Web 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

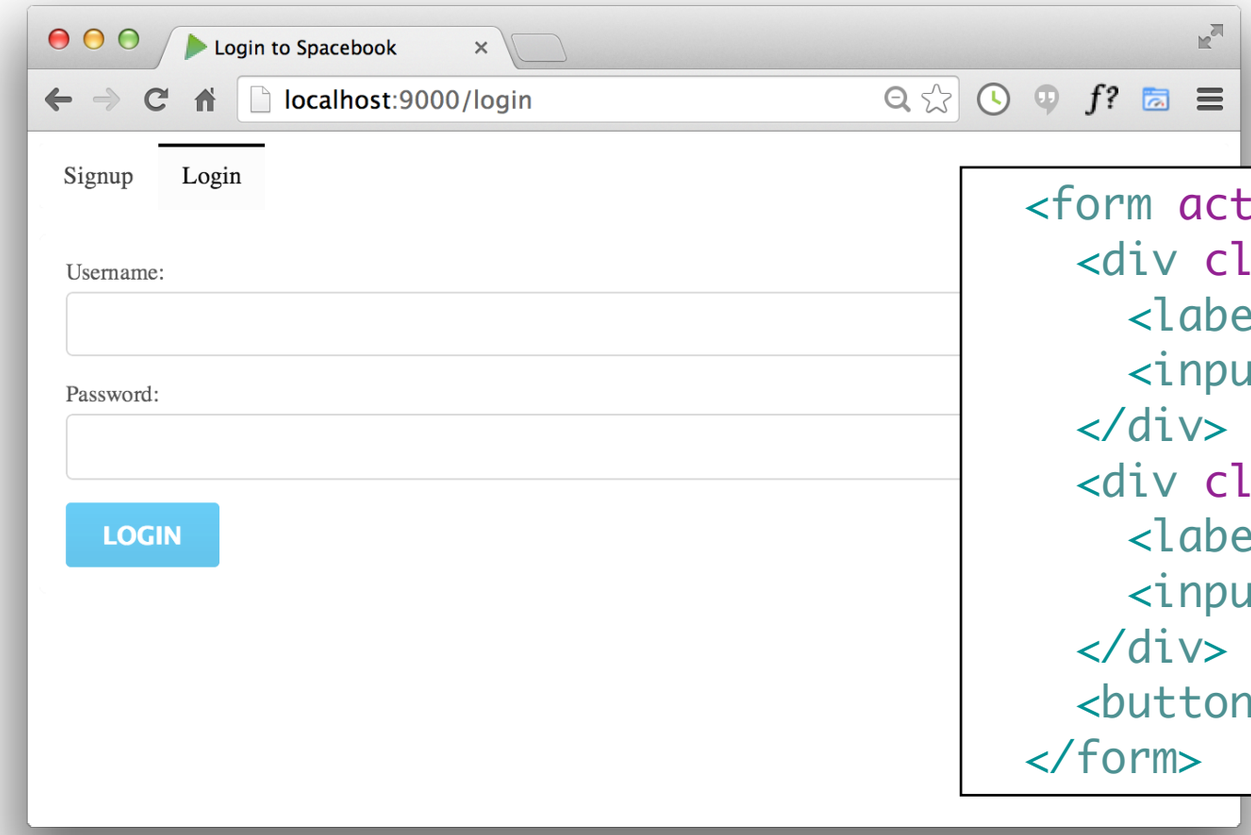
Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Sessions in Play

Web Development

Login




```
<form action="/authenticate" method="POST">
  <div class="field">
    <label> Username: </label>
    <input 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>
```

```
POST    /authenticate                                   Accounts.authenticate
```

```
public static void authenticate(String email, String password)
 {
   Home.index();
 }
```

Authenticate Action

```
public static void authenticate(String email, String password)
{
  ...
}
```

- Need to decide whether to allow a user to log in (they must register first), and subsequently 'remember' which user has logged in.
  - In the authenticate method, see if the given user is registered or not.
  - If they are registered, place the user 'id' into a 'session' object
  - This session object will be available to other controllers during subsequent page visits.

Extend User Class ....

2 new methods:

Search for a User object matching a specific email

Check if a given objects password matches a specific password.

```
public class User extends Model
{
  public String firstName;
  public String lastName;
  public String email;
  public String password;

  public User(String firstName, String lastName,
              String email,     String password)
  {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.password = password;
  }

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

  public boolean checkPassword(String password)
  {
    return this.password.equals(password);
  }

}
```

Authenticate Action

```
public static void authenticate(String email, String password)
{
  Logger.info("Attempting to authenticate with " + email + ":" +  password);

  User user = User.findByEmail(email);
  if ((user != null) && (user.checkPassword(password) == true))
  {
    Logger.info("Authentication successful");
    session.put("logged_in_userid", user.id);
    Home.index();
  }
  else
  {
    Logger.info("Authentication failed");
    login();
  }
}
```

- user.id
  - Although the class User does not explicitly have a field called ‘id’, because User is a ‘model’ class - and id field is always generated.
  - This is unique - and we will use it widely in the application.

Authenticate

Search for matching user

If one is found, see if password matches

if they match, store user ‘id’ in ‘session’

Let user in to home page

if not, revert to start page

```
public static void authenticate(String email, String password)
{
  Logger.info("Attempting to authenticate with " + email + ":" +

  User user = User.findByEmail(email);

  if ((user != null) && (user.checkPassword(password) == true))
  {
    Logger.info("Authentication successful");
    session.put("logged_in_userid", user.id);

    Home.index();
  }
  else
  {
    Logger.info("Authentication failed");
    login();
  }
```

Sessions

- Every time a user make a ‘request’ - i.e.
    - presses a link
    - navigates to a new page
    - submits a form
- The ‘action’ has no idea who the user is each time such a request arrives
- Remember - there may be hundreds or thousands of requests, from different users, arriving concurrently.

Session Objects

- A mechanism whereby our program can ‘know’ who the ‘current’ user is.
- Implemented by a complex process involving ‘cookies’, ip address, + various other techniques.
- Simplified for the programer in Play as follows:

- If we ‘know’ who the user is, then we store the id in the ‘session’ object:

```
session.put("logged_in_userid", user.id);
```

- Later, in another action, if we want to find out who the is, we ask the session object:

```
String userId = session.get("logged_in_userid");
User user = User.findById(Long.parseLong(userId));
String name = user.firstName;
```

Session - put and set

- put into the session object the user.id value at the key ‘logged_in_userid’

```
session.put("logged_in_userid", user.id);
```

- Ask the session for the value corresponding to the key ‘logged_in_userid’

```
String userId = session.get("logged_in_userid");
```

- Use that value to look up the database for a corresponding user object

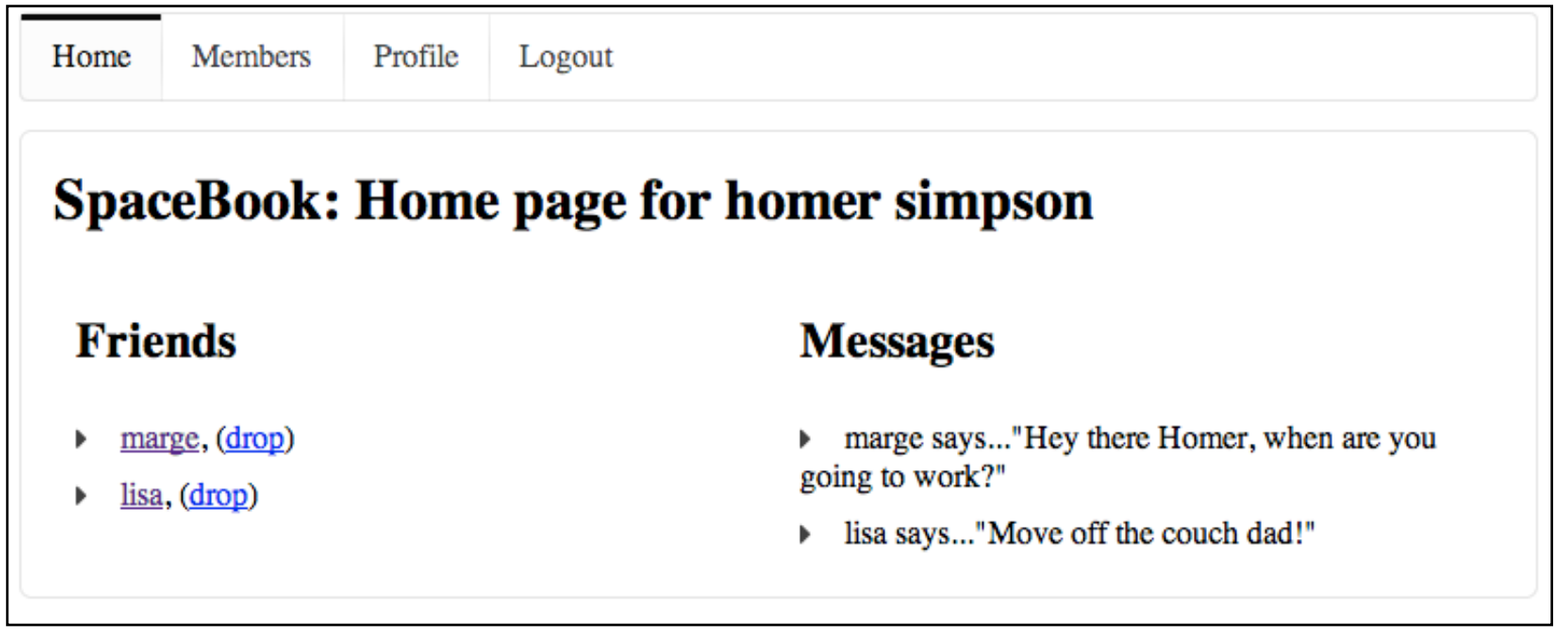
```
User user = User.findById(Long.parseLong(userId));
```

- Get the name of the user from the user object

```
String name = user.firstName;
```

Home Page Heading

- Once a user is successfully logged in, we would like to display the user name in the title of some of the pages.
- Currently ‘hard coded’ to “Homer Simpson”




```
...
 <h2 class="ui header">SpaceBook: Home page for homer simpson </h2>
...
```

views/Home/index.html

```
public class Home extends Controller
{
  public static void index()
  {
    render();
  }
```

controllers/Home.java

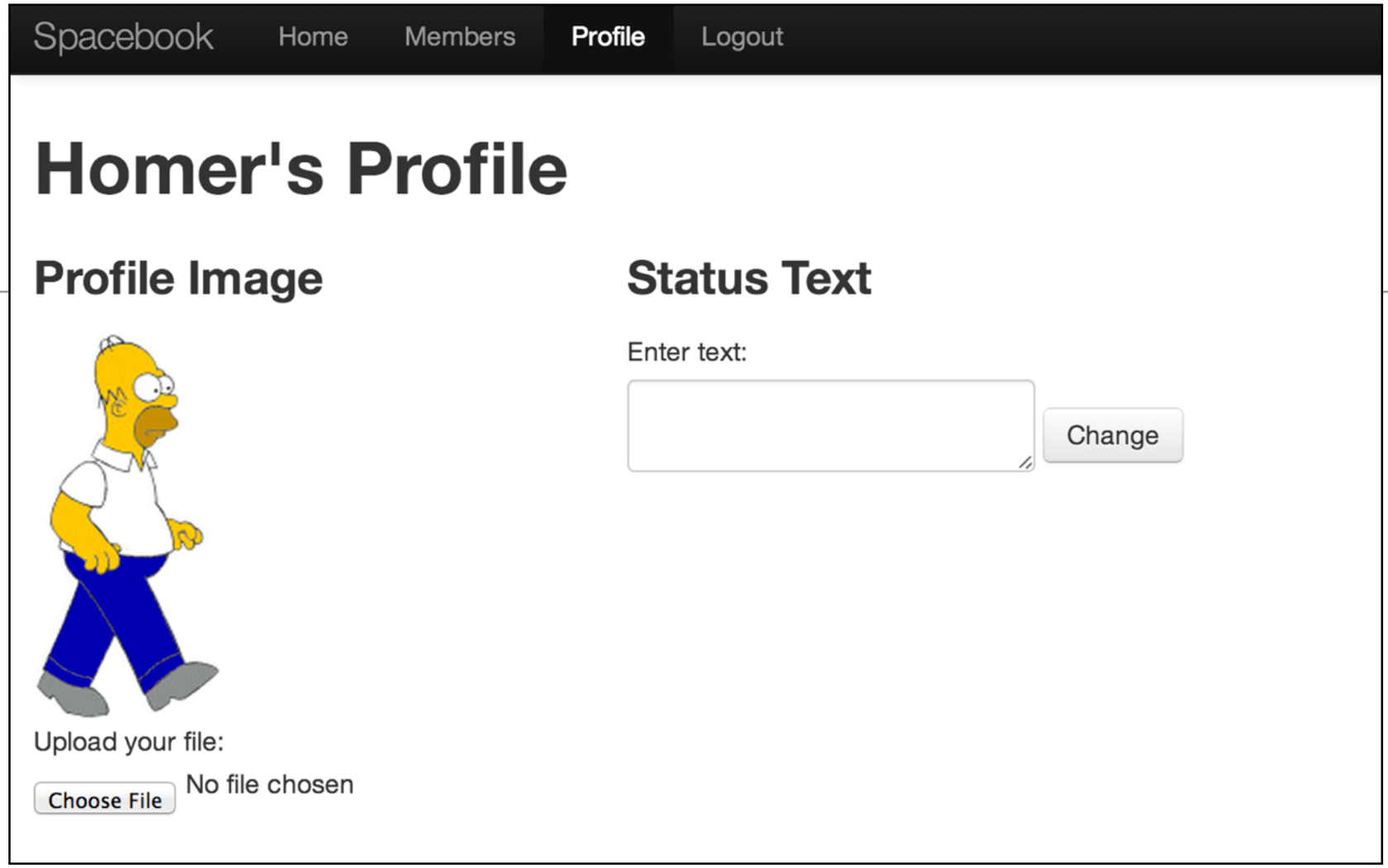
controllers/Home.java

```
public static void index()
{
  String userId = session.get("logged_in_userid");
  User user = User.findById(Long.parseLong(userId));
  render(user);
}
```

views/Home/index.html

```
<h2 class="ui header">SpaceBook: Home page for ${user.firstName} ${user.lastName}</h2>
```

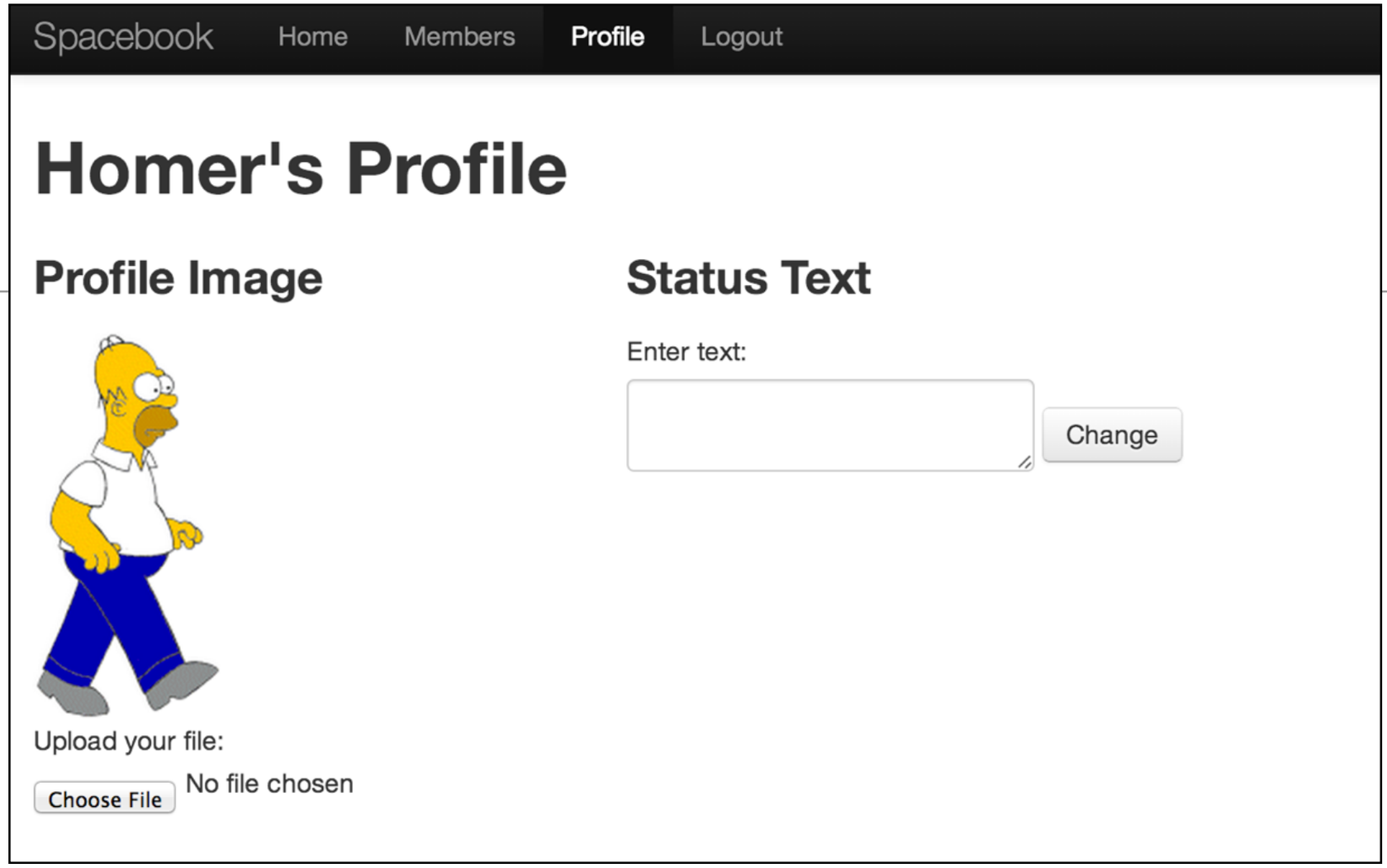
- Assuming the user is ‘logged in’,
  - retrieve the user identity from the session
  - look up the database of users to get the user object
  - get the user name
  - pass the name to the view



Home
Profile Title
(hardcoded)

```
<h1>Homers's Profile</h1>
```

```
public static void index()
{
  render();
}
```



Home
Profile Title
(Dynamic)

```
<h1>${user.firstName} 's Profile</h1>
```

```
public static void index()
{
  String userId = session.get("logged_in_userid");
  User user = User.findById(Long.parseLong(userId));
  render(user);
}
```

Destroy the Session

- In the corresponding action, delete the session

```
public static void logout()
{
  session.clear();
  index();
}
```

- Any attempts to recover the information from the session object will fail