Creating a Web Service in Ruby

1. Installing Ruby and Gems

1. Install Ruby

If you don't have Ruby installed, goto:

O:\IT Solutions\Resource\TEST1\Ruby Install Stuff

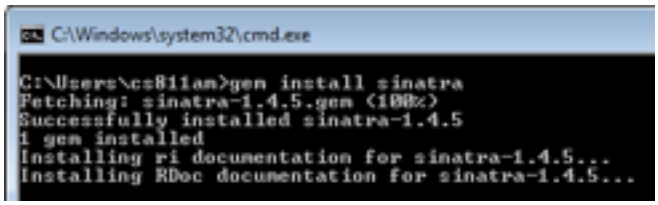
And run **Ruby Install (win 7).bat**

2. Install Gems

We will be building a web service using the Sinatra function library and will also be connecting to MongoDB, displaying the results in JSON. Open a comment prompt and run the commend:

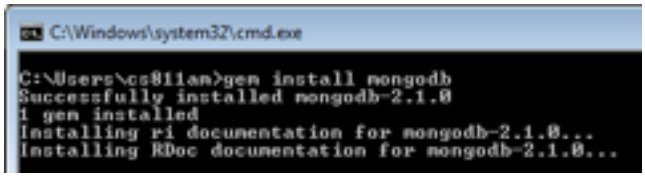
To install Sinatra, open a comment prompt and run:

Gem install sinatra



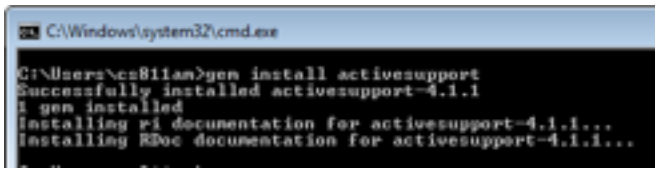
Now to install MongoDB run:

Gem install mongodb



Now to install ActiveSupport run:

Gem install activesupport



What have I done?
You have installed Ruby on your PC (If it wasn't there already), and you have downloaded the function libraries Sinatra and MongoDB, they will now be available to any Ruby scripts. You have also installed ActiveSupport, which is a library which allows you to convert an object to a json output.

2. Creating a Web Service

2.1 Creating a Hello World Web Service

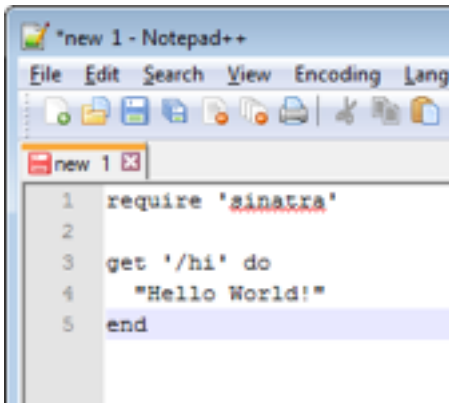
Create a folder on your C Drive called **RubyRestService**.

Open Notepad++ and type the following:

```
require 'sinatra'

get '/hi' do
  "Hello World!"
end
```

This adds the Sinatra functions to the ruby script.

Your code should look like:

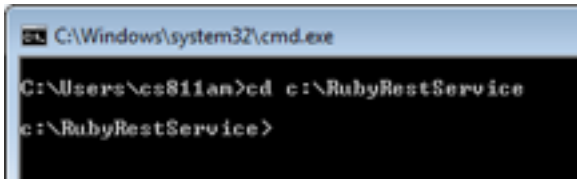


Save this file now as **getProperty.rb** in **c:\RubyRestService**.

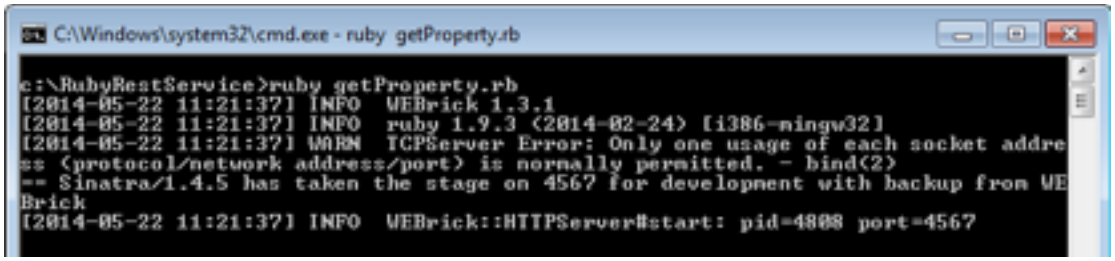
2.2 Running the Hello World Web Service

Open a command prompt and navigate to **c:\RubyRestService** by typing:

```
cd c:\RubyRestService
```



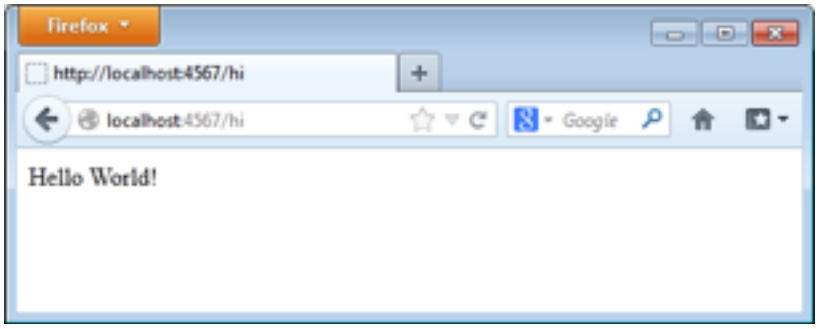
Now to run the web service, type **ruby getProperty.rb**



What have I done?
You have created a Ruby web service (using the Sinatra function library) and have run the ruby script. When running the ruby script (ruby getProperty.rb) it starts it own webserver and deploys the code onto that. The webserver is running on the port 4567.

Open a web browser and navigate to:

http://localhost:4567/hi



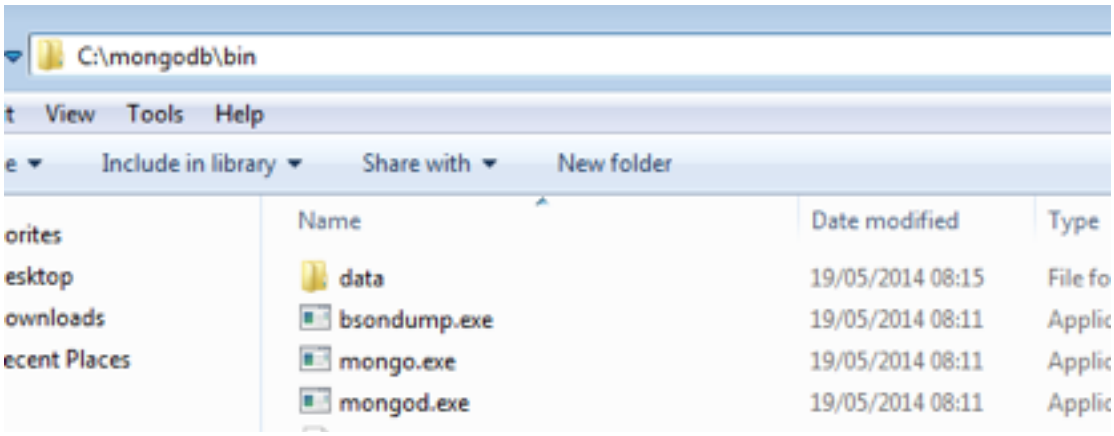
How to stop the Ruby Script?
The Ruby script is running a webserver, it won't stop until you ask it to. Press Ctrl and the C keys on the keyboard at the same time and it will stop the web server.

3. MongoDB

3.1 Running MongoDB

Navigate to where you have installed MongoDB. (e.g. C:\mongodb).

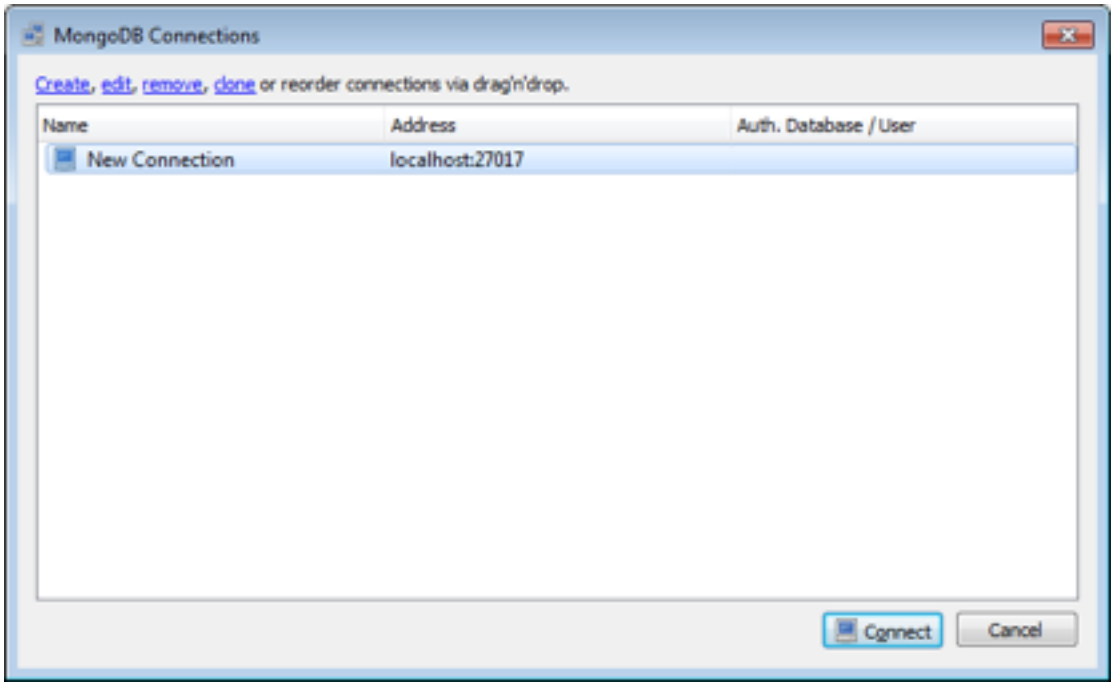
Within the Bin folder, run the mongod.exe file.



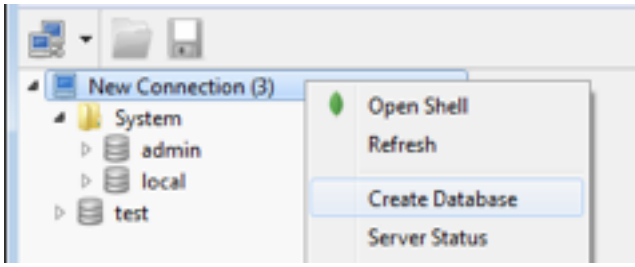
This will load a command prompt window, if you close this window you will close your mongoDB server. It is best to minimize this window. If you do accidently close it, just start mongod.exe again.

3.2 Creating a Collection (aka Table)

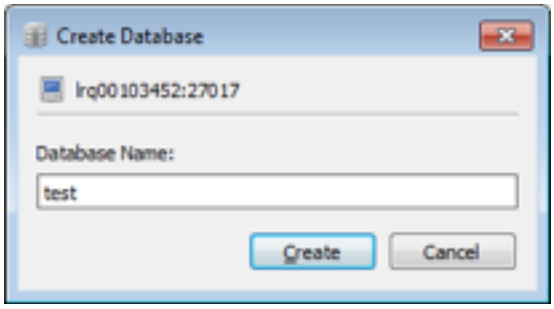
Open Robomongo.



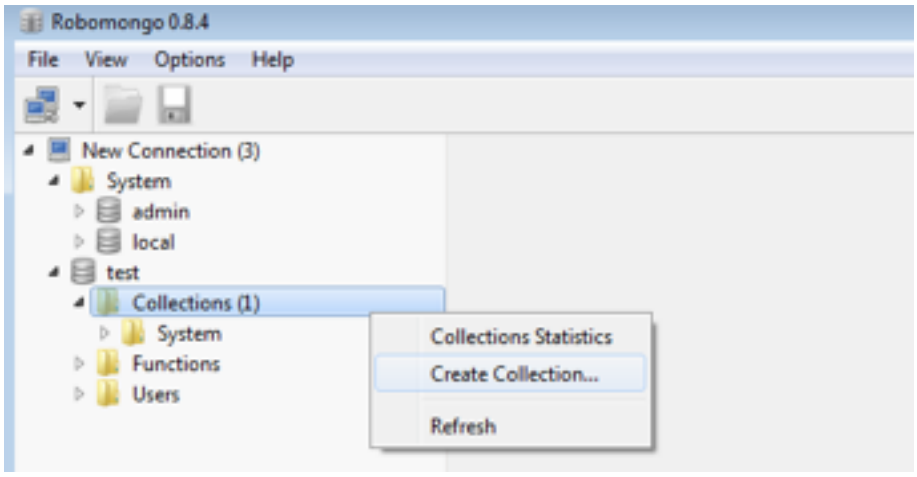
If you don't have any connections listed. Create one for localhost.
Right click on New Connection (3) and click **Create Database**



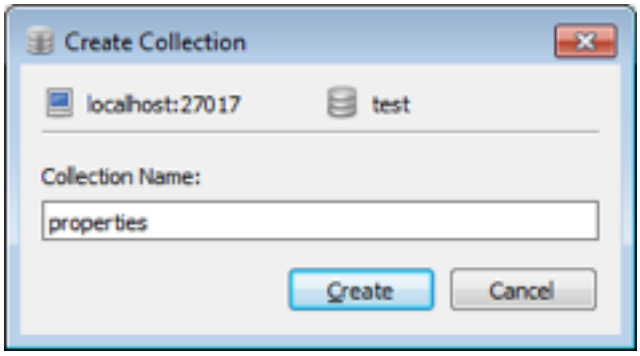
Give the database a name of **test**.



Expand test on the right side. Right click on Collections and choose **Create Collection**.

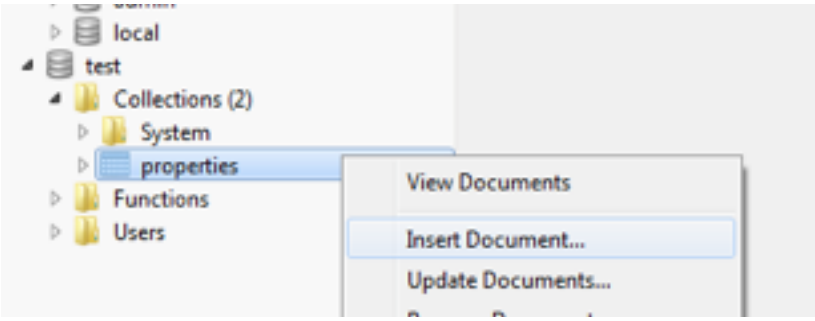


Give the collection a name of **properties**. Click Create

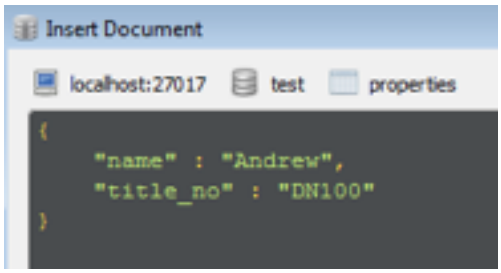


3.3 Creating a Document (aka Row)

Right click on properties and click **Insert Document**.

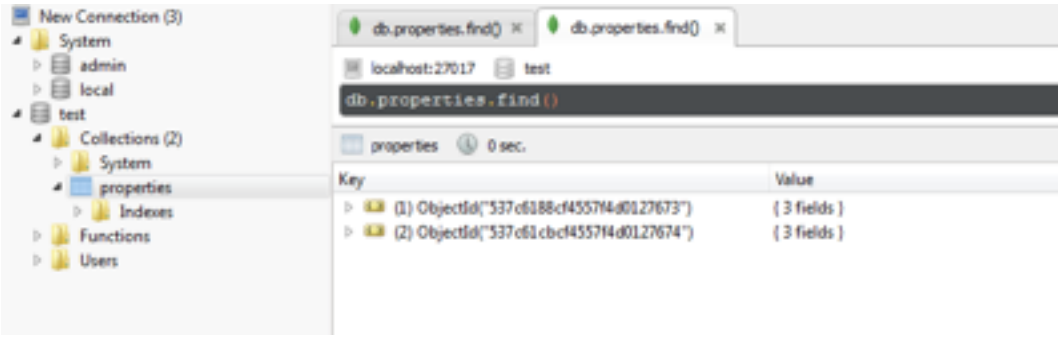


MongoDB uses JSON syntax. We will create a new document which contains two items of data, a name and a title number. Click Save.



Repeat this now to create additional data.

Now double click on properties.



If you expend out the objects you can see the data that was inserted:

Key	Value	Type
(1) ObjectId("537c6188cf4557f4d0127673")	{ 3 fields }	Object
_id	ObjectId("537c6188cf4557f4d0127673")	ObjectId
name	Andrew	String
title_no	DN100	String
(2) ObjectId("537c61cbcf4557f4d0127674")	{ 3 fields }	Object

What have I just done?
You have a mongodb database running which now has a collection (table) called properties, you have inserted several documents (rows) and you have browsed the data in the collection.

4. Connecting Ruby to MongoDB

4.1. Creating Folder Structure

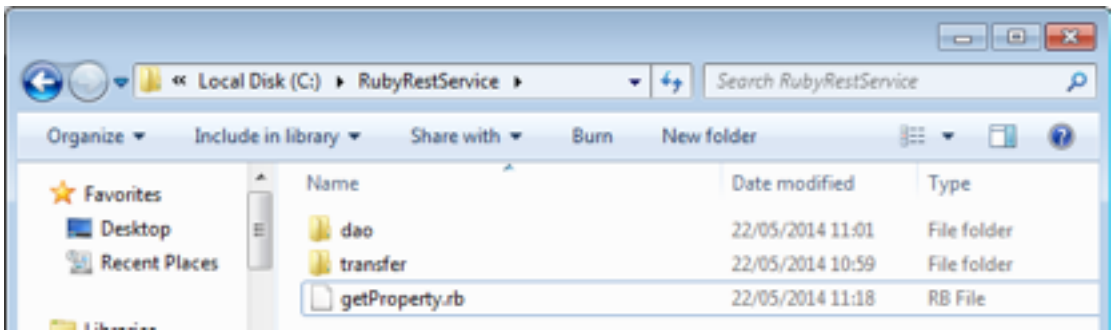
It isn't good standard to include all your code in a single file, this prevents you from reusing it in other scripts and also can become unmanageable.

We want to create two folders to help organise our code:

- A Transfer folder which will include our data classes.
- A Data Access Object folder which will perform our database query

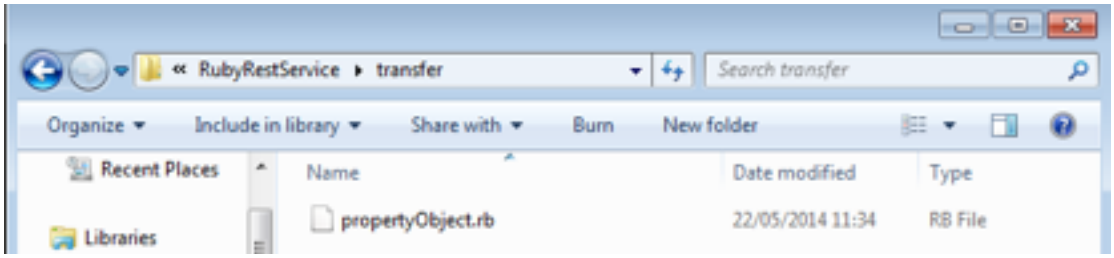
In C:\RubyRestService create the following folders:

- transfer
- dao



4.2. Create the Property Transfer Object

Open notepad++ and create a new file, save this file as **propertyObject.rb** in the **C:\RubyRestService\transfer** folder.

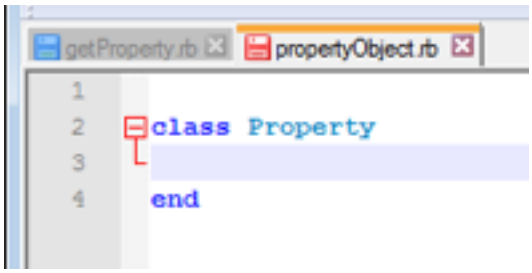


You are going to create a class called **Property**.

Type the following code:

```
class Property

end
```

So your code now looks like:



We want the Property class to match the data structure you created in the MongoDB.

To do this you need to create get and set classes.

After **class Property**, add the following code:

```
def setName(name)
  @name = name
end
def getName()
  @name
end

def setTitleNo(titleNo)
  @titleNo = titleNo
end
def getTitleNo()
  @titleNo
end
```

Sets the object's name variable.

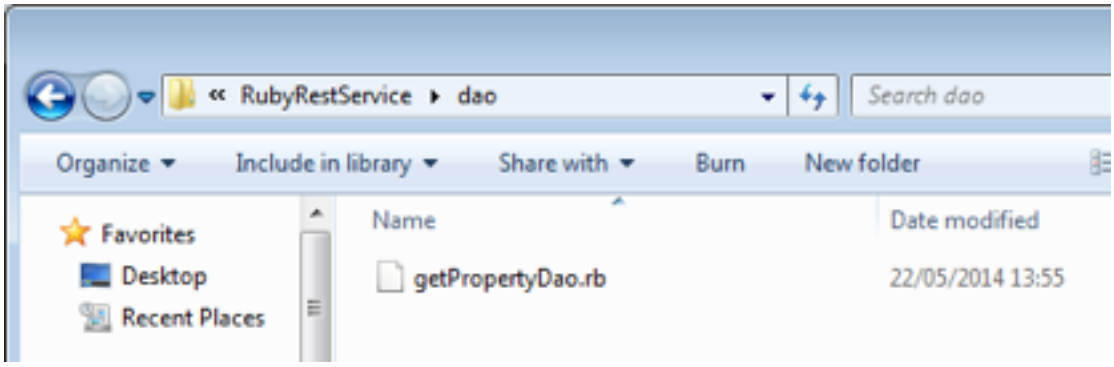
So your code now looks like:

```
class Property

  def setName(name)
    @name = name
  end
  def getName()
    @name
  end

  def setTitleNo(titleNo)
    @titleNo = titleNo
  end
  def getTitleNo()
    @titleNo
  end

end
```

What have I just done?
You have created a Ruby class which contains 4 functions, two which set variables (name and titleNo) and two which returns the values of those variables.

4.3. Create the Data Access Object

Open notepad++ and create a new file, save this file as **getPropertyDao.rb** in the **C:\RubyRestService\dao** folder.



You are going to create a class called GetProperty.

Type the following code:

```
class GetProperty

end
```

So your code now looks like:

```
class GetProperty

end
```

We are going to connect this class to mongo db, so we need to include the mongodb function library:

Above **class GetProperty**, add:

```
require 'mongo'

include Mongo
```

Your code should now looks like:

```
require 'mongo'

include Mongo

class GetProperty

end
```

Also we want to use the Property Object you created in the transfer folder.

Ruby doesn't know where this file is, so you need to specify the relative path to it.

Below **require 'mongo'** add:

```
require_relative('../transfer/propertyObject.rb');
```

Your code should now looks like:

```
getProperty.rb   propertyObject.rb   getPropertyDao.rb
1  require 'mongo'
2  require_relative('../transfer/propertyObject.rb');
3
4  include Mongo
5
6  class GetProperty
7
8  end
9
```

Inside the class GetProperty class, we want to define a **getPropertyByName** function which will perform the mongo query searching for title numbers by the name specified in the web service.

Below **class GetProperty** add:

```
def getPropertyByName(name)

end
```

The getPropertyByName function requires a value to be submitted when the function is called. This value then is stored in the variable called **name**

This function has an input of a name, this means when you call this function elsewhere, you need to specify 1 input.

Your code should now looks like:

```
getProperty.rb   propertyObject.rb   getPropertyDao.rb
1  require 'mongo'
2  require_relative('../transfer/propertyObject.rb');
3
4  include Mongo
5
6  class GetProperty
7
8      def getPropertyByName(name)
9
10     end
11
12 end
```

Now we want to create an array that will be returned from this function:

Below **def getPropertyByName(name)** add:

Creates a variable called items which is an

```
        items = Array.new();

        # Comment: Add Mongo DB Connect

        return items
```

Your code should now looks like:

```
getProperty.rb   propertyObject.rb   getPropertyDao.rb
1   require 'mongo'
2   require_relative('../transfer/propertyObject.rb');
3
4   include Mongo
5
6   class GetProperty
7
8       def getPropertyByName(name)
9
10          items = Array.new();
11
12          # Comment: Add Mongo DB Connect
13
14          return items
15
16      end
17
18  end
```

Now we want to add the connection to the mongodb.

Replace **# Comment: Add Mongo DB Connect** with:

```
        mongo_db = MongoClient.new("localhost", 27017).db("t
localhost
        coll = mongo_db.collection("properties")

        # Comment: Add MongoDB Query
```

Connects to the mongo database on localhost. And the database called test

Your code should now looks like:

```
getProperty.rb   propertyObject.rb   getPropertyDao.rb
1  require 'mongo'
2  require_relative('../transfer/propertyObject.rb');
3
4  include Mongo
5
6  class GetProperty
7
8      def getPropertyByName(name)
9
10         items = Array.new();
11
12         mongo_db = MongoClient.new("localhost", 27017).db("test")
13
14         coll = mongo_db.collection("properties")
15
16         # Comment: Add MongoDB Query
17
18
19         return items
20
21     end
22
23 end
```

Now we want to query mongodb and create a Property object for each document (row) returned. Then we will add this object to an array to make an array of objects.

Replace **# Comment: Add MongoDB Query** with:

```
        coll.find({"name" => name}).each { |row|

            data = Property.new();

            data.setName(row['name']);
            data.setTitleNo(row['title_no']);

            items << data

        }
```

Performs a query on the properties collection where the name field in mongo matches the content of the name

Your code should now looks like:

getProperty.rb | propertyObject.rb | getPropertyDao.rb

```
1   require 'mongo'
2   require_relative('../transfer/propertyObject.rb');
3
4   include Mongo
5
6   class GetProperty
7
8       def getPropertyByName(name)
9
10          items = Array.new();
11
12          mongo_db = MongoClient.new("localhost", 27017).db("test")
13
14          coll = mongo_db.collection("properties")
15
16          coll.find({"name" => name}).each { |row|
17
18              data = Property.new();
19
20              data.setName(row['name']);
21              data.setTitleNo(row['title_no']);
22
23              items << data
24
25          }
26
27          return items
28
29      end
```

What have I just done?
You have created a class that will connect to MongoDB. A query will be performed to return all data matches the name entered. The values returned from MongoDB query are written to an instance of the Property object, this instance is then added to an array which is then returned from this function.

4.4. Define the getPropertyByName web service.

You have created the Property object, which has the properties of name and titleNo. You have then created a getPropertyByName() function (in the getProperty) class, which connects to mongodb and creates instances of the Property object. We now want to create a webservice which will call this function and display the results in JSON.

Reopen the getProperty.rb file and at the end of the file add the following code at the end of the file:

```
get '/getpropertybyname/:name' do

end
```

Your code should now looks like:

```
getProperty.rb   propertyObject.rb   getPropertyDao.rb
1  require 'sinatra'
2
3  get '/hi' do
4    "Hello World!"
5  end
6
7  get '/getpropertybyname/:name' do
8
9  end
```

Parameters in the URL
This web service will have a variable of name. You can access this variable by using params[‘name’] in the ruby code. This allows your URL to be dynamic and return different data.

We want to output the results in JSON, we need to add the JSON function library called activesupport. Below **require 'sinatra'** add:

```
require 'active_support/all'
```

Your code should now looks like:

```
getProperty.rb   propertyObject.rb   getPropertyDao.rb
1  require 'sinatra'
2  require 'active_support/all'
3
4  get '/hi' do
5    "Hello World!"
6  end
7
8  get '/getpropertybyname/:name' do
9
10 end
```

We also need to call the getPropertyByName() function which is stored in the getProperty.rb file in the dao folder.

Below **require 'active_support/all'** add:

```
require_relative('./dao/getPropertyDao.rb');
```

Your code should now looks like:

```
getProperty.rb   propertyObject.rb   getPropertyDao.rb
1   require 'sinatra'
2   require 'active_support/all'
3
4   require_relative('./dao/getPropertyDao.rb');
5
6   get '/hi' do
7     "Hello World!"
8   end
9
10  get '/getpropertybyname/:name' do
11
12  end
```

Now we want to call the getPropertyByName() function in the GetProperty object, feeding in the name which was specified in the URL. The getPropertyByName() will return an array of objects, we then want to convert that to json and display the it out on the screen.

Below **get '/getpropertybyname/:name' do** add:

Defines the output type of the web service.

```
content_type :json

getPropertyDao  = GetProperty.new();

list = getPropertyDao.getPropertyByName(params['name']);

list.to_json
```

Your code should now looks like:

```
getProperty.rb   propertyObject.rb   getPropertyDao.rb
1   require 'sinatra'
2   require 'active_support/all'
3
4   require_relative('./dao/getPropertyDao.rb');
5
6   get '/hi' do
7     "Hello World!"
8   end
9
10  get '/getpropertybyname/:name' do
11      content_type :json
12
13      getPropertyDao  = GetProperty.new();
14
15      list = getPropertyDao.getPropertyByName(params['name']);
16
17      list.to_json
18
19  end
```

4.5. Running the web service.

Open a command prompt and navigate to **c:\RubyRestService** by typing:

cd c:\RubyRestService

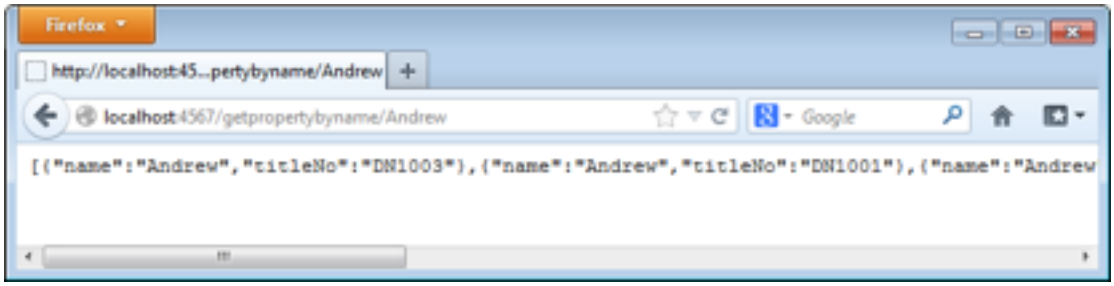
```
C:\Windows\system32\cmd.exe
C:\Users\cs811an>cd c:\RubyRestService
c:\RubyRestService>
```

Now to run the web service, type **ruby getProperty.rb**

```
C:\Windows\system32\cmd.exe - ruby getProperty.rb
c:\RubyRestService>ruby getProperty.rb
      *** Notice: The native BSON extension was not loaded. ***

      For optimal performance, use of the BSON extension is recommended.

      To enable the extension make sure ENV['BSON_EXT_DISABLED'] is not set
      and run the following command:

        gem install bson_ext

      If you continue to receive this message after installing, make sure that
      the bson_ext gem is in your load path.

[2014-05-22 13:36:52] INFO  WEBrick 1.3.1
[2014-05-22 13:36:52] INFO  ruby 1.9.3 (2014-02-24) [i386-mingw32]
[2014-05-22 13:36:52] WARN  TCPServer Error: Only one usage of each socket addre
ss (protocol/network address/port) is normally permitted. - bind(2)
== Sinatra/1.4.5 has taken the stage on 4567 for development with backup from WE
Brick
[2014-05-22 13:36:52] INFO  WEBrick::HTTPServer#start: pid=348 port=4567
127.0.0.1 - - [22/May/2014 13:36:52] "GET /getPropertyByName/Andrew HTTP/1.1" 20
0 217 0.0468
LRQ00103466.diti.lr.net - - [22/May/2014:13:36:52 GMT Daylight Time] "GET /getPr
opertyByName/Andrew HTTP/1.1" 200 217
- -> /getPropertyByName/Andrew
```

Open a web browser and navigate to:

http://localhost:4567/getpropertybyname/Andrew



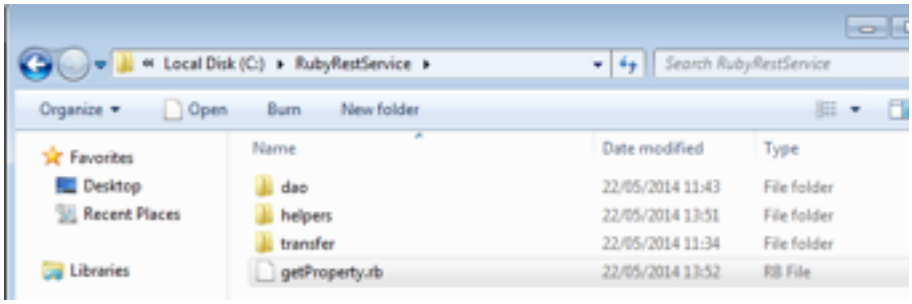
How to stop the Ruby Script?
The Ruby script is running a webserver, it won't stop until you ask it to. Press Ctrl and the C keys on the keyboard at the same time and it will stop the web server.

5. Outputting in both XML and JSON

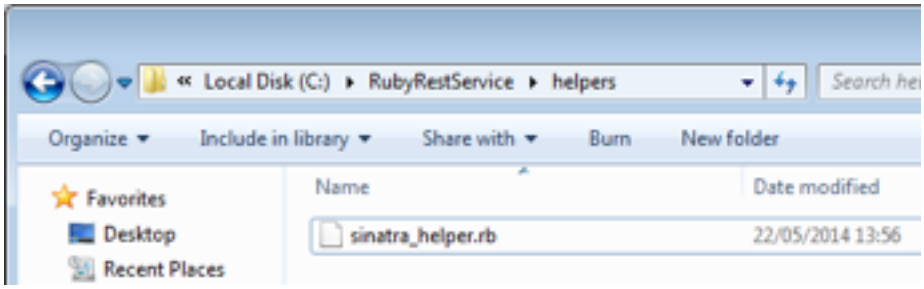
5.1 Creating a helper file.

The web service you just created only outputs in JSON. You can create a function which will support outputting the result in both XML and JSON.

Create a new folder in **C:\RubyRestService** called **helpers**.



Using notepad++ create a new file. Save the file in **C:\RubyRestService \helpers** folder with a name of **sinatra_helper.rb**.



Add the following to the file:

```
module Sinatra
	module DynFormat
		def display(data,format=(params[:format] || 'json'))
			jsonOutput = data.to_json
			case format
			when 'xml'
				content_type 'text/xml'
				parsed = JSON.parse(jsonOutput)
				return parsed.to_xml
			else
				content_type 'application/json'
				return jsonOutput
			end
		end
	end

	 helpers DynFormat
end
```

What does that code do?
If you append ?format=xml to the URL, it will convert the object to xml, when not specifying the format as being XML, it will default to JSON. This is based on code from:
http://stackoverflow.com/questions/2897929/what-is-the-best-way-to-handle-dynamic-content-type-in-sinatra

5.2 Amending the getPropertyByName() function

You have now created your helper file which has the display() function, which will allow the object to be displayed in both JSON and XML

Reopen the getProperty.rb file.

Below **require 'active_support/all'** add:

```
require_relative('./helpers/sinatra_helper.rb');
```

Your code should now looks like:

```ruby
require 'sinatra'
require 'active_support/all'

require_relative('./helpers/sinatra_helper.rb');
require_relative('./dao/getPropertyDao.rb');

get '/hi' do
  "Hello World!"
end

get '/getpropertybyname/:name' do
    content_type :json

    getPropertyDao  = GetProperty.new();

    list = getPropertyDao.getPropertyByName(params['name']);

    list.to_json

end
```

Now amend the **get '/getPropertyByName/:name'** do function.

Delete the **content_type :json** line.

And amend **list.to_json** to:

display(list)

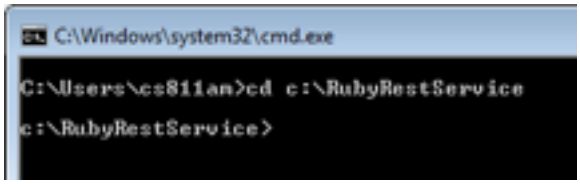
Your code should now looks like:

```ruby
require 'sinatra'
require 'active_support/all'

require_relative('./helpers/sinatra_helper.rb');
require_relative('./dao/getPropertyDao.rb');

get '/hi' do
  "Hello World!"
end

get '/getpropertybyname/:name' do

    getPropertyDao  = GetProperty.new();

    list = getPropertyDao.getPropertyByName(params['name']);

    display(list)

end
```

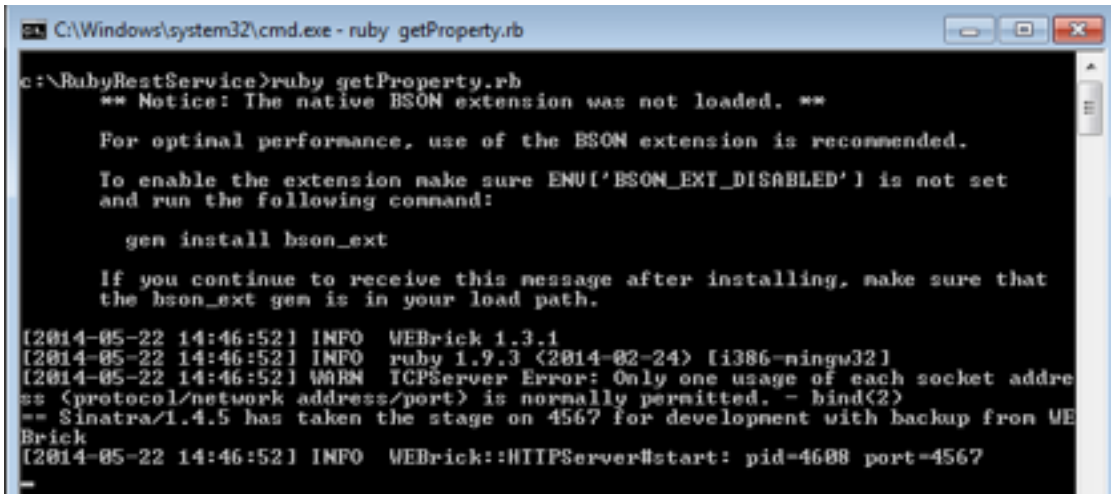
5.3 Running the web service.

Open a command prompt and navigate to **c:\RubyRestService** by typing:

cd c:\RubyRestService

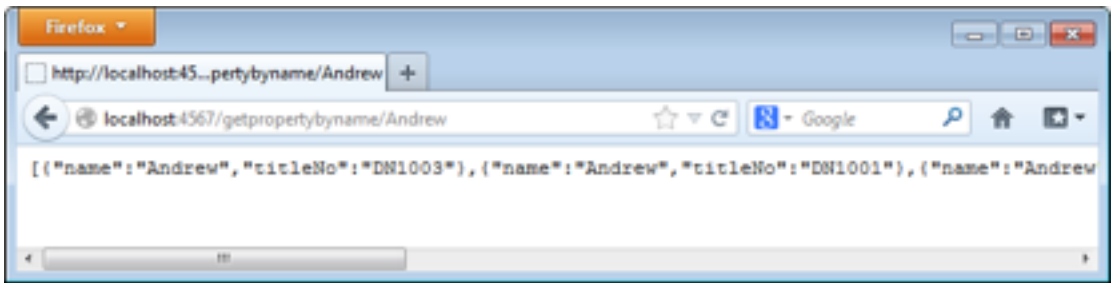


Now to run the web service, type **ruby getProperty.rb**



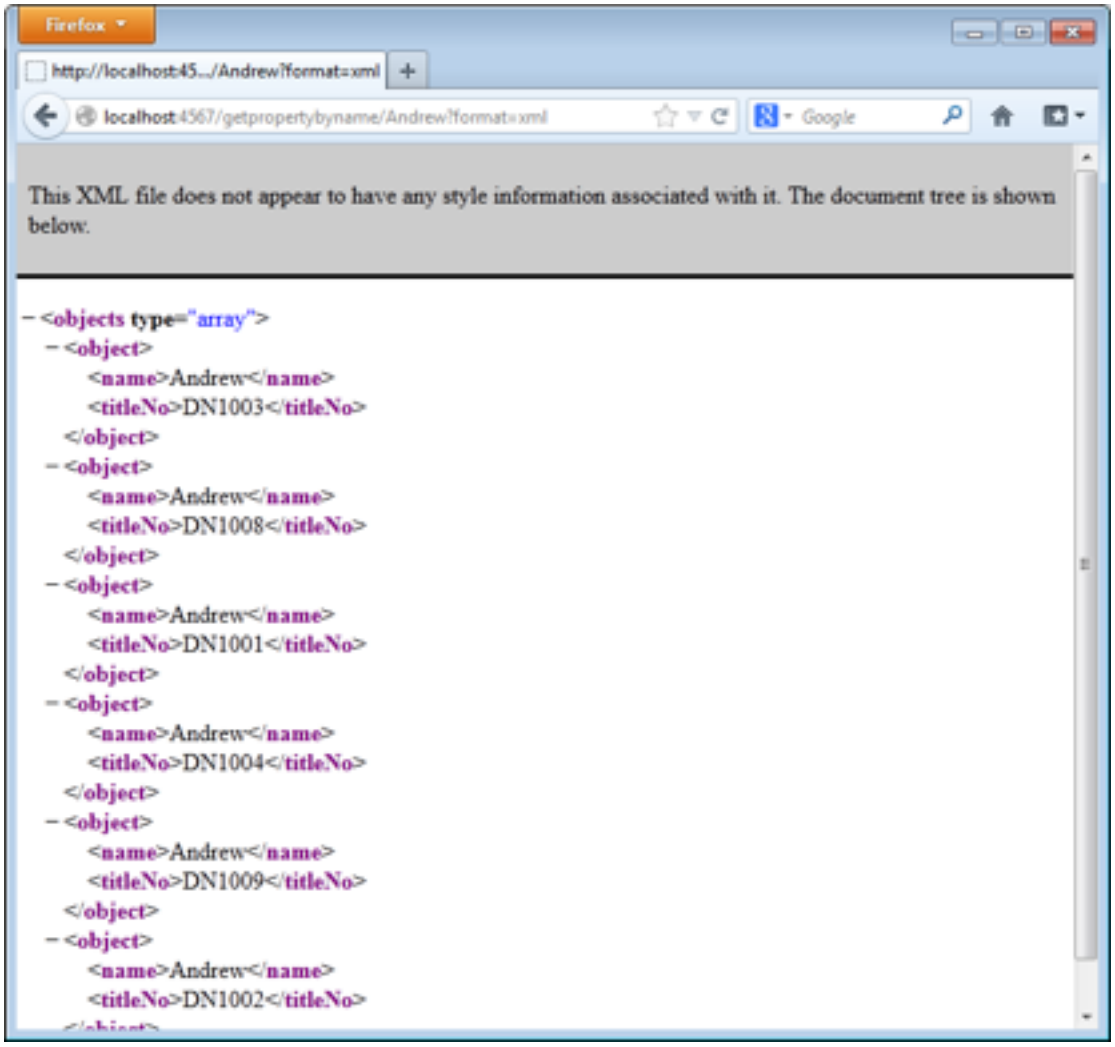
Open a web browser and navigate to:

http://localhost:4567/getpropertybyname/Andrew



And now navigate to:

http://localhost:4567/getPropertyByName/Andrew?format=xml



How to stop the Ruby Script?
The Ruby script is running a webserver, it won't stop until you ask it to. Press Ctrl and the C keys on the keyboard at the same time and it will stop the web server.

6. Alternative Approach to Creating a Web Service

6.1 Creating a Simple web Service

The guide so far has forced you to map to map the results returned from MongoDB to an object, that object is then converted to Json. This is the Java style approach, but in Ruby can also query mongoDB and return the data without having to map it in to an object.

Using notepad++ create a new file. Save the file in **C:\RubyRestService** folder with a name of **getPropertySimple.rb**.

Add the following to the file:

Adds the function library for Sinatra (web service) and active_support (json)

```
require 'sinatra'
require 'active_support/all'

require_relative('./helpers/sinatra_helper.rb');

require 'mongo'

include Mongo

get '/getPropertyByName/:name' do


end
```

Your Code should now look like:

```
getPropertySimple.rb
1  require 'sinatra'
2  require 'active_support/all'
3
4  require_relative('./helpers/sinatra_helper.rb');
5
6  require 'mongo'
7
8  include Mongo
9
10 get '/getPropertyByName/:name' do
11
12
13
14 end
```

Below **get '/getPropertyByName/:name' do** add the following:

Connects to the test database on mongodb.

```
        mongo_db = MongoClient.new("localhost", 27017).db("test")
        coll = mongo_db.collection("properties")

        list = coll.find({"name" => params[:name]}).to_a

        display(list)
```

Your Code should now look like:

getPropertySimple.rb

```ruby
require 'sinatra'
require 'active_support/all'

require_relative('./helpers/sinatra_helper.rb');

require 'mongo'

include Mongo

get '/getPropertyByName/:name' do

    mongo_db = MongoClient.new("localhost", 27017).db("test")
    coll = mongo_db.collection("properties")

    list = coll.find({"name" => params[:name]}).to_a

    display(list)

end
```

6.2 Running the web service.

Open a command prompt and navigate to **c:\RubyRestService** by typing:

cd c:\RubyRestService

```
C:\Windows\system32\cmd.exe
C:\Users\cs811am>cd c:\RubyRestService
c:\RubyRestService>
```

Now to run the web service, type **ruby getPropertySimple.rb**

```
C:\Windows\system32\cmd.exe - ruby getPropertySimple.rb
C:\RubyRestService>ruby getPropertySimple.rb
      *** Notice: The native BSON extension was not loaded. ***

      For optimal performance, use of the BSON extension is recommended.

      To enable the extension make sure ENV['BSON_EXT_DISABLED'] is not set
      and run the following command:

        gem install bson_ext

      If you continue to receive this message after installing, make sure that
      the bson_ext gem is in your load path.

[2014-05-27 10:47:04] INFO  WEBrick 1.3.1
[2014-05-27 10:47:04] INFO  ruby 1.9.3 (2012-12-25) [i386-mingw32]
[2014-05-27 10:47:04] WARN  TCPServer Error: Only one usage of each socket addre
ss (protocol/network address/port) is normally permitted. - bind(2)
== Sinatra/1.4.5 has taken the stage on 4567 for development with backup from WE
Brick
[2014-05-27 10:47:04] INFO  WEBrick::HTTPServer#start: pid=3524 port=4567
```

Open a web browser and navigate to:

http://localhost:4567/getPropertyByName/Andrew