Setting up a Maven Project

This documentation describes how to set up a Maven project for CaptainCasa, using Eclipse as development environment.

In principal things are really simple... - at least from structural point of view. But to actually do it means some kind of little nightmare: as usual you stumble over the small, but annoying issues.

Overview

Dependencies

From a project's point of view a CaptainCasa Enterprise Client project is a web project. So the goal is to create some ".war"-structure.

CaptainCasa has two relevant artifacts:

- eclntjsfserver.jar This is the jar file that you require for developing your server side classes. This jar contains the server-side components, the PageBean-management, the Workplace-management, etc. etc. So this jar is required from your side to build your Java-classes that are the ones you plug to the dialog processing. The jar is one to be part of WEB-INF/lib within the .war structure.
- webappaddons This is a directory with non-Java files. It contains the standard .jsp dialog definitions (e.g. for OK, YesNo-Popup, Workplace-dialgos, etc.), images for these standard dialogs and it contains all the client processing, which for the RISC-HTML client is all the JavaScript that is later on executed on client side. All this content is not relevant for developing your Java-classes, but it is relevant to run CaptainCasa Enterprise Client as part of your web application.

As consequence there are two CaptainCasa artifacts that you depend from: a compile time dependency for the eclntjsfserver.jar file, and an install time dependency to the webappaddons-content.

Project <==> Deployment Tomcat

By default the CaptainCasa installation comes with an own Tomcat instance running the application that you build. And it comes with some tooling (Layout Editor), that manages this Tomcat:

• Within the tools you "reload" (or "hot deploy") the application. This means that actually the result of your project is copied into the corresponding Tomcat/webapps-application and that the application somehow is restarted so that changes take effect.

Of course you may also think about running some Tomcat instance which is managed by your development environment (here: Eclipse). But this is not the CaptainCasa-default scenario, so this documentation assumes to develop your project on the left side - and run your project in some Tomcat on the right side, having the CaptainCasa toolset as linkage between.

In other words:

• Maven/Eclipse are the ones to produce the ".war"-like structure into the target directory of the project.

• The CaptainCasa toolset is the one to take over this data into the Tomcat runtime.

Creating and configuring the Eclipse project

Create the Maven project in Eclispe

The project is created in Eclipse. Select File => New => Project...:

💭 Java - Eclipse					
File Edit Navigate Search	Project Run	Window Hel	p		
New		Alt+Shift+N	> 🖄	Java Project	
Open File			Ľ	Project	
0		<u></u>	et de la		20
Select "Maven Pro	oject":				
💽 New Project	R		_	D X	
Select a wizard					
Create a Maven Project					
					_
<u>W</u> izards:					
> 🗁 JavaFX				^	
> 🗁 JavaScript					
> 🗁 JAXB					
> 🗁 JPA					
🗸 🗁 Maven					
📰 Check out Maven Proj	ects from SCM				
Maven Module					
Maven Project					
> 🔁 Plug-in Development					
> B SVN					
> 🥟 Web					
> C Atenu				¥	
? < <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish		Cancel	
-					J

Store the project in the default workspace:

🔘 New Maven Project		-	□ ×
New Maven project Select project name and location			M
Create a <u>s</u> imple project (skip arc	hetype selection)		
Use default Workspace location			
Location:		~	Brows <u>e</u>
☐ <u>A</u> dd project(s) to working set			
Working set:			Mor <u>e</u>
 Advanced 			
	2		
?	< <u>B</u> ack <u>N</u> ext >	<u>F</u> inish	Cancel

Select "maven-archetype-webapp" as archetype:

🔘 New	Maven Project			
New Ma	iven project			-
Select ar	Archetype			M
Ca <u>t</u> alog:	Internal		~	Configure
<u>F</u> ilter:				×
Group I	d	Artifact Id	Version	^
org.apa	che.maven.archetypes	maven-archetype-quickstart	RELEASE	
org.apa	che.maven.archetypes	maven-archetype-site	RELEASE	
org.apa	che.maven.archetypes	maven-archetype-site-simple	RELEASE	
org.apa	che.maven.archetypes	maven-archetype-webapp	RELEASE	
org.apa	che.maven.archetypes	softeu-archetype-jsf	RELEASE	
org.apa	che.maven.archetypes	softeu-archetype-seam	RELEASE	
org.apa	che.maven.archetypes	softeu-archetype-seam-simple	RELEASE	~
A simple	e Java web application			^
▶ Ad <u>v</u> an	the last version of Archetype only	r ∐Include snapshot archetypes	Add	Archetype
?		Back Next > Einish		Cancel

Define the Maven group id and the Maven artifact id. The artifact id will be the default name of your web-application and the name of the corresponding CaptainCasa project. So if you define "mavendemo" here, then this is the name of the project within the CaptainCasa toolset, and this is the default name of your web-application within your Tomcat ("localhost:50000/mavendemo").

🔘 New Ma	aven Project						\times
New Mav Specify Are	en project chetype paran	neters	la l			M	1
Group Id:	my.test						~
Artifact Id:	mavendemo						~
Version:	0.0.1-SNAPS	нот	~				
Package:	my.test.mav	endemo					~
Properties a	vailable from	archetype:					
Name		Value				<u>A</u> er	id nove
Advance	:d		< <u>B</u> ack	<u>N</u> ext >	Einish	Cance	I

Setting up Maven

The project contains some pom.xml:



Update the POM to be:

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd"> <modelVersion>4.0.0</modelVersion> <groupId>my.test</groupId> <artifactId>mavendemo</artifactId> <packaging>war</packaging>
<version>0.0.1-SNAPSHOT</version>

```
<name>mavendemo Maven Webapp</name>
<url>http://maven.apache.org</url>
<repositories>
           <repository>
                      <id>org.eclnt</id>
                       <url>http://www.captaincasademo.com/mavenrepository</url>
</repository>
<dependencies>
           <dependency>
                      <proupId>junit</groupId>
<artifactId>junit</artifactId>
<version>3.8.1</version>
                       <scope>test</scope>
           </dependency>
           <dependency>
                      <groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>3.1.0</version>
<scope>provided</scope>
           </dependency>
<dependency>
                      <proupId>org.eclnt</proupId>
<artifactId>eclntjsfserver</artifactId>
<version>20161031</version>
           </dependency>
           <dependency>
                      <groupId>javax.faces</groupId>
<artifactId>jsf-api</artifactId>
<version>2.2.14</version>
       <scope>provided</scope>
           </dependency>
           <dependency>
                      <groupId>org.glassfish</groupId>
<artifactId>javax.faces</artifactId>
<version>2.2.14</version>
<vided</score>
       <scope>provided</scope>
           </dependency>
</dependencies>
<build>
           <finalName>mavendemo</finalName>
           <plugins>
                      <plugin>
                                  <proupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>3.1</version>
                                  <configuration>
                                             <source>1.6</source>
<target>1.6</target>
                                  </configuration>
                      </plugin>
                           - Ŭnpack webapp addons into webcontentcc -->
                       <plugin>
                                  <proupId>org.apache.maven.plugins</proupId>
<artifactId>maven-dependency-plugin</artifactId>
                                  <executions>
                                             <execution>
                                                        <id>unpack</id>
                                                         <phase>package</phase>
                                                         <goals>
                                                                   <goal>unpack</goal>
                                                        </goals>
                                                         <configuration>
                                                                   <artifactItems>
                                                                               <artifactItem>
<proupId>org.eclnt</proupId>
<artifactId>eclntwebappaddonsRISC</artifactId>
<version>20161031</version>
<outputDirectory>webcontentcc</outputDirectory>
                                                                               </artifactItem>
                                                                    </artifactItems>
                      </con
</execution>
</plugin>
<!-- مربع
                                                         </configuration>
                             Add webcontentcc directory to the .war file -->
                      <1
                       <plugin>
                                  <proupId>org.apache.maven.plugins</groupId>
<artifactId>maven-war-plugin</artifactId>
<version>2.6</version>
                                  <configuration>
                                             <webResources>
                                                        <resource>
                                                                   <directory>webcontentcc</directory>
                                                         </resource>
                                             </webResources>
                                  </configuration>
                      </plugin>
```

```
Maven
```

</plugins> </project>

What is the basic content within the pom.xml?

- You define the repository location, where CaptainCasa provides its artifacts: http://www.captaincasademo.com/mavenrepository
- You define some dependency to servlet-API, JSF-API and JSF-implementation. Please note: all of them are defined with scope "provided". The servlet-API comes with the Tomcat you deploy to, the JSF-API and JSF-implementations come with CaptainCasa and are part of the "webappaddons" package.
- The plugin content is:
 - "maven-compiler-plugin": You define that the project is sing Java version "1.6".
 - "maven-dependency-plugin": You define that the webappadons-part of CaptainCasa Enterprise Client is copied in the "webcontentcc" directory of your project.
 - "maven-war-plugin": You define that the content of the project's "webcontentcc" directory is copied into the .war-structure.

Now you create this "webcontentcc" folder, so open the corresponding menu...

✓ ₩ mavend	emo			4» <groupia>my.</groupia>
🕮 s	New	>	鬯	Java Project
> =\ 1	Go Into			Project
> 🛋 r > 🦢 s	Open in New Window		₿	Package
> 🗁 t	Open Type Hierarchy	F4	G	Class
E E E	Show In	Alt+Shift+W >	Ø	Interface
testr	Conv	Ctrl+C	G	Enum
E	Copy Qualified Name		@	Annotation
	Paste	Ctrl+V	₽3	Source Folder
	Delete	Delete	惨	Java Working Set
			C	Folde
<u>6</u>	Remove from Context	Ctrl+Alt+Shift+Down	F\$	Eil-

...and create the folder:

💓 New Folder	5	_		×
Folder Create a new folder resource.				
Enter or select the parent folder:				
mavendemo				
Image: Second	C:/bmu_jtc/svrepository/ecl C:/bmu_jtc/svrepository/ec mu_jtc/svrepository/eclnt: ///c:/bmu_jtc/svrepository/eclnt: tru 411 [file:///C:/bmu_jtc/svr /bmu_jtc/svrepository/ecln :/bmu_jtc/svrepository/ecln	Int, Trunk: Int, Trunk: trunk] /ecInt, Tru ink] epository/ t: trunk] nt: trunk]	ecInt_clie ecInt_dat nk: ecInt_r ecInt, Trui	ntui ^ apo ecli; nk: : >
Folder name: webcontentcc				
Advanced >>				
?	<u> </u>	ish	Canc	el

And now, finally you can first time execute the Maven build to check if things come together as planned. So call "Maven => Update project..." first:

✓ ₩ mave		New	>	Ì.	5» <artifactid>maxendemo<!--</th--></artifactid>
> 🛋 JF		Go Into		ι.	7 » <version>0.0.1-SNAPSHOT</version>
> 🛋 N		Onen in New Window		1	8 » <name>mavendemo·Maven·W</name>
🗸 🇁 sr		Open in New Window		ι.	9» <url>http://maven.apach</url>
> 🖻		Open Type Hierarchy	F4	ι.	10 ···· <repositories></repositories>
> 🍋 ta		Show In	Alt+Shift+W >	ι.	12
M P	m.	Comu	Chill C	1	13 ·····
📋 testm		Сору	Ctri+C	ι.	14 ·····
	1	Copy Qualified Name		ι.	15
	ß	Paste	Ctrl+V	ι.	16 » <dependencies></dependencies>
	×	Delete	Delete	ι.	18 » » saroupId>junit<
					19» » » <artifactid>iun</artifactid>
	<u>.</u>	Remove from Context	Ctrl+Alt+Shift+Down	ι.	20 » » » <version>3.8.1<</version>
		Build Path	>	ι.	21 » » » <scope>test</scope>
		Source	Alt+Shift+S >	ι.	22 » » //dependency>
		D.C.	All CLID TA	ι.	24 sourcependencysh
		Refactor	Alt+Shift+1 >		25 ····· <artifactid>jav</artifactid>
	e na	Import		ι.	26 ·····
	5.0	Export		ι.	27 ····· <scope>provided</scope>
	-	ciportal			28 · · · · · · 1
	÷	Refresh	F5	ι.	30 ·····saroupId>org.ec
		Close Project		ι.	31 ····· <artifactid>ecl</artifactid>
<		Close Uprelated Projects			> 32 ·····
di conuiu		close officiated Projects		F	33 ····· 1
CC Plojec		Assign Working Sets		E	34 ×
Start		Profile As	>		36 » <finalname>mavendem</finalname>
Scanning pr		Debug Ar	,	E	37 ····· <plugins>1</plugins>
Scanning pr		Debug As		ι.	38 · · · · · · · · · · · · · · Unpack web
Scanning pr		Run As	>	ι.	39 · · · · · <plugin>1</plugin>
Scanning pr		Validate		ι.	40
Scanning pr		Team	>	ι.	42 ····· <executions< td=""></executions<>
Scanning pr		Compare With	>	ι.	43 ····· <execut< td=""></execut<>
Scanning pr		Parters from Land History		ι.	<
Scanning pr	_	Restore from Eocal History		1	real and the second
Scanning pr		Maven	>		Add Dependency
Scanning pr		Java EE Tools	>		Add Plugin pr
Scanning pr		Configure	>	M	New Maven Module Project
Scanning pr Scanning pr		Properties	Alt+Enter		Download JavaDoc
Scanning pr	oject	s (10:17:16)			Download Sources
Scanning pr	oject	s (10:17:14) (10:17:13)		49	Update Project Alt+F5
Scanning pr	ojecti	s (10:17:10)		- M	6
scanning pr	-jett				Salart Mavan Drofilar

And then start the Maven build by right clicking onto the project and selecting "Run => Maven build..":

🗸 🔁 maye	-				4 × <groups< th=""><th>u>my.cesc</th></groups<>	u>my.cesc
i 🕮 si		New	>	1	6 snackar	ing wars (nack aging >
		Go Into			7 » <versio< th=""><th>n>0.0.1-SNAPSHOT</th></versio<>	n>0.0.1-SNAPSHOT
					8 » <name>r</name>	navendemo Maven Weba
		Open in New Window			9» <url>ht</url>	tp://maven.apache.o
		Open Type Hierarchy	F4		10 · · · · < reposi	tories>¶
		Channels .	Alto Children Mark		11 · · · · · <re< th=""><th>pository>¶</th></re<>	pository>¶
M P		snow in	Ait+Shift+W >		12	<pre><1d>org.eclnt<!--1d--></pre>
🔲 testn	Ð	Copy	Ctrl+C		14	epositorys
	Ra	Come Overlife at Name			15 ···· repos</th <th>itories></th>	itories>
		Copy Qualified Name			16 » <depend< th=""><th>lencies>¶</th></depend<>	lencies>¶
	Ē	Paste	Ctrl+V		17 » » <de< th=""><th>ependency>¶</th></de<>	ependency>¶
	×	Delete	Delete		18 » » »	<pre><groupid>junit</groupid></pre>
					19» » »	<artifactid>junit<</artifactid>
	<u>.</u>	Remove from Context	Ctrl+Alt+Shift+Down		20	<version>3.8.1</version>
		Build Path	>		22 8 8/1	lependency>
		Source	Alt+Shift+S >		23 · · · · · <de< th=""><th>ependency>1</th></de<>	ependency>1
			AB (110 T)		24	<pre><groupid>javax.ser</groupid></pre>
		Refactor	Ait+Shift+1 >		25 · · · · · · · ·	<pre><artifactid>javax.</artifactid></pre>
	240	Import			26 · · · · · ·	<version>3.1.0</version>
		Emert			27	<scope>provided</scope>
	229	Export			28	rependency>1
	S.	Refresh	F5		30	<pre>spendency>s <<pre>saroupId>org.eclpt</pre></pre>
	×.	Class Designt			31	<pre><artifactid>eclnti</artifactid></pre>
<		Close Project)	32	<pre><version>20161024<</version></pre>
(1981 - 1991 - 19		Close Unrelated Projects			33 · · · · · 0</td <td>lependency>1</td>	lependency>1
CC Project		Assign Working Sets		12	34 » <th>idencies>¶</th>	idencies>¶
Start					35 × <00100	nol Namas may on domos (
c		Profile As	>		37	ugins>
Scanning pr		Debug As	>	ι	38	Unnack webapp</th
Scanning pr		Run As	>	×1	1 Run on Server	Alt+Shift+X, R
Scanning pr		Validate		2	2 Java Applet	Alt+Shift+X, A
Scanning pr		Team	>		3 Java Application	Alt+Shift+X, J
Scanning pr		Compare With	>	Ju	4 JUnit Test	Alt+Shift+X, T
Scanning pr		Restore from Local History		m2	5 Maven build	Alt+Shift+X, M
Scanning pr		Mayen	>	m2	6 Mayen build	
					2	-

In the dialog that pops up define "clean install" as goal:

dit configu	uration and launch.			
ame: mave	ndemo			
Main 🛛 🛛	🛦 JRE 🧽 Refresh 🦆 Source 📧 Enviror	nment 🔲 <u>C</u> ommon		
Base directory	r.			
C:/bmu_jtc/	eclipse/workspace/mavendemo			
		Browse Workspace	Browse File System	<u>V</u> ariables
<u>G</u> oals: Profiles:	clean install			Select
Jser settings:	Offline Undate Spanshots			File
	Debug Output Skip Tests N Resolve Workspace artifacts <u>I</u> V <u>I</u> hreads	lon-recursive		
Parameter N	Debug Output Skip Tests N Resolve Workspace artifacts	ion-recursive		<u>A</u> dd E <u>d</u> it <u>R</u> emove
Parameter N Maven Runtjr	Debug Output Skip Tests N Resolve Workspace artifacts Value Value Ne EMBEDDED (3.2.1/1.5.1.20150109-1819)	on-recursive	~	Add Edit Remove

Now press "Run". You may have to wait a bit of time when executing the build the first time because the CaptainCasa artifacts are loaded into your Maven repository in the background.

After the build you may want to see the results... So refresh your project first:



The project file structure now looks like:



You see that the webcontentcc directory now was populated and contains the webappadons-content of CaptainCasa Enterprise Client.

You also see dome error indication within the "target" directory. The reason is that all JavaScript files are validated within this directory - and this validation is also applied to the ones coming from CaptainCasa. Well, these files are valid - but they are shrinked and cryptified when they are part of the CaptainCasa delivery, which makes them appear as error-files within the target-directory.

As consequence, take out the corresponding directory out of the list of JavaScript source directories within the project. Open the project properties...



...select section "JavaScript => Include path" and remove the corresponding directory:

ype filter text	Include Path	↓ → ⇒ +
ype filter text > Resource Builders Deployment Assembly Java Build Path > Java Build Path > Java Code Style > Java Godi Path > Java Godi Path > Java Godi Cocation > Code Style > Editor SCode Style > Editor > Editor > Kuldation > SP Fragment Maxen > Maxen	Include Path Include Path Source files and folders in the Global Scope: Source files and folders in the Global Scope	
Project Facets Project References Run/Debug Settings Server Service Policies Targeted Runtimes > Task Repository Task Tags > Validation Web Content Settings	<	> OK Cancel

Now the target should be cleared. Maybe you have to clean the project to apply the changes.

Updating Eclipse Project Setup

Before continuing with "nice" things, let's come to some special ones - which took us some time to find out, based on quite cryptic error messages that you see in the problem list of Eclipse...:

Update Project Facets

The default project facets have to be updated.

You normally may update the Eclipse project facets by some dialog which is part of the project-properties-setup, but we failed... So please open the facet configuration directly. It is part of the ".settings" of your project.



Please note: you need to switch into the "Navigator" view to see these files, you do not e.g. see them in the "Package Explorer".

Update the versions in the following way:

```
<?xml version="1.0" encoding="UTF-8"?>
<faceted-project>
   <fixed facet="wst.jsdt.web"/>
   <installed facet="java" version="1.6"/>
   <installed facet="wst.jsdt.web" version="1.0"/>
   <installed facet="jst.web" version="3.0"/>
   <installed facet="jst.jsf" version="2.2"/>
</faceted-project>
```

Clean the project ("Project => Clean..." in the top menu) after applying the changes.

Check Project's Java Version

Check that your project is using Java 1.6 (or higher). This is part of the project properties:



Creating the CaptainCasa Project

In order to use the CaptainCasa toolset for your project you need to create a CaptainCasa project definition, which tells the toolset where to find certain information it requires.

Examples:

- Within the CaptainCasa toolset dialog definitions (.jsp) are created. These are stored in the project, so CaptainCasa needs to know where to store.
- Within the toolset you reload the project, this means the toolset needs to know where the project result (".war"-structure) is located and where to copy it to.

Define .ccproject

This information is contained in the ".ccproject" file, which is part of your project's top directory. You now need to create this file, so in Eclipse select "New => File":

🗙 🔛 maven:				II 6ia	vaclassdirag
> > set	New		> 🛃	Java Project	Î
🗸 🇁 src	Go Into			Project	ŀ
× 🗁	Open in Ne	ew Window	ŧ	Package	
>	📄 Сору		G	Class	
🗸 🗁 tar	Paste		6	Interface	
e	💥 Delete		G	Enum	-
> 🗁	Move		e	Annotation	
	Rename		₩	Source Folder	F
> 🗁	import		4	🖇 Java Working S	et
WAR.	Export			Folder	
> 🗁 we			[File	N
.cc	Refresh		_		10

And then define ".ccproject" as file name:

New File	G₂				\times
File Create a new file resource.					
Enter or select the parent folder:					
mavendemo					
<pre>> Cont_datapool 2867 [file:///C:/bm > Cont_datapool 2867 [file:///C:/bm > Cont_demos 558 [file:///C:/bmu_j > Cont_eclipseplugin 972 [file:// > Cont_eclipseplugin 972 [file:///C:/bmu_j > Cont_eclipseplugin 972 [file://C:/bmu_j > Cont_eclipseplugin 972 [file://C:/bmu_j > Cont_eclipseplugin 972 [file:///C:/bmu_j > Cont_eclipseplugin 972 [file://C:/bmu_j > Cont_eclipseplugin 972 [file://cont_eclipseplugin 972 [file://cont_eclips</pre>	'bmu_jtc/svrepositor nu_jtc/svrepository/e (11 [file:///C:/bmu_jtc mu_jtc/svrepository/ omu_jtc/svrepository	y/eclnt, 1 :clnt: trun itory/ecln t: trunk] c/svrepos (eclnt: tru ı/eclnt: tr	Trunk: er ik] it, Trunk iitory/ec ink] unk]	cint_data : ecint_e :int, Trun	npo ^ clip k: +
File name: _ccproject					
?		<u>F</u> inish		Cance	ł

The file content is some simple XML definition:

<project

managedbycctoolset="false"
<pre>webcontentdirectory="\${project}/src/main/webapp" javasourcedirectory="\${project}/src/main/java"</pre>
javaclassdirectory="\${project}/target/classes"
<pre>webappaddonsdirectory="\${project}/webcontentcc"</pre>

```
Maven
```

</project>

When defining your own file, apply the following changes:

- Make sure to replace "mavendemo" with the name of your artifact id.
- Replace that the parameter "webcontentdeploydirectory" is pointing into your CaptainCasa Tomcat, which is located in the "<CaptainCasaInstallDir>/server/tomcat" directory.

Import project into CaptainCasa Toolset

Please note: the import functions are part of CaptainCasa 20161031. Update your CaptainCasa environment if using an earlier version.

Start your CaptainCasa toolset, and Select "File => Import project...".

File Documentation	0	
New project		
Import Project	Import project	
Refresh	Create basic project files	
Deploy		
Create new layout		
Recent layouts		
Cose all layouts	≓ 🗏 🔍	

Define the directory location of your project and press "Import":

Import project		×
Project Directory	I	U
	Import	

The project will be registered within the CaptainCasa toolset.

Create Project Basic Files

When using CaptainCasa Enterprise Client within an web application then there are certain items you need to do first:

- Update the WEB-INF/web.xml so that CaptainCasa filters and servlet are included.
- Add the WEB-INF/faces-config.xml to the project.
- Define a META-INF/context.file
- Start programming by typically using some Dispatcher-class.

In order to simplify the process of jump-starting the development there is a helper function:

File Documentation		o	
New project		Import project	
Refresh		Create basic project files	- 1
Deploy	Þ		
Create new layout			
Recent layouts Cose all layouts	Þ	≓ 🖻 🔍	

Select "File => Import Project => Create basic project files". A dialog will show up:

Create basic project files	×
When e.g. creating a project by using Maven then you need to define some default files to start development If having created the project with the CaptainCasa toolset, then the default files are automatically added to your project.	
✔ WEB-INF/web.xml	
WEB-INF/faces-config.xml	
META-INF/context.xml	
🗸 <sourcedir>/managedbeans/Dispatcher.java</sourcedir>	
Create default files	
Existing files will be replaced by the default files!	

For a "fresh" project select all the items and press "Create default files".

Refresh Eclipse Project

After registering the project in the CaptainCasa toolset and after adding the basic files, please refresh the project in the Eclipse environment.

....finished!

And, now you have reached the end of creating a Maven project! You now can develop/make your project with Eclipse/Maven and you can in parallel work with the CaptainCasa toolset editing dialogs (.jsp) and for deploying the application ("Reload") to the Tomcat-runtime.

Develop - Make - Deploy - Test

The environment that was created during all this procedure now is:



Tomcat runtime

- In the project the compilation and building of the web application is managed by Maven. The result is the target/mavendemo directory, in which the web application is produced.
- From the target/mavendemo directory the content of the web application is copied over into the Tomcat runtime using the "reload" (or "hot deploy") function with the CaptainCasa toolset.