

Adding Data Model Generation to a project

API Project

1. Add directory `src/main/dmgen` with `datamodel.xml` and `generator.xml`
 - Specify entities and their attributes in `datamodel.xml`
 - Specify what the generator should do in `generator.xml` (in an API project usually generating dtos and joinDtos)#
 - Contents of `generator.xml` e.g.

```
<GeneratorConfig>
  <Generate concept="dto" package="at.itteg.tis.prediction.api.dto"/>
  <Generate concept="joinDto" package="at.itteg.tis.prediction.api.joinDto"
dtoPackage="at.itteg.tis.prediction.api.dto"/>
</GeneratorConfig>
```

2. In `pom.xml`: Add resources to `build/resources` section, e.g.:

```
<resource>
  <directory>src/generated/java</directory>
  <includes>
    <include>at/iteg/tis/prediction/api/dto/*.java</include>
    <include>at/iteg/tis/prediction/api/joinDto/*.java</include>
  </includes>
</resource>
<resource>
  <directory>src/main</directory>
  <includes>
    <include>dmgen/datamodel.xml</include>
  </includes>
</resource>
```

3. In `pom.xml`: Register `maven-clean` plugin

```
<plugin>
  <artifactId>maven-clean-plugin</artifactId>
  <configuration>
    <filesets>
      <fileset>
        <directory>src/generated/java</directory>
        <includes>
          <include>at</include>
          <include>**/*.java</include>
        </includes>
        <followSymlinks>>false</followSymlinks>
      </fileset>
    </filesets>
  </configuration>
</plugin>
```

4. In `pom.xml`: Register `dmgen` plugin

```

<plugin>
  <groupId>org.clazzes.dmgen</groupId>
  <artifactId>maven-dmgen-plugin</artifactId>
  <executions>
    <execution>
      <phase>generate-sources</phase>
      <goals>
        <goal>dmgen</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <dmSpecFile>${basedir}/src/main/dmgen/datamodel.xml</dmSpecFile>
    <dmConfigFile>${basedir}/src/main/dmgen/generator.xml</dmConfigFile>
    <outputDir>${basedir}/src/generated</outputDir>
  </configuration>
</plugin>

```

Impl Project

1. Add directory src/main/dmgen with **only** generator.xml (datamodel.xml will be copied from API project using the plugin calls shown below). Contents of generator.xml e.g.

```

<GeneratorConfig>
  <Generate concept="tableDefinitions" package="at.itteg.tis.prediction.impl.schema" />
  <Generate concept="daoInterface" package="at.itteg.tis.prediction.impl.dao" dtoPackage="at.itteg.
tis.prediction.api.dto"
                                                                    joinDtoPackage="at.itteg.tis.
prediction.api.joinDto" />
  <Generate concept="jdbcDAO" package="at.itteg.tis.prediction.impl.dao.jdbc" dtoPackage="at.itteg.
tis.prediction.api.dto"
                                                                    joinDtoPackage="at.itteg.tis.prediction.api.joinDto"
supportImpExp="false"
tableDefinitionsClass="at.itteg.tis.prediction.impl.schema.TableDefinitions"
daoInterfacePackage="at.itteg.tis.prediction.impl.dao" />
</GeneratorConfig>

```

2. In pom.xml: Add call to maven-clean-plugin

```

<plugin>
  <artifactId>maven-clean-plugin</artifactId>
  <configuration>
    <filesets>
      <fileset>
        <directory>src/generated/java</directory>
        <includes>
          <include>at</include>
          <include>**/*.java</include>
        </includes>
        <followSymlinks>>false</followSymlinks>
      </fileset>
    </filesets>
  </configuration>
</plugin>

```

3. In pom.xml: Add call to maven-dependency-plugin

```

    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-dependency-plugin</artifactId>
      <executions>
        <execution>
          <id>unpack</id>
          <phase>generate-sources</phase>
          <goals>
            <goal>unpack</goal>
          </goals>
          <configuration>
            <artifactItems>
              <artifactItem>
                <groupId>at.itteg.prediction<
/groupId>
                <artifactId>prediction.api<
/artifactId>
                <outputDirectory>src/main<
/outputDirectory>
                <includes>dmgen/datamodel.xml<
/includes>
              </artifactItem>
            </artifactItems>
          </configuration>
        </execution>
      </executions>
    </plugin>

```

4. In pom.xml: Finally add call to maven-dmgen-plugin

```

    <plugin>
      <groupId>org.clazzes.dmgen</groupId>
      <artifactId>maven-dmgen-plugin</artifactId>
      <executions>
        <execution>
          <phase>generate-sources</phase>
          <goals>
            <goal>dmgen</goal>
          </goals>
        </execution>
      </executions>
      <configuration>
        <dmSpecFile>${basedir}/src/main/dmgen/datamodel.xml</dmSpecFile>
        <dmConfigFile>${basedir}/src/main/dmgen/generator.xml<
/dmConfigFile>
        <outputDir>${basedir}/src/generated</outputDir>
      </configuration>
    </plugin>

```

5. Work around a battle between Eclipse and Maven:

```

        <pluginManagement>
            <plugins>
                <!-- The following boilerplate code is just for convincing eclipse that
unpack is
                not an unsupported goal. -->
                <!--This plugin's configuration is used to store Eclipse m2e settings
                only. It has no influence on the Maven build itself. -->
                <plugin>
                    <groupId>org.eclipse.m2e</groupId>
                    <artifactId>lifecycle-mapping</artifactId>
                    <version>1.0.0</version>
                    <configuration>
                        <lifecycleMappingMetadata>
                            <pluginExecutions>
                                <pluginExecution>
                                    <pluginExecutionFilter>
                                        <groupId>
plugins
                                        org.apache.maven.
                                        </groupId>
                                        <artifactId>
plugin
                                        maven-dependency-
                                        </artifactId>
                                        <versionRange>
                                        [2.1,)
                                        </versionRange>
                                        <goals>
                                        <goal>unpack<
                                        </goals>
                                        </pluginExecutionFilter>
                                        </pluginExecution>
                                </pluginExecutions>
                            </lifecycleMappingMetadata>
                        </configuration>
                    </plugin>
                </plugins>
            </pluginManagement>

```

6. Necessary dependencies (sql-util, jdbc2xml):

```

<dependency>
    <groupId>org.clazzes.util</groupId>
    <artifactId>sql-util</artifactId>
</dependency>
<dependency>
    <groupId>org.clazzes</groupId>
    <artifactId>jdbc2xml</artifactId>
</dependency>

```

Subversion

1. Refresh in Eclipse
2. Delete all possibly (as result of testing the changes above) generated sources below src/generated/java. I.e. src/generated/java stays, but not src/generated/java/at.

3. Commit the changes so far.
4. Perform a maven build both on api and on impl project
5. Refresh both projects in eclipse
6. Both in api and in impl project, *Team* -> *Add to svn.ignore* by name the top level package below *src/generated/java*, e.g. "at".
7. Commit.

Java Build Path

1. In both projects (api and impl), add *src/generated/java* to the Build Path.
2. Commit.