Application Development & Modelling

BSc in Applied Computing

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

Assignment 2 Solution

Story 1

- Lab09 introduced a 'data.yml' file in the conf directory containing initial user data loaded at startup
- Extend this data to include the full homer clan (and friends). Note, the number of spaces (4) before each field is significant.

```
User(homer):
    firstName: Homer
    lastName: Simpson
    email: homer@simpson.com
    password: secret

User(marge):
    firstName: marge
    lastName: Simpson
    email: marge@simpson.com
    password: secret
```

Story 1 Solution

- Database records loaded at startup Bootstrap class

```
@OnApplicationStart
public class Bootstrap extends Job
{
  public void doJob()
  {
    Fixtures.deleteDatabase();
    Fixtures.loadModels("data.yml");
  }
}
```

```
User(lisa):
    firstName: Lisa
    lastName: Simpson
    email: lisa@simpson.com
    password: secret

User(marge):
    firstName: Marge
    lastName: Simpson
    email: marge@simpson.com
    password: secret

User(homer):
    firstName: Homer
    lastName: Simpson
    email: homer@simpson.com
    password: secret

User(bart):
    firstName: Bart
    lastName: Simpson
    email: bart@simpson.com
    password: secret

User(maggie):
    firstName: Maggie
    lastName: Simpson
    email: maggie@simpson.com
    password: secret
```

Story 2

- Extend the User Model to include the following new fields:
  - Age
  - Nationality
- These fields must be filled in when a user registers.

Story 2 Solution - View

- view/Accounts/signup.html

```html
<form action="/register" method="POST">
  <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="two fields">
    <div class="field">
      <label>Nationality</label>
      <input placeholder="Nationality" type="text" name="nationality">
    </div>
    <div class="field">
      <label>Age</label>
      <input placeholder="Age" type="text" name="age">
    </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>
```

Story 2 Solution - Model

```
@Entity
public class User extends Model
{
...
  public int    age;
  public String nationality;
;
...
  public User(String firstName, String lastName,
              int    age,       String nationality,
              String email,     String password)
  {
    this.firstName = firstName;
    this.lastName = lastName;
    this.age = age;
    this.nationality = nationality;
    this.email = email;
    this.password = password;
  }
}
```

Story 2 Solution - Controller

```
public static void register(String firstName, String lastName,
                            int    age,       String nationality,
                            String email,     String password,
                                              String password2)
{
  Logger.info(firstName + " " + lastName + " " + email + " " + password);
  User user = new User(firstName, lastName, age, nationality, email, password);
  user.save();
  index();
}
```

Story 3

- For the new fields accepted in Story 2, display them on the users Home Profile page.
- In addition, on the users 'Public' profile (then one a friend can see), display just the 'Nationality' field

Story 3 Solution

- views/Profile/index.html

```
<table class="ui table segment">
   <thead>
     <tr>
       <th>Email</th>
       <th>Nationality</th>
       <th>Age</th>
     </tr>
   </thead>
   <tbody>
     <tr>
       <td>${user.email}</td>
       <td>${user.nationality}</td>
       <td>${user.age}</td>
     </tr>
   </tbody>
 </table>
```

- views/PublicProfile/visit.html

```
<table class="ui table segment">
  <thead>
    <tr>
      <th>Email</th>
      <th>Nationality</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>${user.email}</td>
      <td>${user.nationality}</td>
    </tr>
  </tbody>
</table>
```

Story 4

Even if a user seems to have logged out, we can still access the spacebook pages by entering the url of some of the controllers. For instance, log in, then log our immediately and try these links:

- http://localhost:9000/home
- http://localhost:9000/profile

How could you prevent this?

HINT: The session object has a method called 'clear()'. If we call this (say in logout action), then it will remove any things we have put in there. For every action, we could first check to make sure a valid ID is in the session, otherwise redirect to the start page.

Story 4 Solution - Accounts

- Introduce useful method to return the currently logged in user.
- If no user logged in - then display the login screen.

```
public class Accounts extends Controller
{
  ...
  public static User getLoggedInUser()
  {
    User user = null;
    if (session.contains("logged_in_userid"))
    {
      String userId = session.get("logged_in_userid");
      user = User.findById(Long.parseLong(userId));
    }
    else
    {
      login();
    }
    return user;
  }

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

Story 5 Solution

- getLoggedInUser method in Accounts can be invoked in each controller index method.
- This is redirect to login page if no user logged in

```
public class Accounts extends Controller
{
  ...
  public static User getLoggedInUser()
  {
    User user = null;
    if (session.contains("logged_in_userid"))
    {
      String userId = session.get("logged_in_userid");
      user = User.findById(Long.parseLong(userId));
    }
    else
    {
      login();
    }
    return user;
  }
...
}
```

```
public class EditProfile extends Controller
{
  public static void index()
  {
    User user = Accounts.getLoggedInUser();
    render(user);
  }
}
```

```
public class Home extends Controller
{
  public static void index()
  {
    User user = Accounts.getLoggedInUser();
    render(user);
  }
```

```
public class Members extends Controller
{
  public static void index()
  {
    User user = Accounts.getLoggedInUser();
    List<User> users = User.findAll();
    users.remove(user);
    render(users);
  }
```

```
public class Profile extends Controller
{
  public static void index()
  {
    User user = Accounts.getLoggedInUser();
    render(user);
  }
```

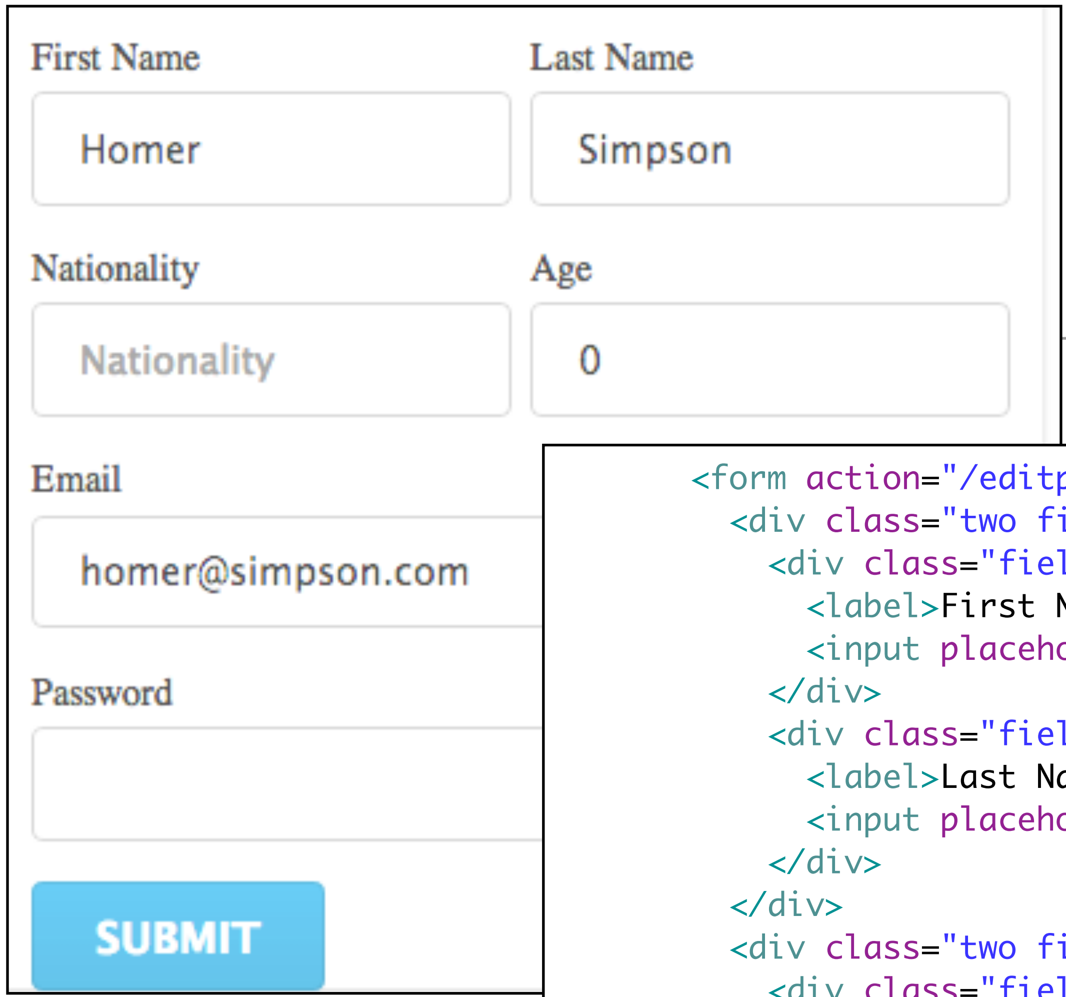
Assignment Story 5

- Provide a way for a user, once logged in, to change some of their profile information. You could take two approaches to this:
  - Provide some extra fields on the home profile which could all be changed when the 'changeText' button is pressed.
  - Provide a link on the home profile - say 'edit details' - which takes you to a new page where you can edit the details.

Story 5 Solution - New Routes

```
# Edit Profile
GET   /editprofile                                          EditProfile.index
POST  /editprofile/change                                   EditProfile.change
```

Story 5 Solution : New View



Homer

Simpson

Nationality

0

homer@simpson.com

```
<form action="/editprofile/change" method="POST">
  <div class="two fields">
    <div class="field">
      <label>First Name</label>
      <input placeholder="First Name" type="text" name="firstName" value="${user.firstName}">
    </div>
    <div class="field">
      <label>Last Name</label>
      <input placeholder="Last Name" type="text" name="lastName" value="${user.lastName}">
    </div>
  </div>
  <div class="two fields">
    <div class="field">
      <label>Nationality</label>
      <input placeholder="Nationality" type="text" name="nationality" value="${user.nationality}">
    </div>
    <div class="field">
      <label>Age</label>
      <input placeholder="Age" type="text" name="age"value="${user.age}">
    </div>
  </div>    <div class="field">
    <label>Email</label>
    <input placeholder="Email" type="text" name="email" value="${user.email}">
  </div>
  <div class="field">
    <label>Password</label>
    <input type="password" name="password">
  </div>
  <button class="ui blue submit button">Submit</button>
</form>
```

views/EditProfile/index.html

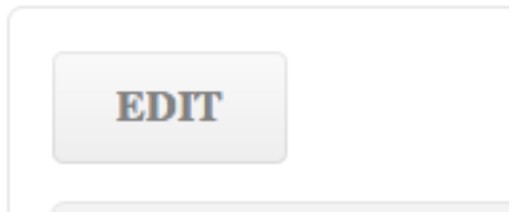
Story 5 : New Controller

```
public class EditProfile extends Controller
{

  public static void change (String firstName,   String lastName, int    age,
                             String nationality, String email,    String password, String password2)
  {
    User user = Accounts.getLoggedInUser();
    user.firstName = firstName;
    user.lastName = lastName;
    user.email = email;
    user.nationality = nationality;
    user.age = age;
    user.password = password;
    user.save();
    Profile.index();
  }

  public static void index()
  {
    User user = Accounts.getLoggedInUser();
    render(user);
  }
}
```

Story 5: New Button

- views/Profile/index.html

```
<a href="/editprofile" class="ui button"> Edit </a>
```



Story 6

- The Members page now shows a list of all members - including the currently logged in member. This clearly makes no sense and we should try to remove the current member from the list.
- HINT: This is the Members controller index method:

```
public static void index()
{
  List<User> users = User.findAll();
  render(users);
}
```

- The challenge is to remove the current user from the users list before we send it to the view. Objects can be deleted from a list using the remove method:

```
user.remove(someUser);
```

Story 6 Solution

```
public class Members extends Controller
{
  public static void index()
  {
    User user = Accounts.getLoggedInUser();

    List<User> users = User.findAll();

    users.remove(user);
    render(users);
  }
```

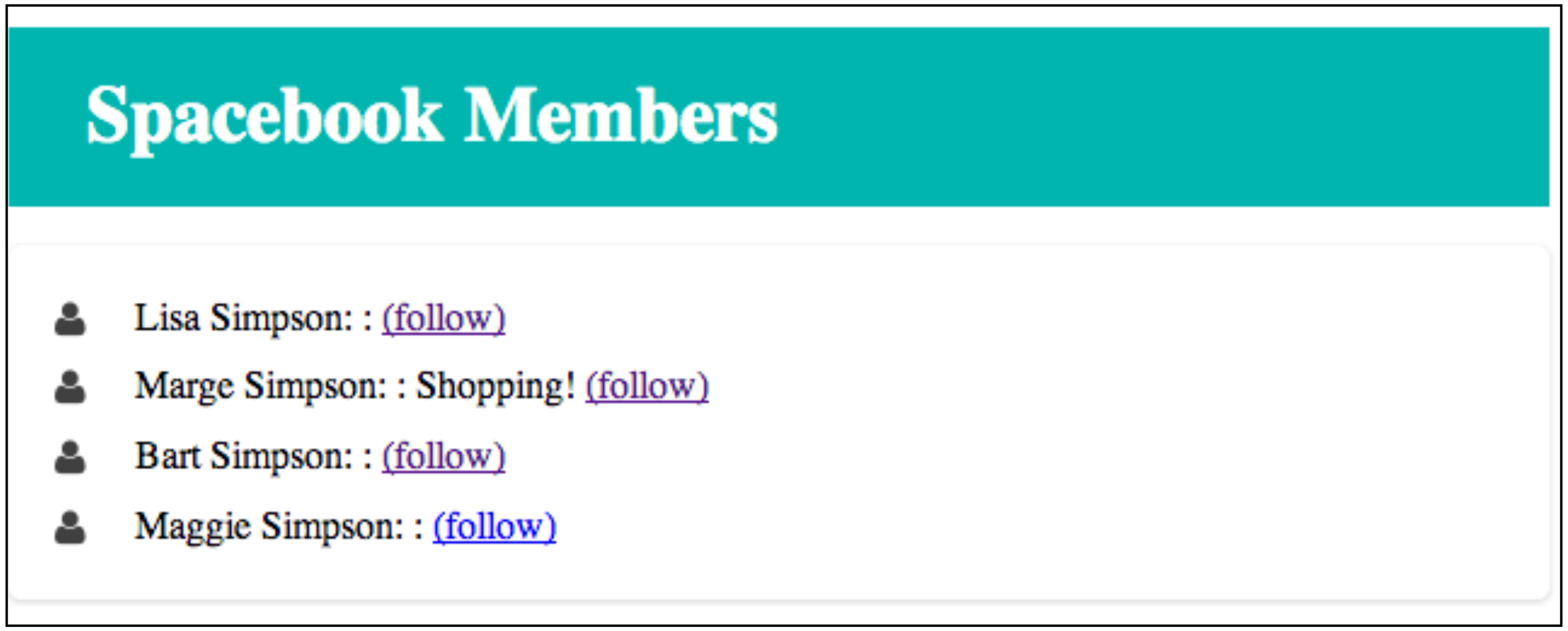
Story 7

- In the members page - show each users name + their current status message.

Story 7 Solution

- views/Members/index.html

```
<section class="ui raised segment">
  <div class="ui list">
    #{list items:users, as:'user'}
      <div class="ui item">
        <i class="user icon"></i>
        ${user.firstName} ${user.lastName}: : ${user.statusText}  <a href="members/follow/${user.id}"> (follow) </a>
      </div>
    #{/list}
  </div>
</section>
```



Spacebook Members

Lisa Simpson: : (follow)

Marge Simpson: : Shopping! (follow)

Bart Simpson: : (follow)

Maggie Simpson: : (follow)

Story 8

- On a users home profile, provide a way for two images to be uploaded - one will be their profile picture as currently implemented, the second is to be to be called the "thumbnail" image, which can be used in story 9 below.

Story 8: Model

- Introduce thumbnailPicture Blob alongside existing profilePicture attribute

```
@Entity
public class User extends Model
{
  //..
  public Blob   profilePicture;
  public Blob   thumbnailPicture;
  //..
}
```

Story 8: Route

```
POST    /profile/uploadpicture/{id}                 Profile.uploadPicture
GET     /profile/getpicture/{id}                    Profile.getPicture

POST    /profile/uploadthumbnail/{id}               Profile.uploadThumbnail
GET     /profile/getthumbnail/{id}                  Profile.getThumbnail
```

- Replicate image upload/fetch routes for thumbmnail

Story 8: Controller

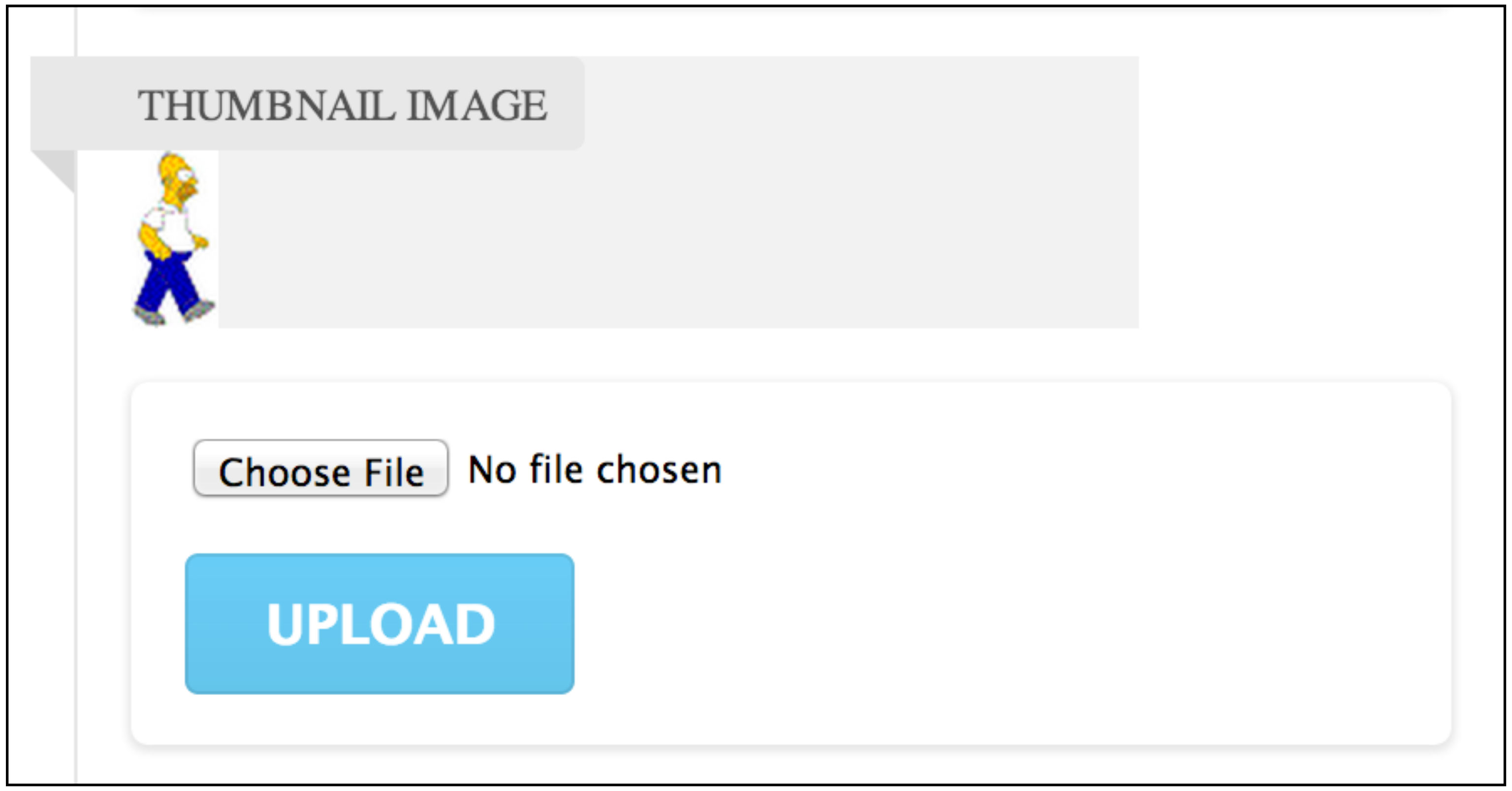
- Replicate actions for thumbnail images

```java
public class Profile extends Controller
{
  ...

  public static void getThumbnail(Long id)
  {
    User user = User.findById(id);
    Blob picture = user.thumbnailPicture;
    if (picture.exists())
    {
      response.setContentTypeIfNotSet(picture.type());
      renderBinary(picture.get());
    }
  }

  public static void uploadThumbnail(Long id, Blob picture)
  {
    User user = User.findById(id);
    user.thumbnailPicture = picture;
    user.save();
    index();
  }
}
```

Story 8: View

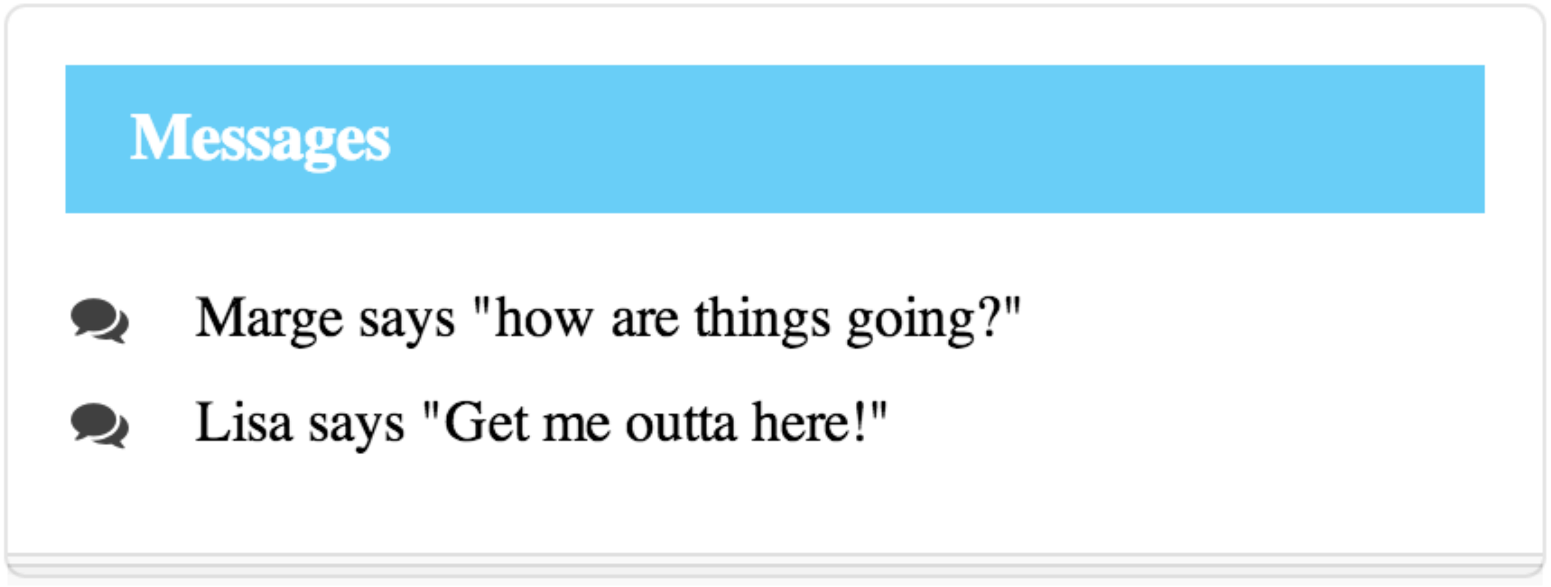
- views/Profile/index.html

```
        <div class="ui medium image">
          <label class="ui ribbon label">Thumbnail Image</label>
          <img src="/profile/getthumbnail/${user.id}">
        </div>
        <section class="ui raised form segment">
          <form action="/profile/uploadthumbnail/${user.id}" method="post" enctype="multipart/form-data">
            <div class="ui field">
              <input type="file" name="picture">  </input>
           </div>
           <button class="ui blue submit button"> Upload</button>
          </form>
        </section>
```



Story 9

- In the messages displayed on the home page, show the message text (as currently) + a link to the senders profile. In this way users can easily navigate from a message directly to the sender, and perhaps leave a message



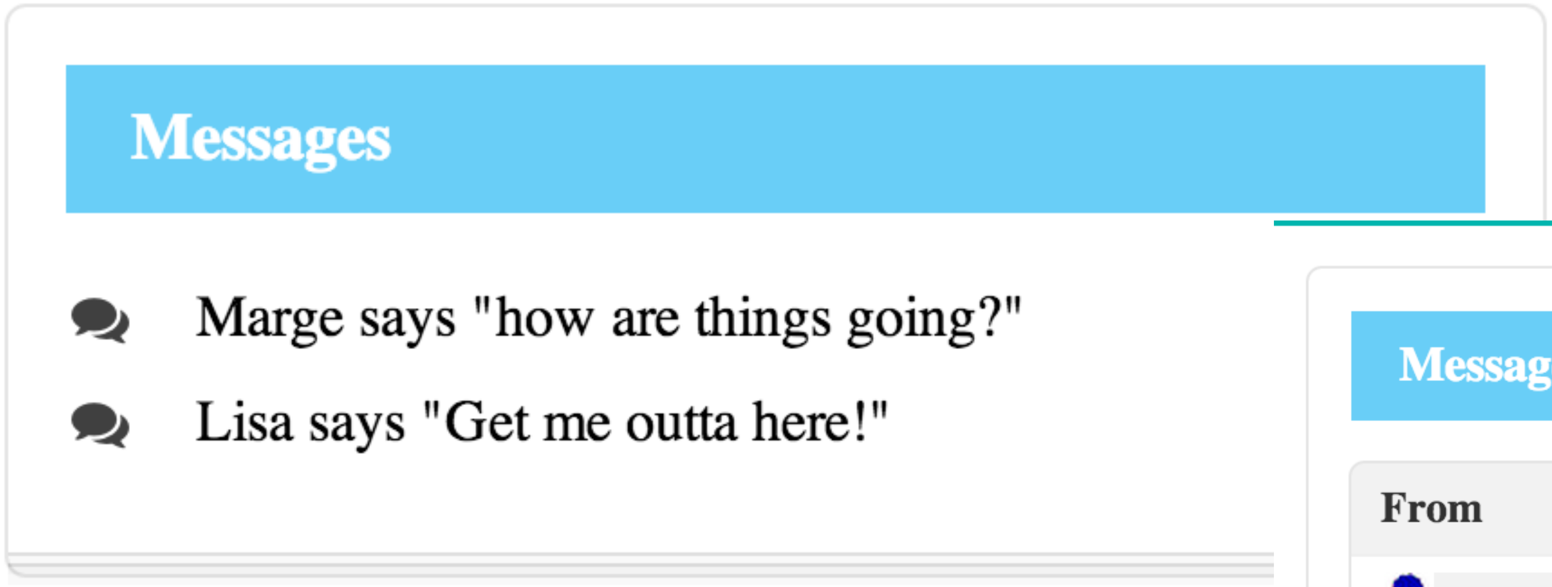
Messages

Marge says "how are things going?"

Lisa says "Get me outta here!"

+ Story 10

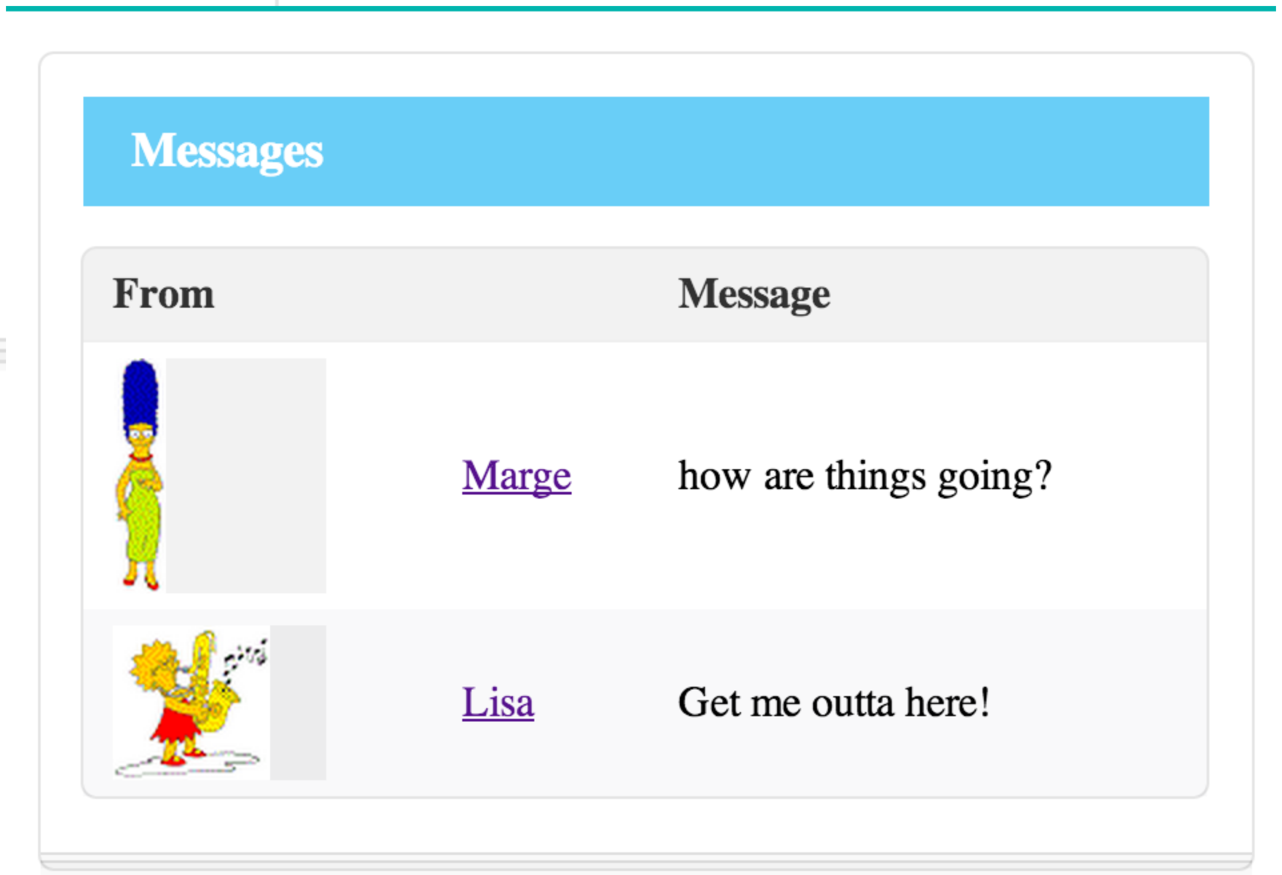
- Turn the list of messages on the home page into a table - with three columns. One column for the message text, one for the senders name, and one showing the senders profile picture.



Messages

Marge says "how are things going?"

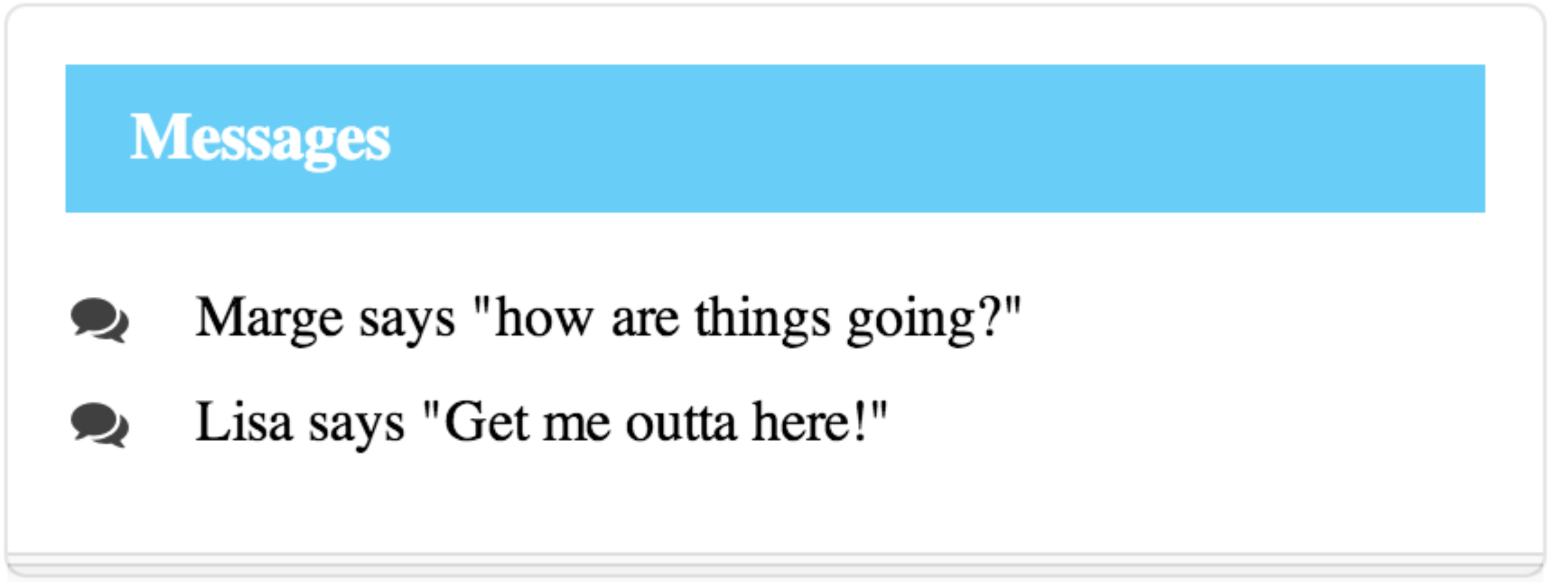
Lisa says "Get me outta here!"



Messages

From		Message
	Marge	how are things going?
	Lisa	Get me outta here!

```
<section class="ui stacked segment">
  <h4 class="ui inverted blue block header">Messages</h4>
  #{if user.inbox.size() ≥ 0}
    <div class="ui list">
      #{list items:user.inbox, as:'message'}
        <div class="item">
          <i class="chat icon"></i>
          ${message.from.firstName} says "${message.messageText}"
        </div>
      #{/list}
    </div>
  #{/if}
</section>
```

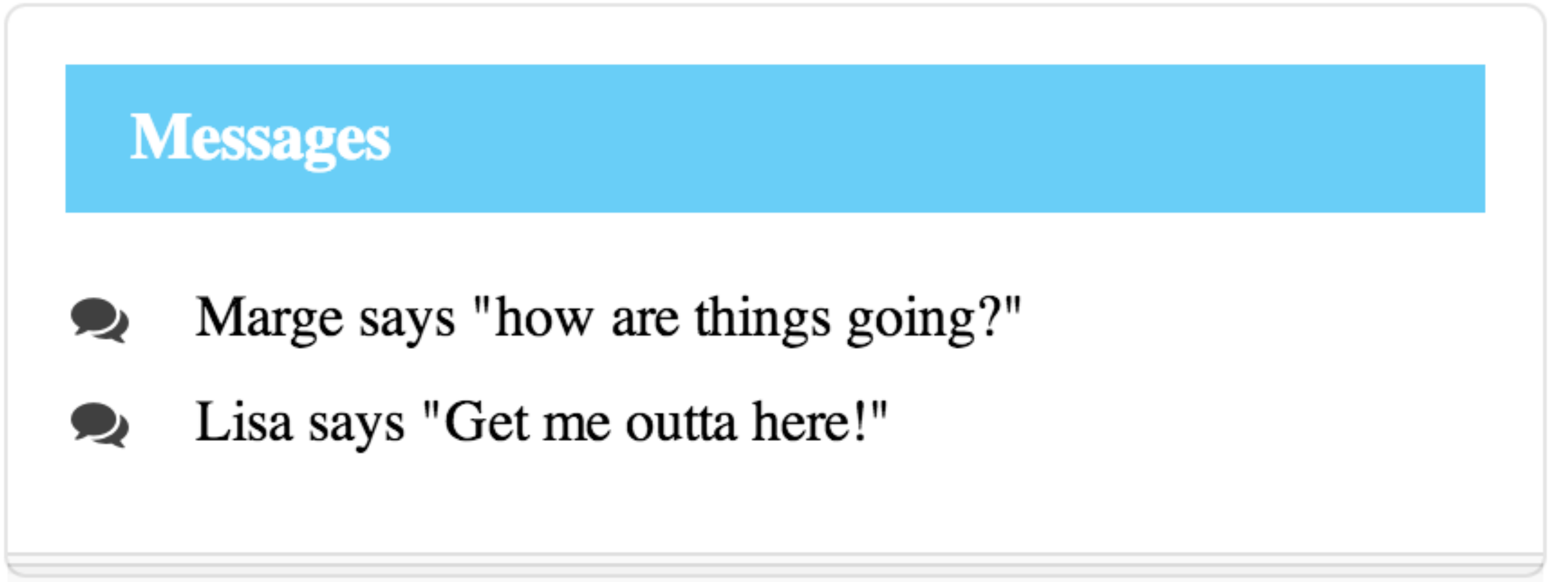


Messages

Marge says "how are things going?"

Lisa says "Get me outta here!"

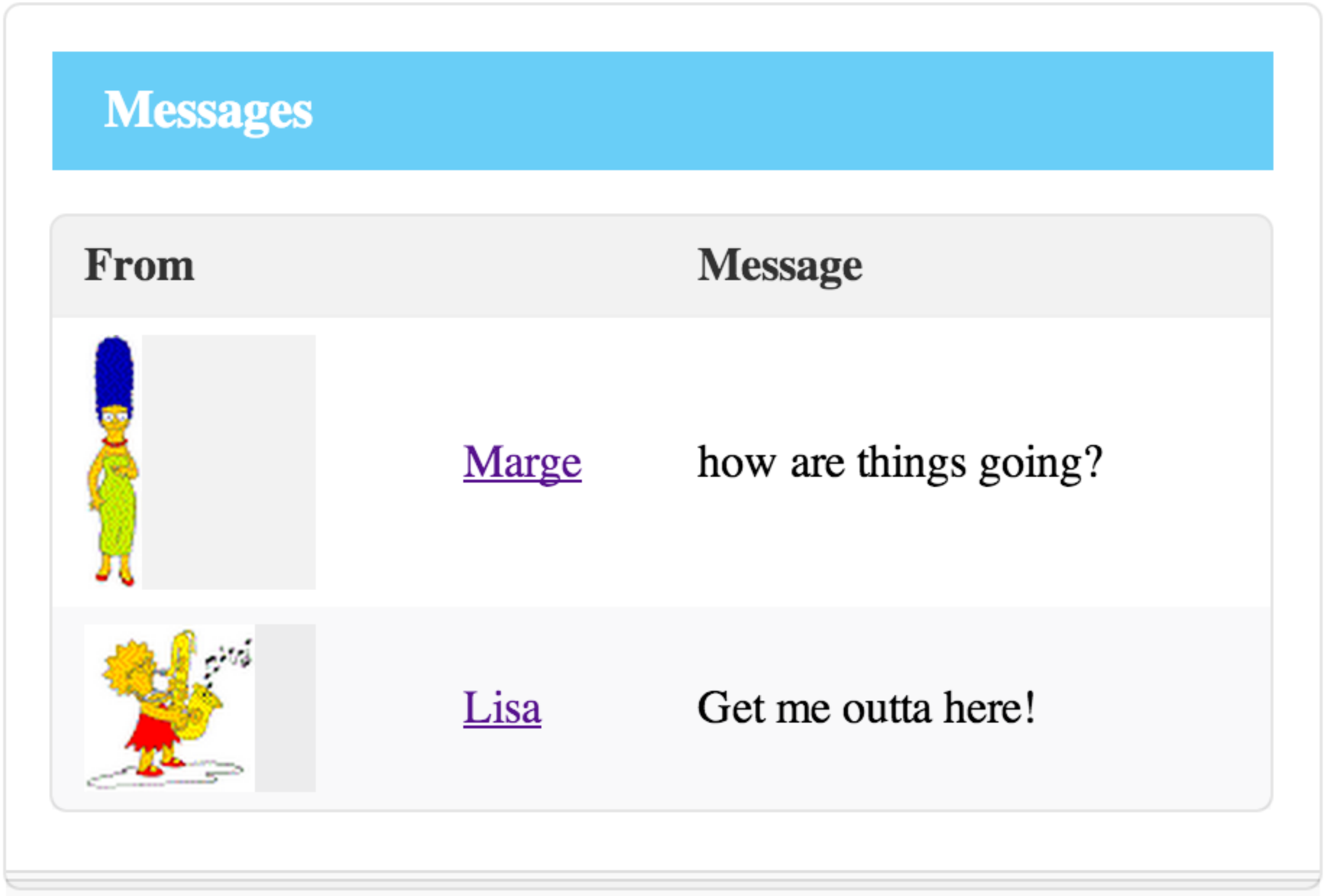
```
<section class="ui stacked segment">
  <h4 class="ui inverted blue block header">Messages</h4>
  #{if user.inbox.size() ≥ 0}
    <div class="ui list">
      #{list items:user.inbox, as:'message'}
        <div class="item">
          <i class="chat icon"></i>
          ${message.from.firstName} says "${message.messageText}"
        </div>
      #{/list}
    </div>
  #{/if}
</section>
```



Messages

Marge says "how are things going?"

Lisa says "Get me outta here!"



Messages

From		Message
	Marge	how are things going?
	Lisa	Get me outta here!

```
<section class="ui stacked segment">
  <h4 class="ui inverted blue block header">Messages</h4>
  #{if user.inbox.size() > 0}
    <table class="ui table segment">
      <thead>
        <tr>
          <th>From</th>
          <th></th>
          <th>Message</th>
        </tr>
      </thead>
      <tbody>
        #{list items:user.inbox, as:'message'}
          <tr>
            <td>
              <div class="ui small image">
                <img src="/profile/getthumbnail/${message.from.id}">
              </div>
            </td>
            <td>
               <a href="/publicprofile/${message.from.id}"> ${message.from.firstName}</a>
             </td>
             <td>
                ${message.messageText}
              </td>
          </tr>
        #{/list}
      </tbody>
    </table>
  #{/if}
</section>
```

9 & 10 Solution