University of Central Lancashire
UCLan

CO3409 Distributed Enterprise Systems

Lab : RESTful Webservices

Summary

Create an application that provides a RESTful Web Service, based around a customer database and test it with a simple test.

Purpose

On completion of this you should be able to

- Use NetBeans to create a RESTful Web Service
- Use NetBeans to create classes from a database.

Things to Remember

Some things to remember...

- Run Netbeans as "Administrator".
- When running your application, if you are using a database please ensure your database is running first (see previous lab worksheets).
- To deploy an Enterprise Application, I suggest using Clean and Build and then Deploy.

Activities

1. Creating and testing a Web Service using the NetBeans IDE

cCarry out the practical in https://netbeans.org/kb/docs/websvc/rest.html. Leave the section "Using a MySQL Database Server" because we'll be using JavaDB (Derby). If you wish to use MySQL database, then you can do that (**in your own time**). Use Ant not Maven, which doesn't support the auto-generated test. If you do use Maven, test the REST service with the client service in Part 2.

> **Notes:**
>
> - During the practical, if you can't find jdbc/sample, then right-click on JavaDB, select Properties. Note the Database Location. Download the sample zip file from Module Materials on Blackboard and extract the sample folder into the Database Location (e.g. `C:\Users\dgcampbell\derby`).
> - You should be able to right-click on JavaDB under Services and select Create Sample Database. Call it "sample". However, the sample database has a format that won't work with NetBeans.
> - When performing "Testing the RESTFul service", ignore the default page from "Test RESTful Web Services". Instead, browse to http://localhost:8080/WebServicesTest/test-resbeans.html
> - If you get a virtually blank screen, change the default browser in the test project **properties** (**Run** page). Chrome appears to work well. Try to retrieve the data in XML format first. To check the data provided through REST, you can see the data in the customer table by connecting to the samples database, expanding to the tables folder and right-clicking on the customer table and choosing View data. Try to retrieve the data in JSON format. If you get a status of 500 – Internal server error and a stackoverflow error message, switch to the payara tab and the beginning of the exception message, copy some of the key words and see if you can find the fix through Google. If not, add the annotation, `@JsonbTransient`, to the fields in `Customer.java` that are also annotated with `@ManyToOne`.
> - If you get an error that WADL is not being generated, check that you have an `ApplicationConfig.java` in your entities.services folder – the folder containing your façade classes. If not create an ApplicationConfig class in that package, and enter the following:
>
> ```
> package entities.service;
>
> import java.util.Set;
> import javax.ws.rs.core.Application;
>
> @javax.ws.rs.ApplicationPath("webresources")
> public class NotApplicationConfig extends Application {
>
>     @Override
>     public Set<Class<?>> getClasses() {
>         Set<Class<?>> resources = new java.util.HashSet<>();
> ```

```
        addRestResourceClasses(resources);
        return resources;
    }

    /* Do not modify addRestResourceClasses() method.
     * It is automatically populated with
     * all resources defined in the project.
     * If required, comment out calling this method in getClasses().
     */
    private void addRestResourceClasses(Set<Class<?>> resources) {
        resources.add(entities.service.ItemFacadeREST.class);
        resources.add(entities.service.UsersFacadeREST.class);
    }
}
```

- If you have problems re-building the application because the output file can't be overwritten. You can try deleting the application from the server (in the Services tab) and try to rebuild and deploy it again.

Understanding the application

Answer the following questions surround the RESTful Web Service you have created.

Background

1. The introduction to the NetBeans tutorial refers to JSR 311. What is a JSR?
2. Jersey is a reference implementation of JSR 311. What is a reference implementation?

Object Relational Model

1. What is an entity class?

Generating a RESTful Service

1. What is a container resource?
2. Explore the generated classes. Can you see how they were created?
3. What does "generating the resource representation from the corresponding entity instance" mean?

Testing the RESTful Serivce

1. Explore both XML and JSON output – try to explain how the data is represented.
2. Why is a GET annotation used with the find method in `CustomerFacade`?
3. What is the purpose of the annotation: `@ProduceMime("text/html")` or `@Produces("text/html")`

2. Creating a Client

1. In `WebServicesTest`, create a servlet in a `restaccess` package.
2. Copy `Customer.java`, `DiscountCode.java` and `MicroMarket.java` from `CustomerDB` into `restaccess`. Ensure the package name in the java files is updated to reflect the new package.
3. Download into a suitable folder the following jars, jackson-databind and iackson-core, from here.
4. Right-click on the WebServicesTest project, select properties and use the libraries branch to add the downloaded jars to the library.
5. Right-click on the WebServicesTest project, use **New**, **Other** to create a **RESTful Java Client** from the **Web Services** branch. Call it `JerseyClient` and put it in the `restaccess` package. Browse to the `CustomerDB` project and select the `CustomerFacadeREST` REST API.
6. Create a Servlet called `RESTTest` in `restaccess`. Add the following code before `out.println("</body>");` :

```
            try {
                JerseyClient client = new JerseyClient();
                String res = client.countREST();
                out.println("<p>Count REST " + res + "</p>");

                res = client.findAll_XML(String.class);
                res = res.replaceAll("&", "&amp;");
                res = res.replaceAll("<", "&lt;");
                res = res.replaceAll(">", "&gt;");

                res = res.replaceAll("\"", "&quot;");
                res = res.replaceAll("\'", "&apos;");

                out.println("<p>All XML <code>" + res + "</code></p>");

                res = client.findAll_JSON(String.class);
                out.println("<p>All JSON <code>" + res + "</code></p>");

                ObjectMapper om = new ObjectMapper();

                int max = 0;
                Customer[] cs = om.readValue(res, Customer[].class);
                if (cs == null) {
                    out.println("<p>no customers</p>");
                } else {

                    for (int i = 0; i < cs.length; i++) {
                        if (max < cs[i].getCustomerId()) {
                            max = cs[i].getCustomerId();
                        }
                        out.println("<p>Customer " + i + "   " + cs[i].getName()
                                + "   " + cs[i].getState() + "</p>");
```

```
                    }
                }

                if (cs.length > 5)
                    for (int i = 0; i < cs.length; i++) {
                        client.remove(""+cs[i].getCustomerId());
                        out.println("<p>Customer " + i + "  " + cs[i].getName()
                                + " deleted </p>");
                    }


                Customer c = new Customer();

                max++;
                c.setCustomerId(max);
                c.setName("Chris" + max);
                c.setState("missouri");
                c.setAddressline1("1 the street");
                String jr = om.writeValueAsString(c);

                client.create_JSON(c);

                out.println("<p>new customer created: "+ c.getName()+"</p>");

                res = client.findAll_JSON(String.class);
                cs = om.readValue(res, Customer[].class);
                if (cs == null) {
                    out.println("<p>no customers</p>");
                } else {
                    for (int i = 0; i < cs.length; i++) {
                        out.println("<p>Customer " + i + "  " + cs[i].getName()
                                + "  " + cs[i].getState() + "</p>");
                    }
                }

                client.close();
            } catch (Exception e) {
                out.println("<h2>Exception Thrown</h2>");
                out.println("<p>" + e.getMessage() + "</p>");
            }
```

7. Build and run the project.
8. Explain the code. You will need to look up the Jackson project, starting at https://github.com/FasterXML/jackson.
9.
10.

3. Another Example

https://blog.payara.fish/create-a-restful-web-service-with-payara-server-netbeans