Agile Software Development

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

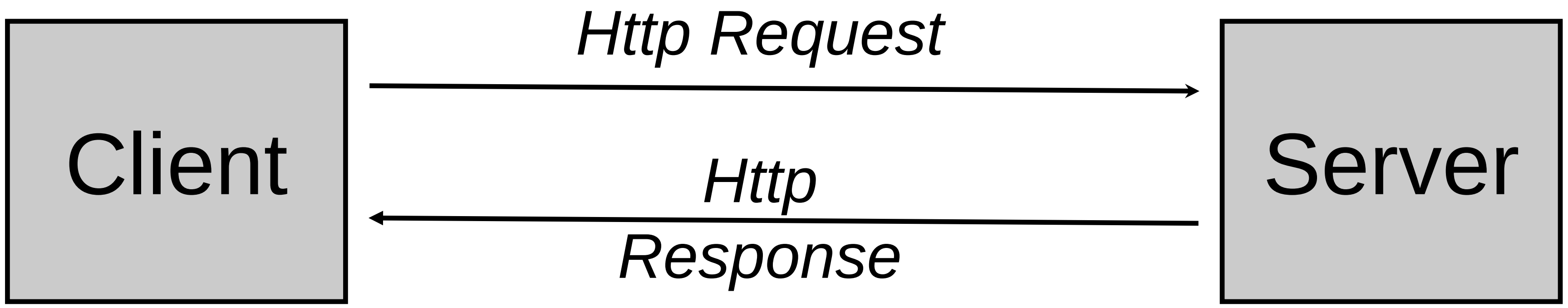
http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Play 2

Web Applications - Request/Response

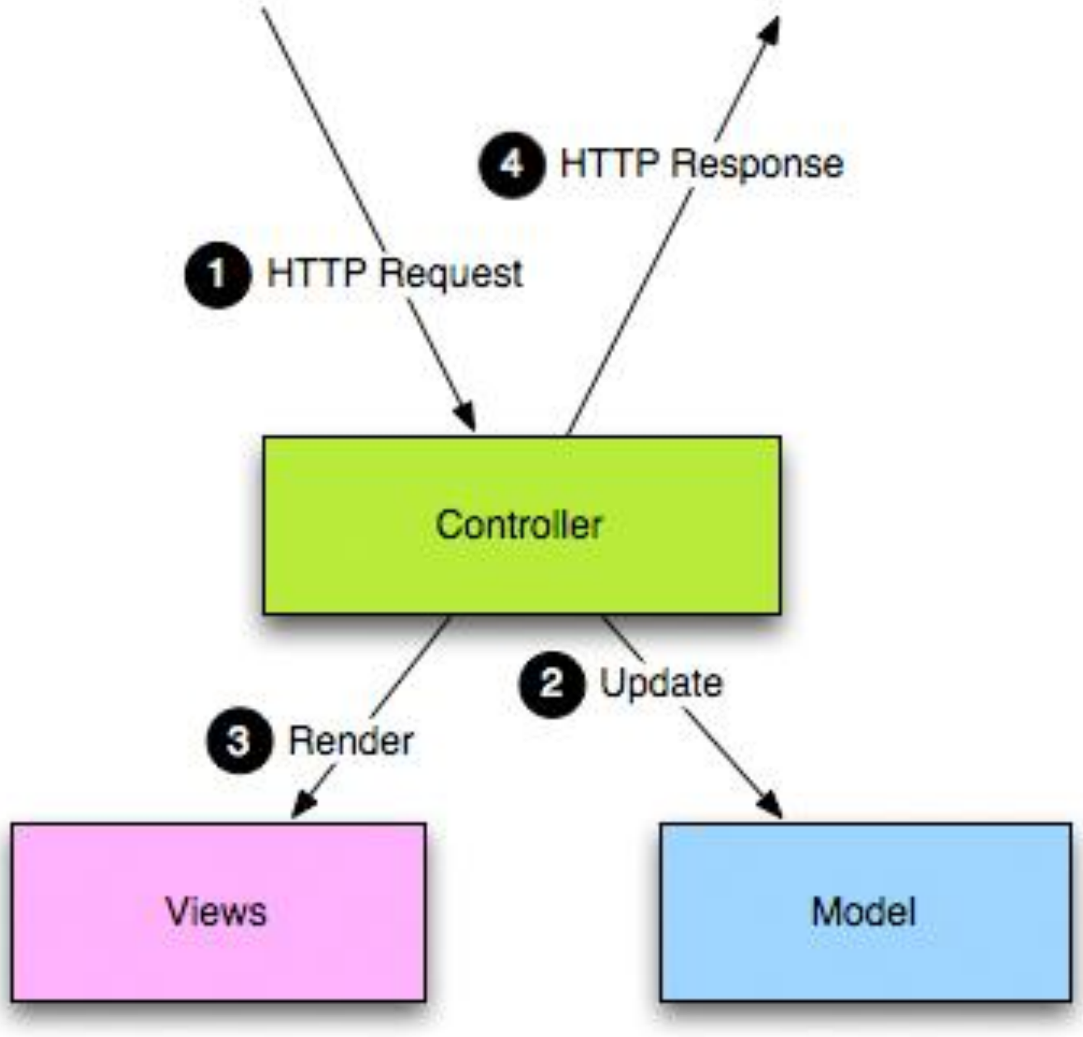


- Request - http request emitted by browser as a result to url in address bar, link, button or form submission on page.

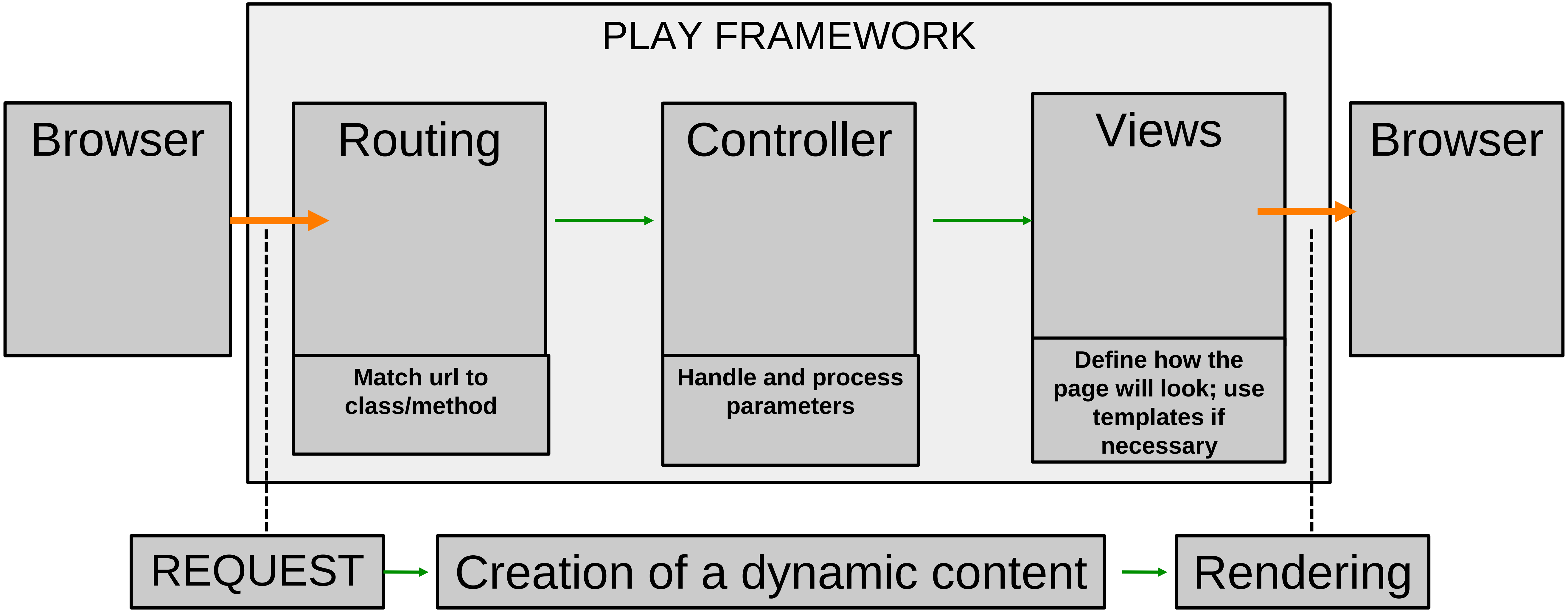
- Response - web page returned from service to be presented in browser.

Web Applications - MVC

- Model View Controller is a generally accepted pattern or separation of concerns within the server.
- **Model:** Core application domain model + database persistence.
- **View**: User Experience.
- **Controlle**r: Directly handle all requests, mediate with Model, build and assemble the response using the views.



Request/Response Lifecycle



PLAY FRAMEWORK

Browser

Routing

Match url to class/method

Controller

Handle and process parameters

Views

Define how the page will look; use templates if necessary

Browser

REQUEST

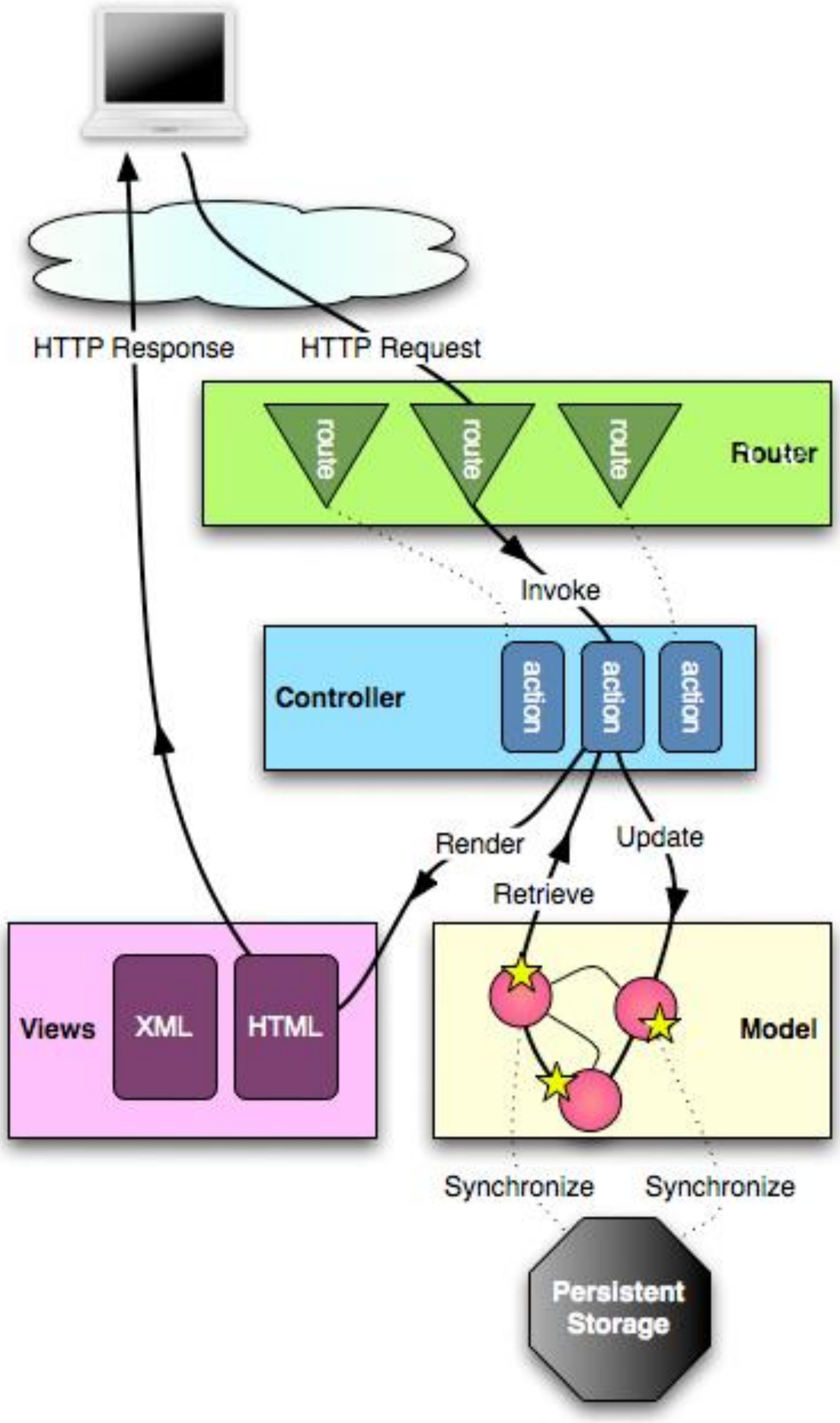
Creation of a dynamic content

Rendering

MVC in Play

- **Router:** examine incoming requests and match to corresponding Controller/Action.

- **Action:** a method in the controller.



Routes File Example (Pacemaker)

```
# UI

GET     /                                controllers.Accounts.index()
GET     /signup                          controllers.Accounts.signup()
GET     /login                           controllers.Accounts.login()
GET     /logout                          controllers.Accounts.logout()
POST    /register                        controllers.Accounts.register()
POST    /authenticate                    controllers.Accounts.authenticate()

GET     /dashboard                       controllers.Dashboard.index()
GET     /upload                          controllers.Dashboard.uploadActivityForm()
POST    /submitactivity                  controllers.Dashboard.submitActivity()

# API

GET     /api/users                       controllers.PacemakerAPI.users()
DELETE  /api/users                       controllers.PacemakerAPI.deleteAllUsers()
POST    /api/users                       controllers.PacemakerAPI.createUser()

GET     /api/users/:id                    controllers.PacemakerAPI.user(id: Long)
DELETE  /api/users/:id                    controllers.PacemakerAPI.deleteUser(id: Long)
PUT     /api/users/:id                    controllers.PacemakerAPI.updateUser(id: Long)

GET     /api/users/:userId/activities          controllers.PacemakerAPI.activities(userId: Long)
POST    /api/users/:userId/activities          controllers.PacemakerAPI.createActivity(userId: Long)

GET     /api/users/:userId/activities/:activityId  controllers.PacemakerAPI.activity(userId: Long, activityId:Long)
DELETE  /api/users/:userId/activities/:activityId  controllers.PacemakerAPI.deleteActivity(userId: Long, activityId:Long)
PUT     /api/users/:userId/activities/:activityId  controllers.PacemakerAPI.updateActivity(userId: Long, activityId:Long)

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

Routes - UI

```
# UI

GET     /                     controllers.Accounts.index()
GET     /signup               controllers.Accounts.signup()
GET     /login                controllers.Accounts.login()
GET     /logout               controllers.Accounts.logout()
POST    /register             controllers.Accounts.register()
POST    /authenticate         controllers.Accounts.authenticate()

GET     /dashboard            controllers.Dashboard.index()
GET     /upload               controllers.Dashboard.uploadActivityForm()
POST   /submitactivity        controllers.Dashboard.submitActivity()
```

- Routes to deliver UI.

- Each of these routes appears in views folder.

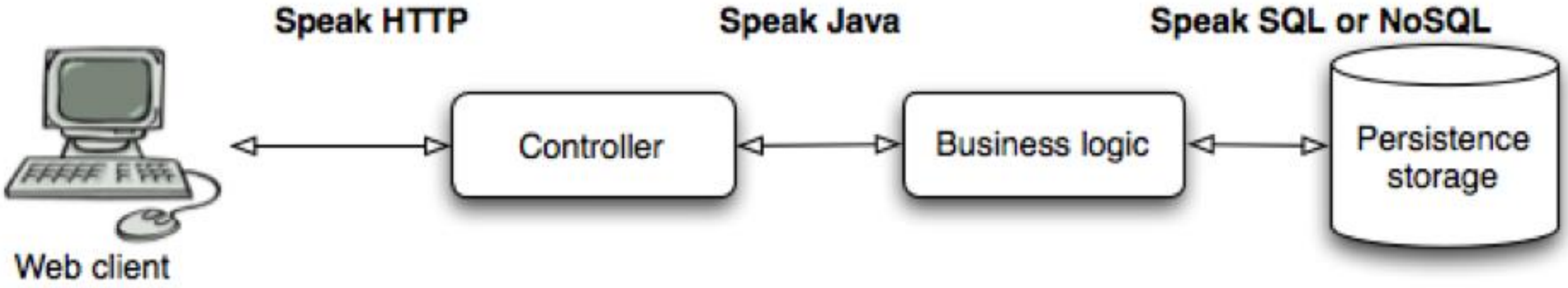
- Each of these actions generates and returns a complete HTML page.

Routes - API

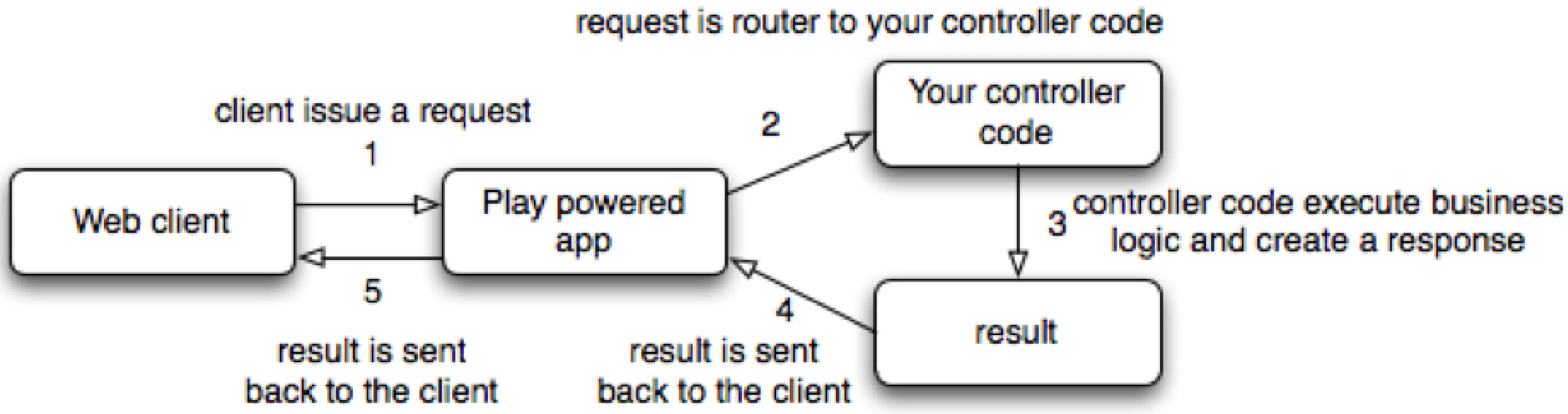
```
# API

GET      /api/users                              controllers.PacemakerAPI.users()
DELETE  /api/users                               controllers.PacemakerAPI.deleteAllUsers()
POST    /api/users                               controllers.PacemakerAPI.createUser()

GET      /api/users/:id                           controllers.PacemakerAPI.user(id: Long)
DELETE /api/users/:id                             controllers.PacemakerAPI.deleteUser(id: Long)
PUT      /api/users/:id                           controllers.PacemakerAPI.updateUser(id: Long)

GET      /api/users/:userId/activities                controllers.PacemakerAPI.activities(userId: Long)
POST    /api/users/:userId/activities                controllers.PacemakerAPI.createActivity(userId: Long)

GET      /api/users/:userId/activities/:activityId    controllers.PacemakerAPI.activity(userId: Long, activityId:Long)
DELETE  /api/users/:userId/activities/:activityId  controllers.PacemakerAPI.deleteActivity(userId: Long, activityId:Long)
PUT      /api/users/:userId/activities/:activityId    controllers.PacemakerAPI.updateActivity(userId: Long, activityId:Long)
```

- Routes to deliver API.
- Each of these routes is an API Request.
- Each of these actions generates and returns a JSON payload.

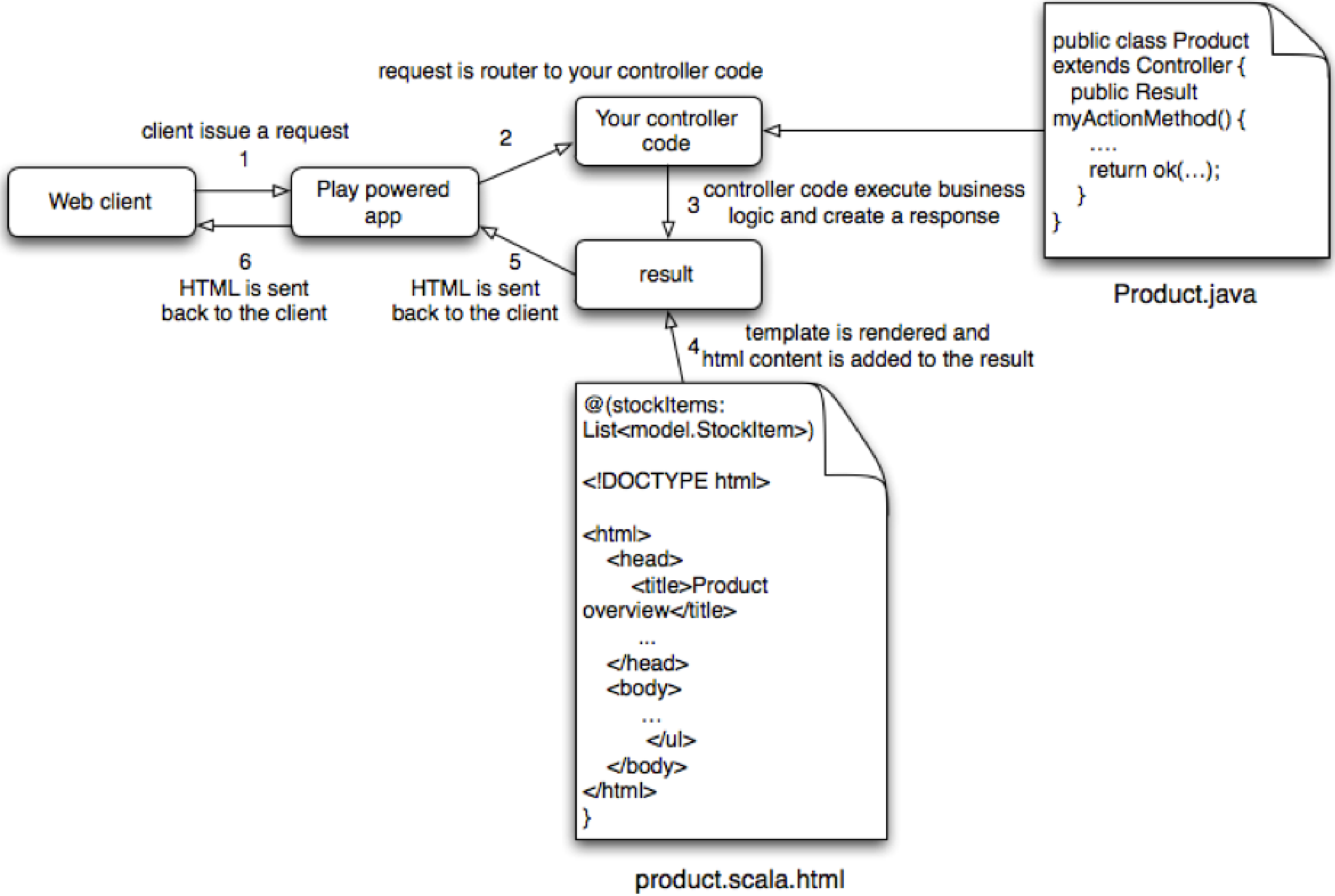
Role of Controller



Controller Lifecycle


request is router to your controller code
client issue a request
1
2
Your controller code
Web client
Play powered app
3
controller code execute business logic and create a response
5
4
result
result is sent back to the client
result is sent back to the client

Controller Lifecycle (detail)

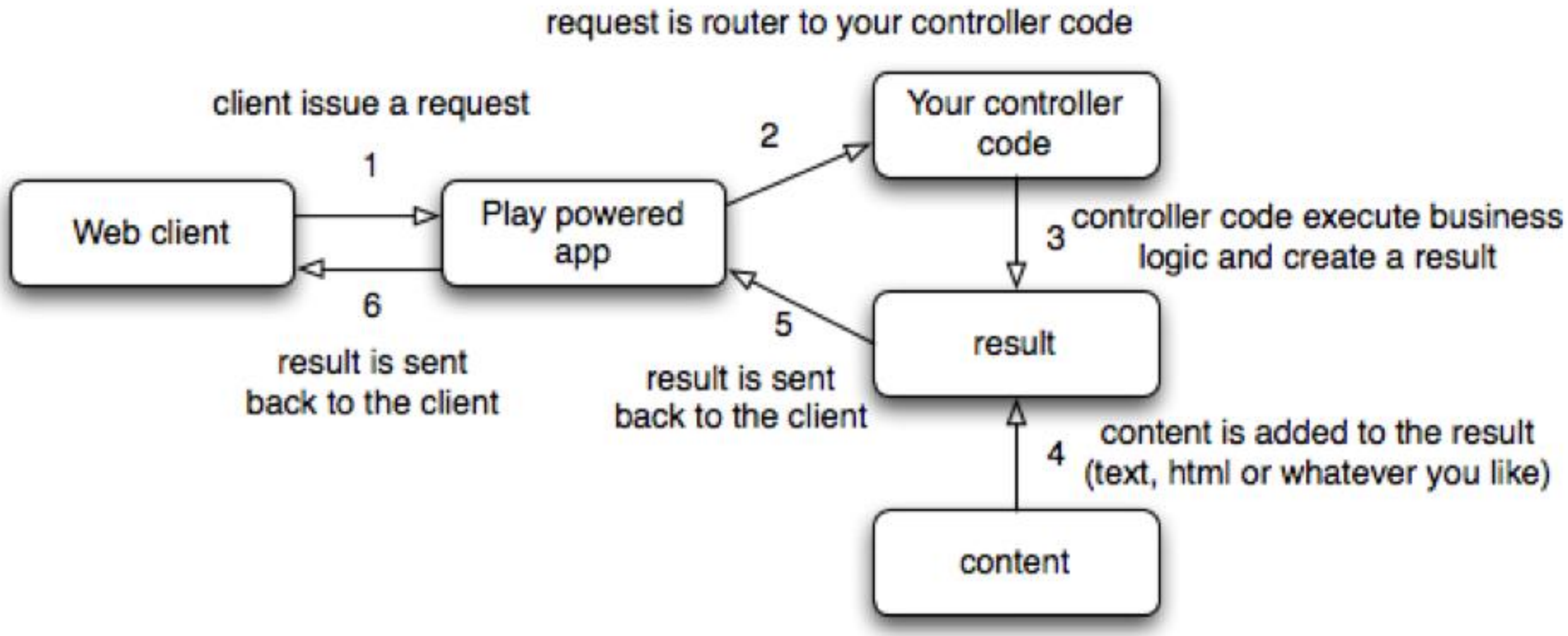


request is router to your controller code
client issue a request
1
2
Web client
Play powered app
Your controller code
3
controller code execute business logic and create a response
result
6
HTML is sent back to the client
5
HTML is sent back to the client
4
template is rendered and html content is added to the result
Product.java
product.scala.html

```
public class Product
extends Controller {
   public Result
myActionMethod() {
     ....
     return ok(...);
   }
}
```

```
@(stockItems:
List<model.StockItem>)

<!DOCTYPE html>

<html>
   <head>
      <title>Product
overview</title>
        ...
   </head>
   <body>
        ...
         </ul>
   </body>
</html>
}
```

Controller Lifecycle with Content


request is router to your controller code
client issue a request
1
2
Your controller code
Web client
Play powered app
3
controller code execute business logic and create a result
6
5
result
result is sent back to the client
result is sent back to the client
4
content is added to the result (text, html or whatever you like)
content

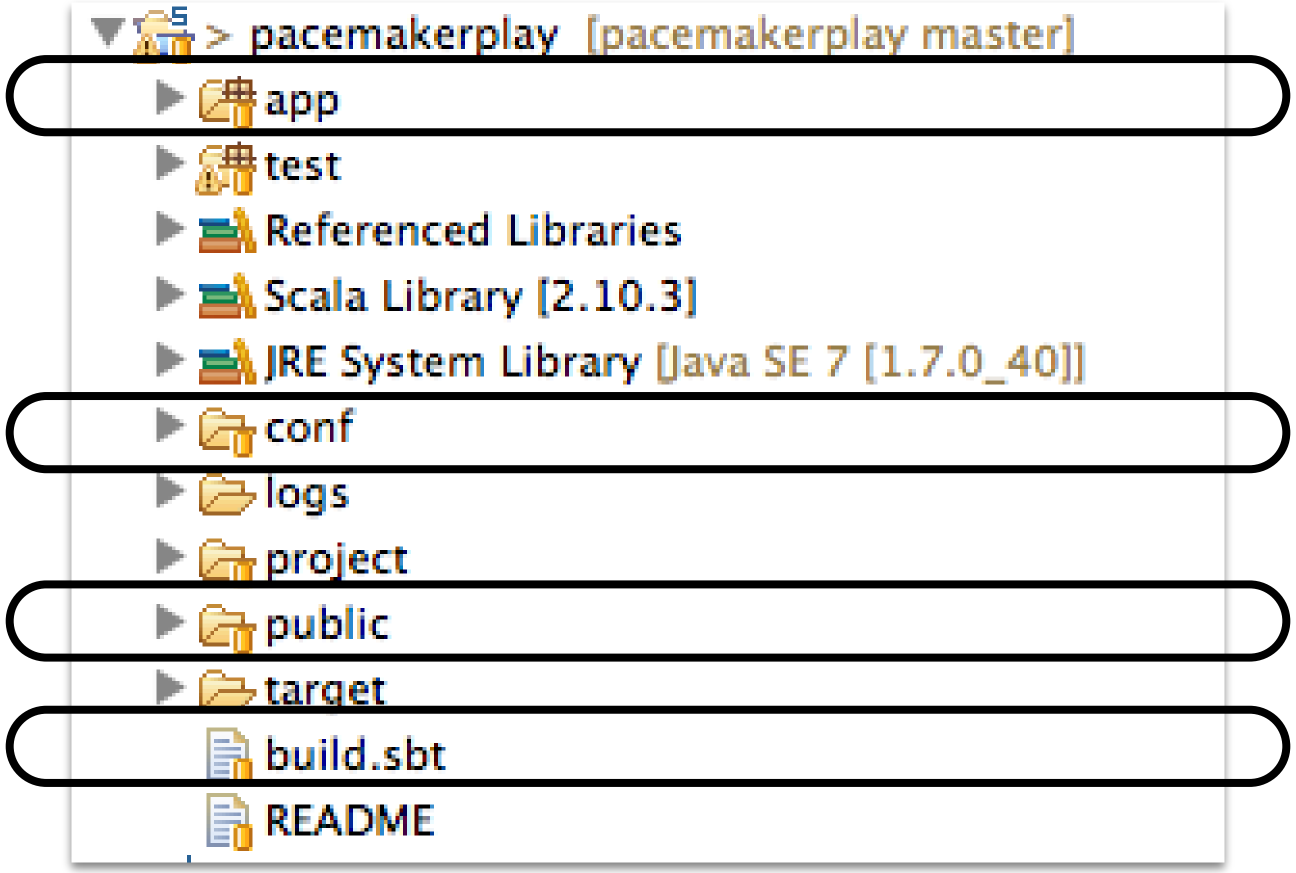
Play Project Anatomy

app

conf

public

build.sbt



pacemakerplay [pacemakerplay master]

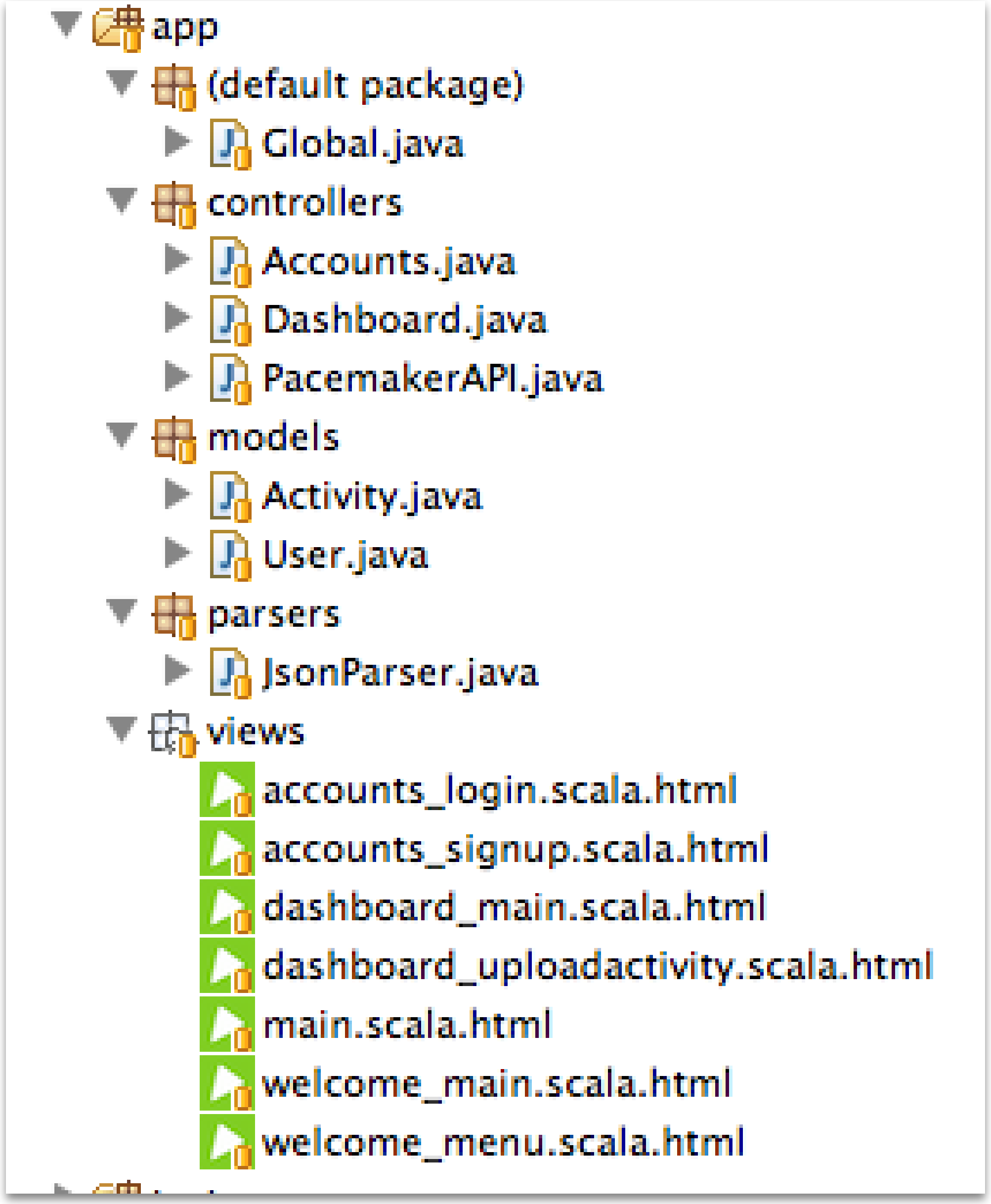
- app
- test
- Referenced Libraries
- Scala Library [2.10.3]
- JRE System Library [Java SE 7 [1.7.0_40]]
- conf
- logs
- project
- public
- target
- build.sbt
- README

app folder

controllers

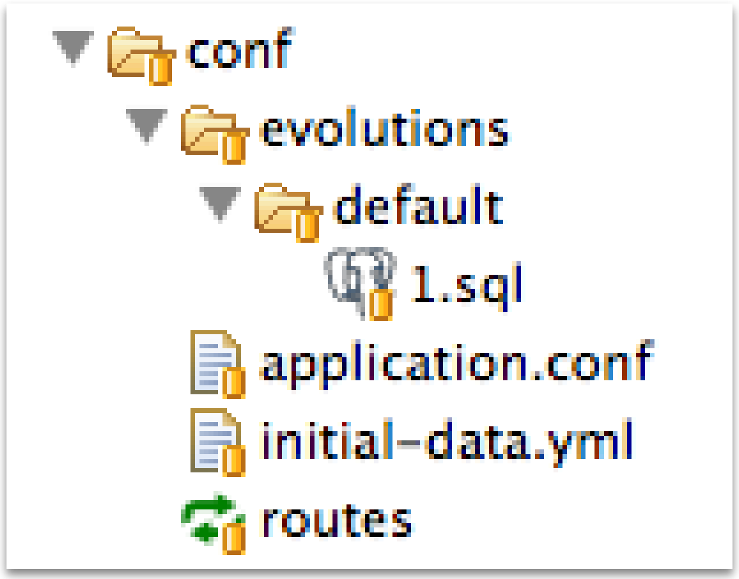
models

views



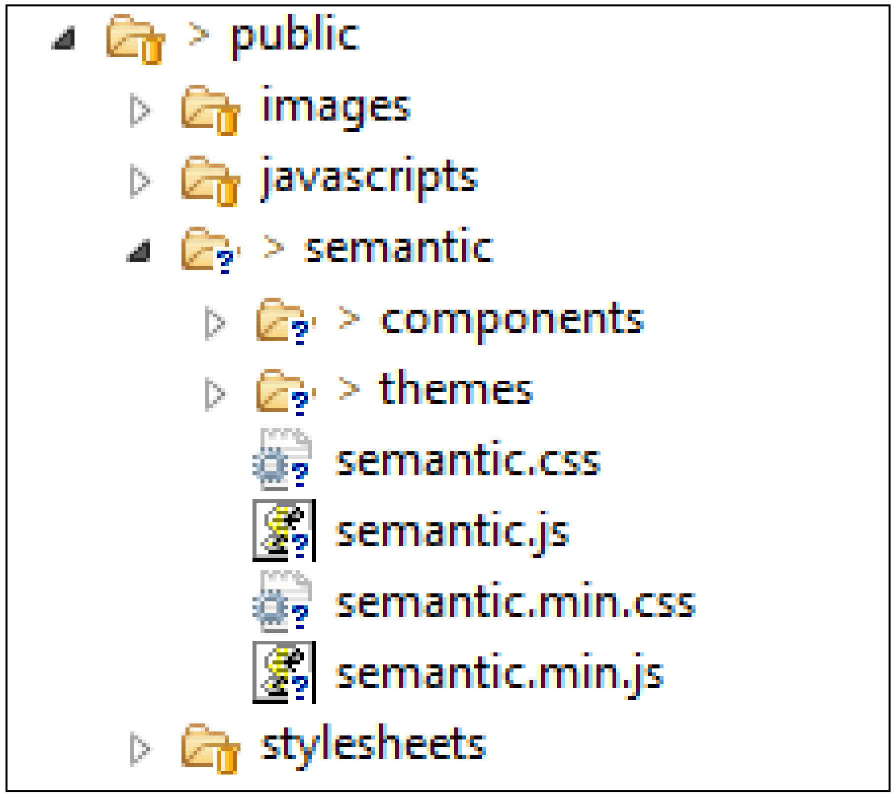
```
app
  (default package)
    Global.java
  controllers
    Accounts.java
    Dashboard.java
    PacemakerAPI.java
  models
    Activity.java
    User.java
  parsers
    JsonParser.java
  views
    accounts_login.scala.html
    accounts_signup.scala.html
    dashboard_main.scala.html
    dashboard_uploadactivity.scala.html
    main.scala.html
    welcome_main.scala.html
    welcome_menu.scala.html
```

conf folder



```
conf
  evolutions
    default
      1.sql
  application.conf
  initial-data.yml
  routes
```

public folder

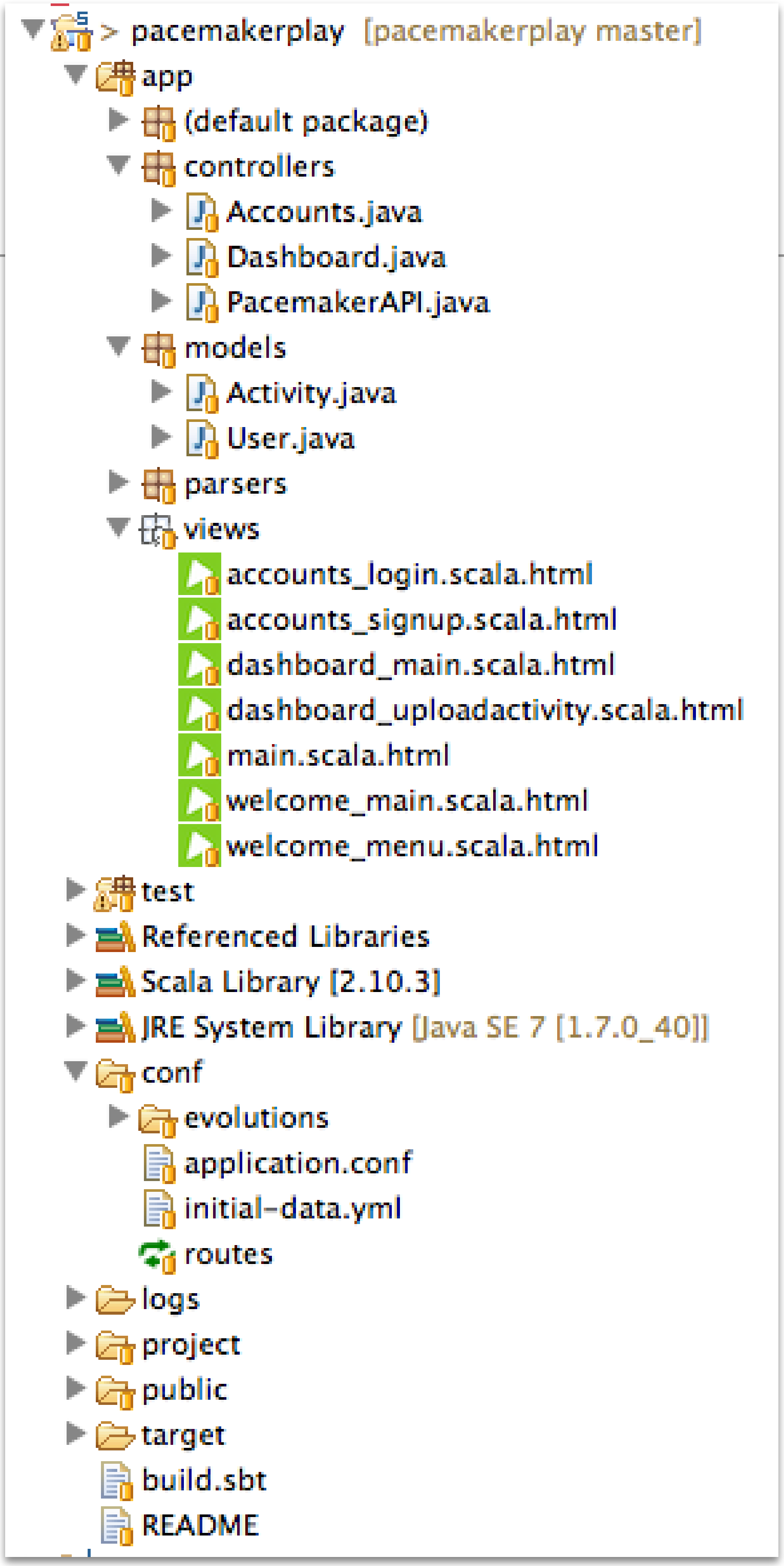


```
◢ > public
    ▷ images
    ▷ javascripts
    ◢ > semantic
        ▷ > components
        ▷ > themes
            semantic.css
            semantic.js
            semantic.min.css
            semantic.min.js
    ▷ stylesheets
```

build.sbt

```
name := "pacemakerplay"

version := "1.0-SNAPSHOT"

libraryDependencies ++= Seq(
  javaJdbc,
  javaEbean,
  cache,
  "net.sf.flexjson" % "flexjson" % "3.1",
  "postgresql" % "postgresql" % "9.1-901-1.jdbc4"
)

play.Project.playJavaSettings
```

pacemakerplay



```
▼ > pacemakerplay [pacemakerplay master]
  ▼ app
    ▶ (default package)
    ▼ controllers
      ▶ Accounts.java
      ▶ Dashboard.java
      ▶ PacemakerAPI.java
    ▼ models
      ▶ Activity.java
      ▶ User.java
    ▶ parsers
    ▼ views
        accounts_login.scala.html
        accounts_signup.scala.html
        dashboard_main.scala.html
        dashboard_uploadactivity.scala.html
        main.scala.html
        welcome_main.scala.html
        welcome_menu.scala.html
  ▶ test
  ▶ Referenced Libraries
  ▶ Scala Library [2.10.3]
  ▶ JRE System Library [Java SE 7 [1.7.0_40]]
  ▼ conf
    ▶ evolutions
      application.conf
      initial-data.yml
      routes
  ▶ logs
  ▶ project
  ▶ public
  ▶ target
    build.sbt
    README
```