Agile Software Development

Produced by

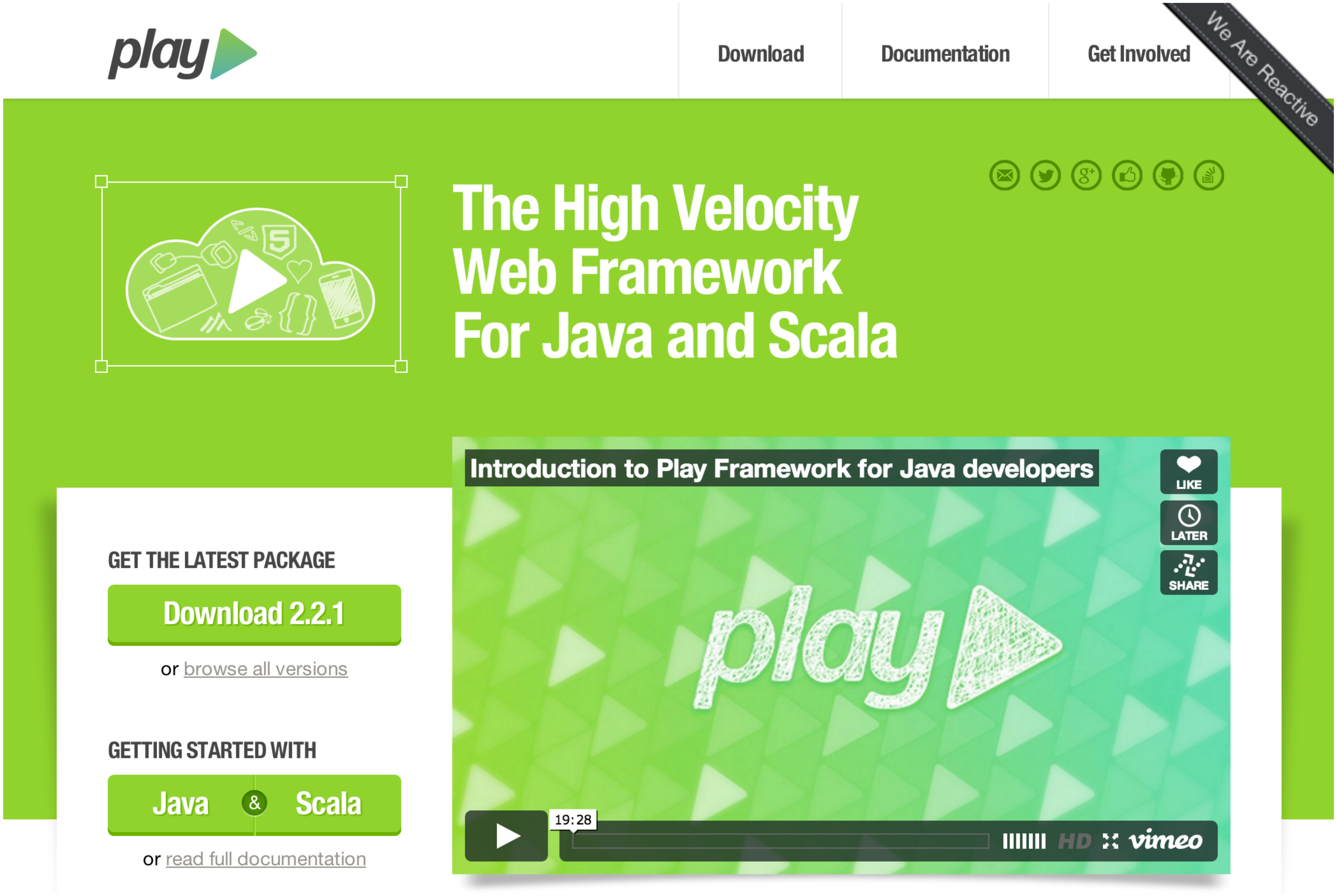
Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Play Framework


Download | Documentation | Get Involved

We Are Reactive

The High Velocity Web Framework For Java and Scala

Introduction to Play Framework for Java developers

LIKE
LATER
SHARE

19:28

HD vimeo

GET THE LATEST PACKAGE

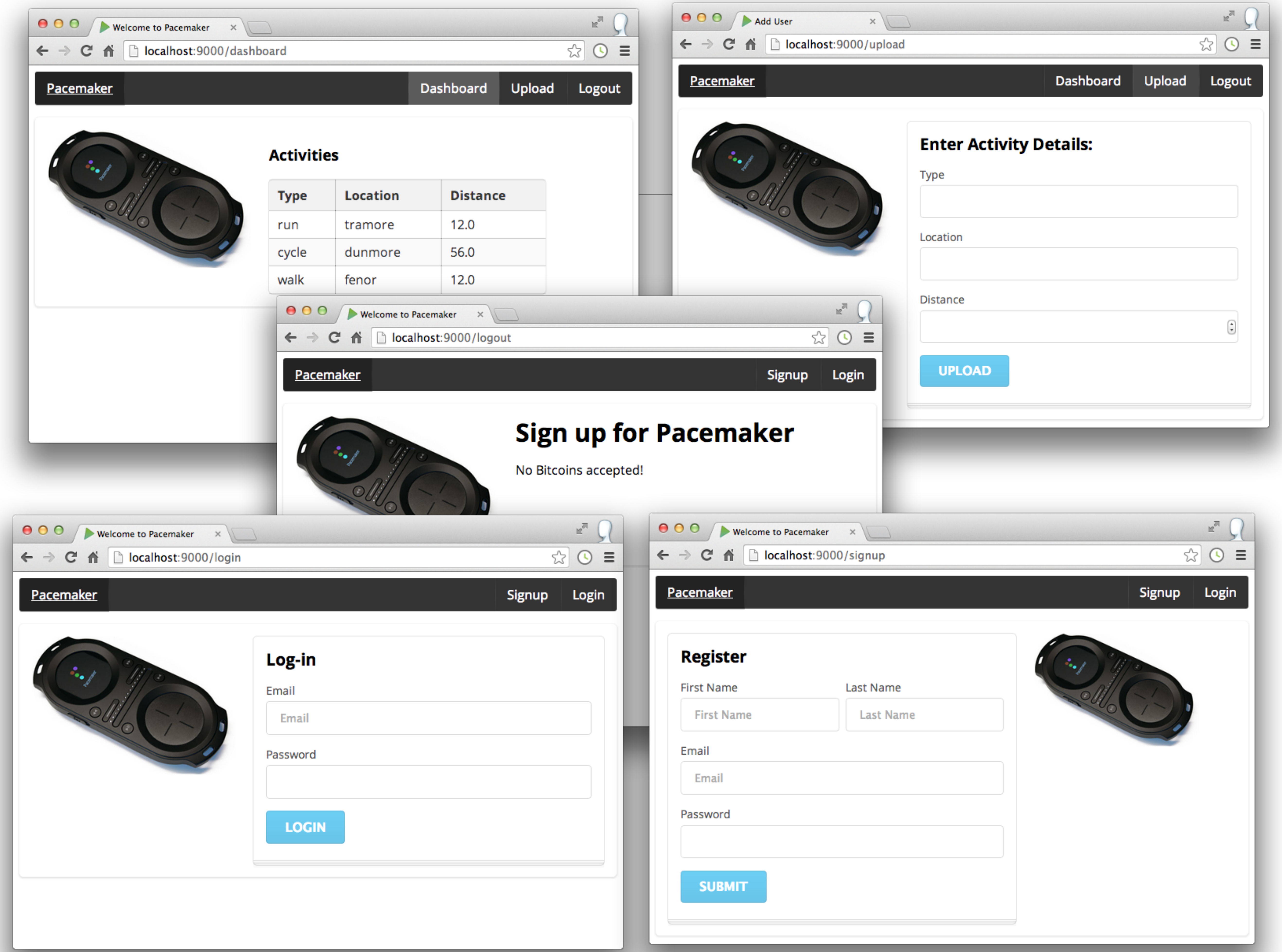
Download 2.2.1

or browse all versions

GETTING STARTED WITH

Java & Scala

or read full documentation



Activities

Type	Location	Distance
run	tramore	12.0
cycle	dunmore	56.0
walk	fenor	12.0

Enter Activity Details:

Type

Location

Distance

Sign up for Pacemaker

No Bitcoins accepted!

Log-in

Email: Email

Password

Register

First Name: First Name

Last Name: Last Name

Email: Email

Password

Pacemaker Play

- Install Play
- User Model
- Routes
- Controllers
- Views

Install Play (1)

- Download and install the latest version of the Play Framework (currently 2.2.1)

http://www.playframework.com

- This will involve simply unzipping the archive, and placing the unzipped folder on the path.

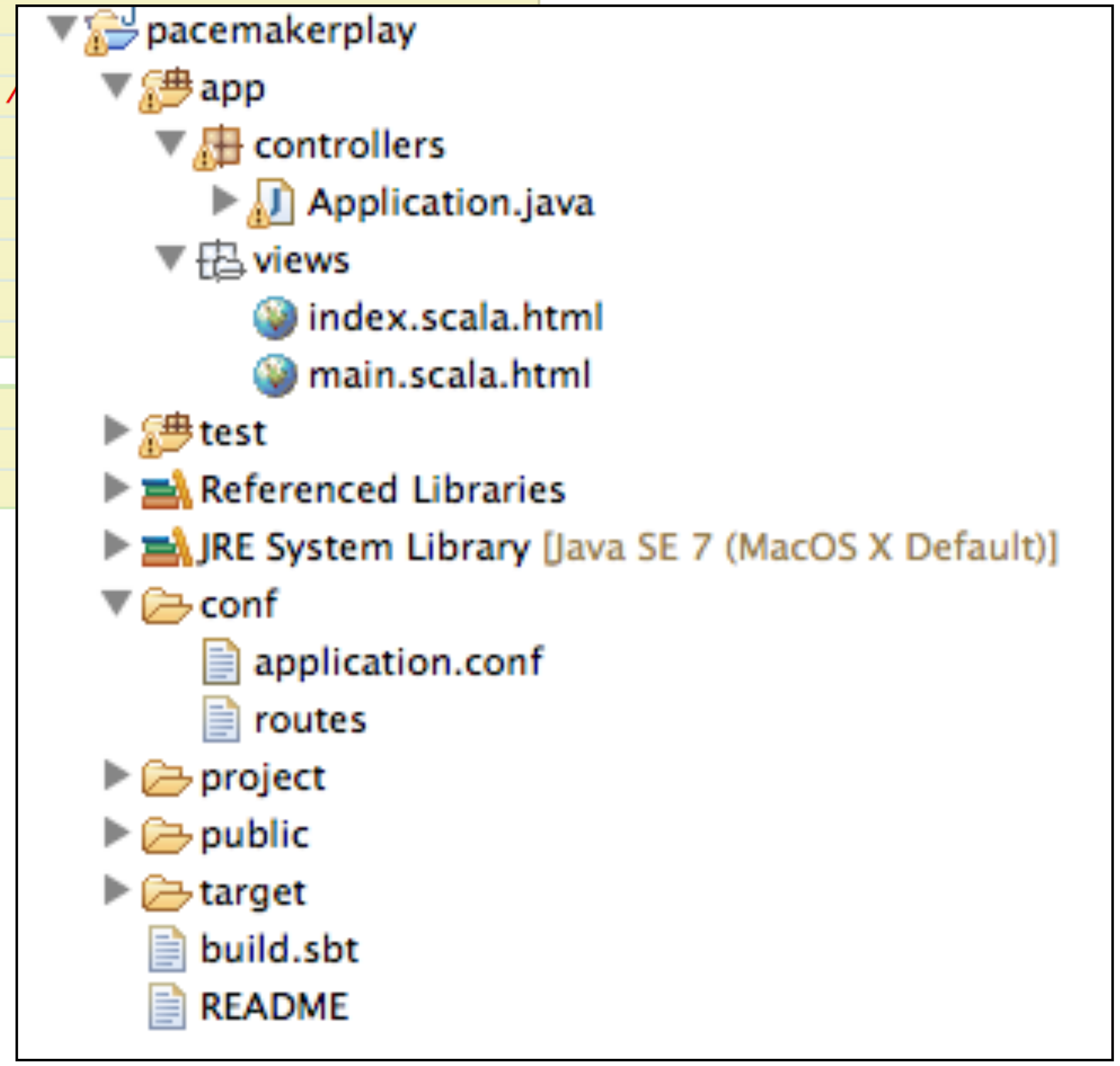
```
play new pacemakerplay
```

```
       _
 _ __ | | __ _ _  _
| '_ \| |/ _' | || |
|  __/|_|\____|\__ /
|_|            |__/

play 2.2.1 built with Scala 2.10.2 (running Java 1.7.0_40), http://www.playframework.com

The new application will be created in /Users/edeleastar/repos/modules/agile/pacemaker/pace
maker-1.0/pacemakerplay

What is the application name? [pacemakerplay]
>

Which template do you want to use for this new application?

  1             - Create a simple Scala application
  2             - Create a simple Java application

> 2
OK, application pacemakerplay is created.

Have fun!
```

Install Play (2)

```
...
       _
 _ __ | | __ _ _  _
| '_ \| |/ _' | || |
|  __/|_|\____|\__ /
|_|            |__/

play 2.2.1 built with Scala 2.10.2 (running Java 1.7.0_40), http:/

> Type "help play" or "license" for more information.
> Type "exit" or use Ctrl+D to leave this console.

[pacemakerplay] $
```

```
eclipse
```



```
▼ pacemakerplay
  ▼ app
    ▼ controllers
      ▶ Application.java
    ▼ views
        index.scala.html
        main.scala.html
  ▶ test
  ▶ Referenced Libraries
  ▶ JRE System Library [Java SE 7 (MacOS X Default)]
  ▼ conf
      application.conf
      routes
  ▶ project
  ▶ public
  ▶ target
    build.sbt
    README
```

Install Play (3)

In the play console, enter

```
run
```

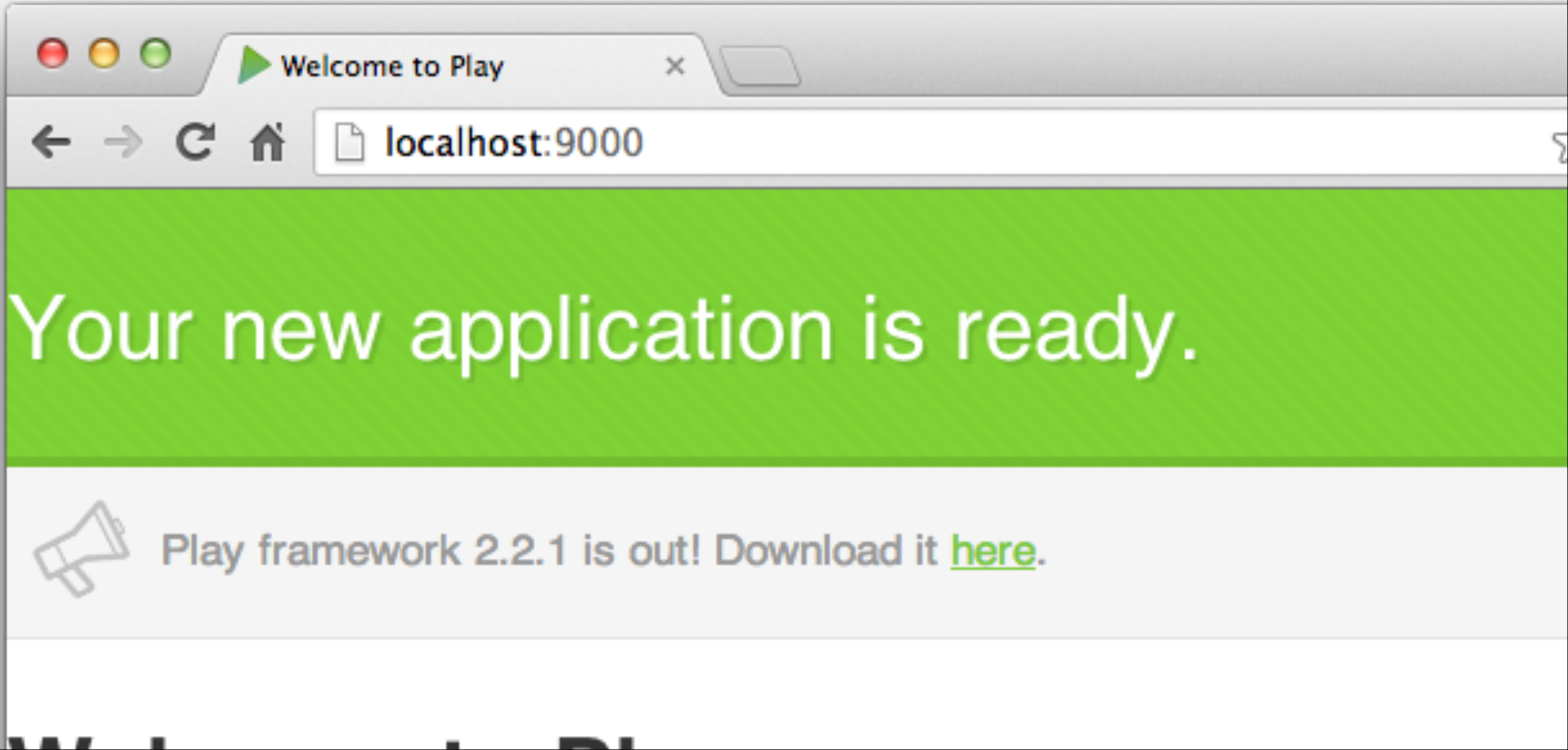
which should display:

```
--- (Running the application from SBT, auto-reloading is enabled) ---

[info] play - Listening for HTTP on /0:0:0:0:0:0:0:0:9000

(Server started, use Ctrl+D to stop and go back to the console...)
```

Browse to :

- http://localhost:9000

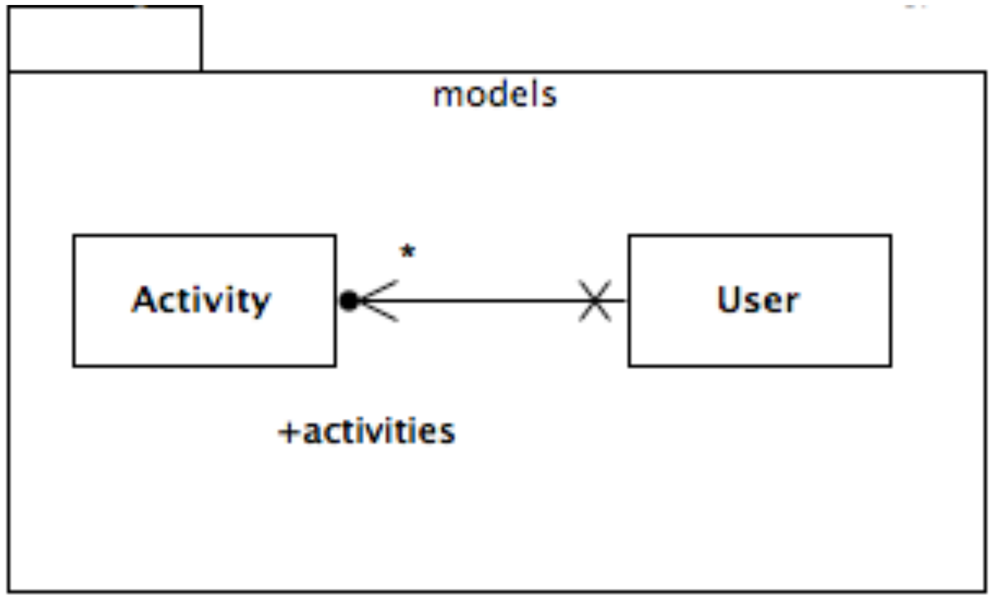
It should display a standard greeting page.



Your new application is ready.

Play framework 2.2.1 is out! Download it here.

Models



```
@Entity
public class User extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String firstname;
  public String lastname;
  public String email;
  public String password;

  @OneToMany(cascade=CascadeType.ALL)
  public List<Activity> activities = new ArrayList<Activity>();

  //…
}
```

```
@Entity
public class Activity extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String kind;
  public String location;
  public double distance;

  //…
}
```

- Uses JPA annotations to manage
  - DB Table generation
  - ID management
  - Relationships to other Models

Models

- Equip Model classes with simple database search and management methods

```
public class User extends Model
{
  //...

  public static User findByEmail(String email)
  {
    return  User.find.where().eq("email", email).findUnique();
  }

  public static User findById(Long id)
  {
    return find.where().eq("id", id).findUnique();
  }

  public static List<User> findAll()
  {
    return find.all();
  }

  public static Model.Finder<String, User> find
    = new Model.Finder<String, User>(String.class, User.class);
}
```

```
public class Activity extends Model
{
  //...

  public static Activity findById(Long id)
  {
    return find.where().eq("id", id).findUnique();
  }

  public static Model.Finder<String, Activity> find
      = new Model.Finder<String, Activity>(String.class, Activity.class);
}
```

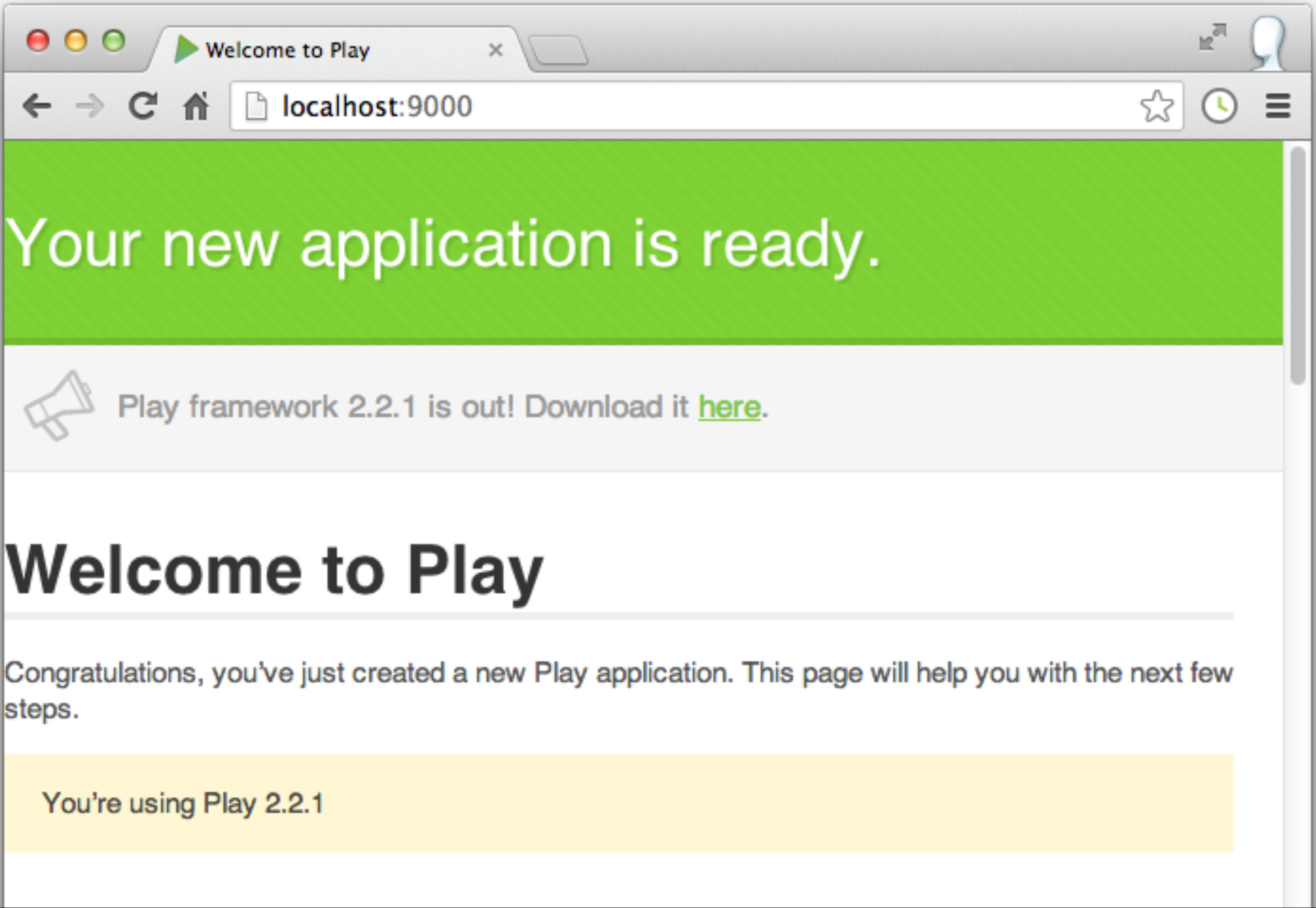
Routes

```
GET     /                                                          controllers.Application.index()
```

- Defines HTTP routes that will be published by this app.
- Route matches http verb + url -> controller.method
- Any browser (or application that can ‘speak’ http) can access the application services through these routes.

```
GET     /                                   controllers.Application.index()
```

```
public class Application extends Controller
{
  public static Result index()
  {
    return ok(index.render("Your new application is ready."));
  }
}
```



Your new application is ready.

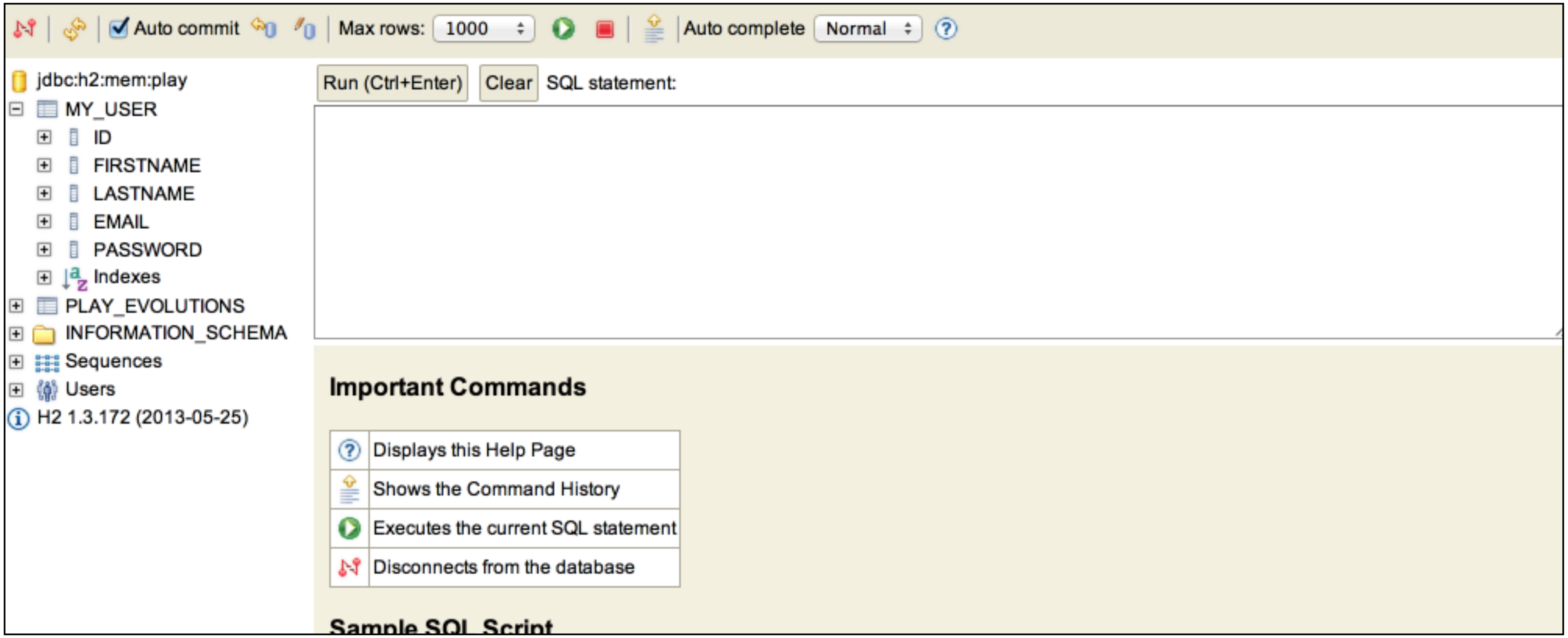
Play framework 2.2.1 is out! Download it here.

Welcome to Play

Congratulations, you've just created a new Play application. This page will help you with the next few steps.

You're using Play 2.2.1

Browse Database



Important Commands

?	Displays this Help Page
	Shows the Command History
▶	Executes the current SQL statement
	Disconnects from the database

Sample SQL Script

- h2 database browser
- Be able to browse tables dynamically

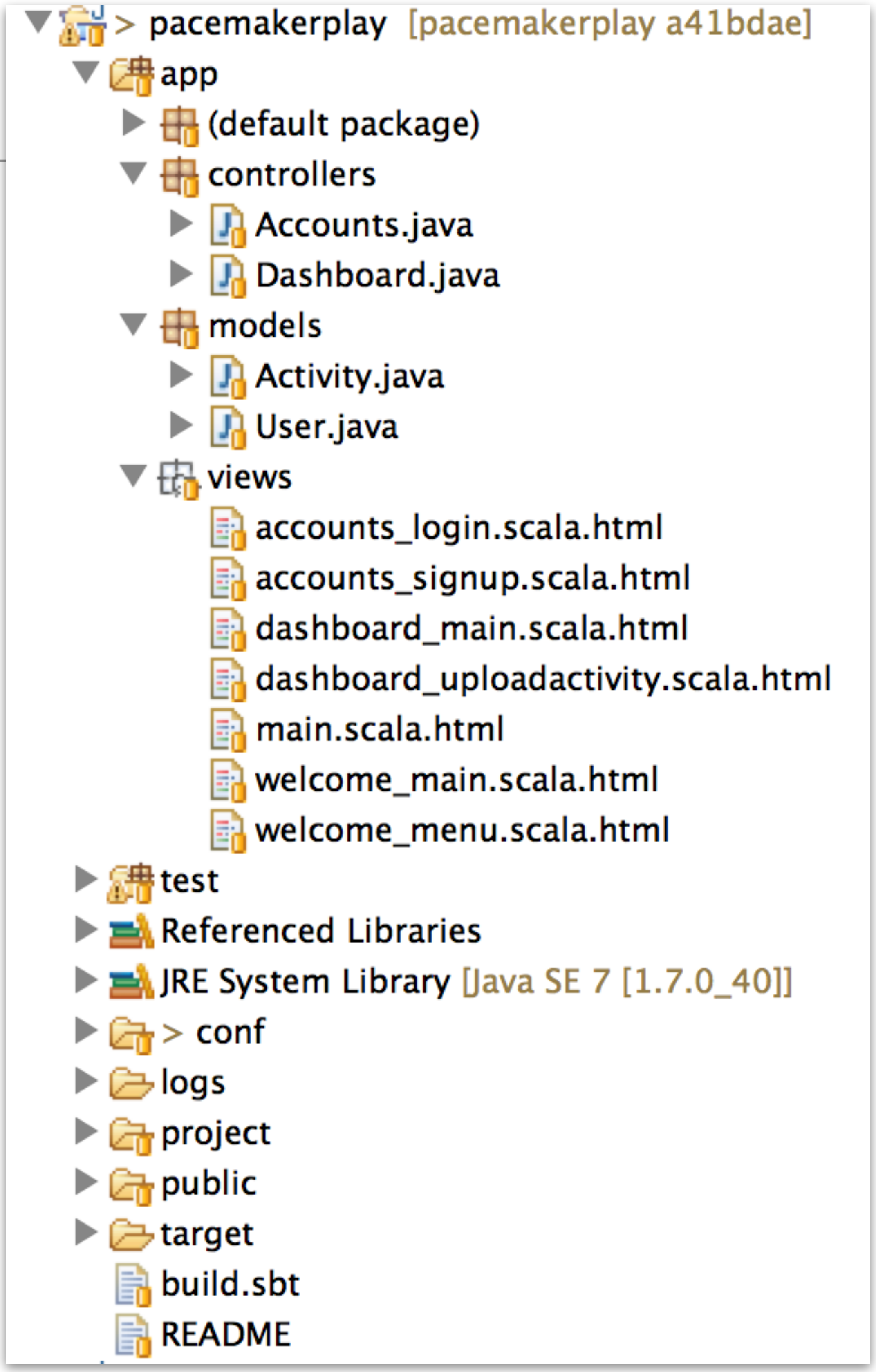
Application Routes

```
# 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 html UX
- Each of these routes appears in views
- Each of these actions generates and returns a complete HTML page

Controllers/Views

- 2 Controllers
  - Accounts
  - Dashboard
- + Set of new views
  - 7 ‘templates’



```
▼ > pacemakerplay [pacemakerplay a41bdae]
  ▼ app
    ▶ (default package)
    ▼ controllers
      ▶ Accounts.java
      ▶ Dashboard.java
    ▼ models
      ▶ Activity.java
      ▶ User.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
  ▶ test
  ▶ Referenced Libraries
  ▶ JRE System Library [Java SE 7 [1.7.0_40]]
  ▶ > conf
  ▶ logs
  ▶ project
  ▶ public
  ▶ target
    build.sbt
    README
```

Accounts Controller

```java
public class Accounts extends Controller
{
  private static final Form<User> userForm = Form.form(User.class);
  private static final Form<User> loginForm = Form.form(User.class);

  public static Result index()
  {
    return ok(welcome_main.render());
  }

  public static Result signup()
  {
    return ok(accounts_signup.render());
  }

  public static Result login()
  {
    return ok(accounts_login.render());
  }

  public static Result logout()
  {
    session().clear();
    return ok(welcome_main.render());
  }
  //...
}
```

Accounts Controller

```java
public class Accounts extends Controller
{
  //...

  public static Result register()
  {
    Form<User> boundForm = userForm.bindFromRequest();
    if(loginForm.hasErrors())
    {
      return badRequest(accounts_login.render());
    }
    else
    {
      User user = boundForm.get();
      Logger.info ("User = " + user.toString());
      user.save();
      return ok(welcome_main.render());
    }
  }

  public static Result authenticate()
  {
    Form<User> boundForm = loginForm.bindFromRequest();
    if(loginForm.hasErrors())
    {
      return badRequest(accounts_login.render());
    }
    else
    {
       User user = User.findByEmail(boundForm.get().email);
       if (user != null && user.password.equals(boundForm.get().password))
       {
         session("email", boundForm.get().email);
         return redirect(routes.Dashboard.index());
       }
    }
    return redirect(routes.Accounts.index());
  }
}
```

Dashboard

```
public class Dashboard extends Controller
{
  private static final Form<Activity> activityForm = Form.form(Activity.class);

  public static Result index()
  {
    String email = session().get("email");
    User user = User.findByEmail(email);
    return ok(dashboard_main.render(user.activities));
  }

  public static Result uploadActivityForm()
  {
    return ok(dashboard_uploadactivity.render());
  }

  public static Result submitActivity()
  {
    Form<Activity> boundForm = activityForm.bindFromRequest();
    Activity activity = boundForm.get();

    if(activityForm.hasErrors())
    {
      return badRequest();
    }
    String email = session().get("email");
    User user = User.findByEmail(email);

    user.activities.add(activity);
    user.save();
    return redirect (routes.Dashboard.index());
  }
}
```

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit