In this chapter we will explore some of the Seaside from the web browser's point of view, including examples available with the default installation of Seaside.

1. Launch the Seaside One-Click Experience (see Chapter 1 for details) and open a web browser on <a href="http://localhost:8080/browse">http://localhost:8080/browse</a>.

This page gives a list of the applications that have been registered with Seaside through the WADispatcher singleton (available by evaluating the Smalltalk expression 'WADispatcher default'). The dispatcher looks at the URL requested, and dispatches the request to one of these registered entry points. This is how the request from the client browser gets to the appropriate Smalltalk code (such as our 'HelloWorld' application).



2. Click on the config link to see the 'Dispatcher Editor.' This tool sets up the information for the Dispatcher Viewer we saw above.

Dispatcher at	t/		-tde
Add Clear D	efault   Open		seasing
./	Dispatcher	Dispatcher: /	
rowse (*)	Application		
onfig	Application	Filters	L
xamples/	Dispatcher	Possible filters:	
iles	File Library		
ello	Application	Innerited Configuration	Lave
avascript/	Dispatcher	WAAccessIntervalReapingStrategyConfiguration	
easide	Legacy redirection	· · · ·	
tatus	Application	Assigned parents:	
ests/	Dispatcher		
ools/	Dispatcher		
elcome	Application		
or of one	Application		
		Configure	
		General	
		Server	
		Server Path (unspecified)	
		(Arely) (Presel	

The left column lists each of the entry points and the initial selection is the root, which is just a dispatcher to some other web component. There are various types of components, including other dispatchers (examples/, javascript/, tests/, and tools/), applications (browse, config, hello, status, and welcome), a file library (files), and a legacy redirection (seaside). Selecting any component in the left column gives you some configuration information about that component.

Note that there is a star next to the *default* component. The default component is the one provided by Seaside if you select the root (e.g., <u>http://localhost:8080/</u>). In chapter 2, step 13, we clicked 'Yes' when Seaside offered to change the default application from 'welcome' to 'browse'.

Note that near the top of the config page there is a 'Clear Default' button. Click the button then open a new web browser (window or tab) and navigate to the root. You should get an error.



Switch back to the config page (at <u>http://localhost:8080/config</u> if you did not save that page), select 'hello' from the links on the left, and then click the 'Set Default' button. This will make our Hello World application the default Seaside application.



In your web browser, switch back to the tab or window with the error and refresh (or enter <a href="http://localhost:8080/">http://localhost:8080/</a>). You should now see our Hello World application at the root.



## Hello world!

## 2016-09-04T15:02:04.024752+08:00

The config page provides a great deal of other configuration capabilities and demonstrates some of the capabilities of a Seaside component. We will not examine this area further at this time.

3. Enter <u>http://localhost:8080/browse</u> to return to the list of components and select 'Status' (or enter <u>http://localhost:8080/status</u> in your web browser's address bar). This will show you a list of tabs containing information about your Smalltalk environment. The first tab identifies the location of the file containing a snapshot (image) of the object space when it was last saved to disk. It also gives you the opportunity to make another snapshot of the current object space (by saving the image).



The 'Seaside' tab identifies the version of Seaside installed, and lists the version of various installed packages.

⇒ C 11 0	localnost:8080/status/WASeasi	deversionStatus?_s=40cea8E1KMX7Hjjy&_k=VKAJ8s 😭 🛈	0
Image Seaside	VM Allocator GC M	femory Processes OS Space (slow!)	
Seaside Version	: 3.1.2		
Grease Version	: 1.1.10 (Pharo)		
Packages	: Pack	age Version	
	Cor	met	
	Comet-Core	pmm.55	
	Comet-Pharo-Core	lr.6	
	Comet-Tests-Core	lr.11	
	Gre	ase	
	Grease-Core	JohanBrichau.94	
	Grease-Pharo30-Core	JohanBrichau.16	
	Grease-Pharo40-Slime	JohanBrichau.3	
	Grease-Tests-Core	pmm.99	
	Grease-Tests-Pharo20-Core	JohanBrichau.12	
	Grease-Tests-Slime	pmm.19	
	Javasci	ript	
	Javascript-Core	pmm.97	
	Javascript-Pharo20-Core	JohanBrichau.4	
	Javascript-Tests-Core	pmm.71	
	Javascript-Tests-Pharo-Core	lr.1	

The final tab scans the entire object space and generates a report of objects in your object space. In this example, it took over 60 seconds to determine that there are almost 150 thousand strings in my environment, taking up about 10 MB of space.

🕒 🔍 🗋 Space Usage per Class 🛛 🗙 📃	5						Jame
$ \rightarrow$ C $(1)$ localhost:8080/status/WASpace	eStatus?_s=	40cea8ET	KMX7Hjjy8	k_k=s6KyrTs	sGAOotX 🕁	1	6
Image Seaside VM Allocator GC 1	Memory	rocesses	OS Spa	ice (slow!)			
Class	de space #	instances	inst space	percent			
ByteString	2625	149869	10467352	21.60 %			
Array	3084	178447	8359796	17.20 %			
CompiledMethod	23292	96494	6251552	12.90 %			
ByteArray	8017	1206	5334252	11.00 %			
Bitmap	3748	1090	3240612	6.70 %			
ByteSymbol	1350	67839	1824912	3.80 %			
MethodDictionary	2466	12436	1181756	2.40 %			
IdentitySet	305	54612	873792	1.80 %			
Association	858	68967	827604	1.70 %			
Float	14228	65807	789684	1.60 %			
MetacelloPackageSpec	6203	8366	468496	1.00 %			
MorphExtension	3124	6264	425952	0.90 %			
Point	7010	35491	425892	0.90 %			
LargePositiveInteger	1111	49891	400680	0.80 %			
OrderedCollection	5176	19296	385920	0.80 %			
WeakArray	875	1215	382484	0.80 %			
MCMethodDefinition	3299	10756	344192	0.70 %			
Protocol	1231	26136	313632	0.60 %			
w Session Configure Halos Profile Memory XH	TML 2/6749	7 ms	*****	~ <i>~</i> ~ ~			

4. Return to <u>http://localhost:8080/browse</u> and select 'examples' (a directory of other entry points). You should see the following:

Dispatcher at /examples					
./	Dispatcher				
/	Dispatcher				
counter	Application				
demo.rss	RSS feed				
examplebrowser	Application				
multicounter	Application				
1					

Click on the 'counter' and you will see the traditional Seaside example of an application that keeps state on the server. We will be using a slightly more complex example to explore some Seaside basics.



5. Use the web browser's back button to return to the examples page and then select the 'javascript/' link. Here you can find demos (with code) showing usage of jQuery and Scriptaculous.

Dispatcher at /javascript					
./	Dispatcher				
/	Dispatcher				
jquery	Application				
jquery-ui	Application				
scriptaculous	Application				
scriptaculous-components	Application				

Select the 'jquery' link to see a page describing Seaside support for JavaScript.



Select the 'Ajax' link in the left column. This gives a demonstration of how clicking on a button sends an Ajax request to Seaside to execute Smalltalk code on the server and send back a response (the current date and time).

## jQuery Functional Test Suite

Say it in Smalltalk, Do it with jQuery

Ajax	Ajax and DOM Manipulation		
Ajax	Demo		
Request	2016-09-04T15:12:28.67406+08:00		
JSON	Replace Prepend Append		

6. Return to <u>http://localhost:8080/browse</u> and select 'tests' and then 'functional.' The Functional Seaside Test Suite shows a drop-down list of various tests, with the test selected. The WABatchFunctionalTest encapsulates a list of class names and shows a horizontal list of the letters of the alphabet that can be used to jump to a particular place in the list. You can also use the previous ('<-') and next ('->') links to move through the list.



7. To reach the second test, click on the drop-down list showing 'WABatchFunctionalTest' and select the second item ('WAButtonFunctionalTest'). This will demonstrate the difference between various HTML button types ('submit', 'reset', and 'button'). If you enter text in the input field and click the Submit button, the new value will be returned to the server. If you enter text in the input field and click the reset button, the old value will be restored.

Seaside Functional Test Suite
← WAButtonFunctionalTest
Value: a text Input: a text
Submit
Clicking the button should submit the form and update the value in "Value:" with the value in "Input"
Submit
Reset
Clicking the button should not submit the form and reset the value in "Input"
Reset
Push
Clicking the button should not do anything.
Push

 The WACanvasTableFunctionalTest demonstrates Seaside's ability to generate various tablerelated HTML elements, including , <caption>, <colgroup>, <thead>, <tfoot>, , >, , and . This test gives you an example to use if you want to build a table in Seaside.

Seaside Functional Test Suite							
			HTML	4.0 entities			
	Chavaster	Entity	Desimal	Hoy	Renderin	g in Your B	rowser
	Character	Entity	Decimar	пех	Entity	Decimal	Hex
	non-breaking space						
	ampersand	&	&	&	å	&	&
	less than sign	<	<	<	<	<	<
	greater than sign	>	>	>	>	>	>
	euro sign	€	€	€	€	€	€
			5 enti	ties shown			
	Currencies against Swiss Franc (CHF)						
			Currency	Rate			
			EUR	1.7			
			USD	1.3			
			DKK	23.36			
		ſ	SEK	19.32			

9. The WAFlowConvenienceFunctionalTest demonstrates three simple built-in components. The first component is the 'WAChoiceDialog' in which Seaside presents a list of options, add a prompt message, and then will return one of the items in the list of options or 'nil' if the user clicks 'Cancel'. (The object 'nil' is the single instance of UndefinedObject, an object that is the something that represents nothing in Smalltalk.) Any subclass of WAComponent (like our 'HelloWorld' class) can send 'chooseFrom:caption:' to self providing a list and a string and get back an answer.

WAConvenienceTest Restart
What's your favorite Cheese?
Greyerzer 🛟 Ok Cancel

If you click 'Ok' on the previous component, Seaside immediately presents the

'WAYesOrNoDialog' in which a message is presented with two buttons, 'Yes' and 'No' that will be returned as 'true' or 'false' Booleans. Any subclass of WAComponent can send 'confirm:' to self with a string and get back a Boolean.



If you clicking on 'Yes' in the previous component, Seaside immediately presents 'WAFormDialog' in which a message is presented with a simple 'Ok' button. Any subclass of WAComponent can send 'inform:' to self with a string to create this page.



10. The WAInputGetFunctionalTest demonstrates a variety of HTML input tags. By reviewing these examples you can get ideas of what can be done and then go to the test class to look at sample code. (As you progress through this tutorial you will learn more about finding classes in Pharo. In this example you would right-click on the first column of a System Browser, select the 'Find Class' menu item, enter WAInputTest' when prompted for a class name fragment, and then select 'WAInputTest' from the list.)

Seaside Func	tional Test Suit	e				
		-				
WAInputGetFunctionalTest						
This form uses a HTTP GET request. The upload is not supposed to work.						
	Control	Print String				
Text Input		nil				
Text Area		nil				
Hidden Input	Seaside	nil				
Checkbox	□ Checked	nil				
Radiogroup	<ul> <li>Quito</li> <li>Dakar</li> <li>Sydney</li> <li>Bamako</li> </ul>	nil				
Radiogroup (Custom)	<ul> <li>Quito</li> <li>Dakar</li> <li>Sydney</li> <li>Bamako</li> </ul>	nil				
Single Selection	Quito 🛟	nil				
Single Selection (Custom)	Short: Qui ≑	nil				
Single Selection (Optional)	(none) ¢	nil				
Multi Selection	Quito Dakar Sydney Bamako					
Nested Selection	Haskell	nil				
Nested Multi Selection	Functional Haskell Lisp ML Dataflow Hartmann pipelines					
Upload	Choose File no file selected	nil				
Submit						