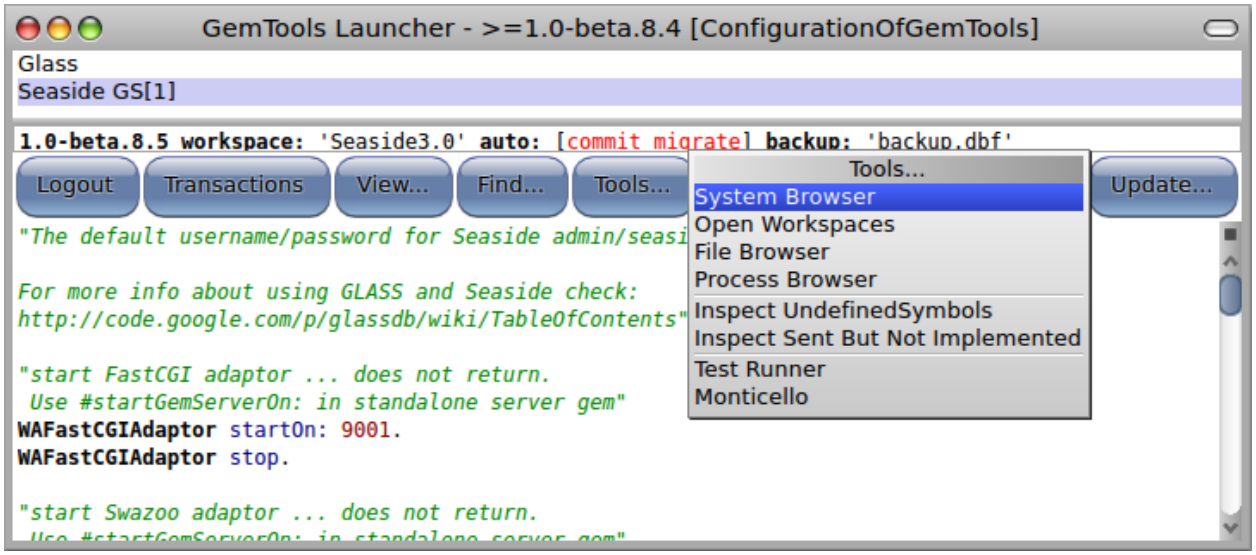
Let's get started using GemStone.

1. First we will quickly create a 'Hello World' application in GemStone.
    a. Start GemStone and the Seaside gems using the instructions from Chapter 16 (if it is not running) and launch GemTools.
    b. In the GemTools launcher, select Seaside, and click the 'Login' button. Enter your name if requested.
    c. Once logged in, click the 'Tools…' button and select 'System Browser'.

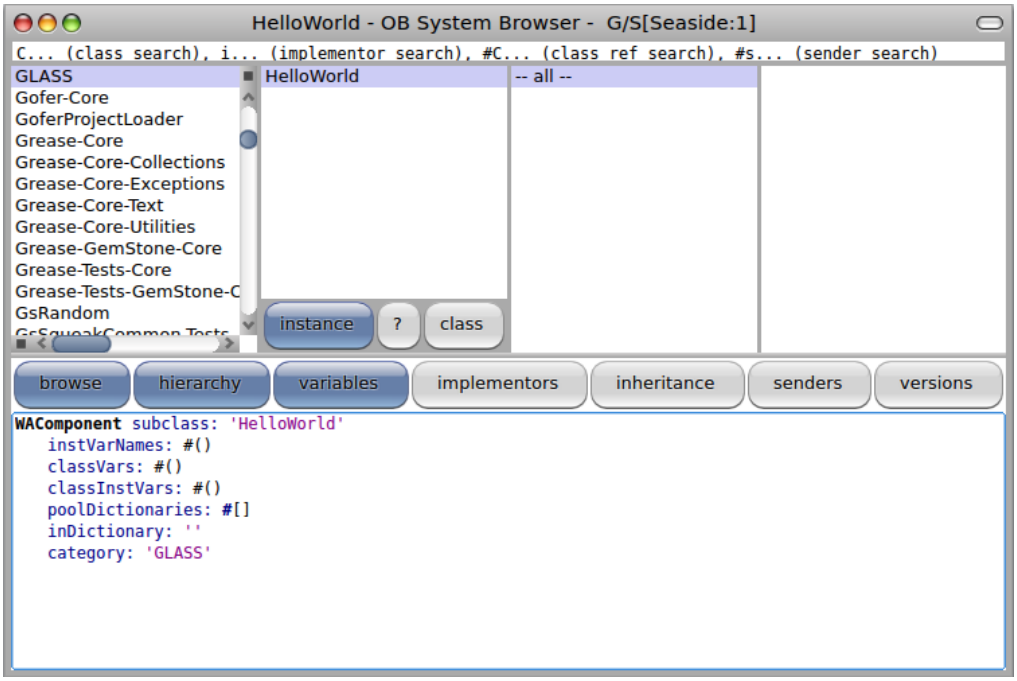



    d. This will open an OB System Browser showing GemStone code. Click in the first column to get a class creation template and enter the following in the text area:

```
WAComponent subclass: 'HelloWorld'
   instVarNames: #()
   classVars: #()
   classInstVars: #()
   poolDictionaries: #[]
   inDictionary: ''
   category: 'GLASS'
```

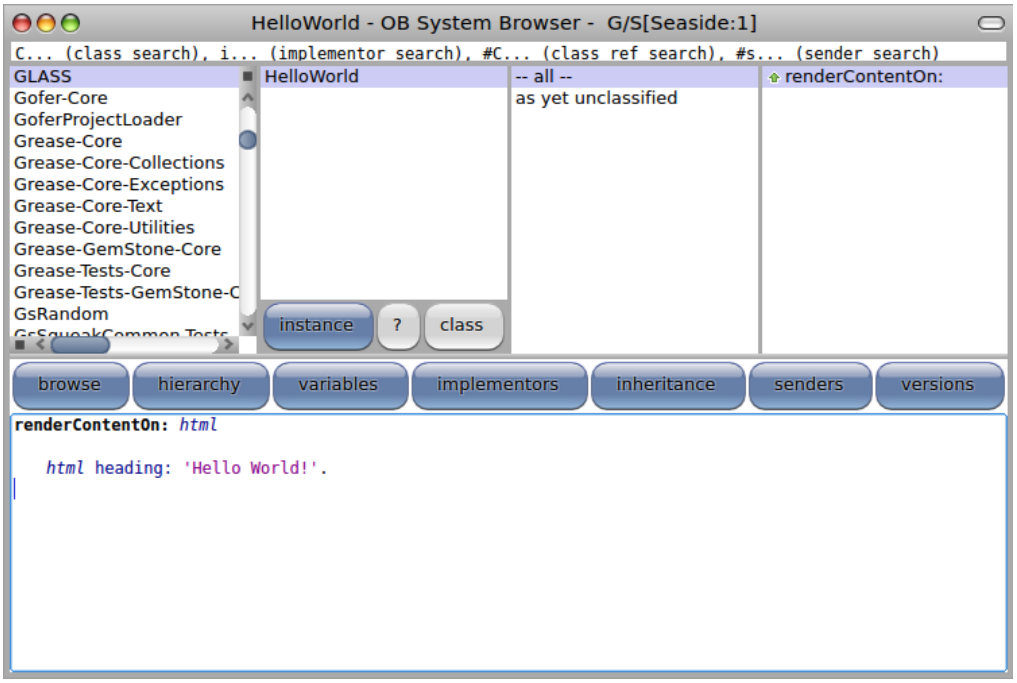
  

e. This should update your System Browser to show the new class.




f. Click in the third column to change the text area from a class definition to a method template. Enter and save the render method.

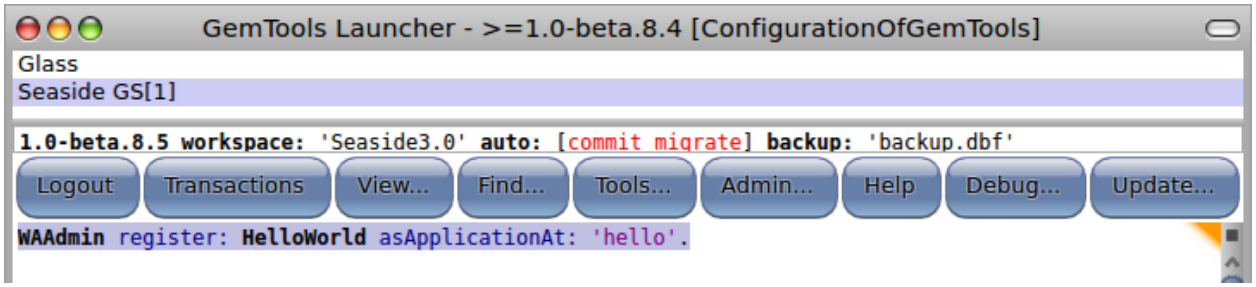
```
renderContentOn: html

   html heading: 'Hello World!'.
```



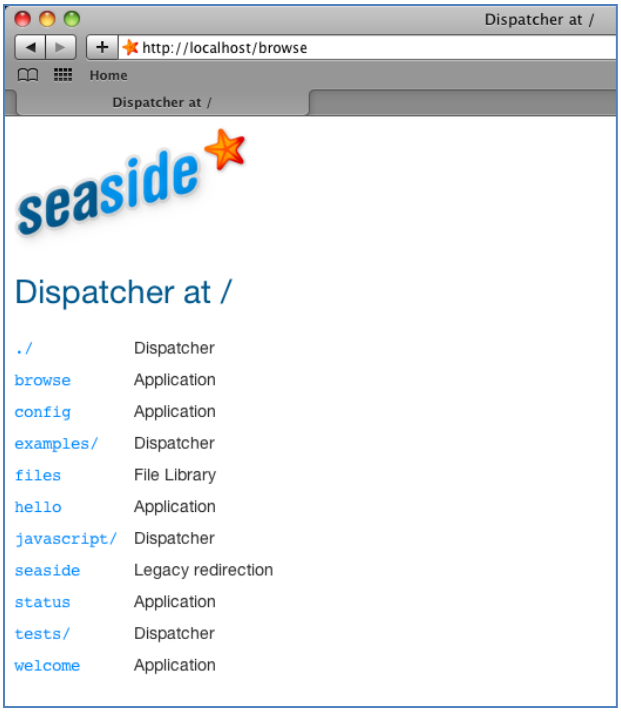

  

g. Register the component as an application. Select the GemTools Launcher, click in the text area below the buttons and enter the expression to register the application. Press <Ctrl>+<D> (for 'do-it') to evaluate the expression.

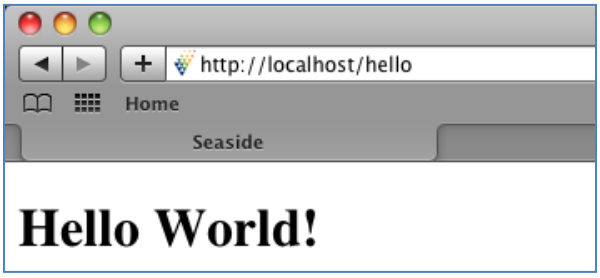
```
WAAdmin register: HelloWorld asApplicationAt: 'hello'.
```




h. Open a web browser on http://glass/browse and note that 'hello' is listed.

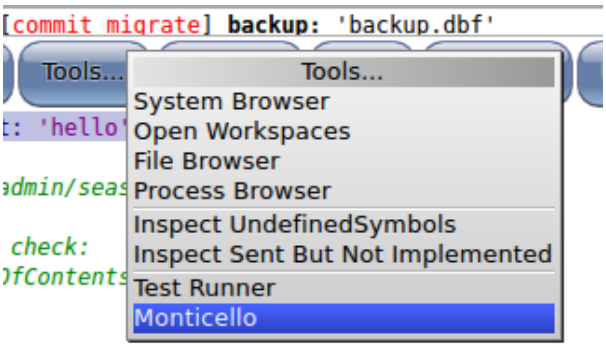



i. Click on the 'hello' link to get the application.

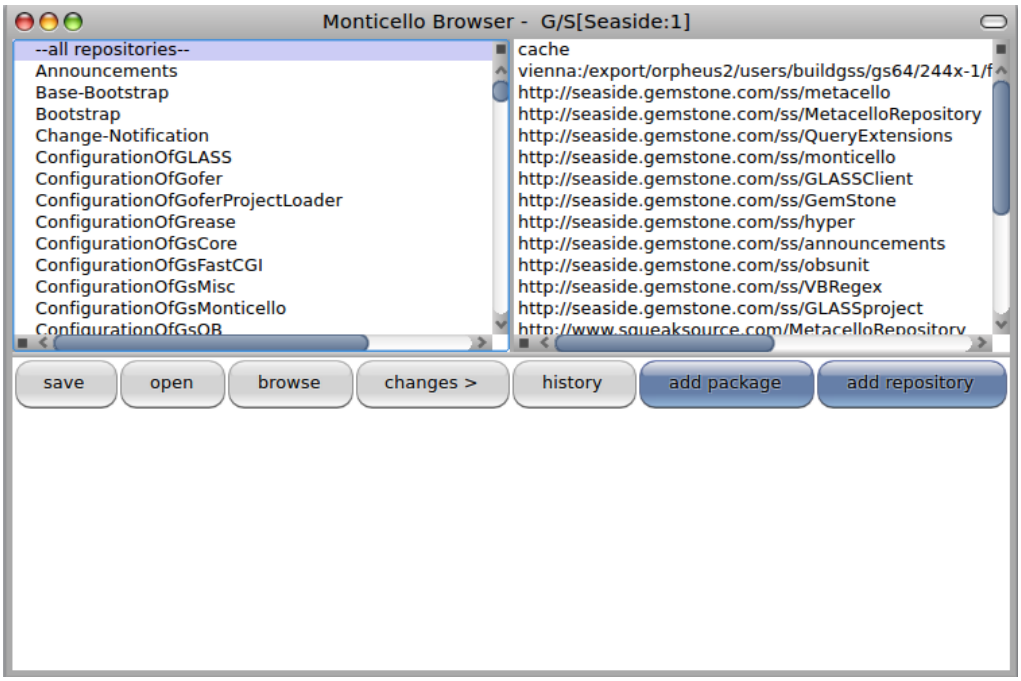


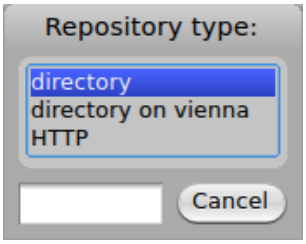
2. Now we will port the Boquitas application from Squeak.
    a. In Chapter 14 we versioned our code to a local disk-based Monticello repository. From the GemTools Launcher, click the 'Tools…' button and then select the 'Monticello' item.




    b. This will open a Monticello Browser viewing GemStone code. Select '--all repositories--' in the left column and then click the 'add repository' button.

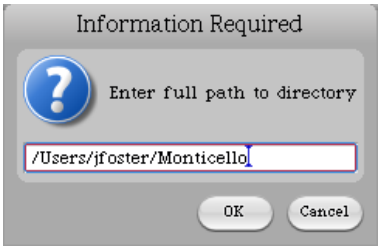



    c. This will open a dialog where you need to identify the repository type. Select the 'directory' option.

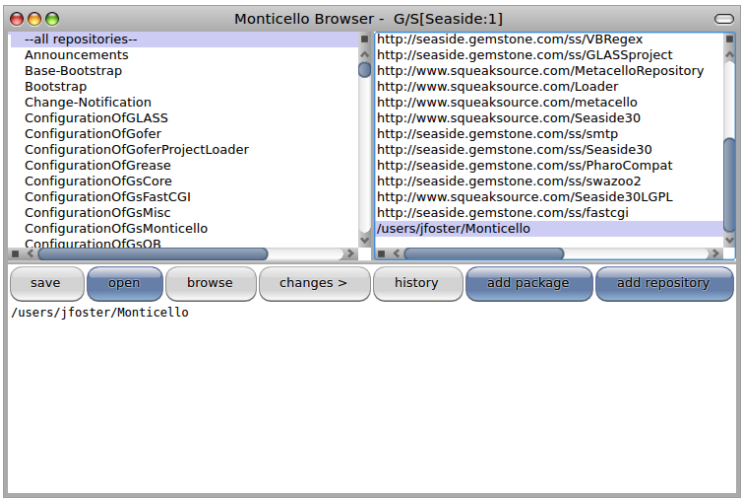


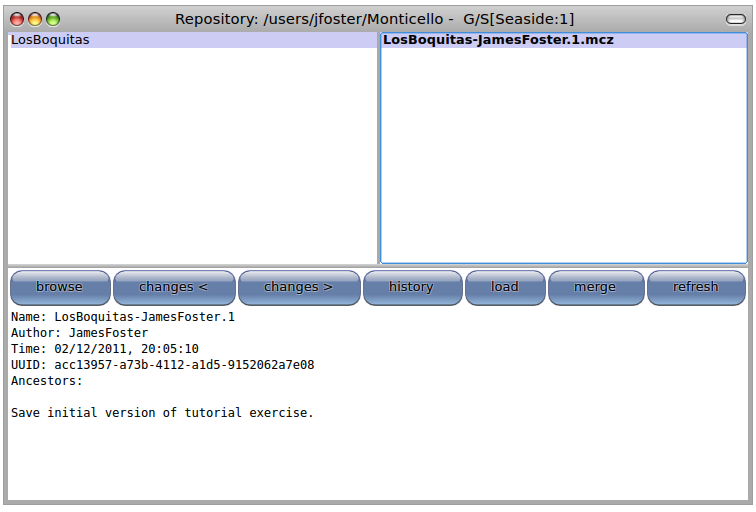
d. When asked to enter the path to the repository, enter the path you used in Chapter 14. You can use a file browser to look for a file named 'LosBoquitas-*.1.mcz'.




e. Once the repository has been added, scroll to the bottom of the repository list (the second column) and select your repository.

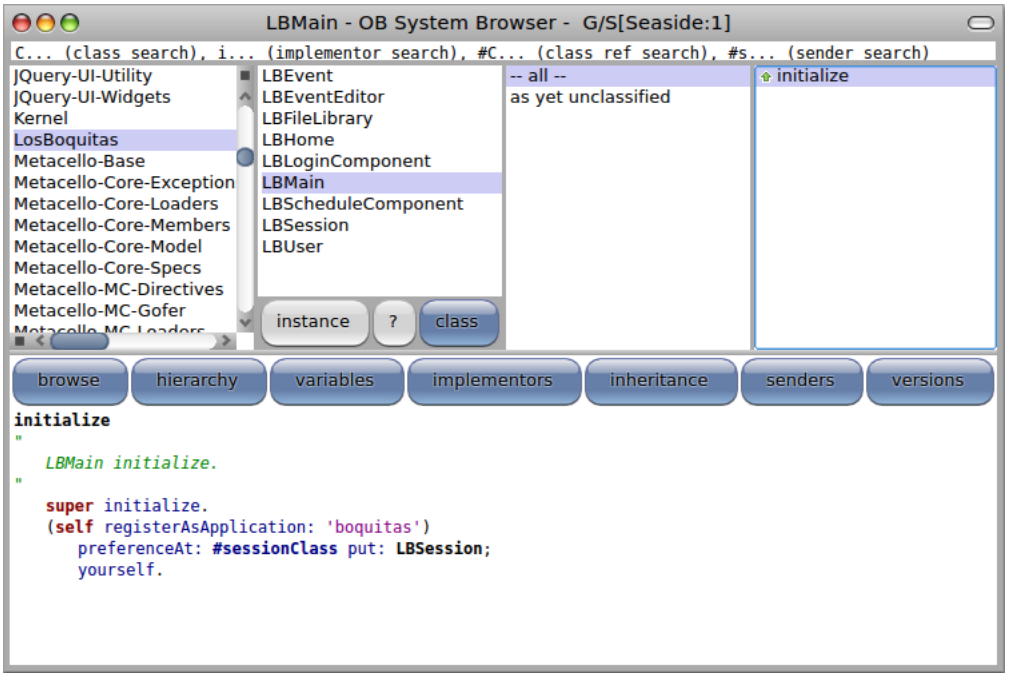



f. With the repository selected, click the 'open' button. This will open a Repository Browser. Select 'LosBoquitas' in the left column (a list of all the packages), and then click the first entry in the right column (a list of the package versions). Click the 'load' button and after the load note that the version name is underlined and no longer bold.



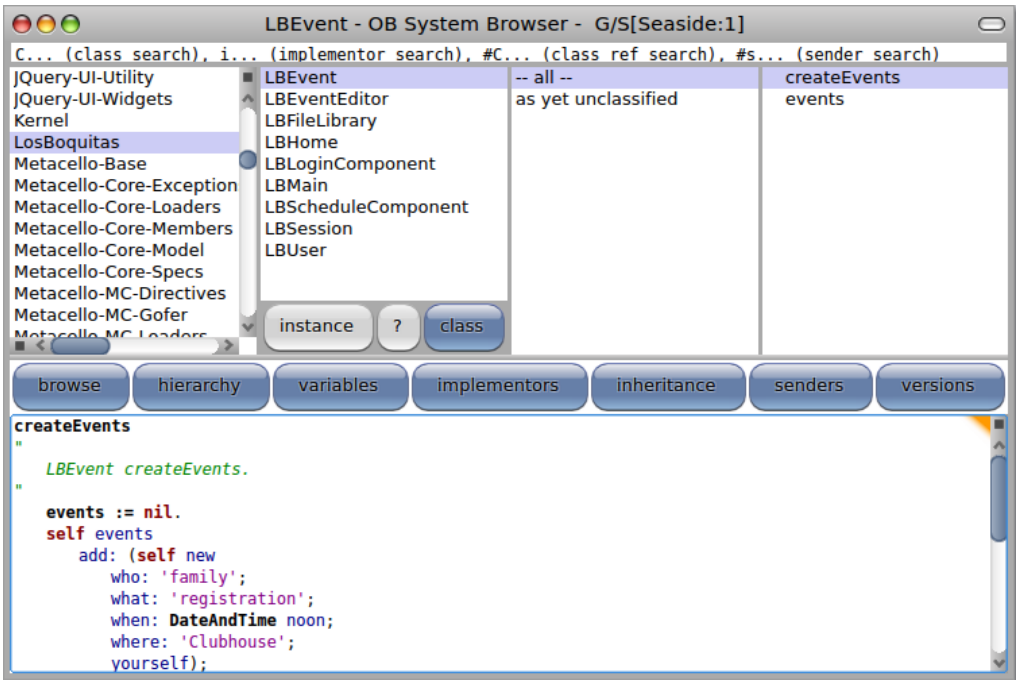
g. Close the Repository Browser and the Monticello Browser. Return to the System Browser and select the 'LosBoquitas' class category in the first column. This shows that the various classes and methods you defined in Squeak have been imported into GemStone. In fact, if you defined #'initialize' as a class-side method on LBMain, then the application will be registered when the class is first loaded.



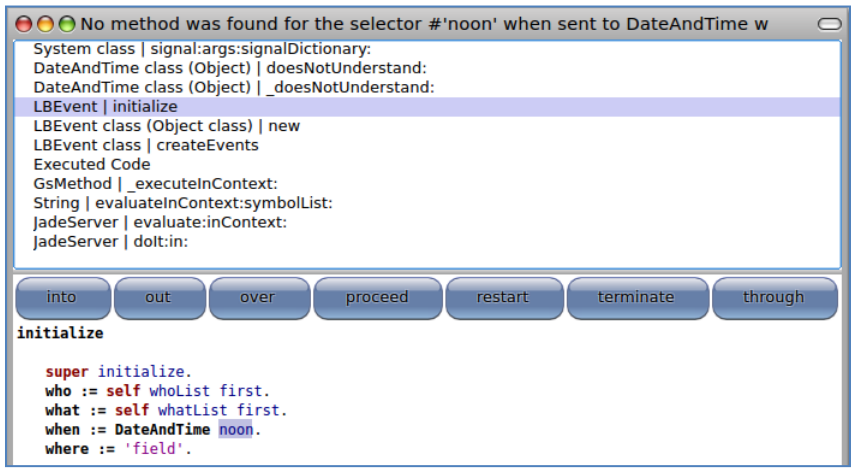

h. Return to your web browser, and navigate to http://localhost/browse and note that 'boquitas' is now available. You should be able to navigate the application and see the features.

3. For the most part GemStone Smalltalk is compatible with Pharo, but sometimes we port an application and find that something doesn't work.
    a. Note that there are no events. Recall that we have a method to create events. In the System Browser, select the LBEvent class, switch to the class side, select the #'createEvents' method, click anywhere in the third line, and type <Ctrl>+<D> (for 'do-it') to evaluate the expression in the comment.

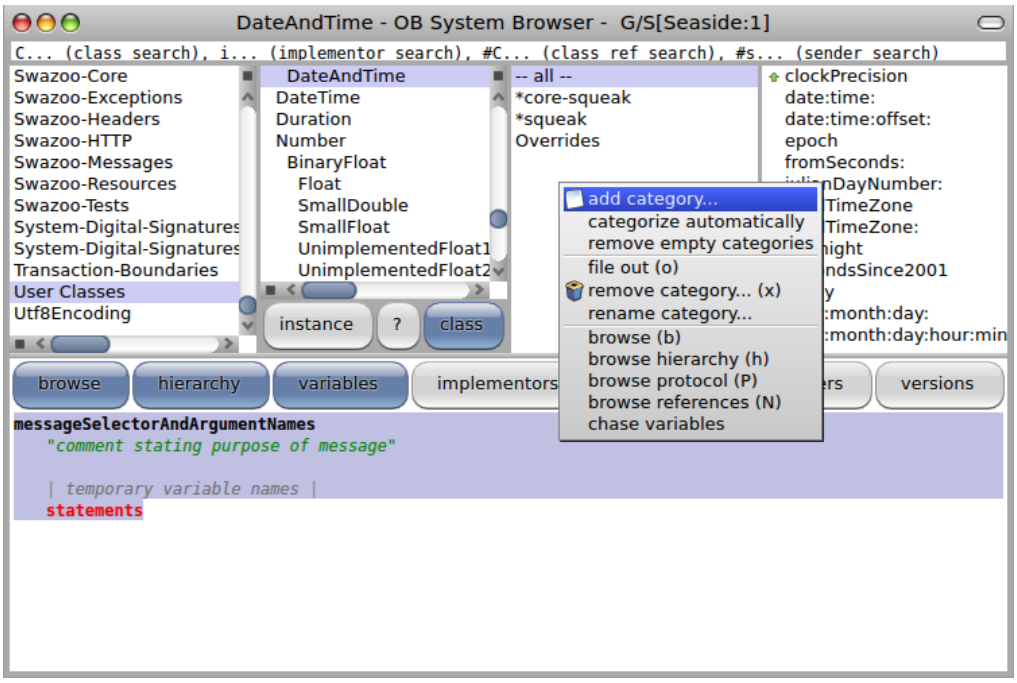



    b. In Pharo this expression generated three events. Unfortunately, in GemStone this generates a walkback reporting that the message #'noon' was not understood by the DateAndTime class. We could, of course, modify our application to not send that message. This would be a relatively simple and safe process. Instead, we will use this opportunity to examine the alternative of adding DateAndTime class>>#'noon' to GemStone. Close the walkback window.

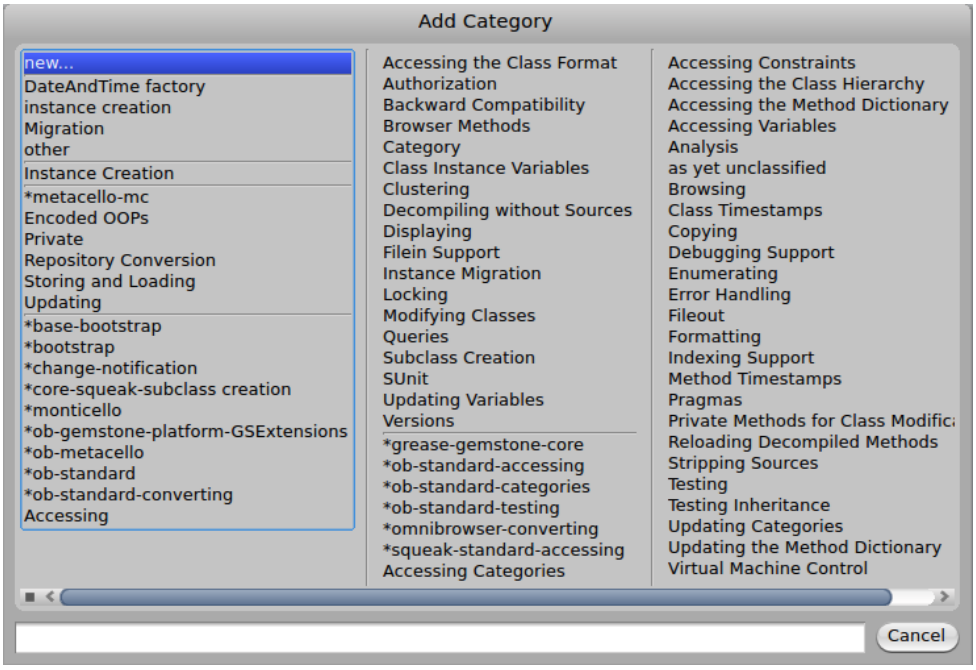


c. In Smalltalk it is customary for all Smalltalk source code to be included in the distribution image and any add-on packages. Furthermore, Smalltalk does not 'close' libraries in the way that, say, Java allows preventing any modification of source code. In the GemTools Launcher, click the 'Find' button, enter 'DateAndTime' in the text entry field, and select the 'DateAndTime' class. In the new System Browser, switch to the class side and right-click on the third column and select the 'add category…' menu.

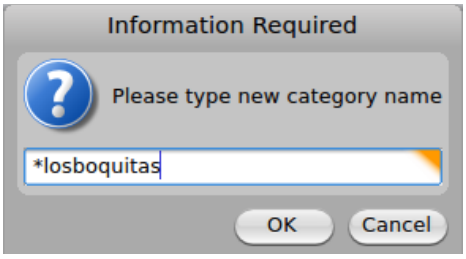



d. In the 'Add Category' dialog, select the 'new…' menu item.



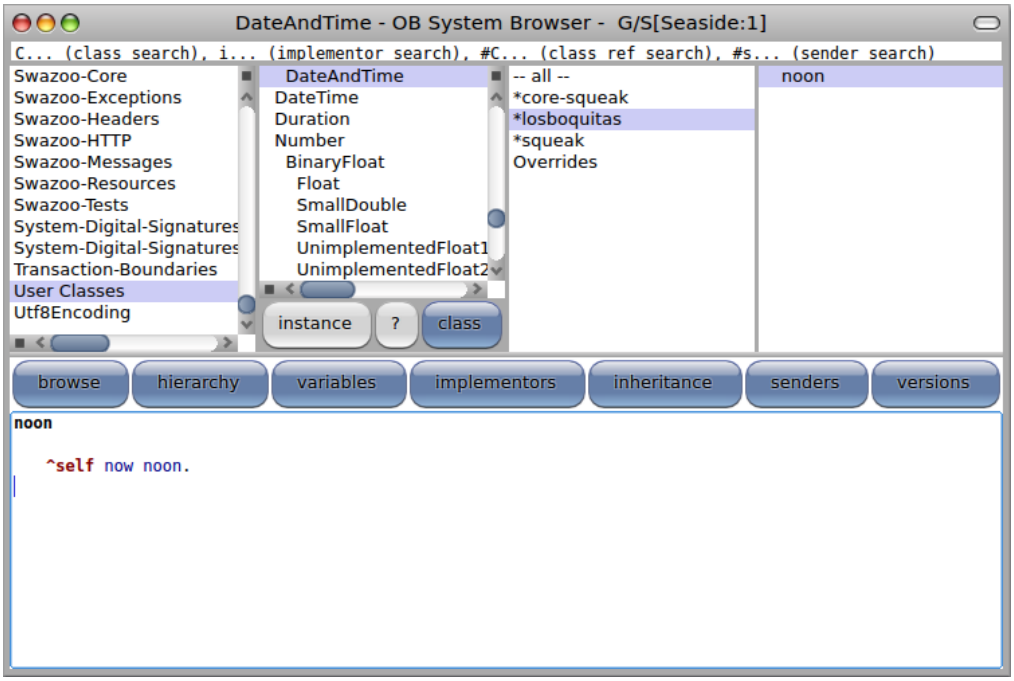

  

e. When asked for a new category name, enter '*losboquitas' and click the 'OK' button. Note that the category name starts with an asterisk, has no spaces, and is all lowercase. The '*' at the beginning means that methods in this category will not be packaged with the class for purposes of version control, but will be included with the package that has a name whose lower-case is 'losboquitas'.




f. Once the category is defined, make sure it is selected and then enter a new method.

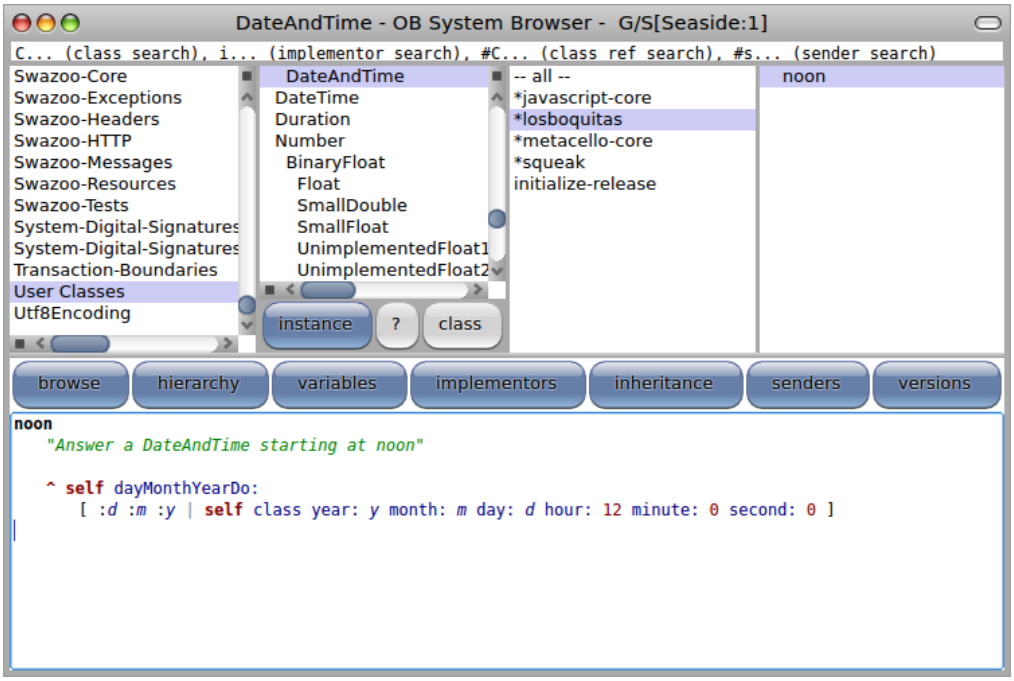
```
noon

   ^self now noon.
```



g. Now, switch from the class side to the instance side (click the 'instance' button), create a '*losboquitas' method category on the instance side, and enter a #'noon' method.

```
noon
   "Answer a DateAndTime starting at noon"

   ^ self dayMonthYearDo:
      [ :d :m :y | self class year: y month: m day: d hour: 12 minute: 0
second: 0 ]
```



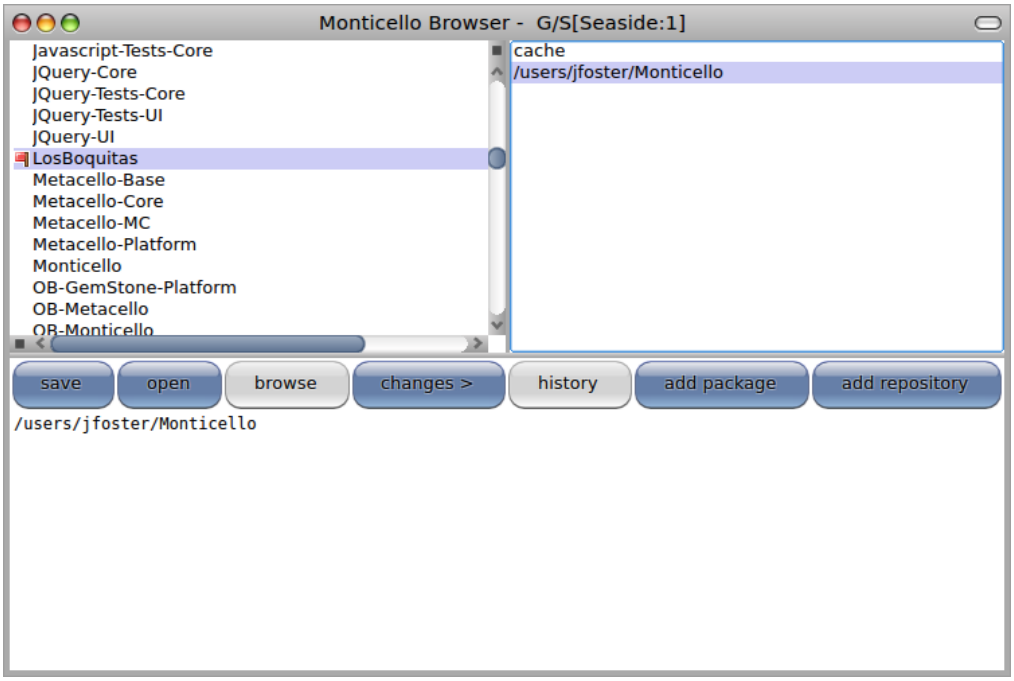

h. Now we will try again to create events. In a workspace, evaluate the following expression. This time we should not get an error.

```
LBEvent createEvents.
```

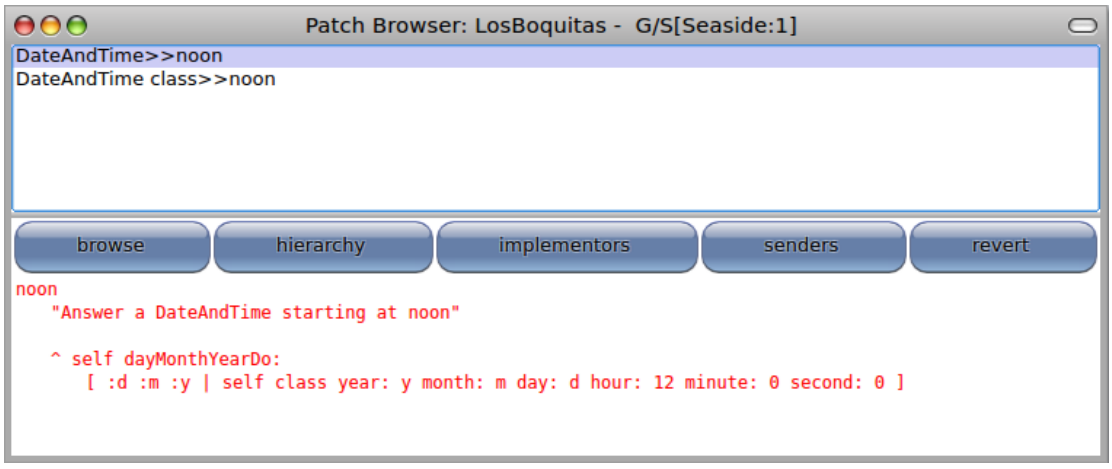
i. Now return to your web browser and observe that the three default events have been created. The approach we have taken of modifying a base class is sometimes called 'Monkey Patching' (see http://en.wikipedia.org/wiki/Monkey_patch) and being able to do this in Smalltalk is considered a powerful feature. Our decision to put the extra methods in our 'LosBoquitas' package is a bit more questionable. A better place for this might be a package like 'Squeak-Kernel-Chronology' so that it can be shared by other packages and so that our LosBoquitas package can be loaded back into Pharo without unnecessary monkey-patching the Pharo methods. The reason we choose not to do so now is because the Squeak extensions for GemStone package is shared (see http://seaside.gemstone.com/ss/monticello) and any changes we made would be lost when we get a new version. (Also, if the main package were fixed, then we wouldn't have this error to use as an example of porting problems!)

j. The changes we made are part of our GemStone object space, but we should save the code outside GemStone for purposes of source control. From the GemTools Launcher, click the 'Tools' button and select the 'Monticello' menu option. This will open a Monticello Browser. Scroll down till you see the 'LosBoquitas' line with a flag next to it. The flag means that the package has been modified.

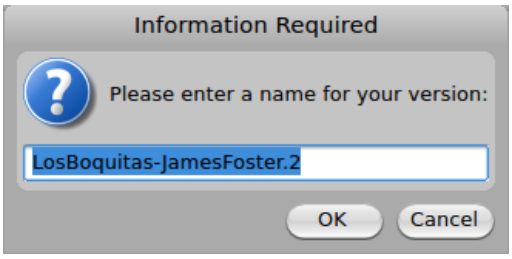



k. Select the local repository from the right-hand list and click the 'changes' button. This will give you a list of the changes in the local environment compared with the version in the repository. In our case, we have added two methods.

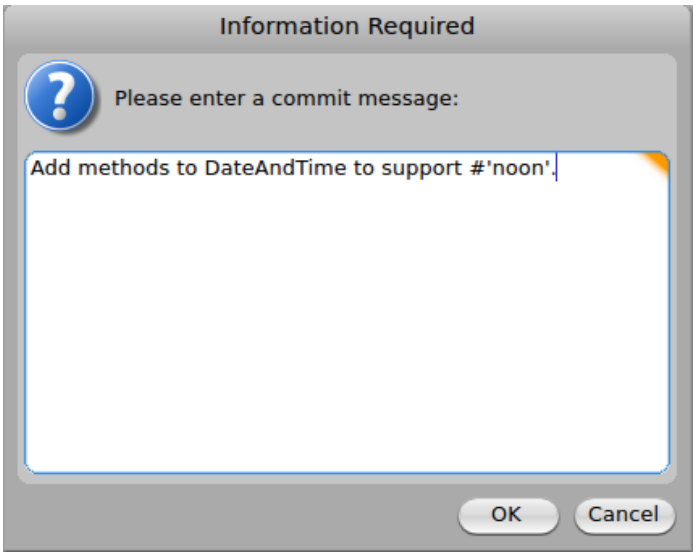


l. Close the Patch Browser and in the Monticello Browser click the ‘save’ button. In the dialog will be a default name for the new version. If you entered your initials when prompted earlier then the default name should be correct. Click the ‘OK’ button.




m. Before committing the modified package to the repository, Monticello will ask for a commit comment. Enter something meaningful.




4. From the GemTools Launcher click the ‘Logout’ button and select ‘Yes’ when asked to confirm. You can quit GemTools without saving. Finally, stop GemStone using the instructions from chapter 16.

With this Chapter we have demonstrated how to create an application in GemStone as well as how to load and modify an application saved in Monticello.