

# Integrating CaptainCasa into “Maven-style Projects”

Document Status: in work, feed back highly appreciated

When learning how to work with CaptainCasa and when doing first steps with CaptainCasa we strongly recommend to use the default project structure that is created by the CaptainCasa toolkit!

Later on you might want to integrate CaptainCasa into project structures that are not defined by CaptainCasa but by yourself or by some tooling that you use, e.g. Maven. Then, it's the perfect time to read and follow this document.

---

## Default Project Structure

By default CaptainCasa projects are created within the toolset of CaptainCasa (Layout Editor). In this case all corresponding project files are created automatically. The project directory will look as follows:

```
<projectdir>
  /src
    ...the UI classes
  /webcontent
    /eclnt
    /eclntjsfserver
    /images
    /WEB-INF
      /classes
        ...compiled classes
      /lib
        ...all the JARS including JSF,
        ...CaptainCasa + dependent libraries
    /META-INF
```

The compilation of the IDE is automatically set up (if working with Eclipse), so that UI sources are copied into the WEB-INF/classes directory.

The “/webcontent” directory of the project is the one that reflects the web application, that is later on deployed (copied) into the runtime environment. In the webcontent directory there are all resources, that are related to the project (.jsp files, images, ...) - and there are the compiled classes.

### Updating a CaptainCasa project

When CaptainCasa provides an updated version, then updates need to be incorporated into your project. For this reason, there is a directory “<install>/resources/webappaddons”, that is part of the CaptainCasa installation. This directory holds everything, that CaptainCasa adds to a web application.

The updating of a project is done by copying the whole “webappaddons”-directory into the “webcontent” directory of the project. This is implicitly done within the toolkit of CaptainCasa, when a project is identified to be updated and when a button “Update Project” appears.

---

## “Other” Project Structures, e.g. Maven

When NOT creating a project by using CaptainCasa tools, but by using other tools like Maven then the project structure might look different. E.g. the typical maven project

structure looks like:

```
<project>
  /src
    /main
      /java          ...the sources
      /resources     ...the libraries
      /webapp        ...the web application sources
      ...
    /test
    ...
  /target
    /classes        ...the compile classes
    /site           ...the application
```

## Populating Project with CaptainCasa

The CaptainCasa addons for a web application are available in the directory “<install>/resources/webappaddons”. The directory looks like:

```
/webappaddons
  /eclnt           ...contains all client/applet/webstart resources
  /eclntjsfserver ...default .jsp pages, default images
  /META-INF       ..."no content"
  /WEB-INF
    /lib
      eclntjsfserver.jar ...CaptainCasa library
      jsf-api.jar        ...JSF
      jsf-impl.jar       ...JSF
      and.other.jar      ...JARS required by JSF and CaptinCasa
```

Taking over CaptainCasa into the project of course needs to be synchronized with your normal Maven dependencies (e.g. you may want to load jsf-related jars by dependency definition) - so what we explain here is the “basic process” only:

- Copy /eclnt to /src/main/webapp/eclnt
- Copy /eclntjsfserver to /src/main/webapp/eclntjsfserver
- Copy /WEB-INF/lib/\*.jar to /src/main/resources

You may create an ANT script in order to automate this procedure, so that after each CaptainCasa update you refresh you project accordingly.

## Creating CaptainCasa-Project in CaptainCasa Toolkit

The next step is to create the CaptainCasa project in the CaptainCasa toolkit. Normally the creation of a project is done through a popup dialog - as result both the project (in its default project structure) and a CaptainCasa-project-file is created.

Now, because we do not want CaptainCasa to create “its” default directories, we create the project by just creating the CaptainCasa-project-file.

CaptainCasa project files are kept in the following directory of the standard installation:

```
<install>/
  /tools
    /embeddedservers
      /webapps
        /editor
          /config
            /projects
              *.xml          ...project files
```

In the project file there are several parameters to be defined. There is an explanation of these parameters in the Developers' Guide already - and there is a short documentation within the template file that is provided within the projects-direcory.

In the concrete example we would set up a file <projectName>.xml with the following content:

```
<project
  projectdirectory    = "<projectDir>"
  webcontentdirectory = "<projectDir>/src/main/webapp"
  webcontentdeployfromdirectory = "<projectDir>/target/site"
  javasourcedirectory = "<projectDir>/src/main/java"
  javaclassdirectory = "<projectDir>/target/classes"
  uitestcasedirectory = "<projectDir>/test/main/ccuitest"

  webcontentdeploydirectory = "<install>/server/tomcat/webapps/<projectName>"
  webcontextroot            = "<projectName>"
  webhostport               = "localhost:50000"

  copywebapp    = "true"
  reloadwebapp = "true"
>
</project>
```

(Replace all <...>-Parameter with the concrete definitions of your project.)

As result, the .jsp files and the .java files that are generated within the CaptainCasa toolkit will be written to the correct locations of your project. And the deployment of the CaptainCasa application will be done from the right directory.

After having created the project file, the project will be automatically visible within the CaptainCasa toolkit, the next time it is refreshed or started.

### Some Comments...

Of course this is not yet “optimal” - you need to do some manual work for making a project visible to the CaptainCasa toolkit. And the populating your project with CaptainCasa is a copying of files - and not done by Maven dependency resolution.

But: the purpose of the document is to describe, how to do it today - and not to talk about what the way will be in some future time.