## CaptainCasa Enterprise Client

# In a Nutshell



## CaptainCasa Enteprise Client - In a Nutshell

CaptainCasa Enterprise Client is a rich client solution that meets the requirements of typical business applications:

- Many dialogs
- Complex, server side business logic
- · Heavily used by employees to constantly support them doing their job
- Long term life cycle developed, sold and used for many years

CaptainCasa Enterprise is based on Java standards (JavaFX, Java Server Faces in the backend) following a server-centric frontend architecture.

Components	Grid with Details	Grid with Details						
Containers								
Grids & Trees	Demo Lay	out XML Server side Java	Code					
4 Grids and Trees								
<ul> <li>Simple Grid</li> </ul>	Select one of the	following items by double-c	lick.			Coloritoria item from the estide list		
<ul> <li>Grid of Bean Objects</li> </ul>	Article Id	Article Text	Unit of Measu	Stock Area	Supplier	either by double clicking or by		
<ul> <li>Grid with Details</li> </ul>	32481	Text for article 32481	pc	Front	28587	navigating with the keyboard and		
<ul> <li>Header and Footer</li> </ul>	25427	Text for article 25427	ka	Back	39425	pressing Enter. Result: the details of		
<ul> <li>Hide empty rows</li> </ul>	88676	Text for article 88676	nc	Front	15641	the item will be shown below the grid.		
<ul> <li>Scroll Animation</li> </ul>	53346	Text for article 53346	ka	Back	31869	In the implementation of the example		
<ul> <li>Row Height Definitions</li> </ul>	197362	Text for article 197362	nc	Front	56036	you see:		
Grid Llander Cell	51063	Text for article 51063	ka	Back	142428			
Double line grid	26826	Text for article 26826	pc	Front	76562	(.) By overriding the onRowExecute()		
Complex Columns	183363	Text for article 183363	ka	Back	85379	coding which is executed when the		
Grid Row Selector	53809	Text for article 53809	pc	Front	166752	user double-clicks a row. (There is a		
Horizontal Scrolling	116318	Text for article 116318	ka	Back	141082	corresponding onRowSelect() method		
<ul> <li>Column Sizing by Content</li> </ul>	4			)	) • •	which is executed when the user		
⊿ Tree						double clicks the grid, as well.)		
<ul> <li>Simple Tree</li> </ul>	Id	53809	Location		Quantity	(.) Inside the grid column fields, and		
<ul> <li>Tree with checkbox</li> </ul>	Description	Text for article 53809	Loc-143		78	inside the detail fields the business		
<ul> <li>Tree with smart text</li> </ul>	Unit of Measure	nc	Loc-345		886	object (Article, Stock) coming from		
<ul> <li>Explicit Status Image</li> </ul>	offic of Micusure	(PC	Loc-96		853	The logic layer is simply referenced.		
<ul> <li>Select vs. Toggle</li> </ul>	Supplier	166/52	Loc-173		573	within the grid row objects.		
<ul> <li>Row Positioning</li> <li>Grid Dense Manual</li> </ul>	Stock Area	Front	Loc-194		787			
Grid Popup Menu			Loc-376		388	(.) The detail area is only rendered		
Selection Issues			Loc-356		740	when a selected article is available. This is defines within the layout		
Sorting			Loc-138		336	definition of the corresponding row		
Drag & Drop			Loc-425	Loc-425		holding the detail area.		
<ul> <li>Load on Demand (1)</li> </ul>	¥.		Loc-239		626			
General			Loc-378		340 👻			

CaptainCasa Demo Workplace

## Frontend Technology

Today's hypes in the area of user interface development are dominated by HTML5 based frameworks. While these frameworks in general significantly simplify the development of web scenarios, they still fail to be an adequate technology solution meeting the requirements of business applications:

- Browser incompatibilities and performance continue to be a constant problem.
- Development efficiency in the web is still way behind expectations.



• The long term stability of frameworks still IS a problem. There's a constant coming and going of frameworks and hypes.

All this means risk and cost for your long term oriented application development. And all this means that many applications are still stuck with UI technology coming from the "late 90s" (C++, VB, Java Swing, Delphi) that more and more fail to meet the usability requirements of modern frontends.

JavaFX - the Java standard in the are of UI processing - is an up to date UI technology that meets these requirements and that provides the long term stability expected by you and your customers.

The client side of CaptainCasa Enterprise Client is based on JavaFX as consequence.

## **Frontend Architecture**

The core architectural question when defining a frontend architecture is, if you should follow a "client-centric" or a "server-centric" approach.



#### **Client-centric**

In the client centric architecture your development is split between a frontend part (developed by frontend developers) and a backend part (developed by backend developers). A couple of APIs are defined by the backend side that are accessed from the UI processing of the frontend.

While being an adequate architecture for smaller projects, the client centric programming comes with significant disadvantages for bigger applications:

- The efficiency of development lacks because of having a split between frontend and backend development.
- The number of APIs is growing and growing. The

client UI logic tends to require very specific, fine granular APIs in order to provide a responsive user interface.

• Many APIs automatically mean potencial security and potential roundtrip problems.

- There is the "real" logic on server side and some "pre"-logic on client side, which always have to be in sync.
- The size of the client is growing with a growing size of the application.

Many application developments start with the client centric approach because it's the easiest to start with - and because of not having thought about the frontend architecture too much. The result are client programs that were called "fat clients" in the 90s.

#### Server-centric

The server centric approach requires some more framework background.



Here a generic client serves as some kind of "form processor": it receives some form definition from the server side form processing and outputs the form onto the user's screen. The user processes the screen (e.g. performs some data input). On certain events (e.g. button pressed) the updated form data is sent back to the server processing. On server side the data changes and the event are processed, the form is updated and sent back to the client side.

In the server centric approach all logical form processing is on server side - while the client is a "stupid" rendering engine. The client does not know what the business content of a form really is, but just know the arrangement of components and the data contained in the components.

The advantages of the server centric architecture are:

• The application development purely resides on server side. There is no split up between development groups.

- There is only one interface between the frontend client and the backend: the "form channel".
- There is no exposing of fine granular business APIs by the server.
- There is a guaranteed one-roundtrip-behavior between client and server.
- The size of the client is stable and independent from the number of forms to be processed. There is no need to constantly roll out the client (e.g. due to application bug fixes), bug fixes are on server side.

## CaptainCasa Enterprsie Client

#### Server-centric Architecture!

CaptainCasa Enterprise Client follows the server centric frontend architecture. The client part is a generic frontend based on JavaFX, the server side is built on top of the Java Server Faces (J2EE-JSF) framework. The client talks to the server via http(s), only changes are transferred between client and server ("lightweight roundtrips").



Java Server Faces is optimally suited to serve as Java standard for the backend processing. It allows the definition of descriptive layout definitions on the one hand and the direct manipulation of forms at runtime on the other hand. It provides strong concepts in the area of binding the form data and functions to server side bean processing.

Java Server Faces does not imply any further server side frameworks, but is open to adapt to any type of server side business logic processing. Frameworks like Spring, Hibernate, Enterprise Java Beans can be used.

CaptainCasa on the one hand uses Java Server Faces, on the other hand completely hides it behind its tools. There is no need at all to learn JSF before starting to develop with CaptainCasa - developers implicitly work on JSF without knowing.

### Component Library

CaptainCasa Enterprise Client comes with a huge set of standard components, that are directly and flexibly usable in application forms:

- Basic components: field, combo box, button, icon, ...
- Grid components: simple and complex grids, trees, dynamic rows, optimized loading of items
- Container components: pane, tab pane, titled pane, ...
- Graphics components: chart, svg, ...
- Drag&Drop, right mouse button menu, tooltips, online help, internationalization consistently supported through all components.

The component library is extensible by adding own components - both composite components and completely new components.

#### **Programming Model**

Programming Enterprise Client is completely done on server side. Forms are descriptively defined in XML layouts. Forms are bound to Java beans, which themselves may connect any business logic below.

Example: the following dialog is built using CaptainCasa Enterprise Client:

JavaFX in front of JSF					
First Tab	Second Tab				
Your Name	JavaFX Hello				
Result	Hello World, JavaFX!				

The server side layout definition is an XML definition:

```
<t:rowtitlebar text="JavaFX in front of JSF" />
<t:rowbodypane>
<t:row id="g_3">
   <t:tabbedpane width="100%">
     <t:tabbedpanetab padding="20"
rowdistance="5"
text="First Tab">
        <t:row>
           <t:label text="Your Name"
width="100" />
           <t:field
               text="#{DemoHelloWorld.name}"
width="200" />
        </t:row>
        <t:row>
           <t:coldistance
width="100" />
           <t:button
               actionListener="#{DemoHelloWorld.onHello}"
               text="Hello" />
        </t:row>
        <t:rowdistance height="20" />
        <t:row>
        <t:Tow>
<t:label text="Result"
width="100" />
<t:field enabled="false"
text="#{DemoHelloWorld.output}"
width="100%" />
        </t:row>
        </t:tabbedpanetab>
        <t:tabbedpanetab
text="Second Tab" />
     </t:tabbedpane>
   </t:row>
</t:rowbodypane>
```

The attributes of components either are directly defined or are bound by using an expression. The expression points into a server side bean processing:

```
package demo;
import javax.faces.event.ActionEvent;
public class DemoHelloWorld
{
    String m_name;
    String m_output;
    public void setName(String value)
    {
        m_name = value;
    }
    public String getName()
    {
        return m_name;
    }
}
```

```
}
public String getOutput()
{
    return m_output;
}

public void onHello(ActionEvent ae)
{
    if (m_name == null)
        m_output = "No name set.";
    else
        m_output = "Hello World,
        "+m_name+"!";
}
```

At runtime the dialog is loaded on server side, the components pick their data via expressions from the bean processing. A corresponding XML definition is sent to the client.

On client side the user keys in his/her name - when pressing the "Hello" button all data changes are sent to the server side.

On Server side the data changes are passed into the bean processing and the event associated with the button is called. As part of the event processing properties are changed.

After event execution the form data is recollected - finding out that some properties (e.g. "output" in the example) have changed. A corresponding delta-XML-definition is sent to client side and their updates corresponding components.

#### Tools

Project 🗢	<ul> <li>/ztest/a347.jsp ×</li> </ul>				. ●	Bean B	rowser Code Generator	Compo
oject Documentation A	bout /ztest/a347 isn					Filter	Editor's Log	
Project Selection / Project Info	e		~			Managed Se	erver Beans	Туре
	File Edit Tools	e 🦈 Refresh Preview Previe	w 🥑 Run		a Reload Server	4 0	Test347	Test347
Configuration	Component Tree	Live View Design View		Important	All Filled	Þ	<ul> <li>actualRootExpressi.</li> </ul>	. String
rrion 5.0.20140128.2	Components			Attribute	Value	Þ	🗇 class	Class
SIGN [5_0_20140120_2	4 trowbodypane	Height 100	•			Þ	height	int
roject Files	Q ⊕ 4 trow	Foldable Pane	Foldable Pane 🗢			Þ	pageName	String
a347.jsp	A Clabel / W100	00				D		Man
a35.jsp	4 100 trow						rootExpressionUse	String
a36.jsp	E tfoldablepane / w	e1					Tort27	. Sung
a38.isp	▷ III trow	0					Test29	Test57
a39.jsp							Test00	Testoo
a40.jsp							Testo9	Test39
💼 a41.jsp			able Parte			PQ	Iest40	Test40
a42.jsp		U					Iest41	lest41
a43.jsp		Pan				P Q	Test43	Test43
a44.jsp		able				► Q	Test47	Test47
a45.jsp		Fold	R			Þ C	Test50UI	Test50U
a40.jsp						Þ d	Test51	Test51
a48.jsp						Þ (	Test56	Test56
a49.jsp						Þ	Test57	Test57
💼 a50.jsp	¥					ÞØ	Test58	Test58
reate new JSP File	0			actionListener	#()	Þ ()	Test59	Test59
v file	isn	Messages			(List of Events)	Þ¢	Test60	Test60
nulata DICU atorialation		Message	Element Id	Style & Macri		Þ (	🗇 Test63	Test63
Rich (Java)		Mandatory attributes: actionList	en t:combo g_4	Style Variant		Þ C	Test67	Test67
(Java) HIML	text: #(d.Test347.height) width: 100	#(d.Test347.height) - 100				Expression		
Create						#Id Test34	(7)	

CaptainCasa Layout Editor

CaptainCasa comes with a set of tools, including:

- WYSIWYG (what you see is what you get) Layout Editor
- Bean code generator
- Internationalization/Translation tools

• Profiling tools

The actual Java development is done within an IDE of your choice (e.g. Eclipse, Netbeans, etc.).

## CaptainCasa Community Concept

User interfaces are a constant area of changes and improvements:

- New components
- New designs
- New ways of interaction

On the one hand this needs to be technically reflected by a solid and open architecture.

On the other hand this needs to be reflected by an organization that quickly reacts when it comes to improving the frontend framework accordingly.

CaptainCasa is a corporate community of independent software vendors, that are commonly using and driving the CaptainCasa Enterprise Client. The framework is released and updated regularly.

## Advantages of using CaptainCasa Enterprise Client

Using CaptainCasa Enterprise Client includes the following advantages:

- Your application's frontend is built using long term oriented technology standards to be supported over the next decade(s).
- The development of frontends is extremely efficient: there is only server side development, there is no expertise on JavaFX and/or on Java Server Faces required.
- The resulting frontends look modern, the look&feel can be easily adapted. Modern aspects of frontends (animations, multi touch, ...) are covered.
- The performance and stability of the client is high. There is no dependency to any browser installation.
- You are part of a community of professional software developers, joining their forces in the area of frontend architecture.

Licensing CaptainCasa starts with no-cost licenses (without restrictions but of course without warranty and without service) and reaches up to priced licenses (with warranty, with defined services, with code ownership).

Please set up contact to CaptainCasa by writing a mail to info@CaptainCasa.com.

CaptainCasa GmbH

Hindemithweg 13 D - 69245 Bammental

06223 484147

http://www.CaptainCasa.com info@CaptainCasa.com