WebObjects

ReportMill 10

Note: WebObjects 5.3:

With WebObjects 5.3, Apple apparently added support for the applicable Java Collections interfaces to the non-standard WebObjects collections classes (NSArray, NSDictionary). Because of this, most of the translation code offered by the RMWebObjects package may no longer be necessary.

Basic API

ReportMill has special support for WebObjects to compensate for WebObjects use of non-standard collection classes (NSArray and NSDictionary instead of List and Map). There are also some built-in WOComponent classes that can be used to return dynamic responses. Additionally, ReportMill for WebObjects will find templates specified by name that have been included in your project.

This support is in the form of several source files available at http://reportmill.com/rm9/RM9WebObjects.zip. Simply add the source files to your project (see "Integrating with WO project" on page 3) and you will be able to use the API covered in this document.

Table of Contents

WebObjects

Basic API

For WebObjects, ReportMill provides an RMDocument subclass called RMWODocument. This subclass is exactly equivalent to RMDocument, except that you may pass a name to the RMDocument constructor (don't include the '.rpt'!). ReportMill uses the WebObjects resource manager to find the template which you should add to your project's resources. You can also pass WebObjects non-standard collection classes NSArray and NSDictionary in place of the Lists and Maps that ReportMill expects.

There are also three WOComponent subclasses to return content to the browser directly: RMPDFPage, RMFlashPage and RMCSVPage. You simply set the "document" attribute of these components to a generated report.

```
import com.reportmill.foundation.*;
import com.reportmill.webobjects.*;

public void generateReport()
{
   RMDocument template = new RMWODocument("MyTemplate");
   RMDocument report = template.generateReport(myObjects, myUserInfo, true);
   report.writePDF("/MyDocRoot/MyReport.pdf");
}
```

Save Dataset

Normally generating a dataset is done with a call to new RMXMLWriter.writeObject() as documented in the Basic API document on the Support page. However, with WebObjects, instead make this a call to RMWebObjects.writeObject() (everything else works normally).

WebObjects (Continued)

Special WOComponent Subclasses

There are also three WOComponent subclasses to return content to the browser directly: RMPDFPage, RMFlashPage and RMCSVPage. You simply set the "document" attribute of these components to a generated report.

```
import com.reportmill.base.*;

public WOComponent generateReport()
{
	RMDocument template = new RMWODocument("MyTemplate");
	RMDocument report = template.generateReport(myObjects, myUserInfo, true);
	RMPDFPage page = (RMPDFPage)pageWithName("RMPDFPage");
	page.document = report;
	return page;
}
```

Note - you should really load the template into a static variable and re-use it rather than re-loading it for every report.

WebObjects (Continued)

Integration With a WebObjects 5 Project

Project Builder can be finicky about working with third party jars (there are many notes about this in the OmniGroup WebObjects mail archive). We've found reliable behavior when using the following steps:

ReportMill.jar:

1. Simply copy **ReportMill9.jar** to /Library/WebObjects/Extensions

RMWebObjects.tgz:

1. Add the following source files to your project:
    - RMWebObjects.java
    - RMWODocument.java
    - RMPDFPage.java

2. Optionally add package statements to your files, depending the placement in your project.

The Code:

1. Now simply add the following code to any action method that returns a WOComponent:

```
import com.reportmill.foundation.*;

RMDocument template = new RMWODocument("MyTemplate");
RMDocument report = template.generateReport(myObjects, myUserInfo);
RMPDFPage nextPage = (RMPDFPage)pageWithName("RMPDFPage");
nextPage.document = report;
return nextPage;
```

WebObjects (Continued)

Integration With a Legacy WebObjects 4.5 Project in ProjectBuilder WO

The WebObjects support previously mentioned is compiled for WO5. For WO4.5, download the source code (which is in the form of a ProjectBuildWO subproject) and add it to your project. (http://www.reportmill.com/examples/ReportMill.subproj.tgz). Then you just need to add the ReportMill.jar path to the OTHER_CLASSPATH variable in the Makefile.preamble of your project and the ReportMill.subproj and to the NSJavaUserPath in the CustomInfo.plist file found in the Supporting Files suitcase.

If your template has images in it, you may have to add the following line to Application.java to workaround a bug in WO45 where the request/response loop gets interrupted when awt initializes:

```
static { java.awt.Toolkit.getDefaultToolkit(); }
```

WebObjects Version 4.5 Fonts

The first time you generate a report, ReportMill builds a font name cache for fonts found on your system. Inspecting many fonts can sometimes exceed the default memory allowed the Java VM under WebObjects 4.5. To temporarily increase this to allow the cache to get built, run your web app with the flag: -NSJavaMaxHeapSize 100000000. Or to set it permanently:

```
prompt> defaults write MyApp NSJavaMaxHeapSize 100000000
```

and to remove:

```
prompt> defaults remove MyApp NSJavaMaxHeapSize
```

WebObjects - Using the Legacy ObjectiveC Framework

When using the older ObjectiveC version of ReportMill.framework with WebObjects the WOComponent code is slightly different, due to the fact that the ReportMill components are written in WebScript (and available in ReportMill.framework/Resources/ReportMillPage.[wo,.wos]).

Simply add the code below to the action for any submit button or WOHyperLink.

```
import com.reportmill.webobjects.*;

public WOComponent generateReport()
{
  RMWOPage page = (RMWOPage)pageWithName("RMPDFPage");
  page.takeValueForKey("MyTemplate", "template"); // Required
  page.takeValueForKey(anArray, "objects"); // optional
  page.takeValueForKey(userInfoDictionary, "userInfo"); // optional
  return page;
}
```