1 TextGridLab Modular Build

With the TextGridLab 2.0 Release Candidate, the lab has been modularized, and it's build system is based on Tycho.

1.1 General Structure

1.1.1 TextGridLab Core

The core module consists of all plugins that form general API for other tools, or that are required by those API plugins. Here live those components that directly interface with the repository. A TextGridLab tool will typically depend on this feature resp. plugins thereof.

1.1.2 TextGridLab Base

The base module contains tools that are present in every TextGridLab installation, and the basic application framework. Here is the place for tools like the navigator, user management, UI for TG-search and TG-publish, welcome screen etc. The workflow tool lives here as well due to the copy workflow dependency. Additionally, this module currently aggregates XML Editor, Text Image Link Editor, and Dictionary Tools; and it aggregates the required stuff from Eclipse like the update mechanism (p2).

The base module contains a product build for the base textgridlab.

1.1.3 Other tools

All other tools or tool groups have their own build etc., where XML Editor, Text Image Link Editor and Dictionaries are aggregated by the base feature and packed into the base product.

1.2 How to set up a typical tool build

This HOWTO should also explain how the build is structured ...

I'll be glad to help with the tycho steps. The build steps can currently only be done by me.

1.2.0.1 Arrange projects in SVN

1. Create a directory for the tool. Typically this will be a direct subdirectory of `trunk/lab`, but this is not a strict requirement.
2. Move the plugins and features that make up the module into the tool directory. Each plugin / feature project MUST reside in a direct subdirectory of the tool directory, and the subdirectory SHOULD typically be named like the plugin / feature to avoid confusion.

1.2.0.2 Prepare Tycho build (you need Maven 3 for that)

1. Check out the tool directory. `cd` to the tool directory.
2. Generate the POM files using the following command line:

   ```
   mvn3 -Dtycho.mode=maven org.eclipse.tycho:tycho-pomgenerator-plugin:0.14.1:generate-poms -DgroupId=info.
   textgrid.lab
   ```

   This will create a file named `pom.xml` in each of the direct subdirectories and in the tool directory itself. The POM file controls the build.
3. Edit the root `pom.xml`: I recommend that you replace everything after the list of modules with the corresponding part of an existing `pom.xml`, e. g., from the link editor.
4. If you need stuff from other update sites, add their URLs to the `<repositories>` section as the ones you see in the link editor snippet.
5. Add the POM files to the repository:

   ```
   svn add **/pom.xml
   ```

1.2.0.3 Prepare the repository

Every tool build must prepare a *p2 repository* (~ an update site) from which other tools, or TextGridLab end users, can install their tools. This is anothe submodule of our tool that I ask you, by convention, to name *tool*-repository.

1. Copy an existing *repository* module, e.g., from the link editor.
2. Edit the `pom.xml` to reflect your project – you'll typically just have to replace *linkeditor* with your tool's name, and adopt the version in the parent section.

   Example Repository pom.xml

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.
   xsd" xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
     <modelVersion>4.0.0</modelVersion>
     <parent>
        <artifactId>linkeditor</artifactId>  <!--  Your tool here -->
        <groupId>info.textgrid.lab</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <groupId>info.textgrid.lab</groupId>
    <artifactId>linkeditor-repository</artifactId>  <!-- Your repository name here -->
    <packaging>eclipse-repository</packaging>
   </project>
   ```

3. Edit the `category.xml` to reflect your feature(s). The category.xml defines features and categories. Only those features mentioned here, and their dependencies, will be packaged into the repository – no category.xml, no p2 repository. The TextGridLab's update mechanism will usually only show those features that are in a category. Use the category ID *lab* to have your feature(s) show up in the *TextGridLab* category.

Example category.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
 <feature url="features/info.textgrid.lab.linkeditor.feature_0.0.0.jar"
          id="info.textgrid.lab.linkeditor.feature" version="0.0.0">
    <category name="lab"/>
 </feature>
 <category-def name="lab" label="TextGridLab"/>
</site>
```

You MUST use your exact feature name in the `id` field. Use the version number *0.0.0* to let the build insert the actual version number of the feature that is built – Tycho will synchronize stuff for you.

4. Add the repository module to the parent pom. To do that, open the pom.xml file in your tool's directory and add an entry like

```
<module>linkeditor-repository</module>
```

to the `<modules>` section of the pom.xml file. The entry must be the subdirectory name of the repository module.

5. Add the repository module to the svn.

```
svn add linkeditor-repository
```

6. Test the build. You need Maven 3 for that.

```
mvn3 clean package            # or mvn clean package, if your maven 3 executable is called mvn
```

7. Commit. You're done. 

1.2.0.4 Set up the Jenkins build (Thorsten only)

Modularized build jobs are called `lab-`*something* to show up in the corresponding view.

1. Create a new job named `lab-`*something* as a copy of `lab-core`
2. Fix the repository URL
3. Optionally, add dependencies
4. Fix the update site name, i.e. replace `core` with *tool* in all places

TODO add details, especially the update site script

1.2.0.5 Set up the update site

a. add subdirectory to `compositeContents.xml`
b. add subdirectory to `compositeArtifacts.xml`
   don't forget to adjust the number of children ...

Update Site Management

Currently it's plain files only. Maybe https://github.com/mechko/nexus-p2-tycho-aggregator-plugin could be of interest, the author uses it to manage tycho-built features in Nexus 2.0 OSS.

1.2.0.6 Signing

Installing a feature from an update site into textgridlab will cause an (ignorable) complaint about "unsigned content". It should be possible to sign all jars using this workflow: http://stackoverflow.com/questions/7956267/tycho-jar-signing . The check could also be disabled setting -Declipse.p2.unsignedPolicy=allow in textgridlab.ini (http://aniszczyk.org/2010/05/20/p2-and-the-unsigned-dialog-prompt/).