A five-minute tour how to prepare a SCORM package

Please download the example SCO archive "hpp" from the WP2 shared documents. You should find a "zip" file on your desktop now. All SCORM packages come in a .zip container, to be prepared by winZIP or any other tool that generates compliant zips. I'm using the Linux command line.

Contents of the ZIP

The most important file, and the file that contains the metadata and all the indexing, is the "imsmanifest.xml" file. We don't worry about the metadata yet. However, something still needs to be edited here. A standard text editor (emacs or notepad) will do. You will find in that file a XML tag describing the organizations, organization and title. You may put there whatever you like. The organization is nothing we or the portal cares about, and the title is the title of the experiment.

What is more important is the resource-tag. You find there (in our example) the file "applet.html" as a resource (keep this! It is the web-page to be rendered) and the files "lib/VideoEasel.jar" (the java .jar file for the applet), "lib/Application.xml" (an XML file describing the experiment for VideoEasel) and "lib/lab.dtd" (another XML relevant file for the laboratory which is used to verify the correctness of the former xml file.

You will need "applet.html" always. If you are using java, the jar needs to be delivered as well which contains the java code. Of course, you should remove "VideoEasel.jar" (as you're not using our lab but your lab) and replace it by your own "jar" file (for example "mylab.jar"). Then remove the file from the zip, add your file to the zip under "lib", and edit the reference in the manifest accordingly. If your lab needs additional files to function, put them under "lib" and edit the imsmanifest accordingly. If you do not need a "jar", then no need to add it either. In our example, "VideoEasel.jar", "Application.xml" and "lab.dtd" are all specific to the Stuttgart lab and not of any concern for your application. Just remove them.

The applet.html file

This is the second most important file, and the file that is rendered in the browser to make your experiment happen. Again, you need an editor, preferably an HTML editor to modify it. A standard text editor will also do.

In the specific example case, it renders a java applet, as seen from the "applet" tag. If you want to render any other active content besides java, the easiest way is just to remove the code of the applet tag in the file, open your own personal home page, view the html source you find on your page, locate in that page the html snippet that renders your lab, and paste that directly into "applet.html" at this place.

You may notice that our applet.html example uses some additional java script we leave for later. Basically, it imports a java script file called "lmsstub.js" you also find in the archive, and this stub file is responsible for communicating between the learning management system on one hand, and your html code on the other. This script file allows you to extract, for example, the name of the student running the experiment, and forwarding this into the applet or your application. The code between "<--" and "//-->" is java-script code that is executed to perform this magic, and this code is called once when the page is initially found and delivered to the browser. If you do not need this information, all javascript magic can go, and you should remove the text between the markers, and the "onload" and "onunload" properties of the tag below. Later on, you can use this data to identify the student using the lab - if you have to. I leave the details on this for a more detailed workshop, we don't need it right now.

The morale is the following: It is **usually** enough to just copy the web page rendering your lab into applet.html, to ensure that all the data required to render the web-page is part of the zip, and mentioned in the imsmanifest. Everything else is advanced wizardry we will leave for later, and which we don't really need right now.

Testing

Testing shouldn't be forgotten. To test, just package all the files you have created in a .zip file (using the linux "zip" command line tool or winzip, or the tool of your choice). If everything is right, you now already hold a "SCORM package". As SCORM is a format understood by not only our portal, but also learning management systems, you can test the package right away by uploading it to your LMS: moodle and Ilias support it right away. Where exactly that functionality is found depends, of course, on your LMS. For Ilias, I have a menu that enables me to upload any content, let it be text, pdf, html or - and this is what we need - a SCORM package. If adding the package works, then test whether it renders correctly. If it does, the job is done.