SPRINGER BRIEFS IN COMPUTER SCIENCE

Deepak Vohra

Rhodes Framework for Android™ Platform and BlackBerry® Smartphones

Springer

SpringerBriefs in Computer Science

Deepak Vohra

Rhodes Framework for Android$^{\text{TM}}$ Platform and BlackBerry® Smartphones

Springer

Deepak Vohra
e-mail: dvohra09@yahoo.com

ISSN 2191-5768 e-ISSN 2191-5776
ISBN 978-1-4614-3578-5 e-ISBN 978-1-4614-3579-2
DOI 10.1007/978-1-4614-3579-2
Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2012934252

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Rhodes is an open source, Ruby-based, lightweight, MVC (model view controller) framework, optimized for mobile devices, which have memory limitations. The Rhodes framework offers several advantages over other mobile frameworks. Some of the unique features of the Rhodes frameworks are as follows.

- The only smartphone framework to offer support for the Model View Controller pattern.
- The only smartphone framework to offer support for the Object-Relational manager.
- The only smartphone framework to offer offline, disconnected access to data with the RhoSynch server.
- The only smartphone framework to support all mobile devices including Android™ platform smartphone, BlackBerry® smartphone, iPhone®, Symbian Platform, and Windows Mobile® operating system.
- Provides Ruby implementations for all smartphone device operating systems.
- Provides a web-based Integrated Development Environment for developing mobile applications for all smartphone platforms with the RhoHub development service.

Google's Android™ platform and RIM's BlackBerry® smartphone are the top two most commonly used smartphone platforms. Android™ platform has more than 40% of the smartphone market share. In Chap. 1 we discuss the Android™ platform. In Chap. 2 we discuss the Rhodes framework with the BlackBerry® smartphone. We develop the same Rhodes applications for Android™ platform and BlackBerry® smartphone; one application for a catalog and another for an RSS feed.

Contents

Chapter 1
Rhodes on Android^{TM} Platform

Smartphones have proliferated in recent years creating a need for smartphone apps. Android^{TM} is the most commonly used smartphone platform. Ruby is an open source, dynamic, interpreted programming language. Rhodes is the only framework for mobile devices that supports MVC architecture, and provides an Object Relational Manager. In this chapter we shall, first, introduce using Rhodes on Android, and subsequently develop a Rhodes application to get RSS feed for a magazine on Android. To parse XML Rhodes includes the RhoXML parser and support for the ReXML parser may be added.

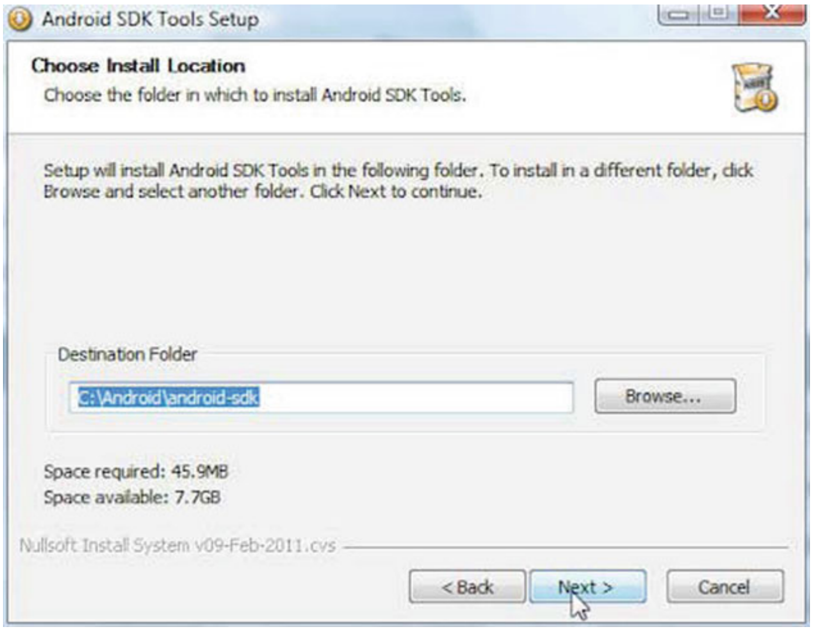
1.1 Overview

This chapter which discusses the procedure to create a Rhodes applications for Android has the following sections.

- Installing the Android SDK
- Installing Ruby
- Installing Rhodes
- Creating a Rhodes Application
- Creating a Rhodes Model for a Catalog
- Creating a Rhodes Model to get RSS Feed

1.2 Installing the Android SDK

Download the Android SDK `installer_r12-windows.exe` from http://developer.android.com/sdk/index.html. Double-click on the `.exe` file. The Android SDK Tools Setup Wizard gets started. Click on **Next**. Android SDK requires Java SE Development Kit (JDK). Download and install the JDK, if not already



Android SDK Tools Setup

Choose Install Location

Choose the folder in which to install Android SDK Tools.

Setup will install Android SDK Tools in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

C:\Android\android-sdk

Space required: 45.9MB
Space available: 7.7GB

Nullsoft Install System v09-Feb-2011.cvs

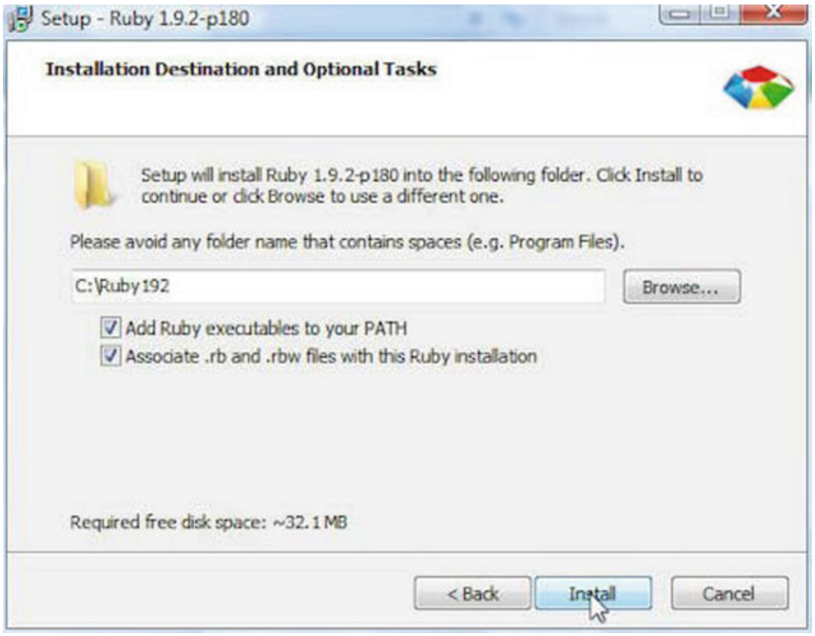
Fig. 1.1 Specifying the install location for Android SDK

installed, from http://www.oracle.com/technetwork/java/javase/downloads/index.html. Install the JDK in a directory without spaces in the directory path. The Android SDK Tools Setup wizard detects if the JDK is installed or not and displays a message if the JDK is required to be installed. Click on **Next**. In **Choose Install Location** specify the **Destination Folder**. Install Android SDK in a directory without spaces in the directory path; for example not in the `C:/Program Files/` sub-directory. Specify Destination Folder as `C\Android\android-sdk` as shown in Fig. 1.1. Click on **Next**. Click on **Install**.

Download Android NDK zip file from http://developer.android.com/sdk/ndk/index.html and extract it to a directory without spaces in the directory path. Create environment variables `ANDROID_HOME` for the Android SDK and `ANDROID_NDK_ROOT` for the Android NDK. Add `ANDROID_HOME/tools`, `JDK_HOME` and `JDK_HOME/bin` to the `PATH` environment variable. In a later section we shall configure a Rhodes application to be used with the Android emulator.

1.3 Installing Ruby

Download the `rubyinstaller-1.9.2-p180.exe` application from http://rubyinstaller.org/. Double-click on the `.exe` file to install Ruby. Install Ruby in a directory without spaces in the directory path as shown in Fig. 1.2. Select the checkbox "Add Ruby executables to your PATH". Click on **Install**.


Setup - Ruby 1.9.2-p180

Installation Destination and Optional Tasks

Setup will install Ruby 1.9.2-p180 into the following folder. Click Install to continue or click Browse to use a different one.

Please avoid any folder name that contains spaces (e.g. Program Files).

C:\Ruby192

☑ Add Ruby executables to your PATH
☑ Associate .rb and .rbw files with this Ruby installation

Required free disk space: ~32.1 MB

Fig. 1.2 Installing Ruby Installer

Next, install the Ruby Installer Development Kit (DevKit), which makes it easy to build native Ruby extensions. Download the `DevKit-tdm-32-4.5.1-20101214-1400-sfx.exe` application from http://rubyinstaller.org/downloads/. Double-click on the `.exe` file and install the self-extracting executables in a directory without spaces, such as `C:/Ruby192/DevKit`. Cd (change directory) to the `DevKit` directory and run the following commands:

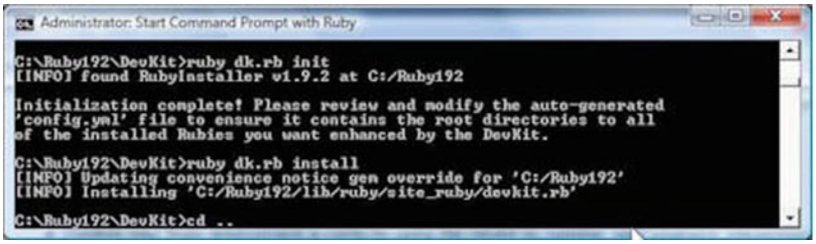
```
rubydk.rbinit
rubydk.rb install
```

The output from running the commands is shown in Fig. 1.3.

The `init` command creates a `config.xml` file, which lists the RubyInstaller installed Rubies. Install RubyGems, a Ruby packaging system. Download the RubyGems zip file and extract the zip file to a directory, Cd to the directory and run the following command:

```
C:Ruby192\rubygems-1.6.2>ruby setup.rb
```

We also need to install `gnuwin32`, which provides win32 ports of GNU tools, GNU being a UNIX-like operating system. Download the `GetGnuWin32-0.6.3.exe` application from http://sourceforge.net/projects/getgnuwin32/files/



```
Administrator: Start Command Prompt with Ruby
C:\Ruby192\DevKit>ruby dk.rb init
[INFO] found RubyInstaller v1.9.2 at C:/Ruby192

Initialization complete! Please review and modify the auto-generated
'config.yml' file to ensure it contains the root directories to all
of the installed Rubies you want enhanced by the DevKit.

C:\Ruby192\DevKit>ruby dk.rb install
[INFO] Updating convenience notice gem override for 'C:/Ruby192'
[INFO] Installing 'C:/Ruby192/lib/ruby/site_ruby/devkit.rb'

C:\Ruby192\DevKit>cd ..
```

Fig. 1.3 Installing Ruby Installer DevKit

getgnuwin32/0.6.30/GetGnuWin32-0.6.3.exe/download and double-click on the `.exe` file. Install in a folder (`C:/Ruby192` for example). Cd to the installed folder (`C:/Ruby192/GetGnuWin32`) and run the `download` command.

```
C:\Ruby192\GetGnuWin32>download
```

Subsequently, run the `install` command:

```
C:\Ruby192\GetGnuWin32>install C:/gnuwin32
```

Add `C:/gnuwin32/bin` to the `PATH` environment variable. Next, install Rake, a Ruby build program, with the following command:

```
C:\Ruby192>gem install rake
```

1.4 Installing Rhodes

Rhodes is a Ruby gem that is installed just like any other Ruby gem. To install Rhodes run the following command:

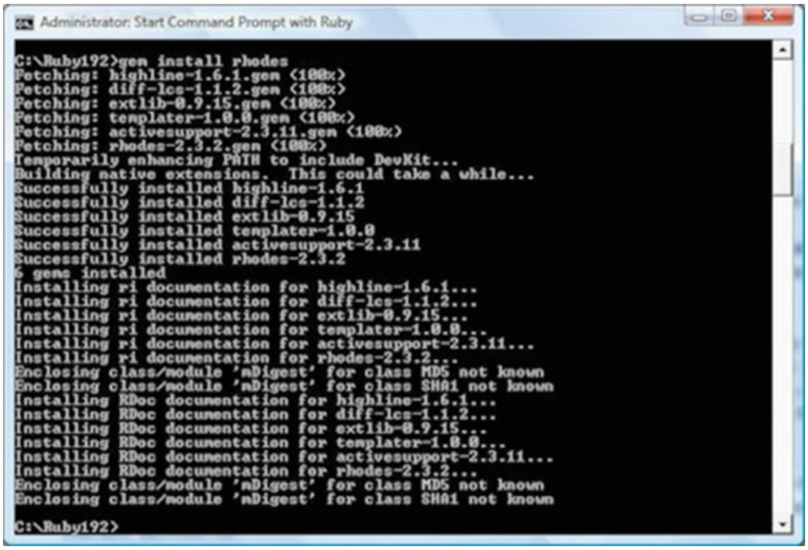
```
C:\Ruby192>gem installrhodes
```

Rhodes 2.3.2 and related gems get installed as shown in Fig. 1.4.

In the next section we shall create a Rhodes application.

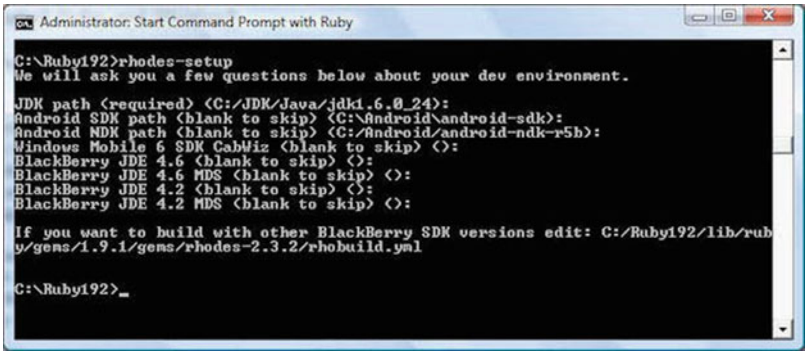
1.5 Creating a Rhodes Application

Rhodes provides an application generator to generate an application. The Rhodes application generator is called ‘rhodes’ and is run with the following command format:



```
C:\Ruby192>gem install rhodes
Fetching: highline-1.6.1.gem (100%)
Fetching: diff-lcs-1.1.2.gem (100%)
Fetching: extlib-0.9.15.gem (100%)
Fetching: templater-1.0.0.gem (100%)
Fetching: activesupport-2.3.11.gem (100%)
Fetching: rhodes-2.3.2.gem (100%)
Temporarily enhancing PATH to include DevKit...
Building native extensions.  This could take a while...
Successfully installed highline-1.6.1
Successfully installed diff-lcs-1.1.2
Successfully installed extlib-0.9.15
Successfully installed templater-1.0.0
Successfully installed activesupport-2.3.11
Successfully installed rhodes-2.3.2
6 gems installed
Installing ri documentation for highline-1.6.1...
Installing ri documentation for diff-lcs-1.1.2...
Installing ri documentation for extlib-0.9.15...
Installing ri documentation for templater-1.0.0...
Installing ri documentation for activesupport-2.3.11...
Installing ri documentation for rhodes-2.3.2...
Enclosing class/module 'mDigest' for class MD5 not known
Enclosing class/module 'mDigest' for class SHA1 not known
Installing RDoc documentation for highline-1.6.1...
Installing RDoc documentation for diff-lcs-1.1.2...
Installing RDoc documentation for extlib-0.9.15...
Installing RDoc documentation for templater-1.0.0...
Installing RDoc documentation for activesupport-2.3.11...
Installing RDoc documentation for rhodes-2.3.2...
Enclosing class/module 'mDigest' for class MD5 not known
Enclosing class/module 'mDigest' for class SHA1 not known

C:\Ruby192>
```

Fig. 1.4 Installing Rhodes



```
C:\Ruby192>rhodes-setup
We will ask you a few questions below about your dev environment.

JDK path (required) (C:/JDK/Java/jdk1.6.0_24):
Android SDK path (blank to skip) (C:\Android\android-sdk):
Android NDK path (blank to skip) (C:/Android/android-ndk-r5b):
Windows Mobile 6 SDK CabWiz (blank to skip) ():
BlackBerry JDE 4.6 (blank to skip) ():
BlackBerry JDE 4.6 MDS (blank to skip) ():
BlackBerry JDE 4.2 (blank to skip) ():
BlackBerry JDE 4.2 MDS (blank to skip) ():

If you want to build with other BlackBerry SDK versions edit: C:/Ruby192/lib/rub
y/gems/1.9.1/gems/rhodes-2.3.2/rhobuild.yml

C:\Ruby192>_
```

Fig. 1.5 Running the Rhodes-setup batch script

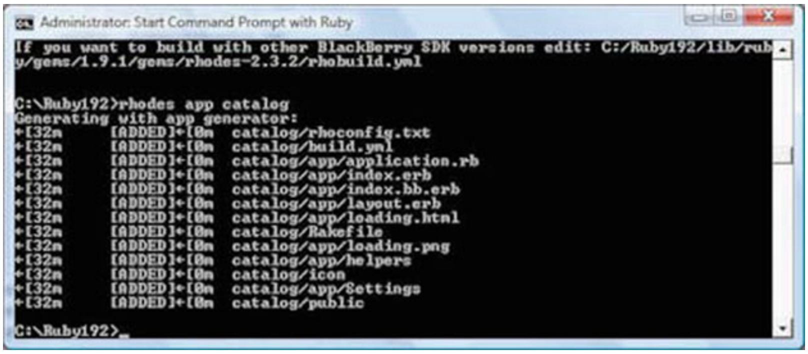
```
rhodes app <application_name>
```

Before we may run the `rhodes` command we need to setup Rhodes using the `rhodes-setup` command. Select Enter for each of the questions. The JDK path should not include any spaces in the directory path. The Android SDK path and NDK path should also not include spaces in the directory path. Cd to the `C:\Ruby192` folder and run the `rhodes-setup` command as shown in Fig. 1.5.

Modify the `C:\Ruby192\lib\ruby\gems\1.9.1\gems\rhodes-2.3.2\rhobuild.yml` configuration file, listed below, to include the Android paths. The JDK and Android paths are shown in bold.

```
env:
app: C:/rhodes-app
paths:
java: C:/JDK/Java/jdk1.6.0_24/bin
android: C:/Android/android-sdk
android-ndk: C:/Android/android-ndk-r5b
cabwiz:
    4.6:
jde:
mds:
sim: 9000
    4.2:
jde:
mds:
sim: 8100
build:
bbpath: platform/bb
wmpath: platform/wm
androidpath: platform/android
iphonepath: platform/iphone
symbianpath: platform/symbian
bb:
bbsignpwd: somepasswordhere
android:
excludedirs:
all:
  - "**/.*.swo"
  - "**/.*.swn"
  - "**/.DS_Store"
bb:
  - public/js/iui
  - public/js/jquery*
  - public/jqtouch*
  - public/js/prototype*
  - public/css/iphone*
  - public/iwebkit
  - public/themes
  - "**/jquery*.js"
    - "**/*.db"
```

The android parameter specifies the directory in which the Android SDK is installed. The `android-ndk` parameter specifies the directory in which the Android NDK is installed. Next, run the Rhodes application generator to create an application called `catalog` with the command:



```
If you want to build with other BlackBerry SDK versions edit: C:/Ruby192/lib/rub
y/gems/1.9.1/gems/rhodes-2.3.2/rhobuild.yml

C:\Ruby192>rhodes app catalog
Generating with app generator:
←[32m      [ADDED]←[0m  catalog/rhoconfig.txt
←[32m      [ADDED]←[0m  catalog/build.yml
←[32m      [ADDED]←[0m  catalog/app/application.rb
←[32m      [ADDED]←[0m  catalog/app/index.erb
←[32m      [ADDED]←[0m  catalog/app/index.bb.erb
←[32m      [ADDED]←[0m  catalog/app/layout.erb
←[32m      [ADDED]←[0m  catalog/app/loading.html
←[32m      [ADDED]←[0m  catalog/Rakefile
←[32m      [ADDED]←[0m  catalog/app/loading.png
←[32m      [ADDED]←[0m  catalog/app/helpers
←[32m      [ADDED]←[0m  catalog/icon
←[32m      [ADDED]←[0m  catalog/app/Settings
←[32m      [ADDED]←[0m  catalog/public

C:\Ruby192>_
```

Fig. 1.6 Creating a Rhodes application

```
C:\Ruby192>rhodes app catalog
```

The application files get generated in the `catalog` (application name) directory as shown in Fig. 1.6.

The `catalog/build.yml` file lists the SDK install directory and the SDK version.

```
sdk: "C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2"
sdkversion: 2.3.2
name: catalog
version: 1.0
vendor: rhomobile
build: debug
bbver: 4.6
wmsdk: "Windows Mobile 6 Professional SDK (ARMV4I)"
applog: rholog.txt
```

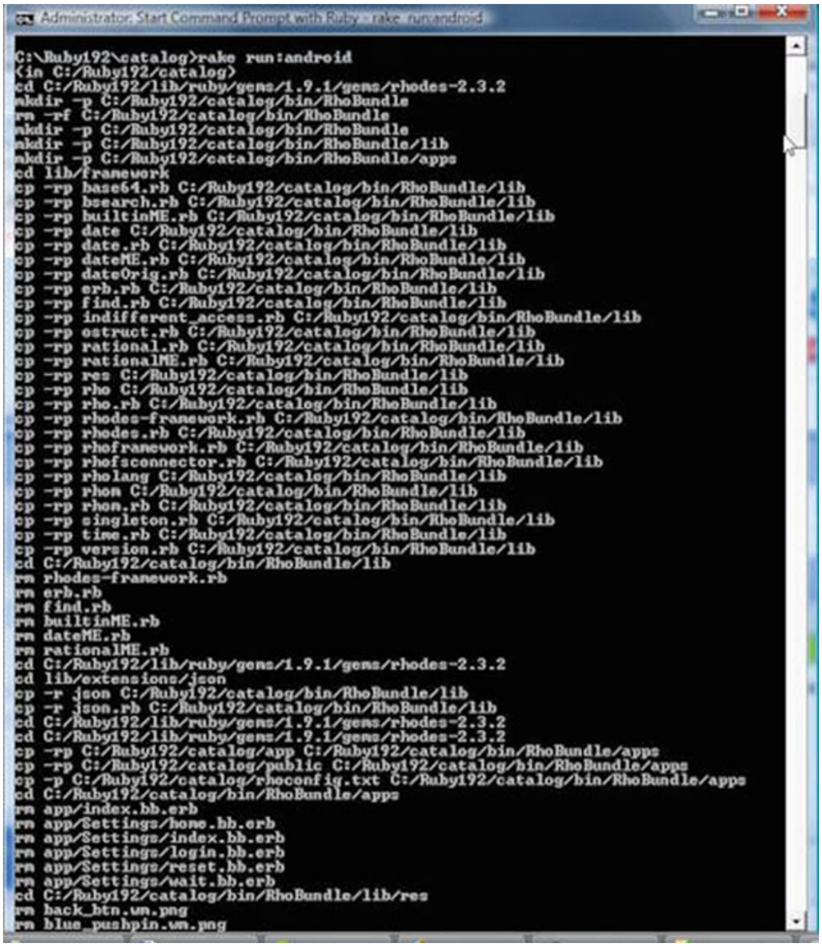
Before developing the application further test the Android emulator. Cd to the `catalog` directory and run the following command:

```
C:\Ruby192\catalog>rake run:android
```

The Android emulator gets started as shown in Fig. 1.7.

Select the default settings for an Android Virtual device, and click Enter when prompted with a question “Do you want to create a custom hardware profile?” as shown in Fig. 1.8. Only the first time the `rake` command is run the user is prompted.

The Rhodes application gets built to an Android application consisting of an Activity (`RhodesActivity`) and gets uploaded to the Android emulator. After loading is complete the Rhodes application gets started as shown in Fig. 1.9.


Administrator: Start Command Prompt with Ruby - rake run:android

```
C:\Ruby192\catalog>rake run:android
(in C:/Ruby192/catalog)
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
mkdir -p C:/Ruby192/catalog/bin/RhoBundle
rm -rf C:/Ruby192/catalog/bin/RhoBundle
mkdir -p C:/Ruby192/catalog/bin/RhoBundle
mkdir -p C:/Ruby192/catalog/bin/RhoBundle/lib
mkdir -p C:/Ruby192/catalog/bin/RhoBundle/apps
cd lib/framework
cp -rp base64.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp bsearch.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp builtinME.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp date C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp date.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp dateME.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp dateOrig.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp erb.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp find.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp indifferent_access.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp ostruct.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rational.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rationalME.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp res C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rho C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rho.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rhodes-framework.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rhodes.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rhoframework.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rhofsconnector.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rholang C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rhom C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp rhom.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp singleton.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp time.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cp -rp version.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cd C:/Ruby192/catalog/bin/RhoBundle/lib
rm rhodes-framework.rb
rm erb.rb
rm find.rb
rm builtinME.rb
rm dateME.rb
rm rationalME.rb
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
cd lib/extensions/json
cp -r json C:/Ruby192/catalog/bin/RhoBundle/lib
cp -r json.rb C:/Ruby192/catalog/bin/RhoBundle/lib
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
cp -rp C:/Ruby192/catalog/app C:/Ruby192/catalog/bin/RhoBundle/apps
cp -rp C:/Ruby192/catalog/public C:/Ruby192/catalog/bin/RhoBundle/apps
cp -p C:/Ruby192/catalog/rhoconfig.txt C:/Ruby192/catalog/bin/RhoBundle/apps
cd C:/Ruby192/catalog/bin/RhoBundle/apps
rm app/index.bb.erb
rm app/Settings/home.bb.erb
rm app/Settings/index.bb.erb
rm app/Settings/login.bb.erb
rm app/Settings/reset.bb.erb
rm app/Settings/wait.bb.erb
cd C:/Ruby192/catalog/bin/RhoBundle/lib/res
rm back_btn.wm.png
rm blue_pushpin.wm.png
```

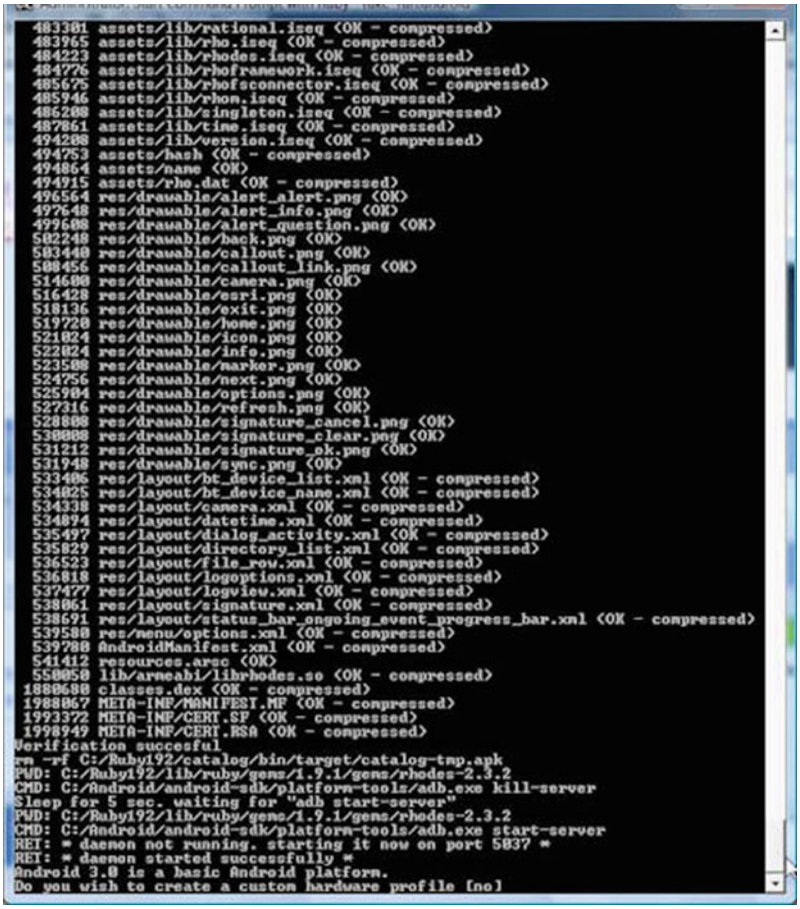
Fig. 1.7 Running the rake command to start an Android Virtual Device instance

The Rhodes application catalog is shown installed on the simulator as shown in Fig. 1.10.

Click on the `catalog` application to start the application. The catalog application loading starts as shown in Fig. 1.11.

Click on the default **Login** button as shown in Fig. 1.12.

Specify **Login** and **Password** and click on **Login** as shown in Fig. 1.13. The **Login** page is just a test login page, it does not really login into an application or website.



```
  483301 assets/lib/rational.iseq (OK - compressed)
  483965 assets/lib/rho.iseq (OK - compressed)
  484223 assets/lib/rhodes.iseq (OK - compressed)
  484776 assets/lib/rhoframework.iseq (OK - compressed)
  485675 assets/lib/rhofsconnector.iseq (OK - compressed)
  485946 assets/lib/rhom.iseq (OK - compressed)
  486208 assets/lib/singleton.iseq (OK - compressed)
  487861 assets/lib/time.iseq (OK - compressed)
  494208 assets/lib/version.iseq (OK - compressed)
  494753 assets/hash (OK - compressed)
  494864 assets/name (OK)
  494915 assets/rho.dat (OK - compressed)
  496564 res/drawable/alert_alert.png (OK)
  497648 res/drawable/alert_info.png (OK)
  499608 res/drawable/alert_question.png (OK)
  502248 res/drawable/back.png (OK)
  503440 res/drawable/callout.png (OK)
  508456 res/drawable/callout_link.png (OK)
  514600 res/drawable/camera.png (OK)
  516428 res/drawable/esri.png (OK)
  518136 res/drawable/exit.png (OK)
  519720 res/drawable/home.png (OK)
  521024 res/drawable/icon.png (OK)
  522024 res/drawable/info.png (OK)
  523508 res/drawable/marker.png (OK)
  524756 res/drawable/next.png (OK)
  525904 res/drawable/options.png (OK)
  527316 res/drawable/refresh.png (OK)
  528808 res/drawable/signature_cancel.png (OK)
  530008 res/drawable/signature_clear.png (OK)
  531212 res/drawable/signature_ok.png (OK)
  531948 res/drawable/sync.png (OK)
  533406 res/layout/bt_device_list.xml (OK - compressed)
  534025 res/layout/bt_device_name.xml (OK - compressed)
  534338 res/layout/camera.xml (OK - compressed)
  534894 res/layout/datetime.xml (OK - compressed)
  535497 res/layout/dialog_activity.xml (OK - compressed)
  535829 res/layout/directory_list.xml (OK - compressed)
  536523 res/layout/file_row.xml (OK - compressed)
  536818 res/layout/logoptions.xml (OK - compressed)
  537477 res/layout/logview.xml (OK - compressed)
  538061 res/layout/signature.xml (OK - compressed)
  538691 res/layout/status_bar_ongoing_event_progress_bar.xml (OK - compressed)
  539580 res/menu/options.xml (OK - compressed)
  539780 AndroidManifest.xml (OK - compressed)
  541412 resources.arsc (OK)
  550050 lib/armeabi/librhodes.so (OK - compressed)
 1880680 classes.dex (OK - compressed)
 1988067 META-INF/MANIFEST.MF (OK - compressed)
 1993372 META-INF/CERT.SF (OK - compressed)
 1998949 META-INF/CERT.RSA (OK - compressed)
Verification succesful
rm -rf C:/Ruby192/catalog/bin/target/catalog-tmp.apk
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe kill-server
Sleep for 5 sec. waiting for "adb start-server"
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe start-server
RET: * daemon not running. starting it now on port 5037 *
RET: * daemon started successfully *
Android 3.0 is a basic Android platform.
Do you wish to create a custom hardware profile [no]
```

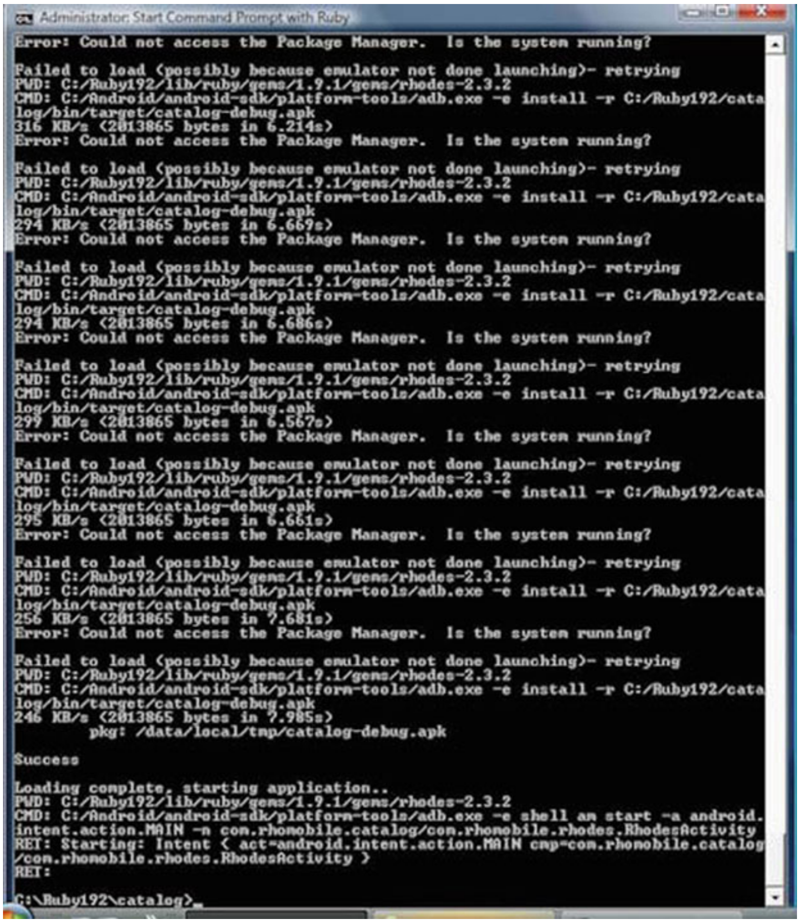
Fig. 1.8 Selecting the default configuration for an Android Virtual device

1.6 Creating a Rhodes Model for a Catalog

Rhodes provides the `rhodes model` command to generate model and controller files, and view templates, and is run with the following command format.

```
rhodes model modelname options
```

The `rhodes model` command generates a scaffolding similar to the Ruby on Rails framework to perform CRUD operations on the model. Cd to the application folder and generate a scaffolding for a catalog, which includes the `journal`, `publisher`, `edition`, `title`, `author` attributes.



```
Administrator: Start Command Prompt with Ruby
Error: Could not access the Package Manager.  Is the system running?

Failed to load (possibly because emulator not done launching)- retrying
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe -e install -r C:/Ruby192/cata
log/bin/target/catalog-debug.apk
316 KB/s (2013865 bytes in 6.214s)
Error: Could not access the Package Manager.  Is the system running?

Failed to load (possibly because emulator not done launching)- retrying
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe -e install -r C:/Ruby192/cata
log/bin/target/catalog-debug.apk
294 KB/s (2013865 bytes in 6.669s)
Error: Could not access the Package Manager.  Is the system running?

Failed to load (possibly because emulator not done launching)- retrying
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe -e install -r C:/Ruby192/cata
log/bin/target/catalog-debug.apk
294 KB/s (2013865 bytes in 6.686s)
Error: Could not access the Package Manager.  Is the system running?

Failed to load (possibly because emulator not done launching)- retrying
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe -e install -r C:/Ruby192/cata
log/bin/target/catalog-debug.apk
299 KB/s (2013865 bytes in 6.567s)
Error: Could not access the Package Manager.  Is the system running?

Failed to load (possibly because emulator not done launching)- retrying
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe -e install -r C:/Ruby192/cata
log/bin/target/catalog-debug.apk
295 KB/s (2013865 bytes in 6.661s)
Error: Could not access the Package Manager.  Is the system running?

Failed to load (possibly because emulator not done launching)- retrying
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe -e install -r C:/Ruby192/cata
log/bin/target/catalog-debug.apk
256 KB/s (2013865 bytes in 7.681s)
Error: Could not access the Package Manager.  Is the system running?

Failed to load (possibly because emulator not done launching)- retrying
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe -e install -r C:/Ruby192/cata
log/bin/target/catalog-debug.apk
246 KB/s (2013865 bytes in 7.985s)
        pkg: /data/local/tmp/catalog-debug.apk

Success

Loading complete, starting application..
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/Android/android-sdk/platform-tools/adb.exe -e shell am start -a android.
intent.action.MAIN -n com.rhomobile.catalog/com.rhomobile.rhodes.RhodesActivity
RET: Starting: Intent { act=android.intent.action.MAIN cmp=com.rhomobile.catalog
/com.rhomobile.rhodes.RhodesActivity }
RET:

C:\Ruby192\catalog>_
```

Fig. 1.9 Uploading the Rhodes application and starting the activity

```
  C:\Ruby192\catalog>rhodes model catalog jour-
nal,publisher,edition,title,author
```

The view templates `index.erb`, `edit.erb`, `new.erb` and `show.erb` get generated. Controller file `catalog_controller.rb` and model file `catalog.rb` also get generated as shown in Fig. 1.14.

The controller class extends the `Rho:RhoController` class and includes actions `index`, `edit`, `show`, `new`, `create`, `update` and `delete` for CRUD operations.

```
classCatalogController< Rho::RhoController
end
```

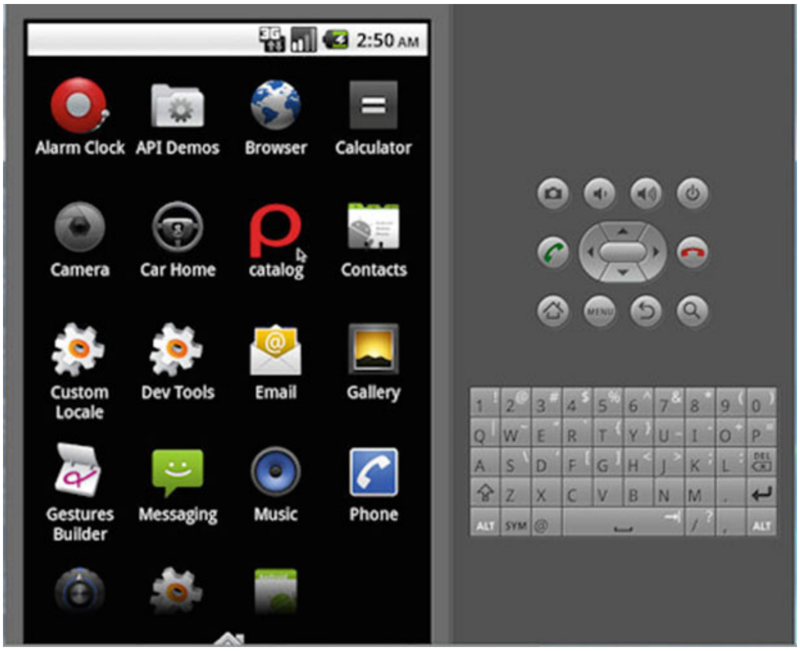


Fig. 1.10 Rhodes application installed in the Android Virtual Device

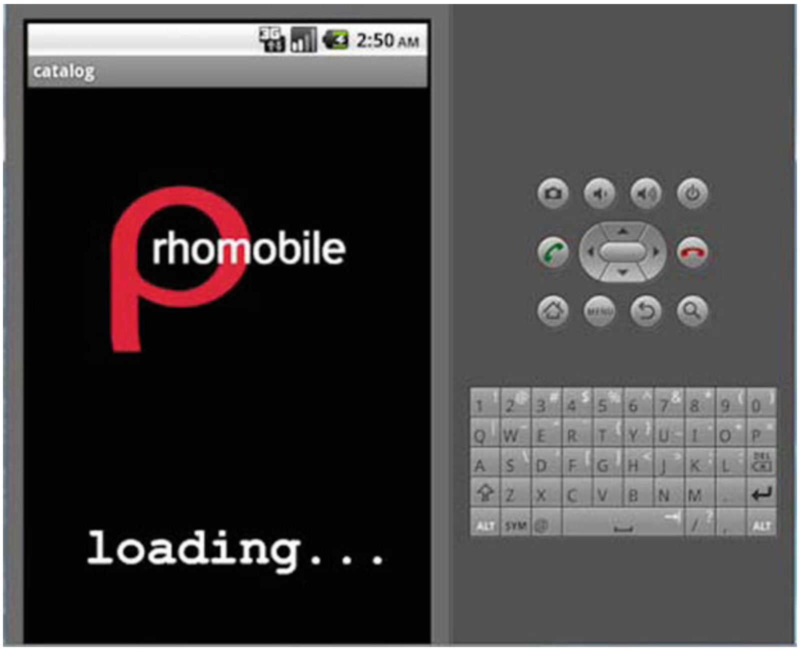


Fig. 1.11 Loading the Rhodes application

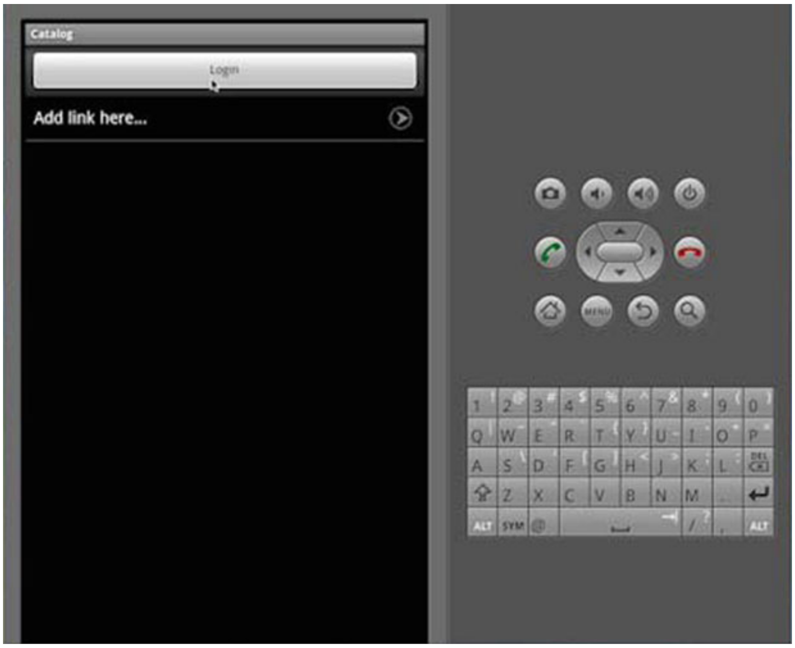


Fig. 1.12 Login button

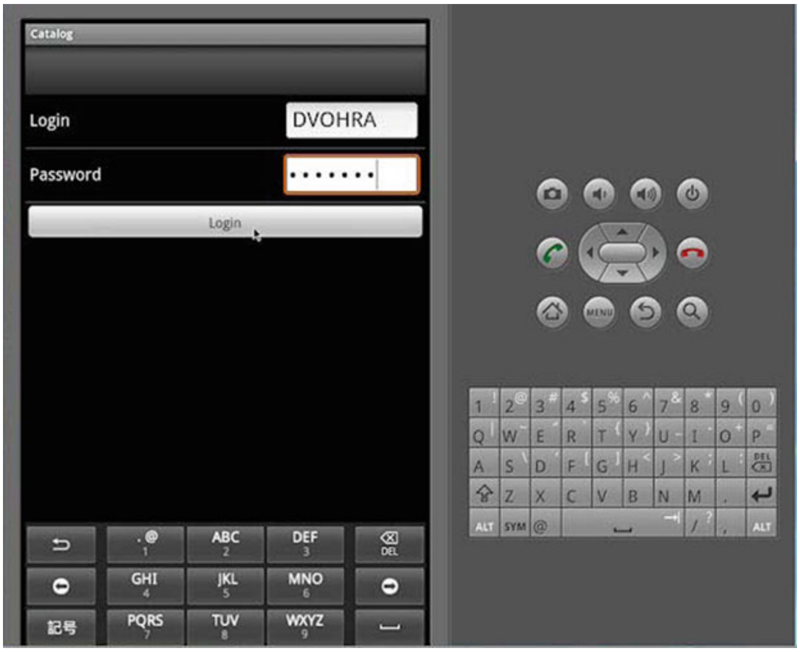
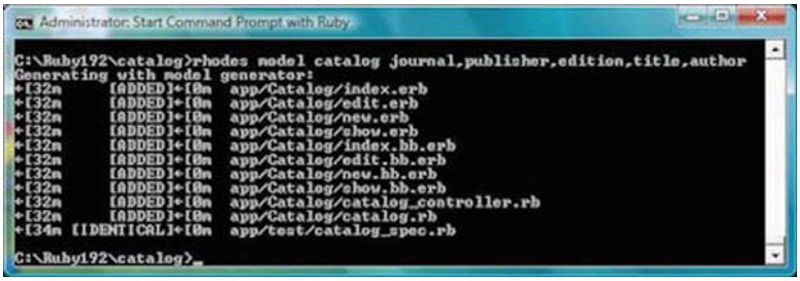


Fig. 1.13 Testing the login page



```
C:\Ruby192\catalog>rhodes model catalog journal,publisher,edition,title,author
Generating with model generator:
←[32m      [ADDED]←[0m  app/Catalog/index.erb
←[32m      [ADDED]←[0m  app/Catalog/edit.erb
←[32m      [ADDED]←[0m  app/Catalog/new.erb
←[32m      [ADDED]←[0m  app/Catalog/show.erb
←[32m      [ADDED]←[0m  app/Catalog/index.bb.erb
←[32m      [ADDED]←[0m  app/Catalog/edit.bb.erb
←[32m      [ADDED]←[0m  app/Catalog/new.bb.erb
←[32m      [ADDED]←[0m  app/Catalog/show.bb.erb
←[32m      [ADDED]←[0m  app/Catalog/catalog_controller.rb
←[32m      [ADDED]←[0m  app/Catalog/catalog.rb
←[34m [IDENTICAL]←[0m  app/test/catalog_spec.rb

C:\Ruby192\catalog>_
```

Fig. 1.14 Creating a model for a Rhodes application

Next, we shall upload the application to the Android emulator and test the application in the emulator. Before we may run the emulator we need to specify the application to run in the `catalog/rhoconfig.txt` file.

# startup page for your application.

```
start_path = '/app/Catalog'
```

To run the emulator and upload the Rhodes model catalog run the command:

```
C:\Ruby192\catalog>rake run:android
```

The Android emulator gets started and the Rhodes application gets uploaded to the emulator. The catalog application gets started in the emulator. Click on **New** to create a catalog entry as shown in Fig. 1.15.

Specify **Journal**, **Publisher**, **Edition**, **Title** and **Author** and click on **Create** as shown in Fig. 1.16.

A new catalog entry gets created. Click on the icon for a catalog entry to display the entry as shown in Fig. 1.17.

The selected catalog entry's detail gets listed as shown in Fig. 1.18.

1.7 Creating a Rhodes Model to Get RSS Feed

In the previous sections we have only tested the default model generated by Rhodes. In this section we shall create a Rhodes model to get a RSS feed and display the feed in the Android. We shall use the IBM developerWorks RSS Feed (http://www.ibm.com/developerworks/views/opensource/rss/libraryview.jsp) for the example. The RSS feed is in XML format and contains entries as `<item></item>` elements, the root element being `<rss></rss>`. Create a Rhodes model `CatalogRSSFeed` with attributes `title`, `link`, `description`, and `date` with the following command:

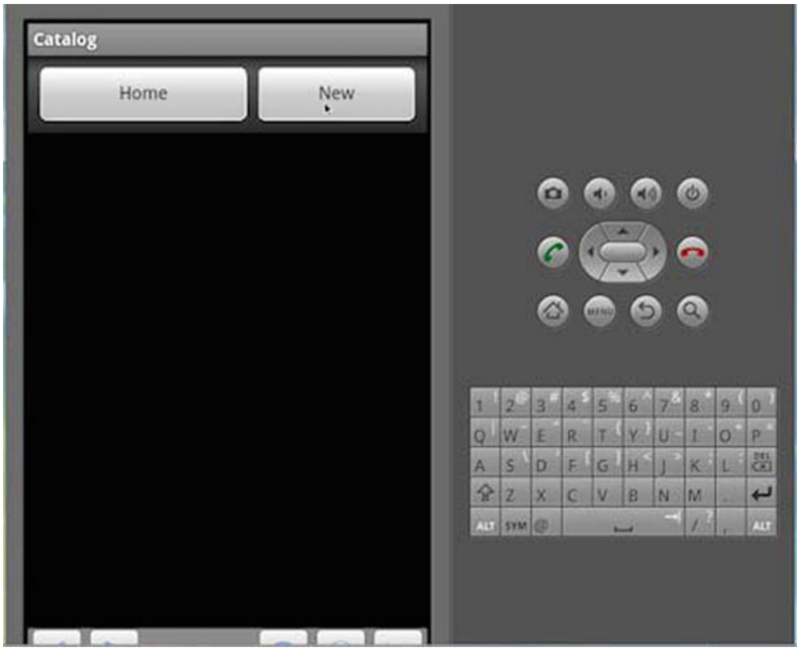
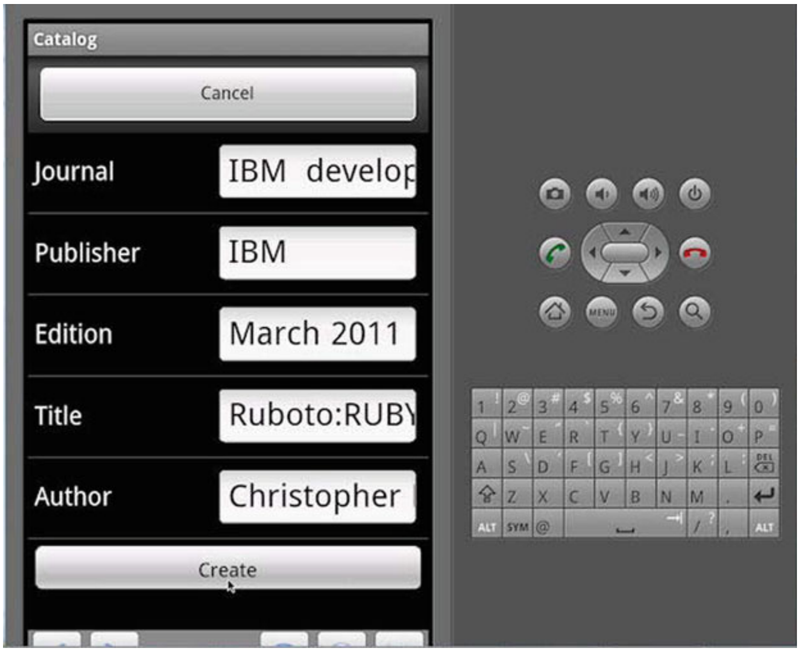


Fig. 1.15 Creating a new catalog entry



Journal IBM develop
Publisher IBM
Edition March 2011
Title Ruboto:RUBY
Author Christopher

Fig. 1.16 Specifying fields for a catalog entry

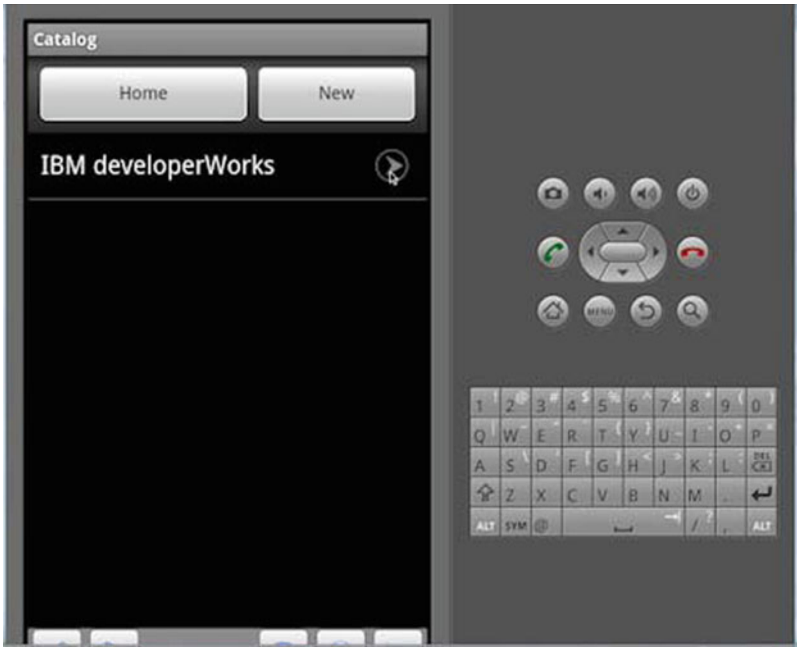
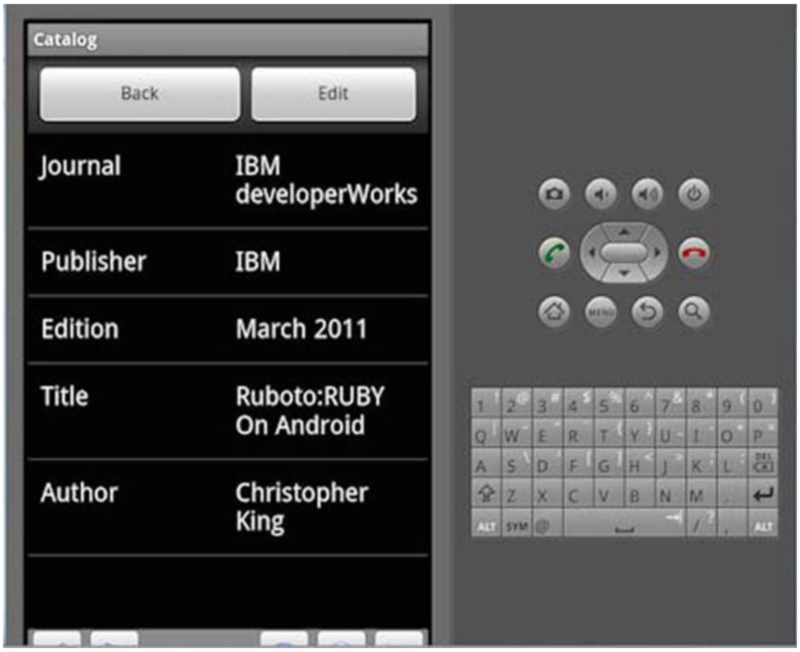
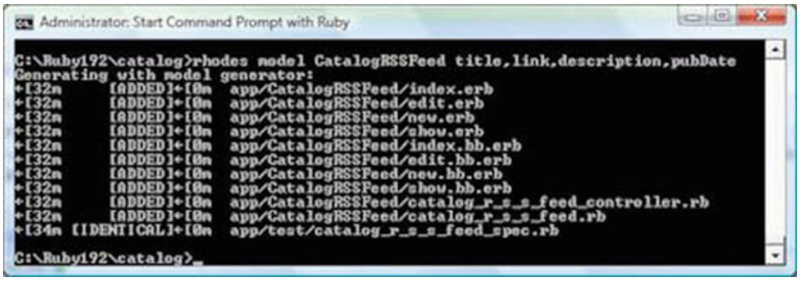


Fig. 1.17 A new catalog entry



Journal	IBM developerWorks
Publisher	IBM
Edition	March 2011
Title	Ruboto:RUBY On Android
Author	Christopher King

Fig. 1.18 A catalog entry detail



```
C:\Ruby192\catalog>rhodes model CatalogRSSFeed title,link,description,pubDate
Generating with model generator:
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/index.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/edit.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/new.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/show.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/index.bb.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/edit.bb.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/new.bb.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/show.bb.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/catalog_r_s_s_feed_controller.rb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/catalog_r_s_s_feed.rb
←[34m [IDENTICAL]←[0m  app/test/catalog_r_s_s_feed_spec.rb

C:\Ruby192\catalog>_
```

Fig. 1.19 Creating a model for a Rhodes RSS application

Table 1.1 Parameters for the get method

Parameter	Description
:url	URL to send the requests to
:headers	Hash of headers to send with the request
:callback	Callback action to execute when the request is completed
:callback_params	Callback parameters (optional)

```
C:\Ruby192\catalog>rhodes model CatalogRSSFeed-
title,link,description,pubDate
```

The model, controller, and view template files get generated in the `app/CatalogRSSFeed` folder as shown in Fig. 1.19.

We won't be using the default view templates and actions for CRUD operations, but shall modify the controller class to get the RSS feed, parse the XML feed and display the results in the Android. For XML feed we shall require an XML parser. Rhodes includes the RhoXML parser, which is a lightweight parser and does not support some features. We shall used the ReXML parser, for which add support in the `catalog/build.yml` file.

```
extensions: ["json", "rexml", "set"]
```

We shall use the `AsyncHttp` API to get the RSS feed. Use the `get(:url, :headers, :callback, :callback_params)` method for an HTTP GET request. The parameters for the `get` method are discussed in Table 1.1.

Send a HTTP request to the IBM developerWorks RSS feed.

```
url =
'http://www.ibm.com/developerworks/views/opensource/rss
/libraryview.jsp'
Rho::AsyncHttp.get(
        :url =>url,
        :callback => (url_for :action =>
        :httpget_callback),
  :callback_param => "" )
```

In the callback method if status is 'ok' get the result of the request.

```
@@get_result = @params['body']
```

Create a `REXML::Document` object from the result.

```
doc = REXML::Document.new(@@get_result)
```

Using the `REXML::XPATH` class iterate over the`//rss//item` elements in the RSS feed and create a `CatalogRSSFeed` object corresponding to each <`item`> element.

```
REXML::XPath.each(doc,"//rss//item/") do |e|
CatalogRSSFeed.create(:title
=>e.elements['title'].text,
:link =>e.elements['link'].text,
:description =>e.elements['description'].text,
:pubDate =>e.elements['pubDate'].text)
  end
```

In the `index` action create an instance variable for all feed results.

```
@catalogrssfeeds = CatalogRSSFeed.find(:all)
```

In the `index.erb` view template iterate over the `@catalogrssfeeds` instance variable, which contains the feed results and output the feed titles. A request may be cancelled with the `Rho::AsyncHttp.cancel` method. The controller file `catalog_r_s_s_feed_controller.erb` is listed below.

```
require 'rho/rhocontroller'
require 'helpers/browser_helper'

classCatalogRSSFeedController< Rho::RhoController
includeBrowserHelper

def index
    @catalogrssfeeds = CatalogRSSFeed.find(:all)
if @catalogrssfeeds.empty? then
self.update
else


render :action => :index, :back => :exit
end
end


def refresh
CatalogRSSFeed.delete_all
redirect :action => :update
end

def update
url =
'http://www.ibm.com/developerworks/views/opensource/
 rss/libraryview.jsp'

    Rho::AsyncHttp.get(
:url =>url,
:callback => (url_for :action => :httpget_callback),
:callback_param => "" )
render :action => :wait, :back => :exit
end
def show
    @catalogrssfeed = Cata-
logRSSFeed.find(@params['id'])
if @catalogrssfeed
      render :action => :show, :back =>url_for(:action
=> :index )
else
redirect :action => :index
end
end
```

```
defhttpget_callback
if @params['status'] != 'ok'
        @error_params = @params
WebView.navigate( url_for :action => :show_error )
else
        @@get_result = @params['body']
begin
require 'rexml/document'
doc = REXML::Document.new(@@get_result)
REXML::XPath.each(doc,"//rss//item/") do |e|



CatalogRSSFeed.create(:title
=>e.elements['title'].text,
:link =>e.elements['link'].text,
:description =>e.elements['description'].text,
:pubDate =>e.elements['pubDate'].text)
end
            @catalogrssfeeds = Cata-
logRSSFeed.find(:all)
if @catalogrssfeeds.empty?
WebView.navigate( url_for :action => :show_error )
else
WebView.navigate( url_for :action => :index )
end
rescue Exception => e
puts "Error: #{e}"
            @@get_result = "Error: #{e}"
end
end
end
defcancel_httpcall
    Rho::AsyncHttp.cancel( url_for( :action =>
:httpget_callback) )
    @@get_result  = 'Request was cancelled.'
render :action => :index, :back => :exit
end

defget_res
    @@get_result
end
```

```
defget_error
    @@error_params
end
defshow_error
    render :action =>:error,:back =>url_for(:action
=> :index )
end

def exit
    Rho::RhoApplication.close
System.exit
end
  end
```

The `index.erb` view template is listed below.

```
<div class="pageTitle">
<h1>CatalogRSSFeeds</h1>
</div>
<div class="toolbar">

<div class="regularButton">
<a class="button" href="<%= url_for :action =>:refresh
%>">Refresh</a>
</div>
</div>

<div class="content">
<ul>
<% @catalogrssfeeds.eachdo |catalogrssfeed| %>
<li>
<a href="<%= url_for :action => :show, :id
=>catalogrssfeed.object %>">
<span class="title"><%= catalogrssfeed.title
%></span><span
class="disclosure_indicator"></span>
</a>
</li>


<% end %>
</ul>
  </div>
```

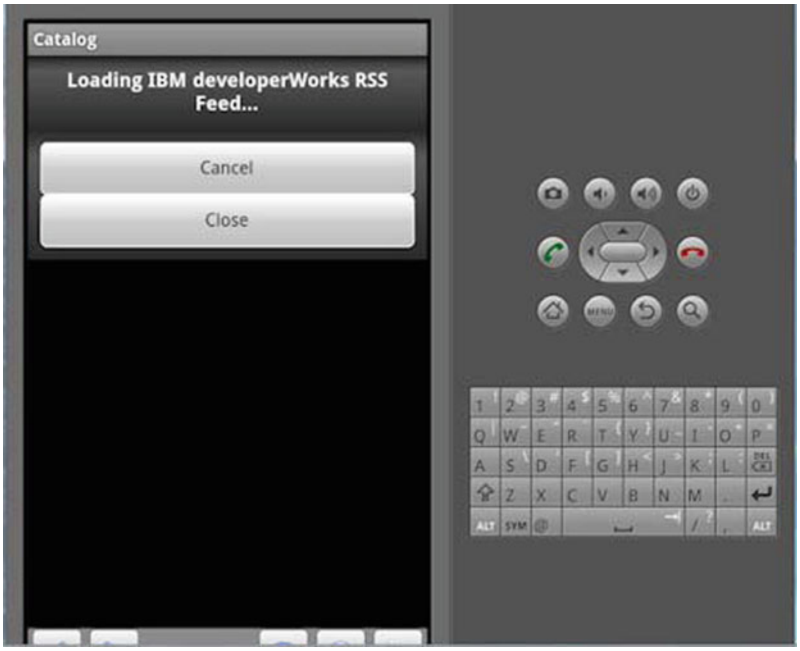
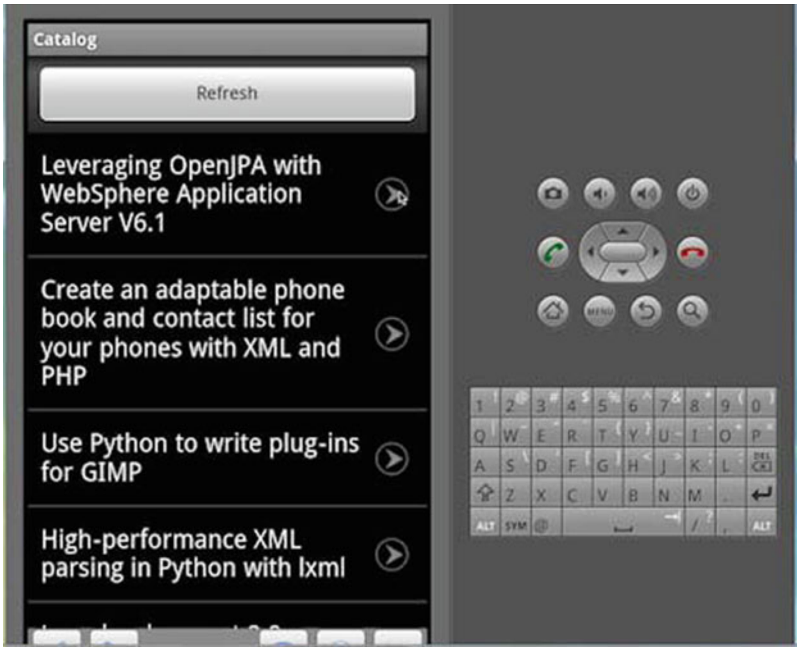


Fig. 1.20 Starting the Rhodes application, the wait message



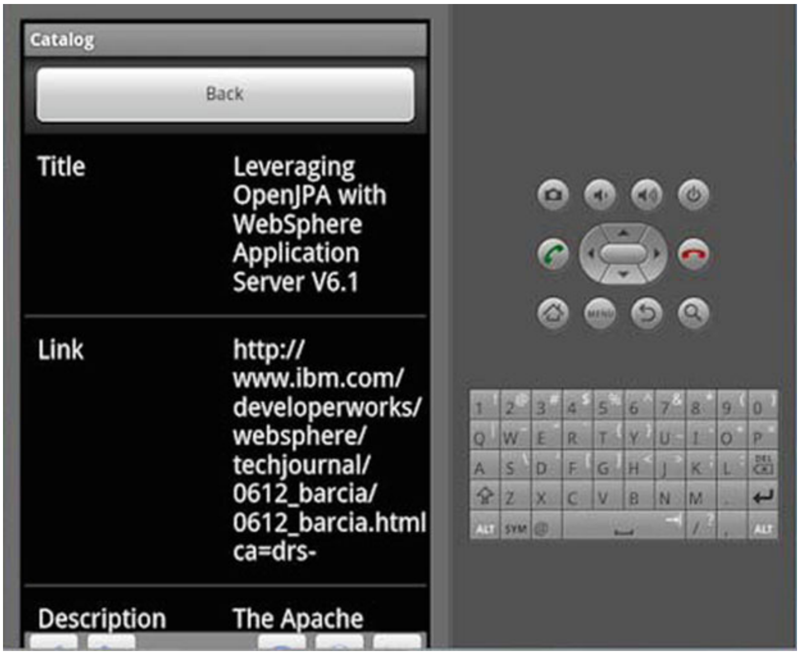
Leveraging OpenJPA with WebSphere Application Server V6.1

Create an adaptable phone book and contact list for your phones with XML and PHP

Use Python to write plug-ins for GIMP

High-performance XML parsing in Python with lxml

Fig. 1.21 The RSS feed



Title Leveraging OpenJPA with WebSphere Application Server V6.1

Link http://www.ibm.com/developerworks/websphere/techjournal/0612_barcia/0612_barcia.html ca=drs-

Description The Apache

Fig. 1.22 A RSS feed entry detail

Modify the start path in the `catalog/rhoconfig.txt` file for the RSS feed application.

```
# startup page for your application
start_path = '/app/CatalogRSSFeed'
```

Start the Android emulator as before, with the command:

```
C:\Ruby192\catalog>rake run:android
```

The IBM developerWorks RSS Feed application gets started in the Android emulator as shown in Fig. 1.20.

The IBM developerWorks RSS Feed gets listed in Android as shown in Fig. 1.21. Select a feed entry to display the entry detail.

The feed entry detail gets displayed as shown in Fig. 1.22.

The RSS feed may be scrolled to display all the entries. In Chap. 2 we shall discuss developing the same Rhodes application on BlackBerry.

Chapter 2
Rhodes on BlackBerry® Smartphones

BlackBerry® has more than 30% (ranked 2nd) of the smartphone market share. In this chapter we shall, first, introduce using Rhodes on Blackberry JDE, and subsequently develop a Rhodes application to get RSS feed for a magazine on Blackberry JDE. Rhodes uses XRuby to generate the Java code from Ruby code. XRuby compiles Ruby classes to Java class files. Though the Rhodes application is a Ruby application, actually the BlackBerry JDE runs Java, which has been compiled from Ruby. The Ruby and Rhodes installation and configuration procedure is the same as in Chap. 1, but is discussed in this chapter for completeness. The Rhodes application is also the same as in Chap. 1, but is discussed in the context of BlackBerry.

2.1 The ReXML Parser

The ReXML parser provides various classes to parse and process an XML document. Some of those classes are discussed in Table 2.1.

We shall be using only the `Document` and `XPath` classes in this article. Some of the methods in the `Document` class are discussed in Table 2.2.

The `XPath` class provides the methods discussed in Table 2.3.

2.2 Installing the BlackBerry JDE

As a pre-requisite to installing BlackBerry on Windows, install the DirectX SDK from http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=6812. Download the BlackBerry Java Development Environment (JDE) v6.0 from http://us.blackberry.com/developers/javaappdev/javadevenv.jsp. Double-click on the `BlackBerry_JDE_6.0.0.exe` to install the JDE. Install BlackBerry JDE

Table 2.1 ReXML parser

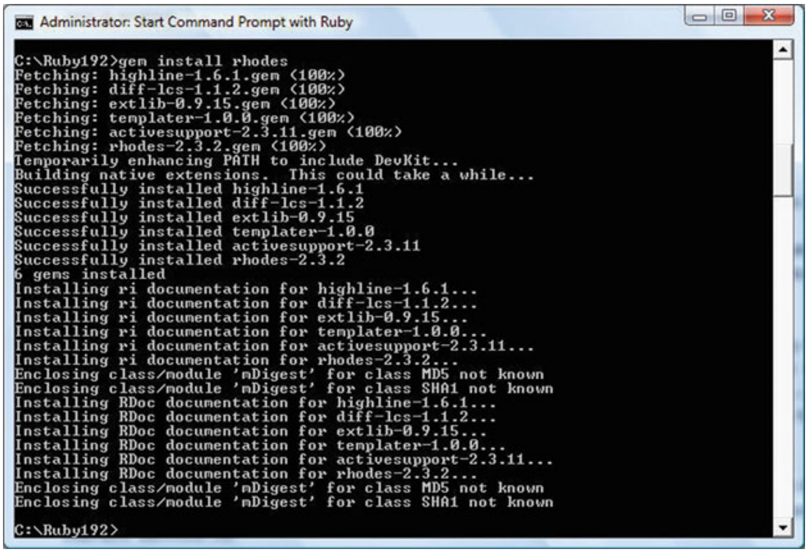
Class	Description
REXML::Attribute	Represents an Element Attribute
REXML::DocType	Represents an XML DOCTYPE declaration
REXML::Document	Represents a full XML document
REXML::Element	Represents an XML Element
REXML::Node	Represents a node
REXML::Parsers:: PullParser	Represents a pull parser
REXML::Parsers:: SAX2Parser	Represents a SAX2 parser
REXML::Text	Represents a text node
REXML::XPath	Wrapper class for XPath functions

Table 2.2 Document class methods

Method	Description
Add	Adds a node
Add_element	Adds an element
Doctype	Returns the DocType of the document if present, or nil
Encoding	Returns the encoding if set, or returns the default encoding
New	Constructor for a new document
Root	Returns the root element
Version	Returns the version if set, or the default version
Write	Outputs the XML document tree
Xml_decl	Returns the XML declaration if set, or the default declaration

Table 2.3 XPath class methods

Method	Description
Each(element, path = nil, namespaces = nil, variables = {})	Takes a context element, the xpath to search for, and a Hash for namespace mapping as parameters, and iterates over nodes that match the specified path. If the xpath is not specified the default xpath is ''*'
First(element, path = nil, namespaces = nil, variables = {})	Returns the first nodes that matches the specified xpath. The parameters are the same as the each method
Match(element, path = nil, namespaces = nil, variables = {})	Returns an array of nodes that match the specified xpath



```
C:\Ruby192>gem install rhodes
Fetching: highline-1.6.1.gem (100%)
Fetching: diff-lcs-1.1.2.gem (100%)
Fetching: extlib-0.9.15.gem (100%)
Fetching: templater-1.0.0.gem (100%)
Fetching: activesupport-2.3.11.gem (100%)
Fetching: rhodes-2.3.2.gem (100%)
Temporarily enhancing PATH to include DevKit...
Building native extensions.  This could take a while...
Successfully installed highline-1.6.1
Successfully installed diff-lcs-1.1.2
Successfully installed extlib-0.9.15
Successfully installed templater-1.0.0
Successfully installed activesupport-2.3.11
Successfully installed rhodes-2.3.2
6 gems installed
Installing ri documentation for highline-1.6.1...
Installing ri documentation for diff-lcs-1.1.2...
Installing ri documentation for extlib-0.9.15...
Installing ri documentation for templater-1.0.0...
Installing ri documentation for activesupport-2.3.11...
Installing ri documentation for rhodes-2.3.2...
Enclosing class/module 'mDigest' for class MD5 not known
Enclosing class/module 'mDigest' for class SHA1 not known
Installing RDoc documentation for highline-1.6.1...
Installing RDoc documentation for diff-lcs-1.1.2...
Installing RDoc documentation for extlib-0.9.15...
Installing RDoc documentation for templater-1.0.0...
Installing RDoc documentation for activesupport-2.3.11...
Installing RDoc documentation for rhodes-2.3.2...
Enclosing class/module 'mDigest' for class MD5 not known
Enclosing class/module 'mDigest' for class SHA1 not known

C:\Ruby192>
```

Fig. 2.1 Installing Rhodes

in a directory without spaces in the directory path; for example not in a `C:/Program Files/` sub-directory. The same applies for the JDK 6, which is required for the BlackBerry JDE; install the JDK in a directory without spaces in the directory path. Add `JDK_HOME` and `JDK_HOME/bin` to the `PATH` environment variable. In a later section we shall configure a Rhodes application to used the BlackBerry simulator.

2.3 Installing Rhodes

As Rhodes is a Ruby gem, we need to install Ruby first. As in Chap. 1, download the `rubyinstaller-1.9.2-p180.exe` application. Double-click on the `.exe` file to install Ruby. Install Ruby in a directory without spaces in the directory path as shown in Fig. 2.1. Select the checkbox Add Ruby executables to your `PATH`.

Next, install the Ruby Installer Development Kit (DevKit), which makes it easy to build native Ruby extensions. Download the `DevKit-tdm-32-4.5.1-20101214-1400-sfx.exe`application. Double-click on the `.exe` file and install the self-extracting executables in a directory without spaces, such as `C:/Ruby192/DevKit`. Cd (change directory) to the `DevKit` directory and run the following commands:

```
rubydk.rbinit
rubydk.rb install
```

The `init` command creates a `config.xml` file, which lists the RubyInstaller installed Rubies. Install RubyGems, a Ruby packaging system. Download the RubyGems zip file and extract the zip file to a directory, Cd to the directory and run the following command:

```
C:Ruby192\rubygems-1.6.2>ruby setup.rb
```

We also need to install gnuwin32, which provides win32 ports of GNU tools, GNU being a UNIX-like operating system. Download the `GetGnuWin32-0.6.3.exe` application from http://sourceforge.net/projects/getgnuwin32/files/getgnuwin32/and double-click on the exe file. Install in a folder (`C:/Ruby192` for example). Cd to the installed folder (`C:/Ruby192/GetGnuWin32`) and run the download command.

```
C:\Ruby192\GetGnuWin32>download
```

Subsequently, run the `install` command:

```
C:\Ruby192\GetGnuWin32>install C:/gnuwin32
```

Add `C:/gnuwin32/bin` to the `PATH` environment variable. Next, install Rake, a Ruby build program, with the following command:

```
C:\Ruby192>gem install rake
```

To install Rhodes run the following command:

```
C:\Ruby192>gem install Rhodes
```

Rhodes 2.3.2 and related gems get installed as shown in Fig. 2.1.

2.4 Creating a Rhodes Application

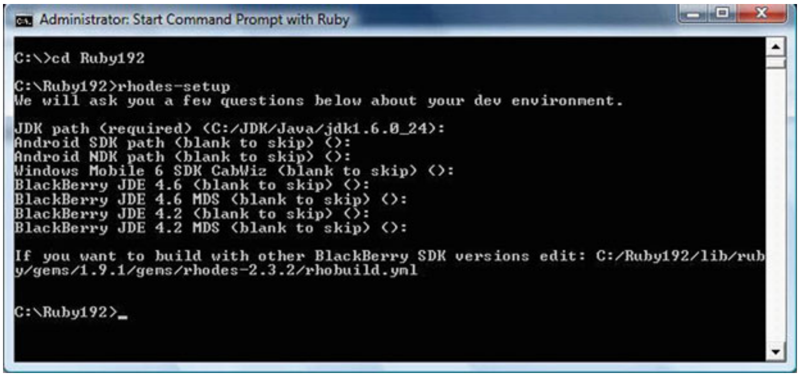
Rhodes provides an application generator to generate an application. The Rhodes application generator is called `rhodes` and is run with the following command format:

```
rhodes app <application_name>
```

Before we may run the `rhodes` command we need to setup Rhodes using the `rhodes-setup` command as shown in Fig. 2.2. Select Enter for each of the questions. The JDK path should not include any spaces in the directory path. By default BlackBerry JDE version 4.6 or less configuration is checked.

As we are running JDE 6, modify the `C:\Ruby192\lib\ruby\gems\1.9.1\gems\rhodes-2.3.2\rhobuild.yml` configuration file, listed below, to include the v6.0. The BlackBerry related settings are shown in bold.

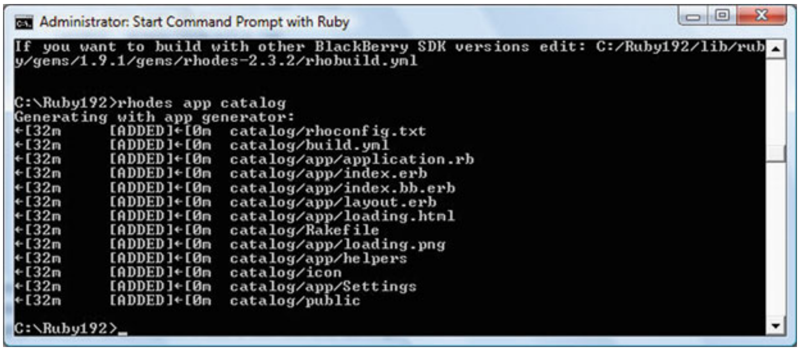
```
env:
  app: C:/rhodes-app
  paths:
    java: C:/JDK/Java/jdk1.6.0_24/bin
    android:
    android-ndk:
    cabwiz:
6.0:
      jde: C:/BlackBerry
      mds: C:/BlackBerry/MDS
      sim: 9800
    4.6:
      jde:
      mds:
      sim: 9000
    4.2:
      jde:
      mds:
      sim: 8100
build:
  bbpath: platform/bb


  wmpath: platform/wm
  androidpath: platform/android
  iphonepath: platform/iphone
  symbianpath: platform/symbian
  bb:
  bbsignpwd: somepasswordhere
android:
excludedirs:
  all:
  - "**/.*.swo"
  - "**/.*.swn"
  - "**/.DS_Store"
  bb:
  - public/js/iui
  - public/js/jquery*
  - public/jqtouch*
  - public/js/prototype*
  - public/css/iphone*
  - public/iwebkit
  - public/themes
  - "**/jquery*.js"
  - "**/*.db"
```



Administrator: Start Command Prompt with Ruby

```
C:\>cd Ruby192

C:\Ruby192>rhodes-setup
We will ask you a few questions below about your dev environment.

JDK path (required) (C:/JDK/Java/jdk1.6.0_24):
Android SDK path (blank to skip) ():
Android NDK path (blank to skip) ():
Windows Mobile 6 SDK CabWiz (blank to skip) ():
BlackBerry JDE 4.6 (blank to skip) ():
BlackBerry JDE 4.6 MDS (blank to skip) ():
BlackBerry JDE 4.2 (blank to skip) ():
BlackBerry JDE 4.2 MDS (blank to skip) ():

If you want to build with other BlackBerry SDK versions edit: C:/Ruby192/lib/rub
y/gems/1.9.1/gems/rhodes-2.3.2/rhobuild.yml

C:\Ruby192>_
```

Fig. 2.2 Setting up Rhodes



Administrator: Start Command Prompt with Ruby

```
If you want to build with other BlackBerry SDK versions edit: C:/Ruby192/lib/rub
y/gems/1.9.1/gems/rhodes-2.3.2/rhobuild.yml


C:\Ruby192>rhodes app catalog
Generating with app generator:
←[32m      [ADDED]←[0m  catalog/rhoconfig.txt
←[32m      [ADDED]←[0m  catalog/build.yml
←[32m      [ADDED]←[0m  catalog/app/application.rb
←[32m      [ADDED]←[0m  catalog/app/index.erb
←[32m      [ADDED]←[0m  catalog/app/index.bb.erb
←[32m      [ADDED]←[0m  catalog/app/layout.erb
←[32m      [ADDED]←[0m  catalog/app/loading.html
←[32m      [ADDED]←[0m  catalog/Rakefile
←[32m      [ADDED]←[0m  catalog/app/loading.png
←[32m      [ADDED]←[0m  catalog/app/helpers
←[32m      [ADDED]←[0m  catalog/icon
←[32m      [ADDED]←[0m  catalog/app/Settings
←[32m      [ADDED]←[0m  catalog/public

C:\Ruby192>_
```

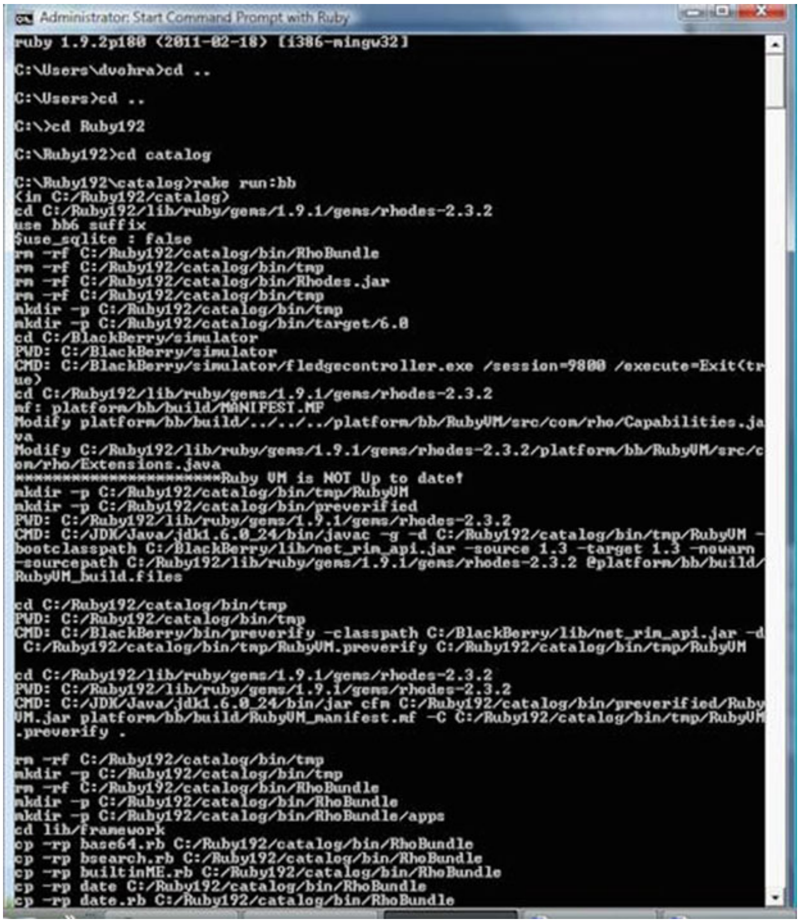
Fig. 2.3 Generating Rhodes application

The `jde` parameter specifies the directory in which the JDE is installed. The `mds` parameter specifies the directory in which the BlackBerry Mobile Data Service (MDS) is installed. The simulator port is specified with the `sim` parameter. Next, run the `rhodes` application generator to create an application called `catalog` with the command:

```
>rhodes app catalog
```

The application files get generated in the `catalog` (application name) directory as shown in Fig. 2.3.

Modify the `catalog/build.yml` file to specify the BlackBerry version as 6.0. The BlackBerry version is specified with the `bbver` property.



```
Administrator: Start Command Prompt with Ruby
ruby 1.9.2p180 (2011-02-18) [i386-mingw32]

C:\Users\dvohra>cd ..

C:\Users>cd ..

C:\>cd Ruby192

C:\Ruby192>cd catalog

C:\Ruby192\catalog>rake run:bb
(in C:/Ruby192/catalog)
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
use bb6 suffix
$use_sqlite : false
rm -rf C:/Ruby192/catalog/bin/RhoBundle
rm -rf C:/Ruby192/catalog/bin/tmp
rm -rf C:/Ruby192/catalog/bin/Rhodes.jar
rm -rf C:/Ruby192/catalog/bin/tmp
mkdir -p C:/Ruby192/catalog/bin/tmp
mkdir -p C:/Ruby192/catalog/bin/target/6.0
cd C:/BlackBerry/simulator
PWD: C:/BlackBerry/simulator
CMD: C:/BlackBerry/simulator/fledgecontroller.exe /session=9800 /execute=Exit(tr
ue)
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
mf: platform/bb/build/MANIFEST.MF
Modify platform/bb/build/../../../platform/bb/RubyVM/src/com/rho/Capabilities.ja
va
Modify C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2/platform/bb/RubyVM/src/c
om/rho/Extensions.java
**************************Ruby VM is NOT Up to date!
mkdir -p C:/Ruby192/catalog/bin/tmp/RubyVM
mkdir -p C:/Ruby192/catalog/bin/preverified
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/JDK/Java/jdk1.6.0_24/bin/javac -g -d C:/Ruby192/catalog/bin/tmp/RubyVM -
bootclasspath C:/BlackBerry/lib/net_rim_api.jar -source 1.3 -target 1.3 -nowarn
-sourcepath C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2 @platform/bb/build/
RubyVM_build.files

cd C:/Ruby192/catalog/bin/tmp
PWD: C:/Ruby192/catalog/bin/tmp
CMD: C:/BlackBerry/bin/preverify -classpath C:/BlackBerry/lib/net_rim_api.jar -d
 C:/Ruby192/catalog/bin/tmp/RubyVM.preverify C:/Ruby192/catalog/bin/tmp/RubyVM

cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/JDK/Java/jdk1.6.0_24/bin/jar cfm C:/Ruby192/catalog/bin/preverified/Ruby
VM.jar platform/bb/build/RubyVM_manifest.mf -C C:/Ruby192/catalog/bin/tmp/RubyVM
.preverify .

rm -rf C:/Ruby192/catalog/bin/tmp
mkdir -p C:/Ruby192/catalog/bin/tmp
rm -rf C:/Ruby192/catalog/bin/RhoBundle
mkdir -p C:/Ruby192/catalog/bin/RhoBundle
mkdir -p C:/Ruby192/catalog/bin/RhoBundle/apps
cd lib/framework
cp -rp base64.rb C:/Ruby192/catalog/bin/RhoBundle
cp -rp bsearch.rb C:/Ruby192/catalog/bin/RhoBundle
cp -rp builtinME.rb C:/Ruby192/catalog/bin/RhoBundle
cp -rp date C:/Ruby192/catalog/bin/RhoBundle
cp -rp date.rb C:/Ruby192/catalog/bin/RhoBundle
```

Fig. 2.4 Compiling and uploading the Rhodes application to BlackBerry JDE

```
  sdk:      "C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-
2.3.2"
  sdkversion: 2.3.2
  name: catalog
  version: 1.0
  vendor: rhomobile
  build: debug
  bbver: 6.0
  wmsdk: "Windows Mobile 6 Professional SDK (ARMV4I)"
  applog: rholog.txt
```

Before developing the application further test the BlackBerry emulator. Cd to the `catalog` directory and run the following command.

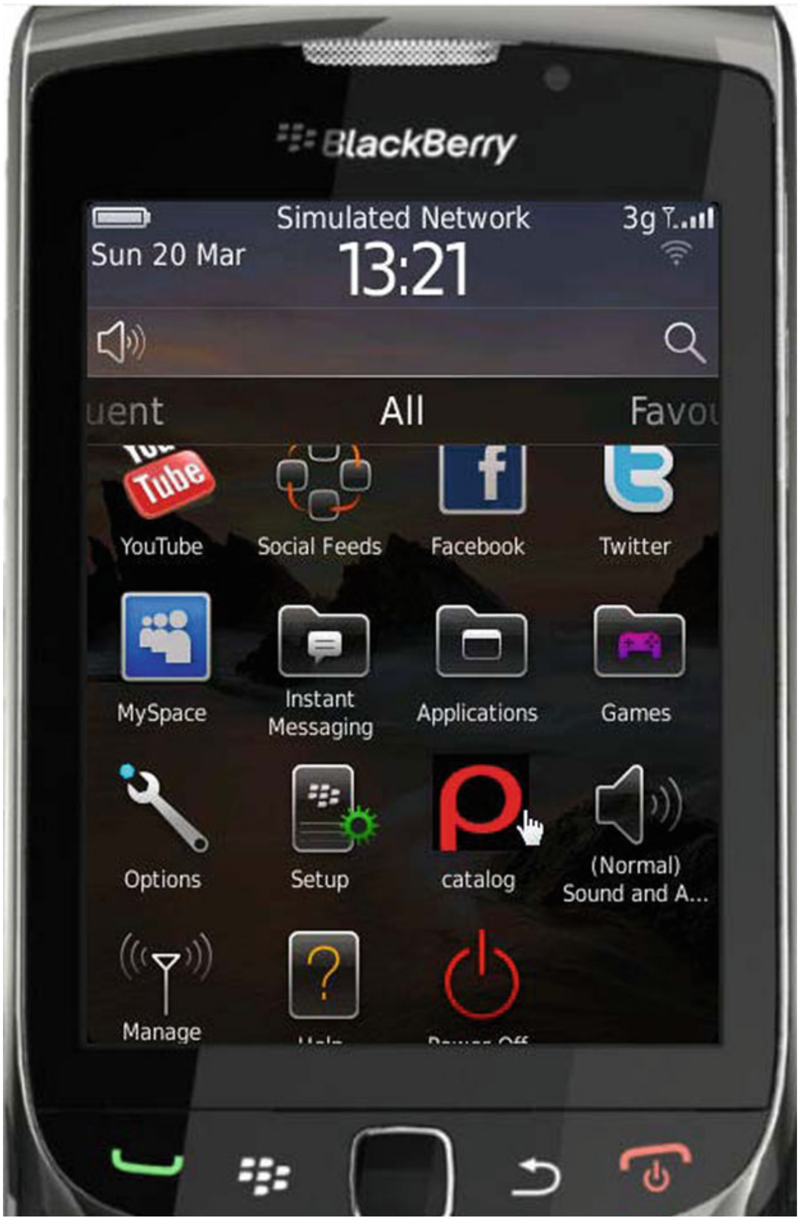


Fig. 2.5 Rhodes catalog application on BlackBerry

```
C:\Ruby192\catalog>rake run:bb
```

The Rhodes application gets built and uploaded to the BlackBerry simulator as shown in Fig. 2.4.

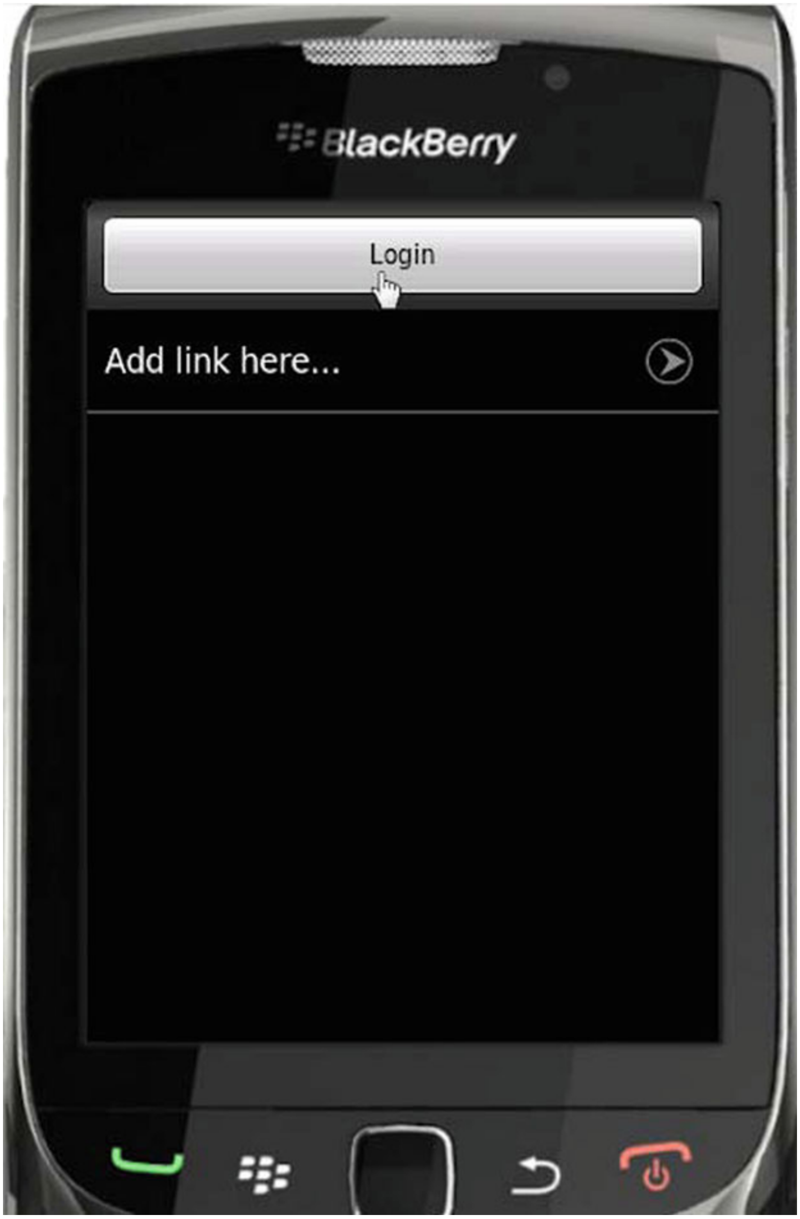


Fig. 2.6 Login page

The BlackBerry simulator gets started. The Rhodes application `catalog` is shown installed on the simulator in Fig. 2.5.

Click on the default Login button as shown in Fig. 2.6.

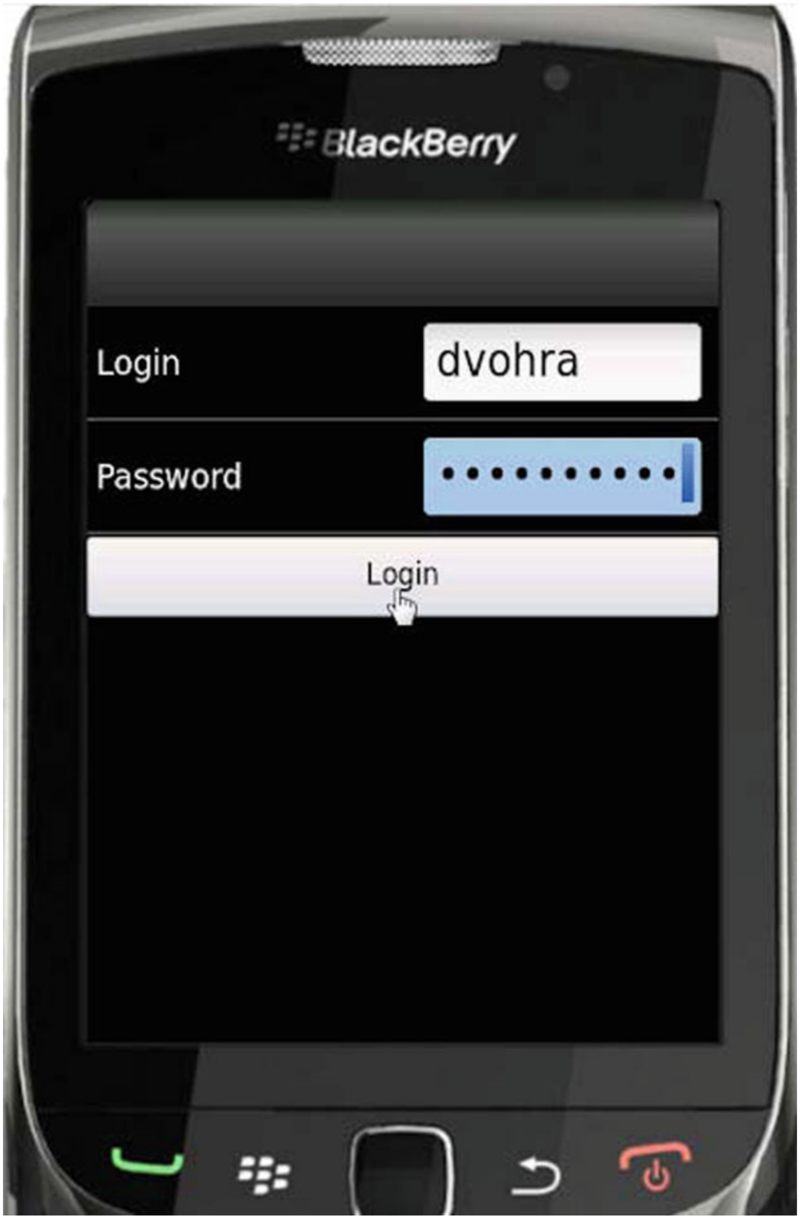
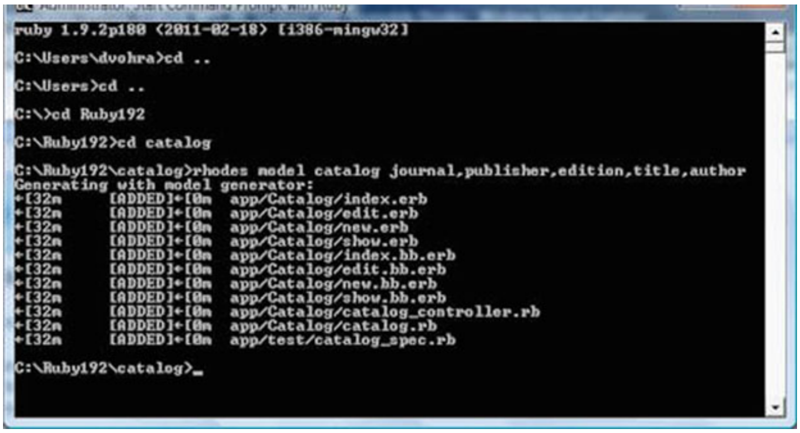


Fig. 2.7 Testing login

Specify Login and Password and click on Login. The Login page is just a test Login as shown in Fig. 2.7.



```
ruby 1.9.2p180 (2011-02-18) [i386-mingw32]

C:\Users\dvohra>cd ..

C:\Users>cd ..

C:\>cd Ruby192

C:\Ruby192>cd catalog

C:\Ruby192\catalog>rhodes model catalog journal,publisher,edition,title,author
Generating with model generator:
←[32m     [ADDED]←[0m  app/Catalog/index.erb
←[32m     [ADDED]←[0m  app/Catalog/edit.erb
←[32m     [ADDED]←[0m  app/Catalog/new.erb
←[32m     [ADDED]←[0m  app/Catalog/show.erb
←[32m     [ADDED]←[0m  app/Catalog/index.bb.erb
←[32m     [ADDED]←[0m  app/Catalog/edit.bb.erb
←[32m     [ADDED]←[0m  app/Catalog/new.bb.erb
←[32m     [ADDED]←[0m  app/Catalog/show.bb.erb
←[32m     [ADDED]←[0m  app/Catalog/catalog_controller.rb
←[32m     [ADDED]←[0m  app/Catalog/catalog.rb
←[32m     [ADDED]←[0m  app/test/catalog_spec.rb

C:\Ruby192\catalog>_
```

Fig. 2.8 Generating Rhodes model and controller

2.5 Creating a Rhodes Model for a Catalog

Rhodes provides the `rhodes model` command to generate model and controller files, and view templates, and is run with the following command format.

```
rhodes model modelname options
```

The `rhodes model` command generates a scaffolding similar to the Ruby on Rails framework to perform CRUD operations on the model. Cd to the application folder and generate a scaffolding for a catalog, which includes the `journal`, `publisher`, `edition`, `title`, `author` attributes.

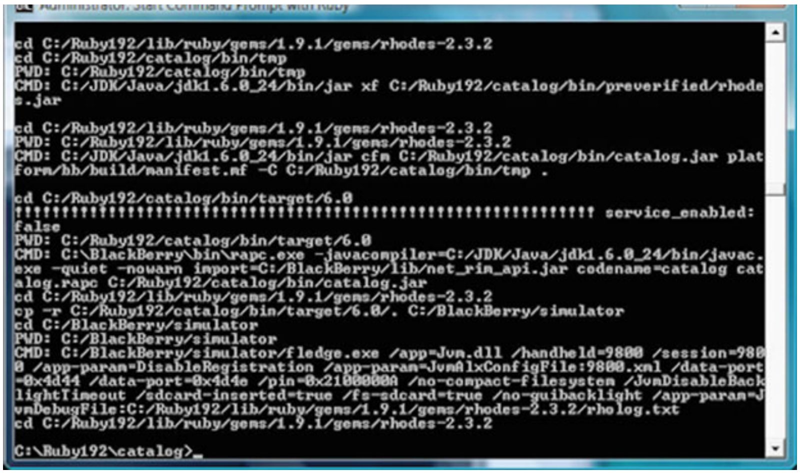
```
C:\Ruby192\catalog>rhodes model catalog jour-
nal,publisher,edition,title,author
```

The view templates `index.erb`, `edit.erb`, `new.erb` and `show.erb` get generated as shown in Fig. 2.8. View templates customized for the BlackBerry get generated as`.bb.erb` extension files; `index.bb.erb`, `edit.bb.erb`, `new.bb.erb`, and `show.bb.erb`. Controller file `catalog_controller.rb` and model file `catalog.rb` also get generated.

The controller class extends the `Rho:RhoController` class and includes actions `index`, `edit`, `show`, `new`, `create`, `update` and `delete` for CRUD operations.

```
class CatalogController < Rho::RhoController

end
```

Next, we shall upload the application to the BlackBerry emulator and test the application in the emulator. Before we may run the emulator we need to specify the application to run in the `catalog/rhoconfig.txt` file.



```
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
cd C:/Ruby192/catalog/bin/tmp
PWD: C:/Ruby192/catalog/bin/tmp
CMD: C:/JDK/Java/jdk1.6.0_24/bin/jar xf C:/Ruby192/catalog/bin/preverified/rhode
s.jar

cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
PWD: C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
CMD: C:/JDK/Java/jdk1.6.0_24/bin/jar cfm C:/Ruby192/catalog/bin/catalog.jar plat
form/bb/build/manifest.mf -C C:/Ruby192/catalog/bin/tmp .

cd C:/Ruby192/catalog/bin/target/6.0
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! service_enabled:
false
PWD: C:/Ruby192/catalog/bin/target/6.0
CMD: C:\BlackBerry\bin\rapc.exe -javacompiler=C:/JDK/Java/jdk1.6.0_24/bin/javac.
exe -quiet -nowarn import=C:/BlackBerry/lib/net_rim_api.jar codename=catalog cat
alog.rapc C:/Ruby192/catalog/bin/catalog.jar
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2
cp -r C:/Ruby192/catalog/bin/target/6.0/. C:/BlackBerry/simulator
cd C:/BlackBerry/simulator
PWD: C:/BlackBerry/simulator
CMD: C:/BlackBerry/simulator/fledge.exe /app=Jvm.dll /handheld=9800 /session=980
0 /app-param=DisableRegistration /app-param=JvmAlxConfigFile:9800.xml /data-port
=0x4d44 /data-port=0x4d4e /pin=0x2100000A /no-compact-filesystem /JvmDisableBack
lightTimeout /sdcard-inserted=true /fs-sdcard=true /no-guibacklight /app-param=J
vmDebugFile:C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2/rholog.txt
cd C:/Ruby192/lib/ruby/gems/1.9.1/gems/rhodes-2.3.2

C:\Ruby192\catalog>_
```

Fig. 2.9 Compiling and uploading Rhodes application to BlackBerry

```
# startup page for your application
start_path = '/app/Catalog'
```

To run the emulator and upload the rhodes model `catalog` run the command:

```
C:\Ruby192\catalog>rake run:bb
```

The BlackBerry simulator gets started. The Rhodes application gets compiled to Java code. The Java code compiled from Ruby code gets packaged into catalog.jar and gets uploaded to the BlackBerry simulator (Fig. 2.9).

The catalog application gets started in the emulator. Click on **New** to create a catalog entry as shown in Fig. 2.10.

Specify **Journal**, **Publisher**, **Edition**, **Title** and **Author** and click on **Create** as shown in Fig. 2.11.

Similarly, new catalog entries may be added. Click on the icon for a catalog entry to display the entry as shown in Fig. 2.12.

The selected catalog entry gets listed as shown in Fig. 2.13.

2.6 Creating a Rhodes Model to get RSS Feed

In the previous sections we have only tested the default model generated by Rhodes. In this section we shall create a rhodes model to get a RSS feed, which is essentially an XML document, parse the XML document, and display the feed in the

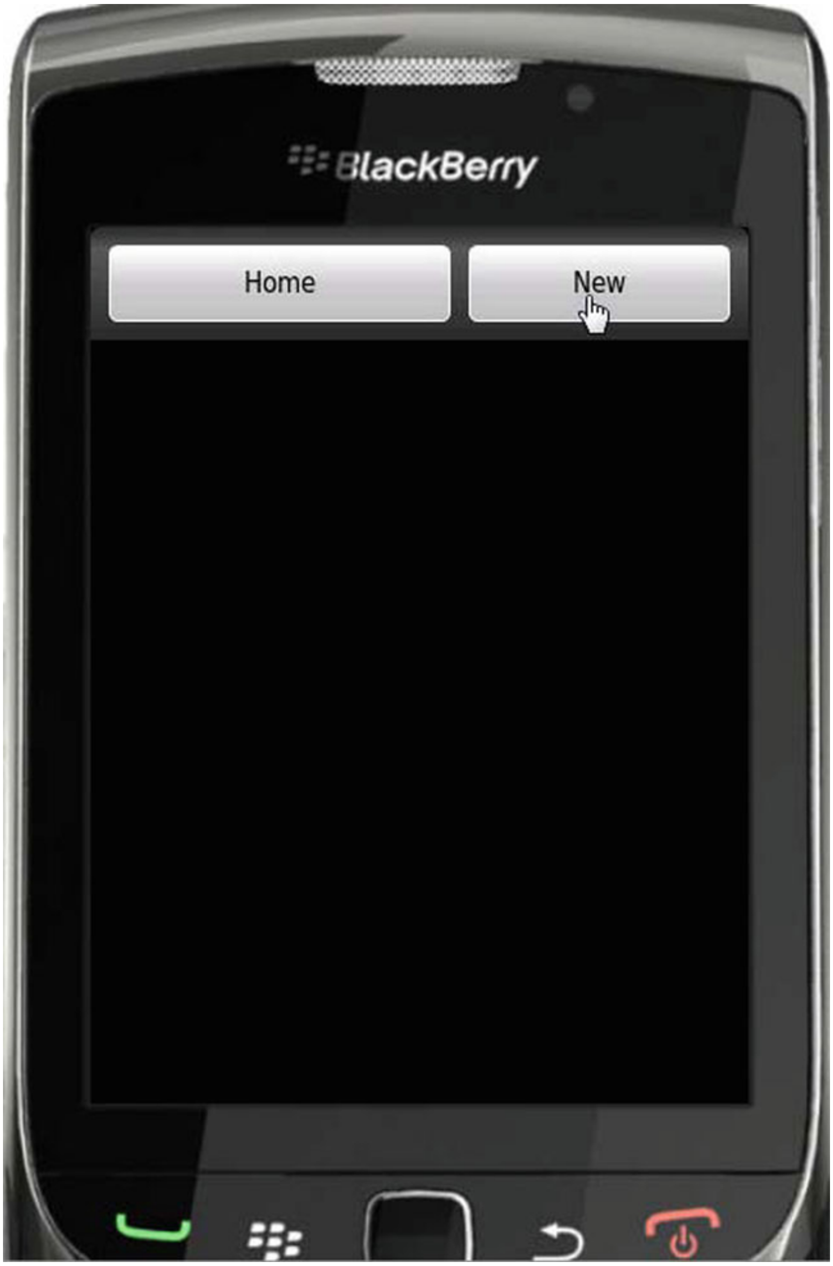
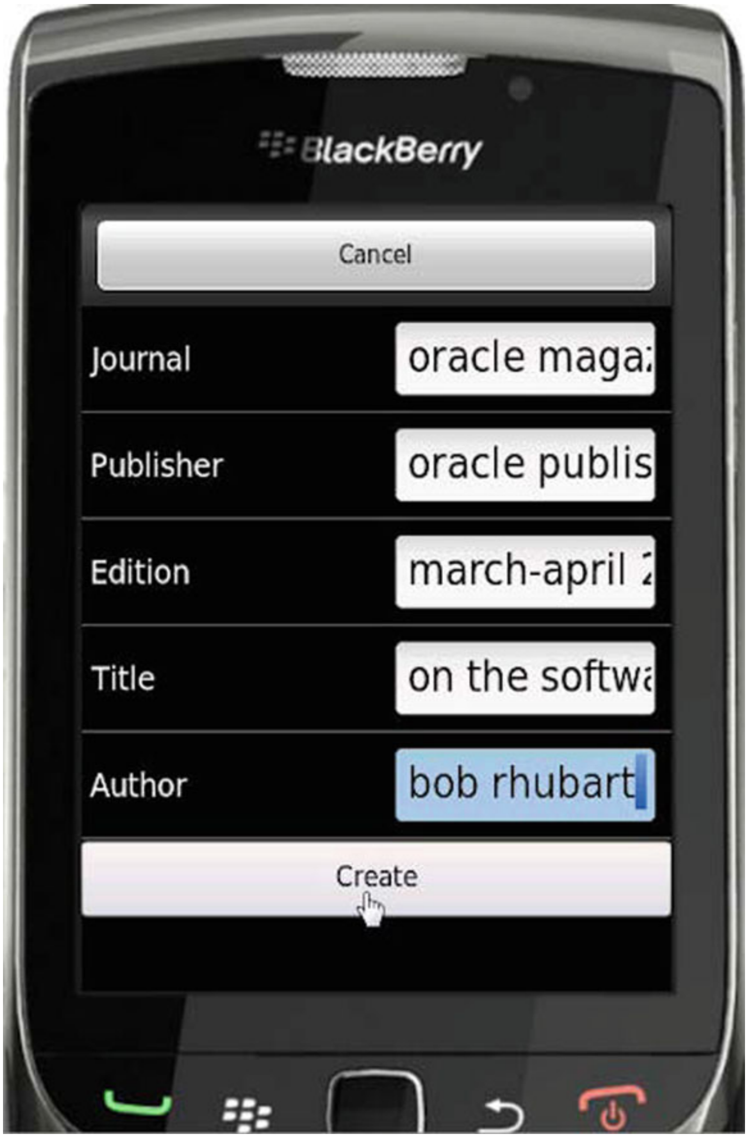


Fig. 2.10 Creating a new catalog entry

BlackBerry. We shall use the Oracle Magazine RSS Feed (http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/rss-oramag-recent.xml) for the example. RSS Feed is essentially an XML document, which we shall parse using



Journal oracle maga:
Publisher oracle publis
Edition march-april 2
Title on the softwa
Author bob rhubart

Fig. 2.11 Specifying catalog entry attributes

the ReXML parser. The RSS feed is in XML format and contains entries as `<item></item>` elements, the root element being `<rss></rss>`. A section of the RSS Feed XML document for the Oracle Magazine is listed below.

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
<title>Oracle Magazine - Most Recent</title>
<link>http://www.oracle.com/technology/oramag/oracle</l
ink>
<description>Here are the latest Oracle Magazine ar-
ticles, columns, and issues.
</description>
<language>en-us</language>
<copyright>Copyright 2008 Oracle. All Rights Re-
served.</copyright>

<managingEditor>opubedit_us@oracle.com</managingEditor>
<pubDate>Mon, 19 Dec 2005 22:04:11 GMT</pubDate>
<lastBuildDate>Fri, 9 Sep 2011 22:21:14
GMT</lastBuildDate>
<item>
<title>Architect: Getting Schooled</title>
<link>http://www.oracle.com/technetwork/issue-
archive/2011/11-sep/o51
      architect-445768.html</link>
<description>Education, training, and experience are
stepping stones to a career as
      a software architect.</description>

<guid isPermaLink="false">{89f1fb2-f946-6951-5b7a-
5f5f54ee330}</guid>
<pubDate>Fri, 9 Sep 2011 22:21:14 GMT</pubDate>
</item>
<item>
<title>PL/SQL: Working with Strings</title>
<link>http://www.oracle.com/technetwork/issue-
archive/2011/11-sep/o51plsql-453456.
      html</link>
<description>Part 3 in a series of articles on under-
standing and using PL/SQL
</description>

<guid isPermaLink="false">{89f1fb2-f946-6951-5b7a-
5f5f54ee330}</guid>
<pubDate>Tue, 6 Sep 2011 20:34:27 GMT</pubDate>
</item>
...
...
</channel></rss>
```

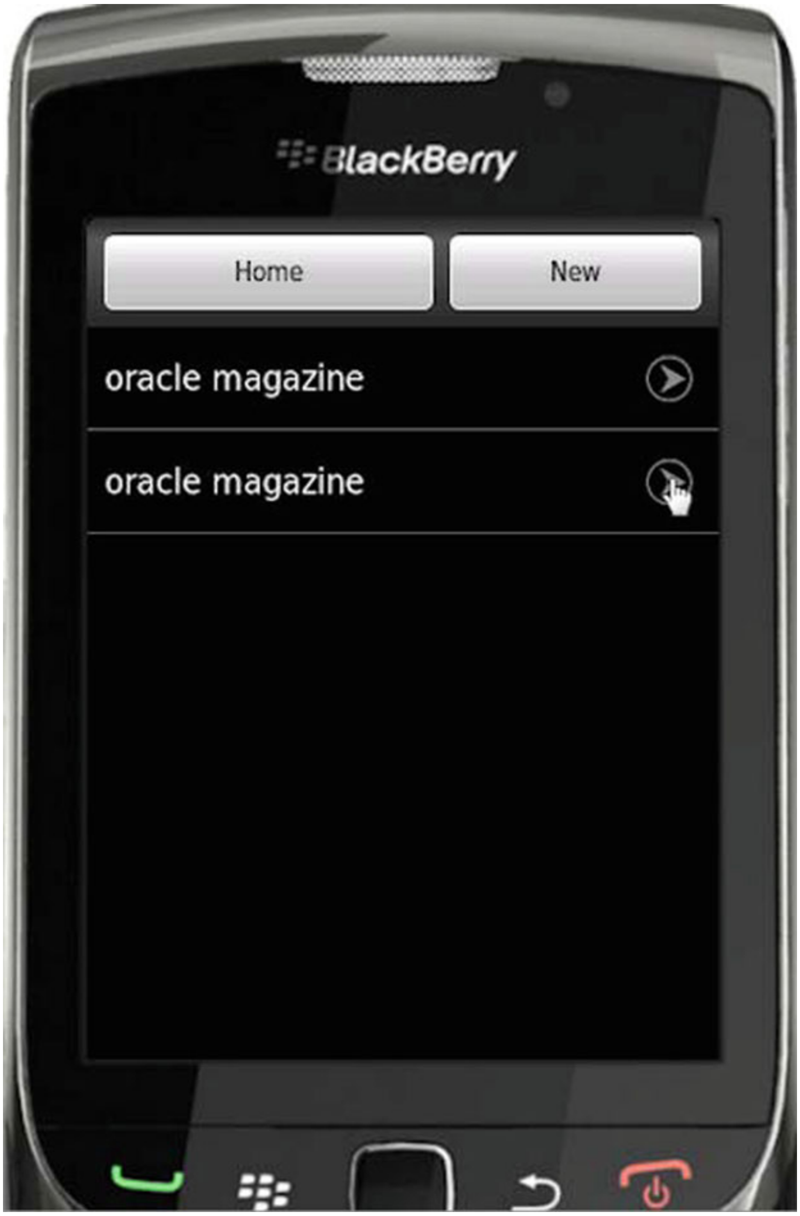
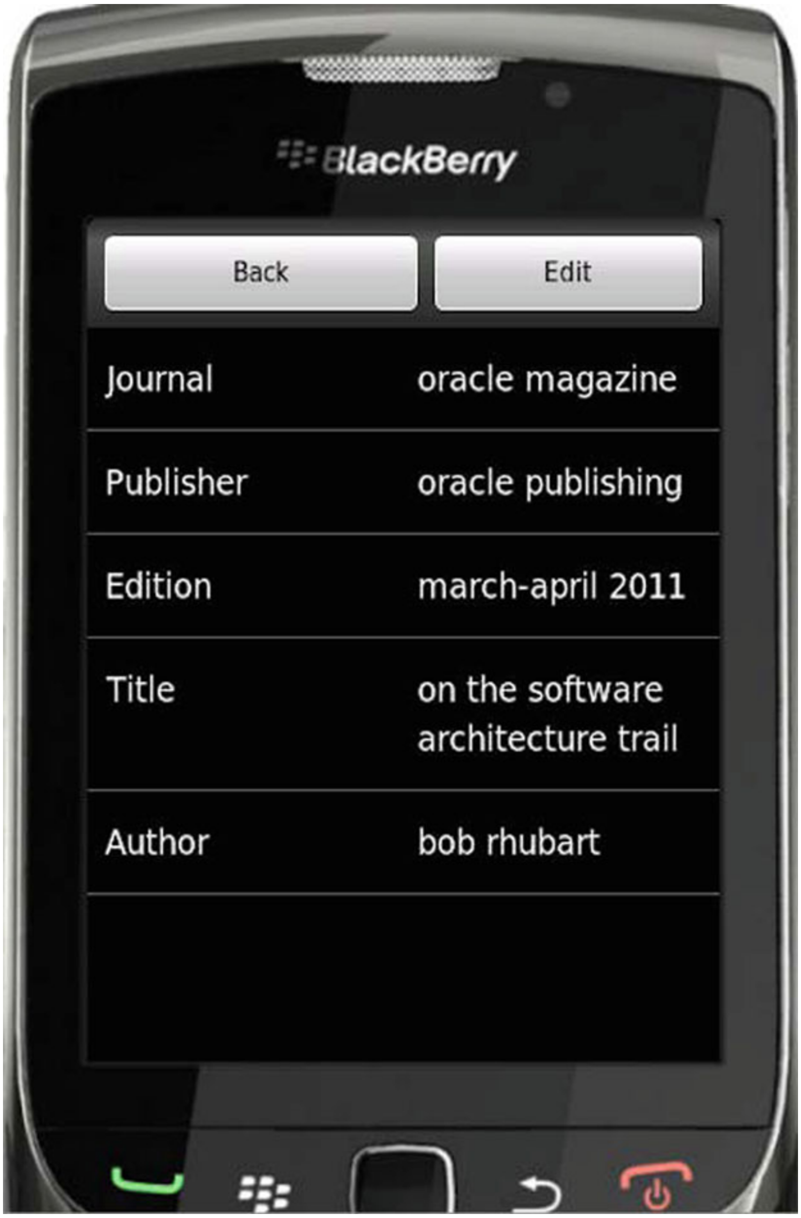


Fig. 2.12 New catalog entry

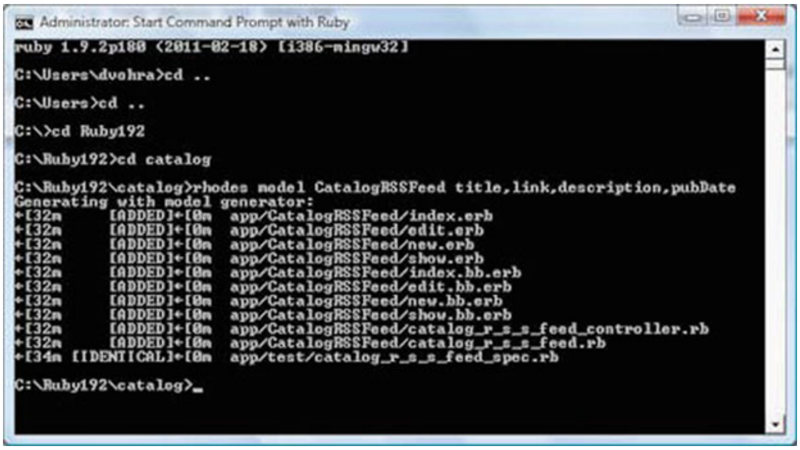
Each `item` element has sub-elements `title`, `link`, `description`, `guid`, and `pubDate`. We shall be parsing these sub-elements and displaying their values using a Rhodes model. Create a Rhodes model `CatalogRSSFeed` with attributes `title`, `link`, `description`, and `date` with the following command:



Journal	oracle magazine
Publisher	oracle publishing
Edition	march-april 2011
Title	on the software architecture trail
Author	bob rhubart

Fig. 2.13 Displaying a catalog entry

```
  C:\Ruby192\catalog>Rhodes    model    CatalogRSSFeed
title,link,description,pubDate
```



```
ruby 1.9.2p180 (2011-02-18) [i386-mingw32]

C:\Users\dvohra>cd ..

C:\Users>cd ..

C:\>cd Ruby192

C:\Ruby192>cd catalog

C:\Ruby192\catalog>rhodes model CatalogRSSFeed title,link,description,pubDate
Generating with model generator:
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/index.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/edit.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/new.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/show.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/index.bb.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/edit.bb.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/new.bb.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/show.bb.erb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/catalog_r_s_s_feed_controller.rb
←[32m     [ADDED]←[0m  app/CatalogRSSFeed/catalog_r_s_s_feed.rb
←[34m [IDENTICAL]←[0m  app/test/catalog_r_s_s_feed_spec.rb

C:\Ruby192\catalog>_
```

Fig. 2.14 Creating a RSS feed application with Rhodes

Table 2.4 GET method request parameters

Parameter	Description
:url	URL to send the requets to
:headers	Hash of headers to send with the request
:callback	Callback action to execute when the request is completed
:callback_params	Callback parameters (optional)
:authentication	Sends basic Auth header with the request (optional)
:ssl_verify_peer	Verifies SSL certificates (optional). True by default

The model, controller, and view template files get generated in the `app/CatalogRSSFeed` folder as shown in Fig. 2.14.

The model class `CatalogRSSFeed` extends the `Rhom:RhomObject` class. We won't be using the default view templates and controller actions for CRUD operations, but shall modify the controller class to get the RSS feed, parse the XML feed and display the results in the BlackBerry. For XML feed we shall require an XML parser. Rhodes includes the RhoXML parser, which is a light-weight parser and does not support some features. We shall used the ReXML parser, for which add support in the `catalog/build.yml` file

```
extensions: ["json", "rexml", "set"]
```

We shall use the `AsyncHttp` API to get the RSS feed. Use the `get(:url, :headers, :callback, :callback_params)` method for an `HTTPGET` request. The parameters for the get method are discussed in Table 2.4.

Specify the url to the Oracle Magazine RSS Feed. Send a HTTP request to the RSS feed using the `Rho::AsyncHttp.get` method.

Table 2.5 AsychHttp callback parameters

Parameter	Description
@params[“body”]	The body of the Http response
@params[“headers”]	The response headers hash
@params[“cookies”]	The server cookies
@params[“http_error”]	The HTTP error code, if response code is not 200

```
  url                                                        =
'http://www.oracle.com/ocom/groups/public/@otn/document
s/webcontent/rss-oramag-recent.xml'
  Rho::AsyncHttp.get(
          :url => url,
          :callback =>
         (url_for :action => :httpget_callback),
          :callback_param => "" )
```

The `AsychHttp` callback has the following parameters, listed in Table 2.5, available. In the callback method if status is ‘ok’ get the result of the request.

```
@@get_result = @params['body']
```

Create a `REXML::Document` object from the result using the new constructor.

```
doc = REXML::Document.new(@@get_result)
```

Using the `REXML::XPath` class iterate over the//rss//item elements in the RSS feed using the `REXML::XPath:each` method, which returns an array of nodes, and create a `CatalogRSSFeed` object corresponding to each node using the `create` method of the model.

```
  REXML::XPath.each(doc,"//rss//item/") do |e|
    CatalogRSSFeed.create(:title                     =>
e.elements['title'].text,
          :link => e.elements['link'].text,
   :description => e.elements['description'].text,
          :pubDate => e.elements['pubDate'].text)
    end
```

In the `index` action create an instance variable for all feed results using the `find(:all)` method.

```
@catalogrssfeeds = CatalogRSSFeed.find(:all)
```

In the `index.bb.erb` view template iterate over the `@catalogrssfeeds` instance variable, which contains the feed results and display the feed titles with a link to the RSS feed entry detail using the `show.bb.erb` view template.

```
<% @catalogrssfeeds.each do |obj| %>
   <td><%=  link_to  "#{obj.title}",  :action  =>  :show,
:id =>
    obj.object %></td>
  <% end %>
```

The `show.bb.erb` view template shows the RSS feed for an entry and displays the title, link, description, and publication date. The `show.bb.erb` view template is listed below.

```
  <div id="pageTitle">
   <h1><%= @catalogrssfeed.title%></h1>
  </div>
  <div id="toolbar">
  <%= link_to "Back", :action => :index %>
  <%=  link_to  "Edit",  :action  =>  :edit,  :id  =>
@catalogrssfeed.object %>
  </div>
  <div id="content">
  <table>
  <tr>
   <td class="itemLabel">Title</td>
  <td                              class="itemValue"><%=
@catalogrssfeed.title%></td>
  </tr>
  <tr>
   <td class="itemLabel">Link</td>
  <td class="itemValue"><%= @catalogrssfeed.link%></td>
  </tr>
  <tr>
   <td class="itemLabel">Description</td>
  <td                              class="itemValue"><%=
@catalogrssfeed.description%></td>
  </tr>
  <tr>
   <td class="itemLabel">PubDate</td>
  <td                              class="itemValue"><%=
@catalogrssfeed.pubDate%></td>
  </tr>
  </table>
  </div>
```

A request may be cancelled with the `Rho::AsyncHttp.cancel` method. The controller file `catalog_r_s_s_feed_controller.erb` is listed below.

```
  require 'rho/rhocontroller'
  require 'helpers/browser_helper'

  class CatalogRSSFeedController < Rho::RhoController
    include BrowserHelper
  def index
      @catalogrssfeeds = CatalogRSSFeed.find(:all)
  if @catalogrssfeeds.empty? then
    self.update
  else
  render :action => :index, :back => :exit
   end
    end

  def refresh
     CatalogRSSFeed.delete_all
     redirect :action => :update
  end

  def update
        url                                            =
'http://www.oracle.com/ocom/groups/public/@otn/document
s/webcontent/rss-oramag-recent.xml'
      Rho::AsyncHttp.get(
          :url => url,
          :callback    =>    (url_for    :action    =>
:httpget_callback),
          :callback_param => "" )
      render :action => :wait, :back => :exit
    end
    def show
      @catalogrssfeed              =              Cata-
logRSSFeed.find(@params['id'])
      if @catalogrssfeed
        render  :action  =>  :show,  :back  =>  url_for(
:action => :index )
      else
        redirect :action => :index
      end
    end
  def httpget_callback
      if @params['status'] != 'ok'
          @error_params = @params
          WebView.navigate   (   url_for   :action   =>
:show_error )
      else
```

```
          @@get_result = @params['body']
          begin
              require 'rexml/document'

   doc = REXML::Document.new(@@get_result)
   REXML::XPath.each(doc,"//rss//item/") do |e|
     CatalogRSSFeed.create(:title                    =>
e.elements['title'].text,
         :link => e.elements['link'].text,
    :description => e.elements['description'].text,
         :pubDate => e.elements['pubDate'].text)
    end
              @catalogrssfeeds         =        Cata-
logRSSFeed.find(:all)
              if @catalogrssfeeds.empty?
                WebView.navigate  (  url_for  :action =>
:show_error )
              else
                WebView.navigate  (  url_for  :action =>
:index )
              end
          rescue Exception => e
              puts "Error: #{e}"
              @@get_result = "Error: #{e}"
          end
      end
    end

    def cancel_httpcall
      Rho::AsyncHttp.cancel(   url_for(   :action   =>
:httpget_callback) )
      @@get_result  = 'Request was cancelled.'
      render :action => :index, :back => :exit
    end

   def get_res
       @@get_result
     end

   def get_error
     @@error_params
   end
   def show_error
      render  :action  =>  :error,  :back  =>  url_for(
:action => :index )
    end
    def exit
      Rho::RhoApplication.close
      System.exit
    end
  end
```

The `index.erb` view template is listed below.

```
  <div class="pageTitle">
  <h1>CatalogRSSFeeds</h1>
  </div>

  <div class="toolbar">

  <div class="regularButton">
  <a   class="button"   href="<%=   url_for   :action   =>
:refresh %>">Refresh</a>
  </div>
  </div>

  <div class="content">
  <ul>
  <% @catalogrssfeeds.each do |catalogrssfeed| %>

  <li>
  <a href="<%= url_for :action => :show, :id => cata-
logrssfeed.object %>">
  <span     class="title"><%=     catalogrssfeed.title
%></span><span class="disclosure_indicator"></span>
  </a>
  </li>

  <% end %>
  </ul>
  </div>
```

Modify the start path in the `catalog/rhoconfig.txt` file for the RSS feed application.

```
  # startup page for your application
  start_path = '/app/CatalogRSSFeed'
```

Start the BlackBerry emulator as before, with the command:

```
  C:\Ruby192\catalog>rake run:bb
```

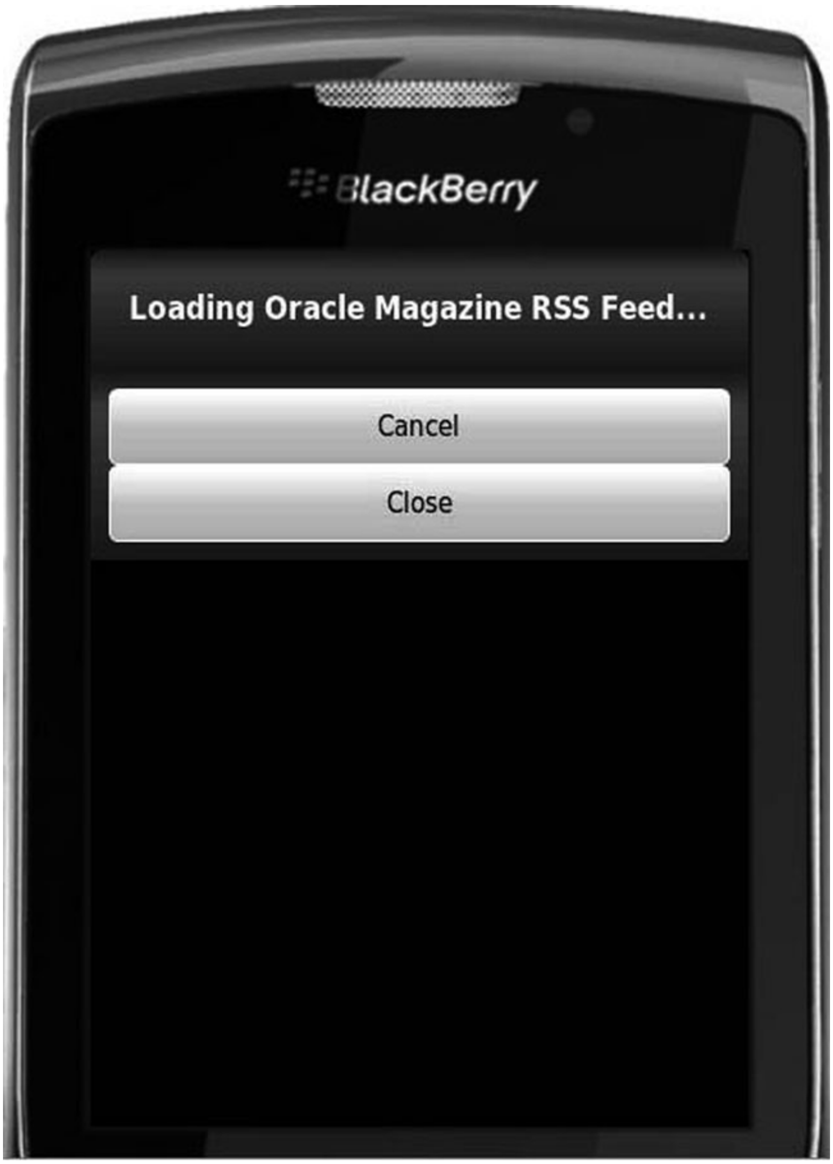
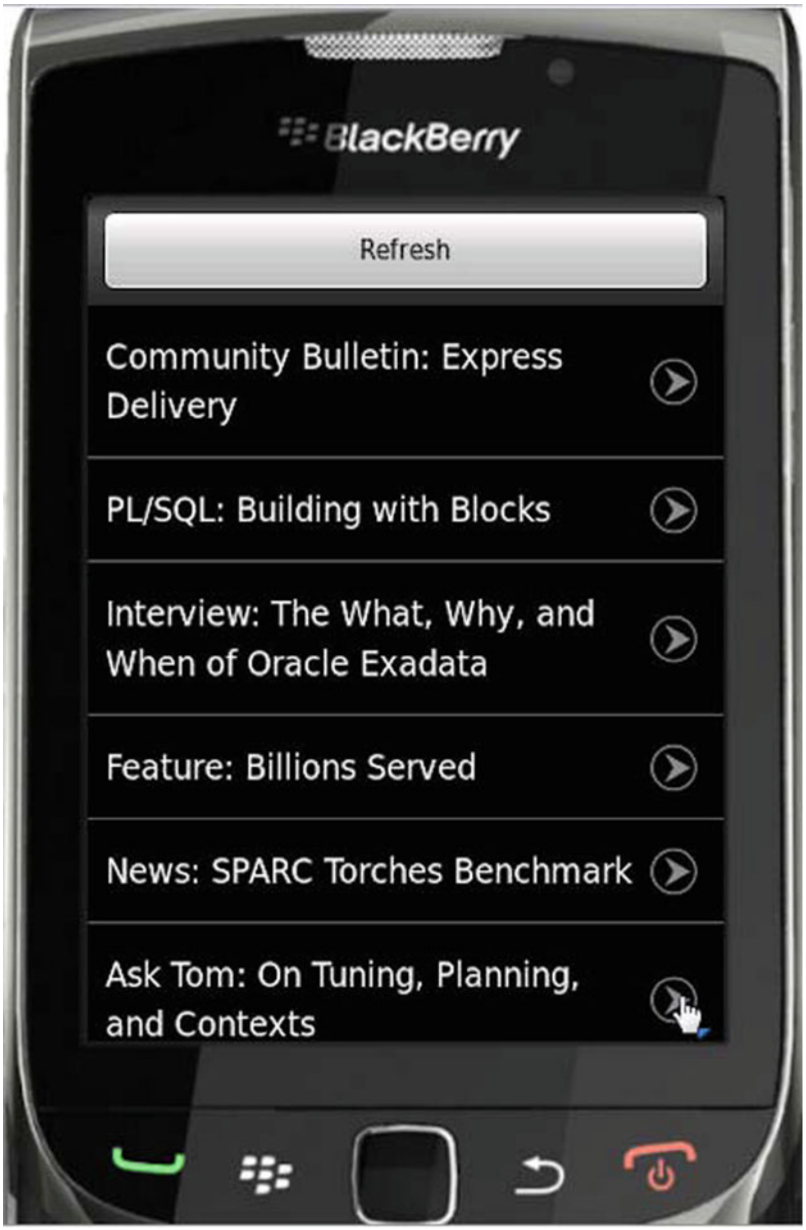


Fig. 2.15 Loading RSS feed

The Oracle Magazine RSS Feed application gets started in the BlackBerry emulator as shown in Fig. 2.15.

The Oracle magazine RSS feed gets listed in BlackBerry as shown in Fig. 2.16. Select a feed entry to display the entry detail.



Community Bulletin: Express Delivery

PL/SQL: Building with Blocks

Interview: The What, Why, and When of Oracle Exadata

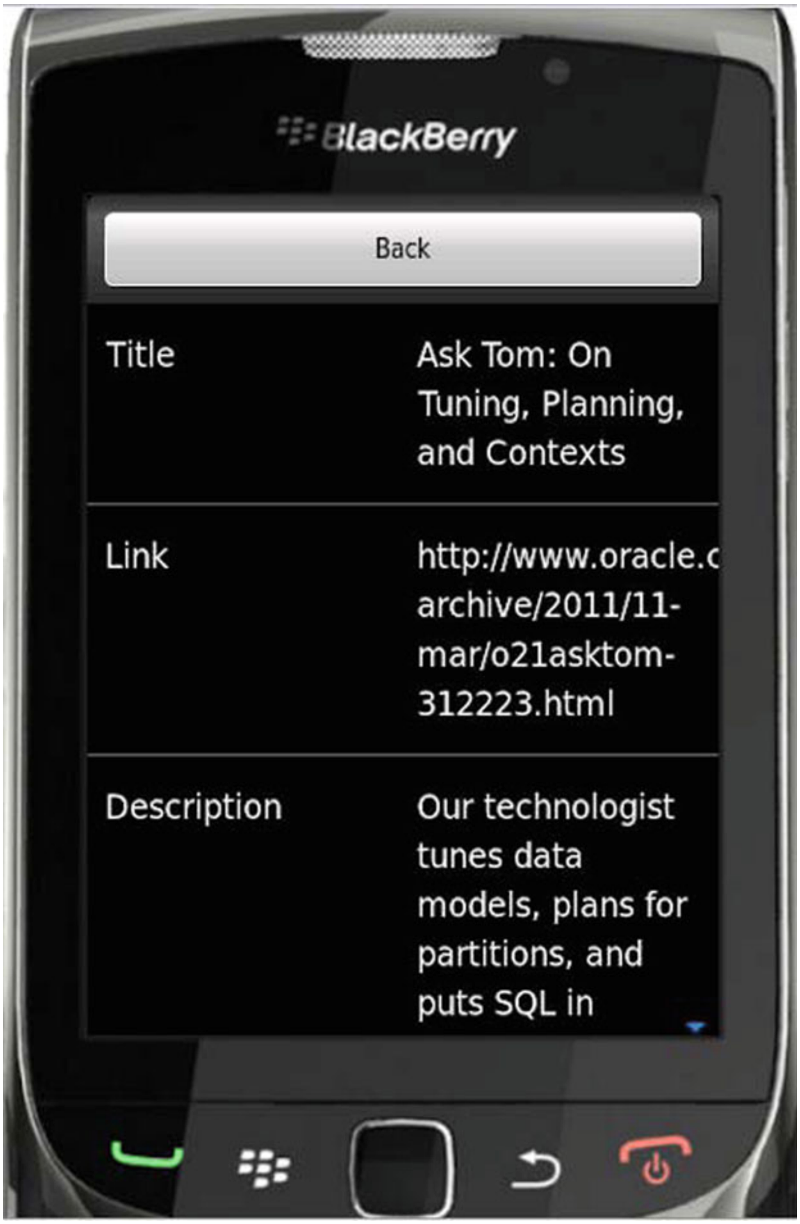
Feature: Billions Served

News: SPARC Torches Benchmark

Ask Tom: On Tuning, Planning, and Contexts

Fig. 2.16 Getting RSS feed with Rhodes on BlackBerry

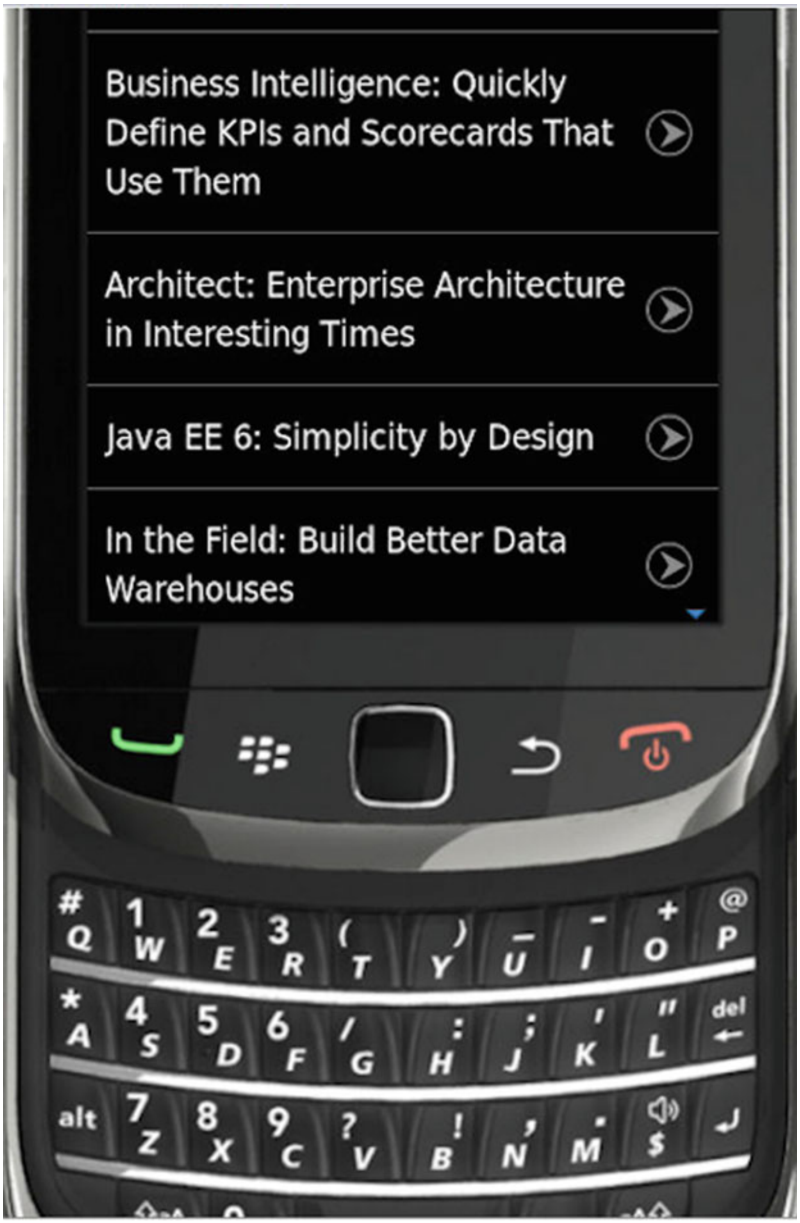
The feed entry gets displayed as shown in Fig. 2.17.
The RSS feed may be scrolled to display all the entries as shown in Fig. 2.18.



Title	Ask Tom: On Tuning, Planning, and Contexts
Link	http://www.oracle.c archive/2011/11-mar/o21asktom-312223.html
Description	Our technologist tunes data models, plans for partitions, and puts SQL in

Fig. 2.17 Displaying a RSS feed entry

In this chapter we discussed using Rhodes with BlackBerry. In the previous chapter we discussed Rhodes with Android. Rhodes with BlackBerry has the following differences from Rhodes with Android.


Business Intelligence: Quickly Define KPIs and Scorecards That Use Them

Architect: Enterprise Architecture in Interesting Times

Java EE 6: Simplicity by Design

In the Field: Build Better Data Warehouses

Fig. 2.18 Scrolled view of RSS feed on BlackBerry

- The view templates used are different. The `*.bb.erb`view templates are used with BlackBerry instead of the `*.erb` view templates with Android.
- The BlackBerry emulator is different from the Android emulator, and the command to run the emulator is different.

- The configuration with BlackBerry is different.
- The required software is different with BlackBerry. The BlackBerry JDE and DirectX SDK are used with BlackBerry instead of Eclipse and ADT with Android.