This page last changed on Mar 13, 2011 by patrik_nordwall.

Sculptor Archetype Tutorial

In this tutorial we will create a simple Java EE application from scratch using the Maven archetypes provided by Sculptor. It consists of the following projects:

- helloworld-parent - Only a maven project for building the other parts.
- helloworld - Business tier. Containing the services and domain objects.
- helloworld-web - Presentation tier. Web application with CRUD GUI.

We will start with a simple war, deployed in Jetty and using HSQLDB in-memory database. Later on it is illustrated how it can be converted to use MySQL and be deployed in JBoss.

Table of Contents:

Setup maven projects

We start with creating a script that calls the 4 maven archetypes to generate the project structures and maven build files. It also does an inital build and generation of Eclipse project with the maven eclipse plugin. Of course you can execute these commands one by one from the command prompt, but the script is useful when doing this several times.

Copy the following script to `sculptor-archetypes.cmd`, located in the root of your Eclipse workspace. Adjust paths to your environment.

Windows script:

```
set MVN_HOME=C:\devtools\Maven-2.0.8
set JAVA_HOME=C:\devtools\jdk1.6.0_03
set path=%MVN_HOME%\bin;%JAVA_HOME%\bin
set PACKAGE=%1
set SYS_NAME=%2

call mvn archetype:generate -DinteractiveMode=false -DarchetypeGroupId=org.fornax.cartridges
-DarchetypeArtifactId=fornax-cartridges-sculptor-archetype-parent -DarchetypeVersion=2.0.0 -
DarchetypeRepository=http://fornax-platform.org/nexus/content/repositories/public -DgroupId=
%PACKAGE% -DartifactId=%SYS_NAME%-parent -Dpackage=%PACKAGE% -Dversion=1.0-SNAPSHOT

call mvn archetype:generate -DinteractiveMode=false -DarchetypeGroupId=org.fornax.cartridges
-DarchetypeArtifactId=fornax-cartridges-sculptor-archetype -DarchetypeVersion=2.0.0 -
DarchetypeRepository=http://fornax-platform.org/nexus/content/repositories/public -DgroupId=
%PACKAGE% -DartifactId=%SYS_NAME% -Dpackage=%PACKAGE% -Dversion=1.0-SNAPSHOT
```

call mvn archetype:generate -DinteractiveMode=false -DarchetypeGroupId=org.fornax.cartridges -DarchetypeArtifactId=fornax-cartridges-sculptor-archetype-jsf -DarchetypeVersion=2.0.0 -DarchetypeRepository=http://fornax-platform.org/nexus/content/repositories/public -DgroupId=%PACKAGE% -DartifactId=%SYS_NAME%-web -Dpackage=%PACKAGE% -Dversion=1.0-SNAPSHOT

pause

cd %SYS_NAME%-parent

call mvn install

pause

call mvn -DdownloadSources=false eclipse:eclipse

cd ..

Unix bash script:

```
#!/bin/bash
if [ -z $1 ] || [ -z $2 ]; then
   echo -e "Usage: $0 PACKAGEID ARTIFACTID\n\tPACKAGEID - name of Java package, for example
org.helloworld"
   echo -e "\tARTIFACTID - project base name, for example helloworld"
   exit 1
fi

PACKAGE=$1
SYS_NAME=$2

mvn archetype:generate -DinteractiveMode=false -DarchetypeGroupId=org.fornax.cartridges -DarchetypeArtifactId=fornax-cartridges-sculptor-archetype-parent -DarchetypeVersion=2.0.0 -DarchetypeRepository=http://fornax-platform.org/nexus/content/repositories/public -DgroupId=$PACKAGE -DartifactId=$SYS_NAME-parent -Dpackage=$PACKAGE -Dversion=1.0-SNAPSHOT

mvn archetype:generate -DinteractiveMode=false -DarchetypeGroupId=org.fornax.cartridges -DarchetypeArtifactId=fornax-cartridges-sculptor-archetype -DarchetypeVersion=2.0.0 -DarchetypeRepository=http://fornax-platform.org/nexus/content/repositories/public -DgroupId=$PACKAGE -DartifactId=$SYS_NAME -Dpackage=$PACKAGE -Dversion=1.0-SNAPSHOT

mvn archetype:generate -DinteractiveMode=false -DarchetypeGroupId=org.fornax.cartridges -DarchetypeArtifactId=fornax-cartridges-sculptor-archetype-jsf -DarchetypeVersion=2.0.0 -DarchetypeRepository=http://fornax-platform.org/nexus/content/repositories/public -DgroupId=$PACKAGE -DartifactId=$SYS_NAME-web -Dpackage=$PACKAGE -Dversion=1.0-SNAPSHOT

sleep 1

cd $SYS_NAME-parent
mvn install
sleep 1

mvn -DdownloadSources=false eclipse:eclipse
cd ..
```

Run this script by defining the package name as the first parameter and the application id (artifact id of business tier) as the second parameter.

```
sculptor-archetypes org.helloworld helloworld
```

Maven 2 Plugin for Eclipse
The above instruction uses `eclipse:eclipse` to generate Eclipse .project and .classpath files. This is a simple and fully functional approach in most cases. An alternative to

`eclipse:eclipse` is to use the Maven 2 Plugin for Eclipse. If you prefer that plugin there is no problem with using that approach instead.

Import into Eclipse

Open Eclipse and import the 3 projects.

Complete Business Tier

Open `model.btdesign` located in `src/main/resources` of the helloworld project.
Add something like this to the design file.

```
Application Universe {
    basePackage=org.helloworld

    Module milkyway {
        Entity Planet {
            scaffold
            String name key;
            String message;
            Integer diameter nullable;
            Integer population nullable;
            - Set<@Moon> moons opposite planet;

        }
        Entity Moon {
            not aggregateRoot // belongs to Planet Aggregate
            String name key;
            Integer diameter nullable;
            - @Planet planet opposite moons;
        }
    }
}
```

Note the `scaffold` feature of `Planet`. It will result in automatically generated CRUD operations in the `PlanetRepository` and `PlanetService`.

You also need to adjust `model.guidesign` in web project, because you have renamed the application.

```
import 'platform:/resource/helloworld/src/main/resources/model.btdesign'
gui UniverseWeb for Universe {

}
```

Build

1. Build the application with `mvn -Dmaven.test.skip=true clean install` from the helloworld-parent project. We skip the JUnit tests in this tutorial, see Hello World Tutorial for more information about the JUnit tests.

2. Refresh the 3 projects in Eclipse.

Fornax Maven Launcher
You can checkout/import the Fornax Maven Launcher into the workspace to be able to run maven inside Eclipse. See Installation Guide.

Run in Jetty

Deploy the application and start Jetty with `mvn jetty:run` from the helloworld-web project. Note that no installation is needed. Jetty is launched from maven.

Open http://localhost:8080/helloworld-web in your browser.

Try the CRUD GUI and create some favourite Planets and Moons.

The features of the generated web application is explained in CRUD GUI Tutorial.

By default an in memory database, hsqldb, is bundled with the the application. Of course, when you restart the application or server all data added will be lost.

Jetty is an excellent development server, and you can stop here if you don't need to run in JBoss. You can also convert to JBoss later if you like.

JBoss

Install JBoss AS

Install JBoss AS 5.1.0.GA.

There is a strange hot deployment problem due to conflicting classloading of Ehcache and ear classloader. Therefore you need to copy ehcache-core-1.7.2.jar jar to `server/default/lib`. ehcache-core-1.7.2.jar is in your maven repository

Adjust for JBoss

Since JBoss has some libraries in its default classpath (lib directory) you need to adjust scope to provided for several dependencies in pom.xml files (both business tier and presentation tier). Hibernate, slf4j, xerces and some more should not be bundled in war or ear when deployed to JBoss.
Search for comments like this and change the scope to provided or test according to the comment.
<!-- Add scope provided when deployed in jboss -->
<!-- Add scope test when deployed in jboss -->

Since you changed the scope of the logging dependencies to scope provided in the business tier you must add them to the presentation tier pom.xml. Add the following to pom.xml in the web project.

```
<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>${slf4j.version}</version>
        <!-- Add scope provided when deployed in jboss -->
    <scope>provided</scope>
</dependency>
        <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>jcl-over-slf4j</artifactId>
        <version>${slf4j.version}</version>
        <!-- Add scope provided when deployed in jboss -->
        <scope>provided</scope>
</dependency>
<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>log4j-over-slf4j</artifactId>
        <version>${slf4j.version}</version>
        <!-- Add scope provided when deployed in jboss -->
        <scope>provided</scope>
</dependency>
<dependency>
```

```
                <groupId>ch.qos.logback</groupId>
                <artifactId>logback-classic</artifactId>
                <version>${logback.version}</version>
                <!-- Add scope provided when deployed in jboss -->
            <scope>provided</scope>
        </dependency>
```

Deploy to JBoss

Adjust the following property in `sculptor-generator.properties` in helloworld project.

```
deployment.applicationServer=JBoss
```

Rebuild with `mvn -Dmaven.test.skip=true clean install` from the helloworld-parent project.

Deploy the war file to JBoss. Copy it to `server/default/deploy/`

Start JBoss and open http://localhost:8080/helloworld-web in your browser.

Use MySQL Database

1. Adjust the following property in `sculptor-generator.properties` in helloworld project.

```
db.product=mysql
```

2. Rebuild with `mvn clean install` from helloworld-parent project.

3. Create the database schema named `universe` in your [MySQL]() database. You can use [MySQL Administrator]() to do that.

4. Run `helloworld/src/generated/resources/dbschema/Universe_ddl.sql` to create the tables. You can use [MySQL Query Browser]() or your [favorite]() database plugin to do that.

5. Add a mbean datasource in JBoss (server/default/deploy/universe-mysql-ds.xml). Use a valid user-name and password.

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
  <local-tx-datasource>
    <jndi-name>jdbc/UniverseDS</jndi-name>
    <connection-url>jdbc:mysql://localhost/universe</connection-url>
    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <user-name>root</user-name>
    <password>root</password>
  </local-tx-datasource>
</datasources>
```

6. Copy mysql-connector jar to JBoss lib directory (server/default/lib/)
It is likely that the jar is in your maven repository:

```
repository\mysql\mysql-connector-java\3.1.14\mysql-connector-java-3.1.14.jar
```

7. Redeploy and test again.

Hot Deploy

A short roundtrip is essential to achieve an efficient development environment. You need to be able to do quick hot deployment from your IDE. The generated projects contain Ant build files named `antbuild-ear.xml` and `antbuild-war.xml`. You can run these from inside Eclipse, right click and select `Run As > Ant Build`. But first you must define the location of your JBoss installation. Define the following Ant runtime property:

```
jboss.home=C:\devtools\JBoss-5.1.0.GA
```

This setting is found in Eclipse: Window > Preferences > Ant > Runtime > Properties

Use `antbuild-ear.xml` if you are deploying ear file, and `antbuild-war.xml` if you are deploying war file.

Run the default target `deploy-copy` from the parent project. It will unzip the ear/war file to JBoss deploy directory. It will also copy your current class and configuration files from the target directory. This means that when you have done some changes you don't need to do a full `mvn install`. Eclipse has compiled classes to the target directory and `deploy-copy` will copy them to JBoss and do a hot redeploy, by touching some files.

When you have done changes to `model.btdesign` you can do a quick generation by using `mvn -Dfornax.generator.force.execution=true -o -npu generate-sources` and thereafter run `deploy-copy`.

Debugging

You can start JBoss AS in Eclipse and debug your application as usual. Right click in the Server view of the Java EE perspective and follow the instructions in the wizard to add a JBoss 4.2 server.

Adding Dependencies

When you need to add dependencies to other jar files you do that by adding them to the maven pom files and thereafter run `mvn eclipse:eclipse` again. You must always run `mvn eclipse:eclipse` from the parent project.

Note that `eclipse:eclipse` also supports Eclipse project dependencies, as opposed to dependencies via jar files in the repository. In the above application the web project will have a project dependency to the business tier project. You can add other project dependencies by adding modules in the pom of the parent project and thereafter run `mvn eclipse:eclipse` again.

```
<module>../another-project</module>
```

Source

The complete source code for this tutorial is available in Subversion.

Web Access (read only):

- http://fisheye3.cenqua.com/browse/fornax/trunk/cartridges/sculptor/sculptor-helloworld-parent
- http://fisheye3.cenqua.com/browse/fornax/trunk/cartridges/sculptor/sculptor-helloworld
- http://fisheye3.cenqua.com/browse/fornax/trunk/cartridges/sculptor/sculptor-helloworld-web

Anonymous Access (read only):

- https://fornax.svn.sourceforge.net/svnroot/fornax/trunk/cartridges/sculptor/sculptor-helloworld-parent
- https://fornax.svn.sourceforge.net/svnroot/fornax/trunk/cartridges/sculptor/sculptor-helloworld
- https://fornax.svn.sourceforge.net/svnroot/fornax/trunk/cartridges/sculptor/sculptor-helloworld-web

👈	2	3	4	5	6	7	8	9	👉