayed as standard background pictures. They will be illustrated according to their actual width and height of the preview page element. If they are smaller than the element, then they will be repeatedly displayed. All of this can be modified with help of the settings fields that appear in the builder.

Floating pictures: If you wish to have floating pictures instead of background pictures then you can set this up by clicking on the field “Position” on the builder. After selecting this possibility you will have several other setting possibilities, such as adjusting the length or the width of the image. You can adjust the length or the width of the picture. It is also possible to maintain always the same scaling. By clicking on the “Position” field you will also be able to re-insert the picture in the background.
Deleting pictures: By pressing “Delete” or by using the combination “Alt+Delete” you can delete the picture you selected.

Tab PDF

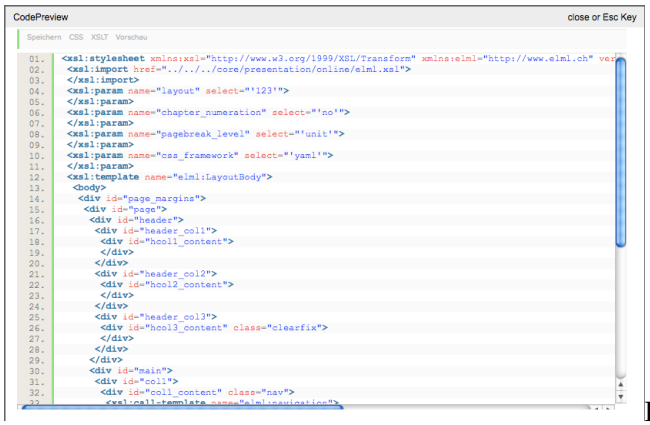
The PDF tab lets you ajust the settings of your PDF layout. It is still in beta status and not "bullet-proof" but you might want to try it out and and let us know if it works.

Tab Website

In this tab you can change those settings that are especially and/or are only used when creating HTML layout templates (online.xsl).
Total size of the preview page: Set the width and margins of the preview page. Additional empty space is always filled with the background color defined under "General". The values can be inserted either in pixel (px) or in font size (em).
Navigation: Set the position and the design of the navigation. If “Up” or “Floating” are chosen the Template Builder adds a default design to the navigation. The design can of course also be modified.
Content section: Depending on the content a minimum width may be required. You can specify it in this section. Other values that are affected due to this change are corrected automatically.

The code preview window

After creating the templates a preview of the templates and CSS-Files will be shown. These can be manually copied, edited and saved. By clicking on the Menu-list you can change your view between the XSLT and the CSS Preview.



By clicking on the “Save” button you can save both documents in the folder with the template names. Please be careful: Previously saved templates or files with the names “online.xsl” or “elml.css” will be overwritten. By clicking on the “Close” button you will close the code preview and the preview page will be shown again.
Click on “Preview” and you can choose a lesson that will be transformed with the help of your created template.

Support

The author of the Template Builder is wokring a totally new version so he offers only limited support for the old version:

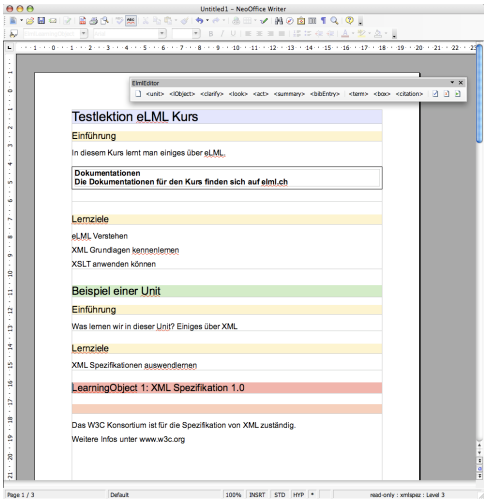
- Need help? **Contact Thomas Linowsky**
- Found a bug? **Submit a bugreport** (choose "Template Builder" as category)

Open Office Plugin

Introduction and Features

In the days before the **Firedocs eLML Editor** the procedure to create an eLML lesson was: Poofessor or tutor writes Word document and - together with some pictures and multimedia files - hands them over to an author that copies the content into an XML editor. In 2006 informatics-student André Locher thought that this workflow could be a lot easier if the tutor would use Open Office instead of Word since Open Office was using XML as a format to store docments. So he started to work on a plugin for Open Office that would allow the creation of eLML lesson directly within Open Office. Many eLML-content element like table, list, column or formatted (bold, italic etc.) were already available as buttons or menu items within Open Office and all he had to do was to "map" them. For structure elements like clarify, look or act the plugin offers a toolbar to insert them.

But there are drawbacks using Open Office: There is no way to validate your content. Basically a tutor can create an eLML lesson that is not valid at all and thus wont transform correctly into HTML or PDF. The plugin can also be used by someone who is familiar with the eLML structure and does not need validation.



The Plugin highlights elements with colors

Development of Open Office Plugin discontinued

The development of the Open Office plugin was started as a students thesis in 2006. The University of Zurich discontinued the development of this plugin and focused on the more promising **Firedocs eLML Editor** instead. We do provide the plugin including the source code "as is" and would welcome any Java programmers willing to continue working on this plugin. In 2010 a student at the Luebeck University of Applied Sciences has continued the development. If you are interest in the latest status of their project contact the project coordinator **Andreas Wittke**.

System Requirements

The eLML OpenOffice plugin has been successfully tested on the following platforms:

- Windows XP Professional
- Mac OS X
- Fedore Core 4
- Gentoo
- Suse Linux 9.3
- Kubuntu 6.10

For the plugin to run on your machine, you need a working installation of **Open Office** or **Neo Office** (OSX). The plugin was thoroughly tested on Open Office/NeoOffice version 2.0.x but should also run on later versions.

Download and Installation
The Open Office Plugin is included in the tools-folder of both the **stable** and the **developer** release the eLML-core. You can also download the ZIP file directly:

1. Get the **eLML Open Office Binary**. If choose to **download the source code** you will have to compile it yourself.
2. Unzip the file and read the `help.pdf` file for exact instructions on how to install the plugin.

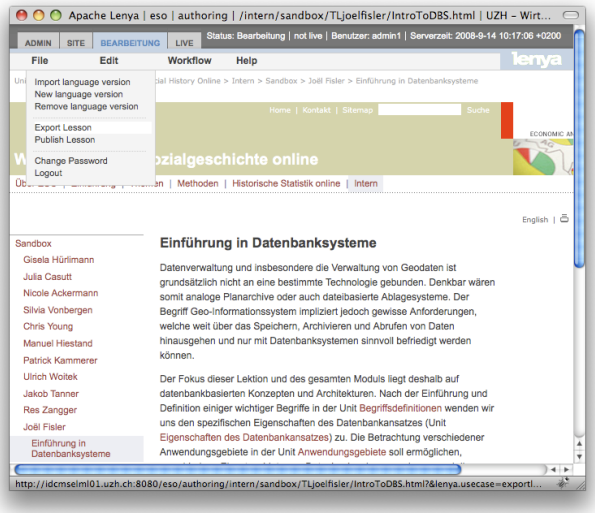
Documentation
The documentation in English is included in the package.

Support
There is no support for the Open Office plugin available because it is dicontinued. Check the comment in the box above for more information.

AddOn for Lenya CMS and UniCMS

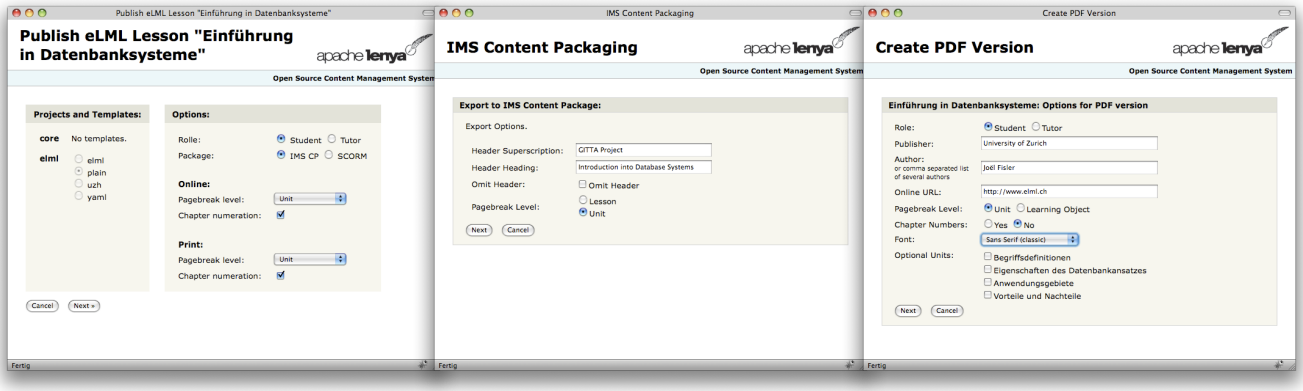
Introduction

The University of Zurich uses an **Apache Lenya** derivate called **UniCMS** as their strategic content management system. Since this content management is used all over the University we also wanted to use it as repository for eLML e-learning lessons. An external company (see under "Support") created an AddOn that allows the import, management, publishing and export of eLML lessons within Lenya or the UniCMS. If you need a webbased repository solution for storing e-learning lessons, this might be your answer :-)



Features

- Import eLML lessons into Lenya or UniCMS
- Store and manage eLML lessons within Lenya or UniCMS
- Multi-Stage Online-Publishing of eLML lessons in different layouts
- Export eLML lessons as HTML/IMS CP, PDF or eLML-XML
- Configure the export format according to your needs (see screenshots below)



Publish lessons choosing a layout template *Export lessons as IMS Content Package* *Export lessons as PDF (HTML)*

System Requirements

You will need either **Apache Lenya** content management system or its derivate **UniCMS** before installing the add ons.

Download and Installation
The eLML-AddOn for Lenya/UniCMS is open source and freely available. But since this tool will only be of interest for a very limited amount of people we dont create and publish different releases. If you are interested in this tool just **send us an email** and we will send you the latest release with some instructions.

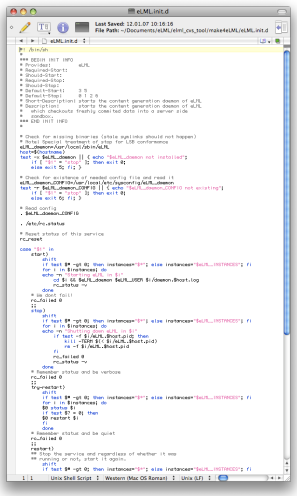
Documentation
A technical documentation is included.

Support
The eLML-AddOn for Lenya/UniCMS was programmed by the **BeCompany GmbH**. Feel free to **contact them** if you want commercial support, new features or extend the existing features. For other inquiries about this or other eLML tools **contact the eLML-team** at the University of Zurich.

make4eLML Serverscripts

Introduction

Using a **content management system with an eLML-AddOn** is one way of storing and management your eLML lessons. Before this solution was available all the eLML projects have been stored and managed using a CVS - *Concurrent Versions System* - server. The University of Zurich is still using and maintaining this **repository server** and it has been enhanced with a collection of smart UNIX shell scripts called "make4eLML" by Timon Roth. The script is executed as soon as the user commits (stores) an updated version of an eLML lesson on the repository server. It validates the lessons, transforms it into all available **output formats** and uploads the results on a project (FTP or SSH) server defined by the project manager. In case of failure (e.g. due to validation errors) the project manager gets automatically notified via email.



Features

- Daemons check if updates of lessons have been committed
- Automatic transformation of lessons into HTML, CP, SCORM, PDF, LaTeX and ODF
- Upload of resulting files via FTP or SSH on a project server defined by project manager
- Automatic notification via email in case of failure or errors

System Requirements

You need a Unix or Linux operating system to run make4eLML. The script collection probably also runs on other operating systems but you will have to adapt the script.

Download and Installation

The make4eLML script is included in the tools-folder of both the **stable** and the **developer** release the eLML-core. You can also download the ZIP file directly:

- **Get it here via Sourceforge**

The installation is described in detail in the included ReadMe file. The file is only available in German at the moment. Please **let us know** if you would be interested in an English version. If you dont install the script in the /usr/local/ path you will have to change the paths accordingly.

Documentation

There is no documentation available besides the very extensive `readme.txt` file included within the package.

Support

We cannot really provide support for this script collection. Although the scripts are in constant use on the University of Zurich's eLML-Linux-server and thus should be working, we cannot help you with making them work on your operating system. Check the **contact page** if you still want to contact us.

eLML support and contact

Since eLML is an open source project, we do not have office opening hours and telephone support. Before contacting our email support, please check the following section on how to submit bugs or feature requests:

- Submit a bug: Check the **Bugtracker** if your bug is already listed, if not please submit it.
- Need a new feature: Submit a **feature request** and/or program the feature yourself.
- Stay informed: We offer **various newsletters** depending on your interests.
- Read the manual: The latest **eLML manual** and schema documentation is included in the eLML package.
- Contact tech support: If the above links did not help feel free to **submit a support request**.

Some of these features only work if you are a registered Sourceforge member. **Becoming a member** is free and we strongly recommend it because this way we will be able to contact you for any further inquiries. Please try to avoid anonymous tracker submissions!

Contact address

Address	Universität Zürich Informatikdienste Multimedia and E-Learning Services MELS Immo Wille Winterthurerstrasse 190 8057 Zürich Switzerland
Phone	+41 44 635 67 44
Email	**elml@id.uzh.ch**
Web	**www.elml.org**

The developers of eLML

In the **Sourceforge Members Page** you will find an up-to-date list of active eLML developers. Many of them are *GITTA* team members or working for the MELS, the **Multimedia and E-Learning Services** of the University of Zurich. The two founders/initiators of eLML are Susanne Bleisch and Joël Fisler. The developers of the different tools and transformations are named in the according chapters. Here's a short overview of past team members:

- Susanne Bleisch (2001-2012): Founder
- **Joël Fisler** (2003-2012): Founder
- Michael Ziege (2005-2011): EasyELML, LaTeX, make4elml
- Timon Roth (2005-2007): make4elml
- Rolf Brugger (2001-2004): Concept&Ideas, Funding
- Marion Werner (2002-2005): Design
- Hans-Jörg Zuberbühler (2005-2011): Project Lead UZH
- Student projects: Alberto Sanz, Thomas Linowsky, Andre Locher

People involved (current or past) with the eLML project

 *Susanne Bleisch (FH-NW)*	 *Joël Fisler (UZH)*	 *Immo Wille (UZH)*	 *Alberto Sanz (UZH)*	 *Michael Ziege (Juleg)*
 *Thomas Comiotto (UZH)*	 *Thomas Linowsky (TU Chemnitz)*	 *Timon Roth*	 *Olaf Schnabel (ETHZ)*	 *Andreas Zangger (UZH)*
 *Hansjörg Zuberbühler*	 *Caspar Nötzli (PHZH)*	 *Franziska Schneider (UZH)*	 *André Locher (UZH)*	 *Frank Mäder (UZH)*
 *Sandra Roth (UZH)*	 *Renata Sevcikova (UZH)*	 *Kristina Isacson (UZH)*	 *Roman von Wartburg (IFEL)*	 *Lukas Stähli (UZH)*

You're missing on this list? We probably dont have your picture. **Send it via email**.

Copyright

Project hosting

The eLML project is hosted at Sourceforge. Sourceforge is a free source code repository for open source projects.

eLML Search

Please use our personalized Google search to find documents on elml.org and Sourceforge.

Glossary

Creative Commons:

Creative Commons (CC) is a nonprofit organization that offers flexible copyright licenses for creative works. CC allows authors of music, films, photos, texts etc. to share their work under a specific license they can define on the **creative commons website**. In the case of *GITTA*: You may use the GITTA content for non-commercial purposes as long as you cite GITTA as the author (including a link to the website) and publish your derivatives under the same license. For more information have a look at the **creative commons deed** or at the **full legal code**.

CSS:

Cascading Style Sheets (CSS): A stylesheet language used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can be applied to any kind of XML document. CSS is a **W3C Standard**.

CVS:

CVS, the Concurrent Versions System, is the most widely used tool for controlling different versions of a source code and for a group of programmers to work simultaneously on a source code. Before working with a file, a user needs to do a "checkout" of the file from a "repository" stored on the project server. When writing updates back to the repository (called "committing"), CVS checks issues like access privileges, actual status of code and if no other group member meanwhile altered the code, CVS writes it back to the repository. By doing a "update" all project group members get the latest version of the code. CVS of course stores information about who altered which part of the code and automatically stores different versions of the code. Therefore using CVS it is possible to always reconstruct a former state of the code. *eLML*, and therefore also GITTA, uses CVS to store the XML code, images and multimedia elements of a lesson.

DocBook:

DocBook is a semantic markup language for technical documentation. It was originally intended for writing technical documents related to computer hardware and software but it can be used for any other sort of documentation.

ECLASS:

ECLASS is based on Steven Gersons Guide to develop online courses. It is an abbreviation for the terms E = Entry; C = Clarify; L = Look; A = Act; S = Self-Assessment; S = Summary. Described in detail in the concept chapter. (Gerson 2000)

eLML:

eLML, the eLesson Markup Language, is an XML framework developed by the GITTA project. The Swiss eLearning project GITTA started working with XML in 2001 but it was only after the official ending of the project in 2004 that its XML structure was released as an open source project under the name of eLML. For more information read the implementation chapter or visit **www.eLML.org**. (Fisler et al. 2005)

ePub:

ePub is an XML based format for electronic books or eBooks. This format is used e.g. bei Apples's iPad. The standards definition can be found on **openebook.org**. More information about creating eBooks with eLML **can be found here**.

firedocs:

Firedocs is a Mozilla Firefox plugin with special extensions for eLML. More information about Firedocs can be found on www.firedocs.org. The special "eLML edition" of the XML editor can only be downloaded through the eLML website.

GIST:

GIST is the abbreviation for Geographic Information Systems Technology.

GITTA:

GITTA is a Swiss eLearning project about GIS and it is the abbreviation for Geographic Information Technology Training Alliance. For more information about GITTA have a look at **www.gitta.info**.

IMS:

The **IMS Global Learning Consortium** (usually known as IMS) is a non-profit standards organization concerned with establishing interoperability for learning systems and learning content and the enterprise integration of these capabilities. Their mission is to "support the adoption and use of learning technology worldwide". Some famous IMS standards are the CP (Content Package) standard used to import/export of content, the "Learning Resource Meta-data Specification" (LOM) or the QTI standard for question and test interoperability.

LaTeX:

This is how **Wikipedia** defines LaTeX: LaTeX is a document markup language and document preparation system for the TeX typesetting program.

It is widely used by mathematicians, scientists, and scholars in academia and the commercial world, and by others as a primary or intermediate format (e.g. translating DocBook and other XML-based formats to PDF) because of the quality of typesetting achievable by TeX. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies.

See the **eLML to LaTeX chapter** for more information.

LMS:

A Learning Management System (or LMS) is a software package, usually on a large scale (that scale is decreasing rapidly), that enables the management and delivery of learning content and resources to students. Most LMS systems are web-based to facilitate "anytime, anywhere" access to learning content and administration. Some famous open source LMS are **OLAT** and **Moodle**, famous commercial LMS are **WebCT** and **Blackboard**.

ODF:

The Open Document Format (ODF) is an open source standard for office documents (text, spreadsheets, presentations etc.). It is used e.g. by OpenOffice or StarOffice and other similar open source tools. Since Microsoft Office 2007 started its own (only partially open...) XML format called "OpenXML" there is a "war of formats" going on because each standard tries to become THE standard for office documents. eLML offers a **ODF converter** because the team thinks that ODF is the better standard and deserves support from the open source community. Read **this comparison** to know more about the differences between ODF and OpenXML. There are many tools around to convert one format into another one and good office tool even support both by default so don't worry :-)

OLAT:

The development of the open source *LMS* OLAT (Online Learning And Training) started at the University of Zurich in 1999 and won the **Medida Prix** for best eLearning software in 2000. Today OLAT is already in its fifth version and is the strategic platform of the University of Zurich. Besides Zurich other universities like Bern, Sachsen (Germany) etc. are using OLAT as their main LMS. More information and download of the software can be found on the **OLAT website**.

SCORM:

The Shareable Content Object Reference Model (SCORM) is a standard for web-based eLearning. It defines how the individual instruction elements are combined at a technical level and sets conditions for the software needed for using the content. SCORM is distributed by the **Advanced Distributed Learning (ADL) Initiative**, a US organization under the Department of Defense (DoD).

Sourceforge:

SourceForge.net is a community website that hosts open source projects. It offers a wide range of services like bugtracker, discussion board, CVS, web-space for working and storing of open source software. We use SourceForge for eLML to host our "core" files.

SVC:

SVC, the Swiss Virtual Campus, was founded in 1999 after a decision of the Swiss Parliament that over 50 Million Swiss Francs should be used to build up eLearning projects at Swiss universities. In the first project phase out of about 200 project drafts a total of 50 projects were accepted and supported. GITTA was one of them. For more information have a look at the **SVC website**.

SVG:

SVG, the Scalable Vector Graphics, is a standard of the **World Wide Web Consortium (W3C)**. It is an open, *XML*-based format to describe graphics and animations and can be used as an alternative to the proprietary Adobe Illustrator and Macromedia Flash formats. To view SVG within a browser use either a modern browser (like **Firefox** or Apple's Safari) that has native support for SVG or download a plugin. For detailed information about SVG **visit the W3C** or read **Wikipedia's explanation**.

XML:

XML, the eXtensible Markup Language, is a standard of the **World Wide Web Consortium (W3C)**. XML documents use elements (tags) known from other markup languages like HTML. Using *XSL transformations* XML files can be transformed into other formats like XHTML or PDF. Many common used languages are based on XML: XHTML, *SVG*, GML, RSS, MathML etc. For detailed information about XML **visit the W3C** or read **Wikipedia's explanation**.

XSLT:

XSLT, the XSL Transformations, is part of the Extensible Stylesheet Language (XSL) family and is a standard of the **World Wide Web Consortium (W3C)**. XSLT files are used to transform XML files into other formats like HTML or formatting objects (FO) for generating PDF files. For detailed information about XSL **visit the W3C** or read **Wikipedia's explanation**.

ZIP:

ZIP is a compression algorithm widely used on most of todays operating systems. A ZIP archive can contain multiple files, folders and subfolders. Both **SCORM and IMS** content packages are actually ZIP files that must contain a valid imsmanifest.xml file on their root level. With MacOS X you can right-mouse-click files or folders in the finder and choose "Create archive of ..." from the context menu. On Windows you have to install a tool like WinZIP to create ZIP archives.

Bibliography

- **Anonymous**, 2010. eLML der Uni Zürich gewinnt Silbermedaille. *e-teaching.org*, 8. Juni.
  Download: http://www.e-teaching.org/news/eteaching_blog/et_showComments?entryid=blogentry.2010-06-08.7097069858
- **Anonymous**, 2010. eLML Wins IMS Learning Impact Award 2010 Silver Medal. *Checkpoint eLearning*, May.
  Download: http://www.checkpoint-elearning.de/?aID=8142
- **alex.gorbatchev**. *Syntax Highlighter* [online]. Available from: http://code.google.com/p/syntaxhighlighter [Accessed 20/04/2008].
- **American Psychological Association** (2007). *APA Style.org* [online]. Available from: http://apastyle.apa.org/ [Accessed 12.7.2007].
- **Behme, H., Mintert, S.**, 2000. *XML in der Praxis – Professionelles Web-Publishing mit der Extensible Markup Language*. München: Addison-Wesley Verlag.
- **Benz, Roman**, 2006. Projekt GITTA am Geografischen Institut: Neue E-Learning-Software, die nicht bloss Fröschen hilft. *unijournal*, No. 3, 8. Mai, 7.
  Download: ../download/publications/unijournal-2006-3.pdf
- **Bleisch, Susanne, Fisler, Joël**, 2005. eLesson Markup Language eLML – eine XML basierte Applikation für die beschreibende Auszeichnung von nachhaltigen und flexiblen e-Learning Inhalten. *In:* Muttenz, Switzerland: Fachhochschule beider Basel (FHBB).
  Download: ../download/publications/DeLFI2005_eLML_Paper.pdf
- **Cody Lindley**. *Thickbox* [online]. Available from: http://jquery.com/demo/thickbox/ [Accessed 02/05/2008].
- **Dirk Jesse** (2008). *YAML* [online]. Available from: http://www.yaml.de [Accessed 01/04/2008].
- **Fisler, Joël** (2006). *The GITTA project website* [online]. Zurich, Switzerland: University of Zurich. Available from: http://www.gitta.info [Accessed 23.02.2006].
- **Fisler, Joël**, 2005. Virtueller Schweizer GIS-Campus. *ArcAktuell*, 1. Oktober, 22-23.
  Download: ../download/publications/arcaktuell_2005_03_campus_gitta.pdf
- **Fisler, Joël** (2007). Erstellung von strukturierten e-Learning Inhalten mit eLML (eLesson Markup Language). *ZInfo - Die elektronische Zeitschrift der Informatikdienste* [online], 23. Available from: http://www.id.uzh.ch/publikationen/zinfo/ [Accessed 1.7.2007].
- **Fisler, Joël**, 2008. Creating Educational Content with eLML. *In:* **Hoffmann, Frank**, ed. *GI2008-Symposium - 8. Sächsisches GIS-Forum, 15./16. Mai 2008, Dresden*. Dresden, Germany: IGN - Innovation. Grenzüberschreitendes Netzwerk, 54.
  Download: http://www.ign-sn.de/GI2008/GI2008_FINAL_Version_Edited_20080731.pdf
- **Fisler, Joël**, 2010. Silver medal for eLML. *Eduhub*, June 3rd.
  Download: http://www.eduhub.ch/news/eduhubnews/2010/news-article-0033.html
- **Fisler, Joël, Bleisch, Susanne**, 2006. eLML, the eLesson Markup Language: Developing sustainable e-Learning Content Using an Open Source XML Framework. *In: WEBIST 2006 - International Conference on Web Information Systems and Technologies, April 11th-13th 2006*. Setubal, Portugal.
  Download: ../download/publications/WEBIST2006_eLML.pdf
- **Fisler, Joël, Bleisch, Susanne, Niederhuber, Monika**, 2005. Development of sustainable e-learning content with the open source eLesson Markup Language eLML. *In: ISPRS Workshop, June 2nd/3rd 2005*. Potsdam, Germany.
  Download: ../download/publications/Potsdam2005_ISPRS_eLML.pdf

- **Fisler, Joël, Bleisch, Susanne, Weibel, Robert**, 2006. Das e-Learning-Projekt GITTA: Frei zugängliche Inhalte für die akademische Ausbildung in Geoinformation. *In:* **Thomas Jekel, Alfons Koller, Josef Strobl**, ed. *Lernen mit Geoinformation (AGIT proceedings - Themenschwerpunkt Geoinformation in der Schule), 5.-7. Juli 2006, Salzburg, Austria*. Wichmann Verlag.
  Download: ../download/publications/agit_gitta.pdf
- **Fisler, Joël, Schneider, Franziska**, 2008. Creating, Handling And Implementing E-Learning Courses and Content Using the Open Source Tools OLAT and eLML at the University of Zurich. *In:* **Shortis, Mark, König, Gerhard**, ed. *ISPRS Conference 2008 -TS ThS-16: New Approaches and Tools for Education and Capacity Building, 3-11 July 2008, Bejing*. Beijing, China: International Society for Photogrammetry and Remote Sensing (ISPRS). [eLML also won the CATCon Gold Award at the ISPRS Conference 2008 in Beijing!]
  Download: ../download/publications/isprs2008_beijing_fisler.pdf
- **Fisler, Joël, Weibel, Robert**, 2006. GITTA: Open Content Material for GIS Education. *In: EUGISES 2006 Conference, September 7th-10th 2006, Krakow*. Krakow, Poland: Faculty of Forestry, Agricultural University.
  Download: ../download/publications/eugises2006.pdf
- **Fuchs, Marita** (2008). Medida-Prix: Drei UZH-Projekte in der Finalistengruppe. *unipublic* [online], 25.08.2008. Available from: http://www.unipublic.uzh.ch [Accessed 27.8.2008].
  Download: http://www.unipublic.unizh.ch/campus/uni-news/2008/3096.html
- **Gerson, Steven M.** (2000). E-CLASS: Creating a Guide to Online Course Development For Distance Learning Faculty. *Online Journal of Distance Learning Administration* [online], Volume 3, Issue 4. Available from: http://www.westga.edu/~distance/ojdla/winter34/gerson34.html [Accessed 20.01.2006].
  Download: ../download/publications/eclass_gerson.pdf
- **Harold, E. R., Means, W. S.**, 2000. *XML in a nutshell – A Desktop Quick Reference*. Sebastopol: O'Reilly & Associates.
- **Horton, W.**, 2000. *Designing Web-Based Training – How to teach anyone anything anywhere anytime*. New York: John Wiley & Sons.
- **John Resig and the jQuery team** (2008). *JQuery Website* [online]. Available from: http://jquery.com [Accessed 06/02/2008].
- **Lütolf, Gregor**, 2006. *Zugänglichkeit von geographischen E-Learning-Kursen für Sehbehinderte und Blinde am Beispiel von GITTA*. Thesis (Master). University of Zurich.
  Download: http://www.gitta.info/website/en/download/gitta/luetolf/gluetolf_diplomarbeit.pdf
- **Peter Rüegg** (2008). Medida-Prix für E-Learning-Projekt: Ausgezeichnete Lernpakete. *ETH Life* [online], 21.10.2008. Available from: http://www.ethlife.ethz.ch [Accessed 11/11/2008].
  Download: http://www.ethlife.ethz.ch/archive_articles/081021_medida_gitta/index
- **Schnabel, O., Stopper, R., Hurni, L.**, 2007. New modular approach for knowledge-transfer in multimedia cartography: The e-learning project CartouCHe. *In: Proceedings of the International Cartographic Conference, Moscow (Russia), Theme 4, Oral 2*. [Chapter 4 talks about eLML, the technique behind the project.]
  Download: http://www.e-cartouche.ch/material/paper_schnabel_stopper_hurni_cartouche.pdf
- **SCILS**, 2004. *Citing References: Harvard System*. United Kingdom: Bournemouth University.
  Download: http://www.bournemouth.ac.uk/library/using/harvard_system.html
- **Weibel, R; Bleisch, S; Nebiker, S; Fisler, J; Grossmann, T; Niederhuber, M; Collet, C; Hurni, L**, 2009. Achieving more sustainable e-learning programs for GIScience. *Geomatica*, 63, 109-118.

Download: https://www.zora.uzh.ch/25589/

- **Werner, M., Bleisch, S., Fisler, J.**, 2005. E-Learning Materials in GIS-Technology and Cartography - Towards an Open-Content Solution. *In: Proceedings of the 22st International Cartographic Conference - Mapping Approaches into a Changing Future, July 9-16, 2005, A Coruña, Spain.*
  Download: ../download/publications/werner_coruna.pdf
- **Zuberbühler, Hans-Jörg**, 2007. Vom E-Learning zum «Blended Learning». *Neue Zürcher Zeitung NZZ*, 9. März.
  Download: ../download/publications/nzz_blended_learning_print.pdf

Index

List of Figures

List of Tables