95-733 Internet Technologies Homework 3

February 13, 2014 Mini 3 S14

Pittsburgh Due Date : Thursday, February 27, 11:59 PM 2014

Adelaide Due Date: Wednesday, March 12, 11:59 PM 2014

# Ruby on Rails

One goal of this project is to expose the student to the Ruby programming language and the Rails framework. Rails is based on the model, view, controller (MVC) architecture.

Another goal is to introduce the student to OData. OData allows one to publish and consume data on the web in a machine readable format. That data will often be from a database but need not be. OData is described here:

http://www.odata.org.

The OData data source that we will use is called Northwind and its schema is here:

http://merc.tv/img/fig/Northwind_diagram.jpg.

Please look over the schema and ask yourself how you would make this data available without OData. And, would your solution easily interoperate with a diverse collection of clients and a variety of client platforms?

I recommend, but do not require, that you use Ruby and Rails command line tools rather than an IDE for this project.

To begin using Ruby on Rails on Windows, the best approach is probably to use the installation site referred to on the course schedule. See

http://railsinstaller.org/.

On Mac, I have used RVM with good success. See

http://www.rackspace.com/blog/installing-rails-on-a-mac/.

For learning Ruby, I recommend the following site:

http://www.ruby-lang.org/en/

For learning Rails, I recommend the following site:

http://rubyonrails.org/

For parsing the JSON data from Northwind, I recommend using a ruby gem described here:

http://rubygems.org/gems/json

## Part I. Stand alone Ruby programming

(0) 20 Points. Write a stand alone Ruby program that reads a JSON representation of the Northwind service document and displays a list of available collections. Document your program with plenty of comments - describing what the program is doing on each Ruby instruction.

Name this program "DisplayServiceDocument.rb".

An example execution of my solution follows:

```
ruby DisplayServiceDocument.rb
Northwind Collections from "http://services.odata.org/Northwind/Northwind.svc/?$format=json".

Categories
CustomerDemographics
Customers
Employees
Order_Details
Orders
Products
Regions
Shippers
Suppliers
```

```
Territories
Alphabetical_list_of_products
Category_Sales_for_1997
Current_Product_Lists
Customer_and_Suppliers_by_Cities
Invoices
Order_Details_Extendeds
Order_Subtotals
Orders_Qries
Product_Sales_for_1997
Products_Above_Average_Prices
Products_by_Categories
Sales_by_Categories
Sales_Totals_by_Amounts
Summary_of_Sales_by_Quarters
Summary_of_Sales_by_Years
```

(1) 20 Points. Write a stand alone Ruby program that prompts the user for a Product ID from the Northwind OData data source. The program will then fetch JSON data from Northwind and display the product name, supplier name and whether the product is discontinued or active. Document your program with plenty of comments - describing what the program is doing on each Ruby instruction.

Name this program "GetSupplierGivenID.rb".

Here is a sample execution:

```
ruby GetSupplierGivenID.rb
Enter Product ID 5
Product ID:5
Product name: Chef Anton's Gumbo Mix
Supplier ID: 2
Active/Discontinued
Chef Anton's Gumbo Mix is a discontinued product
Supplier name : New Orleans Cajun Delights
```

(2) 20 Points. Write a stand alone Ruby program that prompts the user for a product ID from the Northwind Odata data source. The program will then display a list of the order ID's associated with that product along with the total price of each order (taking into account the unit price, quantity and discount.) For example, my solution runs as follows:

```
ruby GivenProductIDListOrders.rb
Enter Product ID 3
Product ID: 3

The number of orders for product ID 3 is 12
Order ID's    Total Price after discount
10289           $240.0
10405           $400.0
10485           $144.0
10540           $600.0
10591           $140.0
10702           $60.0
10742           $200.0
10764           $180.0
10849           $490.0
10857           $300.0
11017           $250.0
11077           $40.0
```

(3) 20 Points. Write a stand alone Ruby program that prompts the user for a product ID from the Northwind Odata data source. The program will then display a list of contact names of customers that have ordered that product.

My solution runs as follows:

```
ruby GivenProductIDListCustomerContactNames.rb
Enter Product ID and I will find Customer's who ordered it 5
Product ID: 5

The number of orders for product ID 5 is 10
Roland Mendel
Paula Wilson
Pedro Afonso
Roland Mendel
```

Giovanni Rovelli
Liz Nixon
Elizabeth Brown
Yvonne Moncada
Jose Pavarotti
Ann Devon

## Part II. Ruby on Rails programming

(4) 20 Points. Develop a Rails web application that performs the same functions as the last three programs written in part 1. The design of the web user interface is in your hands. However, there must be an introductory page. From that page the user will be able to select which one of the three functions that he/she wants to perform. Name the application NorthWindWebApp These three functions are:

- Given the Product ID, find the supplier.
- Given the Product ID, list each Order ID and the price after discount.
- Given the Product ID, list the customer's Contact Name for each order.

5 of the 20 points are reserved for the student who uses git and deploys the solution to Heroku.

## Summary

Submit a single zip file with the following six names in the root directory:

0) DisplayServiceDocument.rb
1) GetSupplierGivenID.rb
2) GivenProductIDListOrders.rb
3) GivenProductIDListCustomerContactNames.rb
4) NorthWindWebApp  Note, this is a directory name.
5) Heroku.txt  Note, this is a text file containing directions on how to visit your web application on Heroku.

## Getting familiar with the OData data source

Visit the Northwind OData service document with a browser that displays XML

http://services.odata.org/Northwind/Northwind.svc/

An OData service document lists available collections. One may view the service document in XML or in JSON.

Review the service document. Note that it contains a collection called Products.

Visit the Products feed at
http://services.odata.org/Northwind/Northwind.svc/Products.

Review the structure of the first two Products. You may need to copy the document into a text editor so that newlines can be entered and so that you can see the document's structure.

Next, visit a particular product at
http://services.odata.org/Northwind/Northwind.svc/Products(2)

Verify that the feed for a particular product is also found in the Products feed.

Note the regular use of URI's. See the following page for more on URI's with OData.
http://www.odata.org/documentation/odata-v2-documentation/uri-conventions/

Visit the service document again, this time retrieve the JSON format.
http://services.odata.org/Northwind/Northwind.svc/?$format=json

Visit the Products feed at
http://services.odata.org/Northwind/Northwind.svc/Products?$format=json

Visit the second Product with
http://services.odata.org/Northwind/Northwind.svc/Products(2)?$format=json

For a discussion of the JSON representation, see
http://www.odata.org/documentation/odata-v2-documentation/json-format/

OData is largely based on the REST design approach. The client is not

```
 only allowed to perform HTTP GET operations but is also allowed to perform
 PUTS and DELETES and so on. In this homework, we will only be performing
 GETS. But the student should be aware that OData is not just about consuming
 data. Writing and modifying data is also provided in many OData environments.

 This is a sample Ruby program that demonstrates handling JSON and how to access the
 Nortwind OData data source.
=============================================================================

# This programs demonstrates how Ruby may be used to parse JSON strings.
# Ruby represents the JSON object as a hash.

require 'net/http'
require 'json'

# Go out to the internet and collect some JSON

# Set up the URL

url = "http://services.odata.org/Northwind/Northwind.svc/Products(2)?$format=json"

# Make an HTTP request and place the result in jsonStr

jsonStr = Net::HTTP.get_response(URI.parse(url))

data = jsonStr.body

jsonHash = JSON.parse(data)

# See if the product is discontinued

if (jsonHash["Discontinued"])
      print jsonHash["ProductName"].to_s + " is a discontinued product"
   else
      print jsonHash["ProductName"].to_s + " is an active product"
end



Here are some tested instructions on getting started with Rails 3.x.x
=====================================================================

# Build a new directory called Rails_Examples.
# Inside the directory Rails_Examples, do the
# following:

# Check the rails version.
$rails -v
Rails 3.2.2

# Create a new rails application. In this case we will use the sqlite3 database.
# Database use is optional.

$rails new CoolRSSJobApp -d sqlite3
$cd CoolRSSJobApp

# Run bundle install to satisfy all dependencies
$bundle install

# Create the model (app/models/job.rb) , migration script,
# database table maintenance controller (app/controllers/jobs_controller.rb).
# Under app/views/jobs, this command creates CRUD views .html.erb files.

$rails generate scaffold Job source:string url:string

# Create the actual database.
$rake db:migrate

# Run the server and visit the CRUD application.

$rails server
=> Booting WEBrick
=> Rails 3.2.2 application starting in development on http://0.0.0.0:3000
=> Call with -d to detach
=> Ctrl-C to shutdown server

# The controller is called jobs_controller.rb. There
# is a method in the controller
# called index. There is an index.html.erb in
```

```
# app/views/jobs.

# Visit the index method with a browser at :
http://localhost:3000/jobs

# You can perform CRUD operations on the new database.

# Experiment! Study the directories, controller and views.

# Add a new controller called main_controller.rb.

$rails generate controller main

# Add a method to main controller.

class MainController < ApplicationController

   def welcome
     @num_sources = Job.count
   end

end

# edit your config/routes.rb file
# add this line before 'end' in the file

match '/main/welcome' => 'main#welcome'

# Create a partial HTML file called welcome.html.erb.
# This files needs to reside in the
# CoolRSSJobApp/app/views/main directory.
# Ruby automatically includes the remaining
# HTML tags. We only need to provide this partial
# document.

<h1>Welcome to the Job Seeker Application</h1>
<p></p>
We currently have <%= @num_sources %> sources.

# Notice how the view is able to extract
# information from the instance variables in
# the controller.

# Shut down and restart the web application.

# Vist with:
http://localhost:3000/main/welcome


# Note the controller file is named main_controller.rb.
# The method within
# the controller is named welcome. The file named welcome.html.erb
# is in app/views/main. The HTML file is delivered to the user
# after the welcome method is run.

# If you want to visit with

http://localhost:3000/main/

# then you need to add an index method to the main controller
# and an index.html.erb to the app/views/main directory.
```