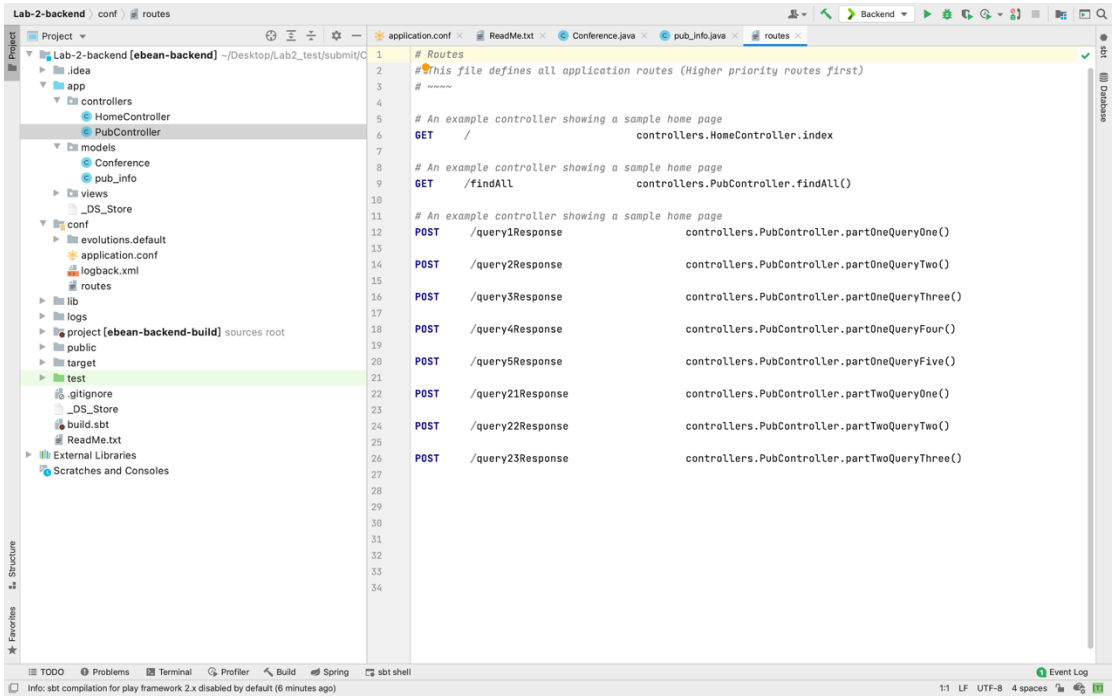
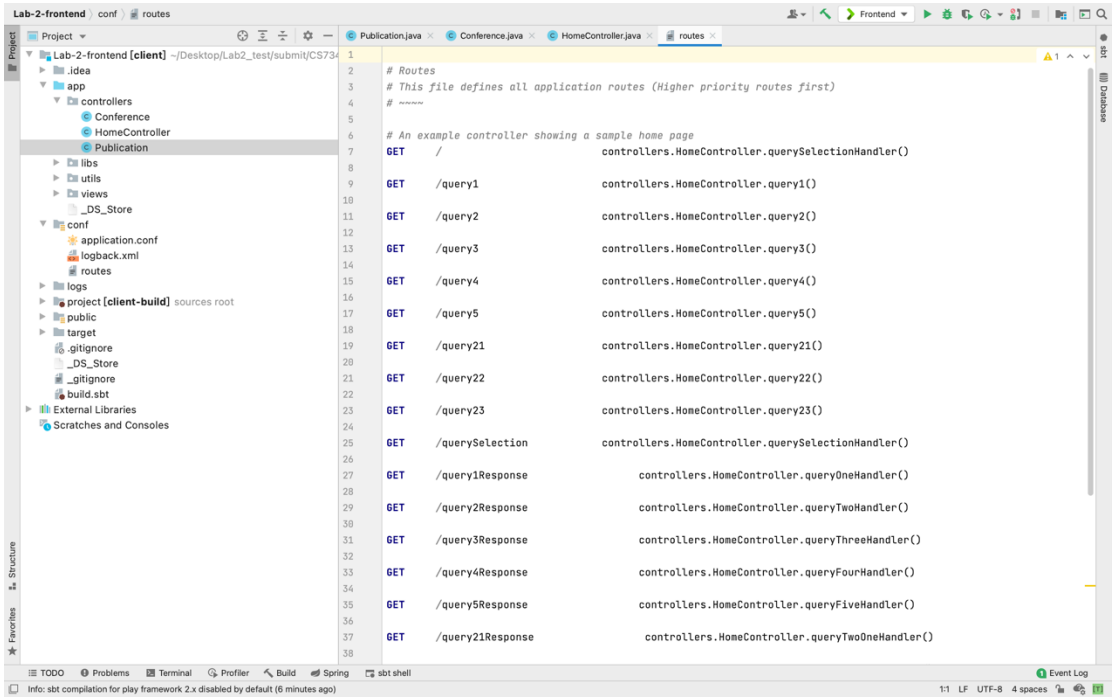
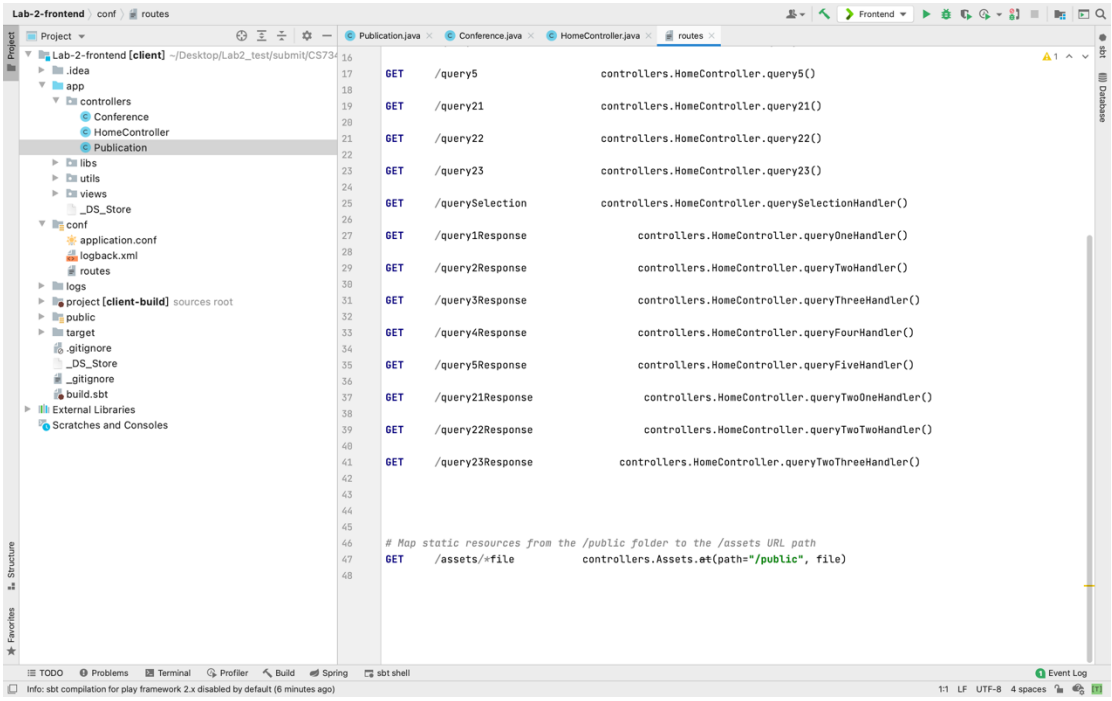
CS 7340 Team 1 Lab 2
API documents

Screenshot of routes file in backend:



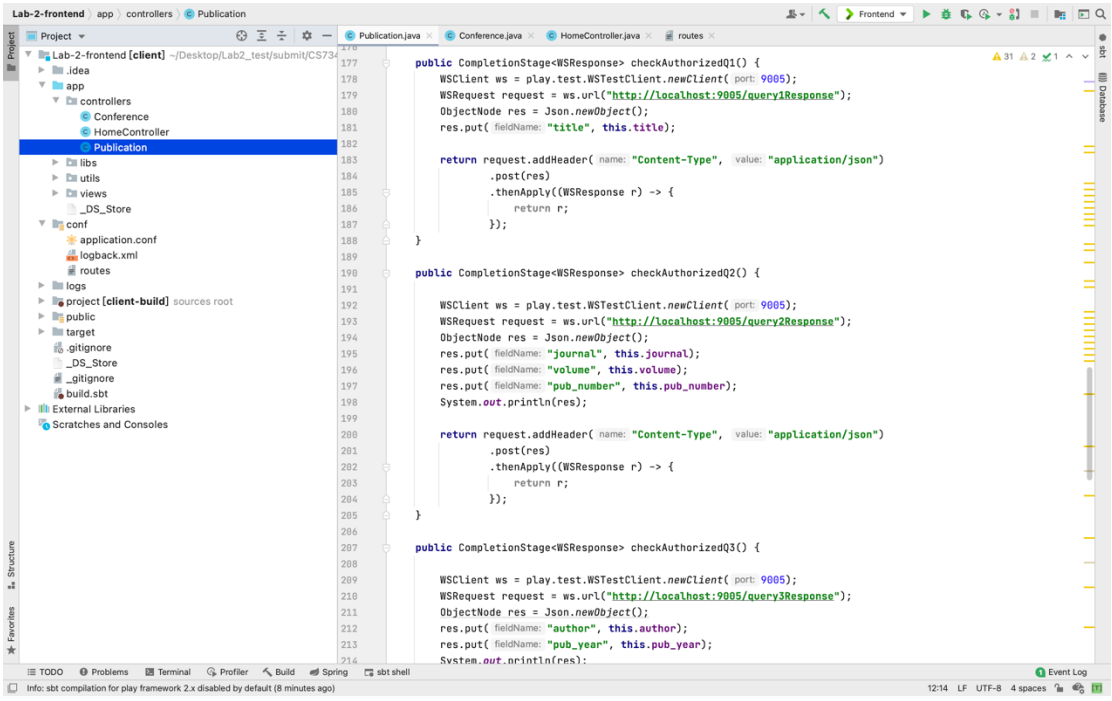
Screenshot of routes file in frontend:

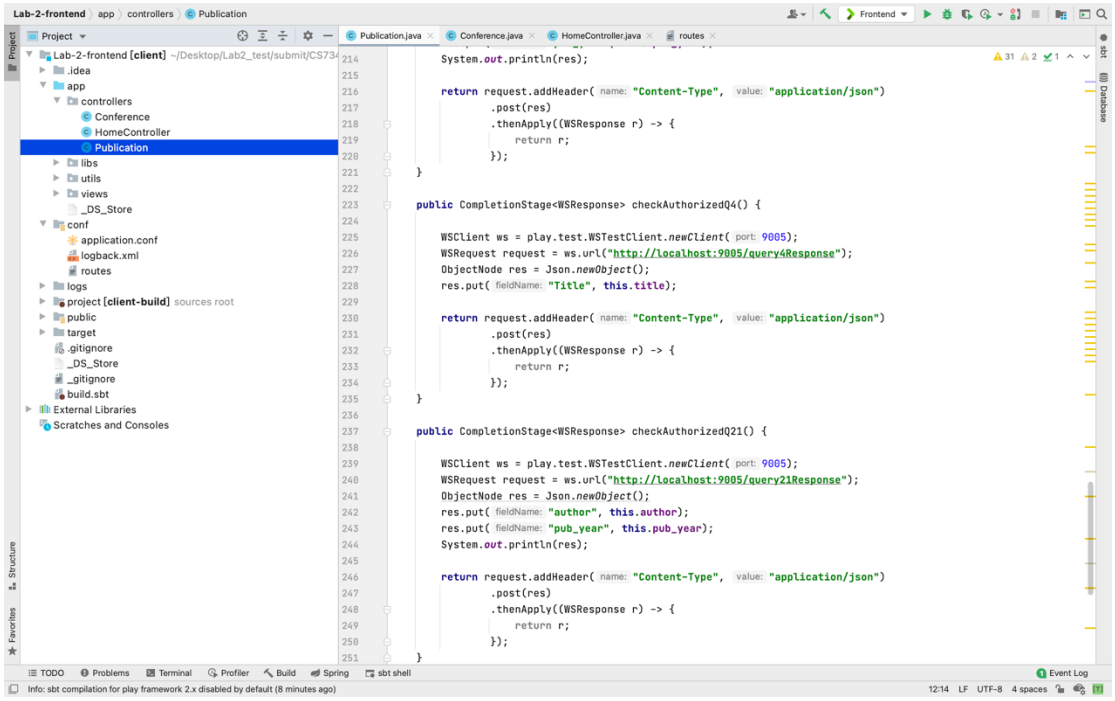


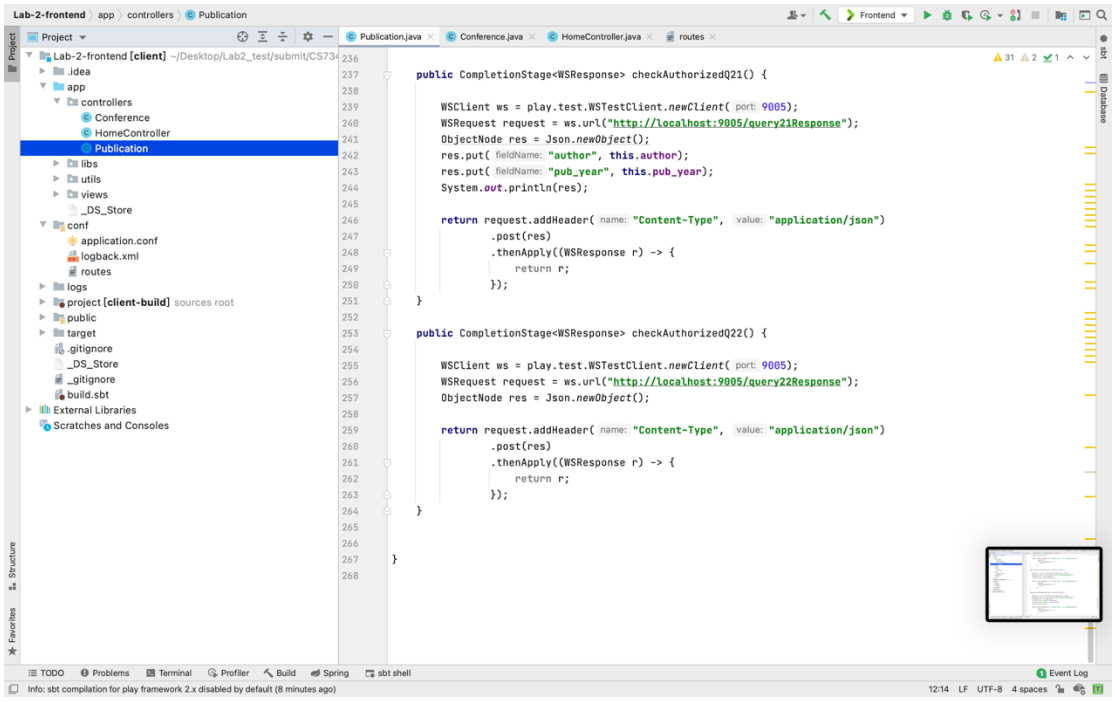


Screenshot of some functions in frontend relate to each query:

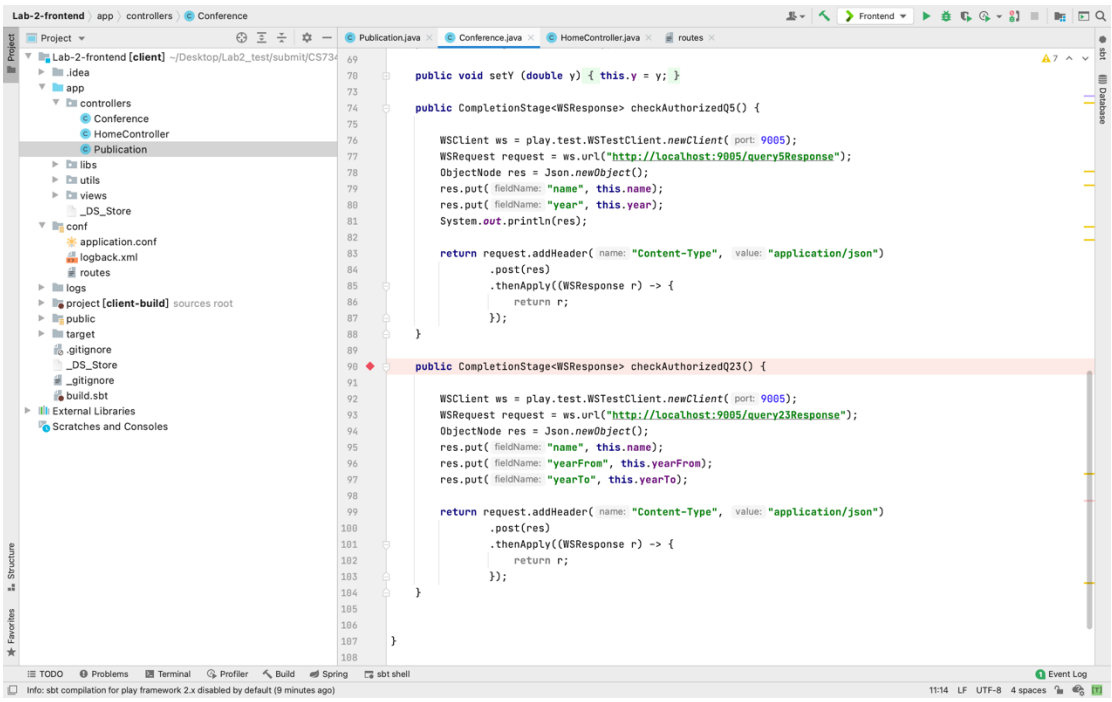
Query 1.1 to 1.4 and 2.1 to 2.2:




Lab-2-frontend app controllers Publication
Frontend
Project
Lab-2-frontend [client] ~/Desktop/Lab2_test/submit/CS734
.idea
app
controllers
Conference
HomeController
Publication
libs
utils
views
_DS_Store
conf
application.conf
logback.xml
routes
logs
project [client-build] sources root
public
target
.gitignore
_DS_Store
_gitignore
build.sbt
External Libraries
Scratches and Consoles
Publication.java
Conference.java
HomeController.java
routes
System.out.println(res);
return request.addHeader(name: "Content-Type", value: "application/json")
.post(res)
.thenApply((WSResponse r) -> {
return r;
});
}
public CompletionStage<WSResponse> checkAuthorizedQ4() {
WSClient ws = play.test.WSTestClient.newClient(port: 9005);
WSRequest request = ws.url("http://localhost:9005/query4Response");
ObjectNode res = Json.newObject();
res.put(fieldName: "Title", this.title);
return request.addHeader(name: "Content-Type", value: "application/json")
.post(res)
.thenApply((WSResponse r) -> {
return r;
});
}
public CompletionStage<WSResponse> checkAuthorizedQ21() {
WSClient ws = play.test.WSTestClient.newClient(port: 9005);
WSRequest request = ws.url("http://localhost:9005/query21Response");
ObjectNode res = Json.newObject();
res.put(fieldName: "author", this.author);
res.put(fieldName: "pub_year", this.pub_year);
System.out.println(res);
return request.addHeader(name: "Content-Type", value: "application/json")
.post(res)
.thenApply((WSResponse r) -> {
return r;
});
}
TODO Problems Terminal Profiler Build Spring sbt shell
Event Log
Info: sbt compilation for play framework 2.x disabled by default (8 minutes ago)
12:14 LF UTF-8 4 spaces


Lab-2-frontend app controllers Publication
Frontend
Project
Lab-2-frontend [client] ~/Desktop/Lab2_test/submit/CS734
.idea
app
controllers
Conference
HomeController
Publication
libs
utils
views
_DS_Store
conf
application.conf
logback.xml
routes
logs
project [client-build] sources root
public
target
.gitignore
_DS_Store
_gitignore
build.sbt
External Libraries
Scratches and Consoles
Publication.java
Conference.java
HomeController.java
routes
public CompletionStage<WSResponse> checkAuthorizedQ21() {
WSClient ws = play.test.WSTestClient.newClient(port: 9005);
WSRequest request = ws.url("http://localhost:9005/query21Response");
ObjectNode res = Json.newObject();
res.put(fieldName: "author", this.author);
res.put(fieldName: "pub_year", this.pub_year);
System.out.println(res);
return request.addHeader(name: "Content-Type", value: "application/json")
.post(res)
.thenApply((WSResponse r) -> {
return r;
});
}
public CompletionStage<WSResponse> checkAuthorizedQ22() {
WSClient ws = play.test.WSTestClient.newClient(port: 9005);
WSRequest request = ws.url("http://localhost:9005/query22Response");
ObjectNode res = Json.newObject();
return request.addHeader(name: "Content-Type", value: "application/json")
.post(res)
.thenApply((WSResponse r) -> {
return r;
});
}
}
TODO Problems Terminal Profiler Build Spring sbt shell
Event Log
Info: sbt compilation for play framework 2.x disabled by default (8 minutes ago)
12:14 LF UTF-8 4 spaces

Query 1.5 and 2.3:



Query 1.1 – 1.5:

frontend => publication.java:

```
177     public CompletionStage<WSResponse> checkAuthorizedQ1() {
178         WSClient ws = play.test.WSTestClient.newClient( port: 9005);
179         WSRequest request = ws.url("http://localhost:9005/query1Response");
180         ObjectNode res = Json.newObject();
181         res.put( fieldName: "title", this.title);
182 
183         return request.addHeader( name: "Content-Type",  value: "application/json")
184                 .post(res)
185                 .thenApply((WSResponse r) -> {
186                     return r;
187                 });
188     }
```

frontend => HomeController.java:

```
49      public CompletionStage<Result> queryOneHandler() {
50          Form<Publication> publicationForm = formFactory.form(Publication.class).bindFromRequest();
51          if (publicationForm.hasErrors()){
52              String authorizeMessage = "Invalid publication Title";
53              List<List<String>> wrongMessage = new ArrayList<>();
54              wrongMessage.get(0).add(authorizeMessage);
55              return (CompletionStage<Result>) badRequest(views.html.query1.render(wrongMessage));
56          }
57 
58          return publicationForm.get().checkAuthorizedQ1().thenApplyAsync((WSResponse r) -> {
59              System.out.println("Body: "+ r.getBody());
60              if (r.getStatus() == 200 && r.asJson() != null && !r.getBody().equals("false")) {
61                  JsonNode res = r.asJson();
```

Backend => PubController.java

```java
public Result partOneQueryOne() {
    System.out.println("In PartOneQueryOne");
    JsonNode req = request().body().asJson();
    String title = req.get("title").asText();
    System.out.println("receive title: "+title);
    try {
        List<ObjectNode> query1_1=getMetadataUsingPaperName(title);
        if (title.equals("")){
            return ok( content: "false");
        } else return ok(query1_1.toString());

    }
    catch (Exception e) {
        return ok( content: "false");
    }
}

public Result partOneQueryTwo() {
```

Backend => pub_info.java

```java
public static List<SqlRow> findByTitle_1_1(String title) {

    try{

        List<SqlRow> query1_1 = Ebean.createSqlQuery("select * from pub_info where title like'"+replacePunctuation
                "limit 1;")
                .findList();
        return query1_1;
    }catch(Exception e){
        e.printStackTrace();
        return null;
    }
}
```

For frontend, we receive a form from front end webpage, then save the frontend information into a JSON object “res”. Then send a post request from front end to back end by using an API url: http://localhost:9005/query(1-5)Response which is the base route url and our Json Object is our passing parameter.

For backend, we receive the request from frontend, we parse the API url. We get the JsonNode by using function request().body().asJson();

Then we call another function findByTitle_(1_1 – 1_4) in model.pub_info or findByConference in model.Conference to search result in our mysql Database.

We save the answer as String and use post request to send our data from backend to frontend.

When the front end receives the result from backend, then we translate the result from String to Json Object, then parse the Json Object and save the result into a List<List<String>> res and pass the res into front webpage.

```
return publicationForm.get().checkAuthorizedQ23().thenApplyAsync((WSResponse r) -> {
    if (r.getStatus() == 200 && r.asJson() != null && !r.getBody().equals("null")) {
        JsonNode res = r.asJson();
        session("name", publicationForm.get().getName());
        session("yearFrom", publicationForm.get().getYear());
        session("yearTo", publicationForm.get().getYear());

        System.out.println("The metadata of conference you are looking for:");
        List<Conference> confList = new ArrayList<>();
        for (int i = 0; i < res.size(); i++) {
            Conference curConf = new Conference();
            JsonNode row = res.get(i);
            curConf.setName(row.get("name").asText());
            curConf.setYear(row.get("year").asText());
            curConf.setLocation(row.get("location").asText());
            curConf.setX(row.get("x").asDouble());
            curConf.setY(row.get("y").asDouble());
            confList.add(curConf);
        }
        return ok(views.html.response.render(confList));
```

For query 1.5, the send API from front to back end is the same as before 4 queries. But when we receive the result from backend, we translate it as a Json Object, then we save the information into a List<Conference> to save all the information about the conference location and other information (coordinate). Then show the result in the query5.scala.html

Query 2.1-2.3:
Frontend:

```
GET     /query21                    controllers.HomeController.query21()

GET     /query22                    controllers.HomeController.query22()

GET     /query23                    controllers.HomeController.query23()

GET     /query21Response            controllers.HomeController.queryTwoOneHandler()

GET     /query22Response            controllers.HomeController.queryTwoTwoHandler()

GET     /query23Response            controllers.HomeController.queryTwoThreeHandler()
```

```
public Result query21() {
    return ok(views.html.query21.render(new ArrayList<>()));
}

public Result query22() {
    return ok(views.html.query22.render(new ArrayList<>()));
}

public Result query23() {
    return ok(views.html.query23.render(new ArrayList<>()));
}
```

Query 2.1:

From “query21” “queryTwoOneHandler” and “CheckAuthorizedQ21”, we are able to send the Json Values of “author” and “pub_year” from frontend to backend with OK object. Also, using them with “GET” in routes, the frontend can receive the “WSResponse” type of objects which store all the metadata for specific publication. We can change the “WSResponse” objects into Json objects.

Moreover, the API url from backend is “http://localhost:9005/query21Response”.

```
public CompletionStage<Result> queryTwoOneHandler() {
    Form<Publication> publicationForm = formFactory.form(Publication.class).bindFromRequest();
    if (publicationForm.hasErrors()){
        String authorizeMessage = "Form has errors";
        List<List<String>> wrongMessage = new ArrayList<>();
        List<String> curMess = new ArrayList<>();
        curMess.add(authorizeMessage);
        wrongMessage.add(curMess);
        return (CompletionStage<Result>) badRequest(views.html.query21.render(wrongMessage));
    }

    return publicationForm.get().checkAuthorizedQ21().thenApplyAsync((WSResponse r) -> {
        if (r.getStatus() == 200 && r.asJson() != null && !r.getBody().equals("null")) {
            JsonNode res = r.asJson();
            List<List<String>> res2_1 = new ArrayList<>();
            String query2_1 = new String();
            // add author and pub_year to session
            session("author", publicationForm.get().getAuthor());
            session("pub_year", publicationForm.get().getPub_year());
            System.out.println(res);
            List<String> curList = new ArrayList<>();
            for (int i = 0; i < res.size(); i++) {
                JsonNode row = res.get(i);
                List<String> curNode = new ArrayList<>();
                //System.out.println(row.findValue("pid").asText());
                curNode.add(""+row.findValue("title"));
                curNode.add(""+row.findValue("mdate"));
                curNode.add(""+row.findValue("author list"));
```

```
public CompletionStage<WSResponse> checkAuthorizedQ21() {

    WSClient ws = play.test.WSTestClient.newClient( port: 9005);
    //add Title
    WSRequest request = ws.url("http://localhost:9005/query21Response");
    ObjectNode res = Json.newObject();
    res.put( fieldName: "author", this.author);
    res.put( fieldName: "pub_year", this.pub_year);
    System.out.println(res);

    return request.addHeader( name: "Content-Type",  value: "application/json")
            .post(res)
            .thenApply((WSResponse r) -> {
                return r;
            });
}
```

Query 2.2:

From "query22" "queryTwoTwoHandler" and "CheckAuthorizedQ22", we are able to send the execution command from frontend to backend with OK object. Also, using them with "GET" in routes, the frontend can receive the "WSResponse" type of objects which store all the co-authors for specific authors. We can change the "WSResponse" objects into Json objects.

Moreover, the API url from backend is http://localhost:9005/query22Response.

```
public CompletionStage<Result> queryTwoTwoHandler() {
    Form<Publication> publicationForm = formFactory.form(Publication.class).bindFromRequest();
    if (publicationForm.hasErrors()){
        List<List<String>> wrongList = new ArrayList<>();
        wrongList.get(0).add("Author form has errors");
        return (CompletionStage<Result>) badRequest(views.html.query22.render(wrongList));  //
    }

    return publicationForm.get().checkAuthorizedQ22().thenApplyAsync((WSResponse r) -> {
        if (r.getStatus() == 200 && r.asJson() != null && !r.getBody().equals("null")) {
            JsonNode res = r.asJson();

  public CompletionStage<WSResponse> checkAuthorizedQ22() {

      WSClient ws = play.test.WSTestClient.newClient( port: 9005);
      //add Title
      WSRequest request = ws.url("http://localhost:9005/query22Response");
      ObjectNode res = Json.newObject();
//        System.out.println(res);

      return request.addHeader( name: "Content-Type",  value: "application/json")
              .post(res)
              .thenApply((WSResponse r) -> {
                  return r;
              });
  }


}
```

Backend:

```
POST    /query21Response                controllers.PubController.partTwoQueryOne()
POST    /query22Response                controllers.PubController.partTwoQueryTwo()
POST    /query23Response                controllers.PubController.partTwoQueryThree()
```

Query 2.1

With routes and request in "partTwoQueryOne" to get the JsonNode of "author" and "pub_year" from frontend.

```
public Result partTwoQueryOne(){
    System.out.println("In PartTwoQueryOne");
    JsonNode req = request().body().asJson();
    String author = req.get("author").asText();
    System.out.println(req);
    int pub_year = req.get("pub_year").asInt();
    System.out.println("receive reasearcher name = "+author);
    System.out.println("pub_year = "+pub_year);
```

Then we use these two functions below as API to handle the inputs. Firstly, it gets author and pub_year and outputs the String List of titles. Then, it uses titles to output list of ObjectNode which contains the matadata of specific publication.

And the URL is "http://localhost:9005/query21Response".

```
301     public List<ObjectNode> getTitlesUsingAuthorAndYear (String author, int pub_year) {
302         List<ObjectNode> reslist = new ArrayList<>();
303         try {
304             List<SqlRow> pub = pub_info.findByTitle_1_3(author,pub_year);
305             System.out.println(pub.get(0)!= null);
306             if (pub.get(0)!= null) {
307                 int i=0;
```

```
217     public List<ObjectNode> getMetadataUsingPaperName(String title) {
218         List<ObjectNode> reslist= new ArrayList<>();
219         try {
220                 List<SqlRow> pub = pub_info.findByTitle_1_1(title);
221
222                 if (pub.size()!= 0) {
223                     for (SqlRow pubs : pub) {
224                         ObjectNode res = Json.newObject();
```

Finally, we use ok to "POST" the ObjectNode list to frontend.

```
        System.out.println("result: "+query2_1_metadata);
        return ok(query2_1_metadata.toString());

    }
    catch (Exception e) {
        return ok( content: "false");
    }
```

Query 2:

With routes and “partTwoQueryTwo” to get the exectution command from frontend.

Then we use the function below as API to get the productive author by using ObjectNode list.

```
 */
public List<ObjectNode> getAuthorsWhosePaperMoreThanTen() {
    List<ObjectNode> reslist = new ArrayList<>();
    try {
        List<SqlRow> pub = pub_info.findByAuthor_1_4();
        System.out.println(pub.get(0)!= null);

        if (pub.get(0)!= null) {
            int i=0;
            for (SqlRow pubs : pub) {
                i++;
                ObjectNode res = Json.newObject();
                res.put( fieldName: "author",(String) pubs.get("author"));

                System.out.println(res);

                reslist.add(res);
            }
            System.out.println(reslist);
```

Next, we get the List of data and apply them into function below. Then we can get the coauthor list with ObjectNode type.

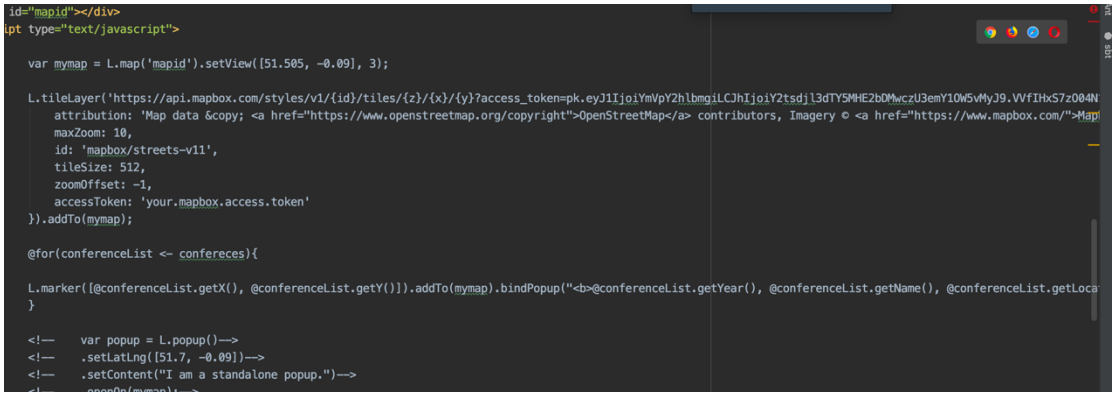
```
 */
public List<ObjectNode> getCoauthorForProductiveAuthor(String name) {
    List<ObjectNode> reslist = new ArrayList<>();
    try {
        List<SqlRow> pub = pub_info.findCoAuthor_2_2(name);
        System.out.println(pub.get(0)!= null);

        if (pub.get(0)!= null) {
            int i=0;
            for (SqlRow pubs : pub) {
                ObjectNode res = Json.newObject();
                res.put( fieldName: "author",(String) pubs.get("author"));
```

Finally, we use OK and routes to send the data back to the frontend. And the URL is http://localhost:9005/query22Response.

```java
public Result partTwoQueryTwo(){
    System.out.println("In PartTwoQueryTwo");

    try {
        List<SqlRow> pub = pub_info.findByAuthor_1_4();
        System.out.println(pub.get(0)!= null);
        List<ObjectNode> query2_2_Coauthor=new ArrayList<>();
        List<ObjectNode> query2_2_Author = getAuthorsWhosePaperMoreThanTen();
        for(int i=0;i<query2_2_Author.size();i++){
            List<ObjectNode> temp=new ArrayList<>();
            String author=query2_2_Author.get(i).findValue( fieldName: "author").asText();
            temp=getCoauthorForProductiveAuthor(author);
            List<String> coauthor=new ArrayList<>();
            for(int k=0;k<temp.size();k++) {
                coauthor.add(temp.get(k).findValue( fieldName: "author").asText());
            }
            List<ObjectNode> temp=new ArrayList<>();
            String author=query2_2_Author.get(i).findValue( fieldName: "author").asTex
            temp=getCoauthorForProductiveAuthor(author);
            List<String> coauthor=new ArrayList<>();
            for(int k=0;k<temp.size();k++) {
                coauthor.add(temp.get(k).findValue( fieldName: "author").asText());
            }
            ObjectNode authorWithCoauthor=Json.newObject();
            authorWithCoauthor.put( fieldName: "author",author);
            authorWithCoauthor.put( fieldName: "coauthor",coauthor.toString());
            query2_2_Coauthor.add(authorWithCoauthor);
        }

        System.out.println("result: "+query2_2_Coauthor);

        return ok(query2_2_Coauthor.toString());
    }
    catch (Exception e) {
        return ok( content: "false");
    }
}
```

Query 2.3:

```html
<!doctype html>
<html lang="en">
  <head>
    <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
    integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZMZ19scR4PsZChSR7A=="
    crossorigin=""/>
    <style>
            #mapid {
                height: 600px;
            }
    </style>
        <!-- Make sure you put this AFTER Leaflet's CSS -->
    <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
    integrity="sha512-XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaVUearIOBhiXZ5V3ynxwA=="
    crossorigin="">

    </script>
```



The screenshot is an external Map API which is leaflet API. So, when we receive the result from query 5 and get the conference metadata. Then we call the external Map APIs, from the screenshot, we can see the API URL should be https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token=pk.eyJ1IjoiYmVpY2hlbmgiLCJhIjoiY2tsdjl3dTY5MHE2bDMwczU3emY1OW5vMyJ9.VVfIHxS7zO04N1trHAnEmwwhere the parameter is the access_token which is created by my account on Internet.