REST API Guidebook

This guidebook provides an independent exercise that supports the *Introduction to Representational State Transfer (REST) with Java, Beginner Part 3* course.

Table of Contents

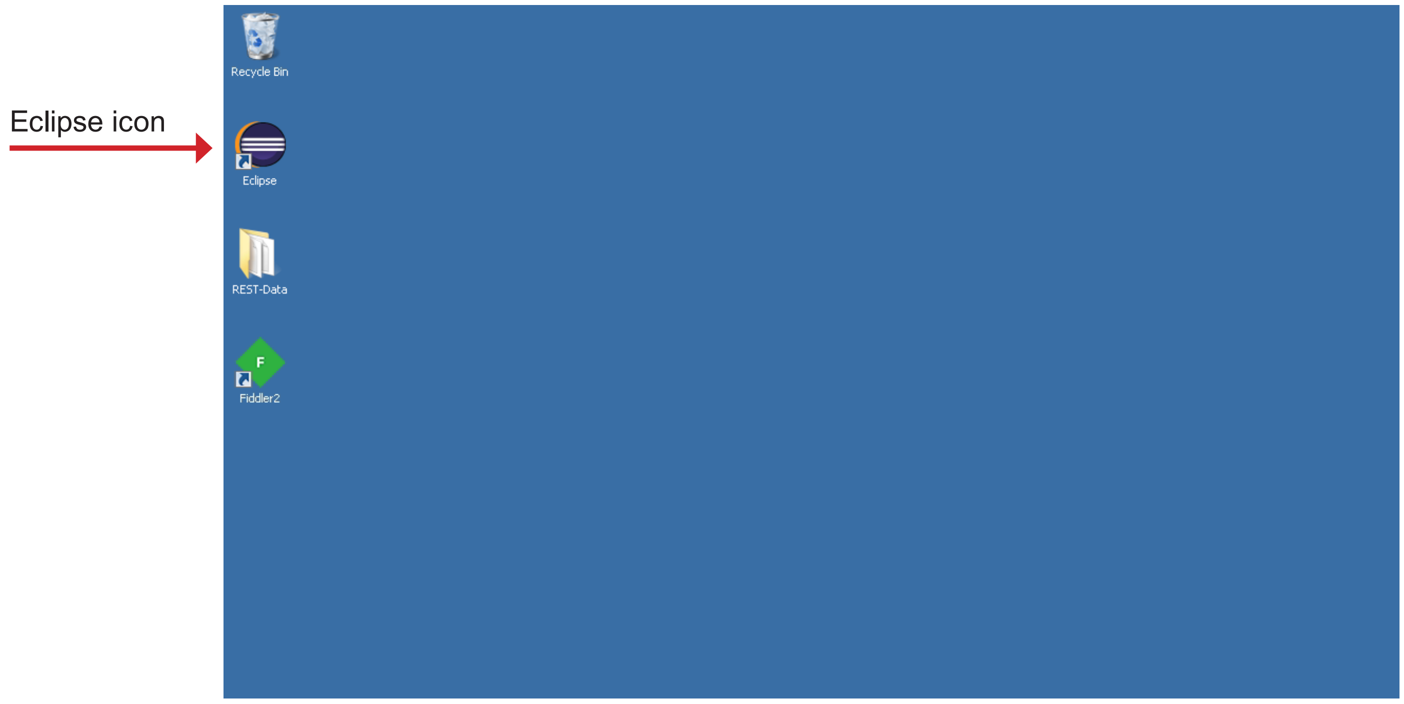
Using the VMware Environment

VMware is a tool offered by VA ITWD that allows you to log in to a virtual machine loaded with all of the tools needed to create a REST API. You can follow the step-by-step directions listed in this guidebook to recreate the demonstrations shown during the course presentation to help you practice in a hands-on environment.

Accessing the VMware Environment

Upon registration for the *Introduction to Representational State Transfer (REST) with Java, Beginner Part 3* course, TMS ID 3878052, you will receive an email message with the link to the VMware virtual machine.

Depending upon your user profile, you may see icons for other tools that you have access to use in this virtual environment.



Setting Up Your Files in the Environment

After opening Eclipse, you need to set up a workspace to make sure your work is in a saved folder designated for you.

1. Select File Tab

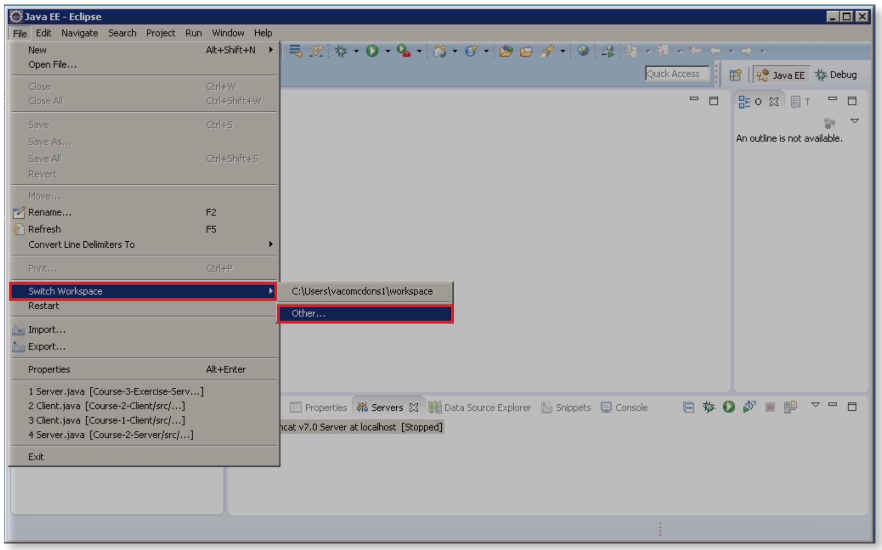
The **File** tab opens to display its menu.

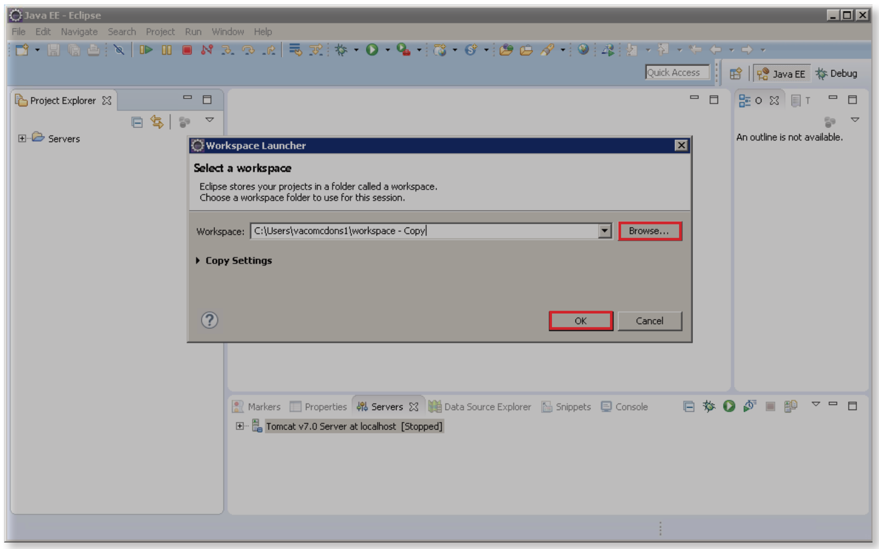
2. Select Switch Workspace

Selecting **Switch Workspace** opens a sub-menu.

3. Select Other

Selecting **Other** opens the **Workspace Launcher** dialog box.


Java EE - Eclipse
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
1 Server.java [Course-3-Exercise-Serv...]
2 Client.java [Course-2-Client/src/...]
3 Client.java [Course-1-Client/src/...]
4 Server.java [Course-2-Server/src/...]
Exit
C:\Users\vacomcdons1\workspace
Other...
Quick Access
Java EE
Debug
An outline is not available.
Properties
Servers
Data Source Explorer
Snippets
Console
ncat v7.0 Server at localhost [Stopped]



Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Users\vacomcdons1\workspace - Copy

4. Browse to the Workspace

In the **Workspace** field of the **Workspace Launcher**, select **Browse** and navigate to the folder where you want to save your work.

Select or create a folder associated with your account, such as a folder on your Desktop or in **My Documents**.

Select **OK** to finish.

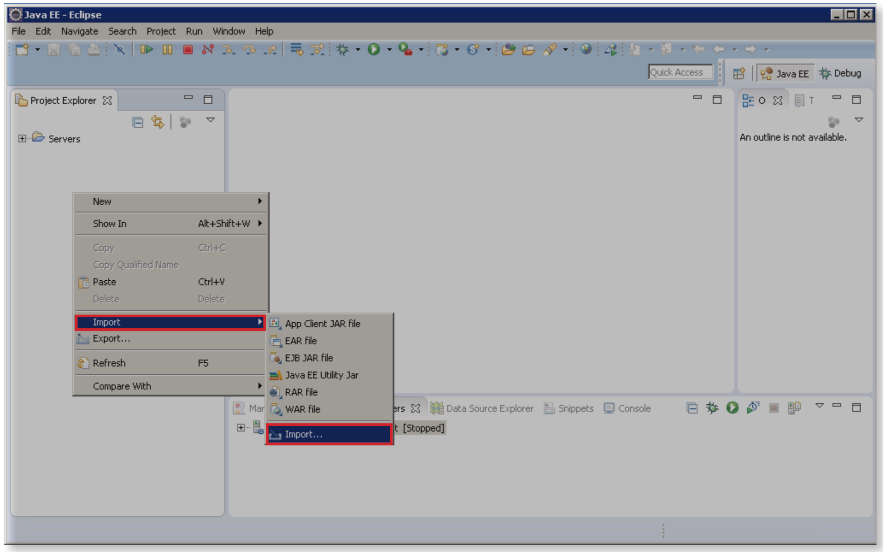
Your REST API project will now be saved in the workspace you designated. Your VM workspace will remain active for the two weeks following the course.

Importing an Existing Project

When you want to import an existing REST project into Eclipse, open Eclipse to start the import.

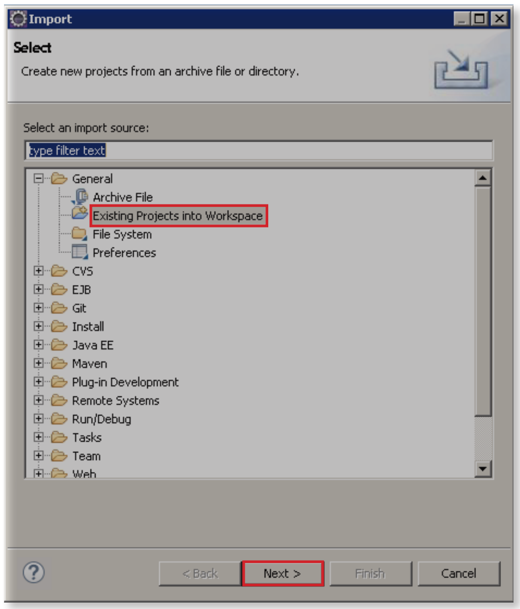
1. Right-Click on the Project Explorer Pane

A dialog box will open with several different options; navigate down to the Import option and select it.



2. Select Import

Once you've selected **Import**, you'll need to navigate down to the bottom of the sub-menu and select the **Import** option again. Selecting **Import** will open the **Import** dialog box shown below in step three.

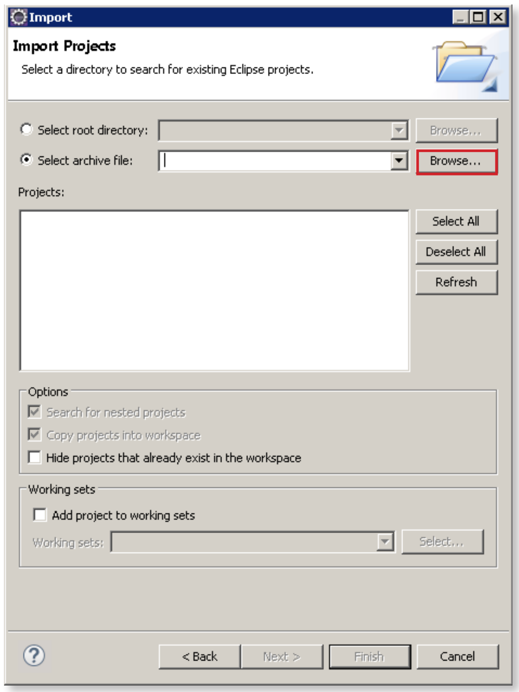


3. Select an Import Source

From the **Import** dialog box, select **General** and **Existing Projects** into Workspace. Select **Next** to continue.

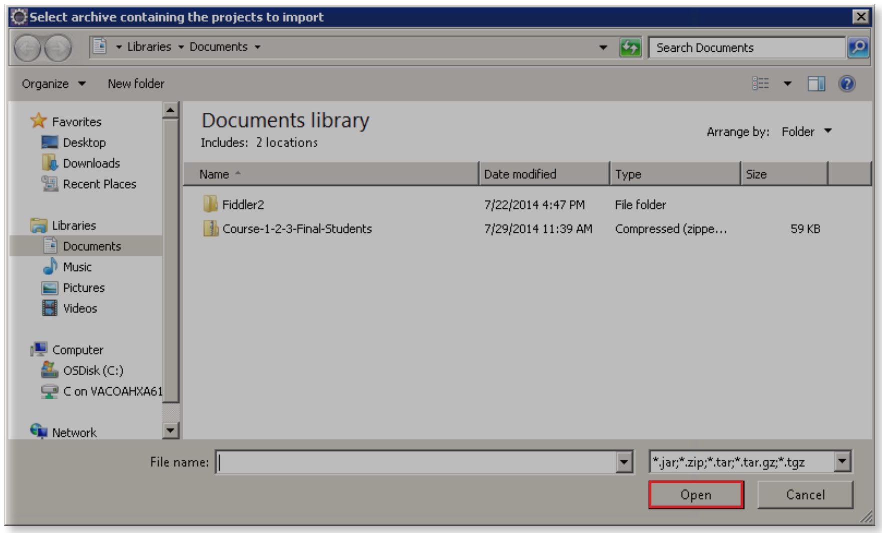
4. Select the Project to Import

On the **Import Project** dialog box, select the **Browse** button.


Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Browse...
Select archive file:
Browse...
Projects:
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Hide projects that already exist in the workspace
Working sets
Add project to working sets
Working sets:
Select...
< Back
Next >
Finish
Cancel

5. Locate on Your Computer Where the Project is Stored

Navigate to and select the zip file that contains the project, then select **Open**.



Select archive containing the projects to import

Libraries ▸ Documents ▸ | Search Documents

Organize ▾ New folder

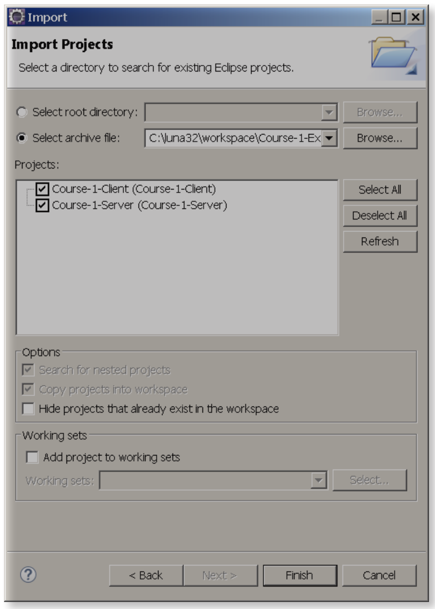
Favorites: Desktop, Downloads, Recent Places
Libraries: Documents, Music, Pictures, Videos
Computer: OSDisk (C:), C on VACOAHXA61
Network

Documents library
Includes: 2 locations
Arrange by: Folder

Name	Date modified	Type	Size
Fiddler2	7/22/2014 4:47 PM	File folder	
Course-1-2-3-Final-Students	7/29/2014 11:39 AM	Compressed (zippe...	59 KB

File name: | *.jar;*.zip;*.tar;*.tar.gz;*.tgz

Open | Cancel



Import

Import Projects

Select a directory to search for existing Eclipse projects.

○ Select root directory: | Browse...
● Select archive file: C:\luna32\workspace\Course-1-Ex | Browse...

Projects:
- [✓] Course-1-Client (Course-1-Client)
- [✓] Course-1-Server (Course-1-Server)

Select All | Deselect All | Refresh

Options
- [✓] Search for nested projects
- [✓] Copy projects into workspace
- [ ] Hide projects that already exist in the workspace

Working sets
- [ ] Add project to working sets

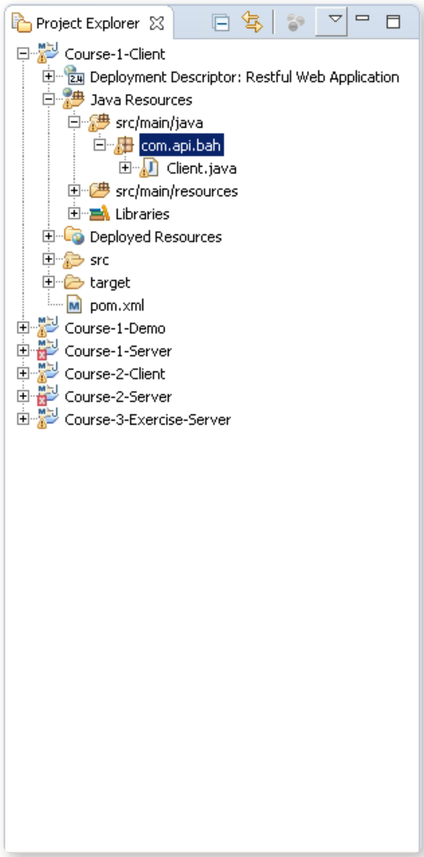
Working sets: | Select...

< Back | Next > | Finish | Cancel

This will show you your projects in the zip file.

6. View the Imported Project in Project Explorer

Go to **Project Explorer** to locate the existing project that was just imported into your Eclipse. From there, you can add or edit the REST API project as you wish.

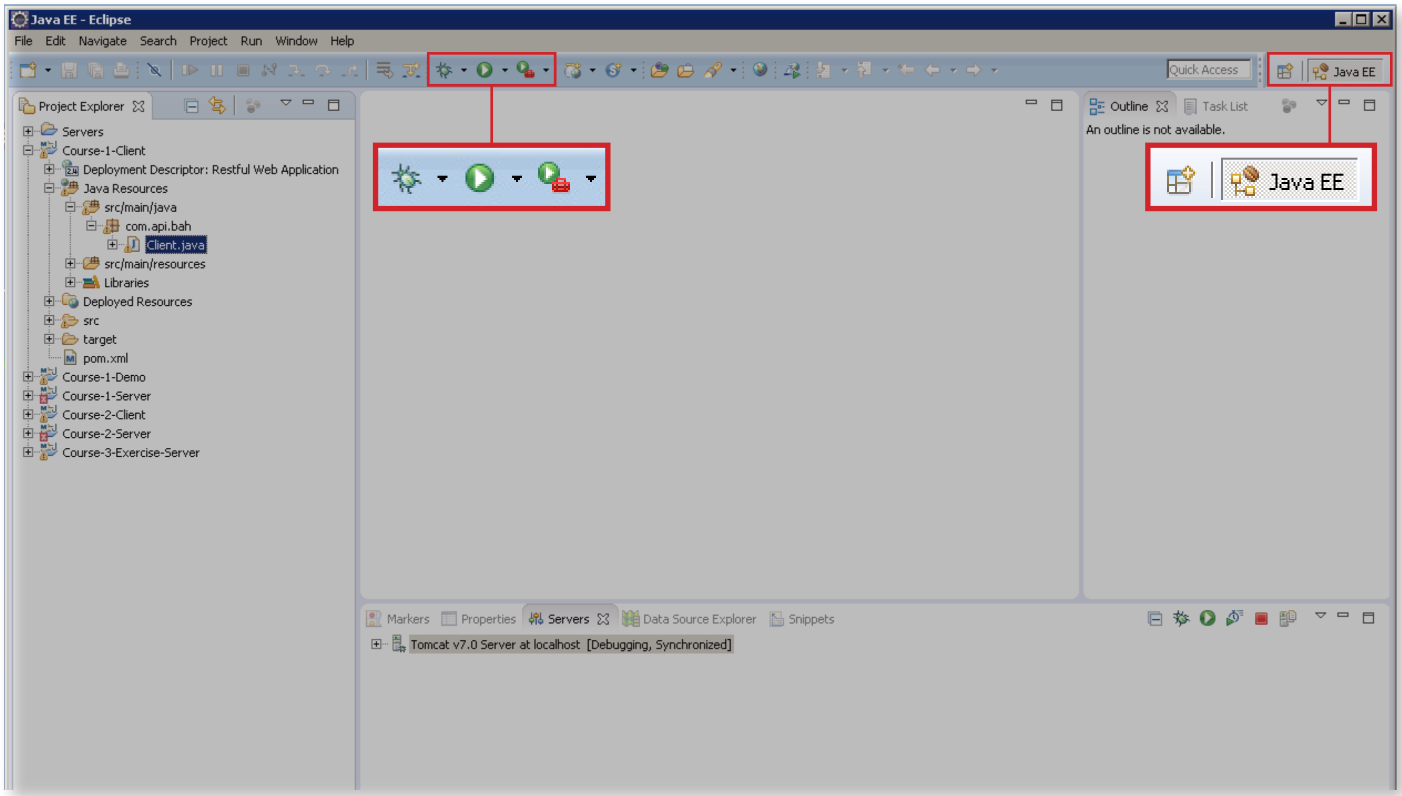


Project Explorer

```
Course-1-Client
  Deployment Descriptor: Restful Web Application
  Java Resources
    src/main/java
      com.api.bah
        Client.java
    src/main/resources
    Libraries
  Deployed Resources
  src
  target
  pom.xml
Course-1-Demo
Course-1-Server
Course-2-Client
Course-2-Server
Course-3-Exercise-Server
```

The Eclipse Interface

To help you become familiar with Eclipse's interface, we've highlighted some of its tools and features and provided the purpose and function of each.



1. Top bar

The buttons on this bar help run and edit your applications. The buttons we're going to highlight are the ones that will help you debug and run your assignments.

a. Debug Button

This button runs your app in debug mode by attaching the debugger. It enables you to discover and diagnose coding and other mistakes that may occur in your application. This is especially useful for determining run-time errors or errors that can only be detected once the application has launched One of the many useful features of the debugger is the ability to step through your code and examine the contents of variables.

b. Run Button

This button runs your API without attaching the debugger. When developing your APIs, it is a best practice to avoid using this button because it does not show you where errors are, only that you have them. Once your system is in production, it should always be compiled without debug options enabled.

c. Open Perspective Button

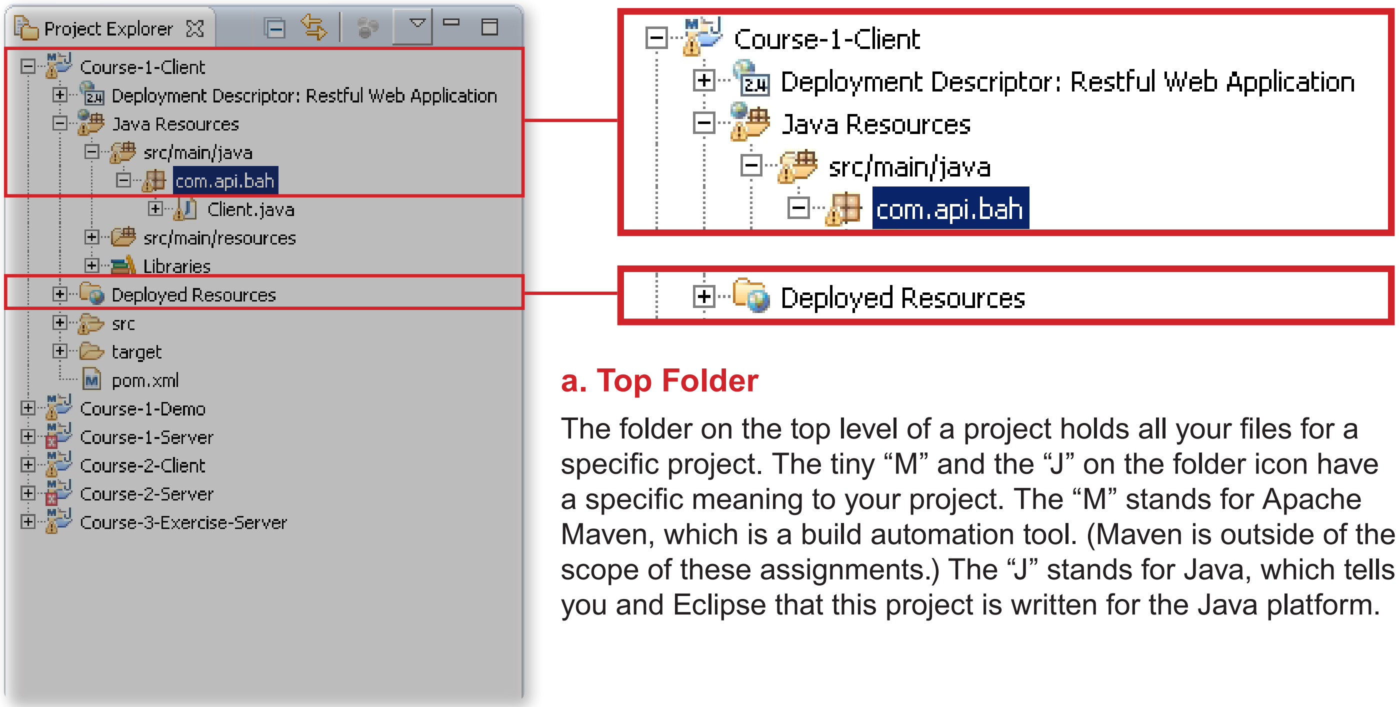
This button provides different views to assist you in completing specific steps while creating REST APIs. This button switches between different perspectives. The perspectives that will be most helpful in your assignments will be the Debug and Java.

d. Java EE Perspective Button

This button is another perspective button and it opens the **Java EE perspective**. It's used for Java projects and will be helpful if you get lost and need a shortcut to get back to your projects.

2. Project Explorer Pane

The purpose of this pane is to organize the files of the project you're working on. Files in this pane are displayed in a hierarchical view to help show how project files are arranged. The project files we're going to highlight will help you understand how a REST API project written in Java is organized in Eclipse.



a. Top Folder

The folder on the top level of a project holds all your files for a specific project. The tiny "M" and the "J" on the folder icon have a specific meaning to your project. The "M" stands for Apache Maven, which is a build automation tool. (Maven is outside of the scope of these assignments.) The "J" stands for Java, which tells you and Eclipse that this project is written for the Java platform.

b. Deployment Descriptor

As the project hierarchy is expanded, the next file you'll notice is the **Deployment Descriptor.** This file identifies the project as a RESTful web application.

c. Java Resources

The next folder in the same level of hierarchy as the **Deployment Descriptor** is the **Java Resources** folder. This folder holds content such as Java code and the file where you will edit your assignments.

src/main/java folder

As we expand down the Java Resources hierarchy one level, you'll see the **src/main/java** folder. This folder holds the file, **Server.java**, to edit your assignments. To locate this file, expand down the **Java Resources.**

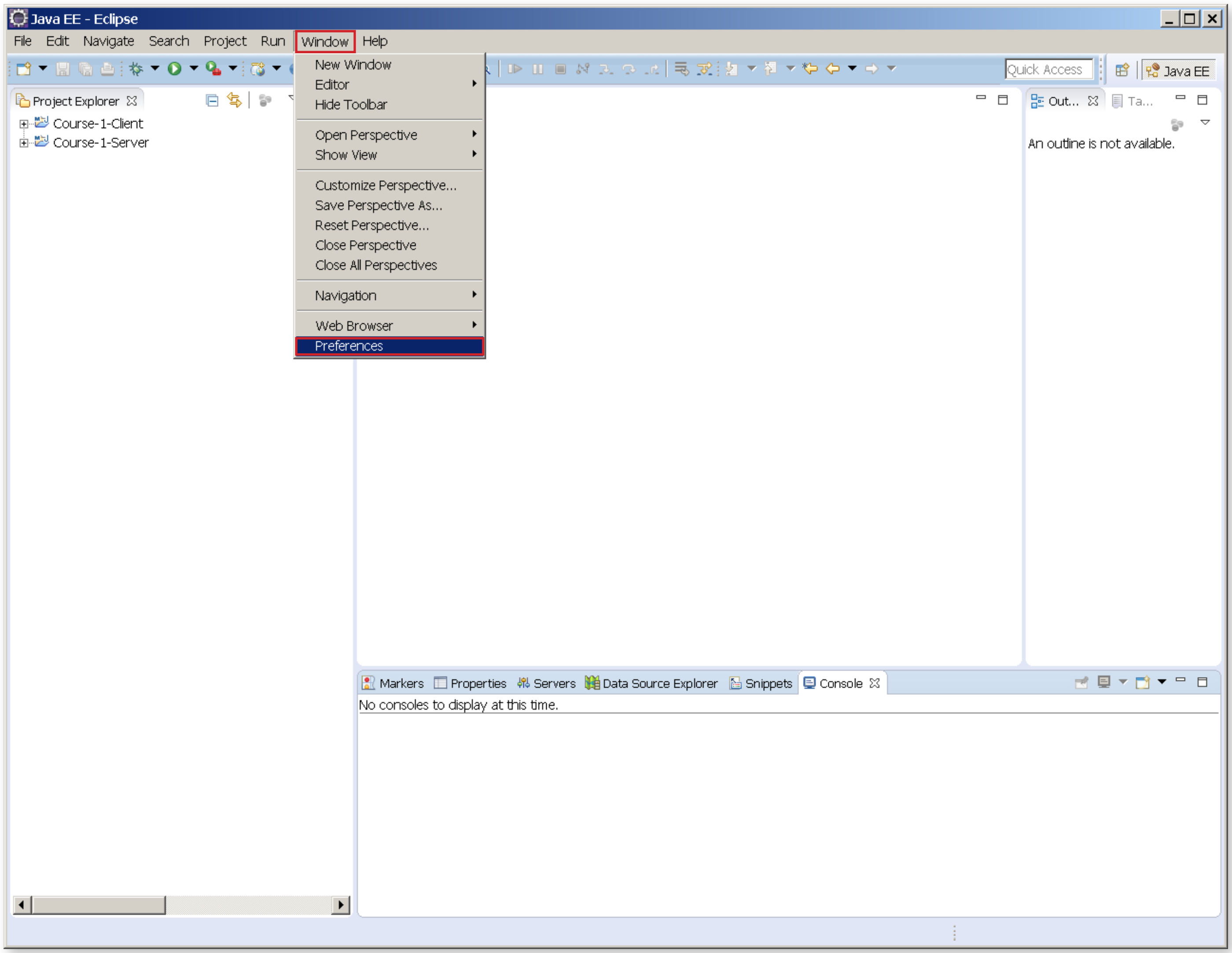
d. src and target folders

The **src** and **target** folders are in the same level of hierarchy as the **Java Resources** folder and shouldn't be edited or removed. The files in these folders were created to point to specific files necessary for running your program.

Setting Up For Testing

Before you can test your server project, you will need to define the server runtime executables that Eclipse will need to load. In our case, we are using Apache Tomcat version 7.

1. Click on **Window** in the top menu and go to **Preferences**.



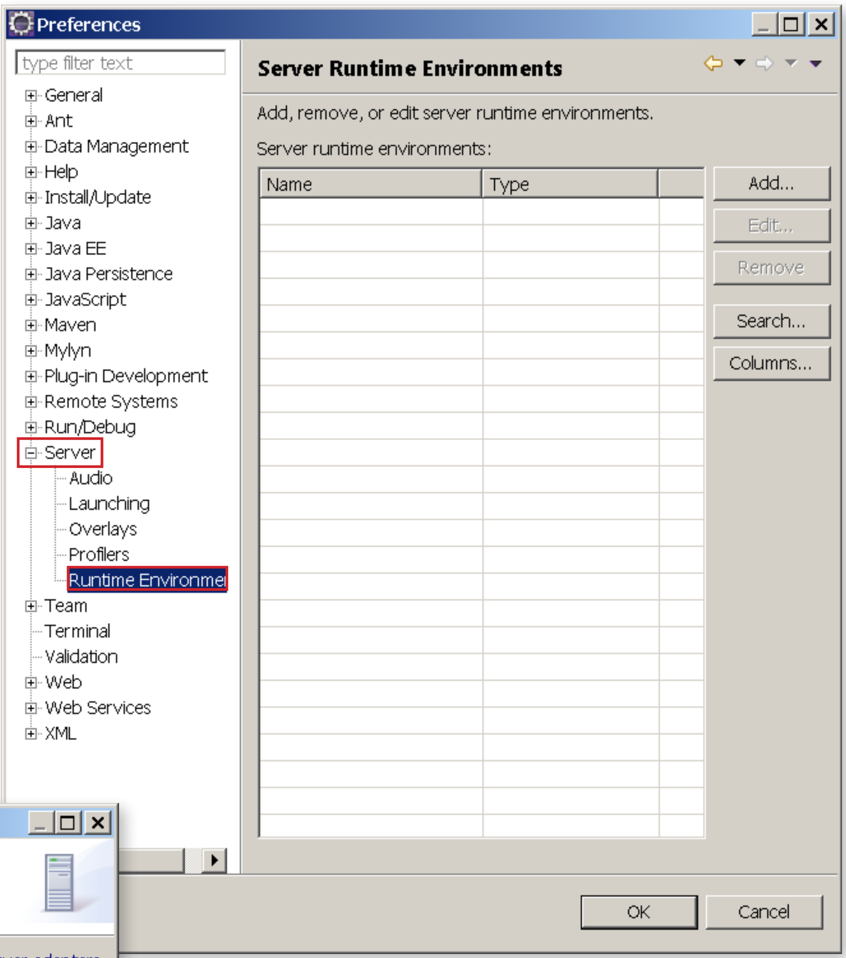
An outline is not available.

No consoles to display at this time.

2. Expand the **Server** tab on the left, and select **Runtime Environment**.

3. Expand the **Apache folder** and select **Apache Tomcat 7.0**. Click **Next.**

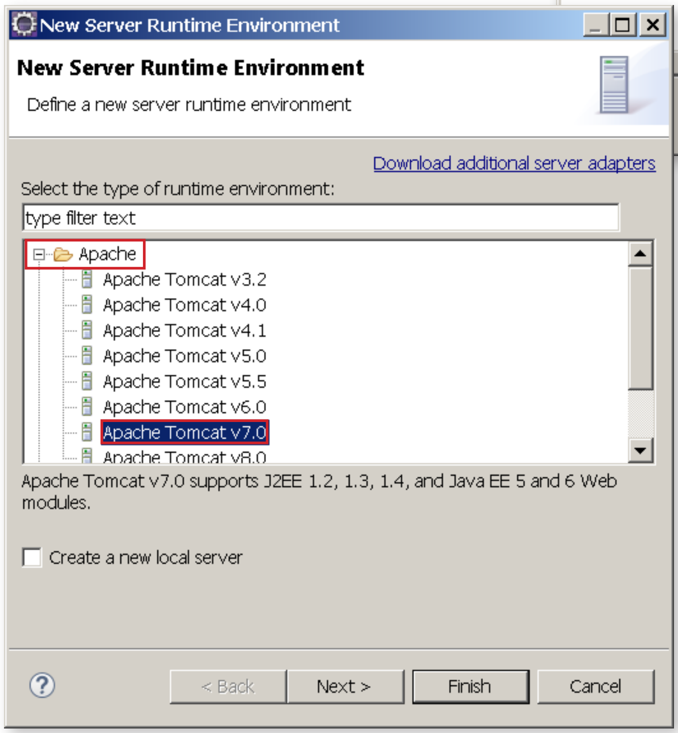
4. Click **Browse** to select the **Apache Tomcat 7.0** folder.


Preferences

Server Runtime Environments

Add, remove, or edit server runtime environments.

Server runtime environments:


New Server Runtime Environment

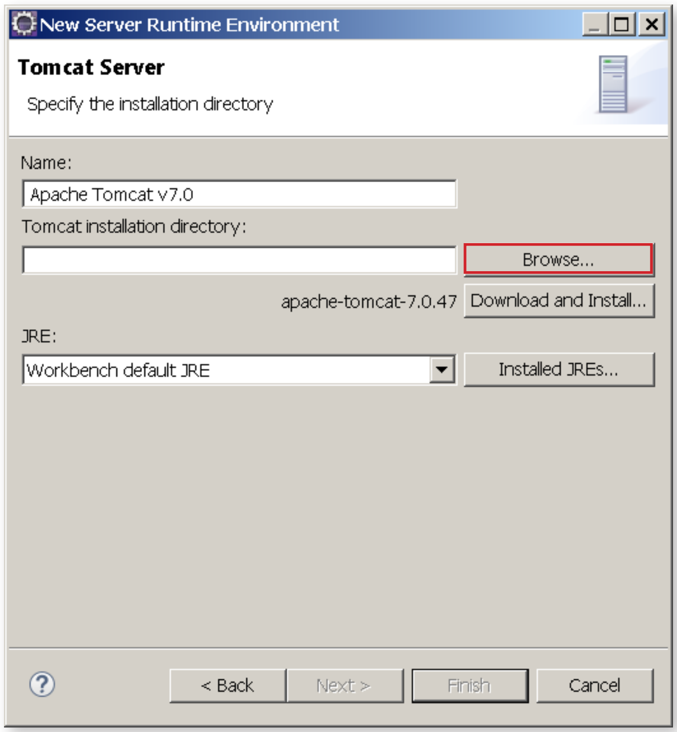
New Server Runtime Environment

Define a new server runtime environment

Select the type of runtime environment:

Apache Tomcat v7.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5 and 6 Web modules.

Create a new local server


New Server Runtime Environment

Tomcat Server

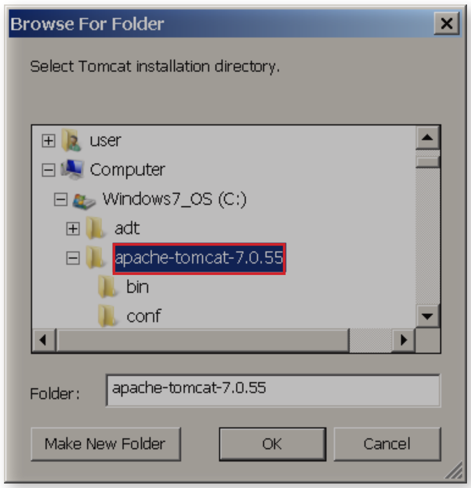
Specify the installation directory

Name: Apache Tomcat v7.0

Tomcat installation directory:

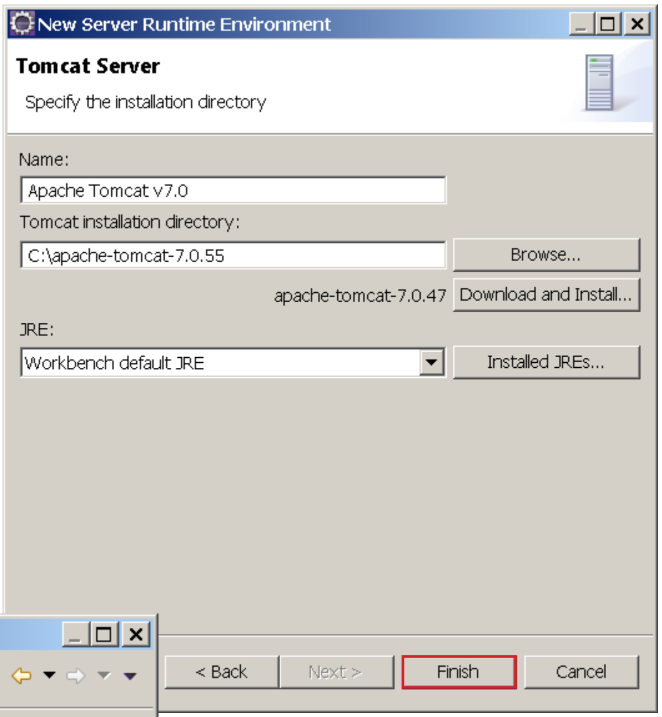
apache-tomcat-7.0.47

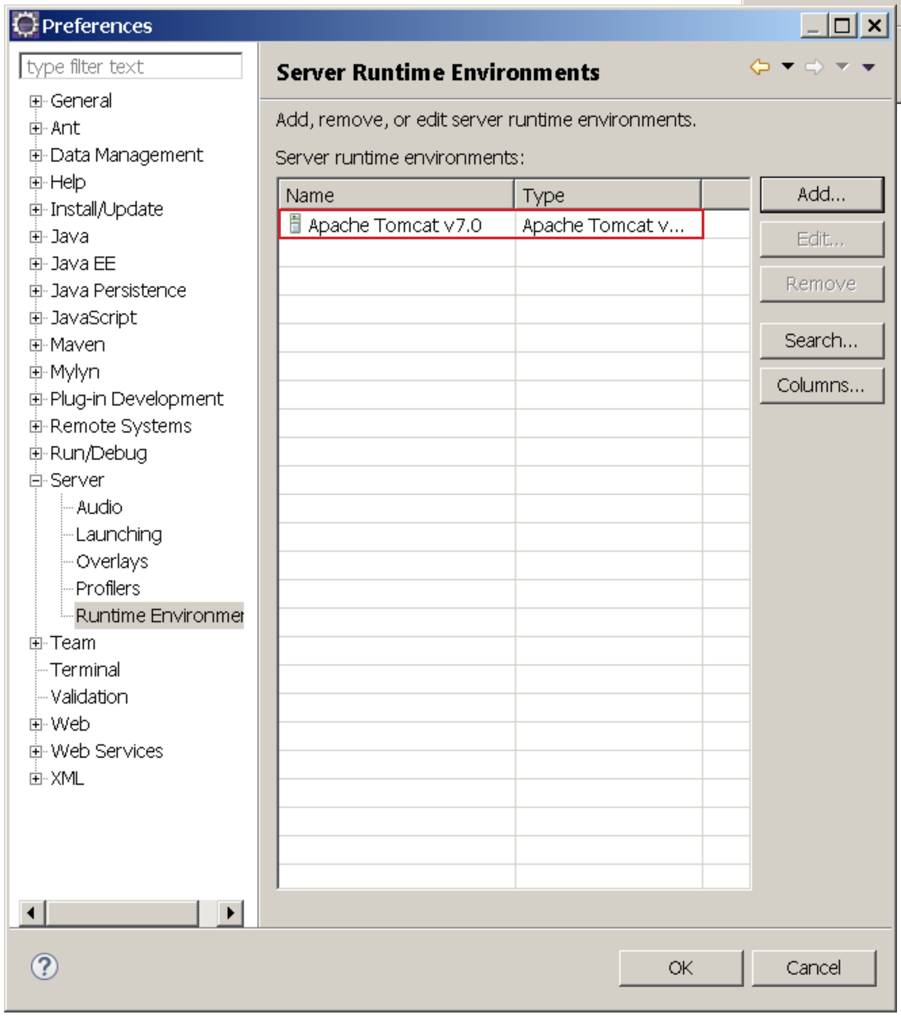
JRE: Workbench default JRE



5. Select the **c:\apache-tomcat-7.0.55** folder and click **OK.**

6. Click **Finish**.





7. You will see that the **Apache Tomcat 7.0** runtime environment has been linked to Eclipse.

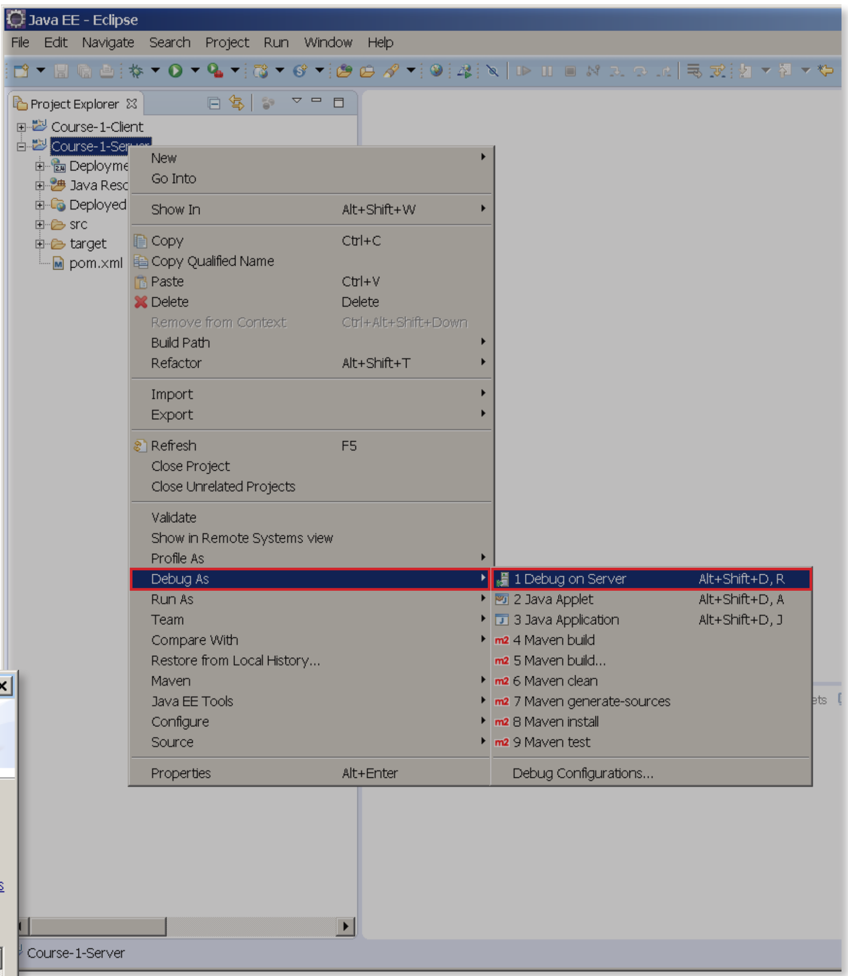
That's it—now Eclipse knows how to run your web server applications.

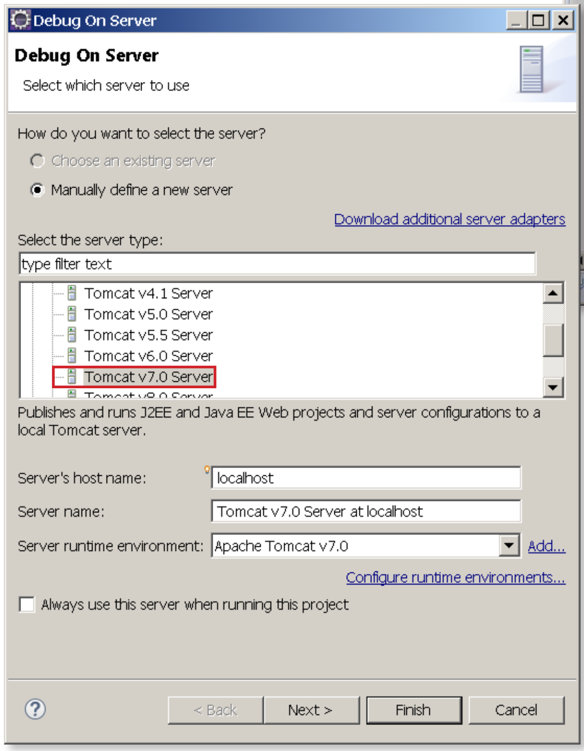
How to Test Your Exercises

Once you have completed the exercises in the next section of this guidebook, and you have resolved all the errors, you are ready to test. To test your exercises, you will need to select the context (click on the project folder), and then debug your project. For the clients (and other projects that do not need to run in a server container), debug as Java Application. For server applications, you will need to launch the debugger from the **Project Explorer** pane. The steps below show you how to test your exercises.

1. Right-Click on the Top Folder of the project you're working on

When you right-click on the top folder a sub-menu will open. Navigate down to **Debug As** and select Java Application for non-server code. For server code, select the first option, **1. Debug on Server**. This starts the application running on the server. Selecting **1. Debug on Server** will open the dialog box shown below.




Debug On Server
Download additional server adapters
Add...
Configure runtime environments...
< Back
Next >
Finish
Cancel

Debug On Server

Select which server to use

How do you want to select the server?

( ) Choose an existing server
(•) Manually define a new server

Select the server type:

type filter text

- Tomcat v4.1 Server
- Tomcat v5.0 Server
- Tomcat v5.5 Server
- Tomcat v6.0 Server
- Tomcat v7.0 Server
- Tomcat v8.0 Server

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name:	localhost
Server name:	Tomcat v7.0 Server at localhost
Server runtime environment:	Apache Tomcat v7.0

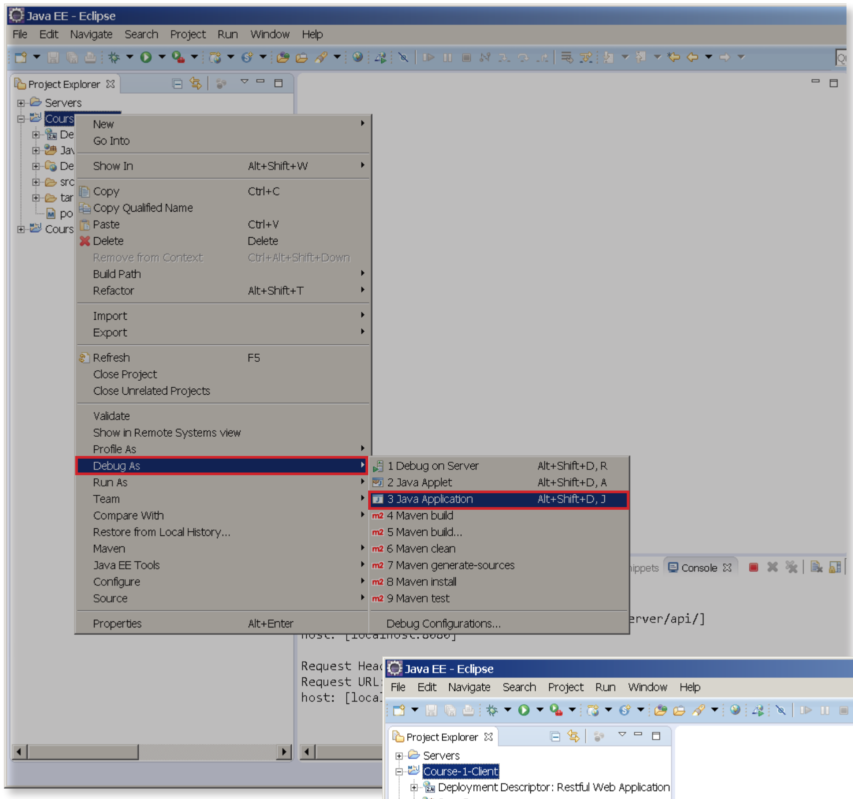
☐ Always use this server when running this project

2. Debug On Server

Selecting **1. Debug on Server** will open the dialog box shown on the left. Expand the **Apache** folder and select **Tomcat v7.0 Server at localhost** and then select **Finish**. This will run the REST API that you're working on.

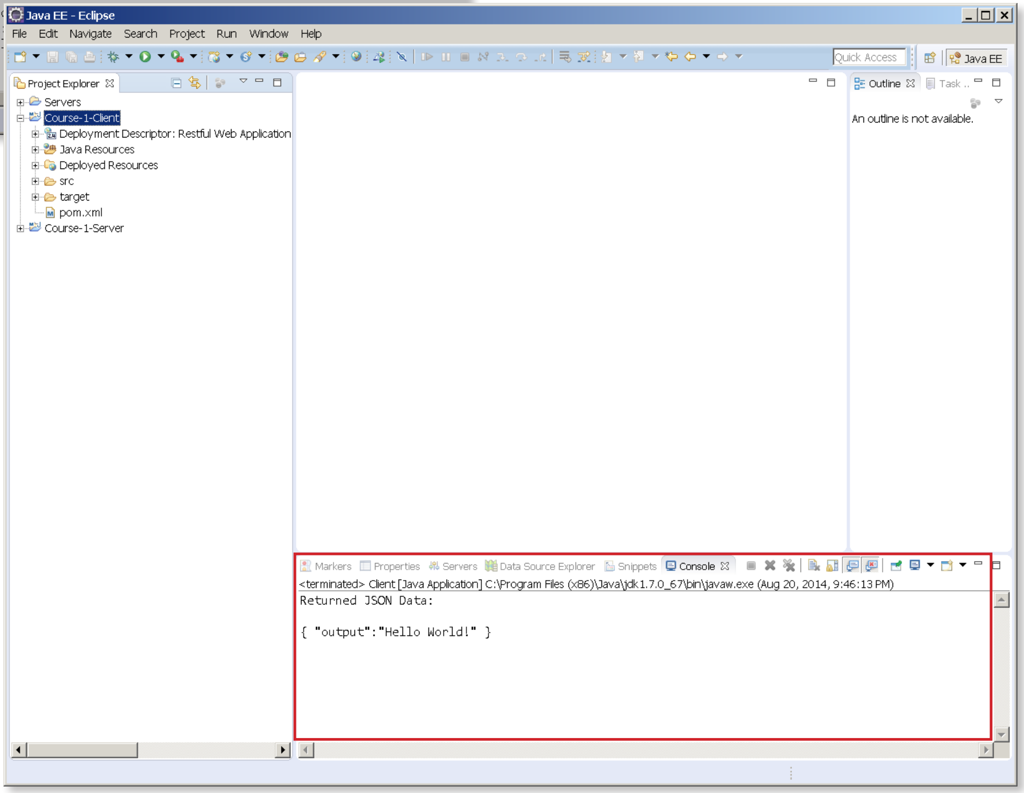
3. Debug As Java Application

Right-click on the top folder of the project you're working on to open a sub-menu. Navigate down to **Debug As** and select the third option, **3. Java Application**. This runs the client, and output will be displayed in the **Console** area.



4. View Results in the Console area

Once the results have been output, you will be able to view them in the console areas of Eclipse.


Java EE - Eclipse
File Edit Navigate Search Project Run Window Help
Quick Access
Java EE
Project Explorer
Servers
Course-1-Client
Deployment Descriptor: Restful Web Application
Java Resources
Deployed Resources
src
target
pom.xml
Course-1-Server
Outline
Task
An outline is not available.
Markers Properties Servers Data Source Explorer Snippets Console

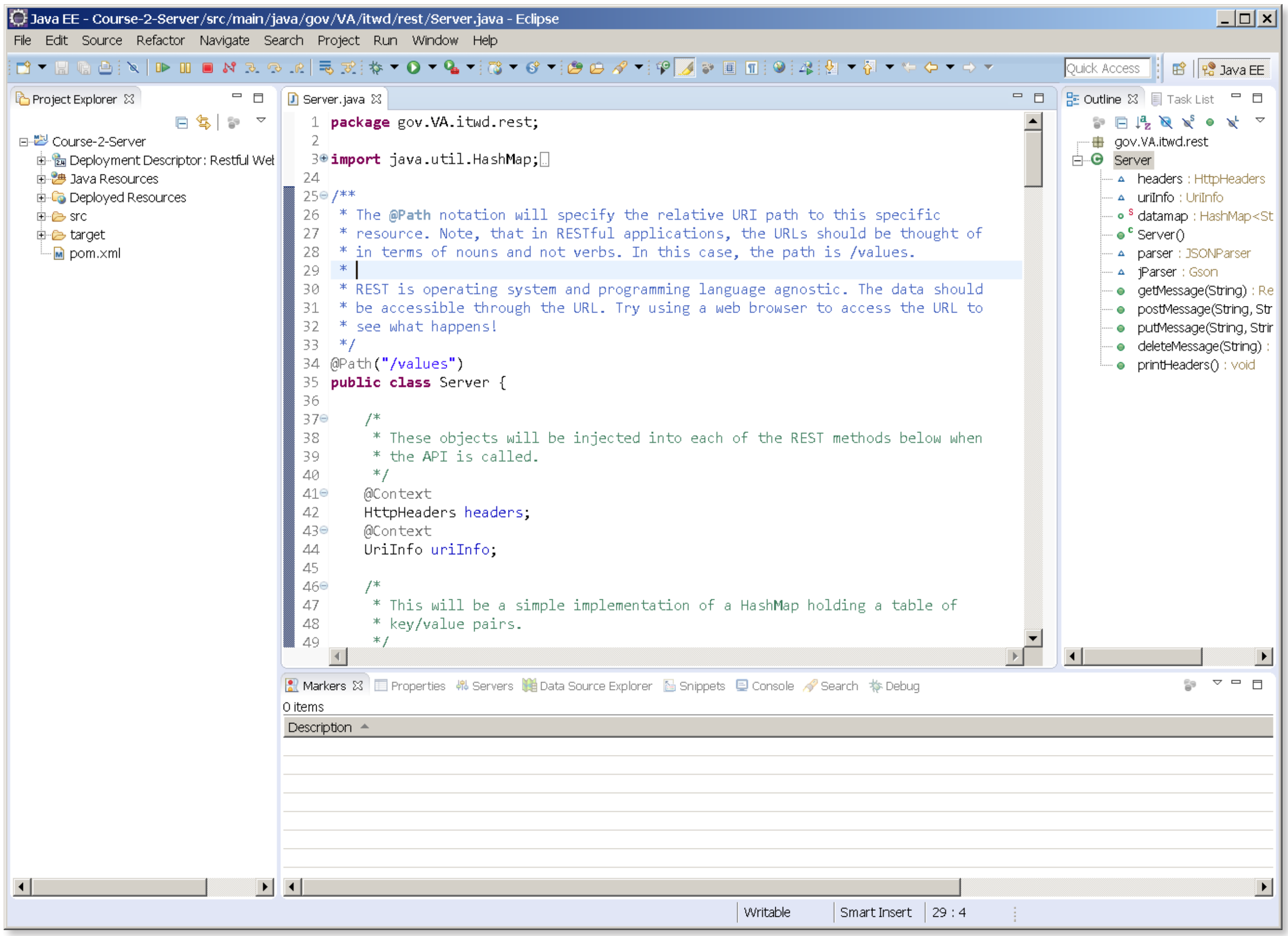
```
<terminated> Client [Java Application] C:\Program Files (x86)\Java\jdk1.7.0_67\bin\javaw.exe (Aug 20, 2014, 9:46:13 PM)
Returned JSON Data:

{ "output":"Hello World!" }
```

Course 3: REST Exercise

Exercise: Debugging a REST API Using Fiddler

In this third exercise, we're going to ask you to go through some basic debugging steps and Eclipse's debugging capabilities. We're also going to introduce you to Fiddler, a web-debugging proxy. Before you use Fiddler for your exercise, you need to open Eclipse and start the Course 2 exercise application on the Tomcat server. In your previous exercises, a custom client was written to test the REST APIs.

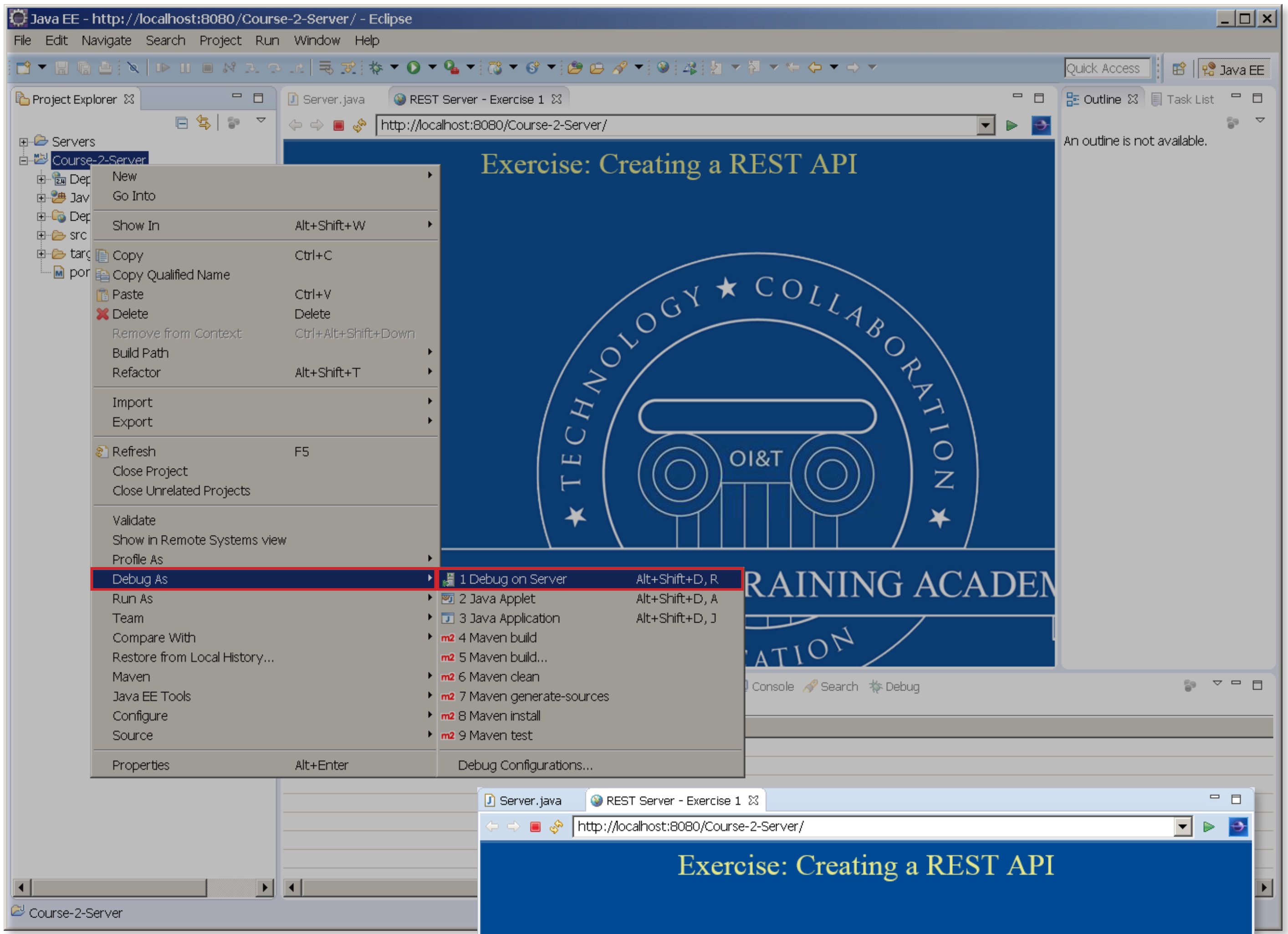
In this exercise, we're going to use Fiddler, an HTTP client to communicate with our API. Please refer to the *Introduction to REST with Java, Beginner Part 2 Guidebook* to set up your project files and run the Course 2 Server Application.

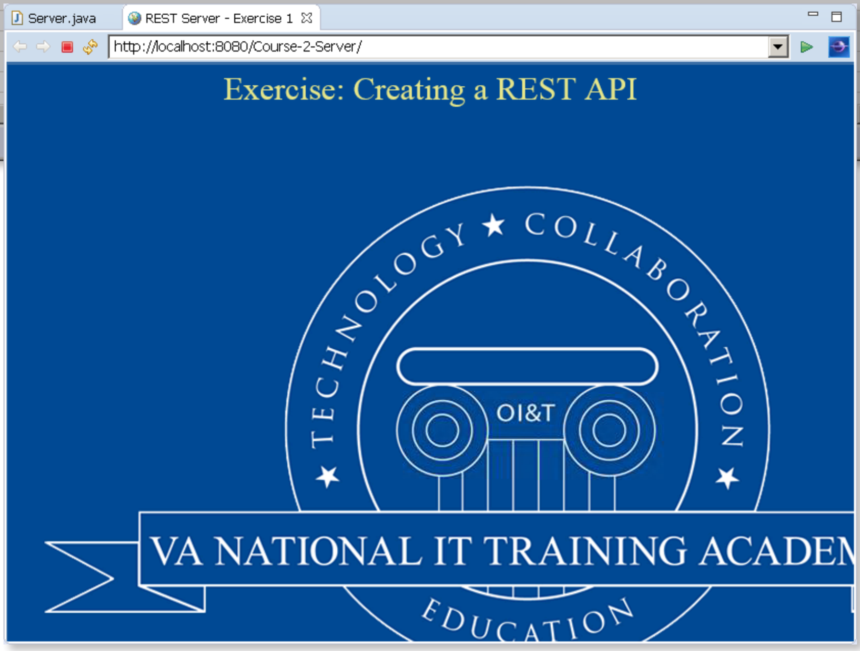


```java
package gov.VA.itwd.rest;

import java.util.HashMap;

/**
 * The @Path notation will specify the relative URI path to this specific
 * resource. Note, that in RESTful applications, the URLs should be thought of
 * in terms of nouns and not verbs. In this case, the path is /values.
 *
 * REST is operating system and programming language agnostic. The data should
 * be accessible through the URL. Try using a web browser to access the URL to
 * see what happens!
 */
@Path("/values")
public class Server {

    /*
     * These objects will be injected into each of the REST methods below when
     * the API is called.
     */
    @Context
    HttpHeaders headers;
    @Context
    UriInfo uriInfo;

    /*
     * This will be a simple implementation of a HashMap holding a table of
     * key/value pairs.
     */
```

1. Run the Course 2 Server Application

Once imported, run the Course 2 Server Application by right-clicking on the top project folder, go to **Debug As**, and then **1. Debug on Server**. Follow the prompts discussed in previous exercises, including setting up the Tomcat 7.x runtime, if necessary.



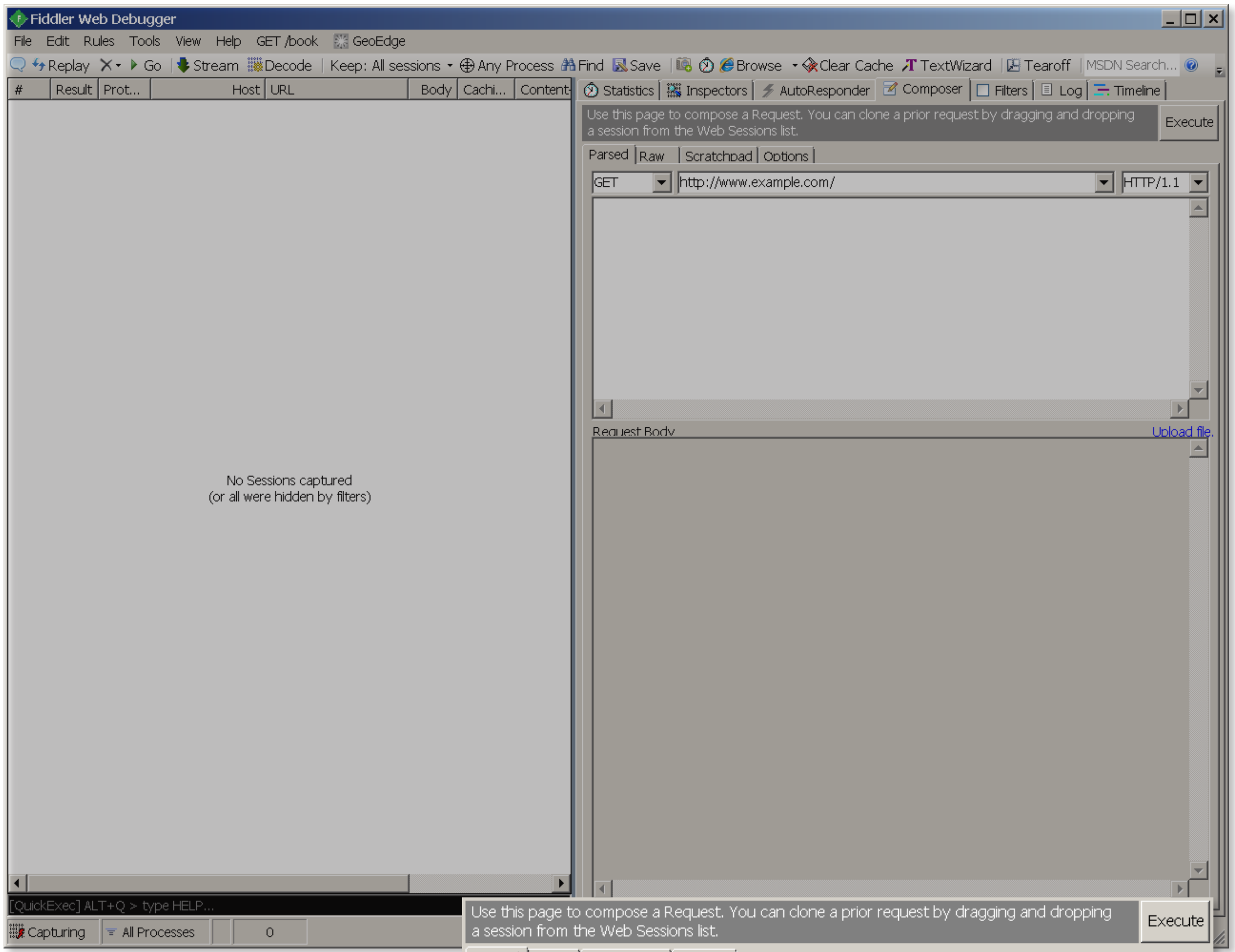

Server.java
REST Server - Exercise 1
http://localhost:8080/Course-2-Server/
Exercise: Creating a REST API
TECHNOLOGY ★ COLLABORATION
OI&T
VA NATIONAL IT TRAINING ACADEM
EDUCATION

Once the application has loaded, Eclipse will launch its own web browser for the application. Please make a note of the URL of the application; – we will need that information for Fiddler. We'll be entering this URL into Fiddler to start our connection to the REST API. Also, keep Eclipse running, since that is where the application is hosted and where we'll be doing our debugging.

2. Start Fiddler

You can launch Fiddler from the desktop by clicking the shortcut, which is a green rhombus with an F in the middle.

Fiddler will default to the last tab used. If not selected, click on the **Composer** tab to select it. The **Composer** tab is where you create an HTTP message and where you'll be doing a majority of the exercise for this course.

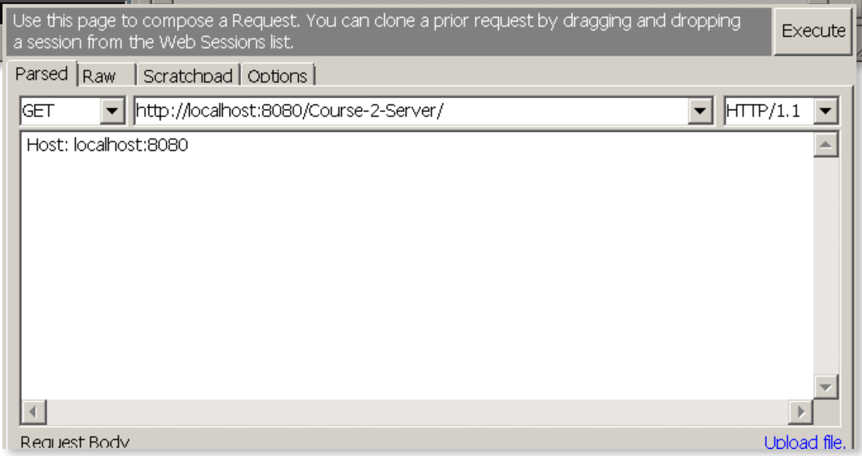

Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search...
# Result Prot... Host URL Body Cachi... Content
Statistics Inspectors AutoResponder Composer Filters Log Timeline
Execute
Parsed Raw Scratchpad Options
Request Body
Upload file.
[QuickExec] ALT+Q > type HELP...
Capturing All Processes 0

Use this page to compose a Request. You can clone a prior request by dragging and dropping a session from the Web Sessions list.

GET | http://www.example.com/ | HTTP/1.1

No Sessions captured
(or all were hidden by filters)

Copy the server URL we previously noted when running the Course 2 Server Application. It should be something like http://localhost:8080/Course-2-Server. Paste the URL in the **Parsed** tab in Fiddler, and click **Execute**. This will execute the default Fiddler action, which is GET, and will return the **base index.htm** file associated with the server.

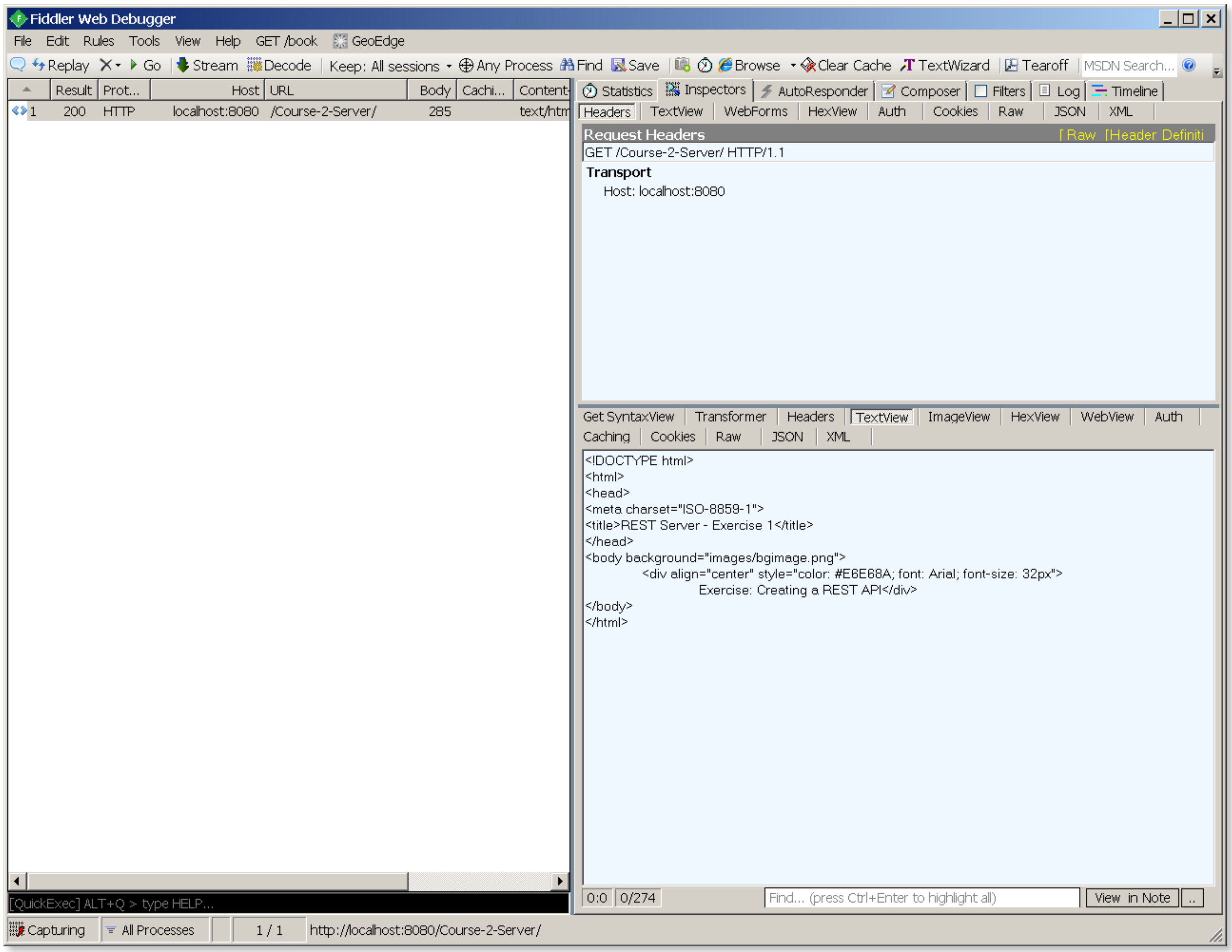


Use this page to compose a Request. You can clone a prior request by dragging and dropping a session from the Web Sessions list.

GET | http://localhost:8080/Course-2-Server/ | HTTP/1.1

Host: localhost:8080

Fiddler is an HTTP client and acts just as any web browser might; however, it gives us the ability to control what is sent and inspect what is received. Once the GET is executed, a result will pop up on the left. Double-click that result line (notice how it shows status 200, which we know means OK/Success). Fiddler will show you the text of the response in the **Text View** tab, which is selected by default. You can click the other tabs to see different renderings of the response, such as **Web View**, which should be almost what you would see in a regular web browser. We are now ready to remote control our API!


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search...
Statistics Inspectors AutoResponder Composer Filters Log Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth Caching Cookies Raw JSON XML
[QuickExec] ALT+Q > type HELP...
Capturing All Processes 1 / 1
Find... (press Ctrl+Enter to highlight all) View in Note

	Result	Prot...	Host	URL	Body	Cachi...	Content
1	200	HTTP	localhost:8080	/Course-2-Server/	285		text/htm

```
Request Headers
GET /Course-2-Server/ HTTP/1.1
Transport
    Host: localhost:8080
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>REST Server - Exercise 1</title>
</head>
<body background="images/bgimage.png">
		<div align="center" style="color: #E6E68A; font: Arial; font-size: 32px">
			Exercise: Creating a REST API</div>
</body>
</html>
```

0:0 0/274

http://localhost:8080/Course-2-Server/

3. Update the Standardized Action, URL, and Protocol Version

We don’t have anything in our database yet, so it would be a good idea to PUT something in it for testing. Specifically, a value named “test.” Select **PUT** from the drop-down **Standardized Actions** window. You’ll notice that there are several actions, but we’re only going to focus on the actions **GET** and **PUT**.

Recall from our deployment descriptor (web.xml) that our API URL pattern is /api/*. Also, recall from the previous exercises that the path (defined by the @Path tag) to the standard actions starts with /values/. So the full URL you will need to use to prefix all of the standardized actions is your base url/api/values/. For the application run from Eclipse using Tomcat, the url should be:

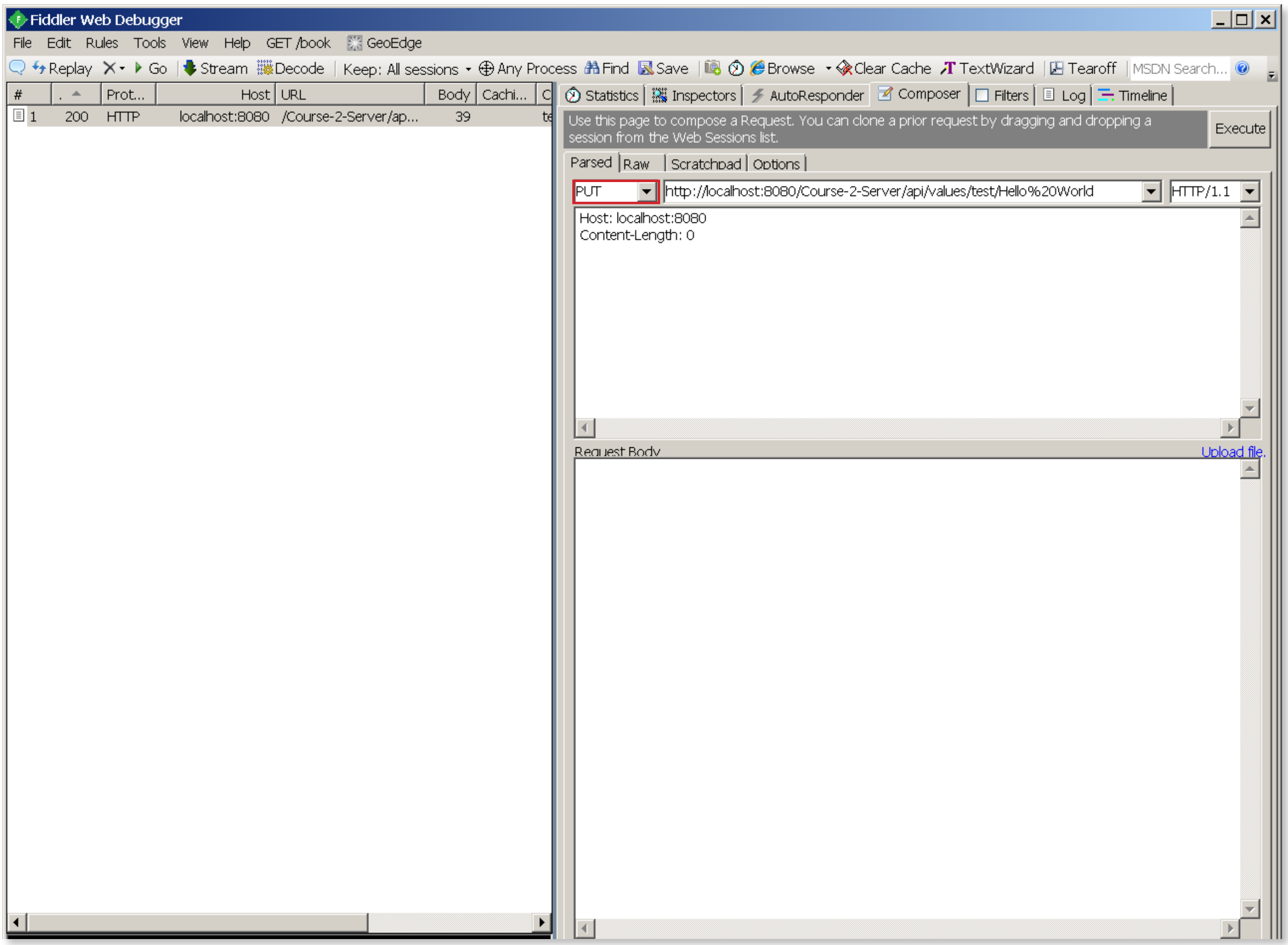
“http://localhost:8080/Course-2-Server/api/values/”

From here, you can add on the additional information to specify the standardized actions. In the URL box, add “/test” at the end of the URL and “/Hello%20World.”

Note: All URLs must be escaped. Escaping a URL is taking characters that are invalid for HTTP resources and converting them into an encoded format that is a valid URL. The “%20” in the URL represents a space.

To the right of the **URL** box, you’ll see the **Protocol Version** drop-down box. Make sure that **HTTP/1.1** is selected.

Select **Execute**.


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search...
Statistics Inspectors AutoResponder Composer Filters Log Timeline
Execute
Parsed Raw Scratchpad Options
Request Body
Upload file.

#	.	Prot...	Host	URL	Body	Cachi...
1	200	HTTP	localhost:8080	/Course-2-Server/ap...	39	

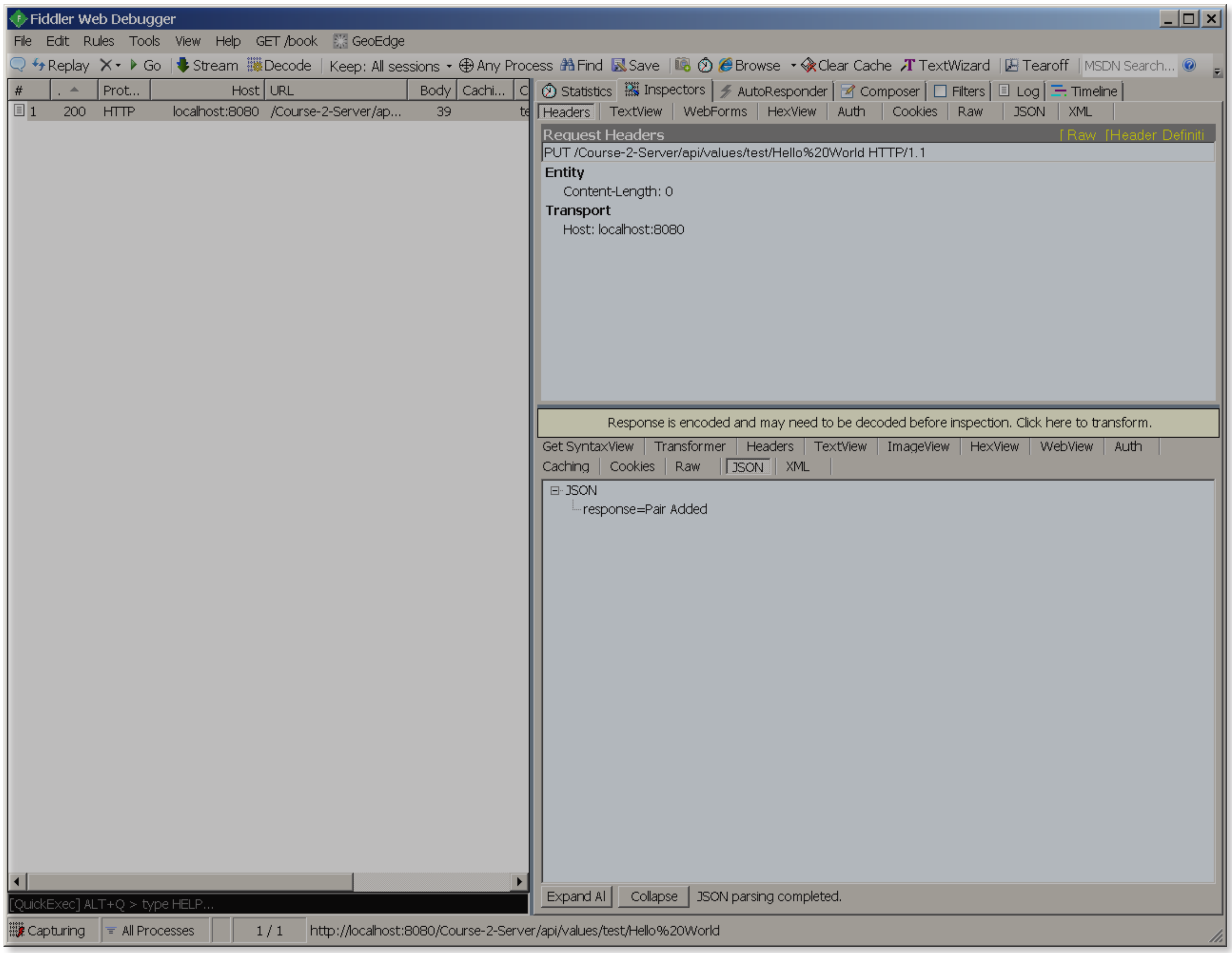
Use this page to compose a Request. You can clone a prior request by dragging and dropping a session from the Web Sessions list.

PUT http://localhost:8080/Course-2-Server/api/values/test/Hello%20World HTTP/1.1

```
Host: localhost:8080
Content-Length: 0
```

4. Select the JSON Tab

Double-click the response at left and then go to the **JSON** tab. Notice how the JSON string you coded is interpreted as a key/value pair. You can select the other tabs to see different renderings of the response. Feel free to experiment with the other view tabs.


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search...
Statistics Inspectors AutoResponder Composer Filters Log Timeline
Headers TextView WebForms HexView Auth Cookies Raw JSON XML
Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth
Caching Cookies Raw JSON XML
Expand All Collapse
[QuickExec] ALT+Q > type HELP...
Capturing All Processes 1 / 1

#	Result	Protocol	Host	URL	Body
1	200	HTTP	localhost:8080	/Course-2-Server/ap...	39

```
Request Headers
PUT /Course-2-Server/api/values/test/Hello%20World HTTP/1.1
Entity
    Content-Length: 0
Transport
    Host: localhost:8080
```

Response is encoded and may need to be decoded before inspection. Click here to transform.

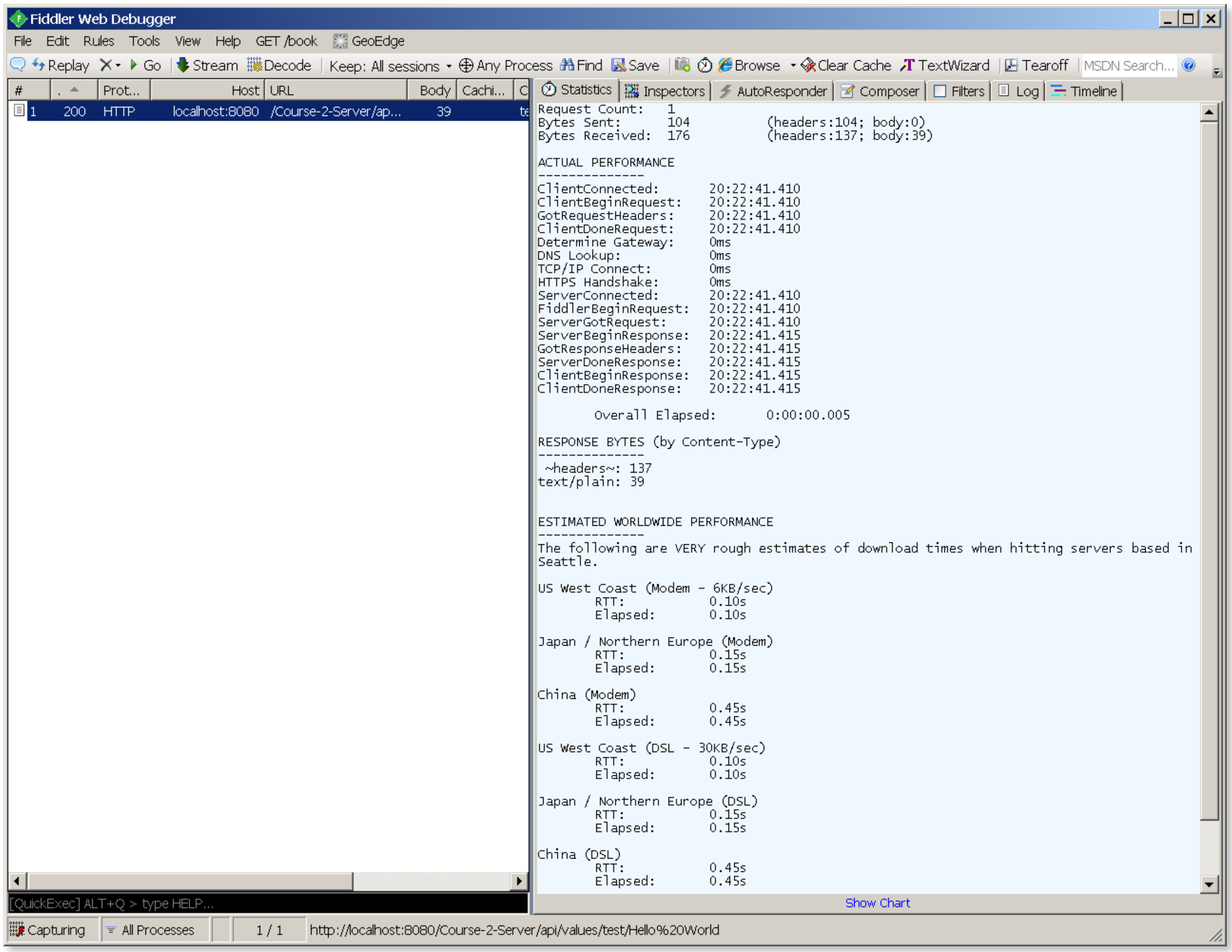
```
⊟ JSON
   └response=Pair Added
```

JSON parsing completed.

http://localhost:8080/Course-2-Server/api/values/test/Hello%20World

5. Open the Statistics Tab

Now that you have executed a **PUT** standardized action, select the **Statistics** tab and open it. Useful performance metrics are shown. On the left, you'll see the **Request Log**. This log shows every HTTP request going in and out of Fiddler. It will also show a status code of your request. You should receive a HTTP response code of 200, which means that the **PUT** request was successful.


Fiddler Web Debugger
File Edit Rules Tools View Help GET /book GeoEdge
Replay Go Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff MSDN Search...
Statistics Inspectors AutoResponder Composer Filters Log Timeline
Show Chart
[QuickExec] ALT+Q > type HELP...
Capturing All Processes

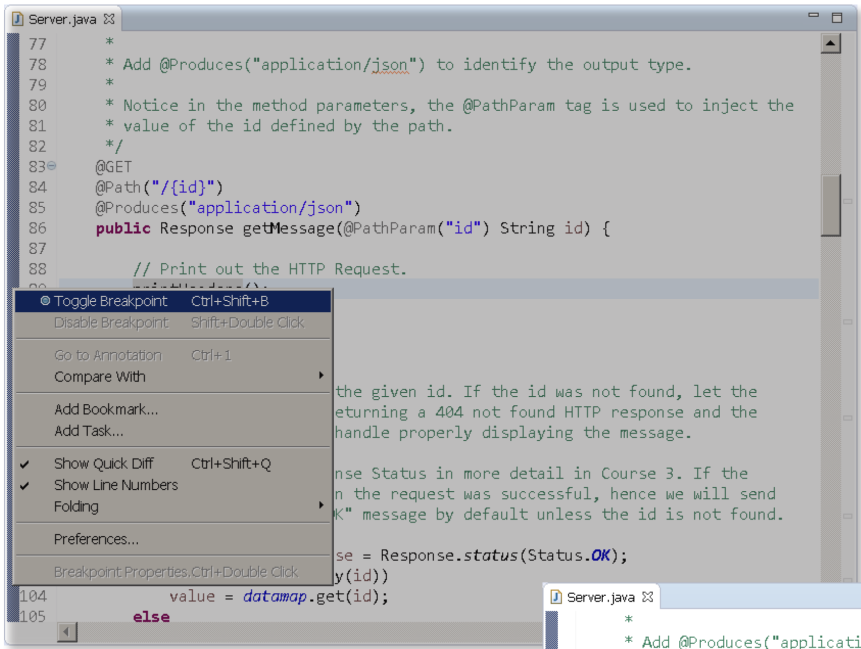
#	Result	Prot...	Host	URL	Body	Cachi...	C
1	200	HTTP	localhost:8080	/Course-2-Server/ap...	39		te

```
Request Count:   1
Bytes Sent:      104          (headers:104; body:0)
Bytes Received:  176          (headers:137; body:39)

ACTUAL PERFORMANCE
--------------
ClientConnected:       20:22:41.410
ClientBeginRequest:    20:22:41.410
GotRequestHeaders:     20:22:41.410
ClientDoneRequest:     20:22:41.410
Determine Gateway:     0ms
DNS Lookup:            0ms
TCP/IP Connect:        0ms
HTTPS Handshake:       0ms
ServerConnected:       20:22:41.410
FiddlerBeginRequest:   20:22:41.410
ServerGotRequest:      20:22:41.410
ServerBeginResponse:   20:22:41.415
GotResponseHeaders:    20:22:41.415
ServerDoneResponse:    20:22:41.415
ClientBeginResponse:   20:22:41.415
ClientDoneResponse:    20:22:41.415

       Overall Elapsed:       0:00:00.005

RESPONSE BYTES (by Content-Type)
--------------
 ~headers~: 137
text/plain: 39


ESTIMATED WORLDWIDE PERFORMANCE
--------------
The following are VERY rough estimates of download times when hitting servers based in
Seattle.

US West Coast (Modem - 6KB/sec)
       RTT:            0.10s
       Elapsed:        0.10s

Japan / Northern Europe (Modem)
       RTT:            0.15s
       Elapsed:        0.15s

China (Modem)
       RTT:            0.45s
       Elapsed:        0.45s

US West Coast (DSL - 30KB/sec)
       RTT:            0.10s
       Elapsed:        0.10s

Japan / Northern Europe (DSL)
       RTT:            0.15s
       Elapsed:        0.15s

China (DSL)
       RTT:            0.45s
       Elapsed:        0.45s
```

1 / 1 http://localhost:8080/Course-2-Server/api/values/test/Hello%20World

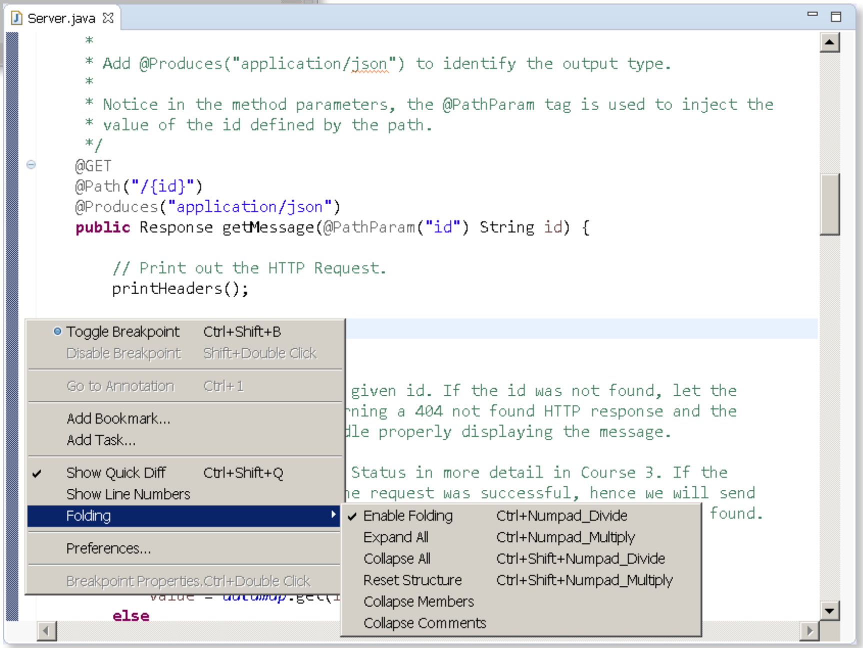
6. Add a Breakpoint into the REST API in Eclipse

In order to debug the REST API, you'll need to go back in to Eclipse. It is often useful to add pauses in the code, called **breakpoints**, while debugging. A **breakpoint** is a break in the execution of the application. It is a way to help developers isolate an issue. It stops the application execution when that specific line of code is about to be executed and you can see the state of your application, such as variable values, object references, etc. To add a **breakpoint**, click on the section of the code where you want to add one. Hit **Control-Shift-B** on your keyboard, or right-click on the line number, and select **Toggle Breakpoint**. Your screen should look similar to the screen shown. The blue dot indicates that a **breakpoint** was added. As long as you are running in debug mode, you can add and remove breakpoints without stopping and starting the server. In some circumstances, to catch a line of code that might only run in your application at startup, you would need to stop the server, add your breakpoint, and then restart the server. In our exercises, this will not be necessary.

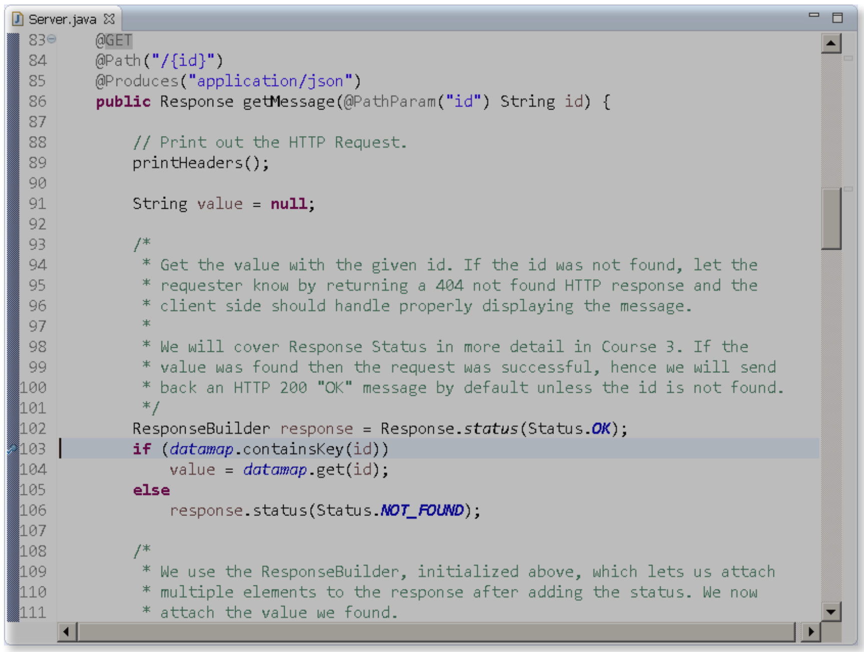

Server.java
Toggle Breakpoint Ctrl+Shift+B
Disable Breakpoint Shift+Double Click
Go to Annotation Ctrl+1
Compare With
Add Bookmark...
Add Task...
Show Quick Diff Ctrl+Shift+Q
Show Line Numbers
Folding
Preferences...
Breakpoint Properties.Ctrl+Double Click

```
     *
     * Add @Produces("application/json") to identify the output type.
     *
     * Notice in the method parameters, the @PathParam tag is used to inject the
     * value of the id defined by the path.
     */
    @GET
    @Path("/{id}")
    @Produces("application/json")
    public Response getMessage(@PathParam("id") String id) {

        // Print out the HTTP Request.

the given id. If the id was not found, let the
eturning a 404 not found HTTP response and the
handle properly displaying the message.

nse Status in more detail in Course 3. If the
n the request was successful, hence we will send
K" message by default unless the id is not found.

se = Response.status(Status.OK);
y(id))
            value = datamap.get(id);
        else
```

Tip: You can control displaying line numbers and folding by right-clicking in the white space to the left of your code and accessing the **Context** menu. Here you can turn on line numbers and "fold" (or hide) blocks of code. Folding is an extremely useful feature when you are working with large amounts of code.


Server.java
Toggle Breakpoint Ctrl+Shift+B
Disable Breakpoint Shift+Double Click
Go to Annotation Ctrl+1
Add Bookmark...
Add Task...
Show Quick Diff Ctrl+Shift+Q
Show Line Numbers
Folding
Preferences...
Breakpoint Properties.Ctrl+Double Click
Enable Folding Ctrl+Numpad_Divide
Expand All Ctrl+Numpad_Multiply
Collapse All Ctrl+Shift+Numpad_Divide
Reset Structure Ctrl+Shift+Numpad_Multiply
Collapse Members
Collapse Comments

```
     *
     * Add @Produces("application/json") to identify the output type.
     *
     * Notice in the method parameters, the @PathParam tag is used to inject the
     * value of the id defined by the path.
     */
    @GET
    @Path("/{id}")
    @Produces("application/json")
    public Response getMessage(@PathParam("id") String id) {

        // Print out the HTTP Request.
        printHeaders();

given id. If the id was not found, let the
rning a 404 not found HTTP response and the
dle properly displaying the message.

Status in more detail in Course 3. If the
e request was successful, hence we will send
found.
        else
```

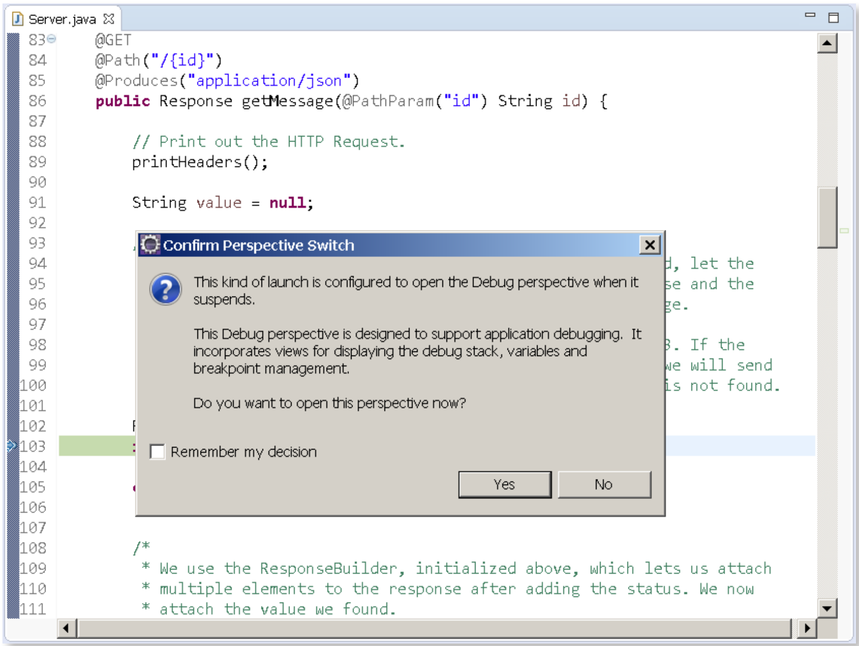


```
@GET
@Path("/{id}")
@Produces("application/json")
public Response getMessage(@PathParam("id") String id) {

    // Print out the HTTP Request.
    printHeaders();

    String value = null;

    /*
     * Get the value with the given id. If the id was not found, let the
     * requester know by returning a 404 not found HTTP response and the
     * client side should handle properly displaying the message.
     *
     * We will cover Response Status in more detail in Course 3. If the
     * value was found then the request was successful, hence we will send
     * back an HTTP 200 "OK" message by default unless the id is not found.
     */
    ResponseBuilder response = Response.status(Status.OK);
    if (datamap.containsKey(id))
        value = datamap.get(id);
    else
        response.status(Status.NOT_FOUND);

    /*
     * We use the ResponseBuilder, initialized above, which lets us attach
     * multiple elements to the response after adding the status. We now
     * attach the value we found.
```

a. Debugging Practice

Feel free to add a breakpoint wherever you'd like. One example, is to add a breakpoint to the GET method in Server.java where the key/value is checked to see if it exists in the datamap. Once you have your breakpoint in, issue the appropriate action in Fiddler to cause the server to execute the code. In this case, send a GET request to the URL: http://localhost:8080/Course-2-Server/api/values/test/, which should return the value we input for test (Hello World), earlier.

Once your breakpoint is reached, execution will stop on the server and Eclipse will ask you if you want to switch to the Debug Perspective. This is a set of screens optimized for walking through and debugging code. All of the tools are available from any of the other perspectives, but this perspective is optimized for debugging. Select **Yes** to continue.



Confirm Perspective Switch

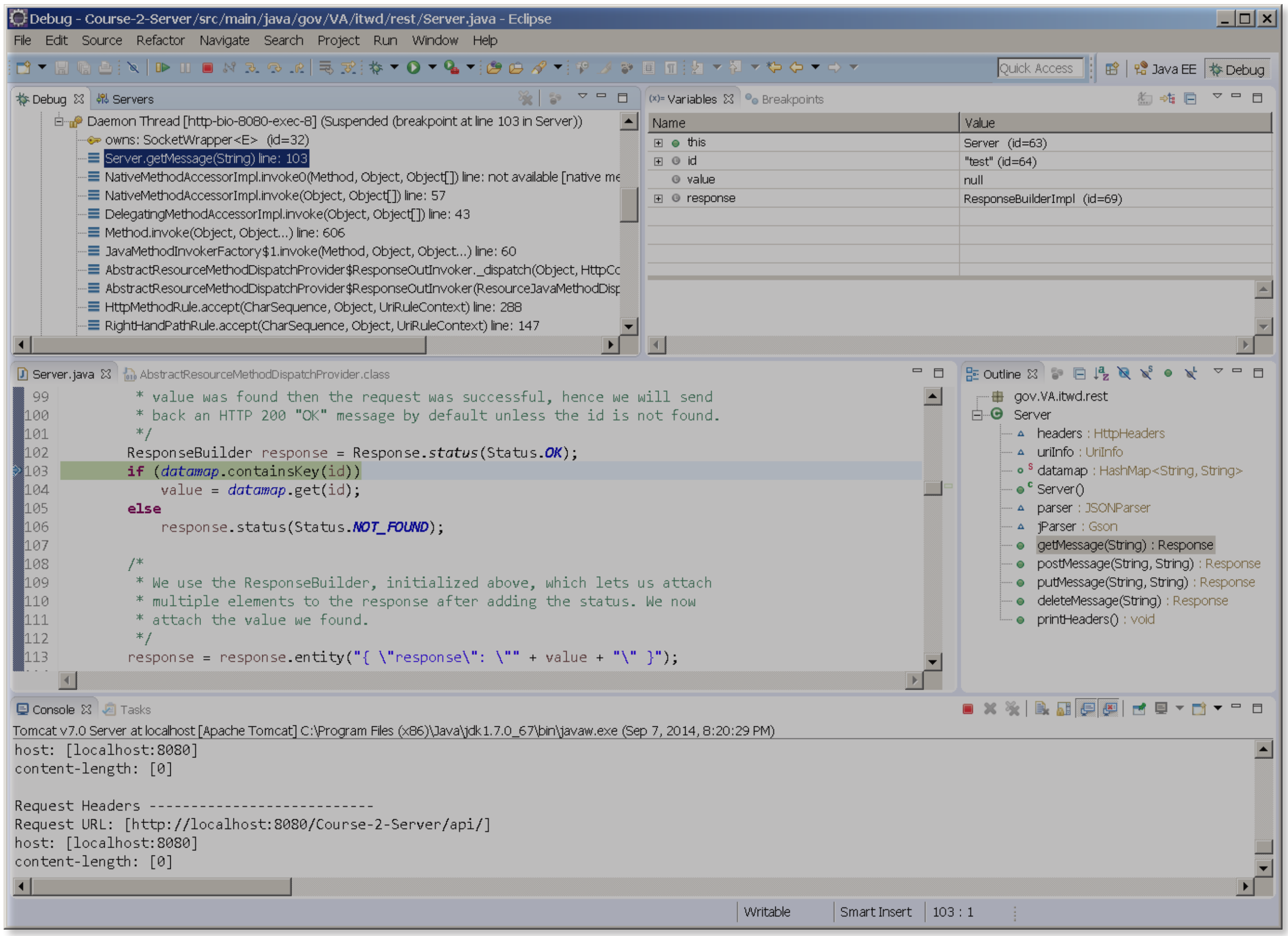
This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

☐ Remember my decision

This is what the default debug perspective looks like. Notice how the **Server.java** line 103 breakpoint is highlighted. Click on the line in the top left button to view the associated variables and values for the block.



Debug - Course-2-Server/src/main/java/gov/VA/itwd/rest/Server.java - Eclipse

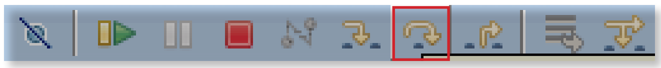
```
Daemon Thread [http-bio-8080-exec-8] (Suspended (breakpoint at line 103 in Server))
    owns: SocketWrapper<E>  (id=32)
    Server.getMessage(String) line: 103
    NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not available [native me
    NativeMethodAccessorImpl.invoke(Object, Object[]) line: 57
    DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: 43
    Method.invoke(Object, Object...) line: 606
    JavaMethodInvokerFactory$1.invoke(Method, Object, Object...) line: 60
    AbstractResourceMethodDispatchProvider$ResponseOutInvoker._dispatch(Object, HttpCo
    AbstractResourceMethodDispatchProvider$ResponseOutInvoker(ResourceJavaMethodDisp
    HttpMethodRule.accept(CharSequence, Object, UriRuleContext) line: 288
    RightHandPathRule.accept(CharSequence, Object, UriRuleContext) line: 147
```

Name	Value
this	Server (id=63)
id	"test" (id=64)
value	null
response	ResponseBuilderImpl (id=69)

```
 99          * value was found then the request was successful, hence we will send
100          * back an HTTP 200 "OK" message by default unless the id is not found.
101          */
102         ResponseBuilder response = Response.status(Status.OK);
103         if (datamap.containsKey(id))
104             value = datamap.get(id);
105         else
106             response.status(Status.NOT_FOUND);
107
108         /*
109          * We use the ResponseBuilder, initialized above, which lets us attach
110          * multiple elements to the response after adding the status. We now
111          * attach the value we found.
112          */
113         response = response.entity("{ \"response\": \"" + value + "\" }");
```

```
gov.VA.itwd.rest
Server
    headers : HttpHeaders
    uriInfo : UriInfo
    datamap : HashMap<String, String>
    Server()
    parser : JSONParser
    jParser : Gson
    getMessage(String) : Response
    postMessage(String, String) : Response
    putMessage(String, String) : Response
    deleteMessage(String) : Response
    printHeaders() : void
```

```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files (x86)\Java\jdk1.7.0_67\bin\javaw.exe (Sep 7, 2014, 8:20:29 PM)
host: [localhost:8080]
content-length: [0]

Request Headers --------------------------
Request URL: [http://localhost:8080/Course-2-Server/api/]
host: [localhost:8080]
content-length: [0]
```



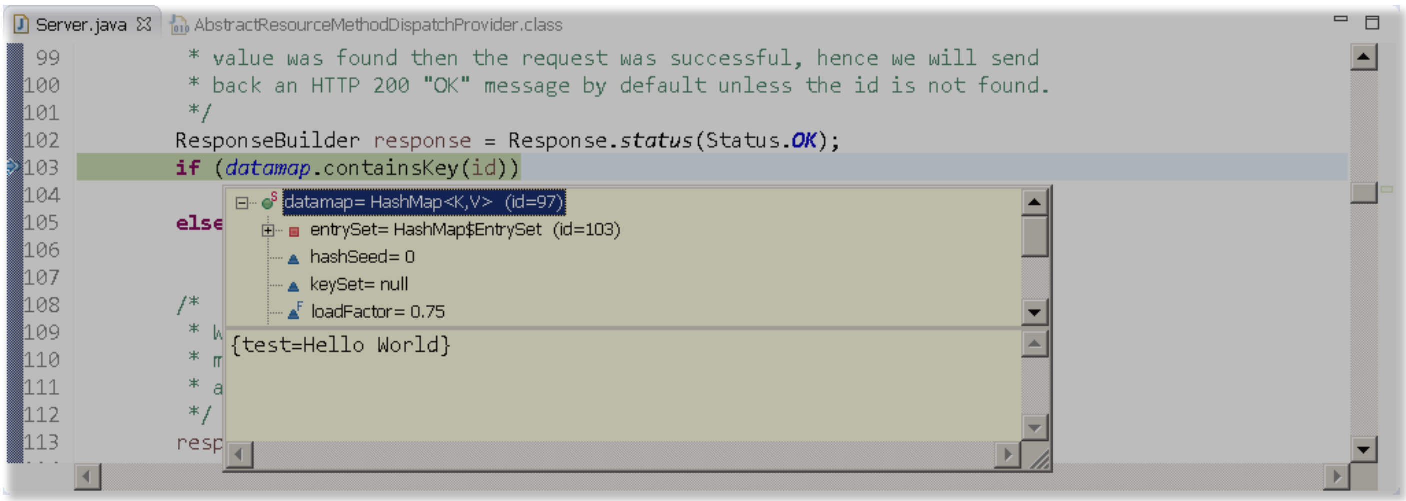
b. Stepping Through Your Code

Depending on the level of debugging you need to do, you can step over, step into, and step out of breakpoints from the debug control menu on the tool bar.

Let's use step over so we don't accidentally start stepping through JRE classes. The step over icon is the second yellow arrow, which looks like an upside down U. You can also use F6 on the keyboard Alternatively, you can stop your program (red square) or resume execution (green arrow).

Click the step over icon or use keyboard F6 to step through the code.

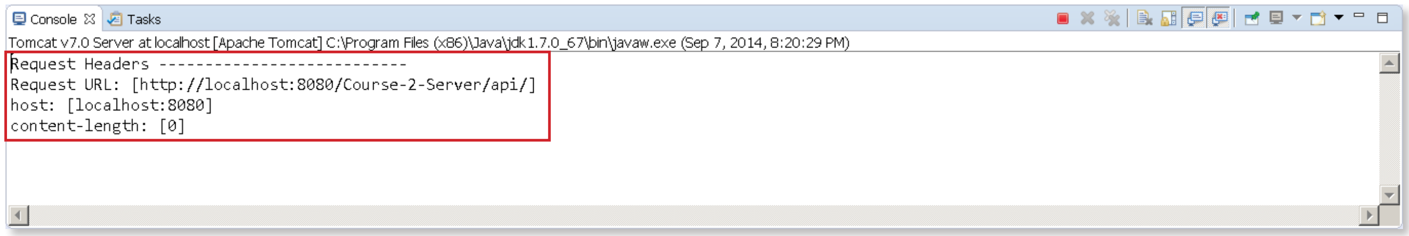
Tip: You can click and hover over objects and variables to see their values. Notice how the datamap contains our test value of **Hello World**.



```
        * value was found then the request was successful, hence we will send
        * back an HTTP 200 "OK" message by default unless the id is not found.
        */
       ResponseBuilder response = Response.status(Status.OK);
       if (datamap.containsKey(id))

       else


       /*
        * W
        * m
        * a
        */
       resp
```

```
datamap= HashMap<K,V>  (id=97)
  entrySet= HashMap$EntrySet  (id=103)
  hashSeed= 0
  keySet= null
  loadFactor= 0.75
{test=Hello World}
```

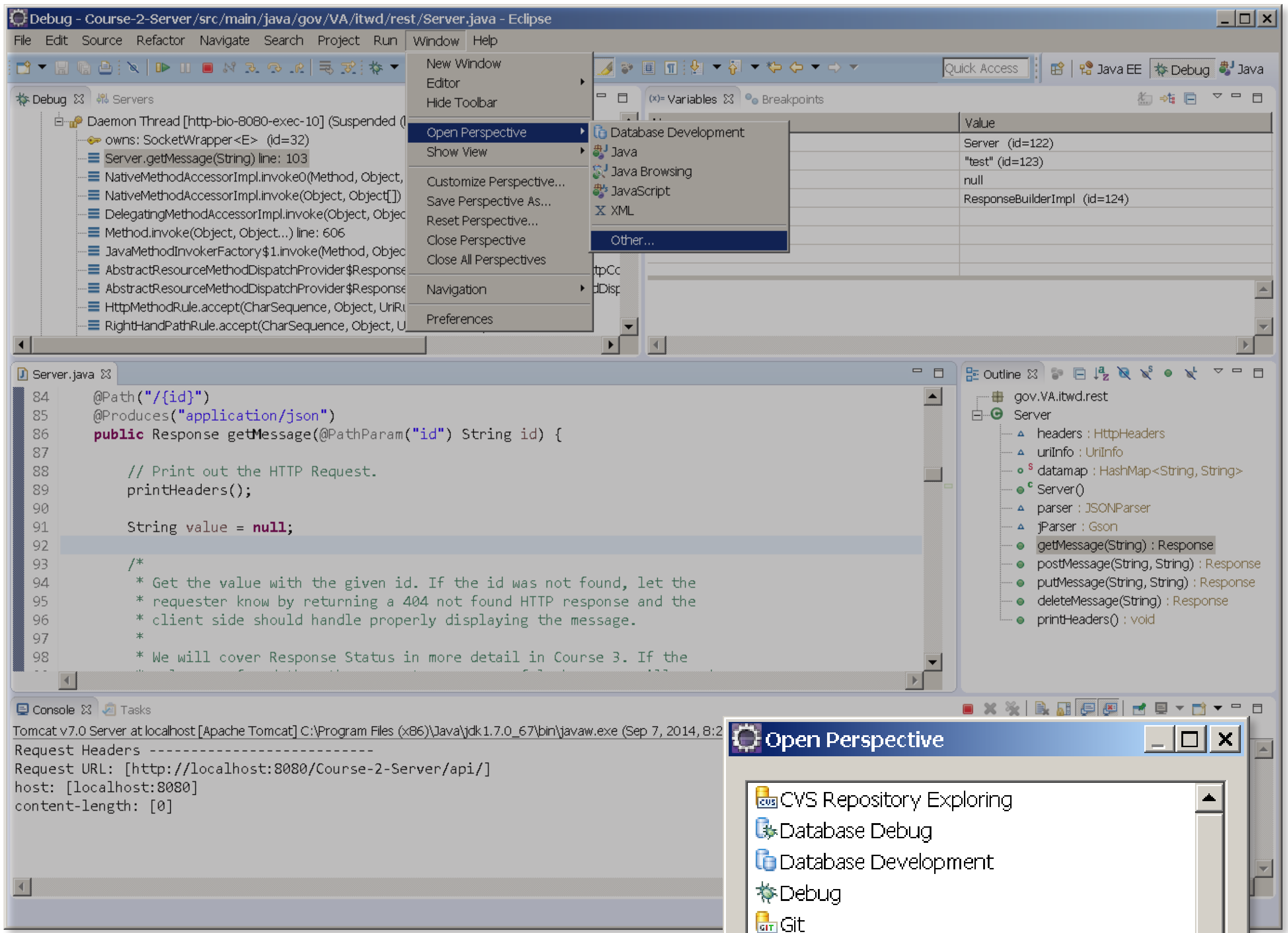
Tip: You can also keep an eye out on the console to see if any messages pop up. If you recall, we had a print headers method in our Server code to print out the headers. Here you can see the headers printed out in the console area.



```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files (x86)\Java\jdk1.7.0_67\bin\javaw.exe (Sep 7, 2014, 8:20:29 PM)
Request Headers ---------------------------
Request URL: [http://localhost:8080/Course-2-Server/api/]
host: [localhost:8080]
content-length: [0]
```

7. Select the Java Perspective

Once you are done with debugging, you can go back to the standard Java EE perspective. To do this, click on **Window** in the menu at the top, go to **Open Perspective**, and select **Other**. The perspective that we've been using is the Java EE perspective, which is optimized for enterprise application development, such as working with RESTful web services. Select Java EE and click **OK** to return to the default view.



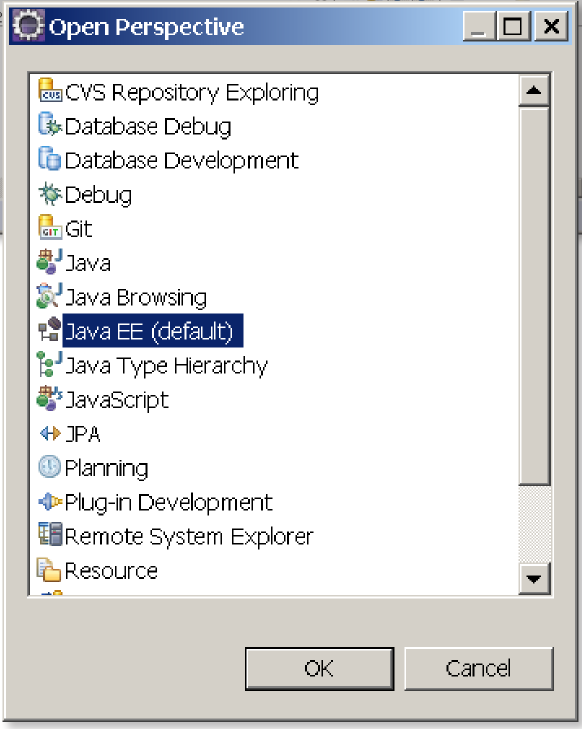
```
Daemon Thread [http-bio-8080-exec-10] (Suspended (
    owns: SocketWrapper<E>  (id=32)
    Server.getMessage(String) line: 103
    NativeMethodAccessorImpl.invoke0(Method, Object,
    NativeMethodAccessorImpl.invoke(Object, Object[])
    DelegatingMethodAccessorImpl.invoke(Object, Objec
    Method.invoke(Object, Object...) line: 606
    JavaMethodInvokerFactory$1.invoke(Method, Objec
    AbstractResourceMethodDispatchProvider$Response
    AbstractResourceMethodDispatchProvider$Response
    HttpMethodRule.accept(CharSequence, Object, UriRu
    RightHandPathRule.accept(CharSequence, Object, U
```

```
Server  (id=122)
"test"  (id=123)
null
ResponseBuilderImpl  (id=124)
```

```
    @Path("/{id}")
    @Produces("application/json")
    public Response getMessage(@PathParam("id") String id) {

        // Print out the HTTP Request.
        printHeaders();

        String value = null;

        /*
         * Get the value with the given id. If the id was not found, let the
         * requester know by returning a 404 not found HTTP response and the
         * client side should handle properly displaying the message.
         *
         * We will cover Response Status in more detail in Course 3. If the
```

```
gov.VA.itwd.rest
  Server
    headers : HttpHeaders
    uriInfo : UriInfo
    datamap : HashMap<String, String>
    Server()
    parser : JSONParser
    jParser : Gson
    getMessage(String) : Response
    postMessage(String, String) : Response
    putMessage(String, String) : Response
    deleteMessage(String) : Response
    printHeaders() : void
```

```
Tomcat v7.0 Server at localhost [Apache Tomcat] C:\Program Files (x86)\Java\jdk1.7.0_67\bin\javaw.exe (Sep 7, 2014, 8:2
Request Headers ---------------------------
Request URL: [http://localhost:8080/Course-2-Server/api/]
host: [localhost:8080]
content-length: [0]
```



Tip: You can customize perspectives and experiment with different ones. If you want to return to the default view, you can always restore the perspective to the original settings from the **Windows** menu by selecting **Restore Perspective**.

Now you have the basic knowledge necessary to debug RESTful applications. Feel free to experiment with Eclipse, Tomcat, and Fiddler doing different combinations and scenarios. Try sending a GET for a value that doesn't exist, or a PUT for a key that exists, or a DELETE on a key that doesn't exist to help you get a better understanding of how these tools can be used to debug applications.

Additional Resources

This section of the guidebook provides you with resources mentioned in the REST course. Several are websites and the others can be found in the Books 24x7 catalog of the Talent Management System (TMS).

Websites

The One-VA Technical Reference Manual (VA TRM)

http://trm.oit.va.gov/TRMHomePage.asp **Note:** This link is only available on VA's Intranet.

Search this site to stay within guidelines for REST vendors and guidelines that have been vetted and approved by VA. This site is also available to search hardware, software, testing tools, applications, etc., to see if that tool is approved, approved with constraints, or unapproved at VA. Do your research here before you start using tools to develop REST APIs.

Eclipse

To download Eclipse to your personal computer visit, https://www.eclipse.org/

Eclipse Reference Websites

Visit these user forums for references on how to use REST with Eclipse:

http://marketplace.eclipse.org/content/rest-client
http://wiki.eclipse.org/EclipseLink/Examples/REST/GettingStarted/RestService

Tomcat 7.0

To download Tomcat 7.0 to your personal computer, visit, http://www.coreservlets.com/Apache-Tomcat-Tutorial/tomcat7-files/tomcat-7.0.34-preconfigured.zip.

1. Unzip the the downloaded file into the root folder of the C drive (C:\apache-tomcat-7.0.34).
2. In Eclipse, go to **Open Window,** down to **Preferences,** and select **Server** and then **Installed Runtimes** to create a Tomcat installed runtime.
3. Click **Add** to open the **New Server Runtime** dialog.
4. Select your runtime under **Apache (v7.0).**
5. Click **Next**.
6. Under **Tomcat Installation Directory,** insert the path to your Tomcat installation (For example: C:\apache-tomcat-7.0.34).
7. Click **Finish.**

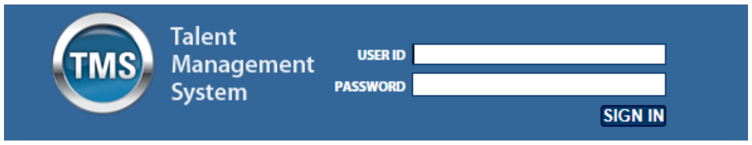
Books 24x7

There are several electronic books about developing REST APIs available to you in Books 24x7 on the TMS (TMS ID 30086). The list of books includes:

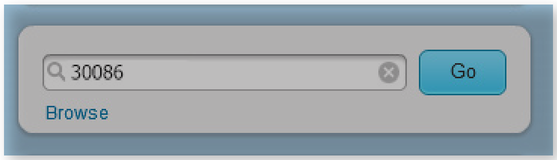
- *Cloud Optimized REST API Automation Framework*
- *The Agile Architecture Revolution: How Cloud Computing, REST-Based SOA, and Mobile Computing Are Changing Enterprise IT*
- *RESTful PHP Web Services: Learn the Basic Architectural Concepts and Steps Through Examples of Consuming and Creating RESTful Web Services in PHP Android Application Development for Dummies*

Five Steps for Accessing Books 24x7

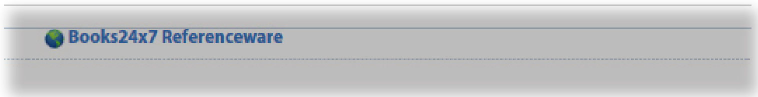
1. Log in to the TMS at https://www.tms.va.gov.



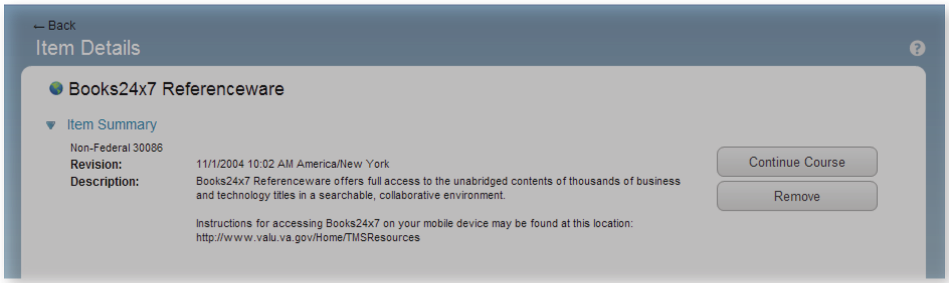
2. Enter TMS ID 30086 in the **Browse** text box and select **Go**.



3. Select the **Books 24x7 Referenceware** title from the Catalog Search Results.



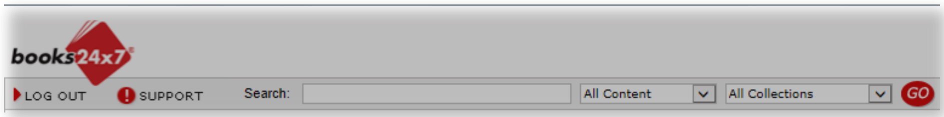
4. Select **Continue Course** (If this is the first time you have accessed Books 24x7, the button will read Start Course).



Non-Federal 30086
Revision: 11/1/2004 10:02 AM America/New York
Description: Books24x7 Referenceware offers full access to the unabridged contents of thousands of business and technology titles in a searchable, collaborative environment.

Instructions for accessing Books24x7 on your mobile device may be found at this location: http://www.valu.va.gov/Home/TMSResources

5. Enter your search terms (e.g., "REST, API") in the **Search** text box and select the **Go** button.


books24x7
LOG OUT
SUPPORT
Search:
All Content
All Collections
GO