# Mobile Application Development

Higher Diploma in Science in Computer Science

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

# Modeling & Testing witpressapp

# Blog

- Each user can have a blog
- A blog consists of multiple ’Posts’
- Each Post consists of:
  - Title
  - Content

# User

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

  @OneToMany(cascade=CascadeType.ALL)
  public List<Post> posts;

  public User(String firstName, String lastName,String email, String password)
  {
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.password = password;
    posts = new ArrayList<Post>();
  }

  public void addPost (Post post)
  {
    posts.add(post);
  }

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

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

# Post

```
@Entity
public class Post extends Model
{
  public String title;
  @Lob
  public String content;

  @OneToMany(cascade = CascadeType.ALL)
  public List<Comment> comments;

  public Post(String title, String content)
  {
    this.title = title;
    this.content = content;
    this.comments = new ArrayList<Comment>();
  }

  public void addComment(Comment comment)
  {
    comments.add(comment);
  }

  public String toString()
  {
    return title;
  }
}
```

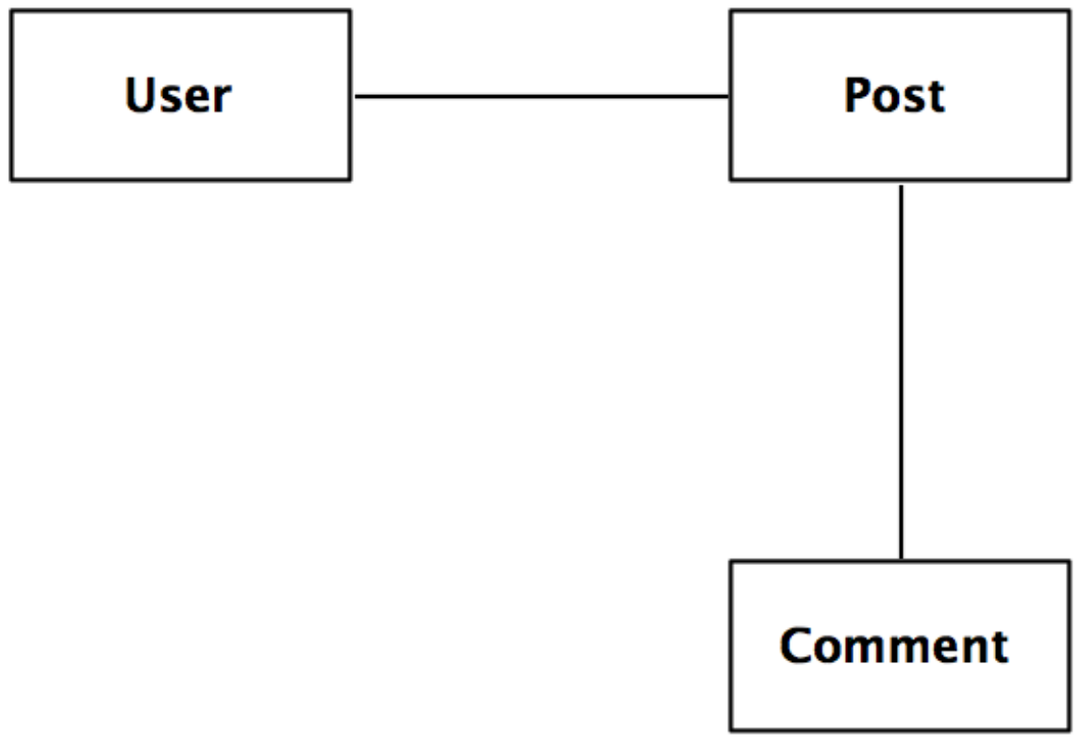
# Comment

```
@Entity
public class Comment extends Model
{
  public String content;

  public Comment(String content)
  {
    this.content = content;
  }
}
```

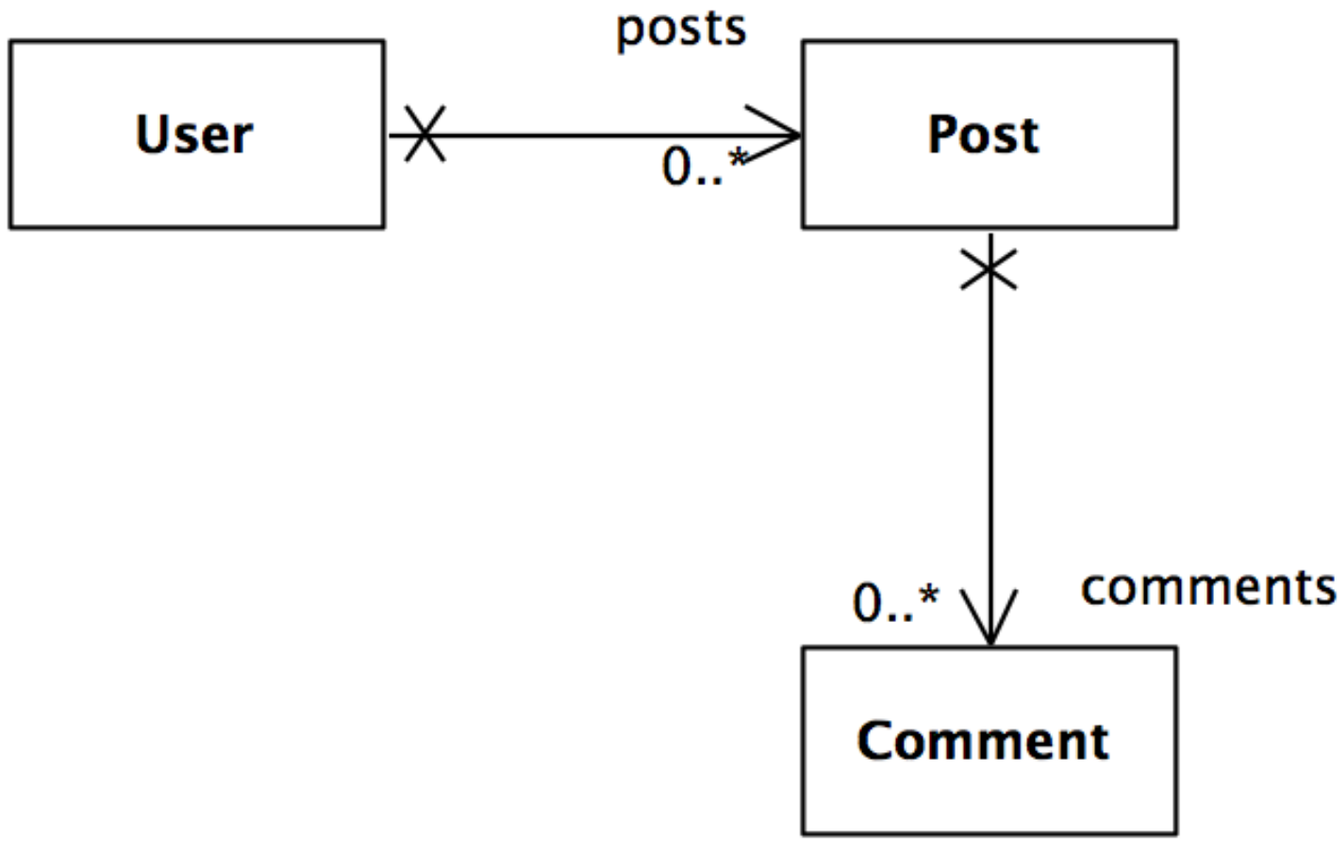
# Model

- Each class represented by a simple rectangle
- If a class is ‘aware’ of another class, then draw a line connecting them.
- These lines are called ‘associations’



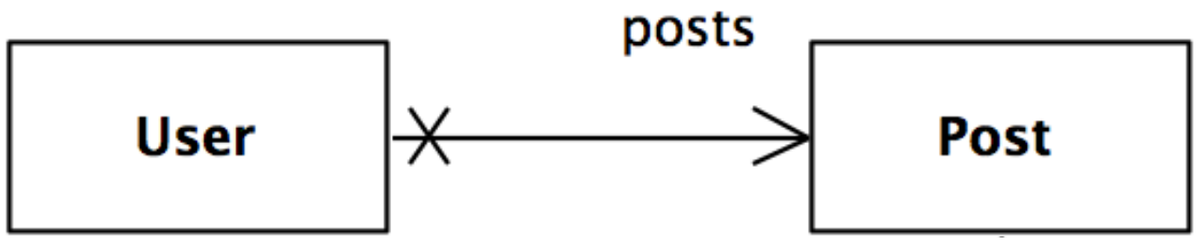
# Model

- Enrich the associations with extra information
  - Navigability
  - Role name
  - Cardinality



# Navigability

- Indicates whether one class can be ‘reached’ from another
- eg:
  - User maintains a collection of posts
  - However, posts (in this model at least) are unaware of the user they belong to




```
public class User extends Model
{
  //...
  @OneToMany(cascade=CascadeType.ALL)
  public List<Post> posts;
  //...
}
```

```
public class Post extends Model
{
  //..
}
```

# Role Name

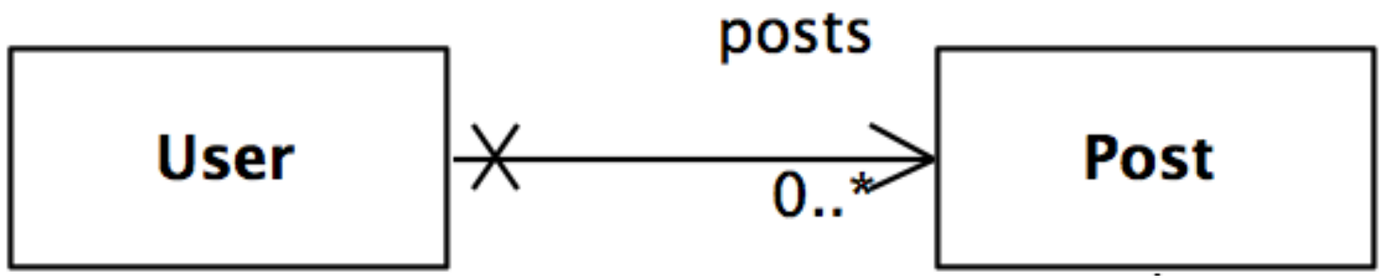
- The name of the attribute used to traverse the navigable path
- eg:
  - User Post collection is called ‘posts’
  - Drawn at the ‘other’ end of the association


```
public class User extends Model
{
  //...
  @OneToMany(cascade=CascadeType.ALL)
  public List<Post> posts;
  //...
}
```

```
public class Post extends Model
{
  //..
}
```

# Cardinality

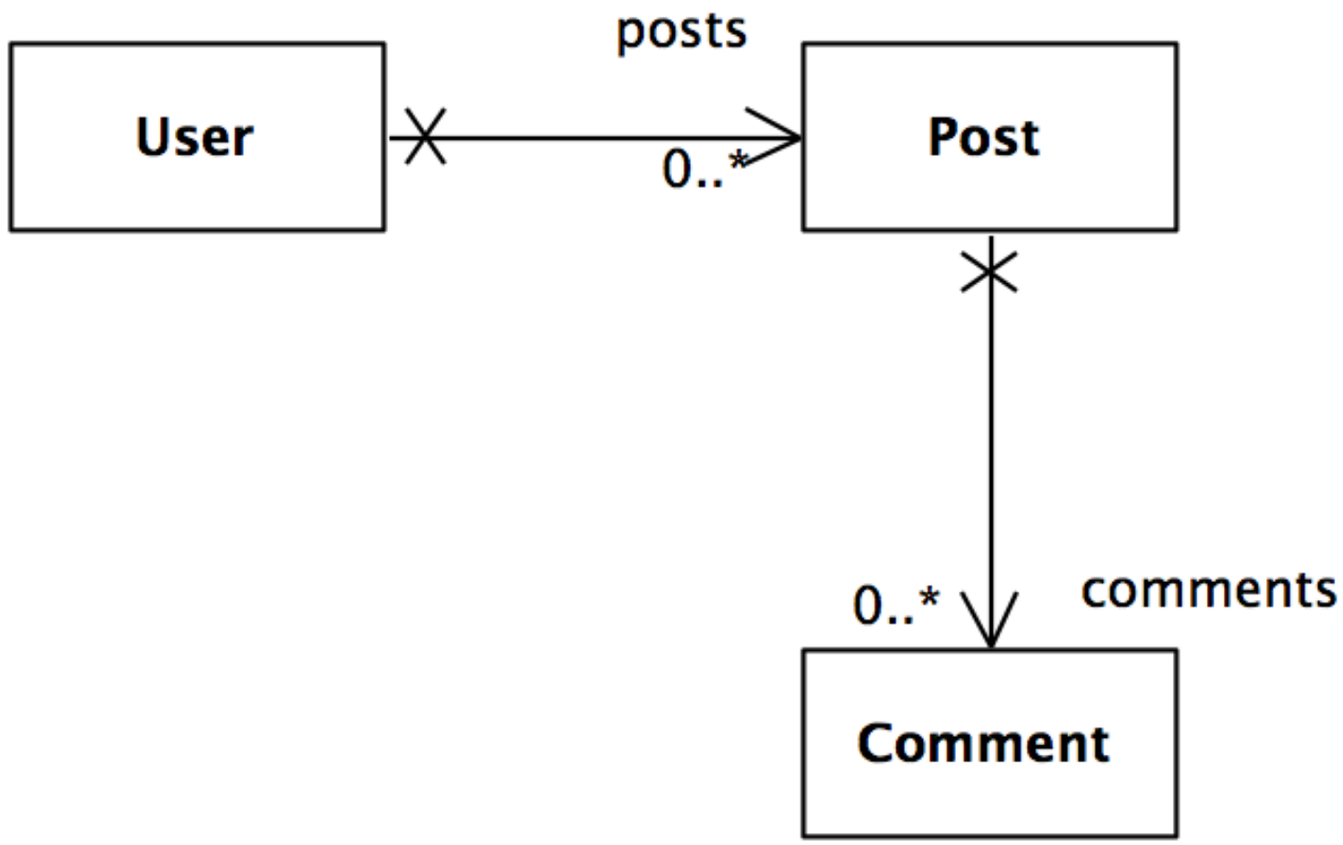
- How objects will be managed during the application lifecycle
  - 1 - a one-to-one relationship
  - 0..* - a zero-to-many relationship
- OneToMany == 0..* in modeling terms (as the collection can be empty)




```
public class User extends Model
{
  //...
  @OneToMany(cascade=CascadeType.ALL)
  public List<Post> posts;
  //...
}
```

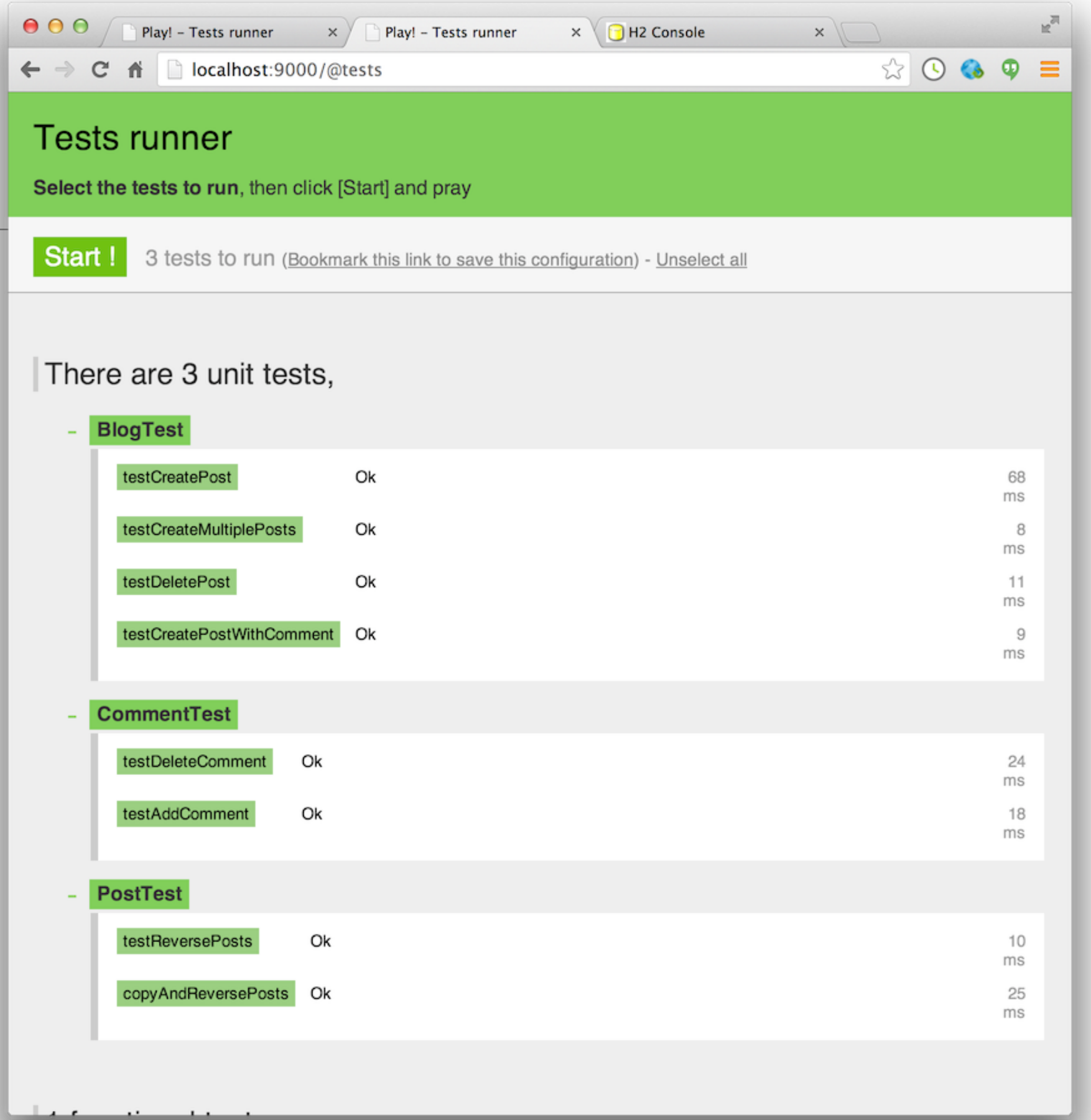
```
public class Post extends Model
{
  //..
}
```

# Witpress Model



# Testing witpress (1)

- Exercise 1
  - Write these tests



# Exercise 2

**Exercise 2:**

Clone and run the tests on your NUCs. To do this you will need Eclipse + play installed.

If not already installed, install Eclipse on the NUCs. This should be relatively straightforward as there is a linux installer in the main Eclipse download repositories.

To install Play on the NUCs - make sure to locate version 1.3.1. To do this, just unzip the archive into

```
$HOME/dev/play
```

Then set the path to include that folder. One way of doing this is to create a file called .bashrc in your $HOME folder, and include the following:

```
PATH="$HOME/bin:$PATH:$HOME/dev/play"
export PATH
```

# Exercise 3

- **Exercise**
    - Currently a Blog is not modeled - a blog just a collection of posts. Model a Blog as a first class object, developing tests to verify the implementation as you go. Do this in two versions:
        - Each user can have a single blog.
        - Each user may have zero or more blogs
- Dont worry about the UI, just focus on the tests