

# Agile Software Development

---

Produced  
by

Eamonn de Leastar (edelestar@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 LÁIRCE




# Pacemaker Play UX

---

Welcome to Pacemaker x  
localhost:9000/dashboard

Pacemaker Dashboard Upload Logout




### Activities

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

Add User x  
localhost:9000/upload

Pacemaker Dashboard Upload Logout



### Enter Activity Details:

Type


Location

Distance

**UPLOAD**

Welcome to Pacemaker x  
localhost:9000/logout

Pacemaker Signup Login




## Sign up for Pacemaker

No Bitcoins accepted!

Welcome to Pacemaker x  
localhost:9000/login

Pacemaker Signup Login



### Log-in


Email

Password

**LOGIN**

Welcome to Pacemaker x  
localhost:9000/signup

Pacemaker Signup Login



### Register

First Name

Last Name

Email

Password

**SUBMIT**

# Outsource UI Styling

- Use class = "ui .." to attach to various styles

The screenshot shows the Semantic UI website. The browser address bar displays "semantic-ui.com". The top navigation bar includes "Content" and "Introduction" with navigation arrows. The main content area has a teal background with the title "Semantic UI" and version "0.15.1". Below the title, it says "UI is the vocabulary of the web." and "Semantic empowers designers and developers by creating a language for sharing UI." There are two buttons: "VIEW UI" and "DOWNLOAD".

**Download**

**Introduction**

- Definitions
- Overview
- Types
- Variations
- What's Different

**Project**

- Contributing
- Local Docs

**Elements**

- Button
- Divider
- Header
- Icon
- Image
- Input
- Label
- Loader
- Progress
- Reveal
- Segment
- Step

**Collections**

- Breadcrumb
- Form
- Grid
- Menu

**VIEW UI**

**DOWNLOAD**

**Lose the Hieroglyphics**

Semantic is structured around natural language con development more intuitive.

Semantic is tag agnostic meaning you can use [any html](#) elements.

**SEMANTIC** **BOOTSTRAP**

```
<main class="ui three column grid">
```

```
<div class=
```

```
<div clas
```

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

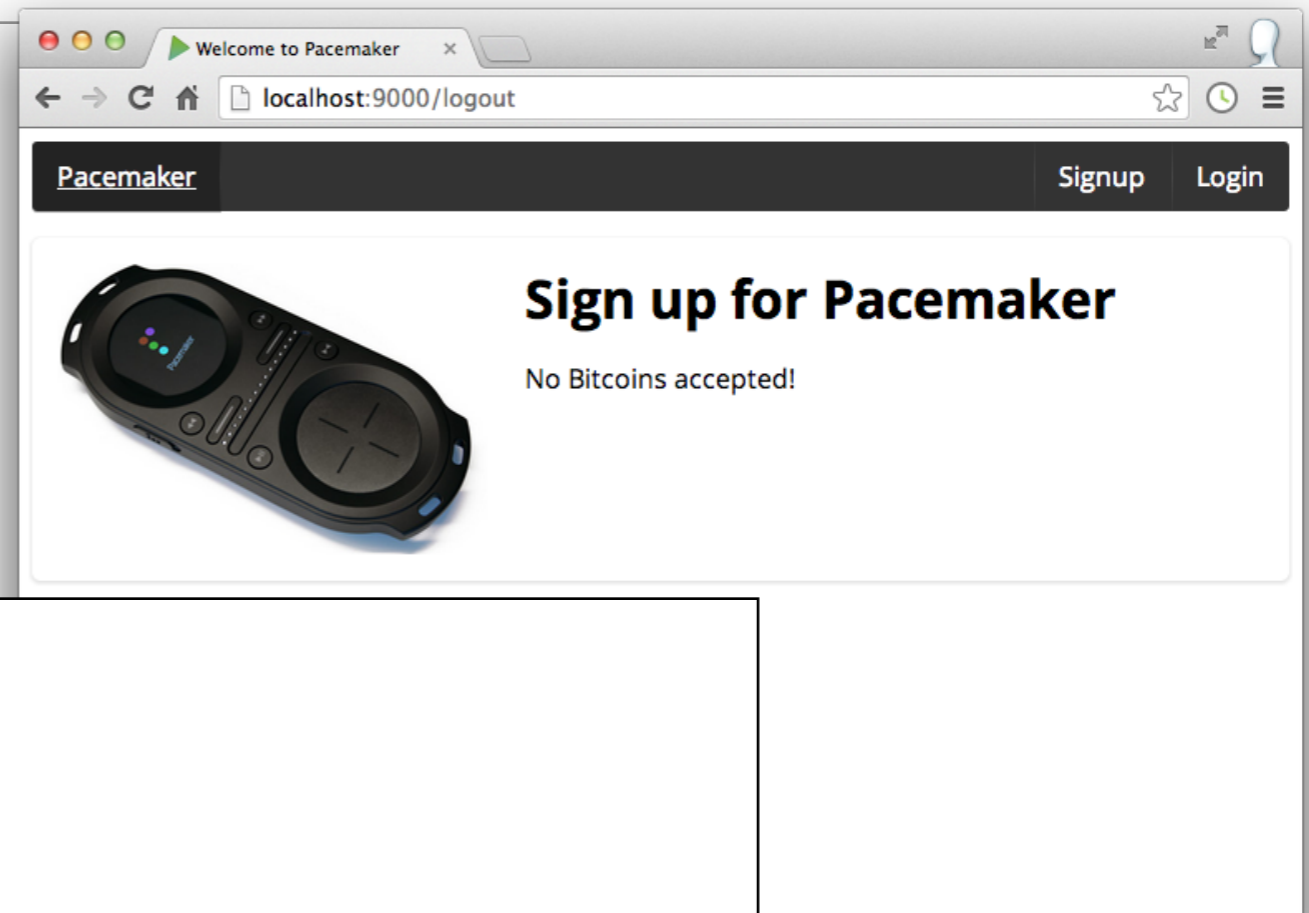
# Welcome

routes

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

## Accounts.java

```
public class Accounts extends Controller
{
    public static Result index()
    {
        return ok(welcome_main.render());
    }
    ...
}
```



```
@()
@main("Welcome to Pacemaker") {
    @welcome_menu()

    <section class="ui raised segment">
        <div class="ui grid">
            <aside class="six wide column">
                
            </aside>
            <article class="ten wide column">
                <h1 class="ui header"> Sign up for Pacemaker </h1>
                <p> No Bitcoins accepted! </p>
            </article>
        </div>
    </section>
}
```

welcome\_main.scala.html

# templating

- Entire view is inserted into **@content** section of page

```
@(title: String)(content: Html)
<!DOCTYPE html>
<html>
  <head>
    <title>@title</title>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0">

    <link rel="stylesheet" type="text/css" href="@routes.Assets.at("semantic/css/semantic.min.css")">
    <link rel="stylesheet" type="text/css" href="@routes.Assets.at("stylesheets/main.css")">
    <link href='http://fonts.googleapis.com/css?family=Source+Sans+Pro:400,700|Open+Sans:300,400,600,700' rel="stylesheet">
    <link rel="shortcut icon" type="image/png" href="@routes.Assets.at("images/favicon.png")">

    <script src="@routes.Assets.at("javascripts/jquery-2.0.3.min.js")"></script>
    <script src="@routes.Assets.at("semantic/javascript/semantic.min.js")"></script>
  </head>

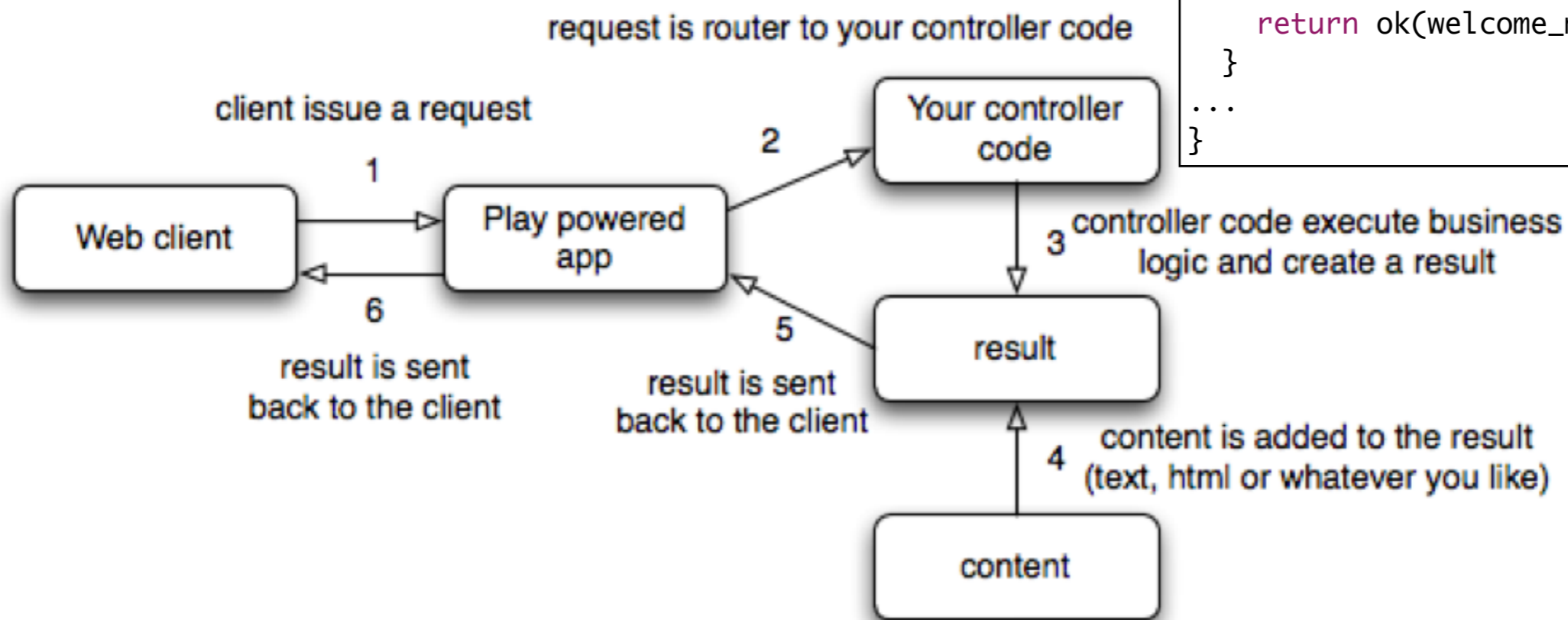
  <body>
    @content
  </body>
</html>
```

```
@()
@main("Welcome to Pacemaker") {
  @welcome_menu()

  <section class="ui raised segment">
    <div class="ui grid">
      <aside class="six wide column">
        
      </aside>
      <article class="ten wide column">
        <h1 class="ui header"> Sign up for Pacemaker </h1>
        <p> No Bitcoins accepted! </p>
      </article>
    </div>
  </section>
}
```

**@main** implies that main.scala.html will define the structure of the generated page





```

public class Accounts extends Controller
{
    public static Result index()
    {
        return ok(welcome_main.render());
    }
    ...
}
  
```

```

@()
@main("Welcome to Pacemaker") {
    @welcome_menu()

    <section class="ui raised segment">
        <div class="ui grid">
            <aside class="six wide column">
                
            </aside>
            <article class="ten wide column">
                <h1 class="ui header"> Sign up for Pacemaker </h1>
                <p> No Bitcoins accepted! </p>
            </article>
        </div>
    </section>
}
  
```



# Includes

```
@()  
  
@main("Welcome to Pacemaker") {  
  @welcome_menu()  
  
  <section class="ui raised segment">  
    <div class="ui grid">
```

```
@()  
  
<nav class="ui inverted menu">  
  <div class="header item"> <a href="/"> Pacemaker </a> </div>  
  <div class="right menu">  
    <a class="item" href="/signup"> Signup</a>  
    <a class="item" href="/login"> Login</a>  
  </div>  
</nav>
```

```
images/pacemaker.jpg)" class="ui medium image">
```

```
for Pacemaker </h1>
```

welcome\_menu.scala.html

```
</section>  
}
```

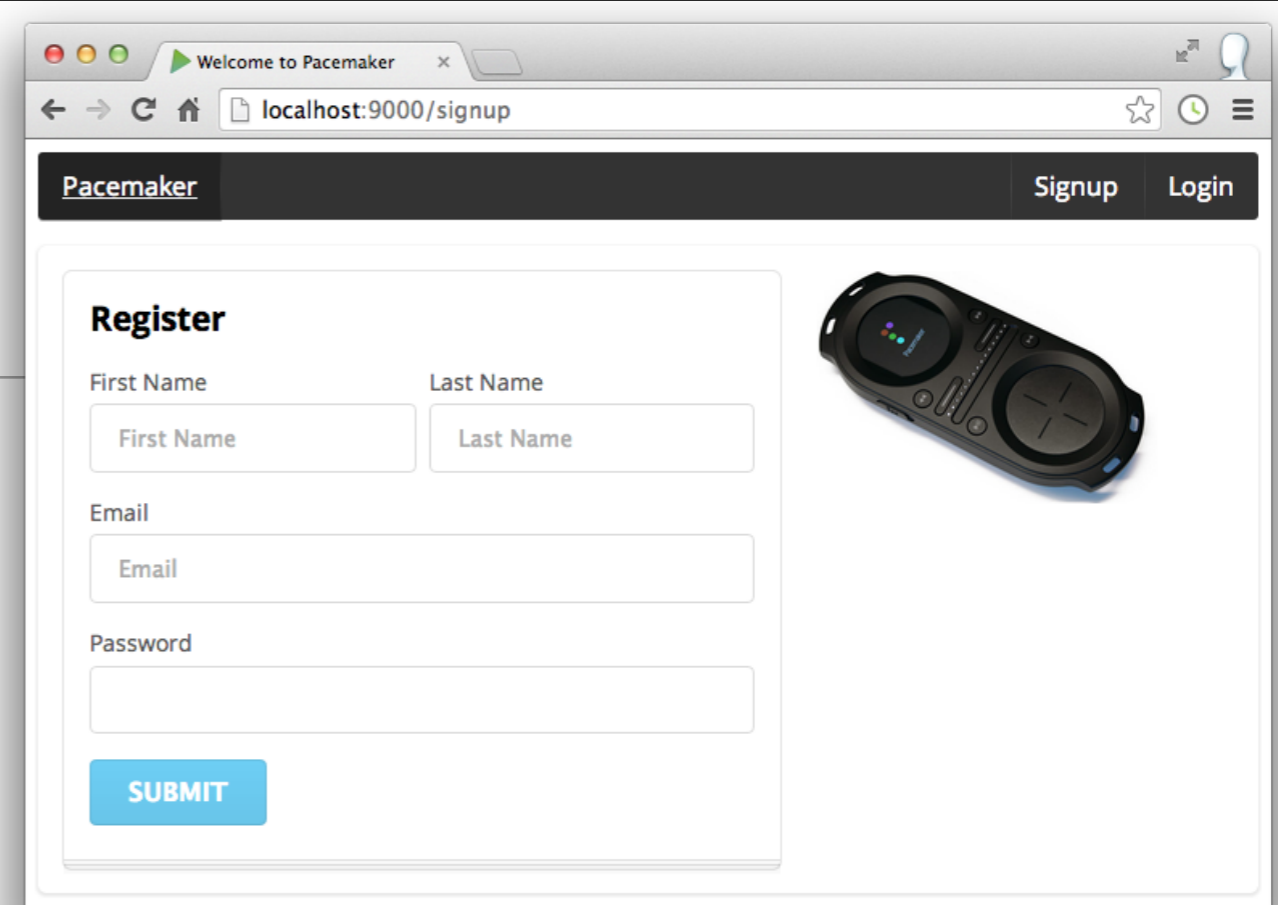
Pacemaker

Signup

Login

# Signup

```
@()  
  
@main("Welcome to Pacemaker") {  
  @welcome_menu()  
  
  <section class="ui raised segment">  
    <div class="ui grid">  
      <div class="ui ten wide column">  
        <div class="ui stacked fluid form segment">  
          <form action="/register" method="POST">  
            <h3 class="ui header">Register</h3>  
            <div class="two fields">  
              <div class="field">  
                <label>First Name</label>  
                <input placeholder="First Name" type="text" name="firstname">  
              </div>  
              <div class="field">  
                <label>Last Name</label>  
                <input placeholder="Last Name" type="text" name="lastname">  
              </div>  
            </div>  
            <div class="field">  
              <label>Email</label>  
              <input placeholder="Email" type="text" name="email">  
            </div>  
            <div class="field">  
              <label>Password</label>  
              <input type="password" name="password">  
            </div>  
            <button class="ui blue submit button">Submit</button>  
          </form>  
        </div>  
      </div>  
      <aside class="ui five wide column">  
          
      </aside>  
    </div>  
  </section>  
}
```



GET /signup controllers.Accounts.signup()

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

# Signup

GET

/signup

controllers.Accounts.signup()

```
<form action="/register" method="POST">
  <h3 class="ui header">Register</h3>
  <div class="two fields">
    <div class="field">
      <label>First Name</label>
      <input placeholder="First Name" type="text" name="firstname" >
    </div>
    <div class="field">
      <label>Last Name</label>
      <input placeholder="Last Name" type="text" name="lastname" >
    </div>
  </div>
  <div class="field">
    <label>Email</label>
    <input placeholder="Email" type="text" name="email" >
  </div>
  <div class="field">
    <label>Password</label>
    <input type="password" name="password" >
  </div>
  <button class="ui blue submit button">Submit</button>
</form>
```

accounts\_signup.scala.html

The screenshot shows a web browser window with the title 'Welcome to Pacemaker' and the URL 'localhost:9000/signup'. The page content includes a 'Pacemaker' header and a 'Register' form. The form has four input fields: 'First Name', 'Last Name', 'Email', and 'Password'. The 'First Name' and 'Last Name' fields are side-by-side. Below them is the 'Email' field, and then the 'Password' field. A blue 'SUBMIT' button is at the bottom of the form. A DJ controller is visible on the right side of the browser window.

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

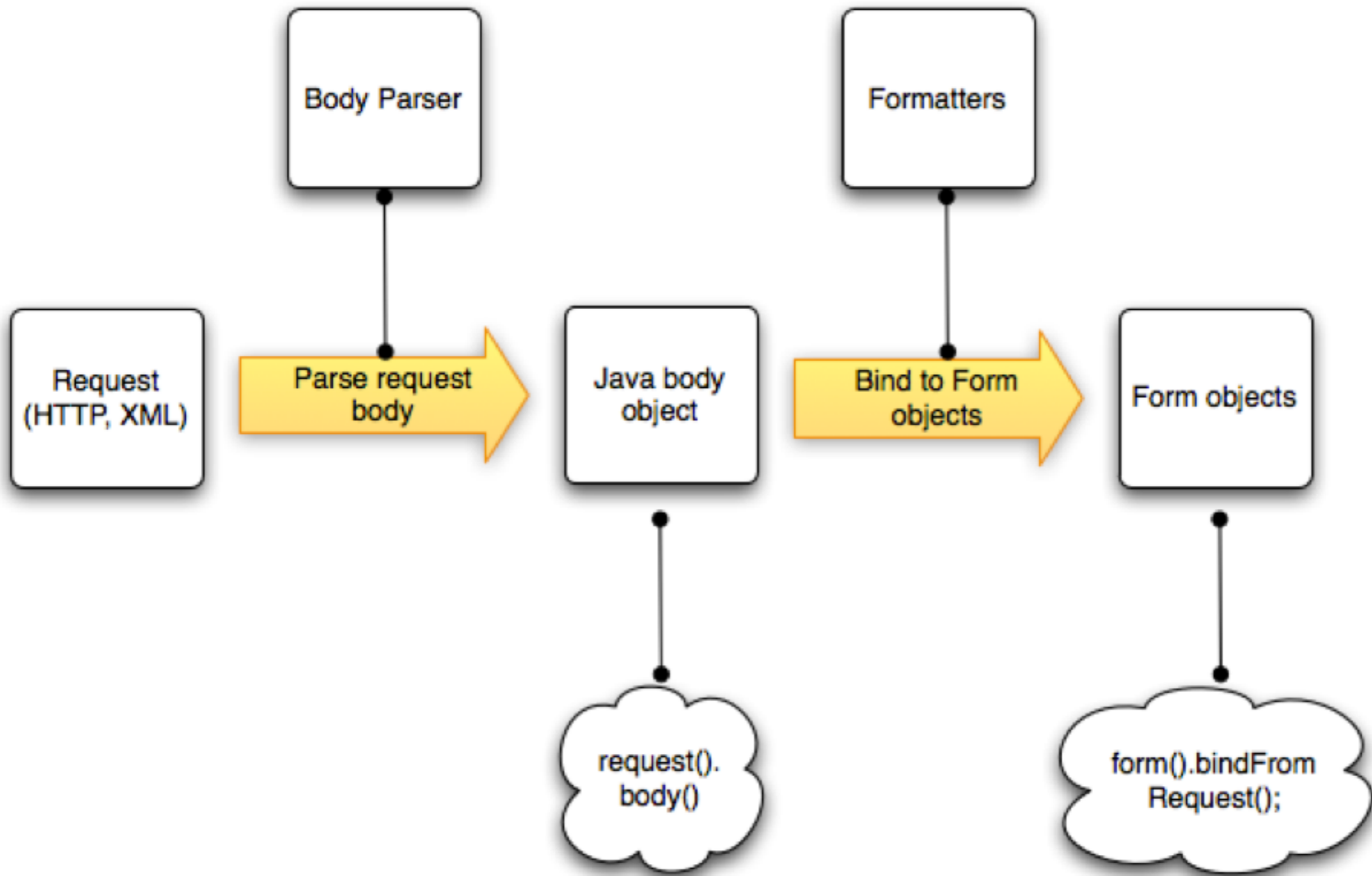
# Signup

---

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

    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());
        }
    }
}

//...
```



# Signup Form Processing

---

- Recover named form input items from request
- Extract these elements into a Java object

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

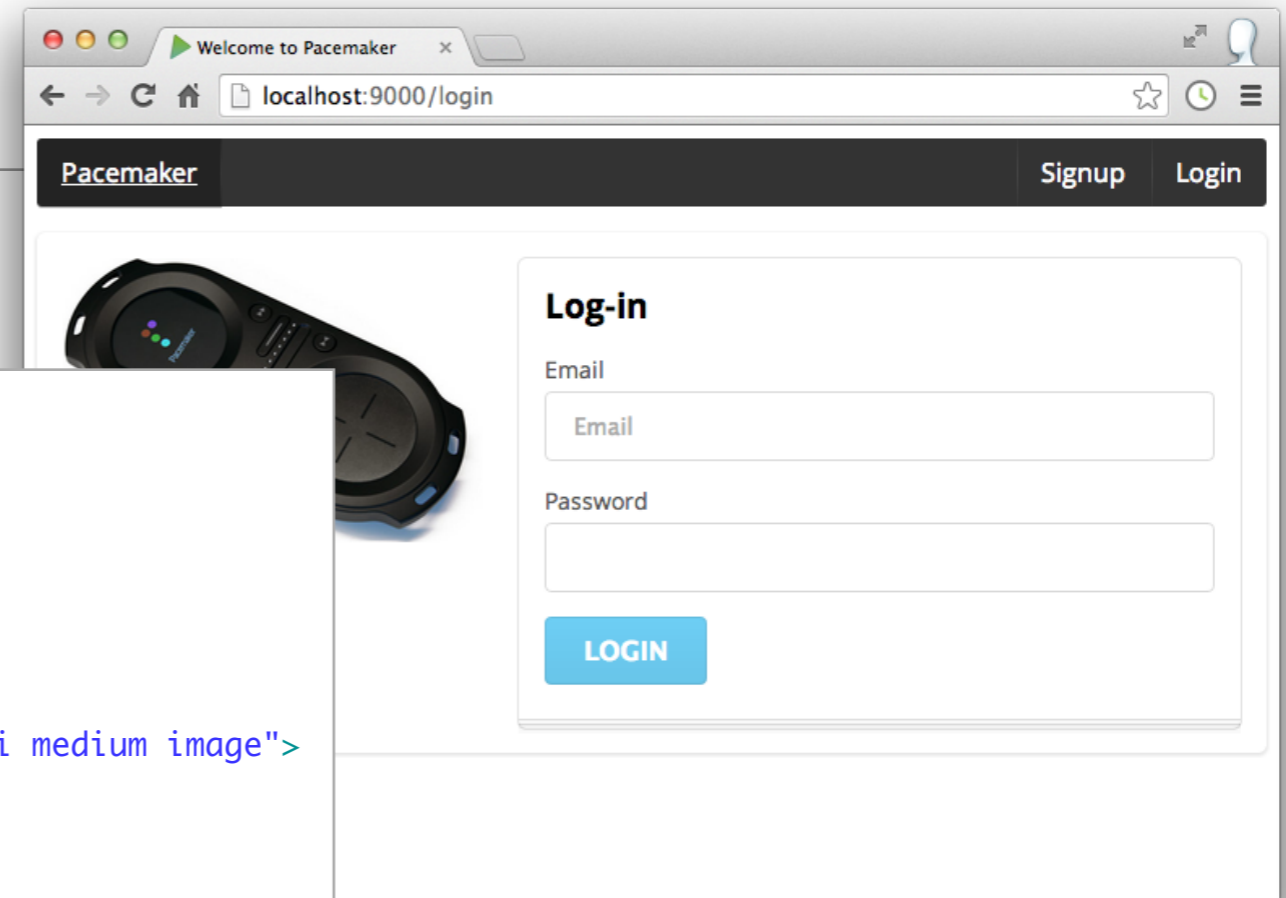
    public static Result register()
    {
        Form<User> boundForm = userForm.bindFromRequest();
        if(LoginForm.hasErrors())
        {
            return badRequest(accounts_login.render());
        }
        else
        {
            User user = boundForm.get();
            Logger.info ("User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

GET

/login

controllers.Accounts.login()

# Login



```
@()  
  
@main("Welcome to Pacemaker") {  
  @welcome_menu()  
  
  <section class="ui raised segment">  
    <div class="ui grid">  
      <aside class="ui six wide column">  
          
      </aside>  
      <div class="ui ten wide column fluid form">  
        <div class="ui stacked segment">  
          <form action="/authenticate" method="POST">  
            <h3 class="ui header">Log-in</h3>  
            <div class="field">  
              <label>Email</label>  
              <input placeholder="Email" type="text" name="email">  
            </div>  
            <div class="field">  
              <label>Password</label>  
              <input type="password" name="password">  
            </div>  
            <button class="ui blue submit button">Login</button>  
          </form>  
        </div>  
      </div>  
    </div>  
  </section>  
}
```

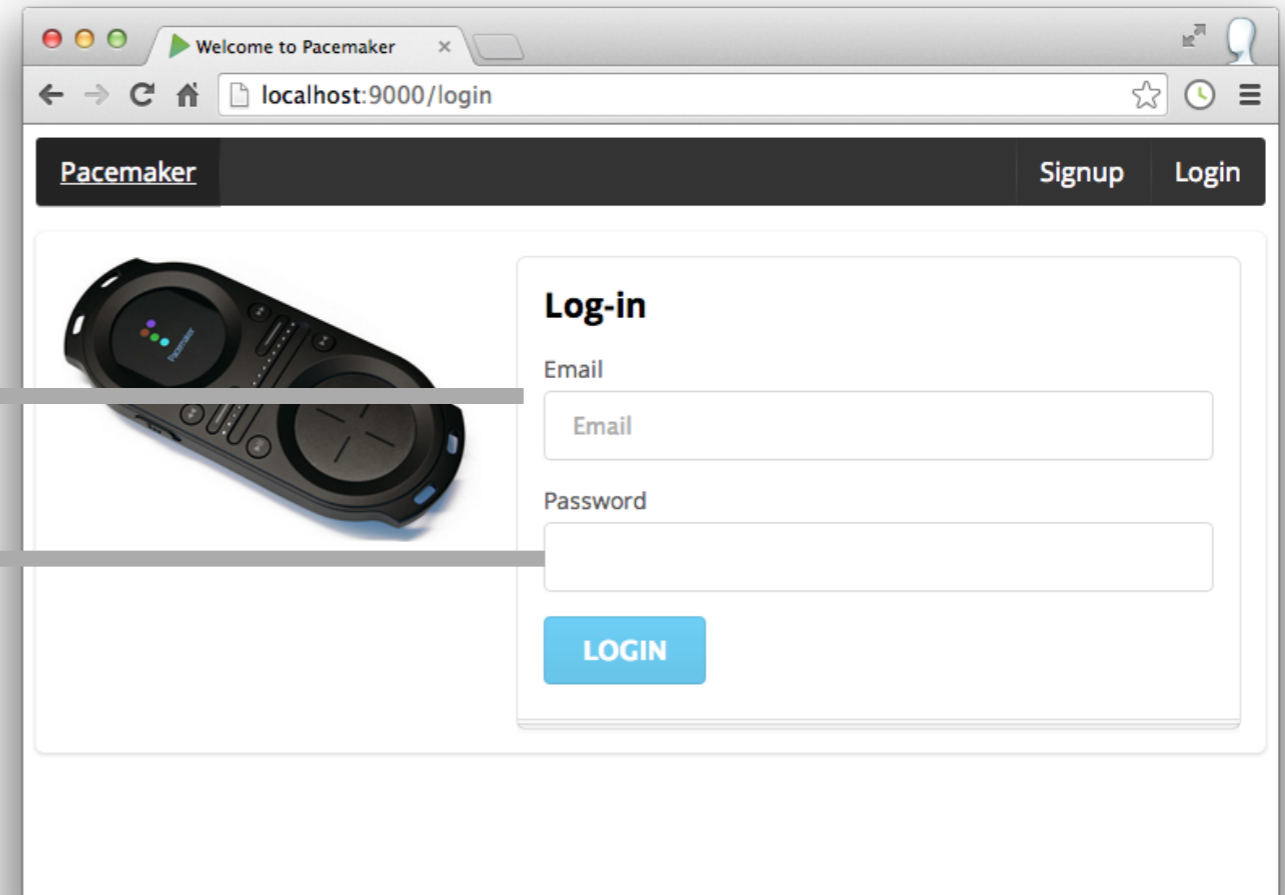
```
public static Result login()  
{  
  return ok(accounts_login.render());  
}
```



# Login

---

```
form action="/authenticate" method="POST">
  <h3 class="ui header">Log-in</h3>
  <div class="field">
    <label>Email</label>
    <input placeholder="Email" type="text" name="email">
  </div>
  <div class="field">
    <label>Password</label>
    <input type="password" name="password">
  </div>
  <button class="ui blue submit button">Login</button>
</form>
```



# Sessions - login

- A globally accessible data structure into which we put details of 'current' user.
- Read back this in other controllers to determine appropriate content

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

    public static Result authenticate()
    {
        Form<User> boundForm = loginForm.bindFromRequest();
        if(loginForm.hasErrors())
        {
            return badRequest(accounts_login.render());
        }
        else
        {
            session("email", boundForm.get().email);
            return redirect(routes.Dashboard.index());
        }
    }
    //...
}
```

- Not checking if user is valid!
- Should compare password/email with database, and only allow in of valid user credential presented

# Sessions - Logout

---

- Destroy the session
- Redirect to Welcome page

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

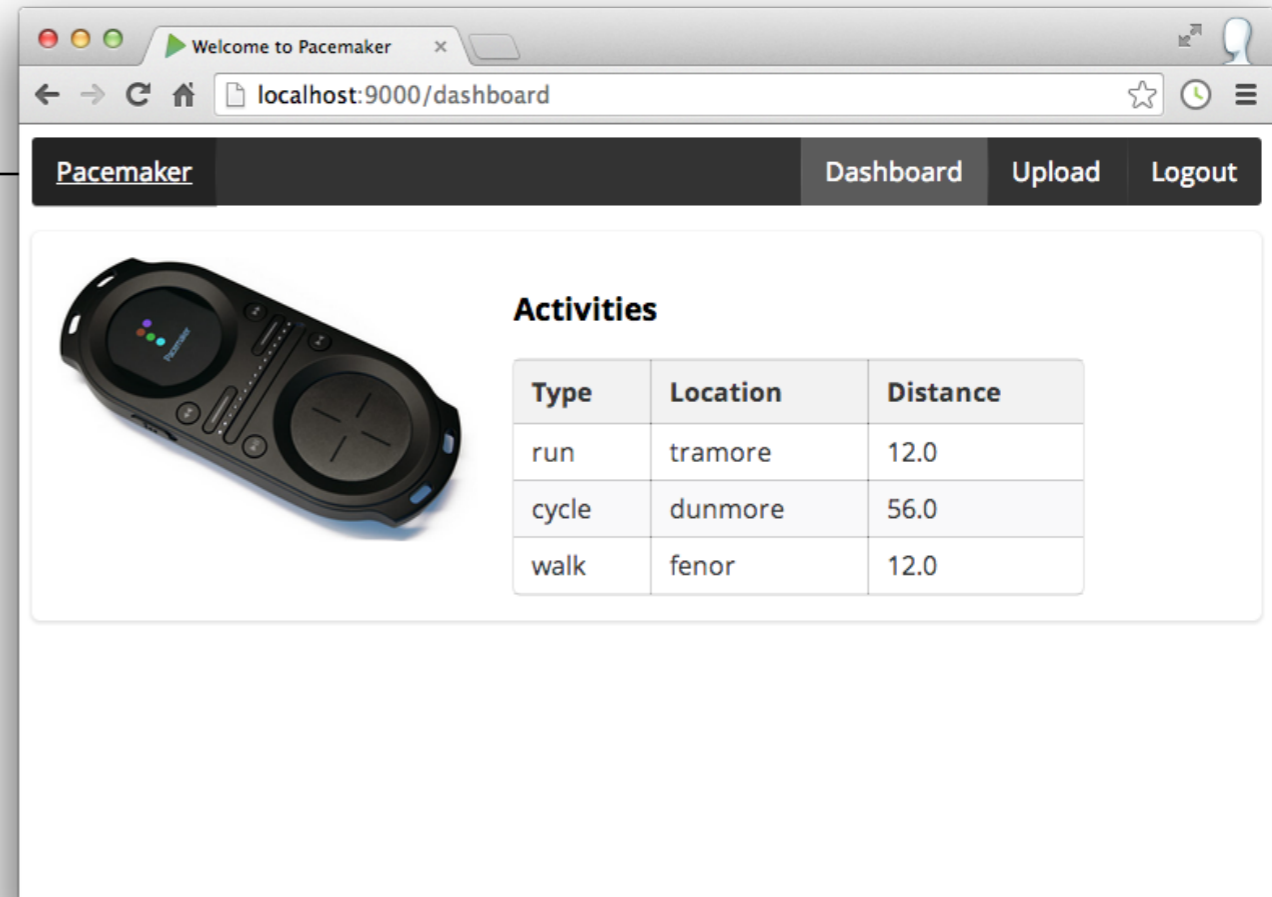
# Dashboard

```
@(activities: List[Activity])

@main("Welcome to Pacemaker") {

  <nav class="ui inverted menu">
    <div class="header item"> <a href="/"> Pacemaker </a> </div>
    <div class="right menu">
      <a class="active item" href="/dashboard"> Dashboard</a>
      <a class="item" href="/upload"> Upload</a>
      <a class="item" href="/logout"> Logout</a>
    </div>
  </nav>

  <section class="ui raised segment">
    <div class="ui grid">
      <aside class="six wide column">
        
      </aside>
      <article class="eight wide column">
        <h3> <class="ui header"> Activities </h3>
        <table class="ui celled table segment">
          <thead>
            <tr>
              <th>Type</th>
              <th>Location</th>
              <th>Distance</th>
            </tr>
          </thead>
          <tbody>
            @for(i <- 0 until activities.size) {
              <tr>
                <td> @activities(i).kind </td> <td> @activities(i).location </td> <td> @activities(i).distance </td>
              </tr>
            }
          </tbody>
        </table>
      </article>
    </div>
  </section>
}
```



GET /dashboard controllers.Dashboard.index()

# Dashboard

GET /dashboard  
GET /upload  
POST /submitactivity

controllers.Dashboard.index()  
controllers.Dashboard.uploadActivityForm()  
controllers.Dashboard.submitActivity()

```
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());
    }
}
```

# Dashboard

## Activities

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

- Activities list sent to view
- Scala for loop to iterate over this list, and present in a table

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

```
<table class="ui celled table segment">
  <thead>
    <tr>
      <th>Type</th>
      <th>Location</th>
      <th>Distance</th>
    </tr>
  </thead>
  <tbody>
    @for(i <- 0 until activities.size) {
      <tr>
        <td> @activities(i).kind </td> <td> @activities(i).location </td> <td> @activities(i).distance </td>
      </tr>
    }
  </tbody>
</table>
```

# Upload Activity

```
<form action="/submitactivity" method="POST">
  <h3 class="ui header">Enter Activity Details: </h3>
  <div class="field">
    <label>Type</label>
    <input type="text" name="kind">
  </div>
  <div class="field">
    <label>Location</label>
    <input type="text" name="location">
  </div>
  <div class="field">
    <label>Distance</label>
    <input type="number" name="distance">
  </div>
  <button class="ui blue submit button"> Upload </button>
</form>
```

**Enter Activity Details:**

Type

Location

Distance

**UPLOAD**

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

  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());
  }
}
```



# Upload Activity

---

Acquire the Activity object

Ask the session who is 'logged in'

Add the new Activity to this users  
activities list

Save the updates

Back to dashboard

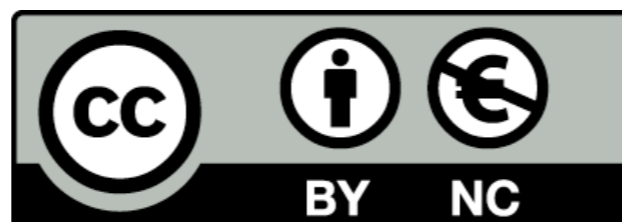
```
public static Result submitActivity()
{
    Form<Activity> boundForm = activityForm.bindFromRequest();
    Activity activity = boundForm.get();
    if(activityForm.hasErrors())
    {
        return badRequest();
    }

    String email = session().get("email");
    User user = User.findByEmail(email);

    user.activities.add(activity);

    user.save();

    return redirect(routes.Dashboard.index());
}
```



Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>



Waterford Institute of Technology  
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRCE

