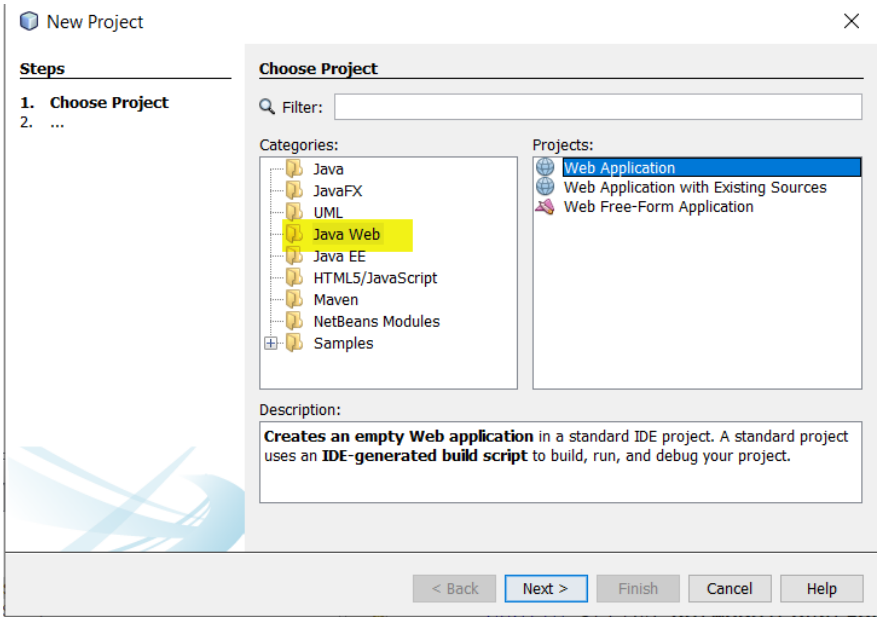
Tutorial – Using EJB as a Jax-RS Web service

In this tutorial we'll be developing an extremely simple web service to cater a hypothetical scenario. What we will be developing is a simple service that takes in a vehicles registration number and based on the first few characters (CAA,NA,PD etc..) determine vehicle category and inform the end user the category of driver's license needed to drive that vehicle. For simplicities sake and since the point of the exercise is merely to further explain underlying concepts, we will not be using any database connections etc. and will purely use hardcoded values.

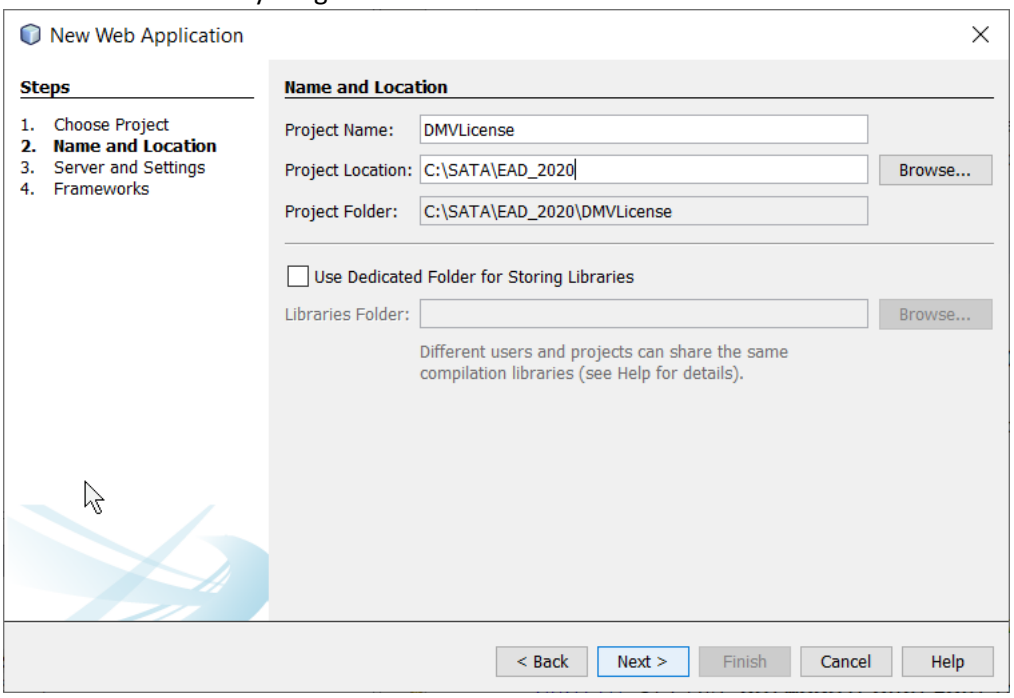
Perquisites : Netbeans 8.2, Postman.

The first step is to fire up Netbeans IDE 8.2

Create a new project – type should be web application.

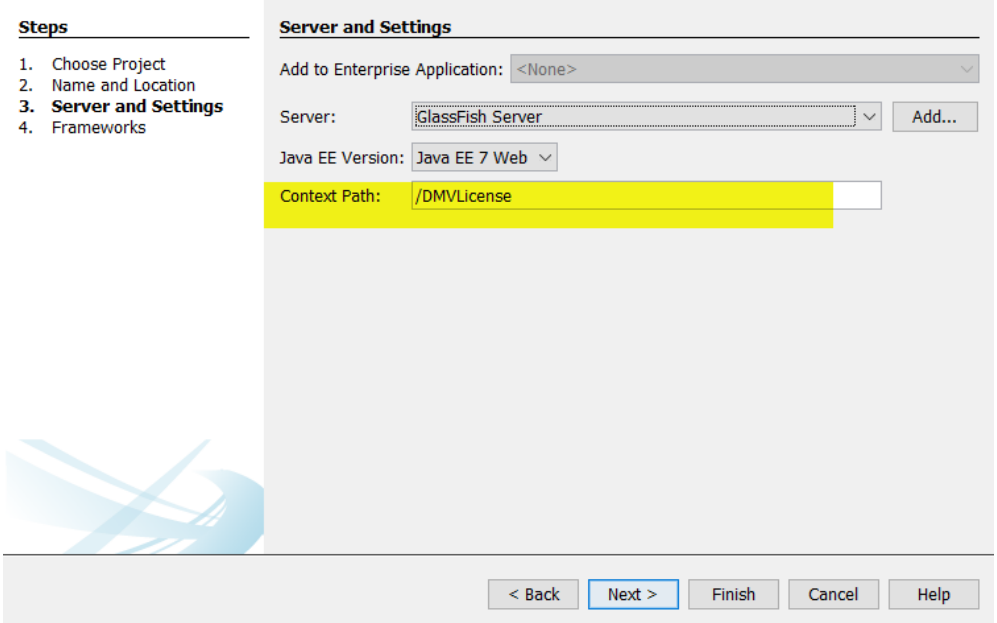


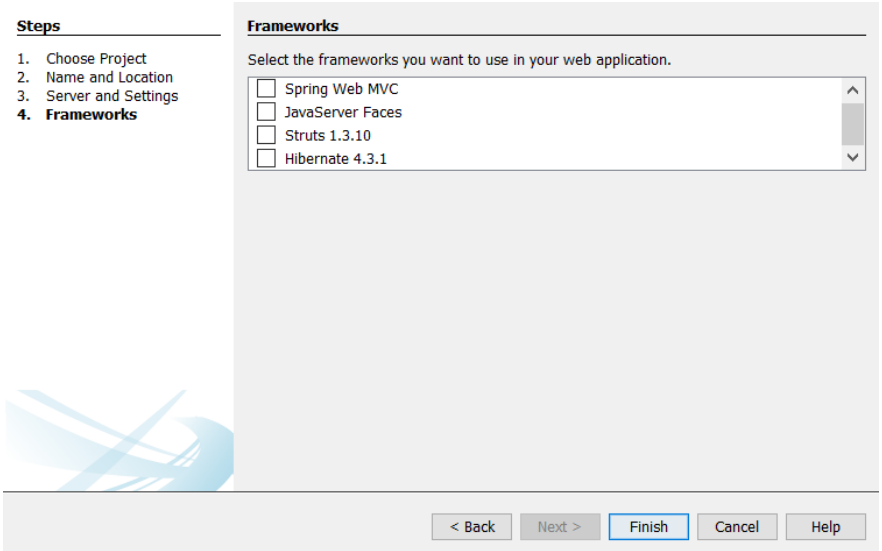
We can name this anything



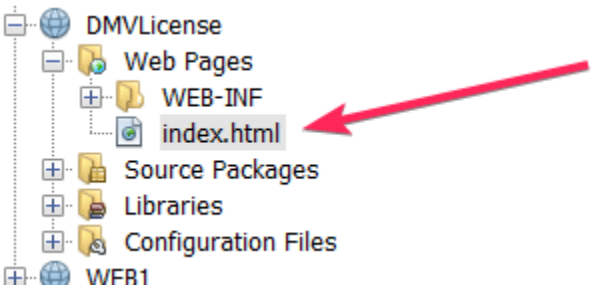
Pay attention to the context path -this will be part of the URL of the service.

Keep the defaults for the next few steps in the Wizard.





Once the project is created locate the index.html file where we will be adding a text box and a submit button.



The following html shows the creation of a simple text box and a submit button that we will be using to submit the Vehicle registration number.

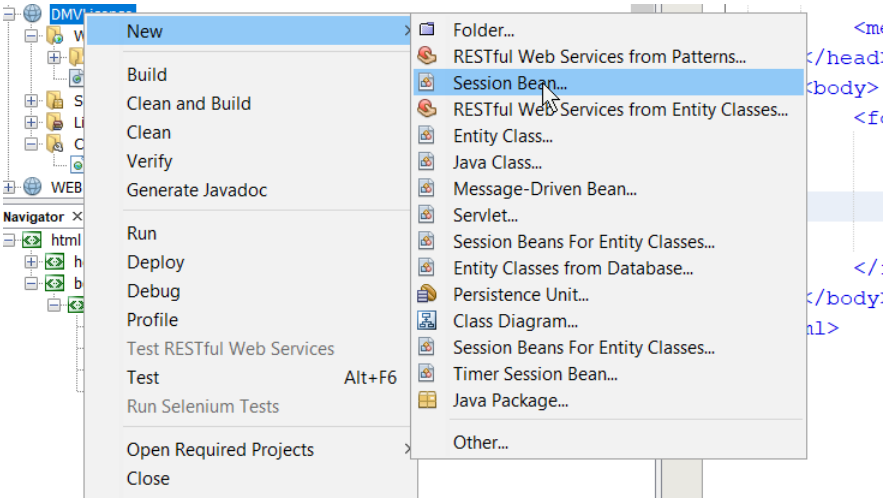
```
<html>
    <head>
        <title>DMV License Category Service</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    </head>
    <body>
        <form action="/DMVLicense/webresources/scanplate"
method="POST">
            Model <input name="regnum" value="CAA1234">
            <br>
            <input type="submit">
        </form>
```

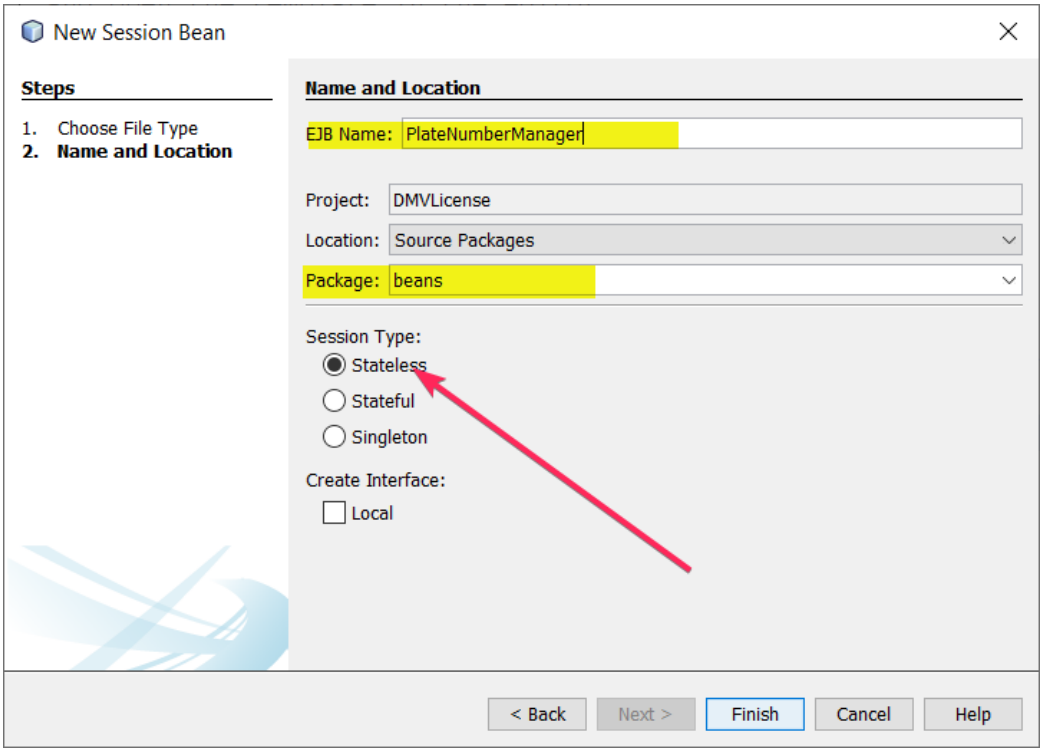
```
    </body>
</html>
```

Vehicle Registration Number : CAA1234

Submit

Once the simple index html page is done, we will create a new session bean – on the project node select new Session bean. Give it a name – it's not important what you call it. For clarities sake we can keep this in a package called 'beans'



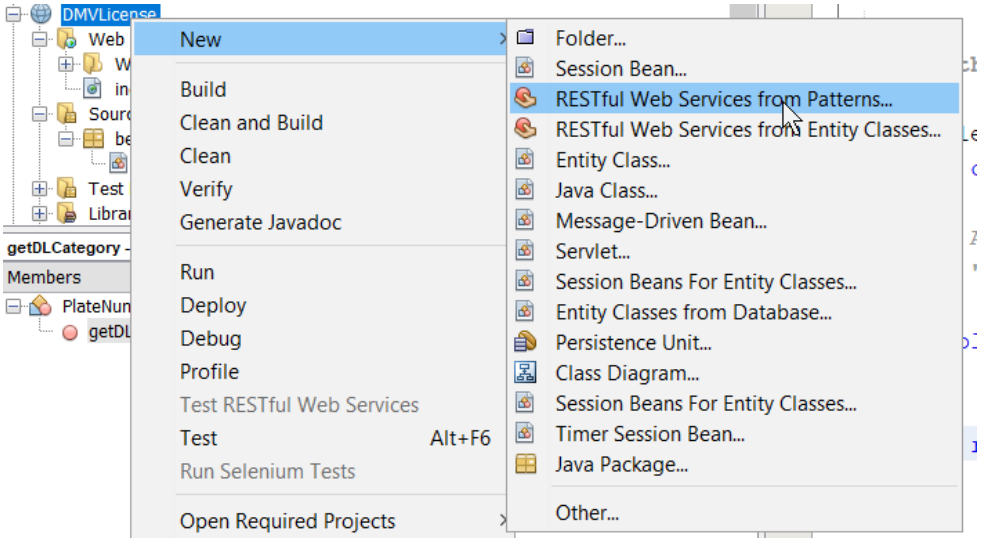


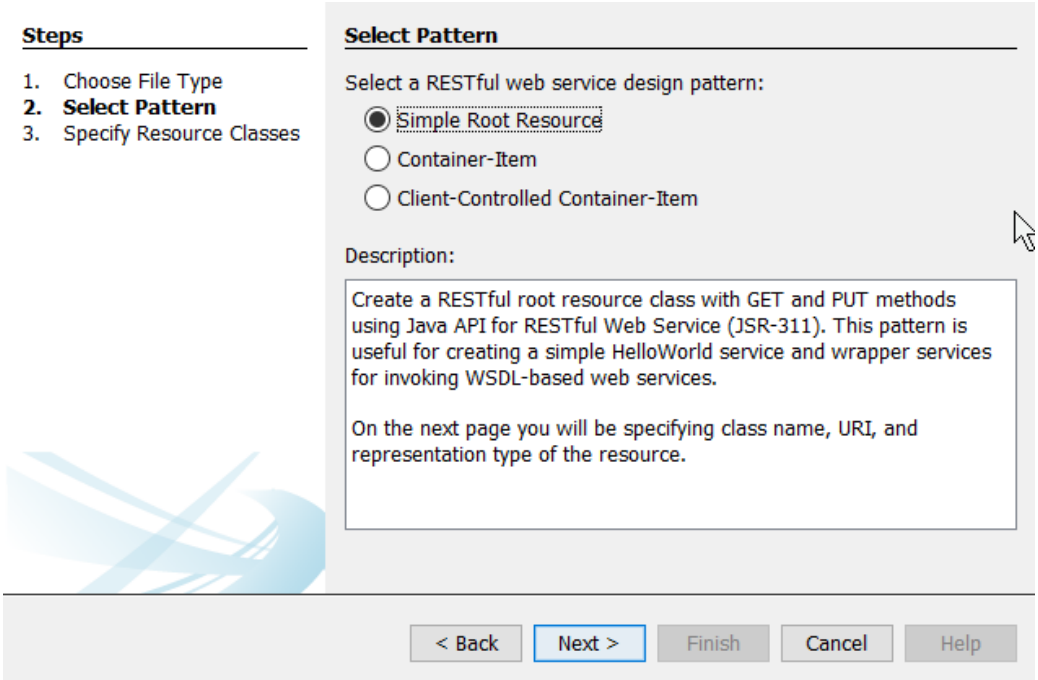
The session bean will be kept as stateless. Inside the session bean we'll be adding the business method/ business logic – the following method accepts the registration number of the vehicle and based on the first character decides what category it is and what license type is required to drive it.

```
public String getDLCategory(String regnum)
    {
       String dlcategory="Unknown Licence Category";
       switch(Character.toUpperCase(regnum.charAt(0))) {
        case 'C':
        case 'K':
        case 'P':
        case 'D':
            dlcategory="Light Vehicle";
            break;
        case 'L':
        case 'N':
            dlcategory="Heavy Vehicle";
            break;
        case 'M':
        case 'B':
            dlcategory="Motor Bicycle";
            break;
        case 'Y':
        case 'A':
            dlcategory="Three Wheeler";
            break;
       }
     return dlcategory;
```

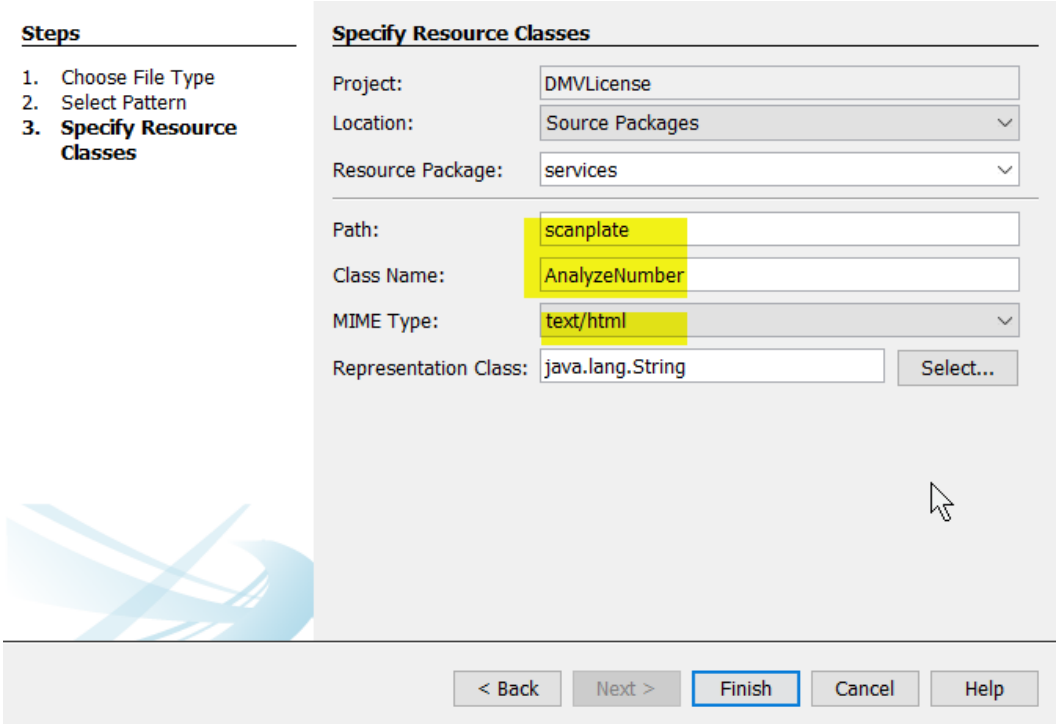
```java
@Stateless
public class PlateNumberManager {

    // Add business logic below. (Right-click in editor and choose
    // "Insert Code > Add Business Method")

    public String getDLCategory(String regnum)
    {
        String dlcategory="Unknown Licence Category";
        switch(Character.toUpperCase(regnum.charAt(0))) {
         case 'C':
         case 'K':
         case 'P':
         case 'D':
             dlcategory="Light Vehicle";
             break;
         case 'L':
         case 'N':
             dlcategory="Heavy Vehicle";
             break;
         case 'M':
         case 'B':
             dlcategory="Motor Bicycle";
             break;
```

Next we can add a new Restful web service.





We will use a new package called services to keep this – we can give a path (this will be appended to the service URL) the class name can be anything basically. Change the MIME type to text html.



```java
import java.util.Set;
import javax.ws.rs.core.Application;

/**
 *
 * @author SaTaLK
 */
@javax.ws.rs.ApplicationPath("webresources")
public class ApplicationConfig 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(services.AnalyzeNumber.class);
    }
```

Two new classes are created – the application config class will have some information pertaining to the path etc. also observe the path you provided in the previous dialog box is appended to the @Path annotation in the AnalyzeNumber class we created.

```java
 *
 * @author SaTaLK
 */
@Path("scanplate")
public class AnalyzeNumber {

    @Context
    private UriInfo context;

    /**
     * Creates a new instance of AnalyzeNumber
     */
    public AnalyzeNumber() {
    }

    /**
     * Retrieves representation of an instance of services.AnalyzeNumber
     * @return an instance of java.lang.String
     */
    @GET
    @Produces(MediaType.TEXT_HTML)
    public String getHtml() {
        //TODO return proper representation object
        throw new UnsupportedOperationException();
    }
```

The annotated path will direct us to this class and depending on the action defined in the index.html page it will call the relevant method with either @GET or @POST Annotations

```java
 * @author SaTaLK
 */
@Path("scanplate")
public class AnalyzeNumber {

    @Context
    private UriInfo context;
    @EJB
    PlateNumberManager pnm;
    /**
     * Creates a new instance of AnalyzeNumber
     */
    public AnalyzeNumber() {
    }

    /**
```

Since we will be calling the business method in the beans class PlateNumberManager we will need to create an instance of that class with the @EJB annotation

```
@EJB
    PlateNumberManager pnm;
```

Now since we defined the action as 'POST' in the index.html we will get rid of the @PUT annotated method and instead write a method with @POST annotation

```
@POST
    @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
    @Produces(MediaType.TEXT_HTML)
public String getCategoryPost(@FormParam("regnum") String
registration)
    {
        return "<h2> Required Driving License Category :
"+pnm.getDLCategory(registration)+"</h2>";

    }
```

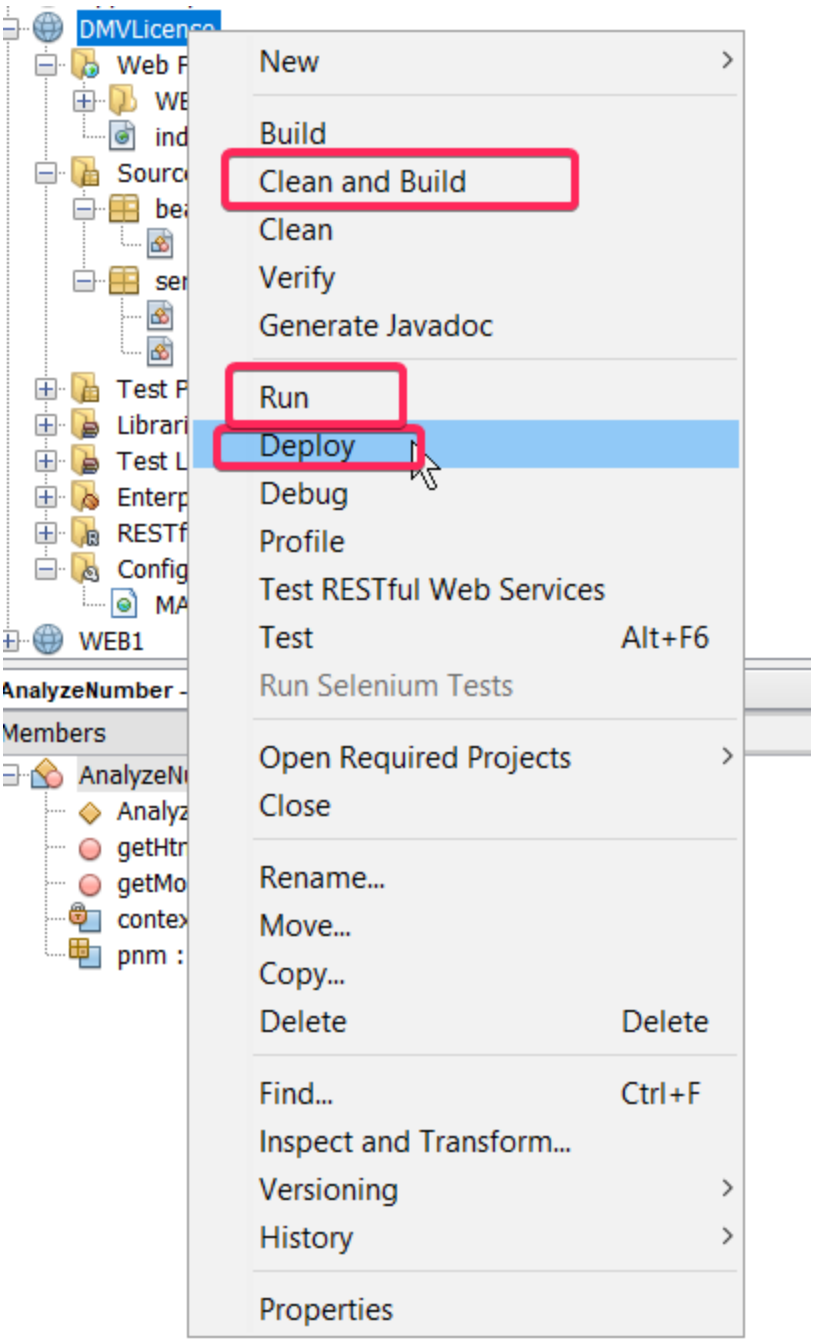
Make sure to fix the imports as follows

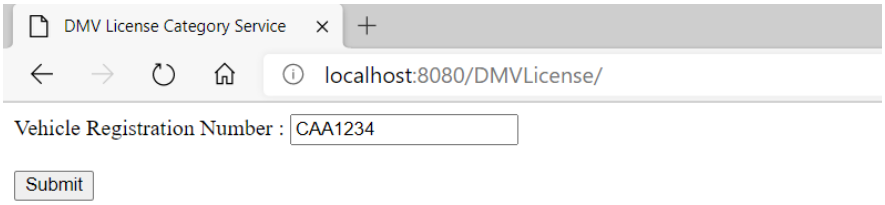
```
46      }
47
48      //POST METHOD TO CAPTURE THE INPUT FROM THE HTML FORM
        @POST
50      Add import for javax.ws.rs.HttpMethod.POST                    ODED)
51      Add import for javax.ws.rs.POST
        Add import for javax.swing.text.html.FormSubmitEvent.MethodType.POST
        Add import for com.sun.mail.imap.Rights.Right.POST            "regnum") String registration)
53      {
54          return "<h2> Required Driving License Category : "+pnm.getDLCategory(registration)+"</h2>";
55
56      }
57  }
58
```

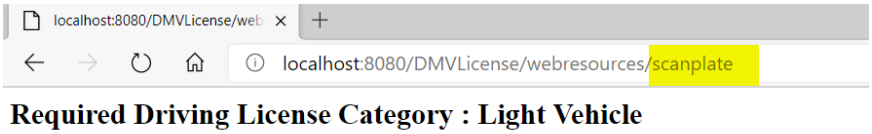
```
//POST METHOD TO CAPTURE THE INPUT FROM THE HTML FORM
    @POST
    @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
    @Produces(MediaType.TEXT_HTML)
    public String getCategoryPost(@FormParam("regnum") String registration)
    {
        return "<h2> Required Driving License Category : "+pnm.getDLCategory(registration)+"</h2>";

    }
```

Now we're basically calling the post action and invoking the beans Business method.

Click on your project clean and build and then ‘Run’







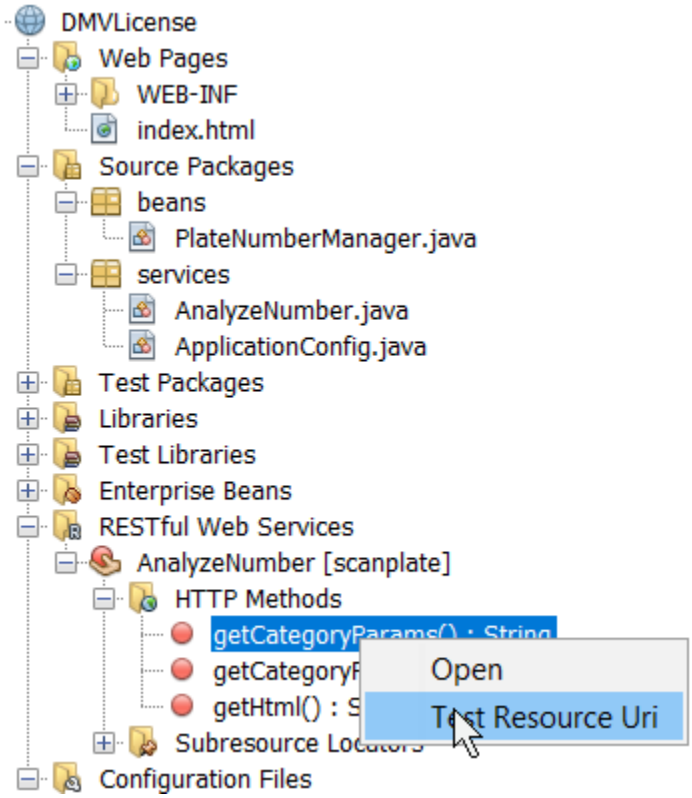
Next up we can explore other options of passing parameters.

We can start by getting the registration number as a query parameter

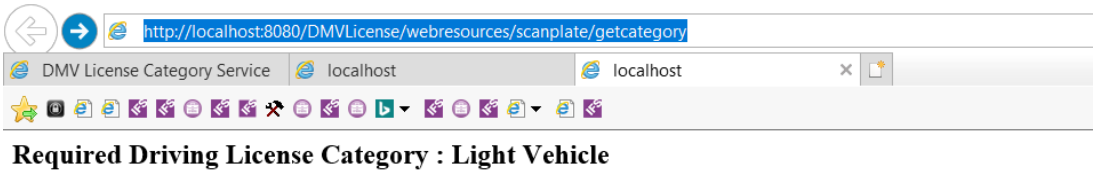
```
@GET
    @Path("getcategory")
     @Produces(MediaType.TEXT_HTML)
    public String
getCategoryParams(@QueryParam("regnum")@DefaultValue("CBE1234")
String registration){
         return "<h2> Required Driving License Category :
"+pnm.getDLCategory(registration)+"</h2>";
    }
```

```
@GET

    @Path("getcategory")

     @Produces(MediaType.TEXT_HTML)

    public String
getCategoryParams(@QueryParam("regnum")@DefaultValue("CBE1234") String
registration){

         return "<h2> Required Driving License Category :
"+pnm.getDLCategory(registration)+"</h2>";

    }
```

Notice that since this is a 'GET' method NetBeans allows us to test this method via the navigator



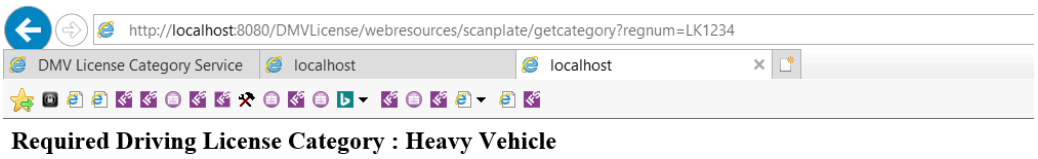
This will call the method via the browser – but without query parameters so it will run with the default value we provided.



to pass in the query parameter we can modify this URL to below

http://localhost:8080/DMVLicense/webresources/scanplate/getcategory?regnum=LK1234

note that we need to provide the correct query parameter name (in our case regnum)



As per our logic L falls into the 'Heavy Vehicle Category' and is correctly fetched.

One more option we can try out is to use path variables

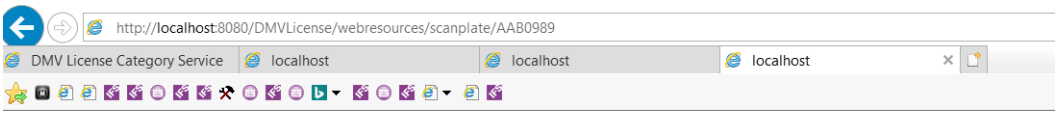
Add the following method to the AnalyzeNumber.java file

```
@GET
    @Path("{regnum}")
    @Produces(MediaType.TEXT_HTML)
    public String getCategoryPath(@PathParam("regnum")String
registration)
    {
         return "<h2> Required Driving License Category :
"+pnm.getDLCategory(registration)+"</h2>";

    }
```

```
    }
//GET METHOD WITH PATH VARIABLES
    @GET
    @Path("{regnum}")
    @Produces(MediaType.TEXT_HTML)
    public String getCategoryPath(@PathParam("regnum")String registration)
    {
        return "<h2> Required Driving License Category : "+pnm.getDLCategory(registration)+"</h2>";

    }
```

To invoke this method, we need to append the query parameter into the URL as follows

http://localhost:8080/DMVLicense/webresources/scanplate/AAB0989

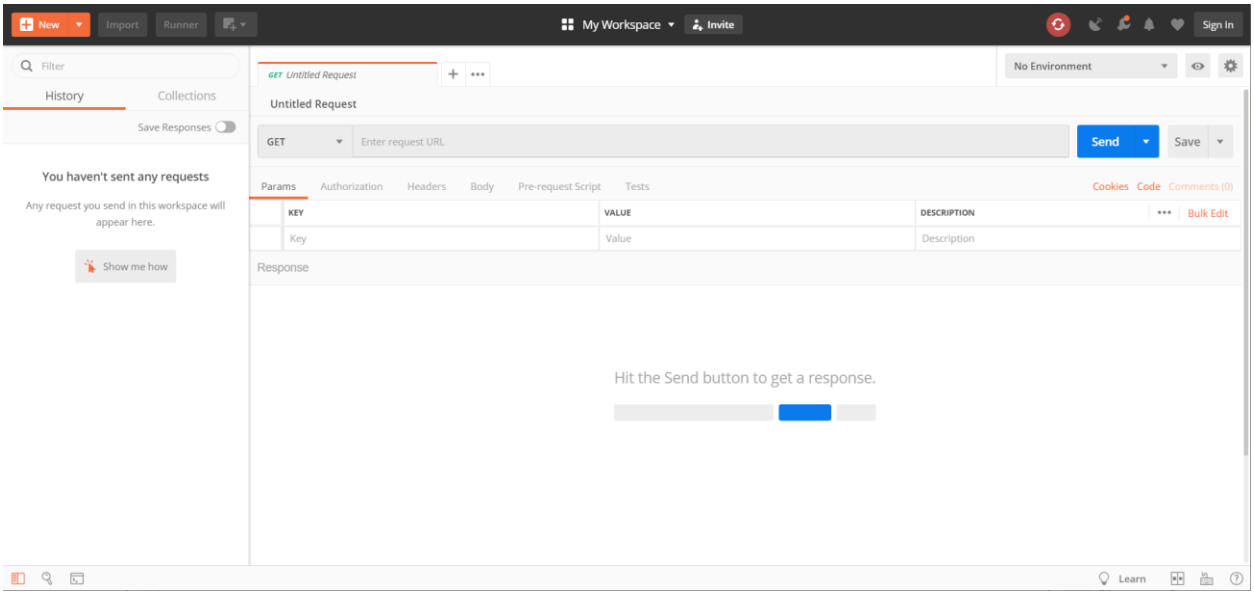


Required Driving License Category : Three Wheeler

Calling the webservices using POSTMAN

Download postman from https://www.postman.com/

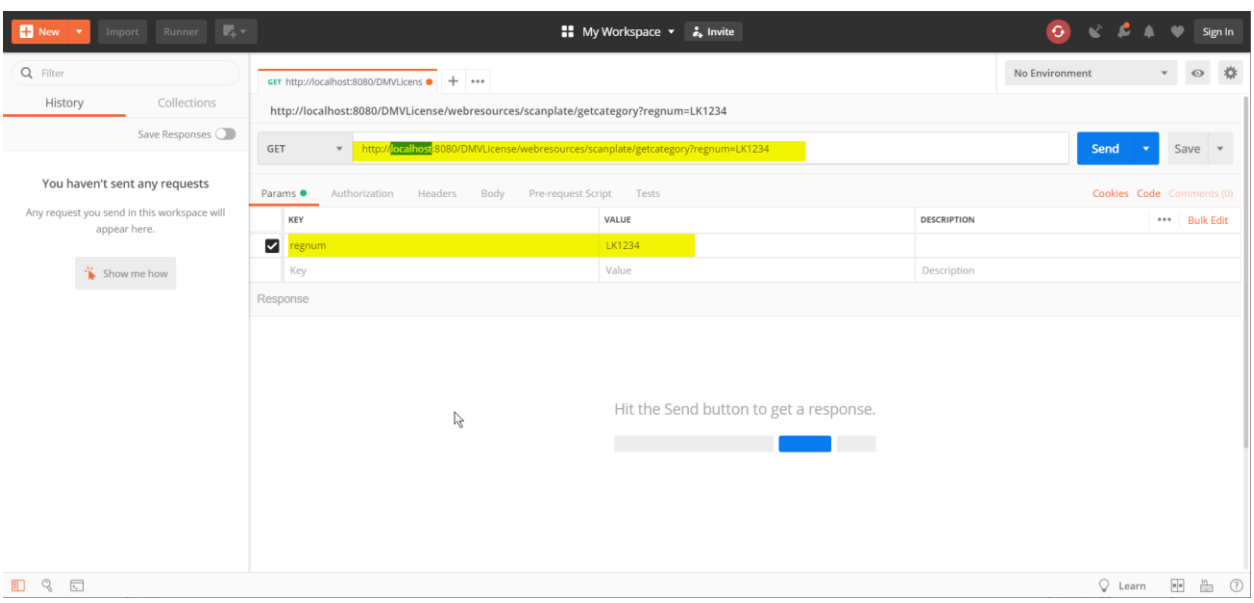
Once installed and when you open the application on the start page you can create a new Get Request



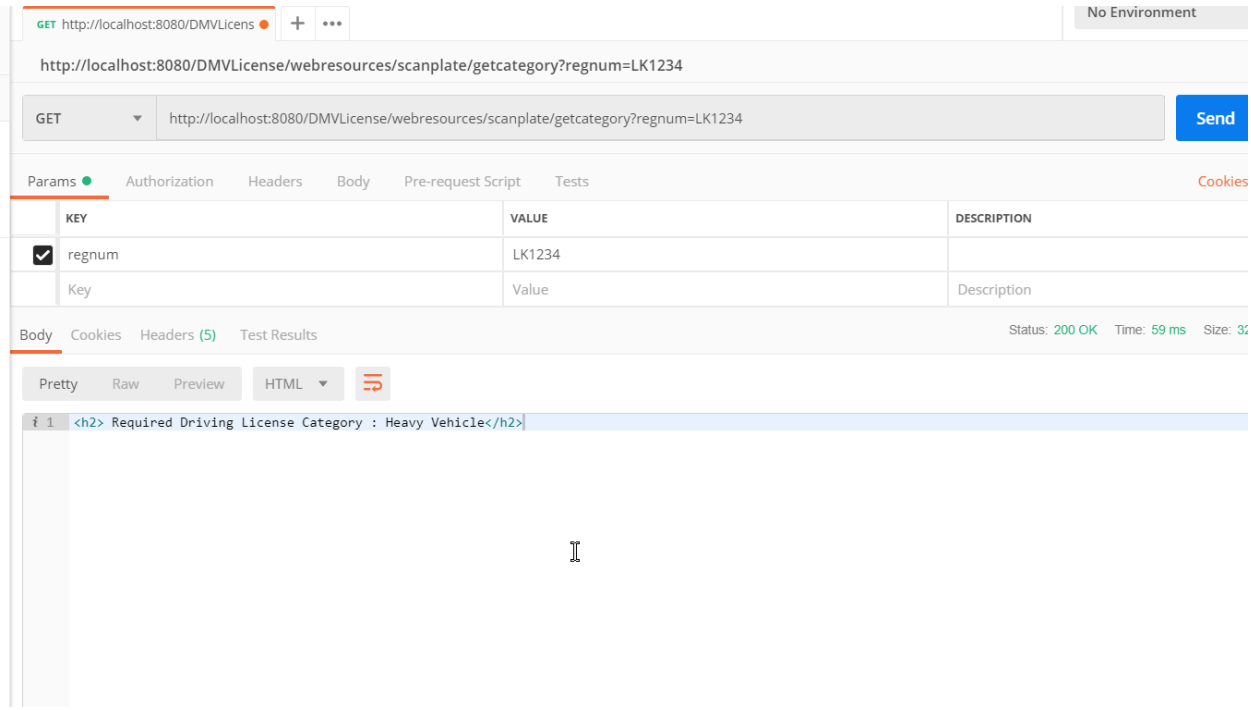
Simply copy the URL with the query parameter.

http://localhost:8080/DMVLicense/webresources/scanplate/getcategory?regnum=LK1234

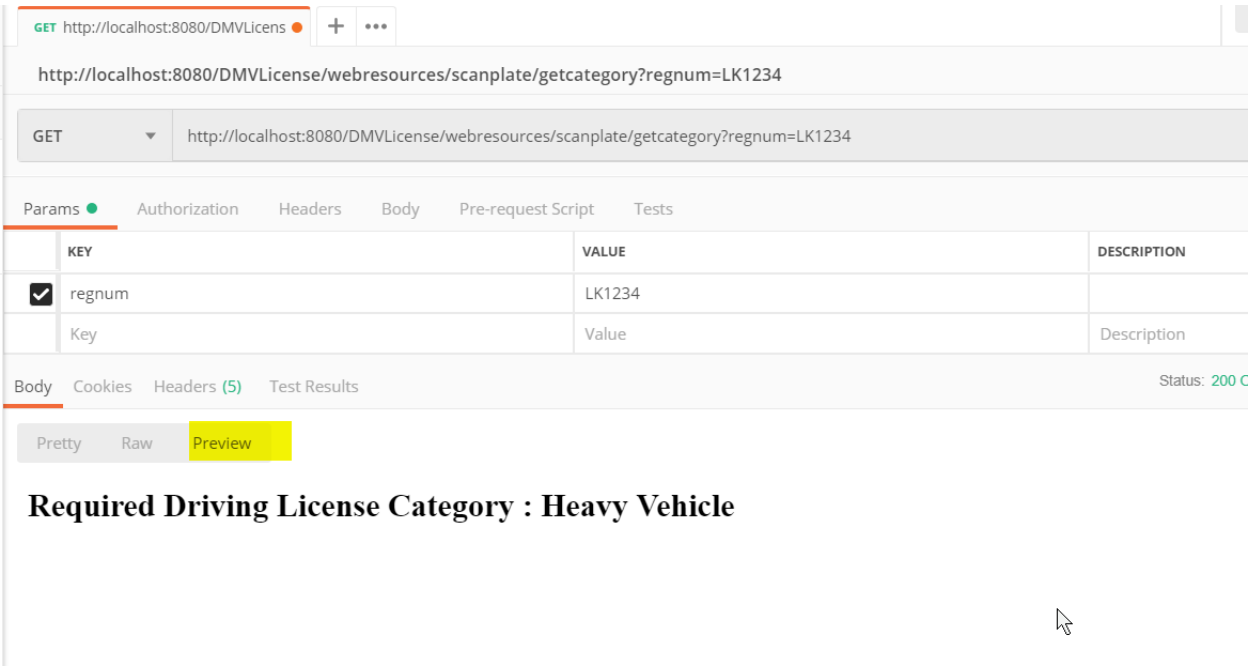
postman automatically detects the query param.



Hit the send button and you should be able to get a response.



The text is returned with the HTML tags so if you want to see the HTML preview you can click on the 'Preview' button.



Similarly, postman will read XML/JSON etc and that can be set by the MediaType on the GetMethod in the restful web service.

You can also try the request with path variable on Postman too …

http://localhost:8080/DMVLicense/webresources/scanplate/AAB0989