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

Repositories - Part 1

Repositories Lab 10 Step Programme!

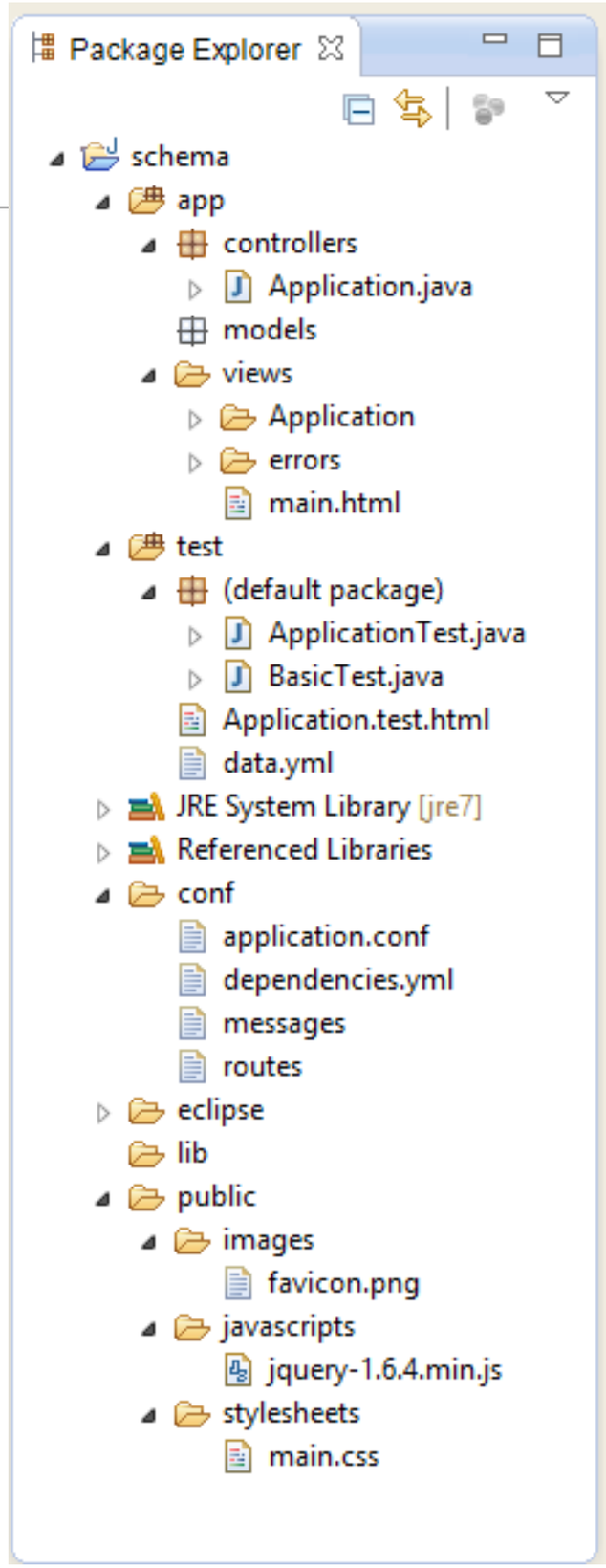
1. Install Git, Sourcetree & Egit
2. Create Schema Project
3. Commit Baseline Schema project to git repository
4. Create 'Walking Skeleton' Version
5. Commit "Walking Skeleton" Version
6. Commit Simple Player & Club Tests
7. Commit Club/Player One to Many Relationship
8. Manipulating Repository using Sourcetree & Bitbucket
9. Commit Bidirectional Player/Club
10. Checkout Earlier Revision

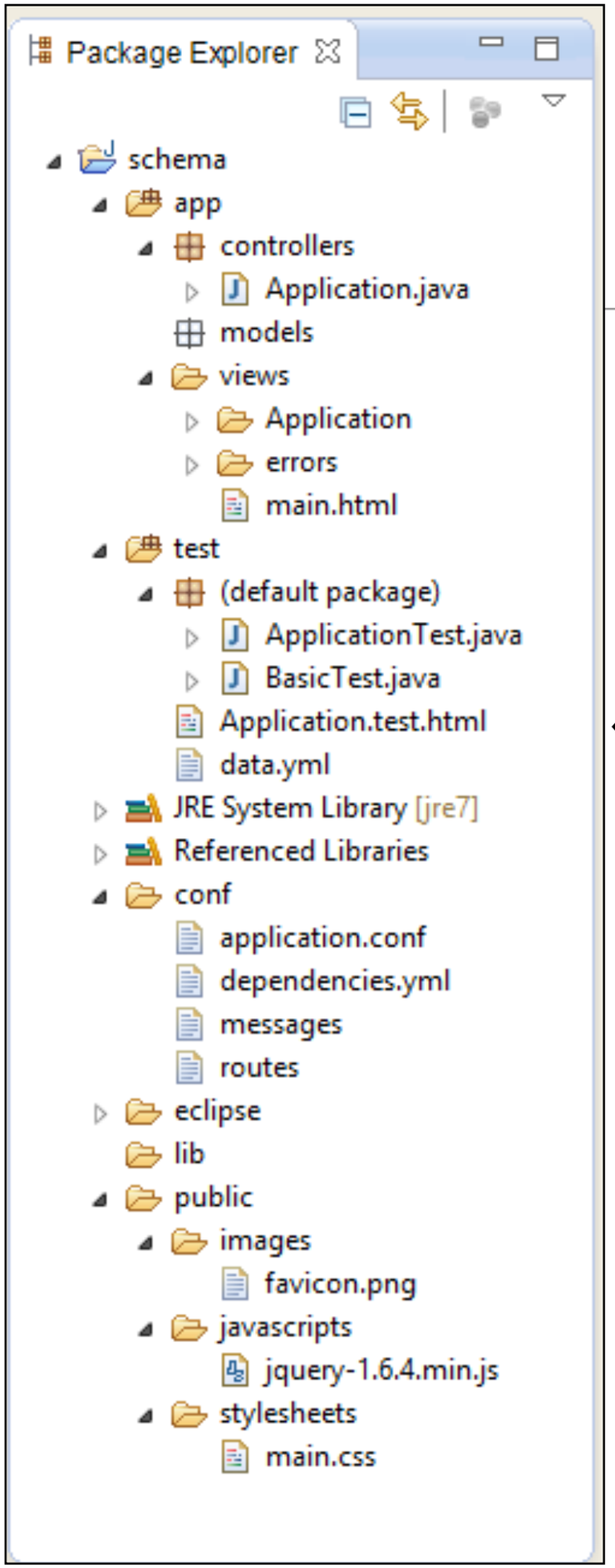
Part 1

1. Install Git, Sourcetree & Egit
2. Create Schema Project
3. Commit Baseline Schema project to git repository
4. Create 'Walking Skeleton' Version
5. Commit "Walking Skeleton" Version

The Need for Repositories

- What, precisely, is in a Play project?

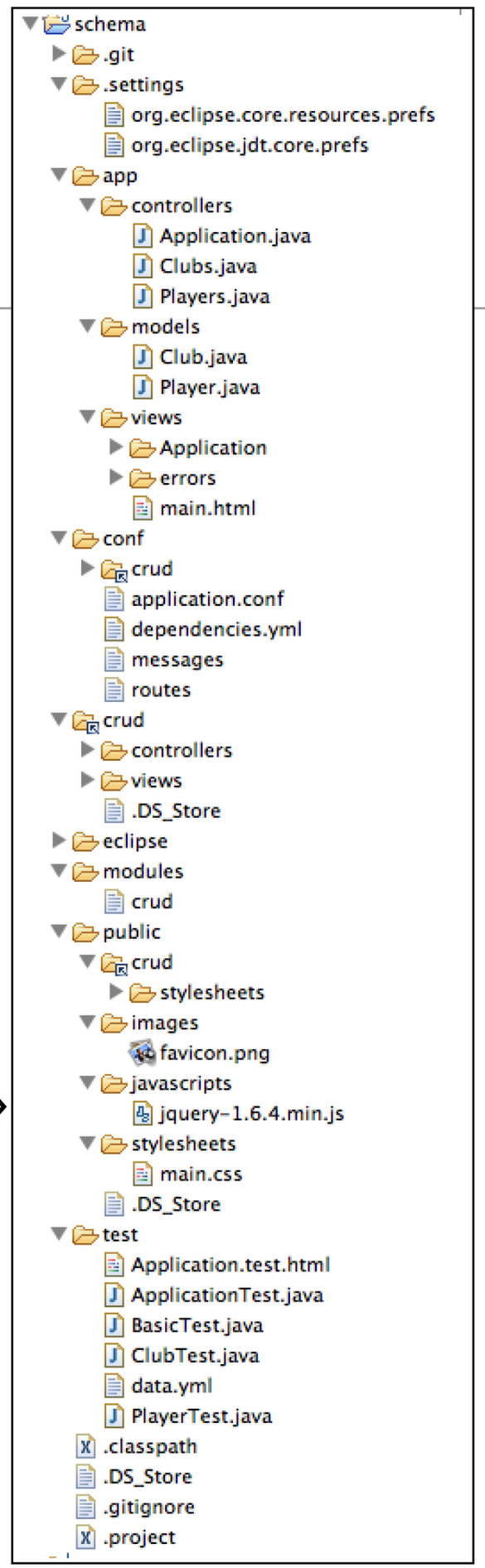




Play Project = collection of files in a folder

- Eclipse ‘Package Explorer’ view provides a ‘logical’ listing

- Eclipse ‘Navigator’ view provides ‘physical’ listing



Where do you store these files?

- In a folder, on hard disk (default)

  +

    - In another folder, on memory stick (backup)

    - In dropbox (easy sharing)

- Is this enough?

What, exactly, is stored in these folders?

- The latest version of the application you have been working on.
- What about earlier versions?
    - The last version that was working correctly
    - The version that you used as the starting point for the current feature you may be working in
    - The version you submitted as an assignment solution
    - The version currently deployed to cloudbees

Versions

- Not only is it important to have the files in your project stored securely somewhere...
- ... you also need to consider the various version of the files over time
- ... and be able to recover earlier versions.
- Additionally, you may like to have different ‘branches’ over time, which may represent versions of your applications targeted at different customers, platforms etc...

Tools

- Git - the core “version control” system
- Sourcetree - a visual client for git
- Egit - an eclipse plugin for git
- Bitbucket - a cloud service for hosting git repositories

git --distributed-even-if-your-workflow-isnt

Search entire site...

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Learn Git in your browser for free with Try Git.

About

The advantages of Git compared to other source control systems.

Documentation

Command reference pages, Pro Git book content, videos and other material.

Downloads

GUI clients and binary releases for all major platforms.

Community

Get involved! Mailing list, chat, development and more.

Pro Git by Scott Chacon is available to read online for free. Dead tree versions are available on Amazon.com.

Mac GUIs **Tarballs**

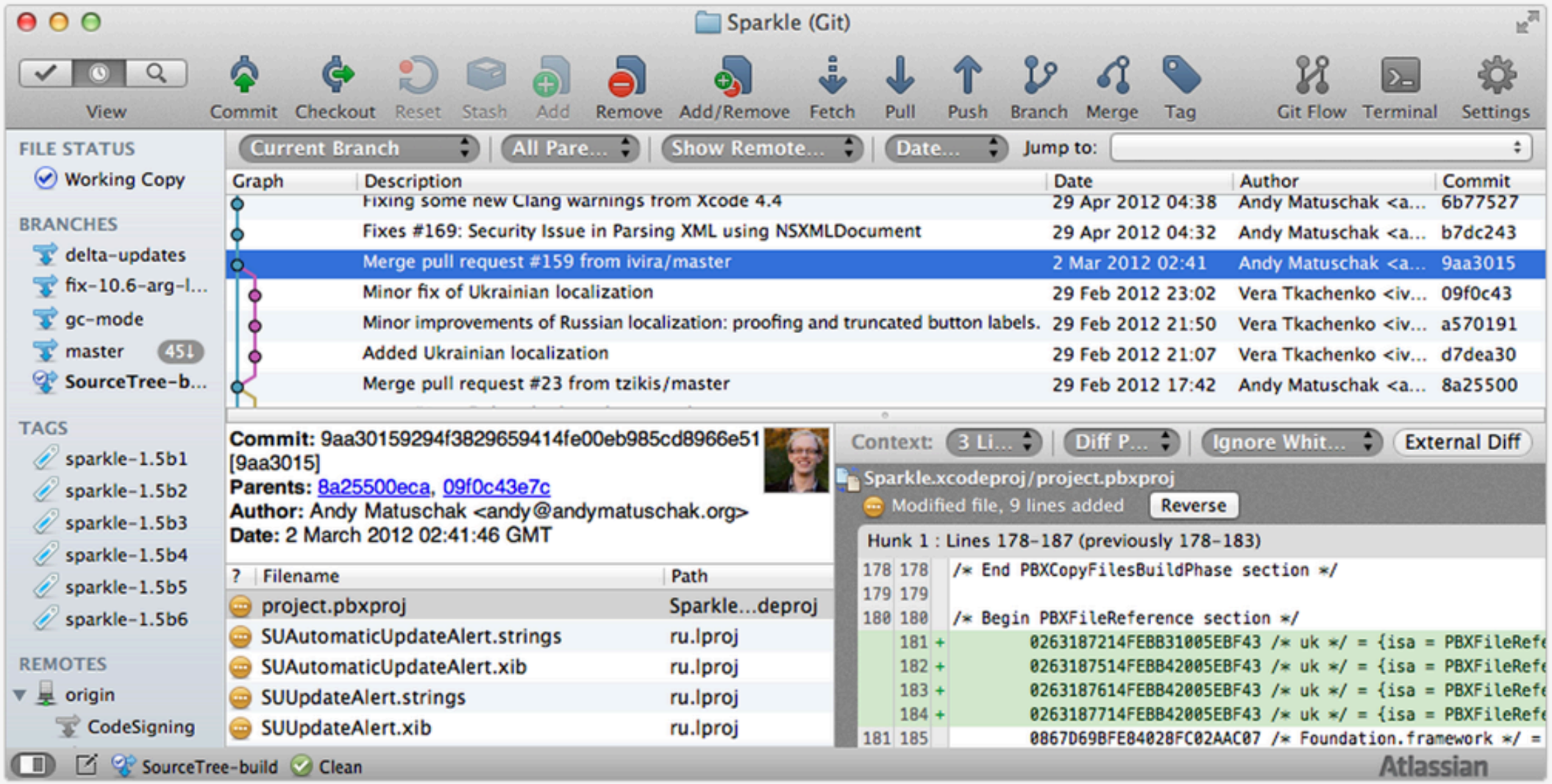
Windows Build **Source Code**

Atlassian SourceTree

A free Git & Mercurial client for Windows or Mac.

Download SourceTree Free
Mac OS X 10.6+

Download SourceTree for Windows 7+



Home Download Documentation Support Developers

EGit

Download
Eclipse Distribution,
Update Site, Dropins

Documentation
Tutorials, Examples,
Videos, Online Referen

EGit

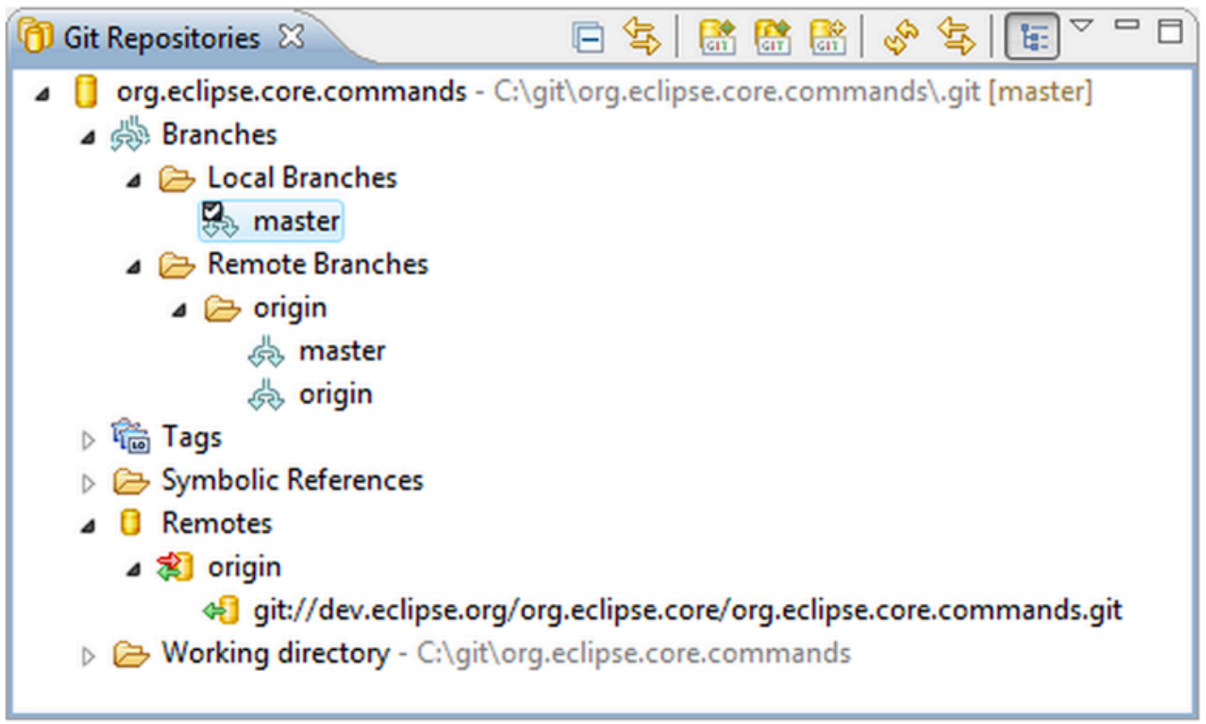
289 7

Favorite this @ Eclipse Marketplace

About This Project

EGit is an Eclipse Team provider for the **Git version control system**. Git is a distributed SCM, which means every developer has a full copy of all history of every revision of the code, making queries against the history very fast and versatile.

The EGit project is implementing Eclipse tooling on top of the **JGit** Java implementation of Git.

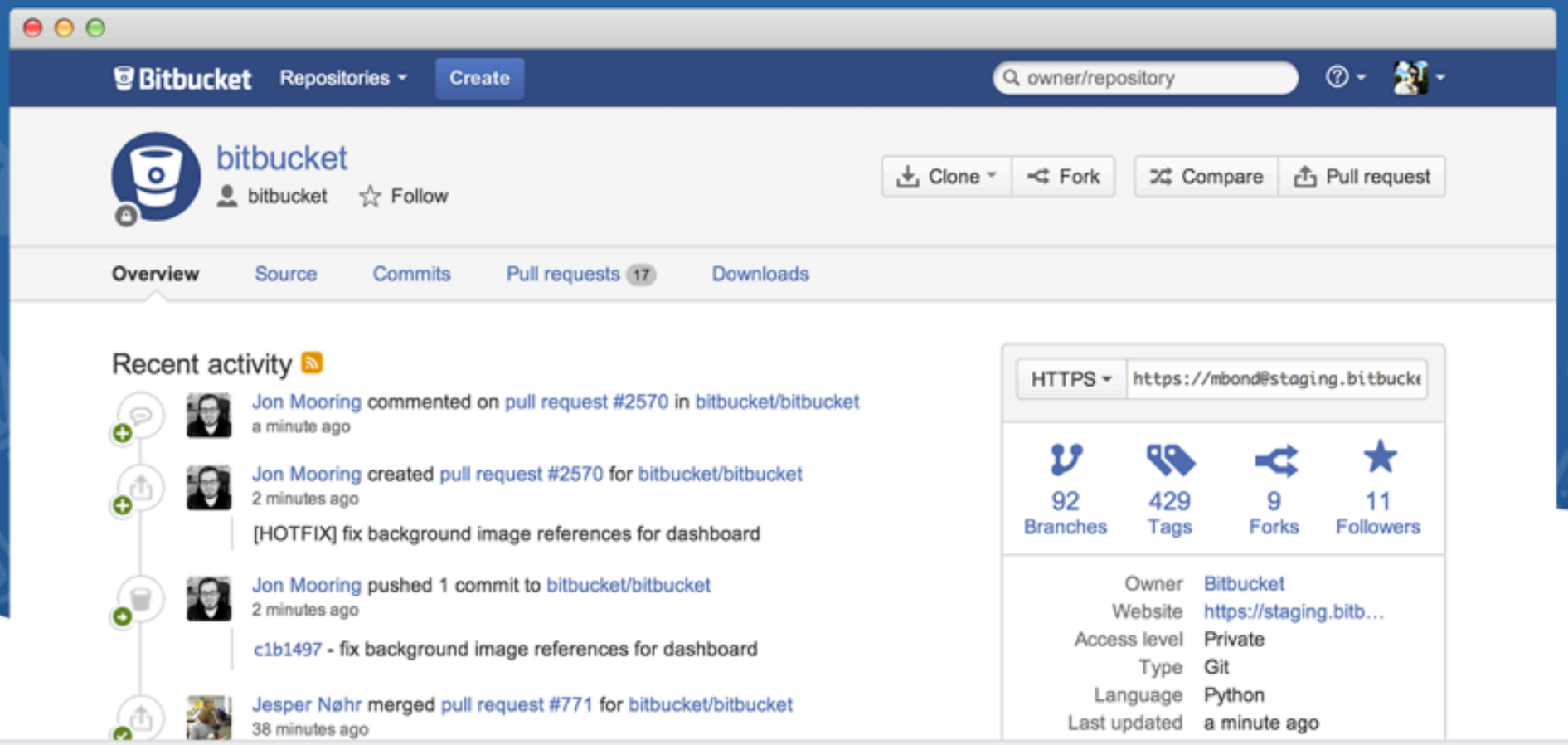


Bitbucket Pricing & signup What's new owner/repository Sign up Log in

Unlimited private repositories

FREE FOR 5 USERS + GIT OR MERCURIAL
LIGHTWEIGHT CODE REVIEW + MAC AND WINDOWS CLIENT

Username
Email
Password
Sign up for free



salesforce TESLA The New York Times PayPal DHL SECOND LIFE

Repositories Lab - Step 1

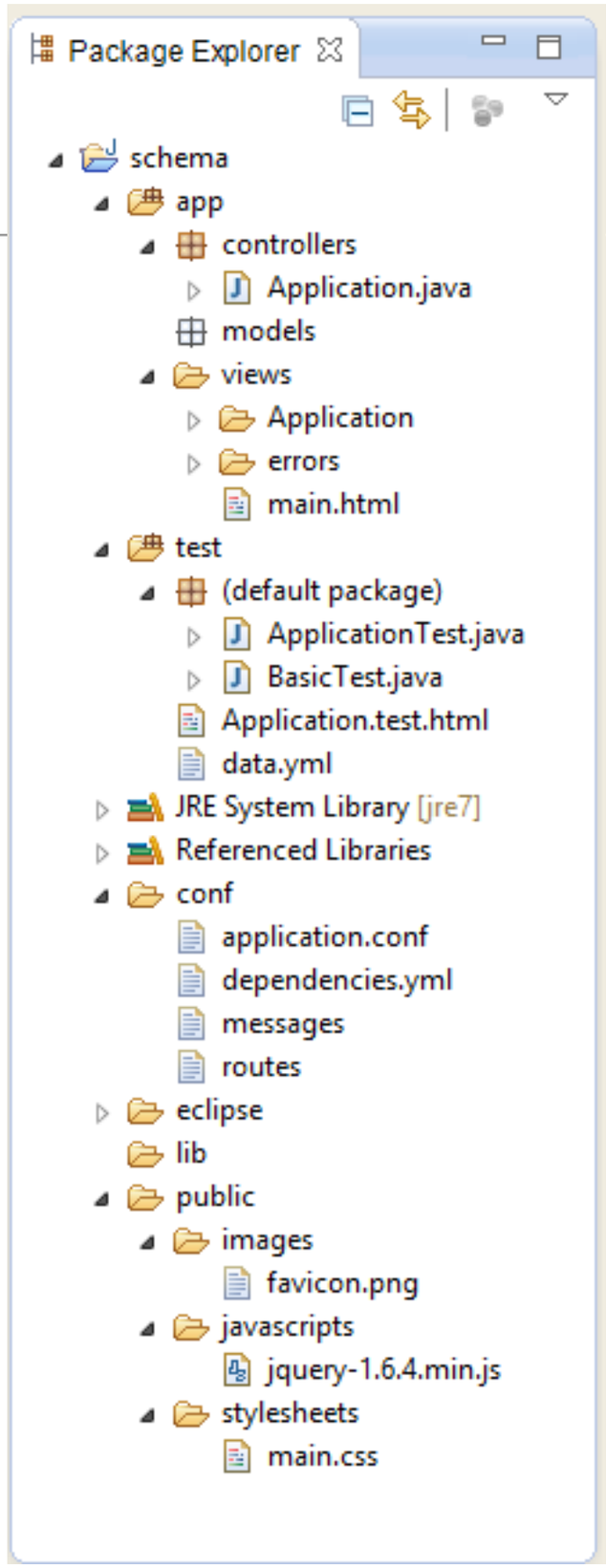
- Install git
- Install SourceTree
- Install Egit
- Create a free account on bitbucket.com

Step 2

- Create a new Play Project called “Schema”

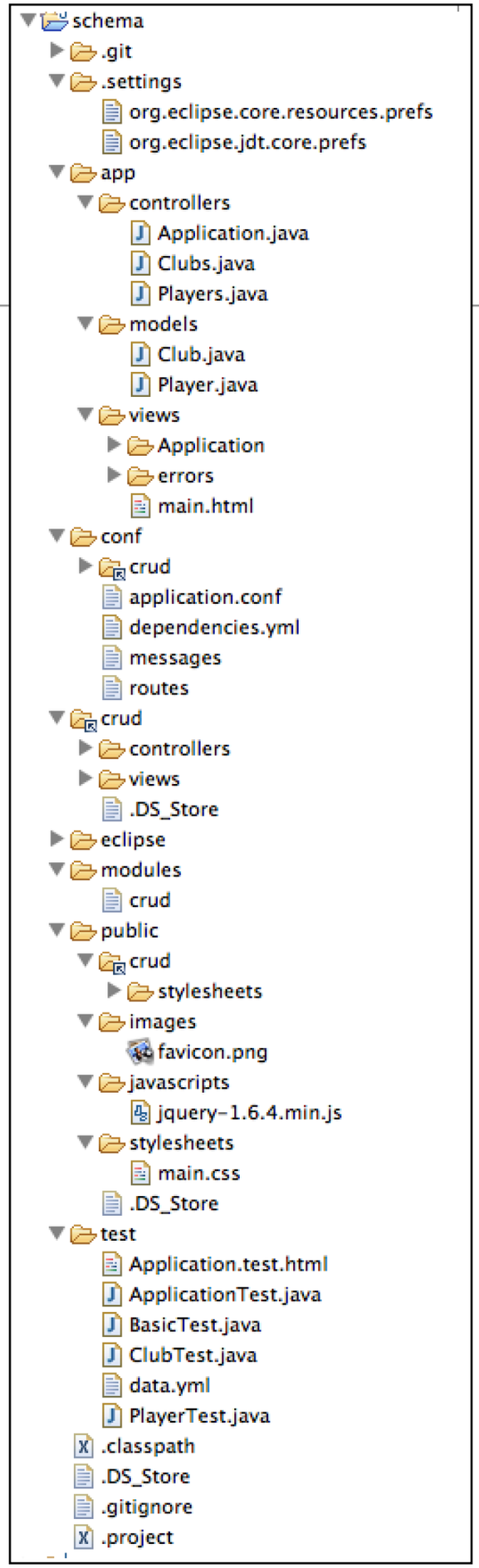
```
play new schema
```

```
cd schema
play eclipsify
```



Step 3 -

- Configure the project to ‘exclude’ generated files
- We do not want the repository to contain ‘generated’ files.
- i.e. Files we did not author and do not need to maintain
- These can be:
  - play generated (cloudbees etc...)
  - Build generated (bin, test etc..
  - Eclipse generated (.settings etc...)

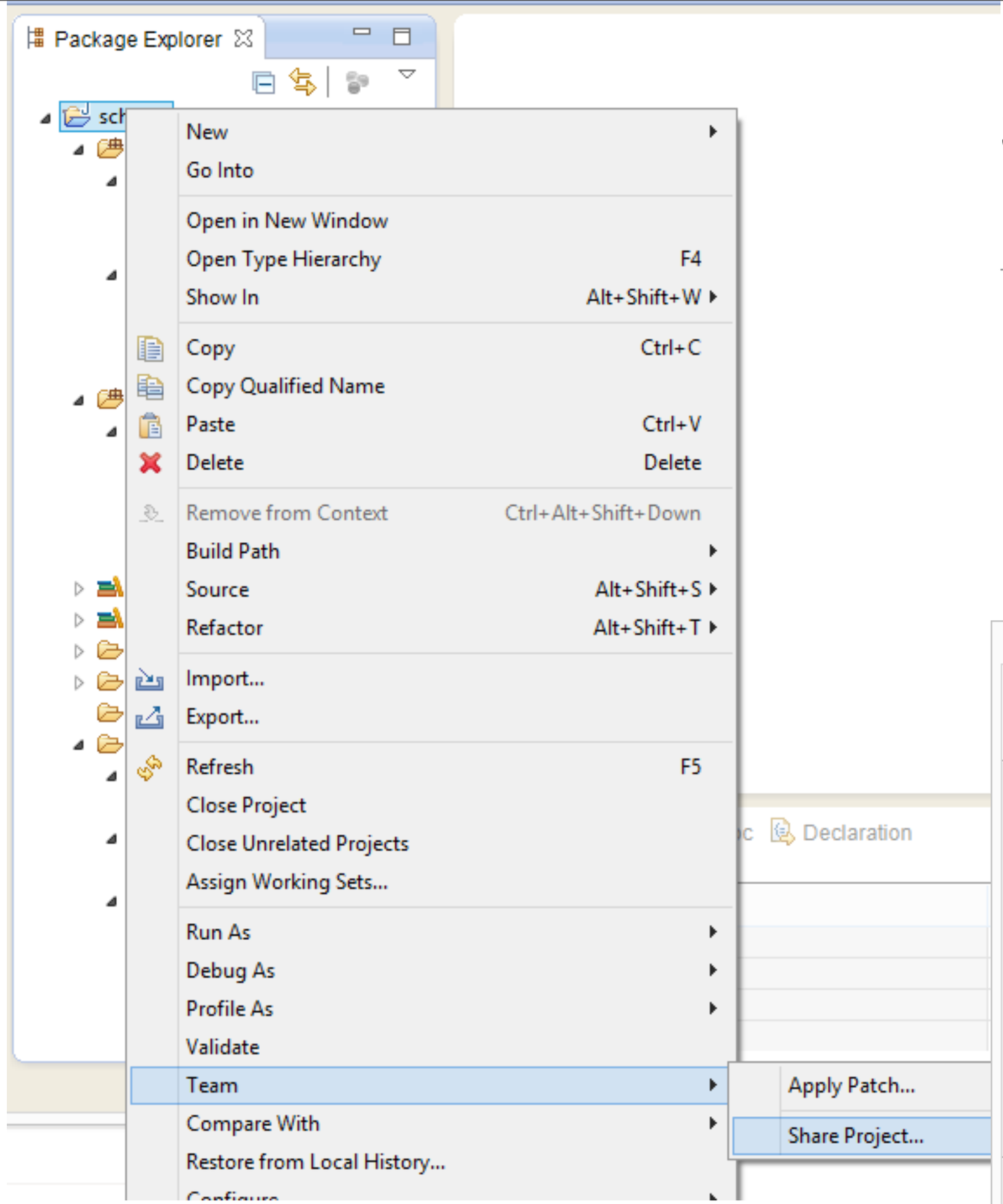


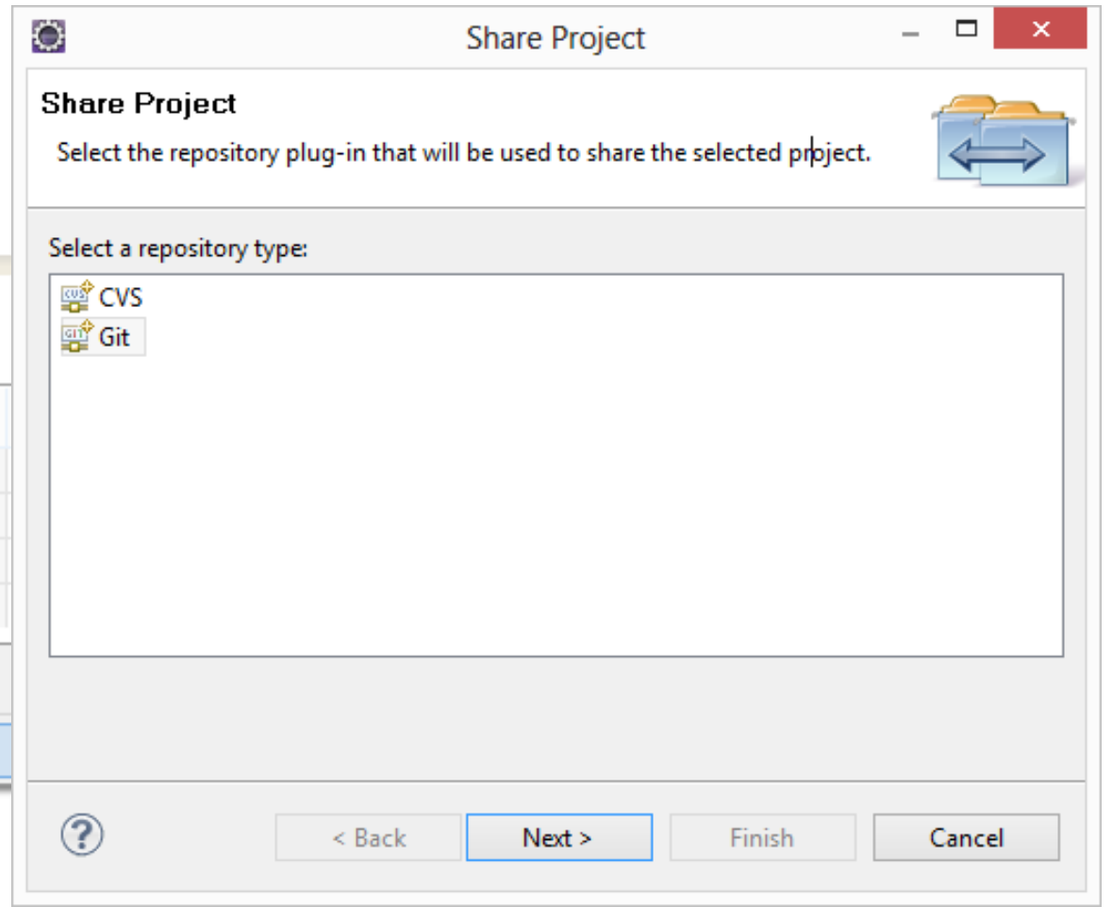
Step 3

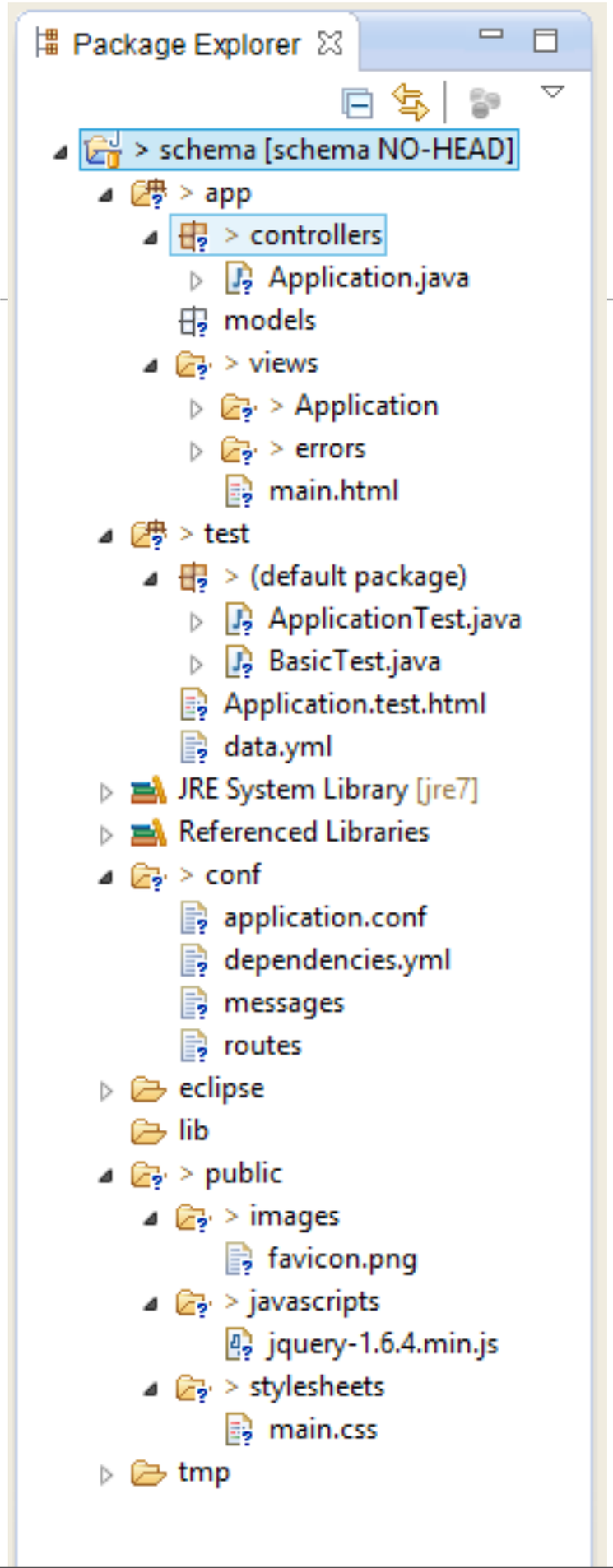
- A single file called ‘.gitignore’ contains patterns describing all of these files.
- Place this file in the root of the “repo”
- Any file in the repo that matches these patterns will be ignored by git

```
# Ignore all dotfiles...
.*
# except for .gitignore
!.gitignore
!.classpath
!.project


# Ignore Play! working directory #
war
db
eclipse
lib
log
logs
modules
precompiled
project/project
project/target
crud/*
data/*
conf/crud/*
conf/cloudbees*
public/crud/*
target
tmp
test-result
server.pid
*.iml
*.eml
```

Step 3 - Create a repository







Step 3 - Create a repository

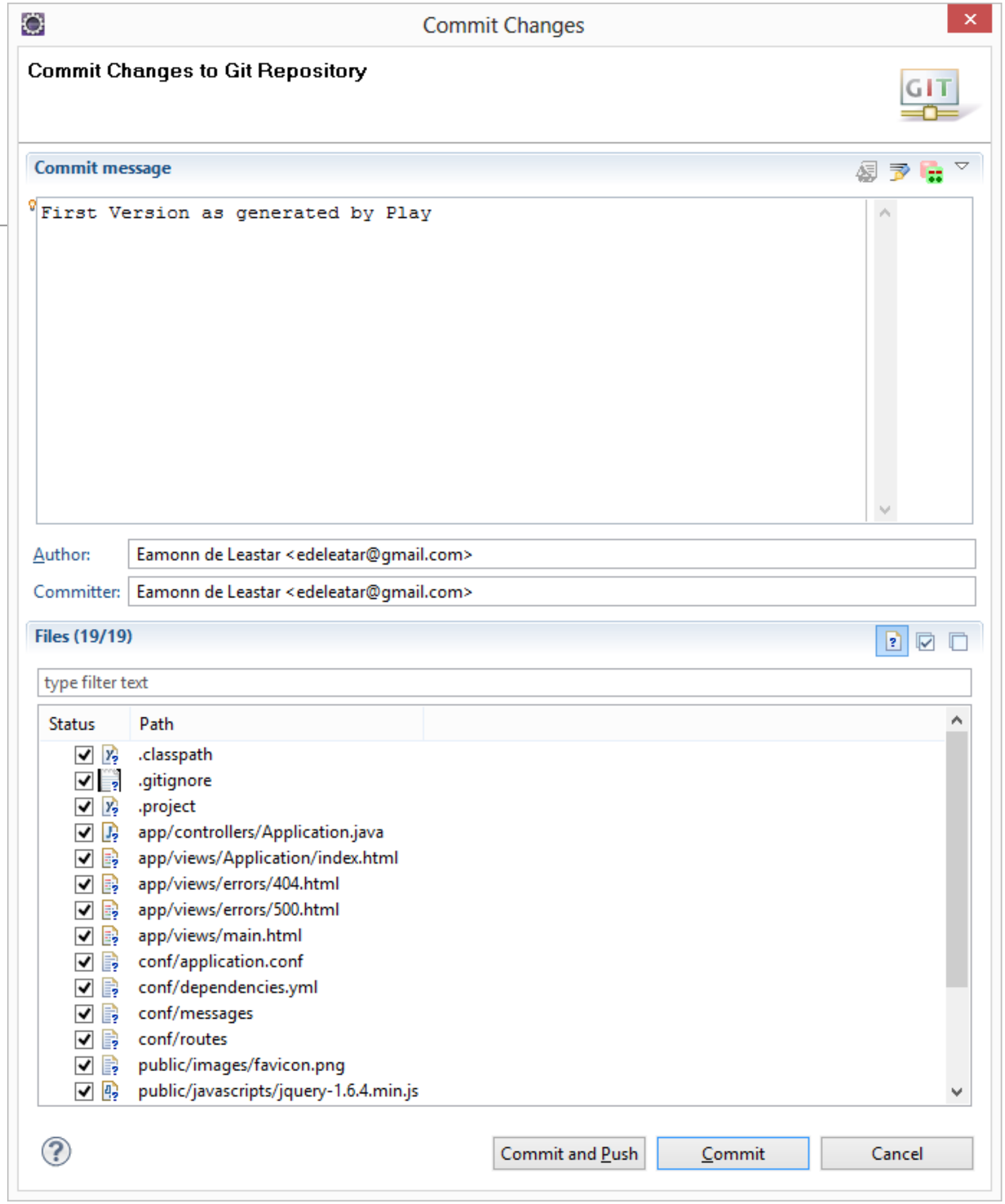
- Workspace is now ‘decorated’ with various icons.
- ‘?’ indicates the following:
  - the file is a candidate for version control (it will not be ‘.gitignored’)

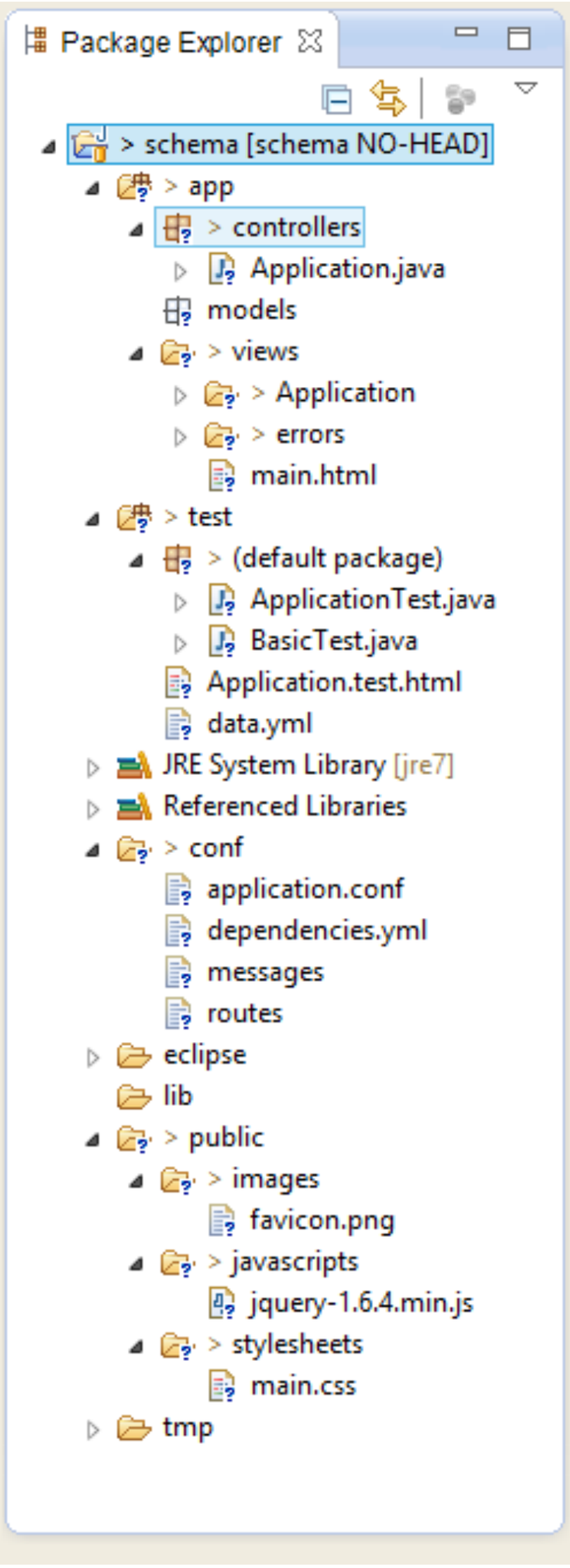
but

  - the file is not currently under control

Step 3 - Commit

- Right click on project, and select ‘Share->Commit’
- Presents presents list of files (only the files which had ‘?’ on them already).
- User enters a “Commit Message”



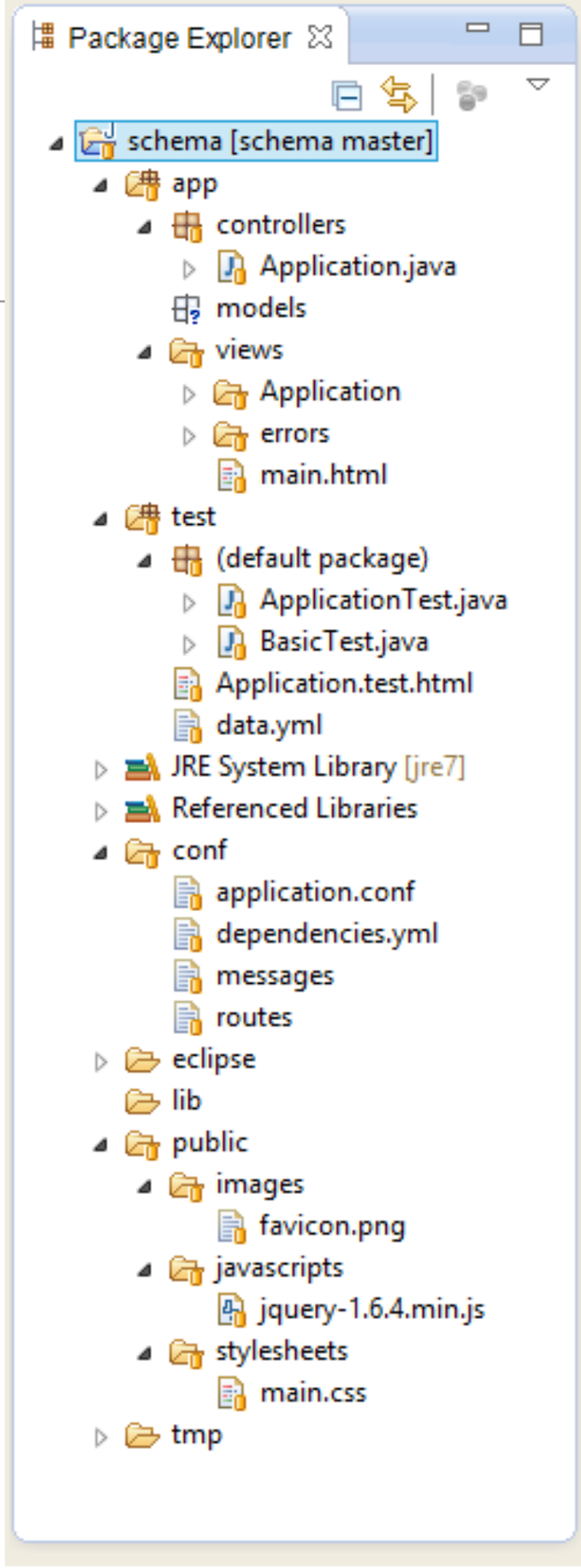


Commit

Icons change

Files under revision control now have another icon indicating they are ‘controlled’

There are a few other icons we will see shortly to reflect different status



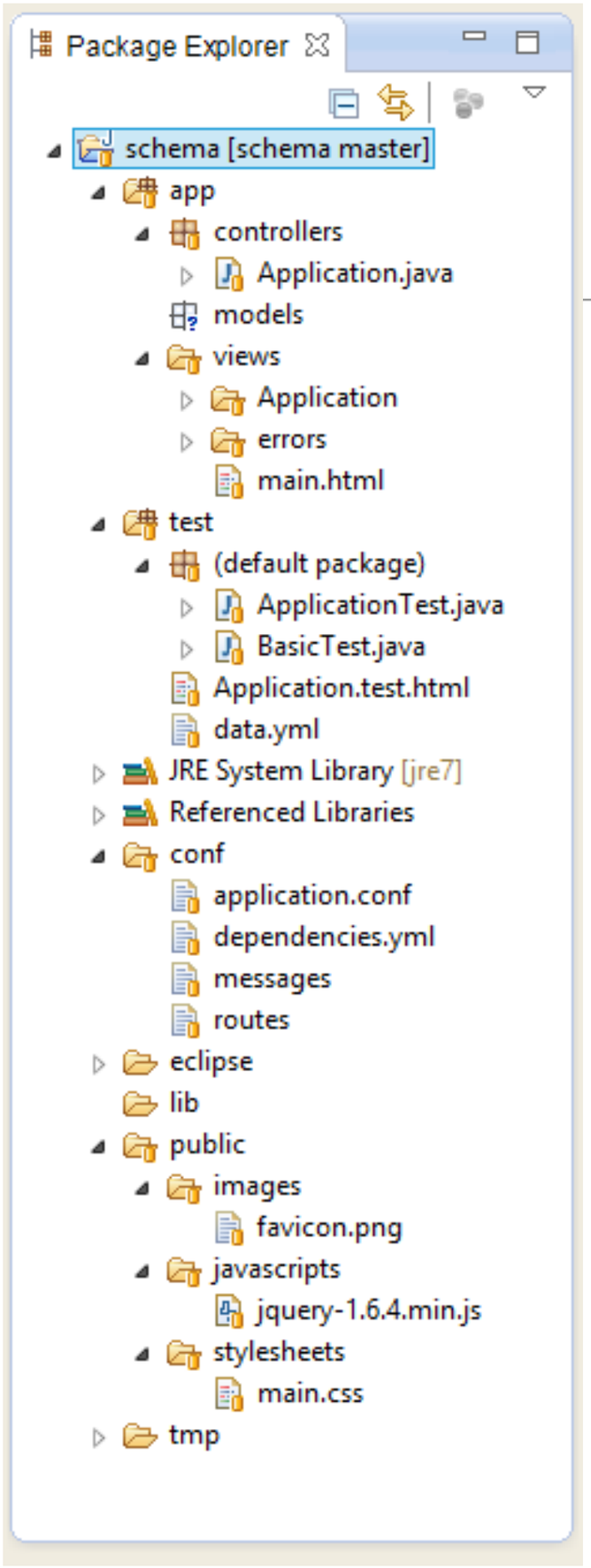
Step 4 - Walking Skeleton

- Introduce Crud
- Introduce standalone Player + Club classes

```
package models;

import javax.persistence.Entity;
import play.db.jpa.Model;

@Entity
public class Player extends Model
{
  public String name;

  public Player(String name)
  {
    this.name = name;
  }

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

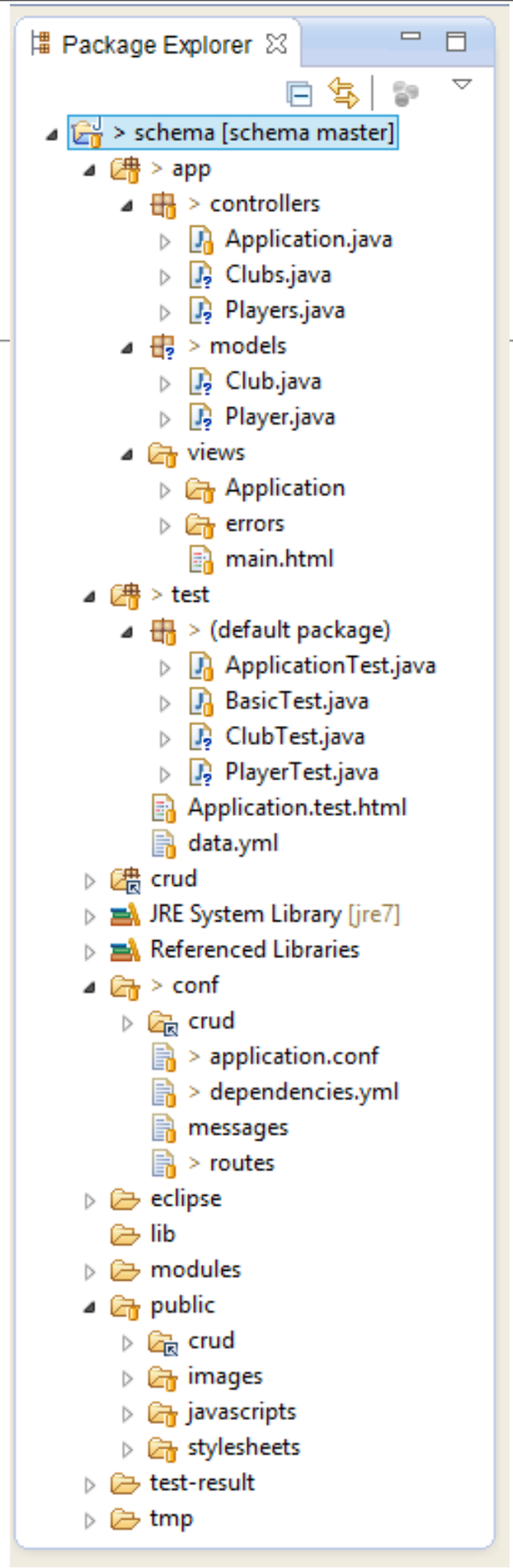
- Introduce Empty Tests
- Run the app, and also explore the model in db browser
- (This is a repeat of lab 02)

```
package models;

import javax.persistence.Entity;
import play.db.jpa.Model;

@Entity
public class Club extends Model
{
  public String name;

  public Club(String name)
  {
    this.name = name;
  }

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



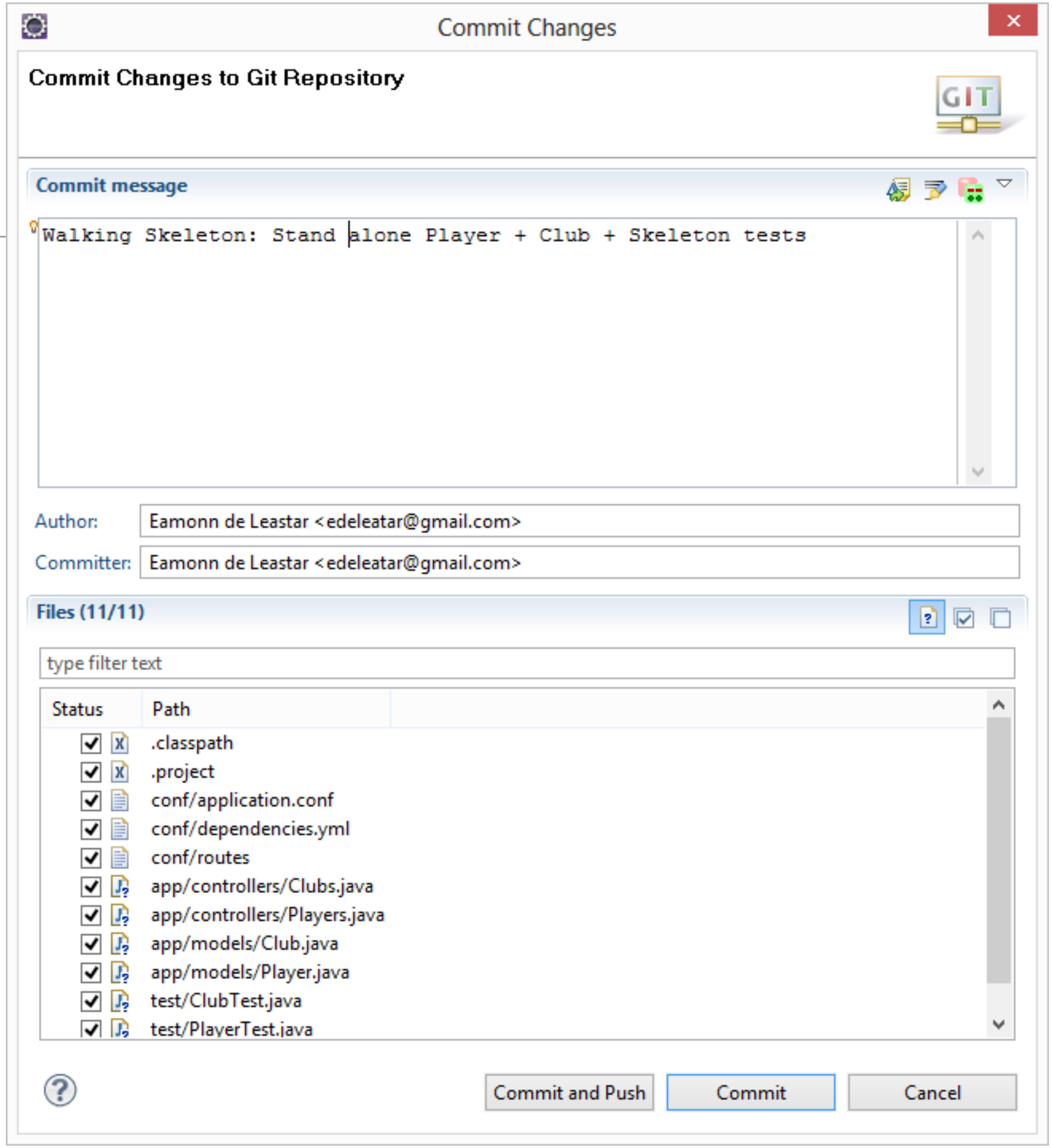
Step 5: Before Commit

- '?' for new files (not yet committed)
- '>' for files already committed, but which have modifications not yet committed



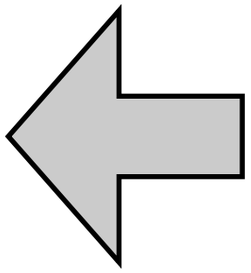
Step 5: Commit

- Commit dialog:
  - list of files (new or modified) to be committed
  - a commit message, which should summarize

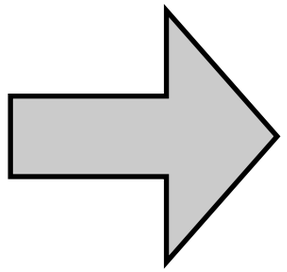


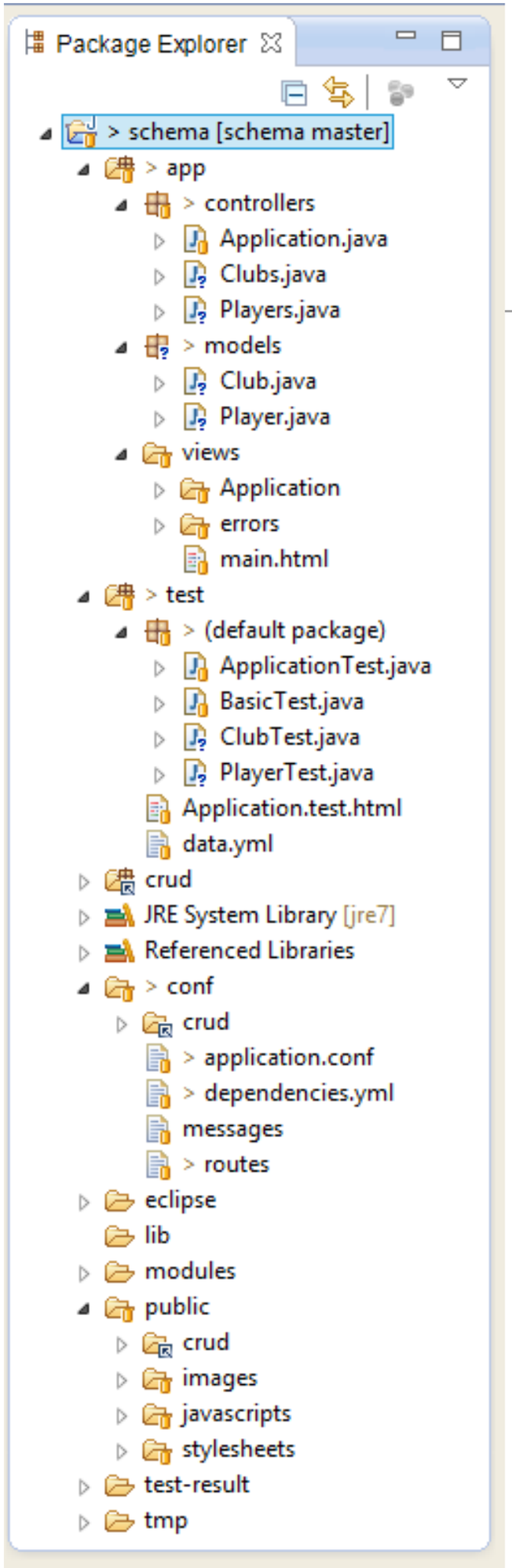
Step 5: After Commit

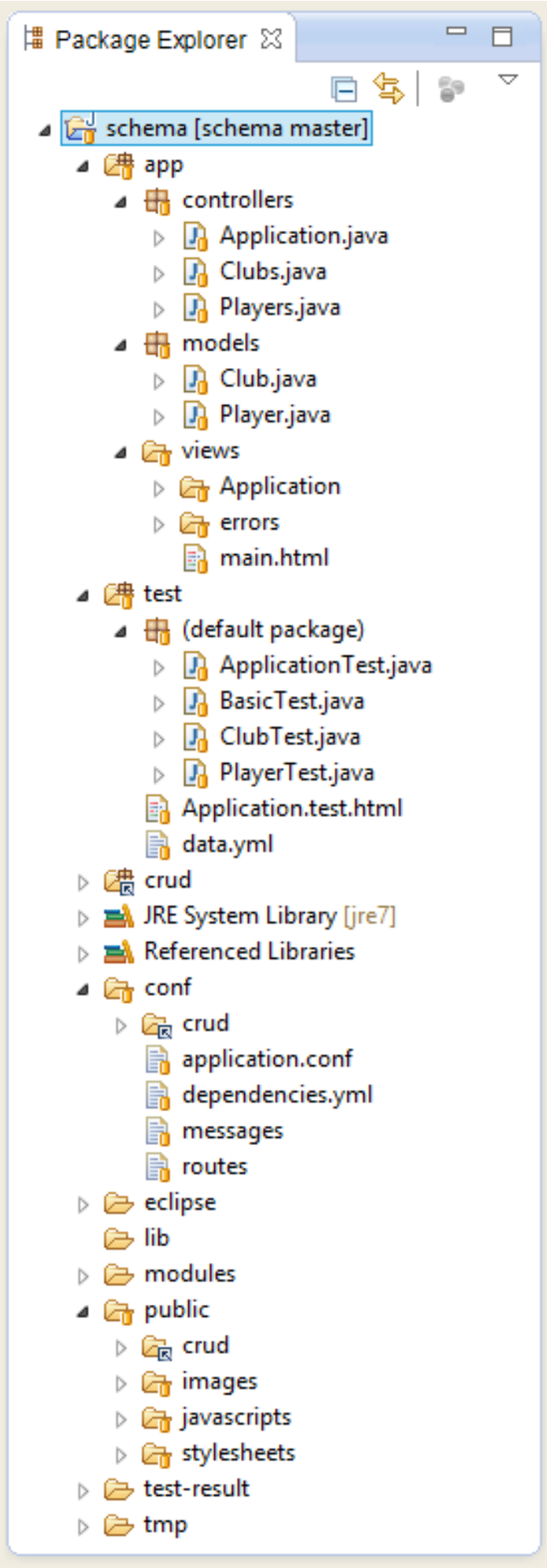
- Changes made but NOT committed



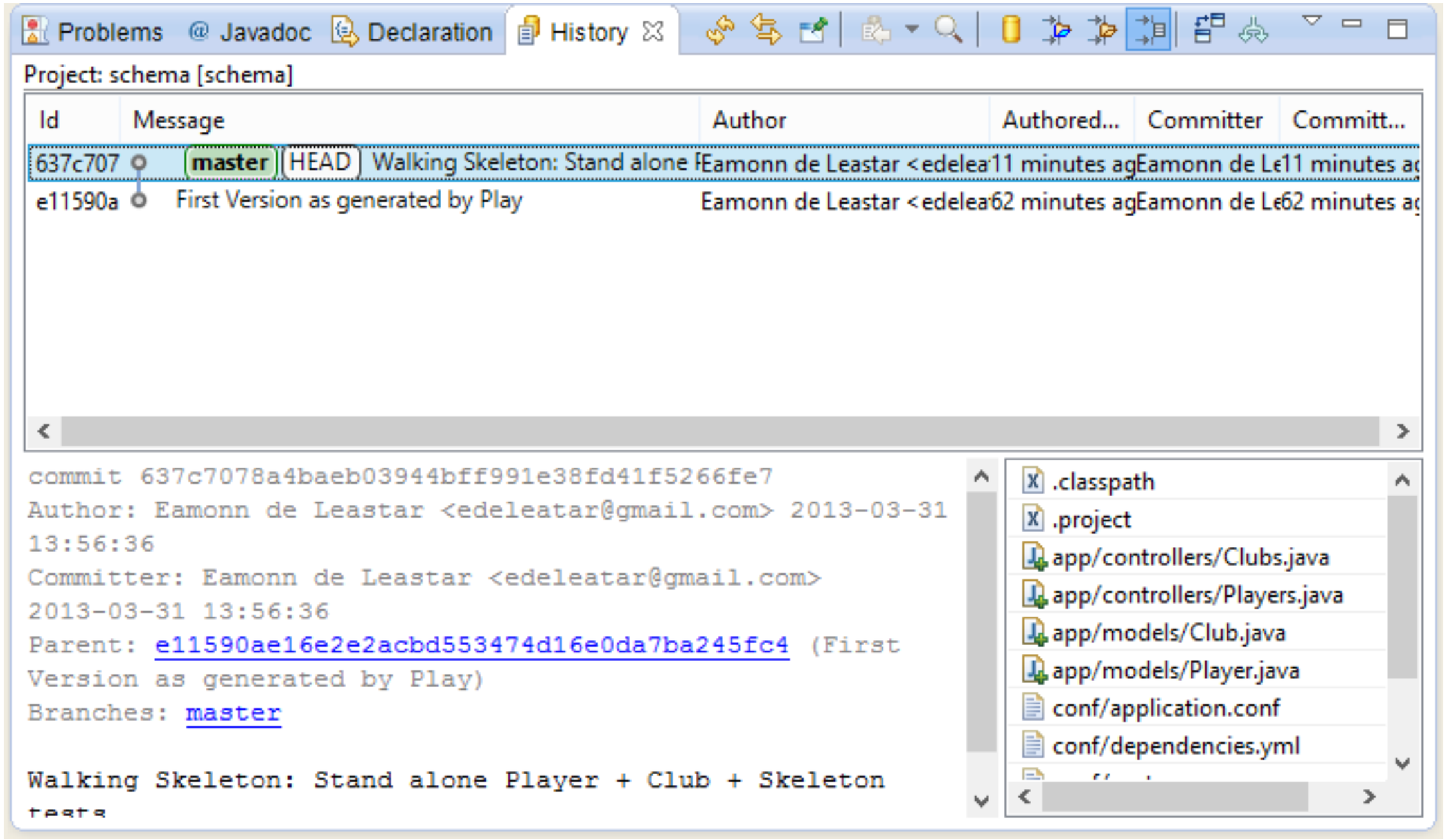
- Changes committed

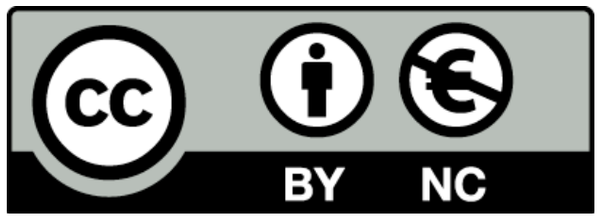






Step 5: Repository History





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 LÁIRGE

eLearning support unit