Steps for creating new visualization

1 - Create new form:

In folder **app\models\forms** create new java class named {new visualization}Form. The created class should extend class VisualizationForm. If you need fields that aren't in VisualizationForm, add them and their getters and setters to the new form. Implement the function *public String getParametersList()* in the new form as you wish.

2 - Add instructions:

In **views/visualizations/calculateDataInstructions.scala.html** add explanations for the new parameters in your form.

3 - Create controller to calculation:

In class **app\controllers\HomeController.java** add the following function:

```
public Result {new visualization}() {
  Form<{new visualization}> form = formFactory.form({new visualization}.class);
  return ok(views.html.visualizations.{new visualization}.{new visualization}.render(form));
}
```

4 - Create controller for result:

In class **app\controllers\HomeController.java** add the following function:

```
public Result calculate{new visualization}() throws java.io.IOException{
      Form<PassengersCountForm> form =   formFactory.form({new
visualization}Form.class).bindFromRequest();
      if(form.hasErrors()){
         flash("danger","Please Correct the Form Below");
         return badRequest(views.html.visualizations.{new visualization}.{new visualization}.render(form));
      }
      {new visualization}Form request = form.get();
   request.setResult(models.queries.queries.getResults(request));
      flash("success","Form Sent Successfully");
      return ok(views.html.visualizations.{new visualization}.{new visualization}Result.render(request));
  }
```

5 - Add routing:

In file **conf\routes** add the following lines:

```
GET     /{new visualization}                controllers.HomeController.{new visualization}
GET     /{new visualization}/Result         controllers.VisualizationController.calculate{new visualization}
```

6 - Add visualization to the navigation bar:

In **views/navigationbar.scala.html** add to <div id="visualizations"...> the following line:

{new visualization}

7 - Create the visualization folder:

Add to **app.views.visualizations** a new folder {new visualization}.

8 - Add calculation page:

In **app.views.visualizations.{new visualization}**, create new file {new visualization}.scala.html that will contain the following:

```
@(form : Form[{new visualization}Form])

@visualizations.layouts.calculateDataLayout("{new visualization}"){
  {description}
}{
  @helper.form(action = routes.VisualizationController.calculate{new visualization}()){
@visualizations.parameters.visualizationFormParameters(form.asInstanceOf[play.data.Form[models.VisualizationForm]])
      {fill other fields here}
  <input type="submit" value="Calculate" id="submit"  onclick="showDiv()">
  }
}
```

9 - Add result page:

In **app.views.visualizations.{new visualization}**, create new file {new visualization}Result.scala.html that will contain the following:

```
@(result: {new visualization}Form)

@visualizations.layouts.resultLayout("{new visualization} result"){
  {styles needed}
}{
@result.getParametersList()
}{
  {visualization ('data': JSON.parse('@result.getResultString()'.replace(/&quot;/g,'"')))}
}{
  {index}
}
```

10 - Add query handler for visualization

In **app.models.queries.{visualization folder}** create new class:
{new visualization}QueryHandler which extends QueryHandler class.
Add the following function to the class: *public JsonNode getResult()*.

This function should perform the query and return all the data in json file. You can use queries.GeojsonTempletes package to create the objects for the json file and use the following return statement:
return queries.mapper.valueToTree({QueryFeatureCollection instance});

11 - Add function in queries to get result from query handler:

Add to **models.queries** the following function:

```
public static JsonNode getResults({new visualization}Form form){
   {new visualization}QueryHandler handler = new {new visualization}QueryHandler(form);
   return handler.getResult();
}
```