USING A STATIC WEB PROJECT FOR AN EXTJS SAMPLE

John Sadd Fellow and OpenEdge Evangelist Document Version 1.0 August 2010





DISCLAIMER

Certain portions of this document contain information about Progress Software Corporation's plans for future product development and overall business strategies. Such information is proprietary and confidential to Progress Software Corporation and may be used by you solely in accordance with the terms and conditions specified in the PSDN Online (http://www.psdn.com) Terms of Use (http://psdn.progress.com/terms/index.ssp). Progress Software Corporation reserves the right, in its sole discretion, to modify or abandon without notice any of the plans described herein pertaining to future development and/or business development strategies. Any reference to third party software and/or features is intended for illustration purposes only. Progress Software Corporation does not endorse or sponsor such third parties or software.

This paper accompanies two videos in a series of presentations that set up a foundation for building a rich user interface on the desktop to communicate with an OpenEdge ABL application running on the server. The first of the two videos is entitled **Defining a Static Web Project in Architect to Support the ExtJS Library**. It's the first of two parts that finally get you all the way to creating the beginnings of a user interface in a non-OpenEdge open source library. The library chosen for this example is **ExtJS**, one of many that you can choose to build a browser-based UI that can get data and execute business logic in an ABL application. In the first video and the first part of this paper I show you where to download the library, and then define a Web project in Architect to hold the content you build using that library.

The company that makes ExtJS also markets an HTML5-based framework for mobile devices, and they've changed their name accordingly, to Sencha, so you'll find their website at www.sencha.com.



The product I'm going to introduce you to is their extended JavaScript library for browser-based UI development, called ExtJS, which you can find in their product list. ExtJS is what we would term a lightweight AJAX library. Lightweight doesn't mean that it doesn't do much. In fact, ExtJS, as the name implies, is a set of JavaScript extensions that provide support libraries for a large number of very powerful user interface controls, designed to be very full-featured as well as browser-independent, so they can save you a tremendous amount of time coding a rich browser-based UI. We categorize it as lightweight because the empasis is on the UI controls, and not on providing a substantial framework for structuring your client-side application code. In any case, you can see that there is both a commercially supported version of ExtJS, and also a no-charge public release for it:

OpenEdge Editor - OpenEdge Architect - C:\WebWorkspace		
Sencha - Ext JS - Downloads - Windows Internet Explorer		_ 🗆 ×
S S 🗢 🗣 http://www.sencha.com/products/js/download.php	💌 🗟 🐓 🗙 🚼 Google	<u>₽</u> •
File Edit View Favorites Tools Help × Google Start + @ + @ Share +	🕲 • 👝 • 🧭 Sidewiki • 🕴 Check • 👼 Translate • 🝗 AutoFil • 🌽 🔌 •	🔵 Sign In 🔹
👷 Pavontes 🦙 🖉 Suggested Sites 🔹 🙋 Free AOL & Unlimited Internet 🦿 Free I 💊 Sencha - Exit 35 - Downloads	fotmal 🍘 Web Sice Galery *	ols • 🔞 • '
EXT JS 3.2.2 SUPPORTED RELEASE Ext JS 3.2.2 (Support Subscribers Only) Patch releases are released on a monthly basis and are available to sup Thank you for supporting the Sencha Team. Released on June 2nd, 2010.	Download Changelog	
EXT JS 3.2.1 PUBLIC RELEASE Ext. JS 3.2.1 Includes full source code, build, HTML documentation and samples. Released on April 27th, 2010.	Download Changelog	
Ext JS 3.2.1 API Documentation Online API documentation	View	-
Ext.Air Pack Ext JS 3.1.1 components for use in Adobe AIR.	Download	
Ext.Direct Pack	Download	×
· · · · · · · · · · · · · · · · · · ·	i i i i i i i i i i i i i i i i i i i	(100% ·

The fact that there's a supported release may be of interest to you as you make a decision among the many libraries that are available. The free public release means that you can try out the library without any initial investment. There's also an ExtJS user interface design tool available (for a fee) to help you get started building a UI and generating the supporting JavaScript code for it, which could also be a consideration. In any case, you can download the **Public Release** from this page.

Once that's done, you have a zip file with the name ExtJS-3.2.1, or whatever the latest release is at the time you do this. You need to unzip that file. Then, to make all the support code it contains available to your work, you can copy or move the top-level folder to the directory where your Web server looks for static application files. In the talks in this series on configuring WebSpeed, I installed the Apache HTTP server. In that case, the target directory for static content is the Apache htdocs directory. I also renamed the top-level installation folder to just ExtJS without the version number, so that I can easily add this as part of the pathname of any references I make to the library in my client-side application.

So below you can see the top level of all the support code that's installed for ExtJS. There's extensive online documentation and examples which you can refer to.

💽 OpenEdge Editor - OpenEdge Archi	tect - C:\WebWorkspace			_0>
File Edit Navigate Search Project (OpenEdge Run Window Help			
] 📬 • 🗟 🖻 🛛 🔬 🗆 🕽 🏂	(+ Q + Q +] 🖉 +] 🖄 + 🕅 - 🕸	• 🗘 • 🔶 •		
📑 💐 OpenEdge Editor				
💞 Resources 🕺	C\Program Files\Apache Software Four	idation\Apache2.2\htdocs\Ext1	is 🗐	× • •
$\leftarrow \Rightarrow \otimes \Box$	File Edit View Fevorites Tools Help		4	7
WebSamples	😋 Back + 🕥 - 🏂 🔎 Search 👔	🖻 Folders 🛛 🕼 😥 🗙 🛤	9	
	Address 🛅 Criprogram Filesi Apache Software I	oundation\Apache2.2\htdocs\Ext35	💌 🔁 G	0
	Folders	× Name	*	-
	🖲 🧰 bin 🤤 ogi-bin 🗷 🧰 conf		pter s mples	
	E Control		s xurces	
	E C Ext35	i src		
	🖻 🧰 webspeed	Otest		
	E icons	a week	ish2	
	E Comanual	Z ext-	al.js	
at Outine 23 Witt DB S Drop	Co modules	2 ext-	all-debug.js	-
An outline is not available.	Ca Apoint	ant-	all-debug-w-comments.js	
	🗉 🧰 Apple Software Update	gpi-	JUDE OPDER HA	-
	E C ArcSoft)	E)
	16 objects (Disk free space: 11.5 GB)	4.49 MB	S My Computer	1.
	🖸 Console 🚶 Problems 🖉 Tasks	🚜 Servers 💠 🎲 OperEdge Se	rver Monitor	002×0 ⁻⁰
	Server +	State	Status	
	ASBroker1Conn	Stopped		
WebSpeedSamples				

Now I have an RIA library installed to provide support for a new user interface. If you want to use a library such as this one, you need to create a special type of project in Architect to hold the code you build for that interface. Under File -> New -> Project, you need to create a non-OpenEdge project, since it's not going to contain any ABL code. It's called a **Static Web Project**. It will hold HTML and supporting JavaScript files with static content, that is, HTML that is not generated on the fly as much WebSpeed content is, hence the term static web project.

DenEdge Editor - OpenEdge Architect - C	WebWorkspace	_ 🗆 ×
File Edit Navigate Search Project OpenEdg	a Run Window Help	
] <u>⊜ • ≋ ≏ ∡</u> ⊔ % ≫ • O	• Q • V • I to a North America III N	1
📑 🎳 OpenEdge Editor	Colort a winned	
	Create a Web project for static content (H1ML files) only	
WebSamples	Minarde-	
WebSpeedSamples	type filter text	
	Library and	
	General Height General	
	🕀 🗁 JavaScript	
	Weight Transformations	
	Copencage	
	Static Web Project	
	E Contraction Examples	
An cutine is not available.		
	(2) <8eck Next> Firish Cancel	
		0.0.0 - 0 - 0
	Server + Qate Qate	
	E ASBrokert Conn Stopped	
1		
🖓 🔁 WebSpeedSamples		

What's distinctive about this project type is that you can associate it with an HTTP server for it to use. For this example I just call the project ExtJSSample.

OpenEdge Editor - OpenEdge Architect - C:\	VebWorkspace	_ [] ×
File Edit Navigate Search Project OpenEdge	Run Window Help	
] 📑 • 🗟 🗎 🖾 🗆 🕲 I 🏇 • 🔘 •	Q_IAISISSSS	
😭 💞 OpenEdge Editor	OE New Static Web Project	
	Static Web Project	
	Configure Static Web module settings.	
WebSpeedSamples	Context root	
	ExtJSSample	
	Web content folder name:	
	WebContent	
An outline is not available		
	Cancel	
		S O S = 10
	Server A State Status	
	🗄 🚜 Abdroker1Conn 🛛 📷 Stopped	
U Lei WebbpeedSamples		

The **context root** on this page of the wizard is the target directory that content will be published to. If you watched the videos on Architect support for AppServer, it's very parallel to that. You develop content in Architect and then some portion of it gets copied – published – to a directory where the AppServer agents in that case, or the Web server in this case, will be able to find it to execute. This is illutsrated in this screenshot:



So ExtJSSample – the default name is just the name of the project I chose -becomes a subdirectory under htdocs, just like the ExtJS product folder. The **Web content folder name** is the source that Architect publishes from – the folder in the project where you develop static web content. That's all you have to define for a new static web project.

Now if you take a look at the properties of the new project, you can select **Project facets**. A facet is an aspect of a project type, in effect a unit of functionality that defines an aspect of the project's behavior. Here you can see that the **Static web module** facet is checked on because this was defined this as a static web project:

El Denni delle Faktor- Openida File Sak Navages Search Control Search Searchage Edate Searchage Edate Searchage Bestatsseels Bestatsseels Wetscheret Wetscheret Wetscheret Wetscheretseples	ar Architect – E. WerkWorkspace Photo: Coerfision: Pun. Window	Heb Project Facets Frogenetions: www.initialized.com Progenetions: www.initialized.com Progenetions: www.initialized.com Progenetics: www.initialized.com Progenetics: www.initialized.com"/>www.initialized.com Progenetics: www.initialized.com Progenetics: www.initialized.com Progenetics: www.initialized.com Progenetics: www.initialized.com Progenetics: <a href="https://www.initial</th> <th>Cetals Rurtmes Adds advened Jave</th>	Cetals Rurtmes Adds advened Jave
E cuaine 33 (FT DE S) [An outline is not available.	0		

That facet defines this as a project with content that can be published as a static web module. Note that this isn't an OpenEdge project, so this isn't where you put ABL procedures and classes. The OpenEdge checkbox here is enabled, by the way, but it probably shouldn't be, and don't think about checking that on, because it could cause unwanted behavior.

Now we've got a Web content folder in the project to put code into that will be published to the Web server. For this example I'm going to import a simple ExtJS sample into the folder so you can get a small taste of what's involved in coding with ExtJS. So under the **File** menu I **Import** from the **File System**.

The the function Such a map to prove the function of the back file system into an existing project. Import the back file system into	DE OpenEdge Editor - OpenEdge Architect - C:	WebWorkspace	_ [] ×
Contract to the contract of the block file system into an existing project. Set an input sources	File Edit Navigate Search Project OpenEdg	Rur OE Import	
Cover dage Edate Import resources from the bod file system risk an existing project. Seed. an import source: Seed] 🗂 • 🗟 🗋 🔏 🗆 🕲 🖄 • 🕗	Q Select	
Send an inport source: Send a	CoenEdge Editor	Import resources from the local file system into an existing project.	
Sector an import source: Sector an import source: Sector and the sector			
Image: State Stat		Select an import source:	
Image: Anthone Image: Anthone Image: Anthone Image: Anthon	E-125 ExtJSSample	bona fiber bard	
Contine in not weakled. Contin not weakled. Contine in not weakled. Contine in not	🖲 🗁 .settings	(Produced Based of Control of Co	
Article in the available Const. Next 2	- WebContent	😇 🇀 General	
VedSangies V	project	- 😥 Archive File	
WebGeedSurples W	- 🔛 WebSamples	Existing Projects into Workspace	
Conting To Control To Contro	- WebSpeedSamples	- Ele System	
Concluse in the available. Concluse in the available		- III, Preferences	
Contres II In Ces Prop Image: State Control International State Control Intern		🖯 🌝 OpenEdge	
Image: Southing in the standard in the standa		- La Database Connections	
Contract Source		Shared AVM Settings	
Coutine 12 Coutine		R C Tam	
Codine is 10 Ex255emple/webContext			
An outline is not available.			
An outre is not available.	Prov. 10 1000		
An outline is not available.	E Outine 23 Mm DB S DProp		
C Cancel	An outline is not available.		
Image: Const. Next. Press.			
C Cancel			
Or Creat Or State State 1286 1286 1286 If # AttricketCorn is Stopped 1286 1286			
C Cancel			
O < Deck Next> 7101 Cancel O 0			
C Content Next > Print Cancel Content Pattern			-
Serie Serie <th< td=""><td></td><td>Cancel</td><td>5.0.0 × 5.9.8</td></th<>		Cancel	5.0.0 × 5.9.8
Sterver State Statu If // R Stroker1Corn Stepped Stepped			200 a a
Image: State		Server - State Status	
T" (E Ex35emple/webContent		H ASBrokert Com	
T* C Exi35angle/webContext			
\Upsilon 🕲 Ext35Sample/WebContent			
	P T ExtJSSample/WebContent		
			,

And then I specify a directory I set up where I have the sample code. Below you can see first that there's a javascript folder. ExtJS provides support for building advanced user interfaces through coding in JavaScript, and using a set of libraries that define the kinds of controls you want to put into your UI. This is where that code would go. Then there's a resources folder. This is where other supporting files such as css files – Cascading Style Sheets – would go. Then you define HTML files to load into the browser to make everything happen.

0E OpenEdge Editor - OpenEdge Architect - C:\\W	bWorkspace	_ _ _ _ _ _
File Edit Navigate Search Project OpenEdge	🕼 💽 Import	
] 🗋 • 🗟 🚖] 🔏 🗆 🧐] 🏇 • 🕥 •	🛓 File system 🦳	
🔛 🎳 OpenEdge Editor	Import resources from the local file system.	
💏 Resources 🗙 📃 🗖		
E-12 ExtJSSample	From directory: C:\Doc\ExtJS Web Example\WebContent	
🕀 🗁 .settings	🖻 👿 😂 WebContent	
- 🗁 WebContent	🕀 🗹 🗁 javascript	
- x .project	E 🖉 🗁 resources	
	Filter Types Select Al Deselect Al	
BE Outline 😫 🐖 DB S 🛄 Prop	Into folder: Ext3SSample/WebContent Browse	
An outline is not available.		
	Overwrite existing resources without warning	
	C Create complete folder structure	
	Create selected folders only	
5		
1		
ExtJSSample/WebContent		

Viewcustomers.html is a simple HTML file that wraps the extended JavaScript code to retrieve and display customers from an ABL procedure running on the server. We'll take a look at the code for all of this a little later.

The **Import** has brought the sample files into my project.

It's worth mentioning here that I didn't bring all of the ExtJS support code that I installed into the project as well. All I did was to move it by hand to the same htdocs folder where the Web server will find my project's WebContent. I won't be editing any of that installation support code, so there's no need to make it part of the project. That would only slow down the development process when Architect is doing a build or searching for content to publish.

The next thing to do is to define a new server to associate with the static web project. Now in my environment I happen to have the **Servers View** open already, because in another of these sessions I use it to define an AppServer connection, but if it isn't already open in your session, you can open it by selecting **Show View** from the **Window** menu, and then under **Other**, and then **Server**, select the **Servers View**:

OF OpenEdge Editor - OpenEdge Architect - Ci Fle Edit Navigate Search Project OpenEdge] ➡ • ➡ △ ↓ ∡ ↓↓ ‰ ↓ ఈ • ●	Weldweikener: ■ 55 Weldwei Heip = 4 → 4 → 1 1 = 11 + 10 ← + - +
Control Contro	Image: Second and Se
St Outher II Tot Sun Dropun D	Conside C
C° (C) Ext25Sample/WebContent	

The screenshot below shows that view with the AppServer connection that I defined elsewhere. In the same way as I did for the AppServer, I can right-click in the View, and select **New ->Server**.



In this case I don't want an AppServer connection. Instead I expand the **Basic** node, and select **HTTP Server**. This is going to be a definition of my Apache HTTP Server for Architect to associate with the static web project.

OF OpenEdge Editor - OpenEdge Architect - C:	WebWorkspace	_ 🗆 X
He Edit Navigate Search Project OpenEdge	• Run Window Hep	
12 B ^R CranEdua Editor	0E New Server	1
P Descented 12	Define a New Server	
	Choose the type of server to create	
E 🖉 ExtJSSample		
🕀 😂 .settings	Server's host name: locahost	
WebContent WebContent H	Download additional server adapters	
E - Corresources	Select the server type:	
wiewcustomers.html		
- 👕 WebSamples	Basc HTTP Preview	
	HTTP Server	
	Progress 5	
🖹 Outline 🕮 🛗 DB S 🗔 Prop 🖓 🗖	Dublisher ABI code to an OpenEdine analysistico cessor	
An outline is not available.	Posibiles nucleose to an opencage approximiser ren	
	Server name: OnenEdine 10.28 at localiset	
	Server runtime environment: OpenEdge 10.28	
	Considure rundime environments	
		* 0 / = 0 - 0
	Cancel Cancel	
		' I
1 m m	L	
		1

I just set the **Server name** to HTTPServer, and the way I associate it with my Apache server is to point it at the htdocs directory that server uses, which is here under the Apache install location:

05 OpenEdge Editor - OpenEdge Architect - D	WebWorkspare	
Ele Edit Navinate Search Project OpenEdos	Run Window Help	A
	0	
E? 20 Constitute Editor	0 New Server	
El el obercode corre	UTTO Comuna	
S Resources 🖾	HTTP server	
$\leftarrow \rightarrow \langle q \rangle \equiv \langle q \rangle \nabla$	Specify the HTTP server information	
E- 🐸 Ext3Sample		
(i) 😂 .settings	Name:	
WebContent	HTTPServer	
a javasoript	Publishing Directory:	
siewrustomers.html	Browse For Folder	
.project	Colord MTD annual data matched	
- 👕 WebSamples	Select HTTP server document root.	
WebSpeedSamples		
E cuttre sz Pros Prop	Image: Applied Software Foundation Image: Applie	
	(7) <back next=""> Presh Cancel</back>	☆ O ⊵ = (2 ⊂ □
] 📭 🗈		1

Remember that this is the same location where I moved the ExtJS install, so it can be found by the server even though it's not part of the published content in the project. This next screenshot shows the full pathname to that publishing directory that you specify in the **HTTP Server** wizard:

De OpenEdge Editor - OpenEdge Architect - C:	WebWorkspace	_ 🗆 ×
📑 雛 OpenEdge Editor	DE New Server	1
Resources 22	HTTP Server	
	Specify the HTTP server information	
ExtJSSample		
🗵 😂 .settings	Name:	
G 😂 WebContent	HTTPServer	
E prescurces	Publishing Directory:	
viewcustomers.html	C:(Program Files)Apache Software Foundation)Apache2.2(htdocs Browse	
WebSamples		
🕃 Outline 🕮 📅 DE S 🖾 Prop 🖓 🗖		
An outline is not available.		
		50 C = D = 0
	(2) < Back Next > Finish Cancel	
		-
] 📪 🖻		

In the following wizard page, the **HTTP Port number** default of 80 is correct, and I don't need a **URL Prefix**. Note that I leave the option checked to enable publishing to this server. That will copy code from **WebContent** to the new folder under htdocs whenever code in the source folder changes:

De OpenEdge Editor - OpenEdge Architect - C:	WebWorkspace	
File Edit Navigate Search Project OpenEdge	e Run Window Help	
] 🗂 • 🖩 🗠] 🔏 ⊔ 🗞 🏇 • 🔾	• 💁 • 🛛 🖉 • 🖓 = 🖓 = 10 + 10 + 10 + 10 + 10 + 10 + 10 + 10	
📑 🎳 OpenEdge Editor	OE New Server	
Resources 22	HTTP Server	
	Specify the HTTP server information	
E-St Ext 25 annia		
B-@ .settings		
E 🗁 WebContent	HTTP Port:	3
III 🗁 javascript	URL Prefix:	
viewcustomers.html	Enable publishing to this server	
.project		
- 📴 WebSamples		
- WebSpeedSamples		
Press M Plans Rev R.P.		
Coutine 23 YE DB S L Prop		
An outline is not available.		
		\$ O S = D = D
	(?) < Back Next > Finish	Cancel
J T° 🗈		

If you watched the AppServer videos, you've seen this next page of the new server wizard before. For an AppServer, it looks for AppServer-enabled projects, those with that facet checked on. In this case, an HTTP server looks for projects with the static web module facet checked on, so it finds my new sample project. I add that to the list of projects associated with this server, and the server definition is complete:

DenEdge Editor - OpenEdge Architect - C:	WebWorkspace	
File Edit Navigate Search Project OpenEdge	Run Window Help	
] 🗈 • 🗟 🖢] 🔏 ⊔ 🦦 🎄 • 💽	• Q. • A • 2 • 2 • 2 • 4 • • • •	
📑 🎳 OpenEdge Editor	DE New Server	
💐 Resources 😫 📃 🗆	Add and Remove Projects	
+ + 4 🔳 🕸 🏹	Modify the projects that are configured on the server	
🕀 🧭 Ext355ample		
🗵 😂 .settings	Move projects to the right to configure them on the server	
e 🕗 WebContent	Available projects: Configured projects:	
H in resources	🔂 Ext3SSemple	
i viewcustomers.html	ANA	
.project		
- WebSamples	Click the Add button	
	Add All >>	
at Outline 23 WT DB S LLI Prop	<< Remove AI	
An outline is not available.		
		50 C = 0 C
	Cancel Cancel	
1 📬 🖻		

Below you can see the new server in the **Servers** View. I can expand it, and see the one project that I have associated with the server. The last button in the **Servers** View toolbar over on the right is the **Publish** button:

DE OpenEdge Editor - OpenEdge Architect - C:	\WebWorkspace		
File Edit Navigate Search Project OpenEdg	e Run Window Help		
] 📬 • 🖩 🛆 🔬 ⊔ 🗞 🅸 • O	• 💁 •] 🖉 •] 🖢 • 🕅 • •	- (+ + -) +	
📑 🚭 OpenEdge Editor			
🐇 Resources 😫 🗖 🗖			- 0
← → ☆ = 🎕 ♡			
🖻 🧭 Ext3SSample			
🖲 🗁 .settings			
e 😕 WebContent			
E Savascript			
i viewcustomers.html			
.project			
- WebSamples			
WebSpeedSamples			
Cudine 23 WTT D6 S Prop			
An outline is not available.			
	🖸 Console 👔 Problems 🧔 Tasks	Servers 22 10 OpenEdge Server Monitor	002×0
	Server +	State Status	
	🗉 📑 ASBroker1Conn	Stopped	Click the Publish to the server
	HITPServer	Started	DUILON
	Lig ExtJSSample		
T 🗈			

Pressing the **Publish** button immediately publishes anything it finds in the **WebContent** project folder. Now that the initial publish is done, I can go back to the **htdocs** folder in the Apache install, and see the **ExtJSSample** folder under it. This is what I named as the context root back in the project definition. And here you can see that the files in the project's **WebContent** folder have been copied here:



One more thing to point out is that if you double-click on a server in the **Servers** View, an **Overview** display comes up with a summary of all the settings that have been defined, including ones that got defaulted. Here you can adjust whether new and modified content is published automatically, for instance, and how frequently Architect should check for it:

05 OpenEdge Editor - HTTPServer - OpenEdg	a architect - C\WebWorkenace	
File Edit Navigate Search Project OpenEdg	e Bun Window Help	
	••••••••	
🔛 🚳 OpenEdge Editor		
Sesources 🖾 🗖 🗆	HTTPServer 🐹	
(⇒ ⊗ ⊟ 🤹 🏹	8 Overview	
ExtJSSample	a over view	
🗄 🇀 .settings 🖻 🇀 WebContent	General Information Specify the host name and other common settings.	Publishing Modify settings for publishing.
🗄 🗁 javascript	Server name: HTTPServer	O Never rubish automatically
E C resources	Host name: localhost	Automatically publish when resources change
in viewcustomers.ntmi	Runtime Environment: HTTPServer	Dubliching internal (in correctly)
WebSamples		Publishing incerval (in seconds).
WebSpeedSamples	HTTP Server Specify publishing directory and URL.	Timeouts
	URL Prefix	
	HTTP Port: 80	
	Enable publishing to this server.	
© Outine ∞ m DBS □ Prop □		
An outline is not available.		
	Overview	
	🕞 Canada 🕑 Buchland 🖓 Santa 🖓 Canada	
	Console To Problems Console was an Servers to State Openical	
	T AStrokert Corp. Scored) Status
	HTTPServer Started	Synchronized
	Strassample	Synchronized
	11	and the second s
		Writable

This brings us to the end of the content covered in the first part of this two-part video session on preparing to use ExtJS. The second video session walks you through a few of the highlights of the kind of code you need to write to use a library like ExtJS, and shows you the output from a simple example.



In this part of the presentation on setting up to use the ExtJS library, I examine the code for a very simple example and show you the output that results from running it.

The first code example shown below is **viewcustomers.html**. This HTML file is what actually gets loaded into the browser. In reality it serves as a wrapper around the extended JavaScript code that defines the UI, which could be embedded here, but is better to reference as a separate file. I'll say just a few things about this example.

```
<html>
<html>
<html>
<html>
<html>
kead>
</title>View Customers</title>
link rel="stylesheet" type="text/css"
href="/extjs/resources/css/ext-all.css">
k rel="stylesheet" type="text/css" href="resources/stylesheets/myapp.css">
<script type="text/javascript" src="/extjs/adapter/ext/ext-base.js"></script>
<script type="text/javascript" src="/extjs/adapter/ext/ext-base.js"></script>
<script type="text/javascript" src="/extjs/ext-all.js"></script>
</script type="text/javascript" src="javascript/viewcustomers.js"></script>
</script type="text/javascript" src="javascript/viewcustomers.js"></script>
</script type="text/javascript" src="javascript/viewcustomers.js"></script>
</script type="text/javascript" src="javascript/viewcustomers.js"></script</script>
</script type="text/javascript" src="javascript"</script"></script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script</script"></script type="text/javascript"</script"</script"></script type="text/javascript"</script"</script"></script type="text/javascript"</script"</script"></script type="
```

The title is the title of the HTML page; that will show up as the label for the tab where output is displayed if you run it from Architect.

Next is a link to one of the install files, ext-all.css, where standard support for cascading style sheets is located. Notice that the pathname to this file starts with extjs. That's the installation folder that I renamed ExtJs and moved to htdocs, so all pathnames here are relative to that directory.

Then there's a reference to a specific stylesheet of my own, which in this case is just a placeholder.

The next two references to ext-base.js and ext-all.js are a standard part of ExtJS definitions, to JavaScript support code the library uses.

Following this there's a reference to my own JavaScript file, viewcustomers.js, which you've seen in the WebContent javascript directory.

Finally, the div tag is the HTML definition of where the customer grid defined in the JavaScript gets placed in the overall UI. And that's it. The rest of the work is done in the JavaScript.

The next blocks of code show viewcustomers.js, where all the work of defining the controls and the data access is done.

I'll just point out a few of the elements of the JavaScript. There's an example of a reference to the extended libraries, named **Ext**. The **namespace** function lets me define a namespace for all the elements of my application.

```
/* repoint the blank image to the local copy */
Ext.BLANK_IMAGE_URL = '/extjs/resources/images/default/s.gif';
/*
 * define the MYAPP namespace that contains all of our application functionality
 */
Ext.namespace('MYAPP');
```

When I open the code in the Architect editor, an error icon appears to the left of the line that references Ext.namespace. Why is this here?



Well, one of the strange things about programming in JavaScript that takes some getting used to for ABL developers, or developers in most other languages for that matter, is that a great deal of the definition of a JavaScript class is done on the fly, dynamically, in executable statements encountered as the file is evaluated. Here in the code segment shown above, for instance, is the beginning of the definition of a function in the MYAPP space called app, which defines two variables. The first defines the URL to the webspeed broker, the same as I defined in Architect when I configured wsbroker1. That's how the JavaScript will communicate with the WebSpeed broker that executes ABL procedures. The second variable APP has a function called init, which actually defines the grid control. This definition all gets executed dynamically, so the editor is really not able to help you determine whether your JavaScript is valid or not. That's just one of the realities about programming in JavaScript.

Further down, you can see the name of an ABL procedure that I'm going to invoke on the webspeed broker's URL, so I append the procedure name to the URL.

var customerURL = WEBSPEED BASE + 'getcustomers.p';

Then there's an HTTPproxy definition, which will establish the connection to WebSpeed, using its URL together with the name of the procedure to run.

Then there's a variable that represents a JsonStore, a place to hold data in the client in the JSON format – JavaScript Object Notation. OpenEdge supports generating this format directly from a temp-table or ProDataSet, as you'll see when we look at getcustomers.p. The root corresponds to what is identified in the ABL as the serialize-name, basically a name for the top-level node in the data tree. And the fields map to fieldnames in the temp-table I'll populate with customers.



Finally we get to the one UI control in the application. It's an ExtJS GridPanel, a grid control. The renderTo property is set to customerGrid. That name is the id in the div tag we looked at back in the html file. So this definition will create an instance of a grid control that will get displayed where that simple HTML definition says to display it. The store property is the customerStore defined just above.

```
var customerGrid = new Ext.grid.GridPanel( {
    renderTo : 'customergrid',
    height : 300,
    width : 500,
    store : customerStore,
```

Then there's a series of column definitions for the grid. Each id maps to one of the field names in the store, which in turn matches a field in the temp-table over in the ABL we'll look at in a moment.

```
columns : [ {
       id : 'Name',
       header : 'Customer Name'
}, {
       id : 'CustNum',
       header : "Customer Number"
}, {
       id : 'Address',
       header : "Address"
}, {
       id : 'City',
       header : 'City'
}, {
       id : 'State',
       header : 'State'
}],
```

Then there are a couple of specialized property definitions, such as making the grid single selection.

Finally here, the load function tells the store to load its data, which it does by invoking the procedure at the end of the URL.



That's the end of the init function in app. Ext.onReady gets called to create the UI, and it invokes this init function to get everything rolling.

```
/*
 * invoke the 'init' method in our 'app' object. The second parameter is the
 * scope where the method is invoked
 */
Ext.onReady(MYAPP.app.init, MYAPP.app);
```

So that's a simple but complete ExtJS UI definition.

Now I've been referring to WebSpeed here and the ABL procedures that get run in that context, so it's time to take a look at that. I open the **WebSpeedSamples** project that I created in another of these presentations.



My static web project gives me a place to develop HTML and JavaScript code and publish that code to the Web server. This WebSpeed project represents the other half of the job. This is where I define ABL code that gets executed in the WebSpeed broker. The wsbroker1 URL that I defined in the JavaScript example connects the two together at runtime, but at development time they give me two different Architect environments to work in, with different characteristics.

Here's the ABL procedure that the JavaScript will invoke in WebSpeed:

```
{src/web2/wrap-cgi.i}
output-content-type ("text/json").
define temp-table ttCust like Sports2000.Customer.
temp-table ttcust:serialize-name = "customers".
for each Sports2000.Customer where Customer.CustNum < 10:
    create ttcust.
    buffer-copy Customer to ttcust.
end.
temp-table ttCust:write-json("stream", "webstream", true).</pre>
```

It uses the standard CGI wrapper include file that's part of WebSpeed.

It defines the output content type as being Json.

Then it defines and populates a temp-table with a few customer records. Remember the root property of the customerStore back in the JavaScript? That corresponds to the serialize-name here in the ABL, a header for the data in the temp-table as it gets converted to JSON.

Finally, the write-json built-in ABL function executes on the temp-table, and in a single statement, converts the data in the temp-table to JSON format and writes it to the webstream where it is returned to the customerstore defined in viewcustomers.js. So once you get the support pieces set up, the process is really very straightforward and well supported by the features of OpenEdge 10.

In my testing environment I also have the **Server Monitor** View open from my work with AppServer, and in that view, which you could also open from the **Open View** option on the **Window** menu, there's a button to provide access to OpenEdge Explorer from within Architect. You can drill down in Explorer to the **WebSpeed** node, select the default WebSpeed broker, wsbroker1, then select **Broker Control**, and **Start WebSpeed**.

OE OpenEdge Editor - http://localhost:9090/	athom.htm - OpenEdge Architect - C:	WebWorkspace		
File Ealt Navigate Search Project OpenEo	e kun window metp	<u>~</u>		
The second secon				
Recover 22	🔝 siawoutomere html 🛛 🔊 siawou	tomers is	OneoEdos Evolorer 32	- F
	admin on NB IOHNXP1 (contain	vars: 1 offlina: 0 unknown: 0)	oponeage exponent of	0 0 0
viewcustomers.is	Resources Ontions 2H	alo	Close	
e resources				Running (11 Seconds)
Viewcustomers.html	a Resources	webSpeed: nbjohnxp1.ws	broker1	Not Polled
webSamples	Sort by: Container •		Pc	bil Count: 0 Fail Count: 0 (0.0%)
🖻 🚰 WebSpeedSamples	= 🗇 nbiohnxn1	WebSpeed Status		
JL Procedure Libraries	E CopenEdge	Host	NBJOHNXP1	
project	App Server Internet	Broker:	ACTIVE	
- propath	App Server	Operating mode:	Stateless	
getoustomers.p	Database	Broker statistics available:	True	
ShowCustomers.html	Messengers	Should register with NameServer?	True	
	Name Server	Command and control		
8 Outline 🛛 📅 DB 5 🗆 Prop 🔍 🖓	B WebSpeed	Broker Control Start or stop broker	Monitoring Plans	s on which to allert
An outline is not available.	wsdynamics1 Web Services Adap	Agent Pool Control Add or trim agent pool	Configuration Adv Suggests rule thresh FathomTrendDatebas	NSOF olds based on data in the
	4	Log File Monitor of Broker	Log File Monitor	of Agents ich to alert
	94-	1242		-
	Constant Constant O Testa	H Carrow Marker Street Marker M		
	AStroker1Conn	servers Str Opencage server Monicor 23	, <u>x v c</u>	
	Broker name: astrokar1 Operatin	g mode: Broker status:	Broker por	t: 0 Broker P1D: 0
	Arthur anerther Agent Summary Agent Detail AppServe	State		
T* t Done				

Once I confirm that the broker is now active, I can go back to my static web project, select the **viewcustomers.html** file that represents the top of my client application, and run it:



I get prompted whether I want to run it as an **OpenEdge Application** or on a **Server**. Of course I want to run it on my **HTTP Server**. My one HTTP Server is displayed, and that's the one I always want to use for files in this project. That's as much as I need to specify:

OF OpenEdge Editor - Ext ISS apple (WebConte	nt/viawewtomars.html - OnenEdae Architect - C\WebWorkenace	
File Edit Source Navigate Search Project (0E Run On Server	
	Run Dn Server	
	Calend which ensure to use	
G GpenEdge Editor	Select which server to use	
🖏 Resources 🖾 👘 🗖		
$\leftarrow \rightarrow \langle a \rangle \equiv \langle b \rangle \nabla$	How do you want to select the server?	×
viewcustomers.js	 Choose an existing server 	
🕀 🗁 resources	C Manually define a new server	
- i viewcustomers.html	Select the server that you want to use:	
.project	type filter text	
WebSamples		- 1
Weospeedsamples	E 😂 localhost	esheets/myapp.css">
- D .dbconnection	i Hillserver is started	base.js">
.project		cipt>
- Di propath		s.js">
getcustomers.p		
getorders.p		
ShowCustomers.html		
as coulde to the best of Prop		
E 🔄 html		
	Publishes and runs projects on an HTTP server.	
	The Advancement of the second second sector with a second se	
	 Hways use uns server when running uns project 	
		1
	(?) <8eck Next > Finish Cancel	
📄 陀 💿 html	WINGONG STIELCENSOL E	

Now viewcustomers.html is loaded, invokes viewcustomers.js, which loads its customerStore by running getcustomers.p in the webspeed broker, and voila. Below you can see the HTML page displayed right here in Architect, with the Extended JavaScript gridPanel control displayed in the customergrid slot in the HTML:

DE OpenEdge Editor - http://localhost/Ext35	Sample/viewcustor e Run Window H	ners.html - Openi eh	Edge Architect - C	:\WebWorkspac	e			
	· Q 1 1/2 ·		-					
T 45 OpenEdge Editor								
Resources 12	viewcustomers.	html 🙆 viewo	ustomers.is	getcustomers.p	Mew Customers	8	0	
		http://ocabootify	* 155 ample (view rust	omerr bini		-		
viewoustomers.js		Loop Alexan poster				<u></u>		
esources	Customer Name	Customer Number	Address	City	State			
wiewcustomers.html	Lift Tours	1	276 North Drive	Burlington	AM			
- project	Urpon Frisbee	2	Rattipolku 3	Oslo	Uusima			
E 12 WebSpeedSamples	Hoops	3	Suite 415	Atlanta	GA			
	Go Fishing Ltd	4	Unit 2	Harrow	Middlesex			
- 📄 .dbconnection	Match Point Ten	5	66 Homer PI	Boston	MA			
.project	Fanatical Athletes	6	20 Bicep Bridge	Montgomery	AL			
.propath	Aerobics valine	7	Petolantie 3	Tikkurila	Uusimaa			
P getorders.p	Game Set Match	8	Box 60	Deatsville	AL			
ShowCustomers.html	Pihtputaan Pyora	9	Putikontie 2	Phtipudas	Etela-Savo			
BE Outline 23 To S Prop								
An outline is not available.								
								*
	📮 Console 🔝 F	robiems 🕗 Tasks	👫 Servers 🙀 O	ipenEdge Server M	anitor 23	🖉 📀 🔗 😽 🤻 🕲		
	ASBroker1Conn							
	Broker name: asbr	okar1 Opera	ting mode:	Broker	rstakus:	Broker port: 0 Broker PID		
	Agent Summary A	gent Detail AppServ	ver State	Arthur	fante:			
Done								

And that's all there is to it.

This two-part session covers a lot of territory, and certainly there's a lot to learn with any of the RIA libraries and frameworks that are available. But I hope the key message of this presentation is clear: that OpenEdge 10 provides the language elements that you need share to data with many client-side environments, as JSON or in other standard formats, and OpenEdge Architect lets you define the kinds of projects and servers you need to use to do development of all the different pieces of a distributed application with a modern user interface.