Spring – MVC – Hibernate

# Prepare the workspace

For a project which uses spring and hibernate you’ve first to prepare a workspace. I use usually eclipse to create a maven project and provide a pom.xml with the needed dependencies. In this case:

For the spring framework:

|  |
| --- |
| <!-- Spring MVC framework -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>3.0.1.RELEASE</version>  </dependency>  <!-- Spring MVC framework -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-orm</artifactId>  <version>3.0.1.RELEASE</version>  </dependency>  <!-- Spring MVC framework -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>3.0.1.RELEASE</version>  </dependency> |

For the log4j framework:

|  |
| --- |
| <!-- Log4J -->  <dependency>  <groupId>log4j</groupId>  <artifactId>log4j</artifactId>  <version>1.2.15</version>  <exclusions>  <exclusion>  <groupId>com.sun.jmx</groupId>  <artifactId>jmxri</artifactId>  </exclusion>  <exclusion>  <groupId>com.sun.jdmk</groupId>  <artifactId>jmxtools</artifactId>  </exclusion>  <exclusion>  <groupId>javax.jms</groupId>  <artifactId>jms</artifactId>  </exclusion>  </exclusions>  </dependency> |

|  |
| --- |
| <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.5.6</version>  <type>jar</type>  </dependency> |

Test- and mock-framework:

|  |
| --- |
| <dependency>  <groupId>org.mockito</groupId>  <artifactId>mockito-all</artifactId>  <version>1.9.5</version>  </dependency>  <dependency>  <groupId>cglib</groupId>  <artifactId>cglib</artifactId>  <version>2.2</version>  <type>jar</type>  <scope>compile</scope>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  <scope>test</scope>  </dependency> |

Servlet engine:

|  |
| --- |
| <!-- JSTL -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.1.2</version>  </dependency>  <dependency>  <groupId>taglibs</groupId>  <artifactId>standard</artifactId>  <version>1.1.2</version>  </dependency>  <!-- for compile only, your container should have this -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>servlet-api</artifactId>  <version>2.5</version>  <scope>provided</scope>  </dependency> |

MySQL and Hibernate:

|  |
| --- |
| <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.16</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-entitymanager</artifactId>  <version>3.4.0.GA</version>  <scope>compile</scope>  </dependency> |

# Tomcat configuration

Unpack the tomcat. To get access via http://localhost:8080/manager/html to the tomcat manager site you’ve to configure a user with manager-gui role. It is good practice to give the administrator admin rights as well.

File: **<tomcathome>/conf/tomcat-users.xml**

|  |
| --- |
| user username="admin" password="nimda" roles="admin,manager-gui"/> |

## Deployment

The easiest way to deploy is to drop the war-file into the <tomcathome>/webapps directory. During development it is a better approach to create a context pointing to the target directory of your web app in development. The deployment directory is <tomcathome>/conf/Catalina/localhost. The structure of the deployment descriptor is like this:

File: **<tomcathome>/conf/Catalina/localhost/mycontext.xml**

|  |
| --- |
| <Context  docBase="<your path>/target/SpringMVC"  path="/mycontext  reloadable="true"  /> |

## Data Source

To configure a data source in tomcat a context has to be configured in <tomcathome>/conf/context.xml. In the context tag you’ve to define a resource:

File**: <tomcathome>/conf/context.xml**

|  |
| --- |
| <Resource name="jdbc/springDataSource"  auth="Container"  type="javax.sql.DataSource"  driverClassName="com.mysql.jdbc.Driver"  url="jdbc:mysql://localhost:3306/neusta"  username="<uername>"  password="<password>"  removeAbandoned="true"  removeAbandonedTimeout="90"  logAbandoned="true"  maxActive="20"  maxIdle="10"  maxWait="-1"/> |

The data base connection is now under the control of the tomcat database pool. In the above example a database connection for mysql is configured. You’ve to place the jdbc-driver into the <tomcathome>/lib directory.