Java EE

J2EE Day 3

Installing Libraries in j2eedemo using maven step1

Open j2eedemo -> pom.xml

Right-click on j2eedemo Project ->
Maven -> Add Dependency

We need hibernate libraries

Search for hibernate-core and add the 4.2.4. Final [jar]

Installing Libraries in j2eedemo using maven step2

You will find that the pom.xml has changed and it now has the following entry :

```
<dependencies>
  <dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>4.2.4.Final</version>
  </dependency>
</dependencies>
```

Installing Libraries in j2eedemo using maven step3

Right-click on j2eedemo Project ->
Maven -> Add Dependency

We need hibernate-validator

Search for hibernate-validator and add the 5.0.1.Final [jar]

Installing Libraries in j2eedemo using maven step4

```
<dependencies>
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-core</artifactId>
		<version>4.2.4.Final</version>
	</dependency>
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-validator</artifactId>
		<version>5.0.1.Final</version>
	</dependency>
</dependencies>
```

Installing Libraries in j2eedemo using maven step5

Copy dependencies from ~/repos/stepin2it/july2013/j2eeprojects/tempfiles/pom_new.xml to

pom.xml

You will see 2 exceptions :
Oracle database driver
Primefaces libraries

Installing Libraries in j2eedemo using maven step6

Primefaces library is not included in the central maven repositories

We need to add additional repositories to our configuration so that maven can find Primefaces libraries and add to our classpath

Installing Libraries in j2eedemo using maven step7

Copy the following from pom_new.xml

```
<repositories>
    <repository>
        <id>prime-repo</id>
        <name>PrimeFaces Maven Repository</name>
        <url>http://repository.primefaces.org</url>
    </repository>
</repositories>
```

and into the pom.xml file

Save and check to see if the error for primefaces disappears

Installing Libraries in j2eedemo using maven step8

Oracle jdbc driver needs to be installed in the local maven repository manually

But, before we add Oracle jdbc driver, we need to add 1 more dependency

Installing Libraries in j2eedemo using maven step9

Right-click on j2eedemo project -> Maven -> Add Dependency -> Search for hibernate-entitymanager ->

Select org.hibernate -> 4.2.4.Final [jar] and click ok

Installing Libraries in j2eedemo using maven step10

Go to the folder where you saved ojdbc14.jar

~/repos/stepin2it/july2013/j2eeprojects/tempfiles/ojdbc14.jar to

Go to that folder in the command line and issue the following command

```
mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc14 -Dversion=10.2.0.1.0 -Dpackaging=jar -Dfile=ojdbc14.jar -DgeneratePom=true
```

Installing Libraries in j2eedemo using maven step11

You should see the following output

==> mvn install:install-file -DgroupId=com.oracle -DartifactId=ojdbc14 -Dversion=10.2.0.1.0 -Dpackaging=jar -Dfile=ojdbc14.jar -DgeneratePom=true

[INFO] Scanning for projects...

[INFO]

[INFO] --

[INFO] Building Maven Stub Project (No POM) 1

[INFO] --

[INFO]

[INFO] --- maven-install-plugin:2.4:install-file (default-cli) @ standalone-pom ---

[INFO] Installing /Users/das/repos/stepin2it/july2013/j2eeprojects/tempfiles/ojdbc14.jar to /Users/das/.m2/repository/com/oracle/ojdbc14/10.2.0.1.0/ojdbc14-10.2.0.1.0.jar

[INFO] Installing /var/folders/fr/sn4l2f393_v_ytcwkb0lx7_40000gp/T/mvninstall5202840656100237522.pom to /Users/das/.m2/repository/com/oracle/ojdbc14/10.2.0.1.0/ojdbc14-10.2.0.1.0.pom

[INFO] --

[INFO] BUILD SUCCESS

[INFO] --

[INFO] Total time: 0.416s

[INFO] Finished at: Sat Aug 24 08:37:49 EDT 2013

[INFO] Final Memory: 3M/81M

Installing Libraries in j2eedemo using maven step12

Verify that the library is installed by following the path to the jar in the mvn output from the previous command as follows :

/Users/das/.m2/repository/com/oracle/ojdbc14/10.2.0.1.0/ojdbc14-10.2.0.1.0.jar

Installing Libraries in j2eedemo using maven step13

Let’s update maven dependencies by

Right-click on project -> Maven -> Update Project -> Click ok

The error disappeared which means that the library was found in our local maven repository and got added to the classpath

Now check the maven dependencies tree and see that the oracle jdbc database driver is included

All dependencies are in the local .m2 folder and driven by maven

Installing Libraries in j2eedemo using maven step14

There’s still a lot more configuration which can be created in the pom.xml file

If we click ctrl-space, there are a lot of tags available, some of which we won’t use

The most important one is the build tag where we will define the build configuration

Installing Libraries in j2eedemo using maven step15

Build the project with all the dependencies folder

Notice that in the target -> com.example. j2eedemo -> WEB-INF folder, there is no lib folder

After we launch the build, we should see the lib folder in the WEB-INF folder and it should contain all the libraries we just added to our project

Installing Libraries in j2eedemo using maven step16

We need to do a build now ->

Right-click on project -> Maven -> Maven install

Refresh project and open target folder

All the jar files will be packaged in the target folder. We can browse the jars on the file-system

Installing Libraries in j2eedemo using maven step17

This application is now standalone and all the libraries will be able to be deployed on the application server

We have to do configuration next.

We didn’t have to look for libraries manually, download them and add them manually to the classpath.

The war size is now pretty heavy (~21 MB)

Installing Libraries in j2eedemo using maven step18

If you open pom.xml file in Maven POM editor, you can look for dependency hierarchy

Effective pom is a lot heavier than the pom. xml that we added

Right-click on JRE System Library -> Properties and change it to JavaSE-1.6 version

This should resolve warnings

Installing Libraries in j2eedemo using maven step19

Open src -> main -> webapp -> WEB-INF -> web.xml

We need to configure tomcat application server using the web.xml file

Installing Libraries in j2eedemo using maven step20

Copy the following into the web.xml file :

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext.xml</param-value>
</context-param>
```

This param is the main spring framework config file and it will hold the Spring javabean definitions

Installing Libraries in j2eedemo using maven step21

```
<context-param>
    <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
    <param-value>.xhtml</param-value>
</context-param>
```

That is the file extension that will hold the java serverfaces views

Installing Libraries in j2eedemo using maven step22

```
<context-param>
    <param-name>facelets.DEVELOPMENT</param-name>
    <param-value>true</param-value>
</context-param>
```

This enables development mode for facelets and this will help with debugging and also add some more features for development.

Installing Libraries in j2eedemo using maven step23

```
<context-param>
    <param-name>javax.faces.FACELETS_REFRESH_PERIOD</param-name>
    <param-value>1</param-value>
</context-param>
```

This config sets the facelets refresh period to 1 second.

Installing Libraries in j2eedemo using maven step24

Next, we need to add listeners, required spring servlets and filters.

```
<listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
```

This is a Spring listener class that is responsible for context loading

Installing Libraries in j2eedemo using maven step25

Now, we need to configure servlets

These are the resources servlet which resolve resources and provide the mapping for resources.

```
<servlet>
        <servlet-name>Resources Servlet</servlet-name>
        <servlet-class>org.springframework.js.resource.ResourceServlet</servlet-class>
        <load-on-startup>0</load-on-startup>
</servlet>

<servlet-mapping>
        <servlet-name>Resources Servlet</servlet-name>
        <url-pattern>/resources/*</url-pattern>
</servlet-mapping>
```

Installing Libraries in j2eedemo using maven step26

The Spring Dispatcher Servlet handles incoming and outcoming requests and responses to and from the application and then resolves them to the correct modules and services.

Even though we are using Spring, Spring uses Servlets as the engine and for the operations.

This is the entry point for the Java web application

```
<servlet>
    <servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value></param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

The following provides the mapping for the application and all requests and responses go through DispatcherServlet

```
<servlet-mapping>
    <servlet-name>Spring MVC Dispatcher Servlet</servlet-name>
    <url-pattern>/app/*</url-pattern>
</servlet-mapping>
```

Installing Libraries in j2eedemo using maven step27

Add the Faces Servlet :

```
<servlet>
        <servlet-name>Faces Servlet</servlet-name>
        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
        <servlet-name>Faces Servlet</servlet-name>
        <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

Installing Libraries in j2eedemo using maven step28

These are encoding filters that application will use

```
<filter>
        <filter-name>charEncodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
  <param-name>encoding</param-name>
  <param-value>UTF-8</param-value>
</init-param>
<init-param>
  <param-name>forceEncoding</param-name>
  <param-value>true</param-value>
</init-param>
  </filter>

<filter-mapping>
  <filter-name>charEncodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Installing Libraries in j2eedemo using maven step29

We will add additional spring security filters later when required

```
<!-- Spring security filters -->
```

Installing Libraries in j2eedemo using maven step30

Save the web.xml file and close it.

In src -> main -> webapp -> WEB-INF

Right-click -> File -> New -> Other -> XML -> XML File
and call it applicationContext.xml

Finish

Installing Libraries in j2eedemo using maven step31

In applicationContext.xml, add the following xml namespaces in it

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

</bean>

We will be adding all the beans that we use in the application, as well as bean configuration and hibernate configuration.

Installing Libraries in j2eedemo using maven step32

We need to import all the bean configuration files in the applicationContext as follows :

```
<import resource="datasource-config.xml" />
```

Create this file in WEB-INF with File -> New -> XML File -> datasource-config.xml

Installing Libraries in j2eedemo using maven step33

In datasource-config.xml -> Add the following

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.
org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/tx http://www.springframework.
org/schema/tx/spring-tx.xsd
                http://www.springframework.org/schema/context http://www.springframework.
org/schema/context/spring-context-3.1.xsd">
    <context:property-placeholder location="classpath:datasource.properties" />
    <bean id="dataSource" class="oracle.jdbc.pool.OracleDataSource"
            destroy-method="close">
            <property name="connectionCachingEnabled" value="true" />
            <property name="URL" value="jdbc:oracle:thin:@localhost:1521:XE" />
            <property name="user" value="user" />
            <property name="password" value="pass" />
    </bean>
```

Installing Libraries in j2eedemo using maven step34

datasource-config.xml

you will see the bean has a unique id identifier

The javabean class has the classname and some additional attributes

It’s injecting the values into the OraclDataSource class with public getters and setters

The OracleDataSource connectionCachingEnabled property is set to true

The next 3 properties are connections strings

Installing Libraries in j2eedemo using maven step35

This will inject the bean into the dataSource bean, which is known as bean injection.

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.
LocalContainerEntityManagerFactoryBean">
    <property name="jpaVendorAdapter">
        <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
            <property name="showSql" value="true" />
            <property name="generateDdl" value="true" />
            <property name="databasePlatform" value="org.hibernate.dialect.Oracle10gDialect" />
        </bean>
    </property>

    <property name="dataSource" ref="dataSource" />
</bean>
```

- Every SQL operation will be logged to the console and what queries it has executed. It can be disabled in production.

- The databasePlatform variable holds the SQL dialect to use.

Installing Libraries in j2eedemo using maven step36

All SQL databases use transactions to preserve the atomicity of operations.

We define a bean called transactionManager and pass the reference to our dataSource as well as the entityManagerFactory defined above

```
<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="dataSource" ref="dataSource" />
        <property name="entityManagerFactory" ref="entityManagerFactory" />
    </bean>
```

In the above, we are passing complex data structures, i.e. a reference to dataSource

Installing Libraries in j2eedemo using maven step37

We need to manage transactions using annotations so we add the following configuration :

<tx:annotation-driven transaction-manager="transactionManager" />

Finally, we add the following bean :

<bean class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor" />

Installing Libraries in j2eedemo using maven step38

Save the datasource config and proceed with webflow config by adding the following to applicationContext.xml :

<import resource="webflow-config.xml" />

Installing Libraries in j2eedemo using maven step39

Right-click on WEB-INF -> File -> New -> XML File -> webflow-config.xml

Add the following namespaces :

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:webflow="http://www.springframework.org/schema/webflow-config"
        xmlns:faces="http://www.springframework.org/schema/faces"
        xsi:schemaLocation="
                http://www.springframework.org/schema/beans
                http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/webflow-config
                http://www.springframework.org/schema/webflow-config/spring-webflow-config.xsd
                http://www.springframework.org/schema/faces
                http://www.springframework.org/schema/faces/spring-faces.xsd">
```

Installing Libraries in j2eedemo using maven step40

Add the following listener bean. This defines to which constructor we pass the entityManagerFactory and the transactionManager factory that we created previously :

```
<bean id="jpaFlowExecutionListener" class="org.springframework.webflow.persistence.JpaFlowExecutionListener">
        <constructor-arg ref="entityManagerFactory" />
        <constructor-arg ref="transactionManager" />
</bean>
```