1 Libraries in TextGridLab

To put a generic library – e.g. client libraries for our services, or common stuff like the link rewriter – into the TextGridLab, the following steps are neccessary:

1. Make the library an OSGi bundle (using `maven-bundle-plugin`) and deploy it to a Maven repository
2. Aggregate the library into an Eclipse feature, and put it into a p2 repository. We use the `textgridlab-dependencies` project for that.
3. Depend on feature in an Eclipse-only feature or the product, and depend on the bundle or its packages in your bundle.

There are some caveats in each of the steps.

2 Making your library an OSGi bundle

In principle, your library can be OSGified by adding specific metadata to its Manifest file. There is a Maven plugin, the maven-bundle-plugin, created in the Apache Felix project and using the `bnd` bundling tool, that is able to auto-generate the metadata for you. There are two alternative ways to use this plugin:

- define a packaging type of `bundle` for your project
- leave the packaging type `jar`, but add an execution to your build that calls the bundle plugin's `manifest` goal to manipulate the bundle manifest.

You should avoid version 2.3.5 of the bundle plugin since it collides with generating source jars. Both earlier and later versions are fine, we have not yet encountered any specific trouble with version 2.4.0.

Maven vs. OSGi version

Your OSGified bundle will have two versions, and this is bound for trouble since the versioning systems are fundamentally different:

- a **Maven version** typically consists of three dot-separated numeric parts with an optional `-SNAPSHOT` suffix. `SNAPSHOT` will be replaced with a timestamp when you deploy the snapshot to a repository, you can still reference the latest snapshot using the version string `1.2.3-SNAPSHOT`. A released version does not have a suffix, so `1.2.3-SNAPSHOT` < `1.2.3`.
- an **OSGi version** typically consists of four segments. In Eclipse and Tycho, the last segment can be specified as `qualifier` in the source, which will be replaced by a timestamp (or an arbitrary override) during build. It is not possible to specify a literal `qualifier` in a version dependency, however it is not customary to specify four-component-version numbers in dependency, typically a dependency on `1.2.3` will take any (= usually the newest available) version starting with 1.2.3.
  There are different semantics than with the maven SNAPSHOT for OSGi versions: `1.2.3.201312281305` > `1.2.3`

The Maven version will be used when (plain) Maven is used to resolve the library, e.g., from a Maven repository. The OSGi version is used by, e. g., Eclipse and p2. Note that Felix, as opposed to Tycho, includes a literal SNAPSHOT in the OSGi version number instead of expanding it, making different SNAPSHOTs of the same version indistinguishable for p2. Thus, **you should update your snapshots' explicit version numbers if you want to force your plugin to land in the TextGridLab**.

3 Including your library in a feature and in a p2 update site

We maintain a project as part of the TextGridLab build that is used for the transformation from the Maven world to the Eclipse/p2 world – textgridlab-dependencies. This project is integrated in the TextGridLab integration build, and we're using a Jenkins build to deploy its result to a p2 update site from which the other TextGridLab components fetch their dependencies.

Like the other TextGridLab components, textgridlab-dependencies uses the Git Flow model, so please push your commits into the `develop` branch.

3.1 Integrating a new dependency

1. Add the maven coordinates of your dependency to the `dependencies` section of textgridlab-dependencies' root POM. Please use a property for the version number.
2. Add a feature submodule. We'd suggest you copy an existing feature submodule and adjust its names, don't forget to add a `modules` entry to the root pom.xml. You can also add your dependency to a suitable existing feature.
    - ⚠️ You SHOULD either use a -SNAPSHOT (pom.xml) / .qualifier (feature.xml) version or you must explicitly increase your feature's version number every time you upgrade the included libraries.
3. Add a `plugin` entry for your library to your feature.xml:

    feature.xml excerpt

    ```
    <plugin
        id="info.textgrid.utils.linkrewriter.core"
        download-size="0"
        install-size="0"
        version="0.0.0"
        unpack="false"/>
    ```

    The ID is the `Bundle-Symbolic-Name` of the OSGi bundle, it is typically generated by the maven-bundle-plugin using a sensible heuristic. You should use `0` for `download-size` and `install-size` as these will by calculated on build. We also recommend to insert `0.0.0` as version – this will be replaced by the actual version used on build (and specified by you in the root `pom.xml`).
4. Commit and push your changes and watch the integration build.

3.2 Upgrading an existing dependency

1. Make sure your library’s version has been increased, see the note on versions above.
2. Update your library’s version number in textgridlab-dependencies' root pom.xml
3. You might also want to update the feature version.

4 Using a new or updated dependency in TextGridLab development

1. Make sure the build of the textgridlab-dependencies project including your update has run.
2. In Eclipse, open your target platform configuration (Window Preferences Plug-in development Target platform, select your platform, *Edit).* If your TP already includes the dependencies update site, select it, *Update* and apply the changes. Otherwise, *Add* a new *Software Site*, enter the URL of the textgridlab-dependencies project's update site (find it on the build page).
3. If your library feature is new, include it into an existing TextGridLab feature or add it as a dependency and include it in the product.
4. In your TextGridLab plugin, add your library's packages to the `Import-Packages` statement or to the `Required-Bundles` statement (both can be found in the bundle manifest editor). You should specify version dependencies. If you intend to use new features after an upgrade of your library, you *must* use version dependencies to make this explicit.