OpenSyllabus Quick Start

Warning

This page relates to an older OpenSyllabus Version. Some parts needs revision.

Quick start for developers

Document Purpose

If you only want to test OpenSyllabus, you can download a bundle that includes a Tomcat instance configured with Sakai and OpenSyllabus.

This tutorial will show you how to load the OpenSyllabus project into Eclipse.

Installation Guide

Installation and deployment of OpenSyllabus with Sakai 2.5 and GWT 1.5

1) First of all, a working installation of Sakai 2.5 is mandatory prior to install OpenSyllabus **but** at this time, OSYL is not synchronized with Sakai 2.5 trunk, thus you will need to download a bundle corresponding to a specific snapshot of Sakai 2.5 (see below).

That said, if you never have installed the Sakai Development Environment, you should read and follow the instructions at the following address. This will allow you to install and configure Subversion, Maven, Eclipse and Tomcat.

Installation of Sakai 2.5 Development Environment

see http://confluence.sakaiproject.org/confluence/display/BOOT/Development+Environment+Setup+Walkthrough)

2) Next, get the OpenSyllabus source-code

The OpenSyllabus source code is not yet available from a Subversion repository but here is how to download it as an archive.

Download OpenSyllabus source code

Download the 2 following zip archives: osyl-sakai-src-alpha.zip (36 MB) and osyl-src-alpha.zip (2 MB).

Extract osyl-sakai-src-alpha.zip first, it will create a folder osyl-sakai25 then extract osyl-src-alpha.zip in that folder.

3) You also need to install the Google Web Toolkit (GWT) library on your PC

Download GWT

GWT library is available here: http://code.google.com/webtoolkit/download.html.
GWT version 1.5.2 (shipping release) is mandatory.

4) Create an environment variable called GWT_HOME

The GWT_HOME environment variable must point to your GWT installation directory.

Considering <path>, the path to the GWT installation directory

Create GWT_HOME environment variable

Windows: Via the Control Panel -> System -> Advanced -> Environment Variables -> System Variables -> New
and type GWT_HOME and the <path> in the appropriate fields

Linux (Bourne, Bash, and related shells):
export GWT_HOME=<path>

5) Next, get and install the google-web-toolkit-incubator library
In fact, OSYL uses some UI components (i.e. Calendar) from the GWT incubator library.
Look at this library to find some more cool GWT components.
This JAR is compatible with jvm 1.5+ and GWT version 1.5.2 is mandatory.

Install the google-web-toolkit-incubator

GWT incubator library can be download from here

Put the gwt-incubator_1-4_final.jar into your GWT base directory (i.e. GWT_HOME)

6) Don't forget to put a copy of the gwt-user.jar into the TOMCAT shared/lib directory

Considering <GWT_HOME>, the path to the GWT installation directory
Considering <TOMCAT_HOME>, the path to the TOMCAT installation directory

Put a copy of the gwt-user library into TOMCAT shared

<GWT_HOME>/gwt-user.jar into <TOMCAT_HOME>/shared/lib

7) Next, you have to compile the JCR module first

Considering <SAKAI_HOME>, the path to the Sakai installation directory (osyl-sakai25 in our case)
Use this MAVEN command from the Sakai's base directory (i.e. <SAKAI_HOME>)

Compile the JCR module

```
cd <SAKAI_HOME>
cd jcr
mvn clean install -Dmaven.test.skip=true
```

8) Then compile and deploy Sakai for the first time with this command :

Considering <TOMCAT_HOME>, the path to the TOMCAT installation directory

Compile the Sakai project

```
cd ..
mvn clean install sakai:deploy -Pmini -Dmaven.test.skip=true -Dmaven.tomcat.home=<TOMCAT_HOME>
```

9) For subsequent compilations and deployment, you can use this command :

Considering <TOMCAT_HOME>, the path to the TOMCAT installation directory

Further compilation of the Sakai project

```
mvn install sakai:deploy -Dmaven.test.skip=true -Dmaven.tomcat.home=<TOMCAT_HOME>
```

Eclipse Configuration

Perhaps the greatest advantage of using GWT is having the capability to leverage advanced software engineering and use established Java integrated development environment (IDE).
We suggest to use the Eclipse IDE since it works very well and has support with GWT to help integration.

Create a configuration variable called GWT_HOME and pointing to your GWT installation directory.

Add a path to the gwt-incubator_1-4_final.jar library

OpenSyllabus Code Structure

Let's have a look at the OpenSyllabus code structure.

- Sakai Source Code Structure
  - osyl-core-gwt (Frontend project)
    - client

        - public
        - shared
    - osyl (Backend project, ie OSYL-Sakai tool code)
        - api dir
        - impl dir
        - tool dir

To keep it simple, in our developpement we created two separate projects (we mean Eclipse/Maven projects) : One project for the client part where you will find al the GWT source code and one project for the backend part where stands the Sakai's tool code.

Frontend Code Structure

OpenSyllabus frontend is essentially organized around the recommended GWT project structure where we have replaced the server side code by a shared package and moved server RPC API definition (i.e. Interfaces files OsylEditorGwtService.java & OsylEditorGwtServiceAsync.java) into the rpc subpackage of the client.

- Frontend project (GWT code)
    - client
        - controller
            - event
            - OsylController.java
            - OsylRPCController.java
            - OsylViewContext.java
        - OsylImageBundle
        - rpc
            - OsylEditorGwtService.java
            - OsylEditorGwtServiceAsync.java
        - view
    - shared
        - api
        - events
        - model
    - public
        - css
        - img
        - html

Package	Purpose
org/sakaiquebec/opensyllabus/	The project root package contains modules XML files (i.e OsylEditorEntryPoint.gwt.xml and shared.gwt.xml)
org/sakaiquebec/opensyllabus/client/	Client-side source files and subpackages
org/sakaiquebec/opensyllabus/client/controller/	GUI controller and RPC controller, plus GUI's event subpackage and ViewContext utility
org/sakaiquebec/opensyllabus/client/OsylImageBundle/	All the images resources to be compressed using the Image Bundles of GWT
org/sakaiquebec/opensyllabus/client/rpc/	The two Service Interfaces required by the GWT RPC mechanism
org/sakaiquebec/opensyllabus/client/view/	The main package of the OSYL Client containing all the GUI components: Composite views, panels, tree, toolbar, buttons, ...
org/sakaiquebec/opensyllabus/shared/	Server-side code, including Interfaces, Model update events, Model definition
org/sakaiquebec/opensyllabus/public/	Static resources that can be served publicly

The project was initially created following the instruction of Creating an Application from Scratch (with Eclipse)

Backend Code Structure

OpenSyllabus backend is organized as a classical Sakai's tool following a File Structure based on classic Sakai's tool directories, such as : API (interfaces), Impl (implementations) and Tool (webapp) directory. In the Tool directory, there are RPC Servlets and RPC Interfaces for Remote Procedure Call exchanges in the src/Java folder. In addition, the Tool directory contains the Index.jsp and something new : the compiled GWT content (all the generated GWT stuff : pure JavaScript and HTML) in the src/webapp folder.

- Backend project (Sakai's tool code)
    - api dir
    - impl dir
    - tool dir
        - java
            - client
                - rpc
                    - OsylEditorGwtService.java (RPC Servlet)
                    - OsylEditorGwtServiceAsync.java (RPC Async Interface)
                - server

        - OsylBackingBean.java
        - OsylEditorGwtServiceImpl.java
    - webap
        - index.jsp
        - org.sakaiquebec.opensyllabus.OsylEditorEntryPoint (all generated JS code + HTML + images)
            - osylcoconfigs (configuration files) I18N properties files, skin: images & css, rules.xml)

Package	Purpose
tool/src/java/	Server-side Java source-code of the Sakai's Tool Package
tool/src/java/org/sakaiquebec/opensyllabus/client/rpc/	The two Service Interfaces required by the GWT RPC mechanism
tool/src/java/org/sakaiquebec/opensyllabus/server/	The Java source-code of the Backing Bean and the Servlet Implementation of the Service required by the GWT RPC mechanism
tool/src/webapp/	Server-side webapp files: index.jsp, WEB-INF, tools config, osylconfigs files and compiled JS & HTML stuff
tool/src/webapp/WEB-INF/	Webapp configuration files: web.xml and applicationContext.xml
tool/src/webapp/org.sakaiquebec.opensyllabus.OsylEditorEntryPoint	All the Client-side compiled JS & HTML stuff

Development Cycle

Thus, the developer can code and debug all in Java by working depending to his needs in the backend project or in the client project.

Since the server is not available, in the client project you have to plan to write into your code mockups of server. For instance, you can simulate the server by using static fake data code.

We've found 3 possible use cases:

Use case A - client update only

When you have made a client code update only.

Hosted Mode

During most of your development, your GWT code runs in "Hosted Mode" that lets you debug like a normal Java application from your Eclipse IDE.

In "Hosted Mode", a JVM executes the GWT code as Java bytecode inside a special embedded browser window (before your Java code having been translated into JavaScript).

Running GWT Webapp in "Hosted Mode" makes debugging easy following a standard edit-run-debug cycle

- Edit your source
- Refresh
- Check the results

> **To launch a hosted mode session with OpenSyllabus from Eclipse**
>
> Your startup file is OsylEditorEntryPoint-shell.cmd (select it, right-click on it + Open With + Default Editor).
> It should launch a program from the console.

Deployment in Web Mode

Once tested in "Hosted Mode", you can compile your Java source code to JavaScript the and deploy your Webapp. GWT Webapp that has been deployed is also said to be running in "Web Mode".

In this use case, you have to invoke the JavaScript compilation then move the compiled JavaScript to the backend code. To deploy your Webapp in production, you would move the files in your www/... directory to your web server, e.g. Tomcat Also if your backend code is already deployed in Tomcat, just replace the JavaScript code & HTML by the new one. When compiled the Client-side is now pure JavaScript and HTML.

To compile your Java source code to JavaScript

Just click on the Compile/Browse button in the embedded browser window 
Or open the startup file OsylEditorEntryPoint-compile.cmd (select it, right-click on it + Open With + Default Editor).
It should launch a program from the console.

Use case B - backend update only

Here, you only have to invoke Sakai Application Compilation and deployment with maven.

Use case C - full compilation

The full compilation process is necessary when for example you change the interfaces of the Remote Procedure Call (RPC).

Note

To save time, use "Hosted Mode" as much as possible for client code developments, because,it is faster than a GWT compilation.