

Installation & Configuration

Release 3.0



Table of Content

Installation Information - WINDOWS.....	3
Installation.....	3
Installed Directories and Files.....	3
“documentation”.....	3
“resources”.....	3
“server”.....	4
“tools”.....	4
Running the Server.....	4
Start Tomcat.....	4
Open Demo Pages.....	4
Open Tools.....	5
Browser Setup.....	5
Upgrading an existing Installation.....	5
Using the Project Overview in order to upgrade your Projects.....	5
Deploying into an existing Tomcat Environment.....	7
Versions expected.....	7
Unzipping.....	7
Demos and Layout Editor Installation.....	7
Start the Applications.....	7
Better Way of Starting the Editor.....	8
Upgrading an existing Installation.....	8
Configuration of Tomcat for Development Usage.....	9
Special Rules for using Tomcat 5.5.....	10
Setup of an Environment for Development.....	12
Servlet Container for running Tools.....	12
Development Project Setup.....	12
Storing the JSP Files.....	13
Storing Java Files.....	13
Defining what happens at Reload Point of Time.....	13
Set up your Scenario, Summary.....	13
Eclipse Plugin.....	15
Installation.....	15
Eclipse View Configuration.....	15
Configuration & Installation Issues.....	16
Logging.....	16
Server Side Logging.....	16
Client Side Logging.....	16
Libraries coming with CaptainCasa Enterprise Client.....	16
JSF Libraries.....	16
CaptainCasa Library.....	17
Crimson SAX Parser.....	17
Other JARs...?.....	17
Disable Usage of Cookies for Session Tracking.....	17

Installation Information - WINDOWS

The Windows installation is the “all inclusive package”. It comes with an installation wizard unzipping quite a lot of information, and it comes with all components required to run CaptainCasa Enterprise client: a JRE and a Tomcat server. And it creates shortcuts that are added to the Start-Menu.

If you have a certain amount of experience dealing with Java and Tomcat and if you do not want to install the “all inclusive package” or if you are not working with Windows... - then have a look into the chapter “Deploying CaptainCasa Enterprise Client”.

Installation

Execute the following steps:

- Start “setup_V_S_YYYYMMDD.exe”
 - V = version
 - S = subversion
 - YYYYMMDD = Year/Month/Day of build
- Select a directory of your choice
- Follow the installation instructions.

The installation does a plain extracting and copying of files into the specified directory. Registry records are only written for installation and starting purposes (e.g. Start Menu items). There is no critical information written into the registry which has to do with the design time and runtime environment of CaptainCasa Enterprise Client.

Installed Directories and Files

Have a look onto the installed directory:

```
<installdirectory>
  /documentation
  /resources
    /eclipseplugin
    /webappaddons
    ...
  /server
    /jre
    /tomcat
  /tools
  A_startserver.bat
  B_startclient_demos.bat
  B_startclient_tools.bat
  ...
  ...
```

“documentation”

This directory includes both the manuals and the JavaDoc API documentation. The most important manual is the “Developers' Guide”.

“resources”

This directory includes the sub-directory “webappaddons”. In this subdirectory all files that need to be added to a web application in order to enable CaptainCasa Enterprise Client are contained.

In the directory “eclipseplugin” you find the zip-file which needs to be extracted into your Eclipse development environment. The plug-in simplifies the synchronization of content between the CaptainCasa Enterprise Client toolset and Eclipse.

“server”

The “server” directory contains a full runnable server for CaptainCasa Enterprise Client. The directory includes a Java Runtime Environment (JRE) and a Tomcat server. As part of the Tomcat server (directory tomcat/webapps) there is one deployed web application: “demos”.

Of course the server part of CaptainCasa Enterprise Client runs on any servlet engine and within other application server than Tomcat. The installation includes Tomcat in order to simplify the running of the first examples and projects.

“tools”

This directory holds the toolset that comes with CaptainCasa Enterprise Client. The toolset includes a WYSIWYG Layout Editor for creating the user interfaces of your applications. The toolset itself is a web application that runs in so called “embedded usage mode”: i.e. the user interface client - which is the normal CaptainCasa Enterprise Client - is directly integrated with an embedded Tomcat installation.

Running the Server

Start Tomcat

Start the demo server by starting the file “A_startserver.bat” or by calling “Start > CaptainCasa Enterprise Client > A Start Server” from the Windows start menu.

The reaction is a command window (“DOS box”) that opens up.

In case the command window only shows up for a short time and then immediately closes the most probable reason is a conflict of port numbers:

- The server is using a default tomcat installation, occupying port 50000 (and some other ports 50001, 50002, 50003 for administration). One of these ports may already used e.g. by a second Tomcat running inside your environment.
- Stop the application occupying the port and then restart the demo server.
- Or edit “demoserver/tomcat/conf/server.xml” and replace the ports according to your environment. - In case of overriding the ports: pay attention when applying future updates of CaptainCasa Enterprise Client! The server.xml file will be overwritten.

Open Demo Pages

The demo pages are rich client applications, of course. They can be started in multiple ways.

- The easiest way is to start them directly as Java application. For this reason have a look into the Windows start menu - you find two items there: one starting the client with some demos, and one starting the client with the Enterprise Client toolset:
 - Start => Programs => CaptainCasa Enterprise Client => B Start Client - Demo Workplace
- Or: start the browser and open one of the following links:

- <http://localhost:50000/demos>

You will see corresponding pages guiding you to the next steps. If problems occur, especially in the area of opening the pages in a browser environment: please have a look into the next section of the documentation.

Please note: for development usage of the tools we recommend starting them through the Windows-Start-Menu.

Open Tools

The CaptainCasa toolset is a rich client application as well. The application part is running in an embedded Tomcat that is directly started as part of the tool's client. Starting the toolset is done through the Windows Start Menu:

- Start => Programs => CaptainCasa Enterprise Client => C Start Development Tools

There is a second start program for starting the tools in a “lighter style”, as well - with no functional difference to the normal start program.

Browser Setup

In order to run Enterprise Client either as applet or by using web start within a browser environment you need to make sure that a Java runtime environment is installed on client side. The Enterprise Client uses a Java runtime engine of version 1.6.

The easiest way to check the installation of the Java runtime is to open up the page <http://www.java.com/download/> inside the browser. This page allows to check the current Java configuration within your browser environment and automatically initiates the loading of updates if required.

Upgrading an existing Installation

A new version of CaptainCasa Enterprise Client needs to be installed in the following way:

- Close the Tomcat of your existing installation - otherwise certain files will be blocked.
- Run the installer - just as with the first installation. The installer will propose the directory that you already used for installation. Do not override this directory, but just install the new version on top of the existing one.

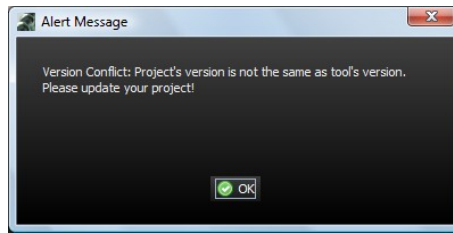
Last, a very important steps: because CaptainCasa Enterprise Client is part of a web application, you need to upgrade all your web application projects accordingly. For each project there is one web application. You now need to copy the content of <install-directory>/resources/webappaddons into each <project>/webcontent directory. As consequence the existing content gets overridden - but only these parts which are the CaptainCasa parts. Your application will not be touched.

This copying can be done in multiple ways:

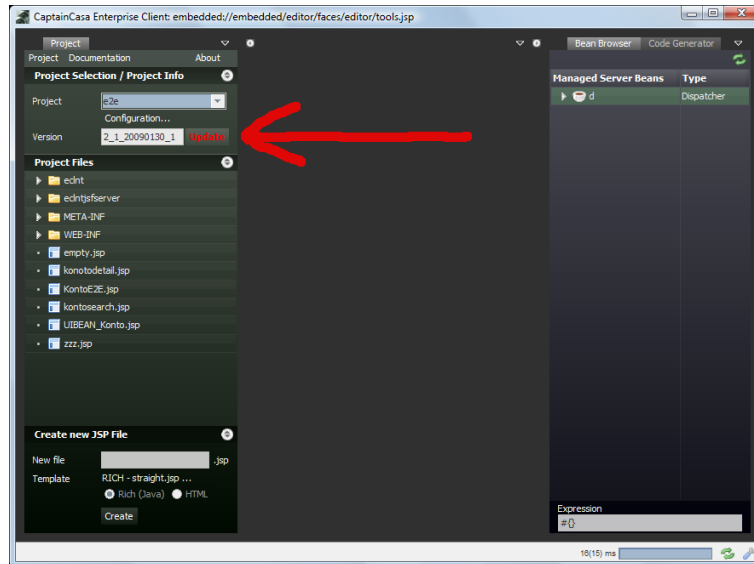
- Manually (well, ...)
- By Shell script or by ANT task (better, ...)
- Interactively inside the tool set of CaptainCasa Enterprise Client (most simple, ...). Have a look into the following text for more information.

Using the Project Overview in order to upgrade your Projects

After having upgraded your installation, restarting Tomcat and re-invoking the project overview you will be notified if your projects do not match the current version:



In addition you will see a button “Update” showing up inside the project overview:



After pressing the button the update of the application will be done by the toolset, i.e. the copying of files from “webappaddons” will be done.

Please check the statusbar message after pressing the button.

Deploying into an existing Tomcat Environment

Versions expected

We recommend the following versions:

- JRE or JDK 1.6
- Tomcat 6

The minimum requirements are:

- JRE / JDK \geq 1.5
- Tomcat 5 (servlet 2.3)

Make sure your tomcat is up and running prior to deploying CaptainCasa Enterprise Client.

Unzipping

Use the “.zip” Installation file that is available with each CaptainCasa Enterprise Client delivery. Unzip the installation file into a directory of your choice.

The directory will look like:

```
<installdirectory>
  /documentation
  /resources
    /webappaddons
    /webapplications
```

The **webappaddons** contains all the basic files that form a CaptainCasa Enterprise Client web application. Tools (e.g. for creating projects) will copy the webappaddons content into the content directory of the web application. When upgrading a web application (e.g. due to an update by CaptainCasa) you need to copy the whole content of /webappaddons into your web application.

The **webapplications** directory contains two .war files:

- editor.war - This is the toolset.
- demos.war - This is the demo workplace.

Demos and Layout Editor Installation

Deploy both web applications by simply copying the corresponding .war file into the /webapps/ directory of your Tomcat installation. Tomcat will automatically extract the .war files and will start the corresponding web applications.

Start the Applications

For starting the demo workplace, open a browser and start the URL:

- <http://localhost:<port>/demos>

For starting the toolset, open a browser and start the URL:

- <http://localhost:<port>/editor>

Better Way of Starting the Editor...

Starting the editor through the browser is possible but has one disadvantage: within the preview area of the editor you are bound to one http session, because all http request coming from the client are automatically assigned to the corresponding http session. This means: that when refreshing the preview area, then the content you see still goes back to the same server side http session, there is no automated creation of a new http session.

For this reason we recommend to start the editor outside the browser environment, i.e. via using an explicit Java application that is started by command line. The script file in Windows looks as follows:

```
set TCP=XXX\resources\webappaddons\ecInt\lib\ecInt.jar
set TCP=%TCP%;XXX\resources\webappaddons\ecInt\lib\swt.jar
set TCP=%TCP%;XXX\resources\webappaddons\ecInt\lib\jasperreports-
3.0.0.jar
set TCP=%TCP%;XXX\resources\webappaddons\ecInt\lib\commons-beanutils-
1.7.jar
set TCP=%TCP%;XXX\resources\webappaddons\ecInt\lib\commons-collections-
2.1.jar
set TCP=%TCP%;XXX\resources\webappaddons\ecInt\lib\commons-digester-
1.7.jar
set TCP=%TCP%;XXX\resources\webappaddons\ecInt\lib\commons-javaflow-
20060411.jar
set TCP=%TCP%;XXX\resources\webappaddons\ecInt\lib\commons-logging-
1.0.2.jar
set TCP=%TCP%;XXX\resources\webappaddons\ecInt\lib\commons-logging-api-
1.0.2.jar

start javaw.exe -cp %TCP% org.ecInt.client.page.PageBrowser
http://localhost:50000 /editor/faces/editor/tools.jsp headerline=false
```

Adapt the “XXX” with your installation directories.

You see: all libraries that are normally loaded into the applet or web start client are now referenced in the classpath. The Java program “PageBrowser” is started with two parameter as minimum: the server to access, and the page to access. Pay attentions: there is a blank space between “<http://localhost:50000>” and “/editor/faces/etc. ...” in the batch file above!

Upgrading an existing Installation

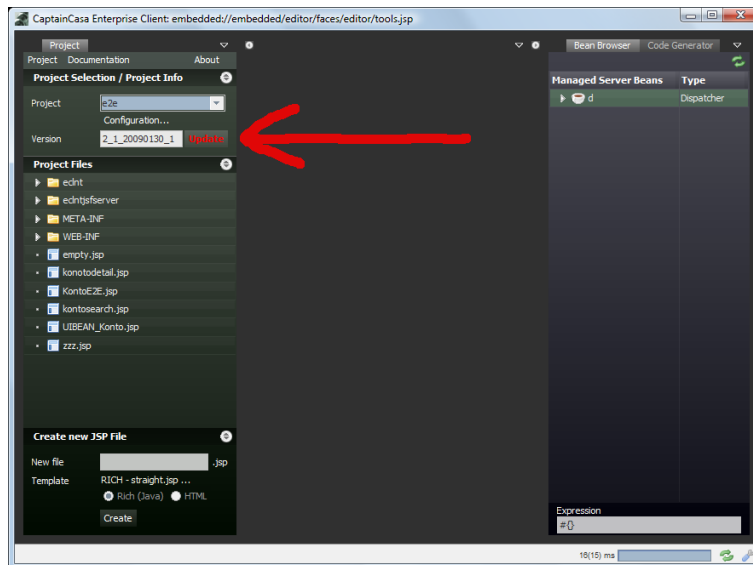
When already having CaptainCasa installed within your Tomcat environment then do the following steps in order to upgrade you installation:

- Refresh the editor and the demos web application.
- The best way to refresh is to...
 - ...remove existing demos.war and editor.war from the tomcat/webapps directory...
 - ...and extract the new demos.war and editor.war on top of the existing one.
 - Please do not remove the existing editor/ directory - it contains project definitions (in config/projects) that are important to survive the upgrade procedure!

Now you need to upgrade the “webcontent” directory of your projects: copy the content of “resources/webappaddons” into the “webcontent” directory. You need to do this for each individual project.

Copying either can be done manually or via script (shell script, ANT script) - or via using the CaptainCasa Enterprise Client tools. The project overview will show a corresponding

button if the project's version is not in synch with the tool environment's version:



Configuration of Tomcat for Development Usage

When using the CaptainCasa Tool Environment (editor.war) then there is a certain function “Reload Server” which can be called from the tool “Layout Editor”. This function on the one hand deploys the application project into the Tomcat runtime and on the other hand tells Tomcat to restart the corresponding web application.

The restart is done by sending a URL to Tomcat's manager application. - And that's where you need to pay attention to. Tomcat's manager application (webapps/manager) by default requires a user and password to log on. This user/password is NOT transferred by the CaptainCasa Enterprise Client tools.

Consequence: you need to switch off the manager's security setting in the /tomcat/webapps/manager/WEB-INF/web.xml file:

```
...
...
<!-- Define a Security Constraint on this Application -->
<!--
<security-constraint>
  <web-resource-collection>
    <web-resource-name>HTMLManger and Manager command</web-resource-
name>
    <url-pattern>/jmxproxy/*</url-pattern>
    <url-pattern>/html/*</url-pattern>
    <url-pattern>/list</url-pattern>
    <url-pattern>/sessions</url-pattern>
    <url-pattern>/start</url-pattern>
    <url-pattern>/stop</url-pattern>
    <url-pattern>/install</url-pattern>
    <url-pattern>/remove</url-pattern>
    <url-pattern>/deploy</url-pattern>
    <url-pattern>/undeploy</url-pattern>
    <url-pattern>/reload</url-pattern>
    <url-pattern>/save</url-pattern>
    <url-pattern>/serverinfo</url-pattern>
    <url-pattern>/status/*</url-pattern>
    <url-pattern>/roles</url-pattern>
    <url-pattern>/resources</url-pattern>
  </web-resource-collection>
```

```

    <auth-constraint>
      <role-name>manager</role-name>
    </auth-constraint>
  </security-constraint>
-->

<!-- Define the Login Configuration for this Application -->
<!--
<login-config>
  <auth-method>BASIC</auth-method>
  <realm-name>Tomcat Manager Application</realm-name>
</login-config>
-->

<!-- Security roles referenced by this web application -->
<!--
<security-role>
  <description>
    The role that is required to log in to the Manager Application
  </description>
  <role-name>manager</role-name>
</security-role>
-->

<error-page>
  <error-code>401</error-code>
  <location>/401.jsp</location>
</error-page>
</web-app>

```

All web.xml aspects having to do with security are commented out.

Of course: this is only a development configuration! Make sure you do not pass the configuration into your operational systems by accident!

Special Rules for using Tomcat 5.5

Soon after publishing Release 3.0 there was some feedback from the community...: Tomcat 5.5 is still in use at some places, sometimes embedded in other server (e.g. JBoss 4.0). Tomcat 5.5 does not support Java Server Pages 2.1, which is required for running JSF 1.2.

What we did as result: we built some compatibility layer into our CaptainCasa server runtime and now can easily switch between running in JSF 1.2 environment and JSF 1.1 environment.

By default “JSF 1.2” is used from CaptainCasa Release 3.0 on. In case you want to still use Release 3.0 with “JSF 1.1” then you need to do the following:

Within the installation there is the directory “/resources/jsf11usage”. In the directory you find the following files:

```

e1-api.jar           <== JSP expression library
jsf-api.jar          <== 1.1 version of JSF
jsf-impl.jar         <== 1.1 reference implementation
eclntjsfserver.jar  <== CaptainCasa server library

```

The eclntjsfserver.jar library (i.e. the one that contains the CaptainCasa server runtime) is exactly the same library that is used when using JSF1.2 - the only difference is in the file “faces-config.xml” that comes with the library within the META-INF directory of the .jar file.

If you want to run your application under JSF 1.1 you need to...

- ...always copy jsf-api.jar, jsf-impl.jar and eclntjsfserver.jar into the webcontent/WEB-

INF/lib directory of your project

- The file el-api.jar is only required if not part of your server runtime. E.g. when using Tomcat 5.5 you need to copy this file as well. It is not needed from functional perspective but is required for proper class loading.

PLEASE NOTE:

- **Every time you update your project from a CaptainCasa update you will need to execute the copying of the JSF 1.1 related libraries! Please set up e.g. an ANT script in order to consistently upgrade your project.**
- You only need to update the libraries within your application project - you do not need to update CaptainCasa projects (e.g. the editor or the demos project).

Setup of an Environment for Development

CaptainCasa (at least the Windows version) comes with a pre-packaged Tomcat and Java environment that allows to jump-start with developing applications. This chapter contains information about how to set up individual scenarios - that e.g. are in synch with a development environment that you may already have in place.

Servlet Container for running Tools

The CaptainCasa toolset is a rich client web application on its own. CaptainCasa tools are a CaptainCasa application themselves: the user interface is running in the rich client, the tool processing is running on server side, within a web application.

As a result you first need to think about: which is the servlet container to run CaptainCasa tools? Of course this container (typically a Tomcat installation) should be one to run independent from the processing of you application: when you need to restart your application, then this should not have the consequence that the tools are restarted as well.

There are two ways to do so:

- Web application isolation: the tools (“editor.war”) run in the same servlet container than your application (“xyz.war”). In case of changing and e.g. re-deploying your application you do not restart the whole servlet container, but you only restart the application (“xyz.war”).
- Servlet container isolation: the tools (“editor.war”) run in an own Tomcat instance - and the application runs in an own Tomcat instance as well. I.e. you for example access the tools via “localhost:50000” and your application via “localhost:8080”. In this case you can do anything on application side and never will see any re-starting of tools.

In the default environment that comes with the Windows installation of CaptainCasa “Web application isolation” is used: the Tomcat that comes with the installation is used both for running the “editor.war” and for deploying application projects. When reloading an application inside the Layout Editor then, after copying some files, the web application of your project is restarted using Tomcat’s manager.

Development Project Setup

After having decided how to set up you servlet container environment you need to define how to set up your development project.

- You somewhere have your project files (.java sources, .jsp sources), typically in a project that is managed within your development environment (e.g. Eclipse). That's the place where you compile your .java sources etc.
- You have defined points of time when you implicitly or explicitly deploy your project into a servlet container (e.g. Tomcat) in order to see and test your application. Maybe you use some “tricky” ways to only deploy parts of your application and keep the core part of the application running - that's completely up to you.

The project definition of the CaptainCasa toolset environment allows to adapt to you individual setup. Hava a look onto the following example file:

```
<project
webcontentdirectory="C:\bmu_jtc\eclipse\workspace\demo_pim\webcontent"
  webcontentdeploydirectory="C:/bmu_jtc/EnterpriseClient/server/to
mcat/webapps/ccpim"
  webcontextroot="ccpim"
```

```
        javasourcedirectory="C:\bmu_jtc\eclipse\workspace\demo_pim/webco
ntent/../../src/"
        copywebapp="true"
        reloadwebapp="true"
        webhostport="localhost:50000">
</project>
```

Storing the JSP Files

The most important thing the Layout Editor does is to store JSP files. Because the update of JSP files should be immediately reflected in the WYSIWYG preview section of the editor, the files are written to two locations:

- (1) the project location itself (parameter “webcontentdirectory”)
- (2) the deployment location of the project (parameter “webcontentdeploydirectory”)

Some details on (2): in order to “force” the servlet container to really use the latest JSP file, the tool environment writes two files into the webcontent deploy directory: the JSP file itself - and a JSP file with a temporary name (“*_ZZZZZ_<timestamp>.jsp”). The preview section of the Layout Editor will show the temporary file when previewing a page.

Storing Java Files

When using the Code Generator and the Resource Editor then corresponding files are written into the directory that is defined via parameter “javasourcedirectory”.

Defining what happens at Reload Point of Time

Within the Layout Editor there is one important button: the “Reload” button. The intention behind is: you maintain your pages (and see results via preview) and you maintain your application’s Java sources. While previewing pages is a thing you permanently do, there are fewer points of times you want to re-deploy your application. When pressing the “Reload” button the re-deployment will be triggered.

There are two parameters to control the reload behaviour:

- “copywebapp”: when defining “true” then the whole content of the directory “webcontentdirectory” will be copied into the directory “webcontentdeploydirectory”. This means that a file-re-deployment will be executed.
- “reloadwebapp”: when defining “true” then the Tomcat that runs the application project (which may be different to the one that runs the tool environment!) will receive a signal to restart the application’s web application. For this purpose the “manager” web application of the Tomcat will receive a URL that is sent from the CaptainCasa tool environment, which requests to update the application’s web application. - Please remember: all security settings of the “manager” web application need to be switched off to enable this scenario!

Set up your Scenario, Summary

When having become familiar with the default CaptainCasa toolset environment you now can start to embed CaptainCasa into your development environment setup. The two major aspects are:

- Make sure that the tools’ web application (“editor.war”) runs independently from your application’s web application.
- Make sure that you can smartly re-deploy your application - and use the reload

features offered by the project definition to support your scenario.

Eclipse Plugin

CaptainCasa Enterprise Client comes with an Eclipse plugin that automatically synchronizes Eclipse project's with the file system when files are change by CaptainCasa tools.

The Eclipse plugin is contained in the directory “resources/eclipseplugin” of your installation.

Installation

Unzip the file “org.eclnt.eclipseplugin.zip” into your Eclipse installation. The zip-file internally starts with the plugin-directory, so you can directly unzip into your Eclipse installation folder.

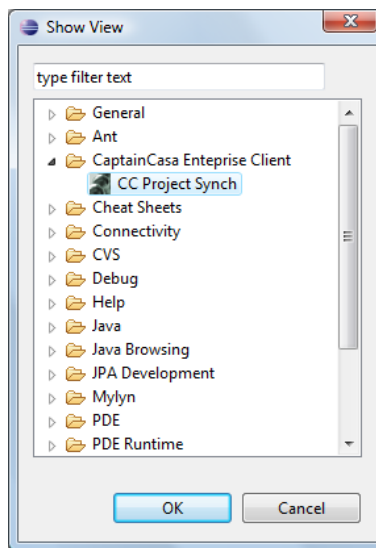
After installation the following file should be part of your Eclipse directory:

```
<Eclipse directory>
  /features
  /plugin
    org.eclnt.eclipseplugin_1.0.0.jar
  ...
```

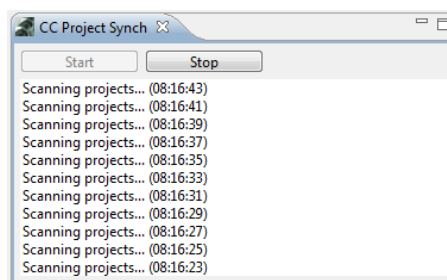
Restart your Eclipse.

Eclipse View Configuration

After having started Eclipse open the view configuration: Window > Show View > Other...:



Open the view “CC Project Synch”:



After pressing the Start button the view will scan for project updates every 2 seconds.

Configuration & Installation Issues

Logging

Server Side Logging

The server stores log messages within the web application's temporary directory by default. In case of using Tomcat this is the directory "tomcat/work/Catalina/localhost/<webappname>".

By defining a log level you can decide upon the granularity of the logged information. The log level is defined in the file <webapp>/ecIntjfsrver/config/logging.xml". If the file does not exist yet, then take a copy from the template file within the same directory.

Within the file you can set the log level:

```
<!--  
level: SEVERE, WARNING, INFO, FINE, ALL  
console: false, true  
directory: directory for logging  
-->  
<logging level="INFO">  
</logging>
```

There is a parameter "directory" by which you can pass an explicit directory to write the log file. Pay attention: in an heterogeneous environment (e.g. different operating systems) the directory needs to be available on all server nodes running the application.

There is an additional parameter "console" that is useful for development purposes. Setting "console" to "true" will in addition to the file based logging start a console logging, so that all relevant messages directly appear in the console.

Client Side Logging

The client writes all log information to the console. When using applet or webstart for starting the client start the console from your browser in order to view log information.

When starting the client from command line then call "java.exe" instead of "javaw.exe" - the command box will not disappear right after starting.

You can define the log level by setting a start parameter of the client - have a look into the corresponding appendix of the Developers' Guide. When setting the log level to "FINE" or "ALL" then you can see the server's XML responses within the log - which is sometimes very useful for debugging purposes.

Libraries coming with CaptainCasa Enterprise Client

When creating a web application using CaptainCasa Enterprise Client then some libraries are copied into the WEB-INF/lib folder of the web application.

JSF Libraries

The default delivery comes with the Sun JSF reference implementation 1.1. The corresponding libraries are:

- jsf-api.jar
- jsf-impl.jar

...and reference libraries:

- commons-logging-*.jar
- commons-collections-*.jar
- commons-digester.jar
- commons-beanutils.jar
- jstl.jar

CaptainCasa Library

All CaptainCasa parts that are related with the server side processing are packaged within the following library:

- eclntjfsserver.jar

Crimson SAX Parser

Yes, a SAX parser is also part of the Java runtime environment, but: this default parser showed one ugly bug - with the result of not properly parsed XML documents. So we use crimson.jar by default. The parser can be configured using configuration file `<webapp>/eclntjfsserver/config/system.xml` - so you can use an own parser if required.

- crimson.jar

Other JARs...?

Please note: there are also a couple of jar files located below `<webapp>/eclnt/lib`. But: these have nothing to do with the server side processing - they are part of the client that is loaded into the frontend. Example: when using applet technology then these libraries are loaded as applet libraries.

Not all of these libraries are always loaded - some of them are optional. When creating an applet/web start page using the CaptainCasa tool environment you can explicitly select which ones are required for the client processing.

Disable Usage of Cookies for Session Tracking

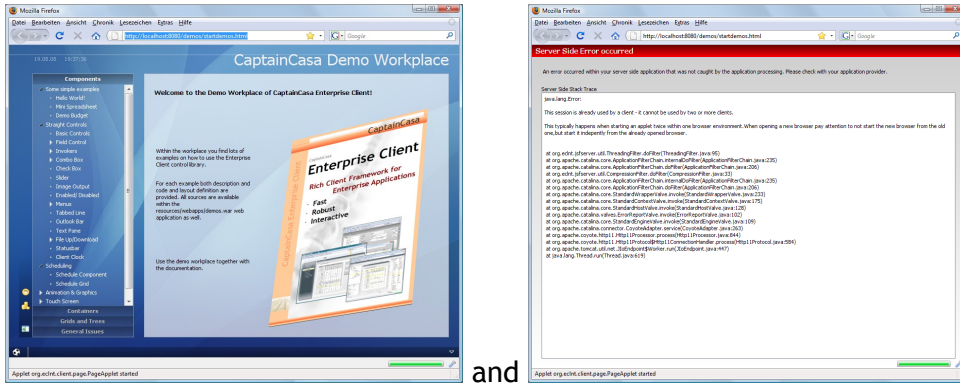
By default session tracking is done using Cookies - this is not CaptainCasa Enterprise Client's default but this is the default of servlet containers. But: there is a better technique for session tracking available: URL rewriting. CaptainCasa Enterprise Client supports URL rewriting. With URL rewriting a server side session id is passed within the URL of every request that is sent from the client to the server. There is no cookie data, that is appended the http request.

It is possible to switch off the usage of cookies by placing the following file into the META-INF directory of your web application:

```
File Name: META-INF/context.xml
File Content:
<?xml version='1.0' encoding='UTF-8'?>
<Context cookies='false'>
</Context>
```

Why and when should you do this? - Have a look onto the following scenario: a user starts the an application using applet in the browser, then opens a new browser and opens the

same application once more. With using cookies the result will be:



The second screen shows an error message telling that this instance of the application is connected to a session that is already in use by another instance.

Background: both browsers use the same session cookie, that's how browser work. On server side there is a mechanism finding out that two client instances are working with the same session - a corresponding error message is output.

When disabling the usage of cookies by adding the context.xml file, then each instance on client side is reflected by a session instance on server side. There is no situation in which a session conflict might occur.

In general we recommend to disable the usage of cookies because sometimes users want to open two or more application instances.