SCS 2204 Functional Programming

7/10/2021

play

Group Members

Index No	Registration No	Name
19001802	2019/CS/180	V.D.L.W.Vitahanage
19000901	2019/CS/090	S.S.Malsha
19000741	2019/CS/074	D.G.Hansika Kumari
19000601	2019/CS/060	H.L.Harith Iduwara

Table of Contents

1.Frameworks available in scala

1.1 play framework

Play Framework is an open-source Scala framework that follows the architectural model of MVC (Model-View-Controller), first released in 2007. Play Framework combines productivity and functionality with Java and Scala, making it easy to build scalable web applications. It is currently being developed by Lightbend, Zengularity and its community of user developers. The basic functionality of the framework is based on the development of RESTful applications in JVM and related libraries. It is currently used by some relatively large websites, including LinkedIn, Walmart, the Samsung's loT Artik platform, and the education website Coursera. At the time of writing, Play 2.8 is the current version of Play that has surpassed Play 1 in development.

Advantages:

- Play is based on a lightweight, stateless and web-friendly architecture.
- Play is open source, which benefits adoptees with more prominent security and a reliably explored codebase.
- Closely related to JVM, it will be familiar and easy to use for Java developers.
- It is based entirely on functional programming concepts and promotes API-first RESTful design applications.
- Extensive support for various tools and IDE systems.
- It provides a wide range of framework support structures for asset management, model handling, database integration, etc.

Disadvantages:

- There are sure to be many good plugins in the community, but their stability and usefulness are not always guaranteed.
- Play 2 uses the SBT build system. Although it is very powerful, some have been concerned about its implications, substitute card imports and other eccentricities that make infrastructure development and integration extremely difficult.

1.2 Akka HTTP

Akka HTTP is an Akka based HTTP library for building Restful services. It has both client-side and client-side utilities. Akka HTTP is a exceedingly modular and extremely dominant Akka application for Scala. Akka HTTP is built on top of the Akka framework and version 1.0 was released in the summer of 2015. It was created with the concept in mind of development occurring within a “frame,” where preset choices and functionalities guide development. Akka HTTP is designed as a "not-a-framework" that provides many tools for development without forcing the developer to make any choices.

Advantages:

- Akka HTTP integrates with Akka functionality. Like Akka, it maintenances many systems that can accomplish parallel commands and progressive computation processing.
- Akka HTTP has a large support base of developers and contributors, all lined up under Lightbend. In addition, it has excellent documentation and an easy-to-understand support center.

Disadvantages:

- Akka HTTP is slower than other implementations, and while it scales efficiently, it has already started on the back of the package, so to speak.
- Lightbend is a great organization, but being stuck in a single vendor can be troublesome for many organizations. The locking of the vendor can be expensive and difficult to break, so this should be considered before adopting the solution.

1.3 Lift

Lift is a Scala based web framework designed for highly interactive, engaging web applications. It is a highly secure and scalable framework. From the ground up, Lift has been designed specifically to address a multitude of safety issues, including inter-site script and code injection. Because of this, Lift has become known as a highly secure alternative to the highly secure Scala API. Lift defines itself as sheltered, developer-centric, scalable, designer-friendly, integrated, and collaborative.

Advantages:

- Lift is designed around safety from its basic level. Form value obfuscation and the use of prepared declarations, in addition to other techniques, Lift promotes code-scale security in the standard implementation.
- Lift is bit fast, this is often its main point of sale. Database queries are concise, calls are handled very quickly, and its support for REST access to content management is helpful.

Disadvantages:

- The Lift community is a bit small compared to other options. But it's existed for a really while, so there’s ample documentation. However, those documents are often outdated and if they are not outdated, it is not always the best.
- Lift by its nature started off quite stateful, and this is still in the current code base, so the scale Lift is somewhat tactical when it comes to REST, can cause some problems in what data is transferred, what is sequenced and shared, and how requests are routed.

1.4 BlueEyes

Blue Eyes is a playful, asynchronous web framework for the Scala programming language that allows you to quickly and easily create high-performance web services that embrace HTTP machinery and language. Blue Eyes bills as the "Web 3.0 Framework for Scala" and focuses entirely on functionality and compactness.

Advantages:

- BlueEyes is extremely scalable and designed specifically for high-performance applications that need modularity.
- BlueEyes benefits from being purely asynchronous - in such cases, it is a strong, almost purpose built solution.
- Built around the concept of modularity and compactness, Blueeyes is an excellent choice for any application that can meet specific needs and standards.

Disadvantages:

- BlueEyes is completely asynchronous. So if the app needs synchronization functionality, BlueEyes is not a good choice.
- By design, BlueEyes has no support for server-side generation of HTML, CSS, or JavaScript, nor does it support static files. Its overall purpose is to build RESTful services that are consumed by customers. This looks great, but if you want such functionality, you need to use an external plugin or solution.

2. Abbreviations

Views

- Views are user interfaces of the application.
- Play framework views contain web markup tags, template tags, and directives.
- Views are simple text files that are designed to make it easier to work with web markup languages such as HTML.

Controller

- Controllers define in domain model layer managed with Business logic. Clients cannot directly invoke code so the functionality of a domain object is display as resources represent by URLs. To manipulate these resources, Clients use uniform API provided by HTTP protocol. This mapping of resources to domain objects can be expressed at different levels. Controller providing a connection between the domain models object and transport layer events. Controllers making easy to access or modify Model objects because they are written in pure java. Like the HTTP interface, Controllers are procedural and focus on Request and Response.
- The Controller layer reduces the impedance inconsistency between HTTP and the Domain Model.

Routes

- The mapping between requests and entry points define by routes in the conf/routes file.
- Routes are consisting of an HTTP method and URI pattern.
- Both are associated with a call to an Action generator.
- Comments start with # and each row in the routes file defines a route associating an HTTP verb and a URL template to a controller call for action.
- The content of the routes file is compiled by the Sbt plugin into a Scala object called Router and contains the dispatching rationale.
- The router tests every route, one after the other, in their reporting order. If the verb and URL match the configuration, the appropriate action is called.
- If we set localhost: 9000/register in the url bar, the controller calls the method register (). Then it renders the register.scala.html page.
- We also have set routing for the css, js, and other assets.

Build.sbt file

- Sbt may be a task engine. Compared to several traditional build tasks sbt function is sort of different.
- The build is represented as a tree of task dependencies that require to be executed.
- Sbt breaks typical build runs into very small tasks.
- This makes Sbt very powerful, but also requires a change of thinking if you have come from other building tools that break your build up into very coarse.

Application.conf file

- Play framework configurations defined using system properties and environment variables.
- This can be very useful and easy to use when settings change between environments and can use this application. Conf for common settings.
- But it uses system properties and environment variables to change settings when you run the application in different environments.
- Arrangement assets supersede application.conf, and application.conf supersede default settings for various reference. Conf files.

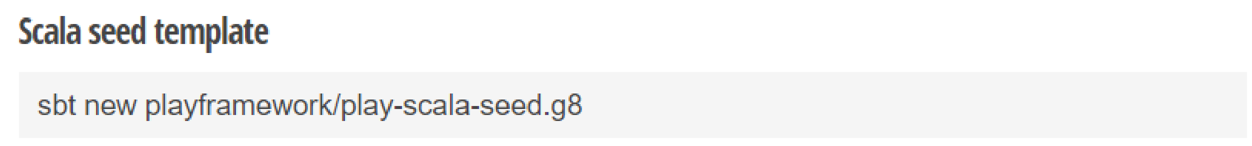
3. Creating a new project

3.1 Installing prerequisites

Prerequisites =>

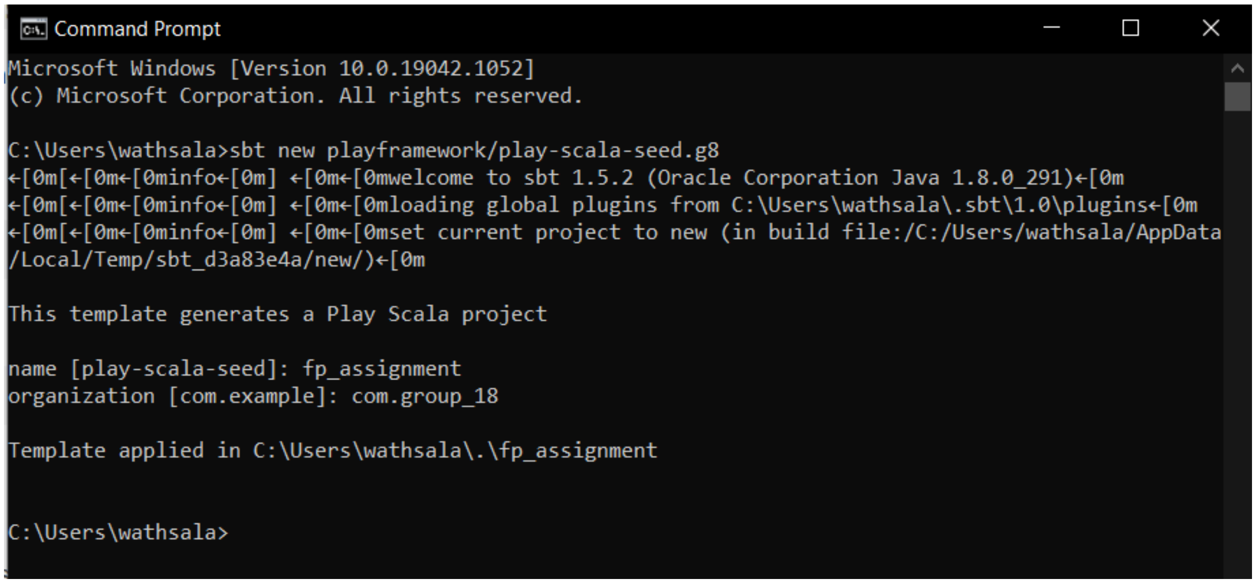
- Java 8 or high version
- SBT
- Play Framework
- Server to run (In our project we used wamp server as our server to run)
- Platform to run (In our project we use intellJ IDEA as our platform. Before starting the project we must install scala plugins in intellJ IDEA)

To create a new project we must run following commands in the command prompt.



Scala seed template

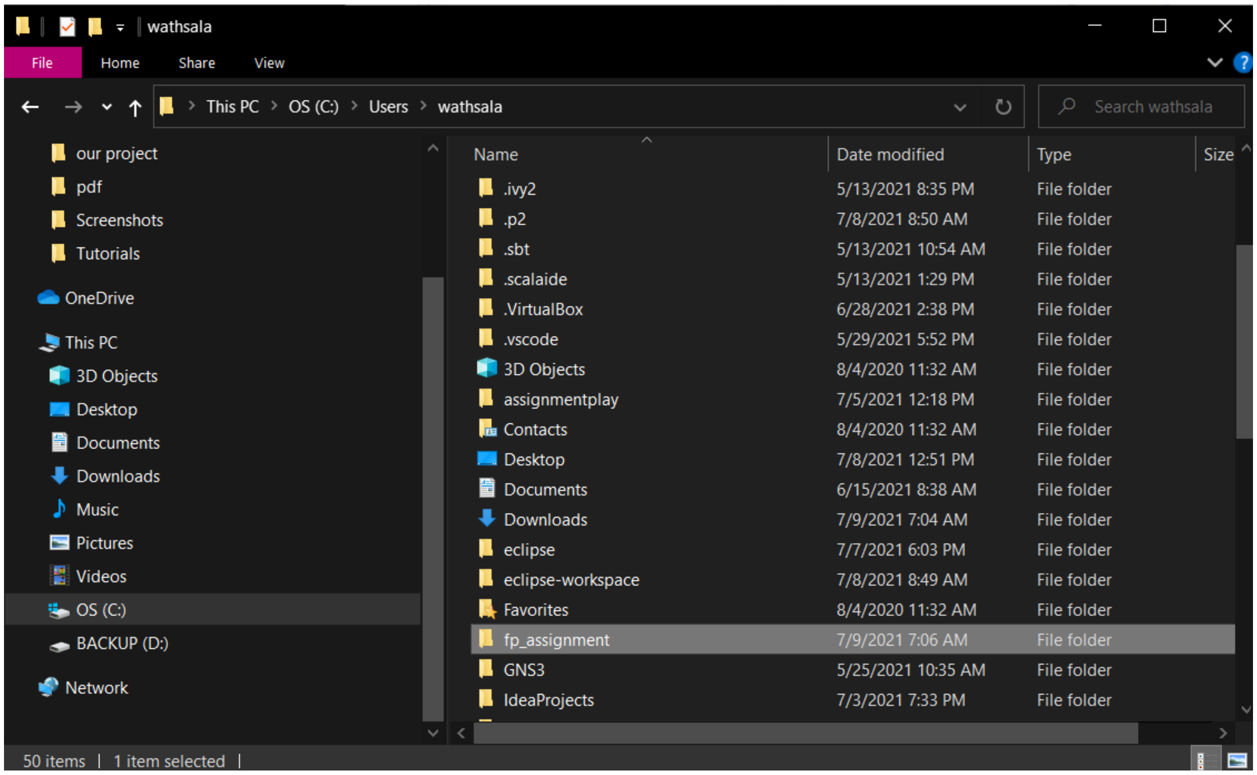
```
sbt new playframework/play-scala-seed.g8
```



```
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\wathsala>sbt new playframework/play-scala-seed.g8
←[0m[←[0m←[0minfo←[0m] ←[0m←[0mwelcome to sbt 1.5.2 (Oracle Corporation Java 1.8.0_291)←[0m
←[0m[←[0m←[0minfo←[0m] ←[0m←[0mloading global plugins from C:\Users\wathsala\.sbt\1.0\plugins←[0m
←[0m[←[0m←[0minfo←[0m] ←[0m←[0mset current project to new (in build file:/C:/Users/wathsala/AppData
/Local/Temp/sbt_d3a83e4a/new/)←[0m

This template generates a Play Scala project

name [play-scala-seed]: fp_assignment
organization [com.example]: com.group_18

Template applied in C:\Users\wathsala\.\fp_assignment


C:\Users\wathsala>
```

After we have to our c folder. Then user folder. Then username folder. (In our case it's wathsala).

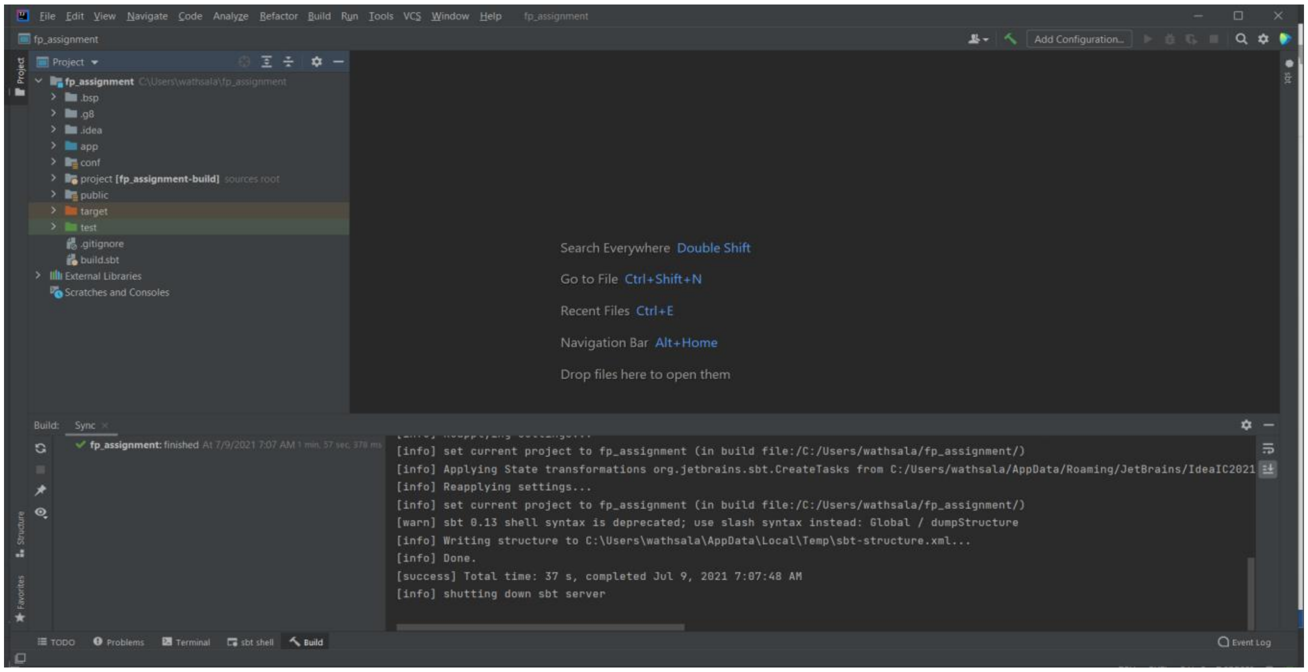
We just have to right click on our newly created project and open with intellJ IDEA.



Name	Date modified	Type
.ivy2	5/13/2021 8:35 PM	File folder
.p2	7/8/2021 8:50 AM	File folder
.sbt	5/13/2021 10:54 AM	File folder
.scalaide	5/13/2021 1:29 PM	File folder
.VirtualBox	6/28/2021 2:38 PM	File folder
.vscode	5/29/2021 5:52 PM	File folder
3D Objects	8/4/2020 11:32 AM	File folder
assignmentplay	7/5/2021 12:18 PM	File folder
Contacts	8/4/2020 11:32 AM	File folder
Desktop	7/8/2021 12:51 PM	File folder
Documents	6/15/2021 8:38 AM	File folder
Downloads	7/9/2021 7:04 AM	File folder
eclipse	7/7/2021 6:03 PM	File folder
eclipse-workspace	7/8/2021 8:49 AM	File folder
Favorites	8/4/2020 11:32 AM	File folder
fp_assignment	7/9/2021 7:06 AM	File folder
GNS3	5/25/2021 10:35 AM	File folder
IdeaProjects	7/3/2021 7:33 PM	File folder

50 items | 1 item selected |

When our project successfully open with our idea it gives us a green tick.



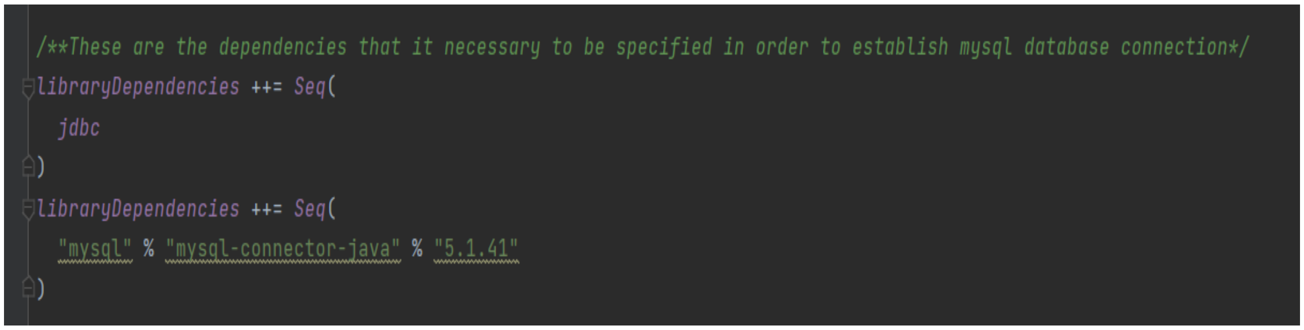
fp_assignment: finished At 7/9/2021 7:07 AM 1 min, 37 sec, 378 ms

```
[info] set current project to fp_assignment (in build file:/C:/Users/wathsala/fp_assignment/)
[info] Applying State transformations org.jetbrains.sbt.CreateTasks from C:/Users/wathsala/AppData/Roaming/JetBrains/IdeaIC2021
[info] Reapplying settings...
[info] set current project to fp_assignment (in build file:/C:/Users/wathsala/fp_assignment/)
[warn] sbt 0.13 shell syntax is deprecated; use slash syntax instead: Global / dumpStructure
[info] Writing structure to C:\Users\wathsala\AppData\Local\Temp\sbt-structure.xml...
[info] Done.
[success] Total time: 37 s, completed Jul 9, 2021 7:07:48 AM
[info] shutting down sbt server
```

Our initialization part is over.

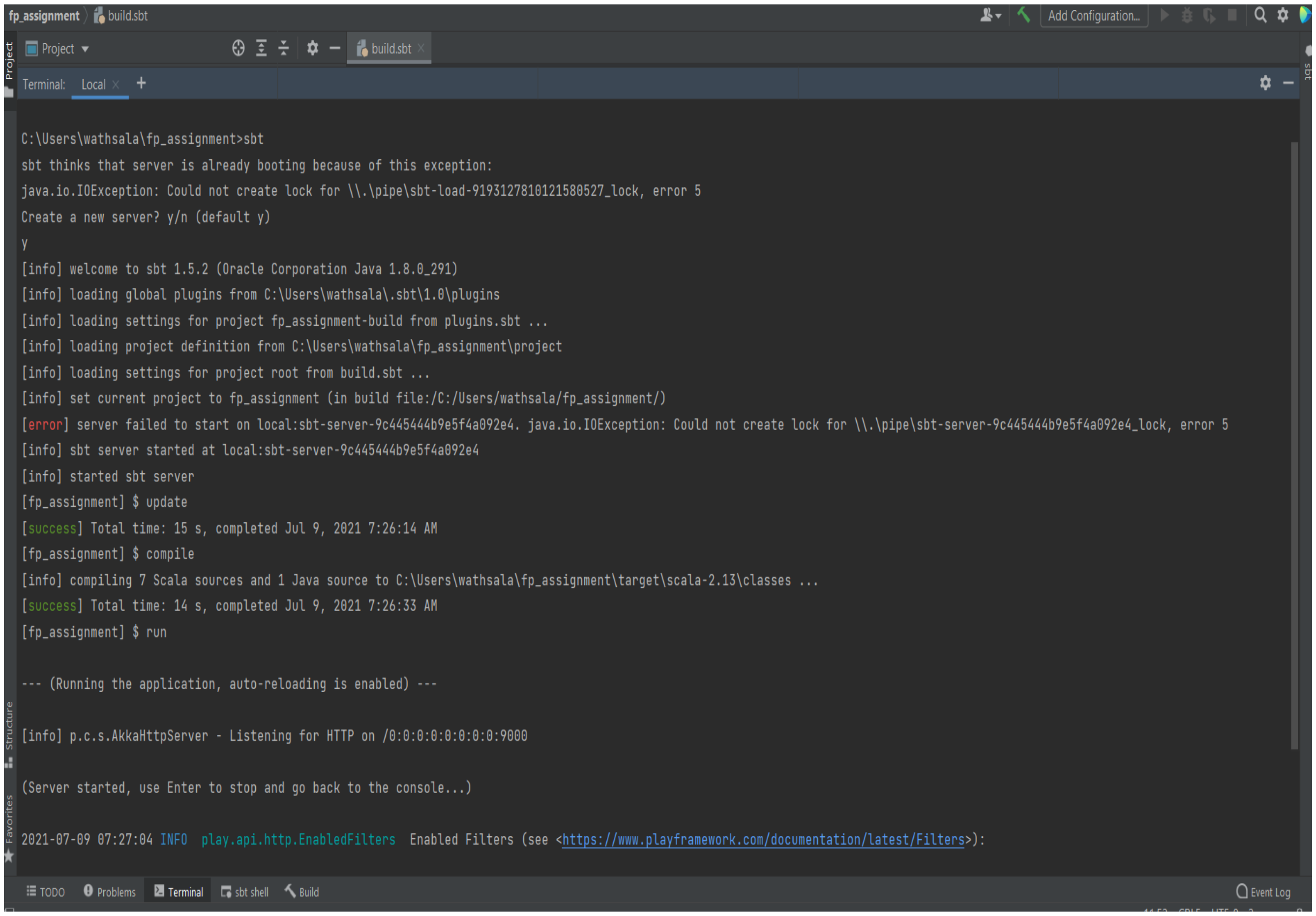
3.2 Specify MYSQL dependencies

In the play framework all the dependencies are hold by bin folder (build.sbt). So we have establish dependencies related to MYSQL database connection in out build.sbt folder.



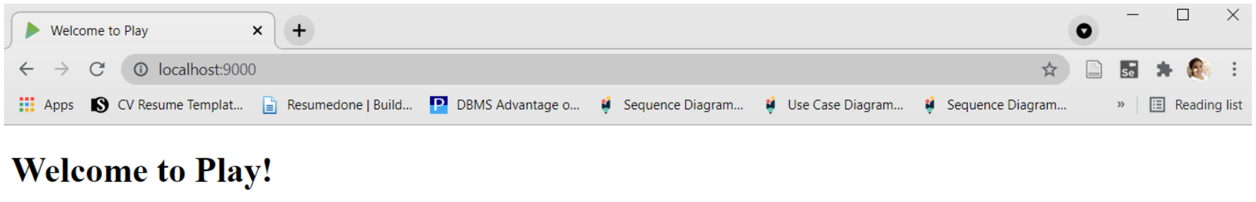
```
/**These are the dependencies that it necessary to be specified in order to establish mysql database connection*/
libraryDependencies ++= Seq(
  jdbc
)
libraryDependencies ++= Seq(
  "mysql" % "mysql-connector-java" % "5.1.41"
)
```

After doing this, we have to run following commands. **Sbt, update, compile, run**



```
C:\Users\wathsala\fp_assignment>sbt
sbt thinks that server is already booting because of this exception:
java.io.IOException: Could not create lock for \\.\pipe\sbt-load-9193127810121580527_lock, error 5
Create a new server? y/n (default y)
y
[info] welcome to sbt 1.5.2 (Oracle Corporation Java 1.8.0_291)
[info] loading global plugins from C:\Users\wathsala\.sbt\1.0\plugins
[info] loading settings for project fp_assignment-build from plugins.sbt ...
[info] loading project definition from C:\Users\wathsala\fp_assignment\project
[info] loading settings for project root from build.sbt ...
[info] set current project to fp_assignment (in build file:/C:/Users/wathsala/fp_assignment/)
[error] server failed to start on local:sbt-server-9c445444b9e5f4a092e4. java.io.IOException: Could not create lock for \\.\pipe\sbt-server-9c445444b9e5f4a092e4_lock, error 5
[info] sbt server started at local:sbt-server-9c445444b9e5f4a092e4
[info] started sbt server
[fp_assignment] $ update
[success] Total time: 15 s, completed Jul 9, 2021 7:26:14 AM
[fp_assignment] $ compile
[info] compiling 7 Scala sources and 1 Java source to C:\Users\wathsala\fp_assignment\target\scala-2.13\classes ...
[success] Total time: 14 s, completed Jul 9, 2021 7:26:33 AM
[fp_assignment] $ run

--- (Running the application, auto-reloading is enabled) ---

[info] p.c.s.AkkaHttpServer - Listening for HTTP on /0:0:0:0:0:0:0:0:9000

(Server started, use Enter to stop and go back to the console...)

2021-07-09 07:27:04 INFO  play.api.http.EnabledFilters  Enabled Filters (see <https://www.playframework.com/documentation/latest/Filters>):
```

Then the local host/9000 gives us following result.

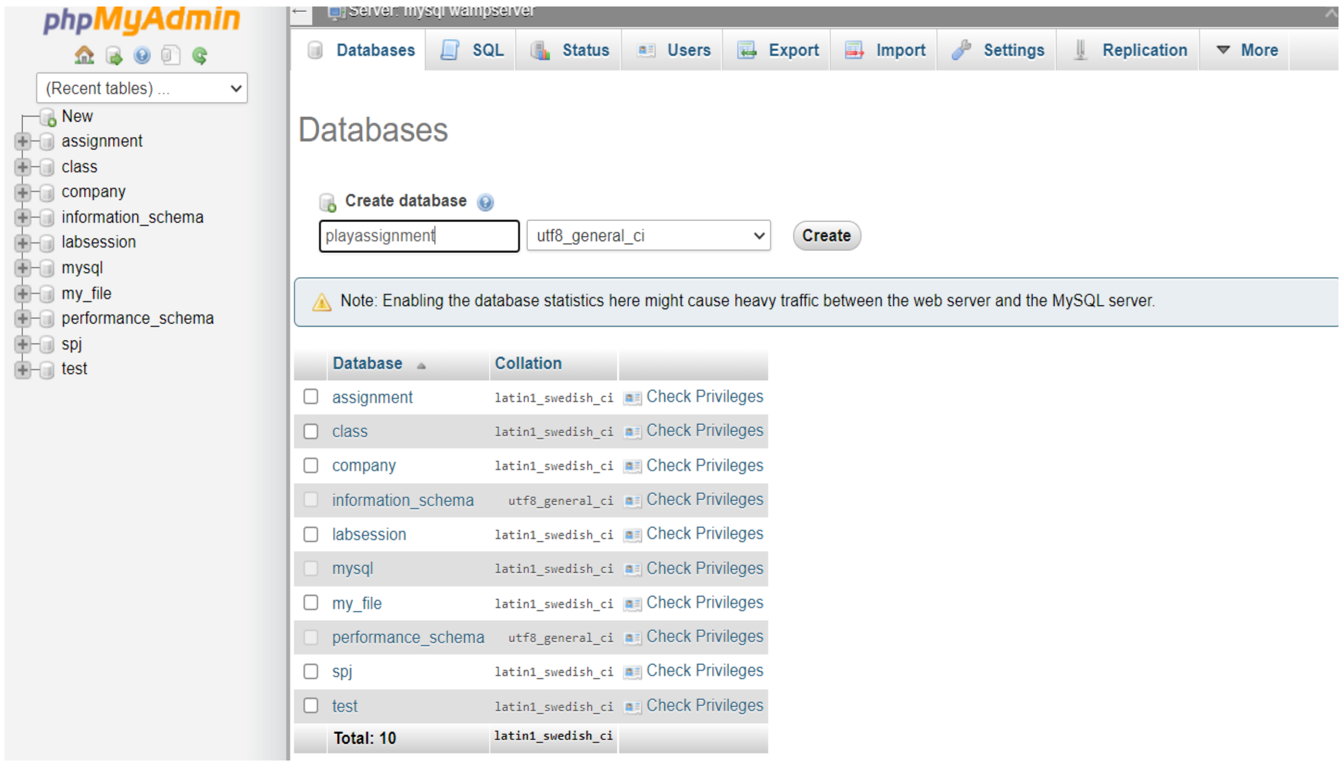


Welcome to Play!

3.3 Create the database

For create the database we have to visit http://localhost/phpmyadmin/.

Create new database.



Databases

Create database

playassignment | utf8_general_ci

Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server.

	Database	Collation	
☐	assignment	latin1_swedish_ci	Check Privileges
☐	class	latin1_swedish_ci	Check Privileges
☐	company	latin1_swedish_ci	Check Privileges
☐	information_schema	utf8_general_ci	Check Privileges
☐	labsession	latin1_swedish_ci	Check Privileges
☐	mysql	latin1_swedish_ci	Check Privileges
☐	my_file	latin1_swedish_ci	Check Privileges
☐	performance_schema	utf8_general_ci	Check Privileges
☐	spj	latin1_swedish_ci	Check Privileges
☐	test	latin1_swedish_ci	Check Privileges
	Total: 10	latin1_swedish_ci	

Create new table in our database.

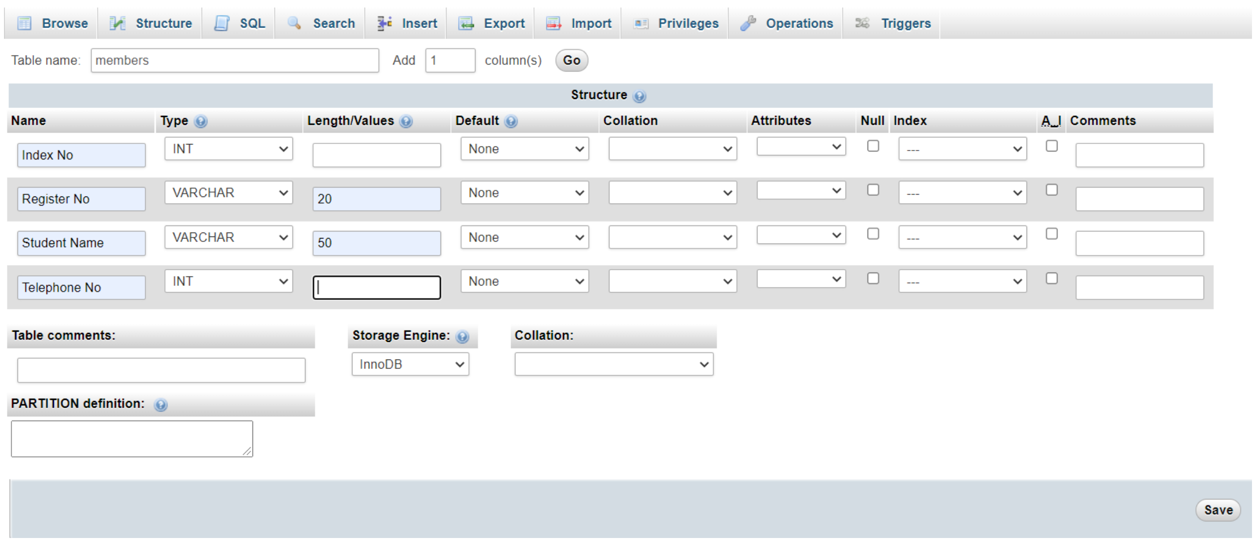


Table name: members Add 1 column(s) Go

Structure

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
Index No	INT		None			☐	---	☐	
Register No	VARCHAR	20	None			☐	---	☐	
Student Name	VARCHAR	50	None			☐	---	☐	
Telephone No	INT		None			☐	---	☐	

Table comments:

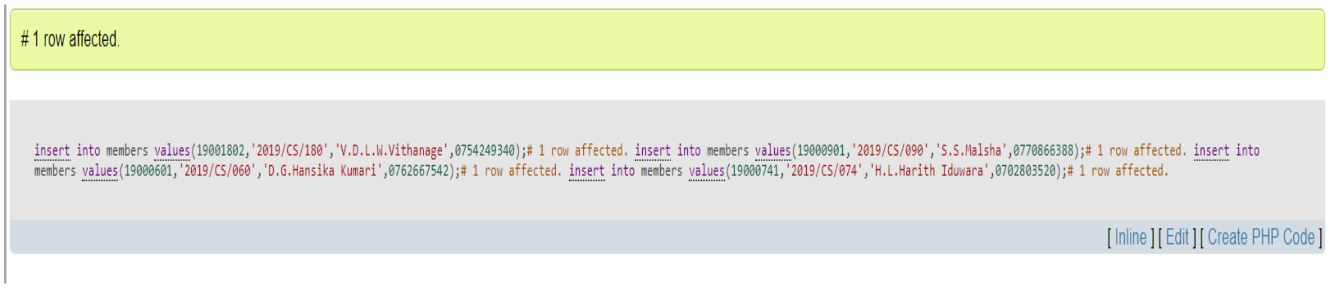
Storage Engine: InnoDB

Collation:

PARTITION definition:

Save

Then feed our member details into that table.

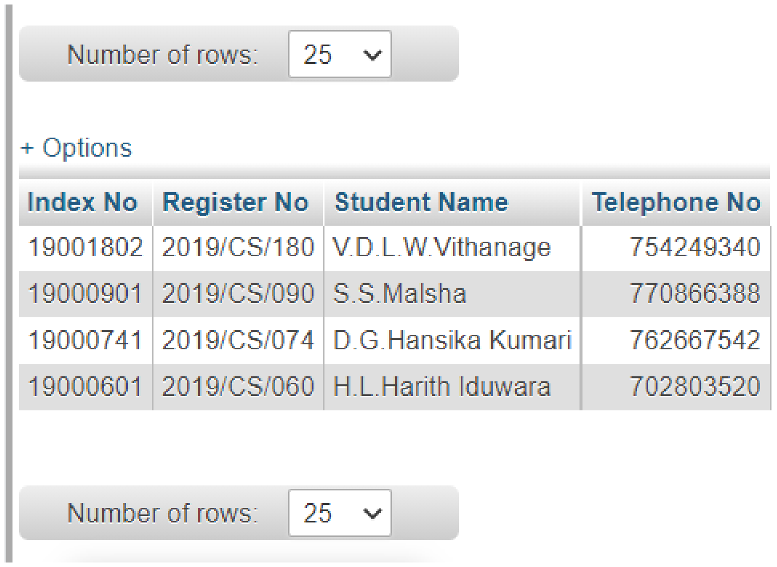


```
# 1 row affected.
```

```
insert into members values(19001802,'2019/CS/180','V.D.L.W.Vithanage',0754249340);# 1 row affected. insert into members values(19000901,'2019/CS/090','S.S.Malsha',0770866388);# 1 row affected. insert into
members values(19000601,'2019/CS/060','D.G.Hansika Kumari',0762667542);# 1 row affected. insert into members values(19000741,'2019/CS/074','H.L.Harith Iduwara',0702803520);# 1 row affected.
```

Now our table loos like;

.

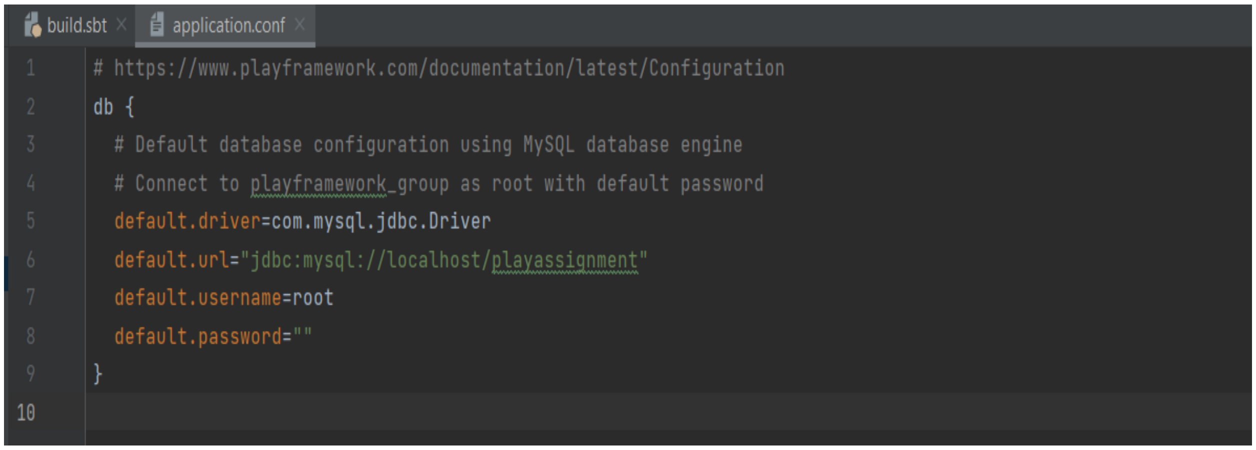


Index No	Register No	Student Name	Telephone No
19001802	2019/CS/180	V.D.L.W.Vithanage	754249340
19000901	2019/CS/090	S.S.Malsha	770866388
19000741	2019/CS/074	D.G.Hansika Kumari	762667542
19000601	2019/CS/060	H.L.Harith Iduwara	702803520

Database creation part is also now over.

3.4 ADD configurations

The configurations of our projects are in configurations files of play project. (Application.conf file). So we must add default database configurations to that file in order to connect with the database.



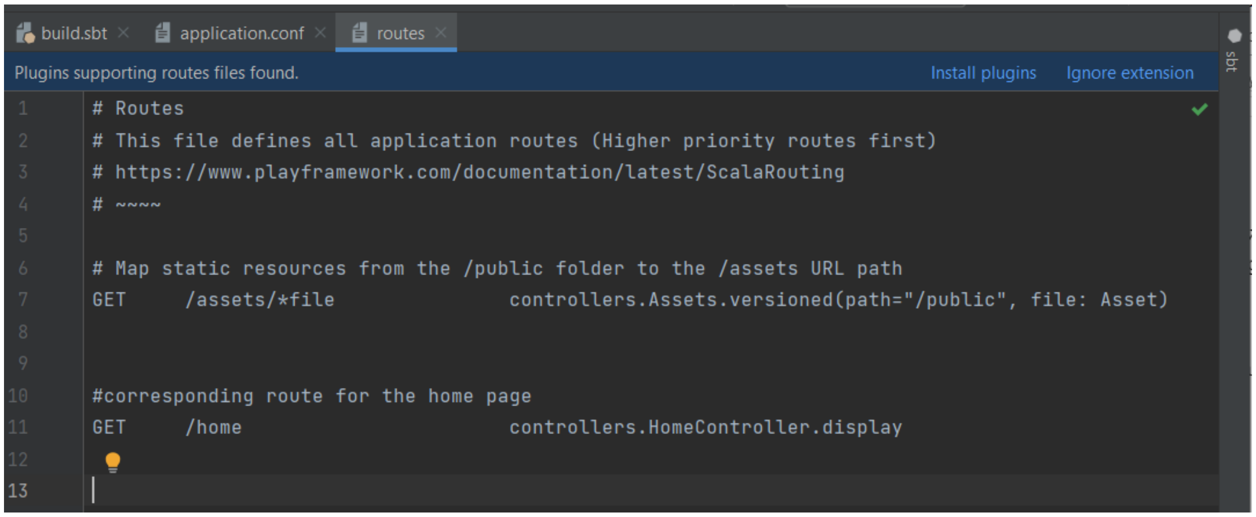
```
# https://www.playframework.com/documentation/latest/Configuration
db {
  # Default database configuration using MySQL database engine
  # Connect to playframework_group as root with default password
  default.driver=com.mysql.jdbc.Driver
  default.url="jdbc:mysql://localhost/playassignment"
  default.username=root
  default.password=""
}
```

All the dependencies and configurations are fine now.

3.5 ADD routes

We are doing our project according to the MVC architecture. In MVC architecture each view have its own controller and the routes. Routes are predefined paths for the controller. The transaction is take placed according to that specified rotes.

In play framework routes are shown in routes file. (Conf->routes)

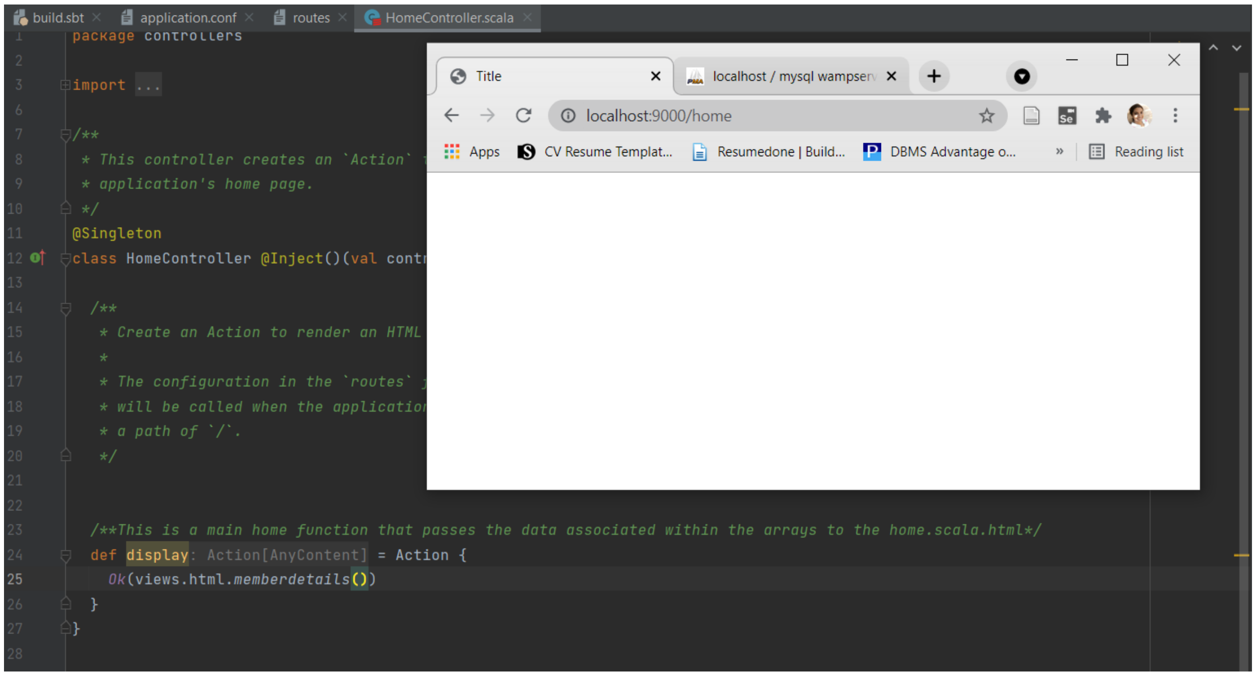


```
# Routes
# This file defines all application routes (Higher priority routes first)
# https://www.playframework.com/documentation/latest/ScalaRouting
# ~~~~

# Map static resources from the /public folder to the /assets URL path
GET     /assets/*file               controllers.Assets.versioned(path="/public", file: Asset)


#corresponding route for the home page
GET     /home                       controllers.HomeController.display
```

Route correspond to the home page is added. The URL (localhost/9000/home) we type in our browser will go to the HomeController.scala file and will execute things in the display method.

3.6 ADD display method

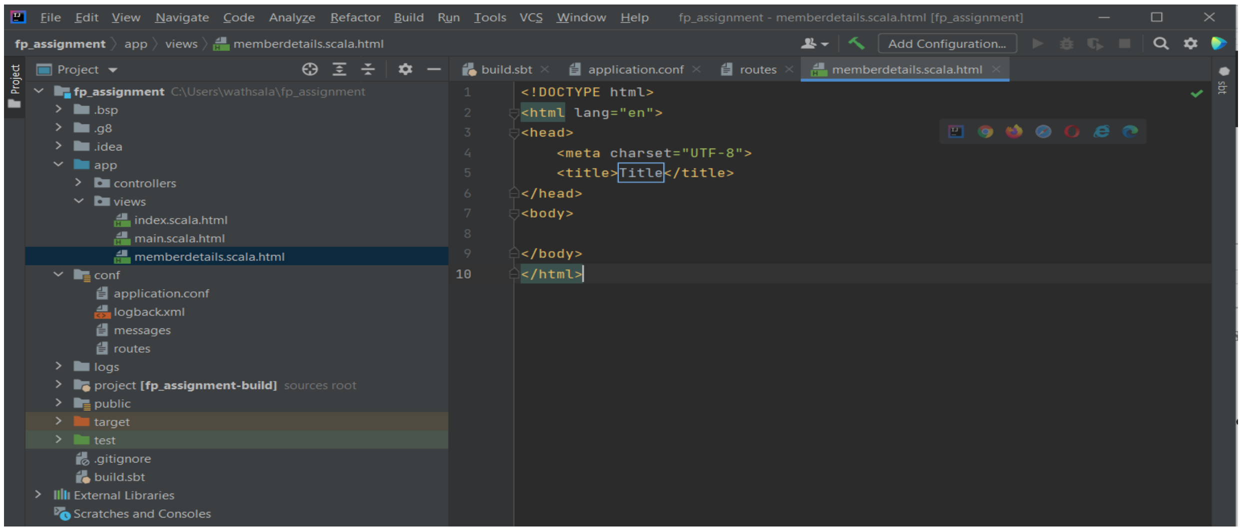
As we mention above after the URL runs it goes to the HomeController file and execute the specified method. In our case it is “display method”. We must add it to the HomeController file.



```
package controllers

import ...

/**
 * This controller creates an `Action`
 * application's home page.
 */
@Singleton
class HomeController @Inject()(val contr

  /**
   * Create an Action to render an HTML
   *
   * The configuration in the `routes`
   * will be called when the applicatio
   * a path of `/`.
   */


  /**This is a main home function that passes the data associated within the arrays to the home.scala.html*/
  def display: Action[AnyContent] = Action {
    Ok(views.html.memberdetails())
  }
}
```

3.7 ADD view file

In MVC architecture view define the layer which the user will be interacted. User see whatever the view file is shown. In our project we will say our view file as **“memeberdetails.scala.html”.** We must create it under the view files section. Then our home page will display whatever in our view file.

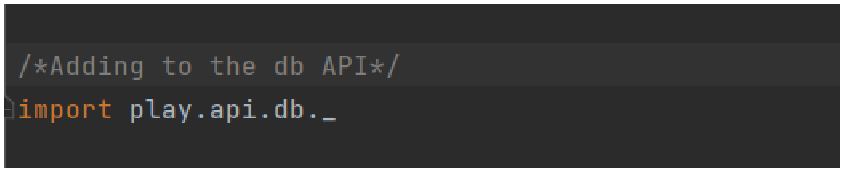


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>

</body>
</html>
```

3.8 Set up HomeController (Embed MYSQL)

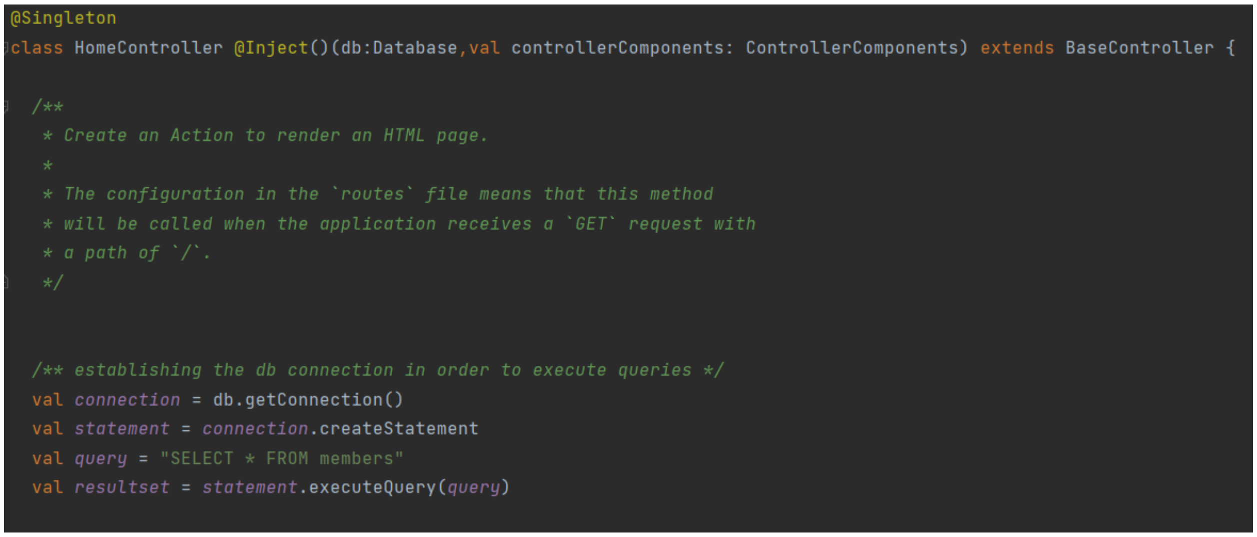
When setting up HomeController first delete al the unnecessary files. Then follow the following steps.

- Import db API.



```
/*Adding to the db API*/
import play.api.db._
```

- Set the database connection.

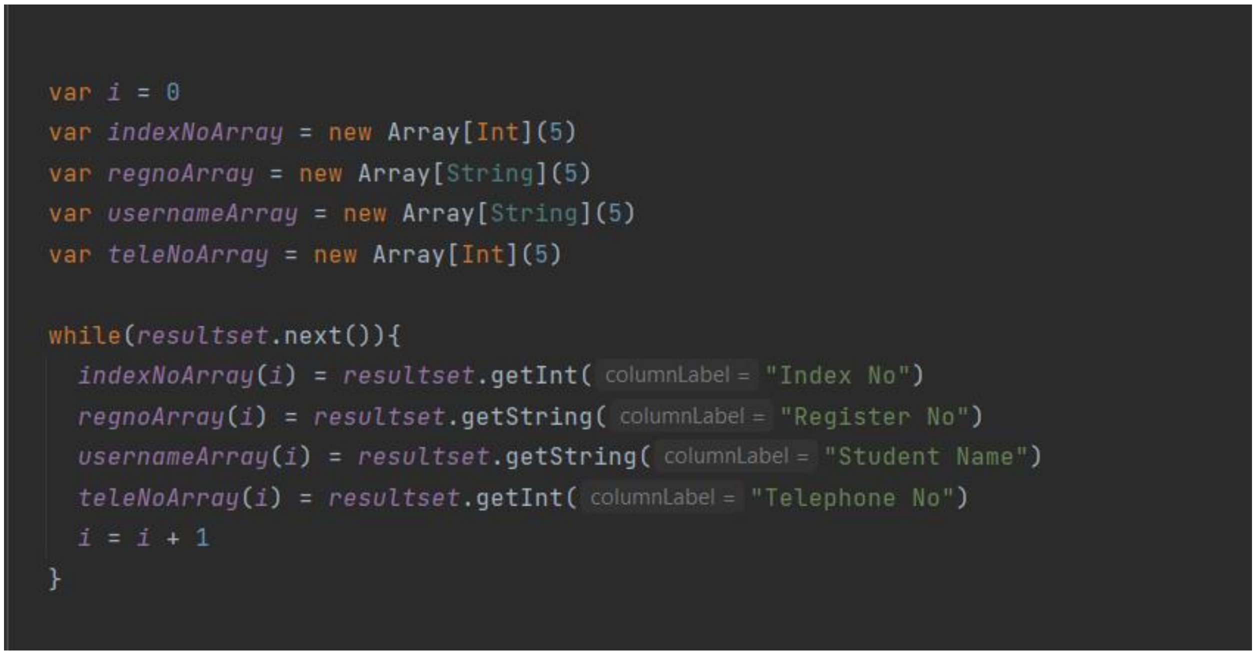


```
@Singleton
class HomeController @Inject()(db:Database,val controllerComponents: ControllerComponents) extends BaseController {

  /**
   * Create an Action to render an HTML page.
   *
   * The configuration in the `routes` file means that this method
   * will be called when the application receives a `GET` request with
   * a path of `/`.
   */


  /** establishing the db connection in order to execute queries */
  val connection = db.getConnection()
  val statement = connection.createStatement
  val query = "SELECT * FROM members"
  val resultset = statement.executeQuery(query)
```

First we have set the database connection (db connection) using “get.Conection () method”. Then we have created a statement using it and then wrote a query string (with the type of Val). Then the results of the query was stored at resultset. Actually resultset is a combination of many values. It stored whole query results. We have to get one by one those results using arrays.

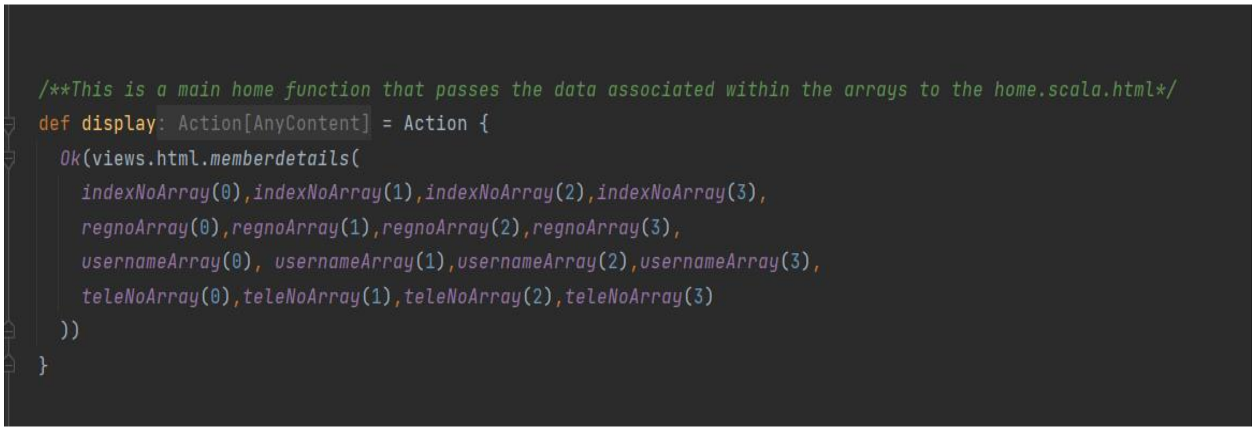
Our table in playassignment database have four columns. (Index No, Register No, Student Name, Telephone No)

- Setting Arrays



```
var i = 0
var indexNoArray = new Array[Int](5)
var regnoArray = new Array[String](5)
var usernameArray = new Array[String](5)
var teleNoArray = new Array[Int](5)

while(resultset.next()){
  indexNoArray(i) = resultset.getInt( columnLabel = "Index No")
  regnoArray(i) = resultset.getString( columnLabel = "Register No")
  usernameArray(i) = resultset.getString( columnLabel = "Student Name")
  teleNoArray(i) = resultset.getInt( columnLabel = "Telephone No")
  i = i + 1
}
```

- Pass data to the display view

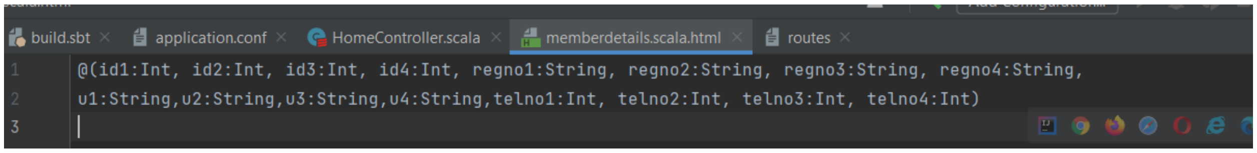


```
/**This is a main home function that passes the data associated within the arrays to the home.scala.html*/
def display: Action[AnyContent] = Action {
  Ok(views.html.memberdetails(
    indexNoArray(0),indexNoArray(1),indexNoArray(2),indexNoArray(3),
    regnoArray(0),regnoArray(1),regnoArray(2),regnoArray(3),
    usernameArray(0), usernameArray(1),usernameArray(2),usernameArray(3),
    teleNoArray(0),teleNoArray(1),teleNoArray(2),teleNoArray(3)
  ))
}
```

Now the HomeController setting part is over.

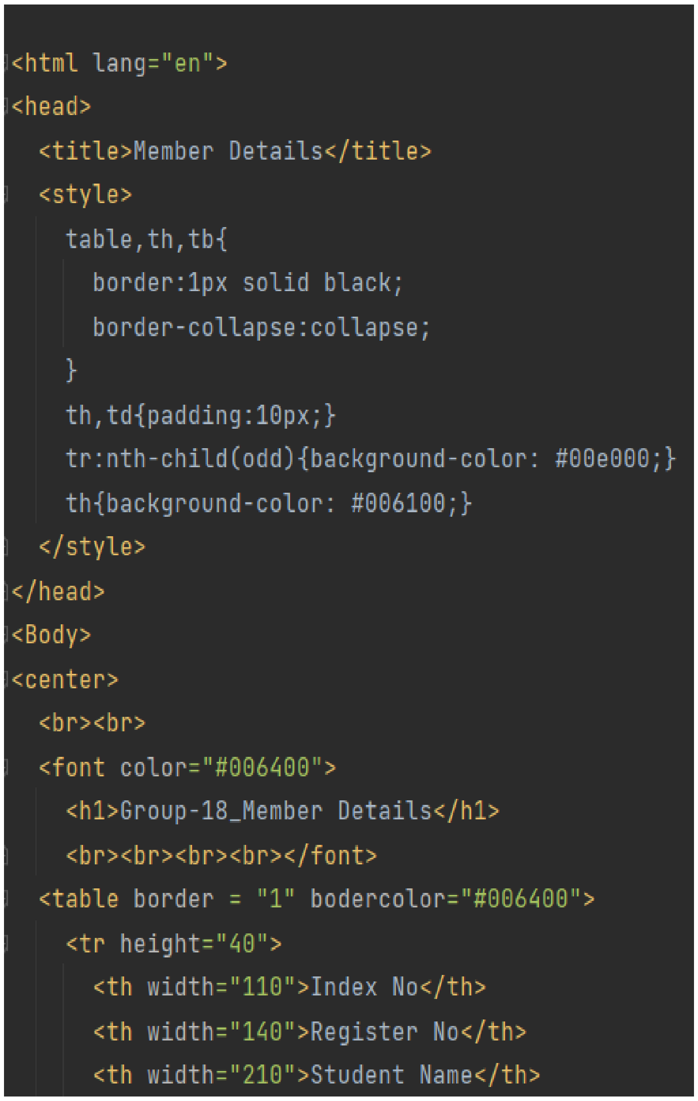
3.9 Set up view file

➢ Get data from HomeController file.

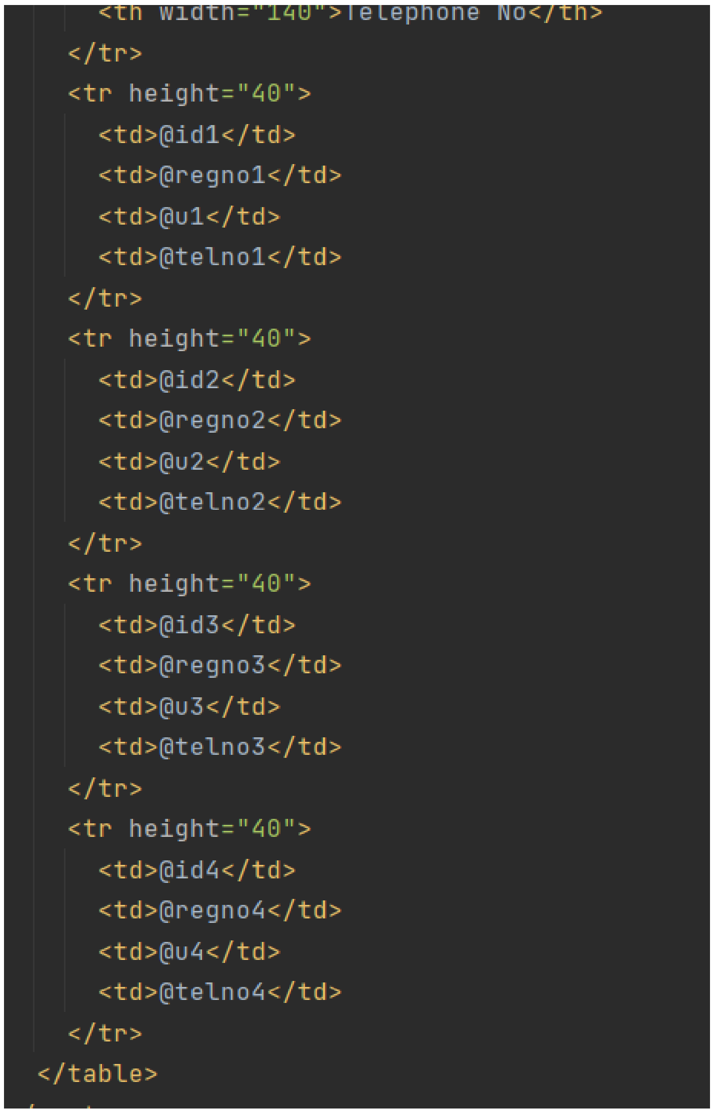


```
@(id1:Int, id2:Int, id3:Int, id4:Int, regno1:String, regno2:String, regno3:String, regno4:String,
u1:String,u2:String,u3:String,u4:String,telno1:Int, telno2:Int, telno3:Int, telno4:Int)
```

➢ Create display view to display the table as output.



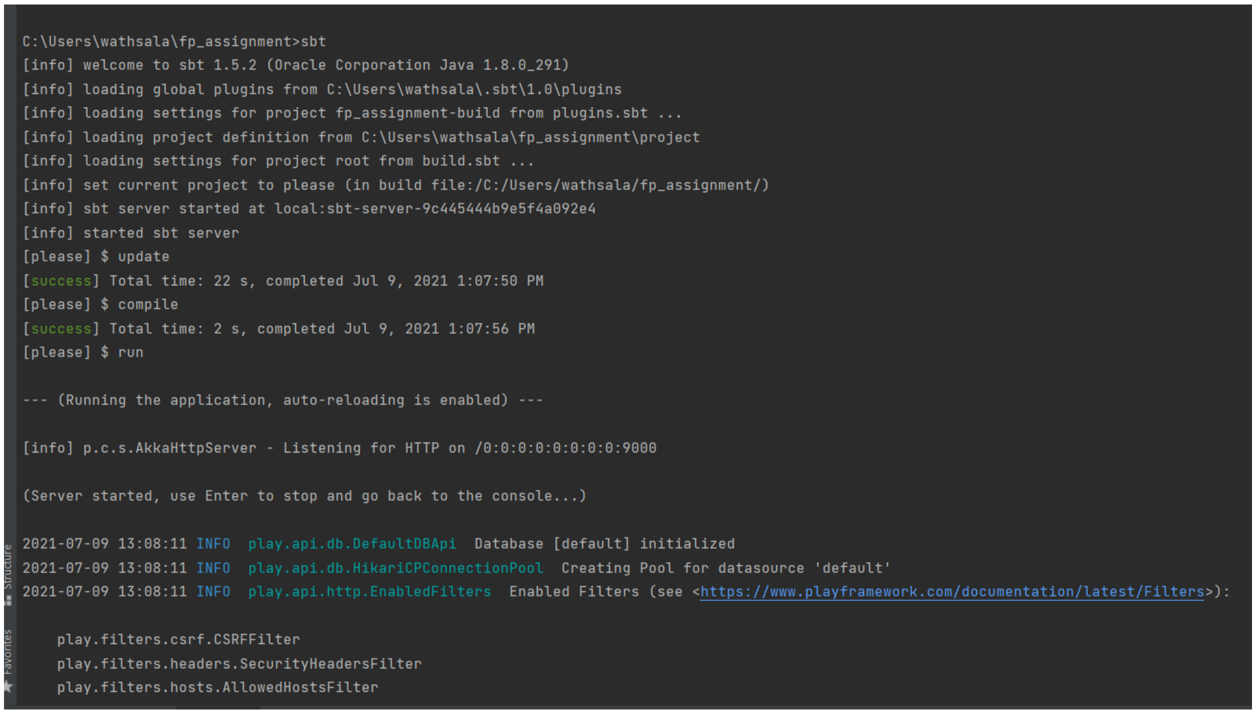
```
<html lang="en">
<head>
  <title>Member Details</title>
  <style>
    table,th,tb{
      border:1px solid black;
      border-collapse:collapse;
    }
    th,td{padding:10px;}
    tr:nth-child(odd){background-color: #00e000;}
    th{background-color: #006100;}
  </style>
</head>
<Body>
<center>
  <br><br>
  <font color="#006400">
    <h1>Group-18_Member Details</h1>
    <br><br><br><br></font>
  <table border = "1" bodercolor="#006400">
    <tr height="40">
      <th width="110">Index No</th>
      <th width="140">Register No</th>
      <th width="210">Student Name</th>
```



```
      <th width="140">Telephone No</th>
    </tr>
    <tr height="40">
      <td>@id1</td>
      <td>@regno1</td>
      <td>@u1</td>
      <td>@telno1</td>
    </tr>
    <tr height="40">
      <td>@id2</td>
      <td>@regno2</td>
      <td>@u2</td>
      <td>@telno2</td>
    </tr>
    <tr height="40">
      <td>@id3</td>
      <td>@regno3</td>
      <td>@u3</td>
      <td>@telno3</td>
    </tr>
    <tr height="40">
      <td>@id4</td>
      <td>@regno4</td>
      <td>@u4</td>
      <td>@telno4</td>
    </tr>
  </table>
```

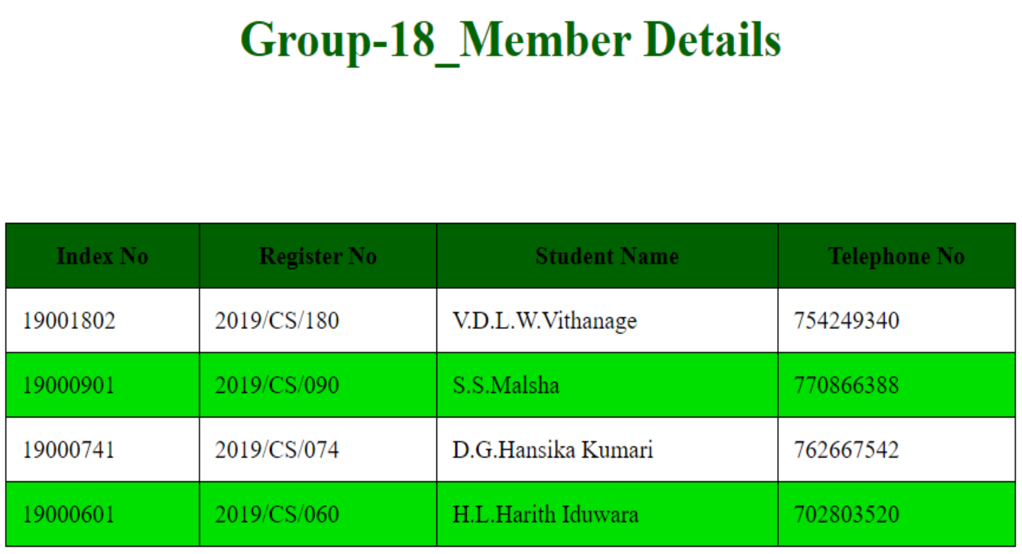
The final step:

Run following steps in the terminal; (**Sbt, update, compile, run**)



```
C:\Users\wathsala\fp_assignment>sbt
[info] welcome to sbt 1.5.2 (Oracle Corporation Java 1.8.0_291)
[info] loading global plugins from C:\Users\wathsala\.sbt\1.0\plugins
[info] loading settings for project fp_assignment-build from plugins.sbt ...
[info] loading project definition from C:\Users\wathsala\fp_assignment\project
[info] loading settings for project root from build.sbt ...
[info] set current project to please (in build file:/C:/Users/wathsala/fp_assignment/)
[info] sbt server started at local:sbt-server-9c445444b9e5f4a092e4
[info] started sbt server
[please] $ update
[success] Total time: 22 s, completed Jul 9, 2021 1:07:50 PM
[please] $ compile
[success] Total time: 2 s, completed Jul 9, 2021 1:07:56 PM
[please] $ run

--- (Running the application, auto-reloading is enabled) ---

[info] p.c.s.AkkaHttpServer - Listening for HTTP on /0:0:0:0:0:0:0:0:9000

(Server started, use Enter to stop and go back to the console...)

2021-07-09 13:08:11 INFO  play.api.db.DefaultDBApi  Database [default] initialized
2021-07-09 13:08:11 INFO  play.api.db.HikariCPConnectionPool  Creating Pool for datasource 'default'
2021-07-09 13:08:11 INFO  play.api.http.EnabledFilters  Enabled Filters (see <https://www.playframework.com/documentation/latest/Filters>):

    play.filters.csrf.CSRFFilter
    play.filters.headers.SecurityHeadersFilter
    play.filters.hosts.AllowedHostsFilter
```

Localhost/9000/home



Group-18_Member Details

Index No	Register No	Student Name	Telephone No
19001802	2019/CS/180	V.D.L.W.Vithanage	754249340
19000901	2019/CS/090	S.S.Malsha	770866388
19000741	2019/CS/074	D.G.Hansika Kumari	762667542
19000601	2019/CS/060	H.L.Harith Iduwara	702803520

4. Resources used:

- Java 8 - https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html
- SBT - https://www.scala-lang.org/download/scala3.html
- IntellJ IDEA - https://www.jetbrains.com/idea/download/#section=windows
- Wamp Server - https://sourceforge.net/projects/wampserver/
- Play Framework - https://www.playframework.com/getting-started