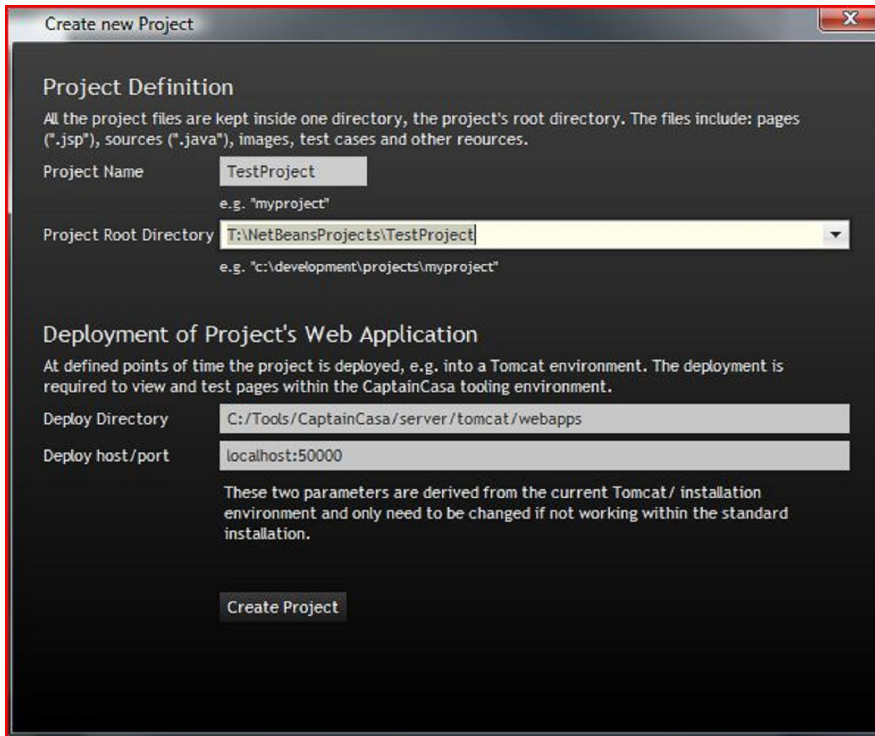


How to use CaptainCasa tools with Netbeans and Glassfishv3

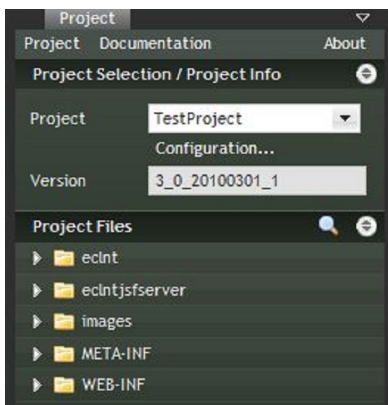
GaryHodgson, 14 March 2010 (created 13 March 2010)

Initial Steps

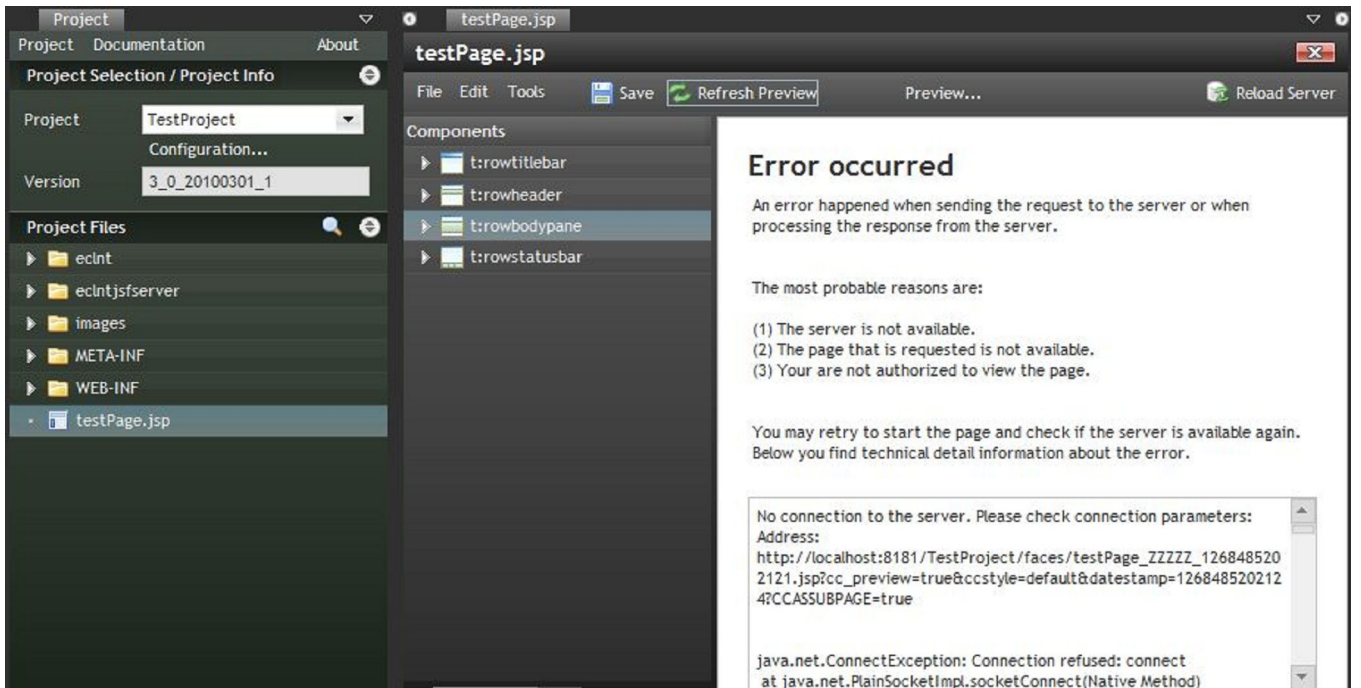
- Start CaptainCasa Development Tools
- Create Project in CaptainCasa
 - Recommended to create project in directory with no spaces, as later the unit test tool has a problem with paths that contain spaces.
 - For now leave the server details as default.



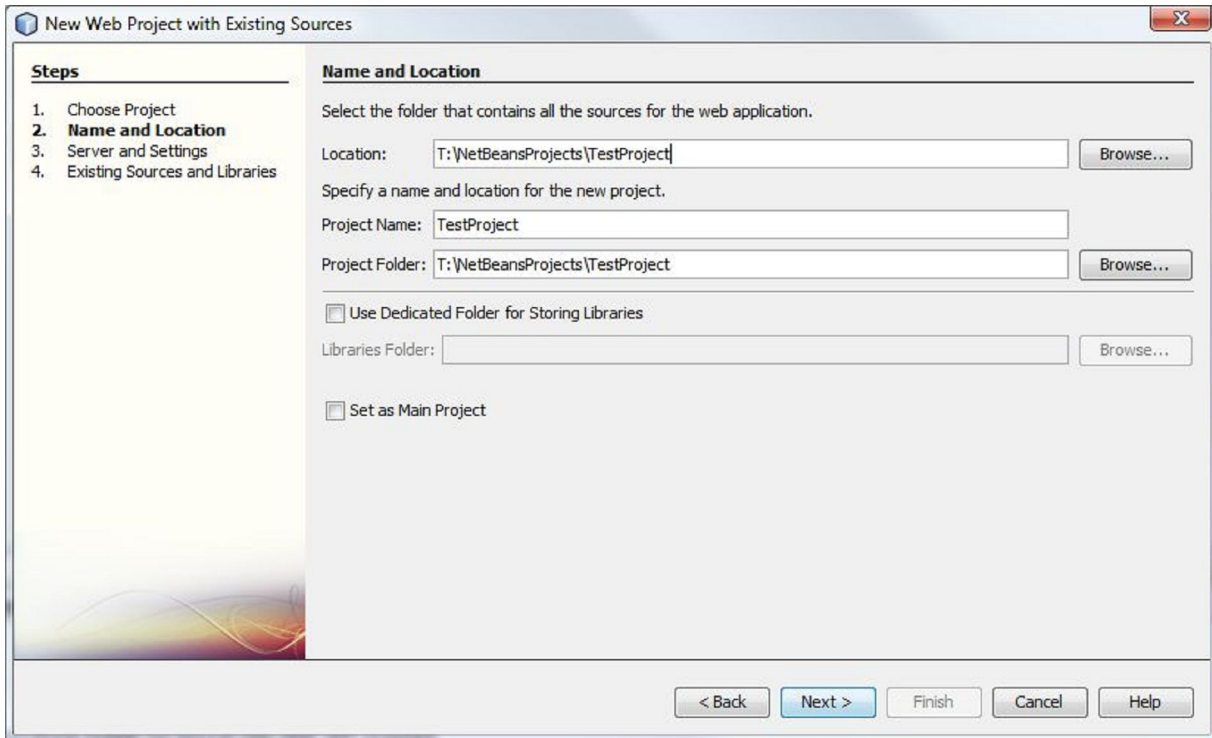
- Choose project in CaptainCasa Editor



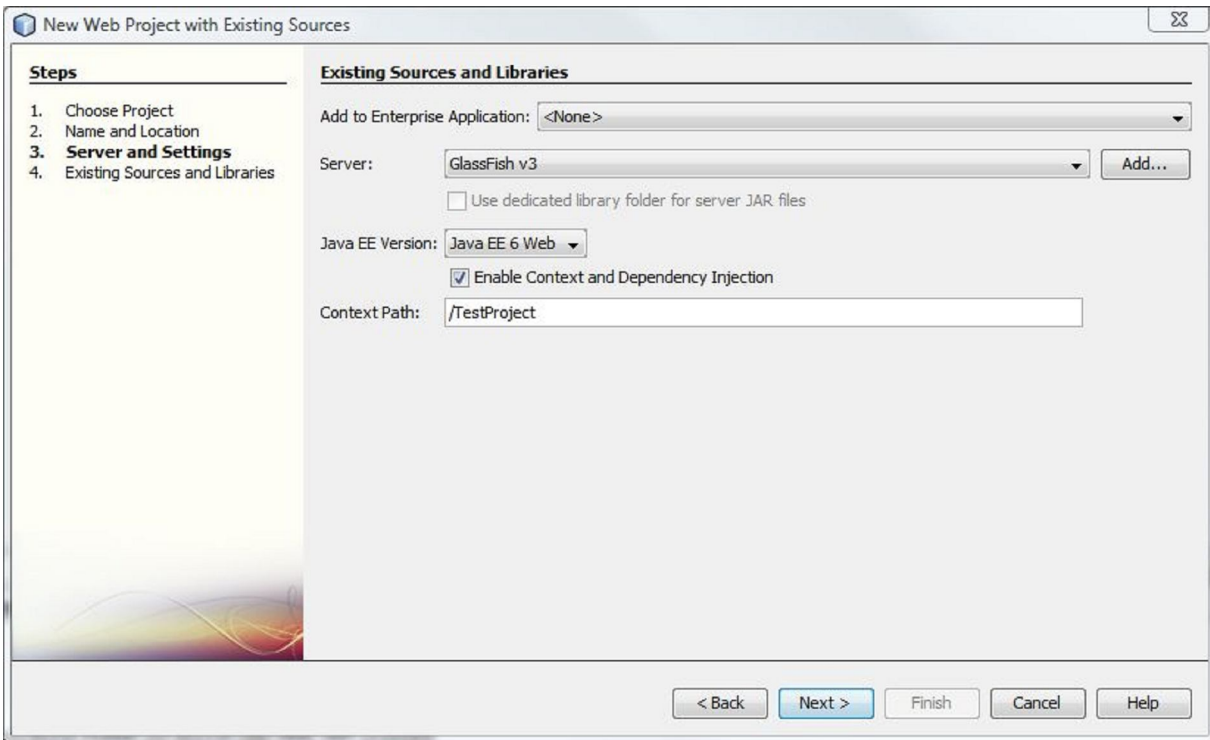
- Create JSP File
 - Note: When opening the test page an error will display because the server is not running.



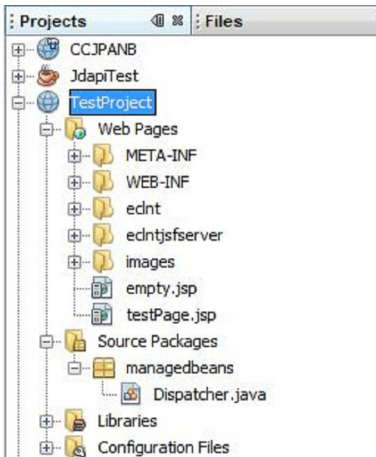
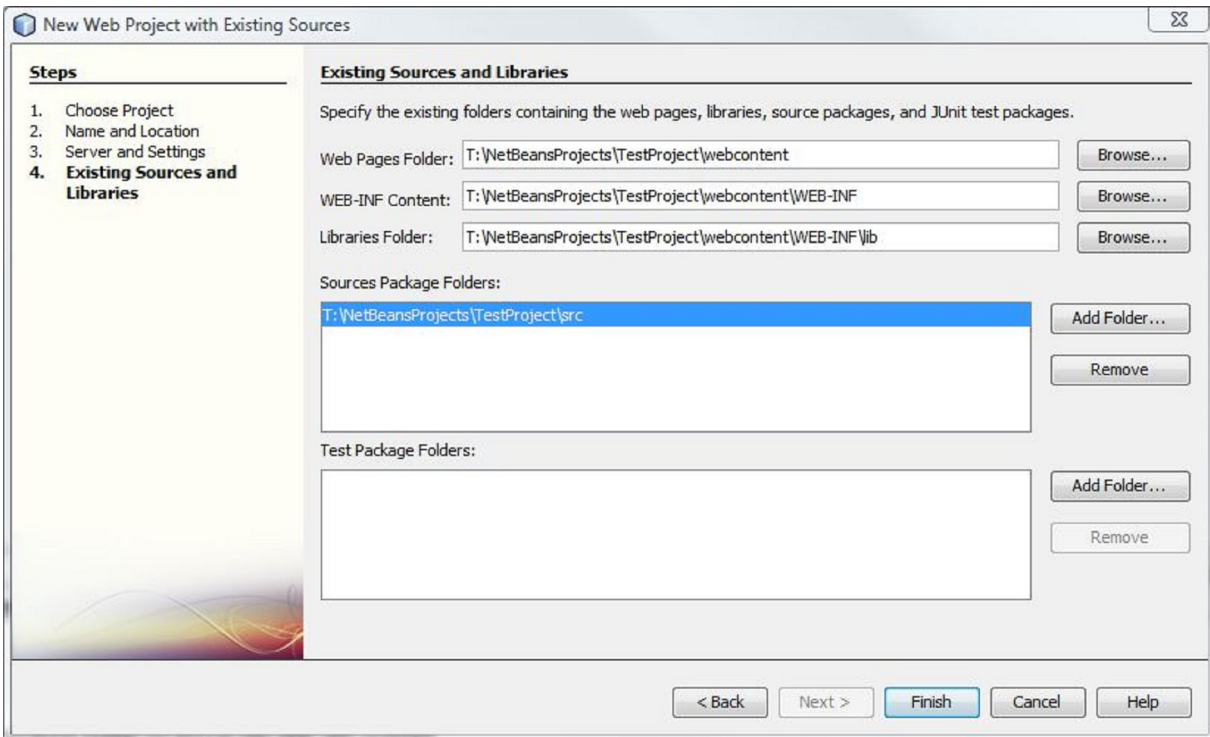
- Start Netbeans
- Create a new Project: Java Web Application with Existing Sources
 - Choose the location of the project created by CaptainCasa above



- Choose Glassfish v3 server (you may have to install and configure this if you have not already done so)



- Netbeans should pick up the correct folder structure



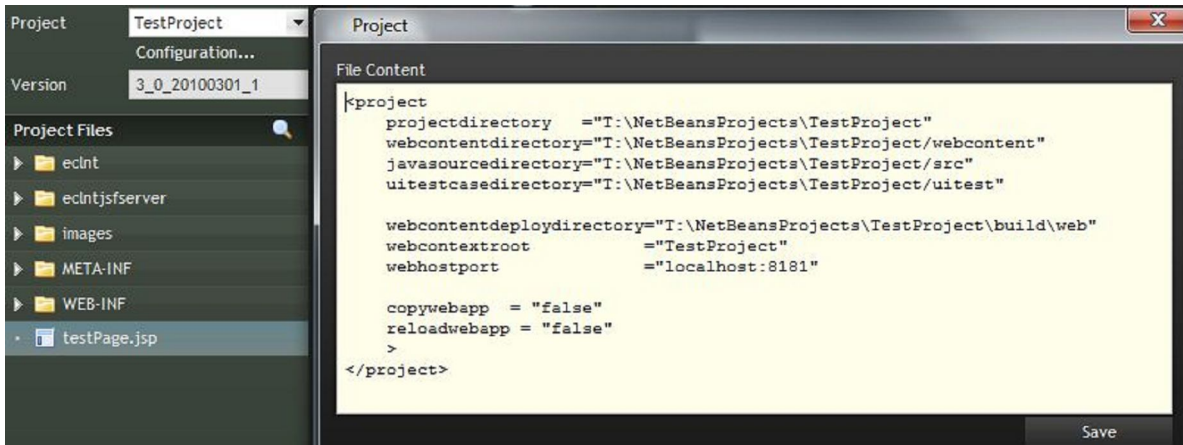
- Build and Deploy the app from within Netbeans. This will create the Netbeans build directory structure and ensure the app can be deployed on Glassfish. The **build** directory should be created, and the application successfully deployed. [008]

```

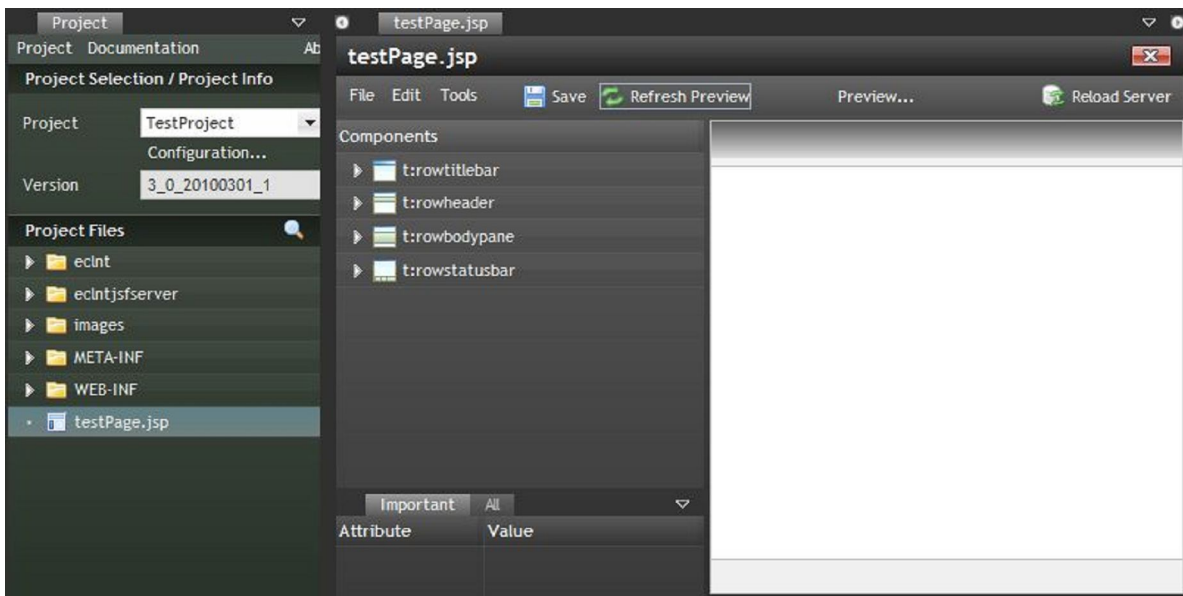
Output
Java DB Database Process  GlassFish v3 *  TestProject (run-deploy)
init:
deps-module-jar:
deps-ear-jar:
deps-jar:
library-inclusion-in-archive:
library-inclusion-in-manifest:
compile:
compile-jsp:
In-place deployment at T:\NetBeansProjects\TestProject\build\web
Initializing...
run-deploy:
BUILD SUCCESSFUL (total time: 6 seconds)

```

- Return to CaptainCasa Editor, and open the Project Setup Configuration Options



- Set `webhostport` to the Glassfish host and port
- Set `webcontentdeploydirectory` to the `web` directory within the Glassfish build directory
- Set `reloadwebapp` to **false**
- Set `copywebapp` to **false**
- Important:** Choose 'Refresh' from the Project menu
- Reload the JSP file in the CaptainCasa Editor, and the jsp preview should display



Keeping the build environment clean

- The CaptainCasa Editor creates a temporary version of each JSP for the preview (indicated by the filename containing 'ZZZZZ' and a timestamp (e.g. testPage_ZZZZZ_1268477933315.jsp). This should be removed from the build directory before packaging the app for distribution.
- The CaptainCasa Editor expects to find the classes of ManagedBeans in the following location: `TestProject\webcontent\WEB-INF\classes`, however Netbeans stores these in the following location: `TestProject\build\web\WEB-INF\classes`. Therefore we add a task to the Ant build script to copy these files across after each build command.
- Sometimes the CaptainCasa Editor classes are out of sync with the state in Netbeans, so add an explicit command to copy across the classes.
- Add to build.xml, after `<import file="nbproject/build-impl.xml" />`:

```
<!-- Removes temporary CaptainCasa JSPs and classes located in the webcontent
```

```

    directory before creating the war file for distribution -->
<target name="-pre-dist">
  <delete includeEmptyDirs="true" quiet="true">
    <fileset dir="${build.web.dir}" includes="*_ZZZZZ_*.jsp"/>
  </delete>
  <delete dir="${webinf.dir}/classes"/>
</target>

<!-- Removes temporary CaptainCasa classes -->
<target name="-post-clean">
  <delete dir="${webinf.dir}/classes"/>
</target>

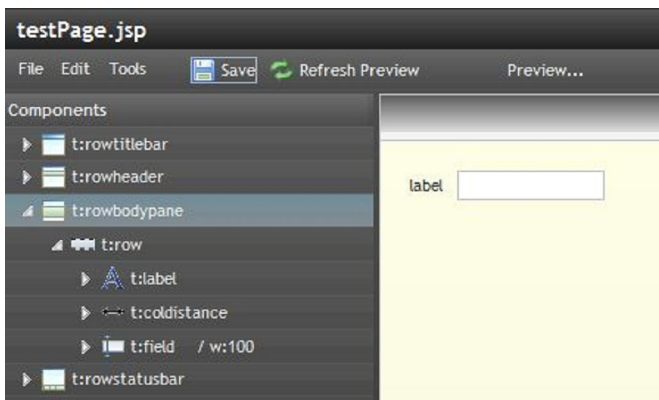
<!-- Copies temporary classes for use by CaptainCasa Editor -->
<target name="-post-run-deploy">
  <copy todir="${webinf.dir}/classes">
    <fileset dir="${build.web.dir}/WEB-INF/classes"/>
  </copy>
</target>

<!-- Explicitly copies temporary classes for use by CaptainCasa Editor -->
<target depends="-init-project" name="sync-captain-casa">
  <copy todir="${webinf.dir}/classes">
    <fileset dir="${build.web.dir}/WEB-INF/classes"/>
  </copy>
</target>

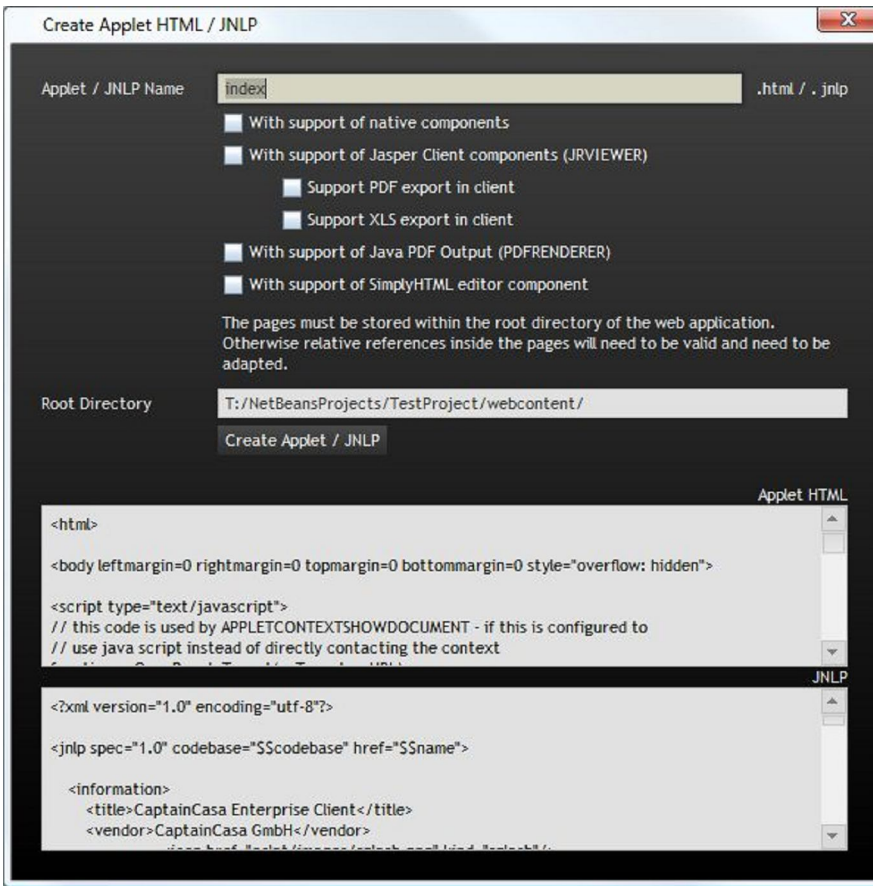
```

Opening the page in Glassfish

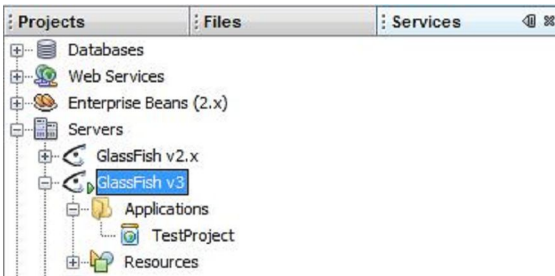
- Create some content in the JSP using the Editor (I found I had to add at least one item to the JSP in order for the page to display when deployed on Glassfish in later steps, otherwise it threw a Null Pointer Exception on startup).



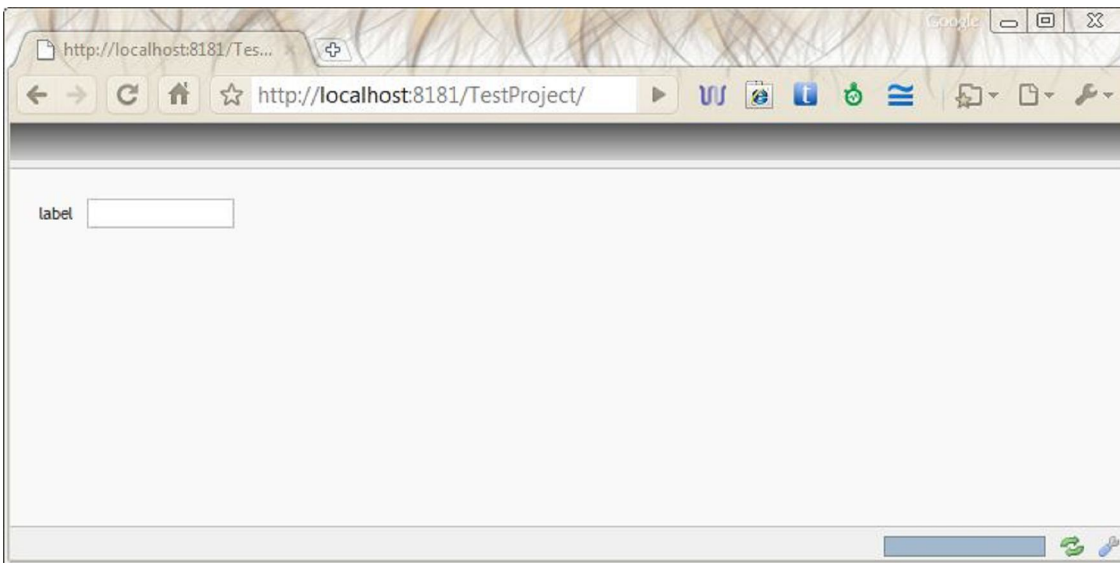
- Save the project
- Create the starter file
 - Under 'Tools' menu of JSP editor, select 'Create Applet HTML/JNLP'
 - Give the name **index** and make any choices as needed by the project. Any name is allowed, but **index** is usually the default for web servers.
 - Press 'Create Applet / JNLP' to create the file



- In Netbeans, clean and redeploy the application (it may also be required to restart the server)

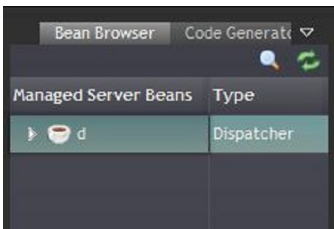


- Open the application in a browser

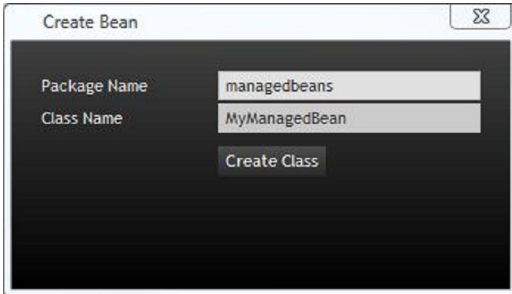


Creating a ManagedBean

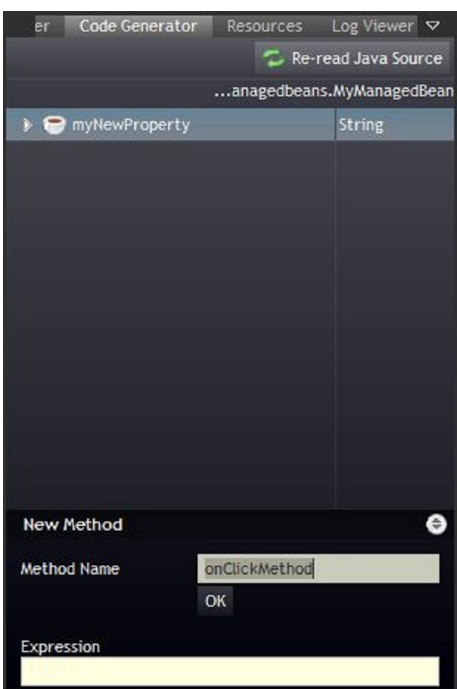
- Ensure the application is compiled and deployed via Netbeans.
- In CaptainCasa Editor open the **Bean Browser**. A ManagedBean called `d` of type **Dispatcher** should be visible. (This helper class allows beans to be added without having to modify `face-config.xml`, see the developer documentation).



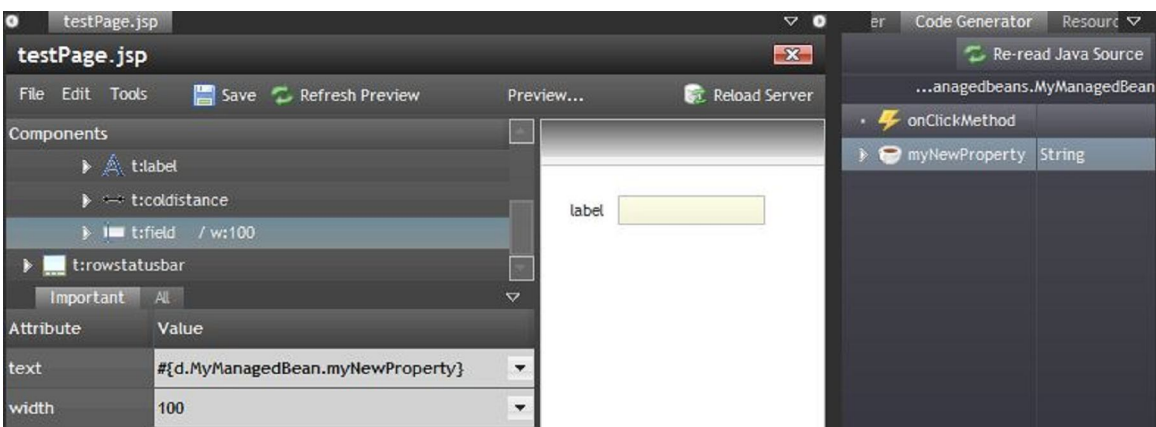
- Right-click on the `d` bean and select **Create Bean**
- Give the bean a name, and press **Create Class**.



- The **Code Generator** tool is opened and this can be used to create properties and methods.



- Properties and methods can be dragged and dropped onto component attributes within the editor.



- **Note:** Explicitly deploy the application in order to copy the class files to the webcontent directory for the CaptainCasa Editor.
- When viewing the ManagedBean in Netbeans, the imports have to be added (*Shift-Ctrl-O*)
- ManagedBeans can also be created directly in Netbeans directly

The screenshot shows an IDE interface with a project tree on the left and two code editors on the right. The project tree shows a 'TestProject' with subfolders for 'Web Pages', 'Source Packages', 'managedbeans', 'Libraries', and 'Configuration Files'. The 'managedbeans' folder contains 'Dispatcher.java', 'MyManagedBean.java', and 'NewJSFManagedBean.java'. The two code editors show the following code:

```

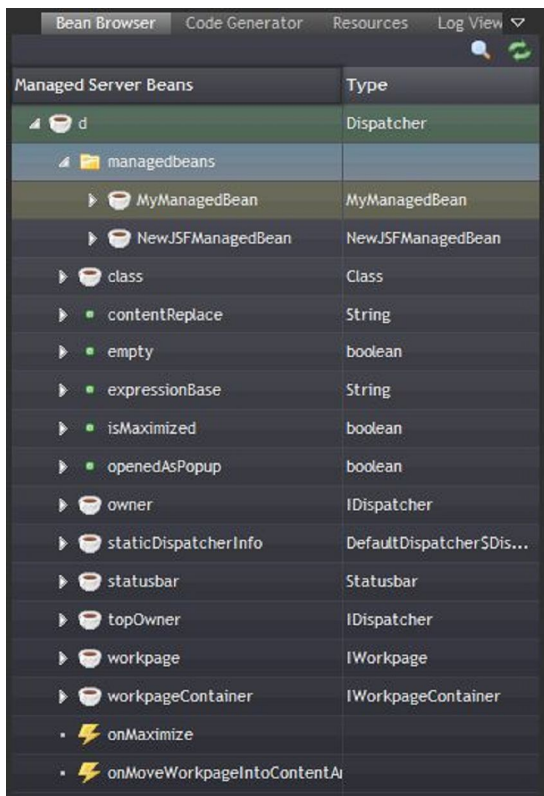
NewJSFManagedBean.java
1 package managedbeans;
2
3
4 import java.io.Serializable;
5 import javax.faces.bean.RequestScoped;
6
7 /**
8  *
9  * @author Gary Hodgson
10 */
11 @RequestScoped
12 public class NewJSFManagedBean implements Serializable {
13     //properties and methods here...
14 }
15

MyManagedBean.java
1 package managedbeans;
2
3 import java.awt.event.ActionEvent;
4 import java.io.Serializable;
5 import org.eclint.editor.annotations.CCGenClass;
6
7 @CCGenClass (expressionBase="#{d.MyManagedBean}")
8
9 public class MyManagedBean implements Serializable
10 {
11     public void onClickMethod(ActionEvent event) {}
12
13     protected String m_myNewProperty;
14     public String getMyNewProperty() { return m_myNewProperty; }
15     public void setMyNewProperty(String value) { m_myNewProperty = value; }
16
17 }
18
    
```

- Again, remember to deploy the application to refresh the webcontent classes folder

The screenshot shows the 'webcontent' folder expanded in the project tree. Inside 'webcontent', there are folders for 'META-INF', 'WEB-INF', and 'classes'. The 'classes' folder is further expanded to show a subfolder named 'managedbeans'. Inside 'managedbeans', three class files are listed: 'Dispatcher.class', 'MyManagedBean.class', and 'NewJSFManagedBean.class'. The 'classes' folder and its contents are highlighted in yellow.

- When created, press the refresh icon in the CaptainCasa Editor to show the bean.

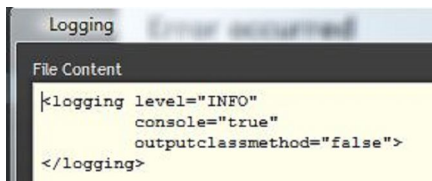


The screenshot shows the Bean Browser interface with a tree view of managed server beans. The tree is expanded to show the following structure:

Managed Server Beans	Type
d	Dispatcher
managedbeans	
MyManagedBean	MyManagedBean
NewJSFManagedBean	NewJSFManagedBean
class	Class
contentReplace	String
empty	boolean
expressionBase	String
isMaximized	boolean
openedAsPopup	boolean
owner	IDispatcher
staticDispatcherInfo	DefaultDispatcherSDis...
statusbar	Statusbar
topOwner	IDispatcher
workpage	IWorkpage
workpageContainer	IWorkpageContainer
onMaximize	
onMoveWorkpageIntoContentA	

Misc. Notes

- Setting the logging level to write to console is useful, but makes the deploy and runtime process slow down considerably.



```
Logging Error occurred
File Content
<logging level="INFO"
  console="true"
  outputclassmethod="false">
</logging>
```