Build Management

An Overview of KIELER Build Automation

In order to keep the KIELER code as stable as possible, we use automatic continuous integration builds. Once you push something into one of the KIELER repositories, you trigger the automatic build process. If the build fails for whatever reason – be it failed unit tests or simply compilation errors – you are notified of the problem.

This page describes the software and setup we use to implement all of this.

Content

Software We Use

To implement our automatic builds, we use the popular Maven tool in conjunction with Tycho, a set of Maven plug-ins that allow Maven to build Eclipse projects. Our KLay layouters library is built using Apache Ant. To implement our continuous integration builds, we use Atlassian Bamboo.

Maven / Tycho

Maven is a build tool for Java projects. It takes care of dependency management, including in-build dependencies (the order in which packages are compiled) as well as dependencies to third-party libraries. The latter are automatically fetched from special Maven repositories. Without getting too technical, a Maven build consists of several phases, such as *compile* and *package*. Within each phase, several Maven plug-ins handle different *tasks* (or *goals*, as Maven calls them). The *maven-compile-plugin* for example compiles `.java` files into `.class` files.

To correctly compile a project, Maven needs to be told about the project. While the popular Ant build tool uses `build.xml` files to describe the steps to be executed for building a project, Maven uses `pom.xml` files to describe the project and figures out the steps for itself. The POM files may inherit settings from a parent POM file.

Tycho is a set of Maven plugins that handles compiling and dependency management as well as bundling of Eclipse plug-ins. Tycho understands Eclipse metadata files such as `plugin.xml` or `feature.xml`, provides dependency information extracted from those files, and provides an Eclipse instance for compiling and packaging Eclipse bundles.

Apache Ant

Ant is a very popular Java build tool. While Maven wants to know metadata about a project and then knows what to do to build it, Ant works by specifying exactly what to do to build a project. These steps are configured in a `build.xml` file. We try to avoid using Ant, but still have Ant build files around for jobs too specialized to be properly handled by Maven.

Bamboo

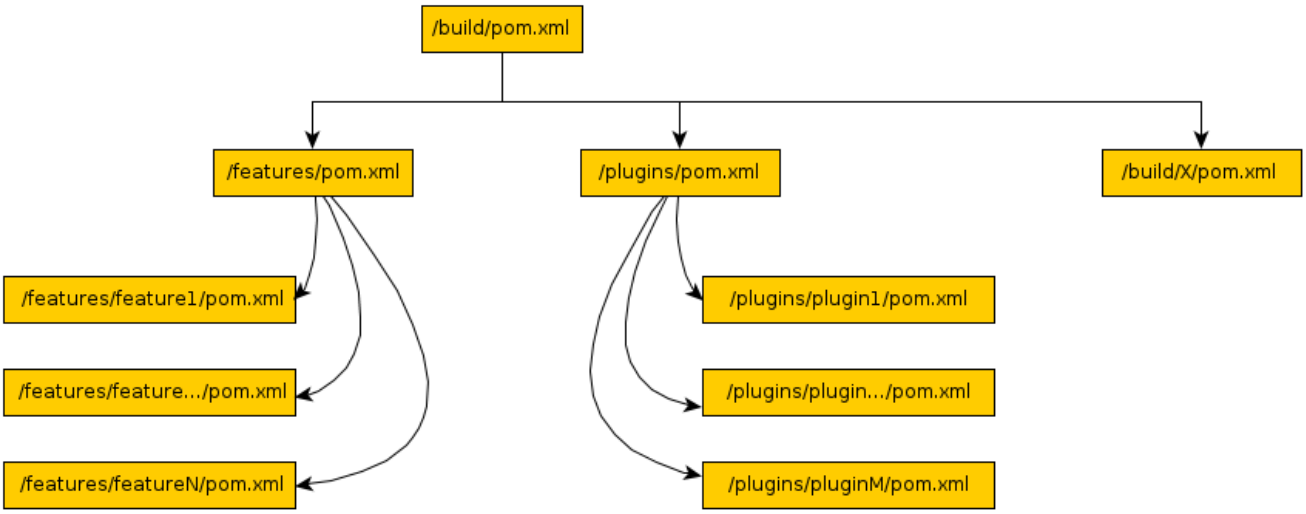
While Maven and Tycho know how to compile KIELER, Bamboo knows when to compile KIELER and what to do with the compiled project. Bamboo has access to our source code repositories and triggers continuous integration builds every time someone pushes new code into a repository. It also does a full build every night and copies the results onto our nightly build update site to be accessed by people all around the world. And beyond. Tell your friends!

The Automatic Build Process

This section describes how our POM files are distributed throughout the repository structure, and how you can trigger an automatic build of KIELER.

POM Files

The basic structure of the POM files is the same for both, the pragmatics and semantics repository, and can be seen below:



Each plug-in and feature has a corresponding (usually rather small) POM file. The POM files in the `features` and `plugins` directories know about the different features and plug-ins. The parent POM file, which all other POM files copy basic configuration from, knows about the feature and plug-in POM files, as well as about every kind of build configuration we have (for building the pragmatics repository, for building the KWebS product, etc.). In addition, the `build` directory also contains a bunch of subdirectories housing POM files that produce our p2 repositories and our products.

Using the KIELER Maven Build

Using the KIELER Maven build requires two things: a working Maven installation (d'oh...) and knowledge about how exactly to trigger a build. We leave the former to you, but the latter is what this section is about.

To actually build KIELER, navigate to the `/build/` directory and run the following command line:

```
. /home/java/java-env           # Only necessary when working on our servers
mvn clean package -P <profile>
```

Once Maven has finished, the different build artifacts may be found in the `/build/de.cau.cs.kieler.<profile>.repository/target` directories. The following build profiles are available in the KIELER Pragmatics repository:

- klighdning – Builds a standalone version of the KLighDning server, based on the current nightly pragmatics update site.
- kwebs – Builds a standalone version of the KWebS server, based on the current nightly pragmatics update site.
- pragmatics – Builds a p2 repository that contains our pragmatics features. This is what would usually be pushed online for people to download as our nightly pragmatics build.
- ptolemy-rcp – Produces a standalone version of our Ptolemy Model Viewer, based on the current nightly build pragmatics update site.

The following build profiles are available in the KIELER Semantics repository:

> **ToDo**
>
> Document the Semantics build profiles.

What to Be Aware of

There are some things that people need to be aware of to keep the build files in a valid state.

1. Eclipse metadata and POM files are not synchronized. Thus, if the version of an Eclipse plug-in changes, its `pom.xml` needs to be updated accordingly.
2. The repository POM directories contain product files and product icons. These are copies of the files found in the corresponding branding plug-ing (such as `de.cau.cs.kieler.core.product`) and have to be manually synchronized.

Continuous and Nightly Builds

There are basically four different kinds of build plans for each of the KIELER projects:

1. **Continuous Plugins** – Compiles the plug-ins and runs the unit tests on them. None of the compiled artifacts are published anywhere (in fact, no update site or product is even produced). This plan is triggered by pushing stuff into the repositories, giving early feedback regarding whether committed changes break anything.

2. **Nightly <Product>** – Assembles distributable product files and/or update sites. Distributable files are published in the nightly build directory `/home/kieler/public_html/files/nightly`. Update sites are published in `/home/kieler/public_html/updatesite/nightly`. These plans are run once every night.
3. **Nightly Rating** – Compiles the plug-ins and runs our code quality rating doclet on them. The result is a website published at `/home/kieler/public_html/rating`. This plan is run once every night.
4. **Release Builds** – Continuous builds of release branches once a release is imminent. These usually run whenever changes are pushed into the repository. All release builds are placed in a special Bamboo project calles *KIELER Releases*.

The *Semantics* project has an additional build plan:

1. **Ptolemy Update Site** – Builds the Ptolemy library and produces an update site for it. The build is triggered whenever changes are pushed into the Ptolemy repository. This build plan is necessary since both, the *Pragmatics* and the *Semantics* projects use the Ptolemy libraries.

Update Sites and Redistributables

Our automatic builds produce a bunch of so-called *artifacts*: redistributable applications as well as a number of update sites. This table lists all artifacts, the project or repository they belong to, the build file responsible for producing them, the Bamboo build plan that builds them, and the directory they are finally placed in.

Timestamps

Don't be alarmed if the timestamps of the plugin and feature jar files on the updatesite don't match the current time. The time is probably UTC. Which is nice.

Artifact	Repository	Build File	Bamboo Build Plan	Final Directory
KWebS RCA	Pragmatics	`...kwebs.repository/pom.xml`	KIELER Pragmatics -> Nightly KWebS	`/home/kieler/public_html/files/nightly/kwebs`
KIELER Pragmatics Updatesite	Pragmatics	`...pragmatics.repository/pom.xml`	KIELER Pragmatics -> Nightly Pragmatics Updatesite	`/home/kieler/public_html/updatesite/nightly/pragmatics/`
Papyrus Layout Updatesite	Pragmatics	`...papyrus.repository/pom.xml`	KIELER Pragmatics -> Nightly Papyrus Updatesite	`/home/kieler/public_html/updatesite/nightly-papyrus/`
KIELER RCA	Semantics	`...semantics.repository/pom.xml`	KIELER Semantics -> Nightly Semantics Product	`/home/kieler/public_html/files/nightly/`
KIELER Semantics Updatesite	Semantics	`...semantics.repository/pom.xml`	KIELER Semantics -> Nightly Semantics Updatesite	`/home/kieler/public_html/updatesite/nightly/semantics/`
Ptolemy Libraries Updatesite	Ptolemy	`...ptolemy.repository/pom.xml`	KIELER Semantics -> Ptolemy Updatesite	`/home/kieler/public_html/updatesite/ptolemy/`