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

Following Friends

Modeling Following List - Current User Model

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

  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);
  }
}
```

Following Model

```
public static void follow(Long id)
{
  User userToFollow = User.findById(id);
  Logger.info("Following " + userToFollow.firstName);
  index();
}
```

- Intent: wish to model a relationship between two users:
  - A User who 'requests' a friendship - ***the source***
  - A User who is the the subject of the request -***the target***

New Model - Friendship

- Model a relationship between two users:
  - A User who ‘requests’ a friendship - ***the source***
  - A User who is the the subject of the request - ***the target***

```
@Entity
public class Friendship extends Model
{
  @ManyToOne()
  public User sourceUser;

  @ManyToOne()
  public User targetUser;

  public Friendship(User source, User target)
  {
    sourceUser = source;
    targetUser = target;
  }
}
```

User->Friendship

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

  @OneToMany(mappedBy = "sourceUser")
  public List<Friendship> friendships = new ArrayList<Friendship>();

  //...
}
```

- User Class has ‘many’ friendship objects representing the list of friends the user has

- Implemented as a List<Friendship> called ‘friendships’

- OneToMany & ManyToOne in a symmetrical relationship
- Visualise as each User having a list of friends
- The list is in fact a collection of friendship objects
- To be explored more throughly in Semester 2!

```java
@Entity
public class User extends Model
{
  //...

  @OneToMany(mappedBy = "sourceUser")
  public List<Friendship> friendships = new ArrayList<Friendship>();

  //...
}
```

```java
@Entity
public class Friendship extends Model
{
  @ManyToOne()
  public User sourceUser;

  @ManyToOne()
  public User targetUser;

  public Friendship(User source, User target)
  {
    sourceUser = source;
    targetUser = target;
  }
}
```

Managing the Relationship: befriend

- Each User class will have a ‘befriend’ method
- This will:
  - create a new friendship object
  - store it in the users list of friendships
  - save the user object, as we have just made a change to its state

```
@Entity
public class User extends Model
{
  //...
  public void befriend(User friend)
  {
    Friendship friendship = new Friendship(this, friend);
    friendships.add(friendship);
    friendship.save();
    save();
  }
  //...
}
```

Managing the Relationship: unfriend

- Each User class will also have an ‘unfriend’ method
- This will:
    - locate the friendship object in the list of friendships
    - Remove this object from the list
    - Delete the object from the database
    - save the user object, as we have just made a change to its state

```
@Entity
public class User extends Model
{
  //...
  public void unfriend(User friend)
  {
    Friendship thisFriendship = null;

    for (Friendship friendship:friendships)
    {
      if (friendship.targetUser== friend)
      {
        thisFriendship = friendship;
      }
    }
    friendships.remove(thisFriendship);
    thisFriendship.delete();
    save();
  }
  //...
}
```

Implementing the ‘Follow’ Action

- Get the User we have been asked to follow (into 'friend')
- Get the currently logged in user (into 'me')
- Befriend the user
- Redisplay the home page

```
public static void follow(Long id)
{
  User friend = User.findById(id);

  String userId = session.get("logged_in_userid");
  User me = User.findById(Long.parseLong(userId));

  me.befriend(friend);
  index();
}
```

Displaying the Friends List on Home View

```
<h2>Friends</h2>
<div class="ui list">
  <div class="item">
    <i class="right triangle icon"></i> <a href="/publicprofile/marge">marge</a>, (<a href="drop/marge">drop</a>)
  </div>
  <div class="item">
    <i class="right triangle icon"></i> <a href="/publicprofile/lisa">lisa</a>, (<a href="drop/lisa">drop</a>)
  </div>
</div>
```

- Currently ‘Hard Coded’
- Will need to replace with a ‘loop’ that lists out the friends.

Displaying the Friends List

- All the information we need is already available to the view.

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

- In the view, write a ‘foreach’ loop to walk through the ‘following’ list in the user object

```
<h2>Friends (${user.friendships.size()})</h2>
<div class="ui list">
  #{list items:user.friendships, as:'friendship'}
    <div class="item">
      <i class="right triangle icon"></i>
      ${friendship.targetUser.firstName} ${friendship.targetUser.lastName}
    </div>
  #{/list}
</div>
```

Displaying the Friends List

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

```
        <h2>Friends (${user.friendships.size()})</h2>
        <div class="ui list">
          #{list items:user.friendships, as:'friendship'}
            <div class="item">
              <i class="right triangle icon"></i>
              ${friendship.targetUser.firstName} ${friendship.targetUser.lastName}
            </div>
          #{/list}
        </div>
```

Displaying the Friends List

```
<h2>Friends (${user.friendships.size()})</h2>
<div class="ui list">
  #{list items:user.friendships, as:'friendship'}
    <div class="item">
      <i class="right triangle icon"></i>
      ${friendship.targetUser.firstName} ${friendship.targetUser.lastName}
    </div>
  #{/list}
</div>
```

Displaying the Friends List

```
<h2>Friends (${user.friendships.size()})</h2>
<div class="ui list">
  #{list items:user.friendships, as:'friendship'}
    <div class="item">
      <i class="right triangle icon"></i>
      ${friendship.targetUser.firstName} ${friendship.targetUser.lastName}
    </div>
  #{/list}
</div>
```

- Just displays list of friends names - no links

Displaying the Friends List

```
<h2>Friends (${user.friendships.size()})</h2>
<div class="ui list">
  #{list items:user.friendships, as:'friendship'}
    <div class="item">
      <i class="right triangle icon"></i>
      <a href="/publicprofile/${friendship.targetUser.id}">
        ${friendship.targetUser.firstName} ${friendship.targetUser.lastName}
      </a>
    </div>
  #{/list}
</div>
```

- Introduce Link to friends public profile page

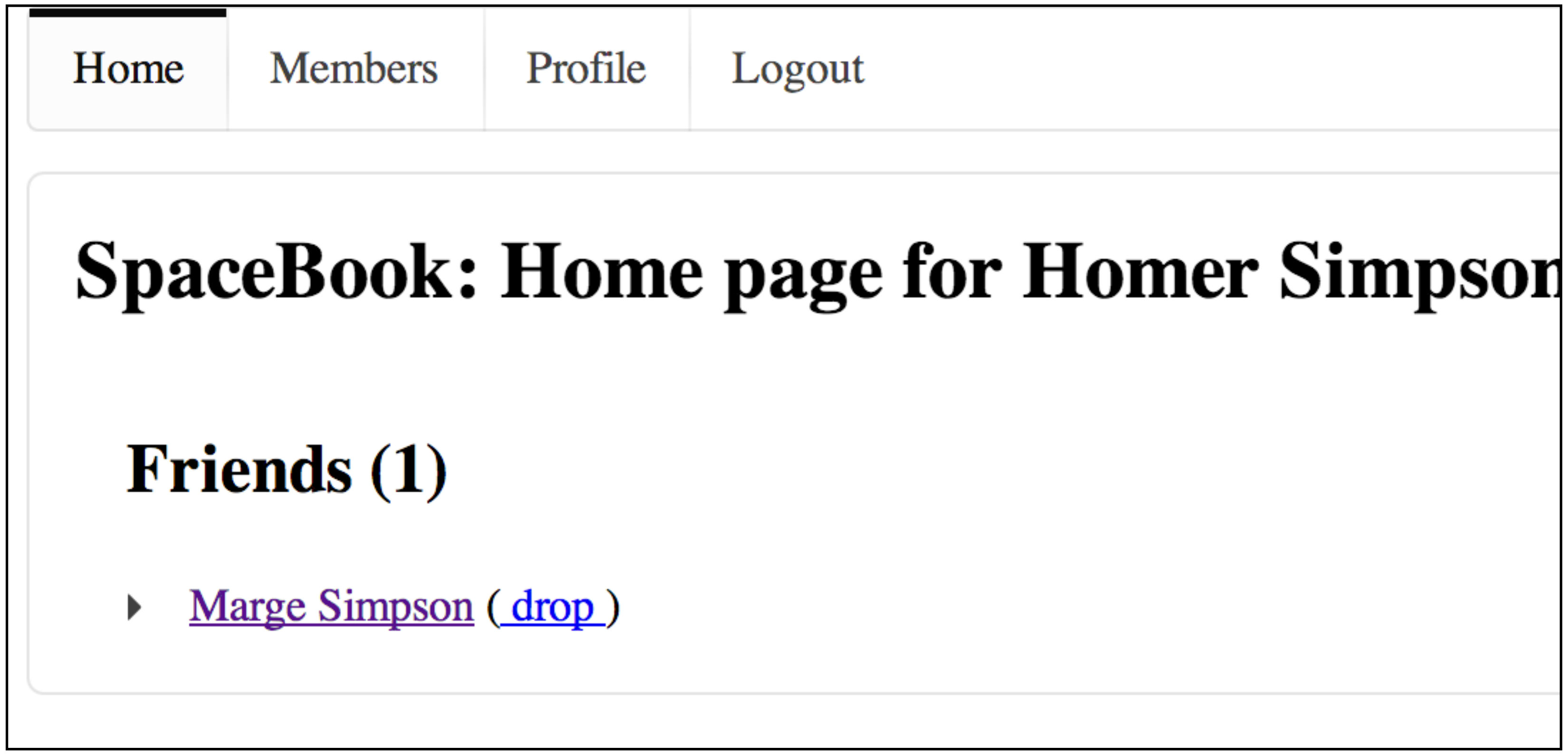
Displaying the Friends List

```
<h2>Friends (${user.friendships.size()})</h2>
<div class="ui list">
  #{list items:user.friendships, as:'friendship'}
    <div class="item">
      <i class="right triangle icon"></i>
      <a href="/publicprofile/${friendship.targetUser.id}">
        ${friendship.targetUser.firstName} ${friendship.targetUser.lastName}
      </a>
      (<a href="/home/drop/${friendship.targetUser.id}"> drop </a>)
    </div>
  #{/list}
</div>
```

- introduce Link to drop friend from friends list

Displaying the Friends List

```
<h2>Friends (${user.friendships.size()})</h2>
<div class="ui list">
  #{list items:user.friendships, as:'friendship'}
    <div class="item">
      <i class="right triangle icon"></i>
      <a href="/publicprofile/${friendship.targetUser.id}">
        ${friendship.targetUser.firstName} ${friendship.targetUser.lastName}
      </a>
      (<a href="/home/drop/${friendship.targetUser.id}"> drop </a>)
    </div>
  #{/list}
</div>
```



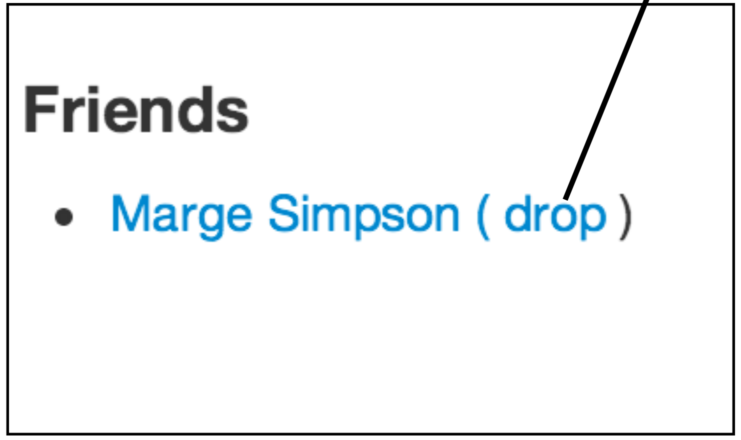
SpaceBook: Home page for Homer Simpson

Friends (1)

Marge Simpson (drop)

Dropping Friends

```
<h2>Friends (${user.friendships.size()})</h2>
<div class="ui list">
  #{list items:user.friendships, as:'friendship'}
    <div class="item">
      <i class="right triangle icon"></i>
      <a href="/publicprofile/${friendship.targetUser.id}">
        ${friendship.targetUser.firstName} ${friendship.targetUser.lastName}
      </a>
      (<a href="/home/drop/${friendship.targetUser.id}"> drop </a>)
    </div>
  #{/list}
</div>
```



Friends

- Marge Simpson (drop)

```
public class Home extends Controller
{
  //...

  public static void drop(Long id)
  {
    String userId = session.get("logged_in_userid");
    User user = User.findById(Long.parseLong(userId));

    User friend = User.findById(id);

    user.unfriend(friend);
    Logger.info("Dropping " + friend.email);
    index();
  }
}
```

- Find the currently logged in user
- Find the user we want to drop
- drop the friend (unfriend)
- Redisplay the view

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit