

App Development & Modelling

Produced
by

Eamonn de Leastar (edelestar@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 LÁIRCE



Repositories - Part 1

Repositories Lab 10 Step Programme!

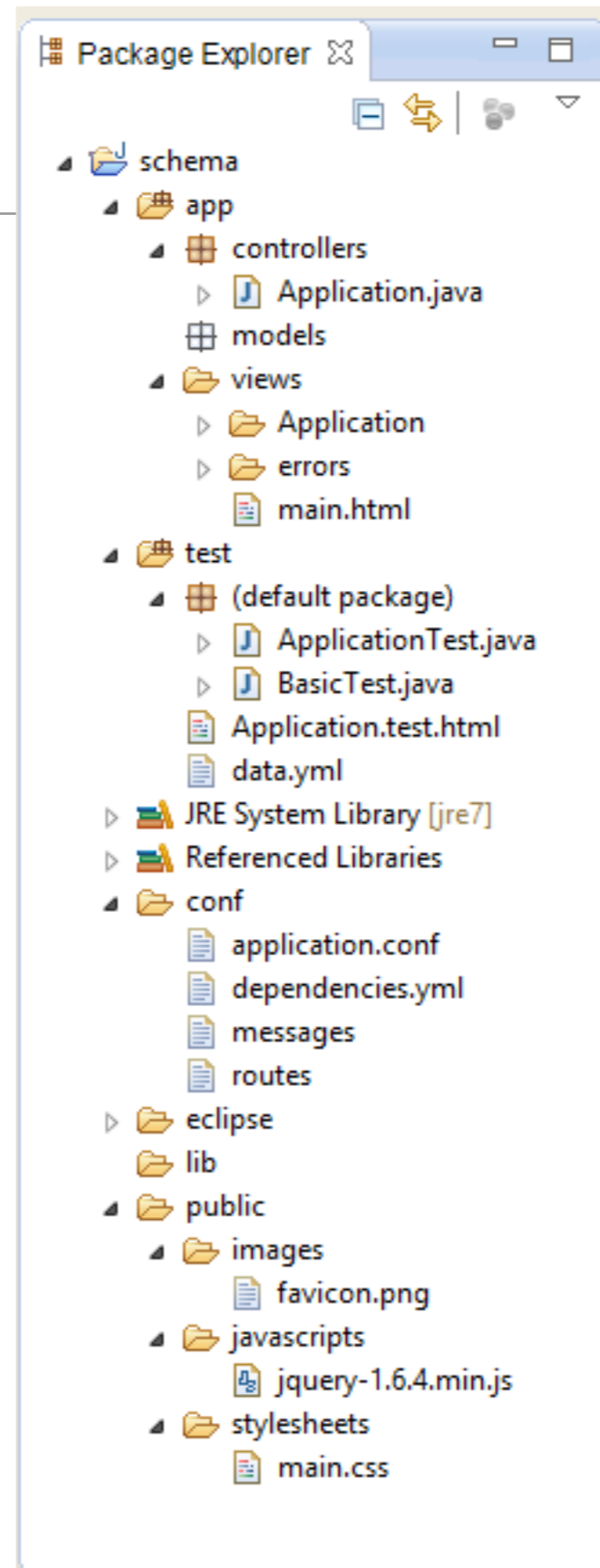
1. Install Git, Sourcetree & Egit
2. Create Schema Project
3. Commit Baseline Schema project to git repository
4. Create 'Walking Skeleton' Version
5. Commit "Walking Skeleton" Version
6. Commit Simple Player & Club Tests
7. Commit Club/Player One to Many Relationship
8. Manipulating Repository using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

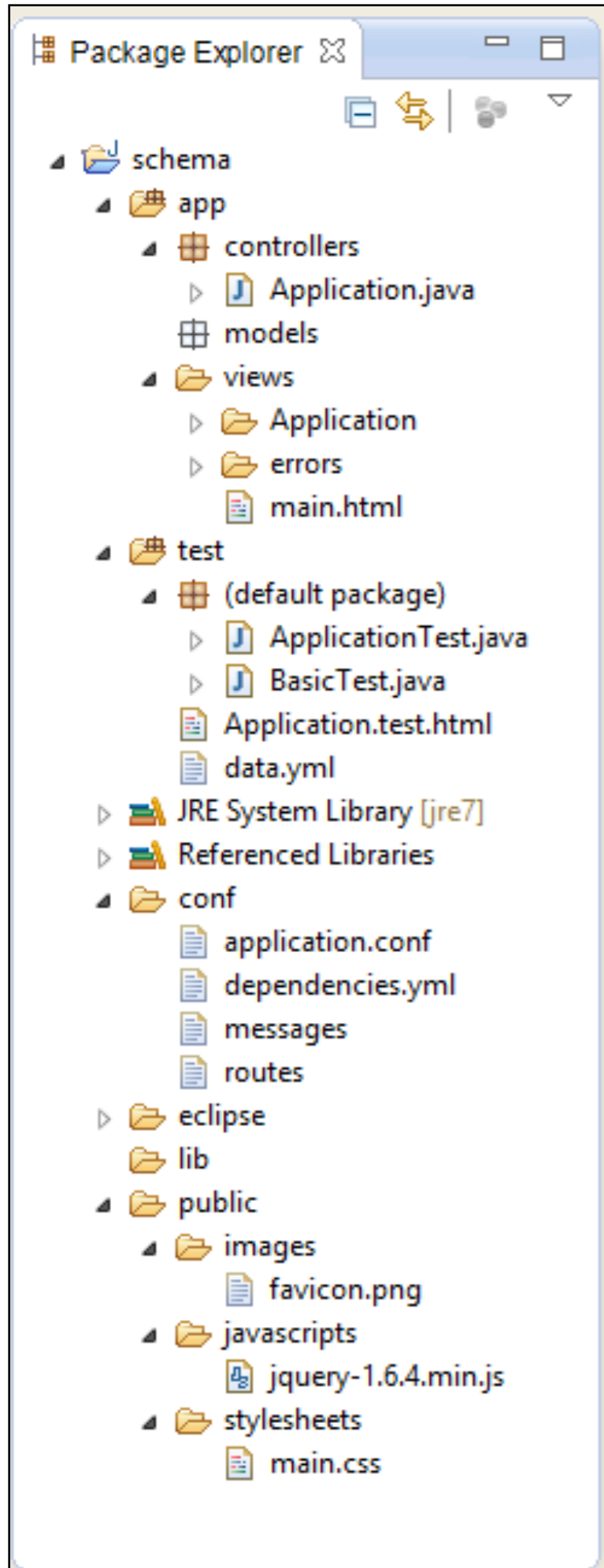
Part 1

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

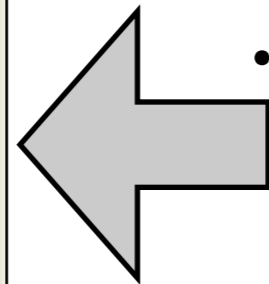
The Need for Repositories

- What, precisely, is in a Play project?



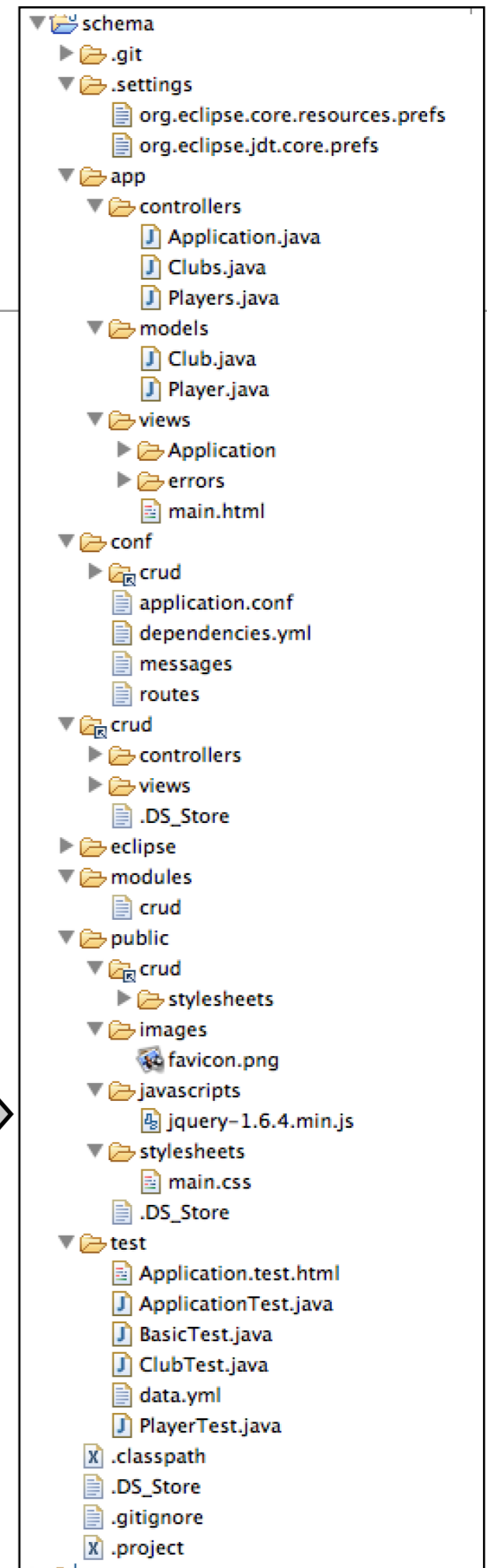
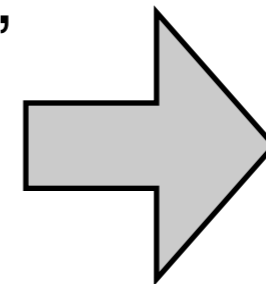


Play Project =
collection of files in
a folder



- 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 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.



 **Mac GUIs**

 **Tarballs**

 **Windows Build**

 **Source Code**



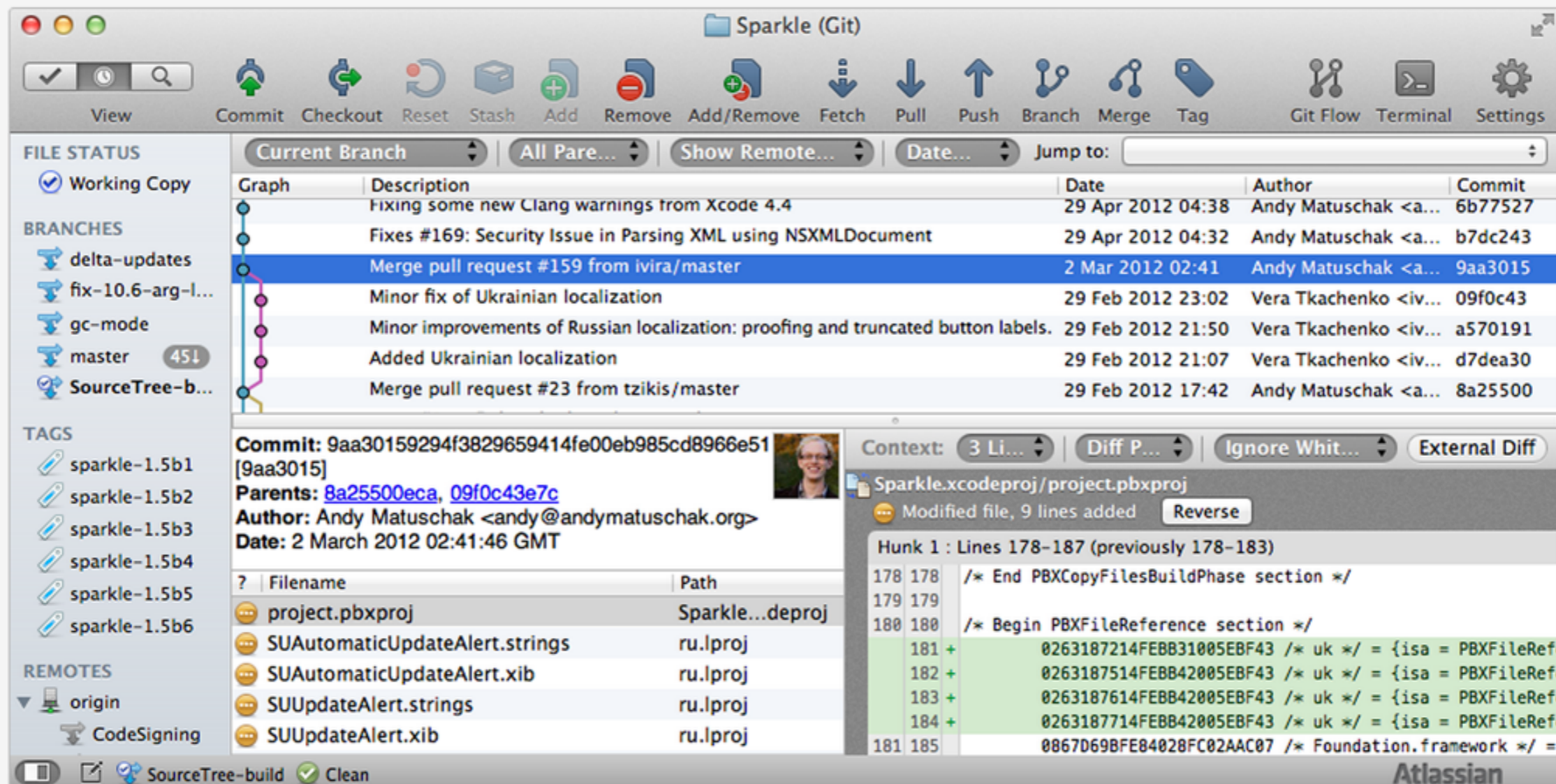
Pro Git by Scott Chacon is available to **read online for free**. Dead tree versions are available on Amazon.com.

A free Git & Mercurial client for Windows or Mac.

Download SourceTree Free

Mac OS X 10.6+

Download SourceTree for Windows 7+



The screenshot shows the SourceTree application interface for a project named 'Sparkle (Git)'. The interface includes a toolbar with various actions like View, Commit, Checkout, Reset, Stash, Add, Remove, Add/Remove, Fetch, Pull, Push, Branch, Merge, Tag, Git Flow, Terminal, and Settings. The main area is divided into several sections:

- FILE STATUS:** Shows 'Working Copy'.
- BRANCHES:** Lists branches like 'delta-updates', 'fix-10.6-arg-l...', 'gc-mode', 'master' (45L), and 'SourceTree-b...'.
- TAGS:** Lists tags from 'sparkle-1.5b1' to 'sparkle-1.5b6'.
- REMITES:** Shows 'origin' and 'CodeSigning'.
- Commit History Table:**

Graph	Description	Date	Author	Commit
	Fixing some new Clang warnings from Xcode 4.4	29 Apr 2012 04:38	Andy Matuschak <a...>	6b77527
	Fixes #169: Security Issue in Parsing XML using NSXMLDocument	29 Apr 2012 04:32	Andy Matuschak <a...>	b7dc243
	Merge pull request #159 from ivira/master	2 Mar 2012 02:41	Andy Matuschak <a...>	9aa3015
	Minor fix of Ukrainian localization	29 Feb 2012 23:02	Vera Tkachenko <iv...>	09f0c43
	Minor improvements of Russian localization: proofing and truncated button labels.	29 Feb 2012 21:50	Vera Tkachenko <iv...>	a570191
	Added Ukrainian localization	29 Feb 2012 21:07	Vera Tkachenko <iv...>	d7dea30
	Merge pull request #23 from tzikis/master	29 Feb 2012 17:42	Andy Matuschak <a...>	8a25500
- Commit Details:**

Commit: 9aa30159294f3829659414fe00eb985cd8966e51 [9aa3015]
Parents: 8a25500eca, 09f0c43e7c
Author: Andy Matuschak <andy@andymatuschak.org>
Date: 2 March 2012 02:41:46 GMT
- Diff View:** Shows a diff for 'Sparkle.xcodeproj/project.pbxproj'. The diff highlights a hunk with 9 lines added, including localization strings for Ukrainian and Russian.



EGit



Download

Eclipse Distribution,
Update Site, Dropins



Documentation

Tutorials, Examples,
Videos, Online Referen

EGit

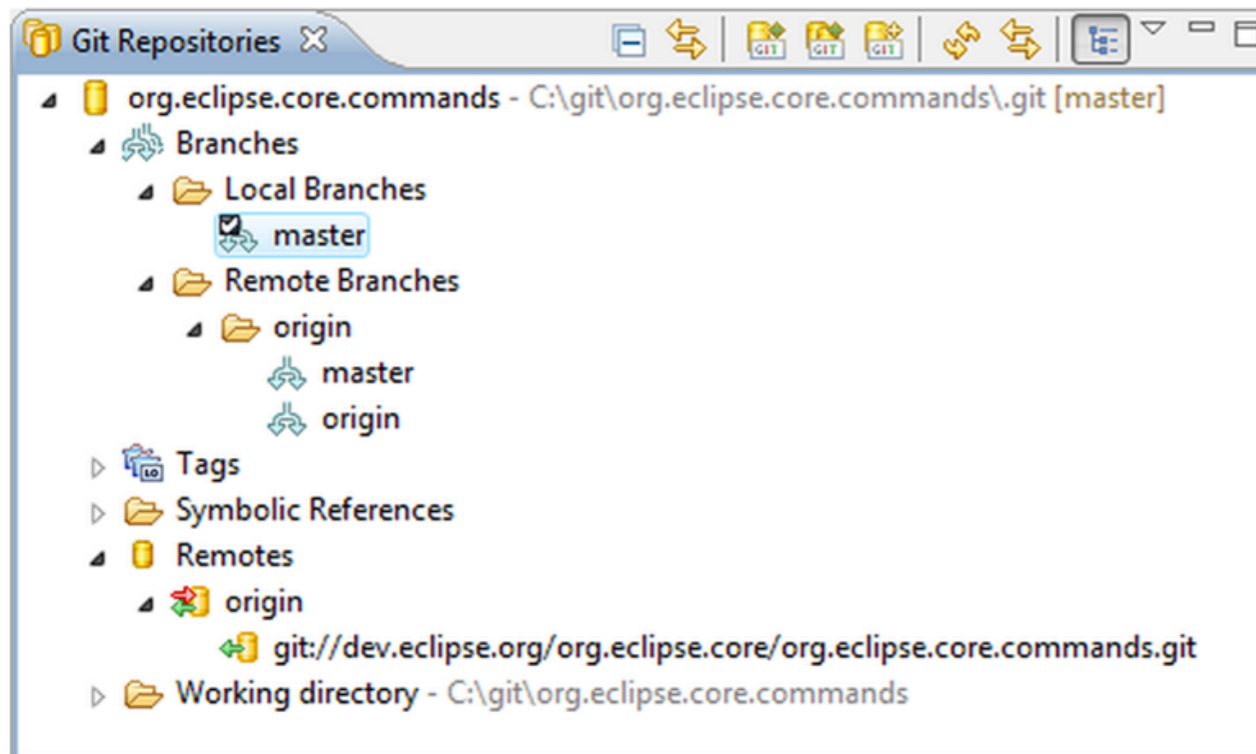


Favorite this @
Eclipse
Marketplace

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.

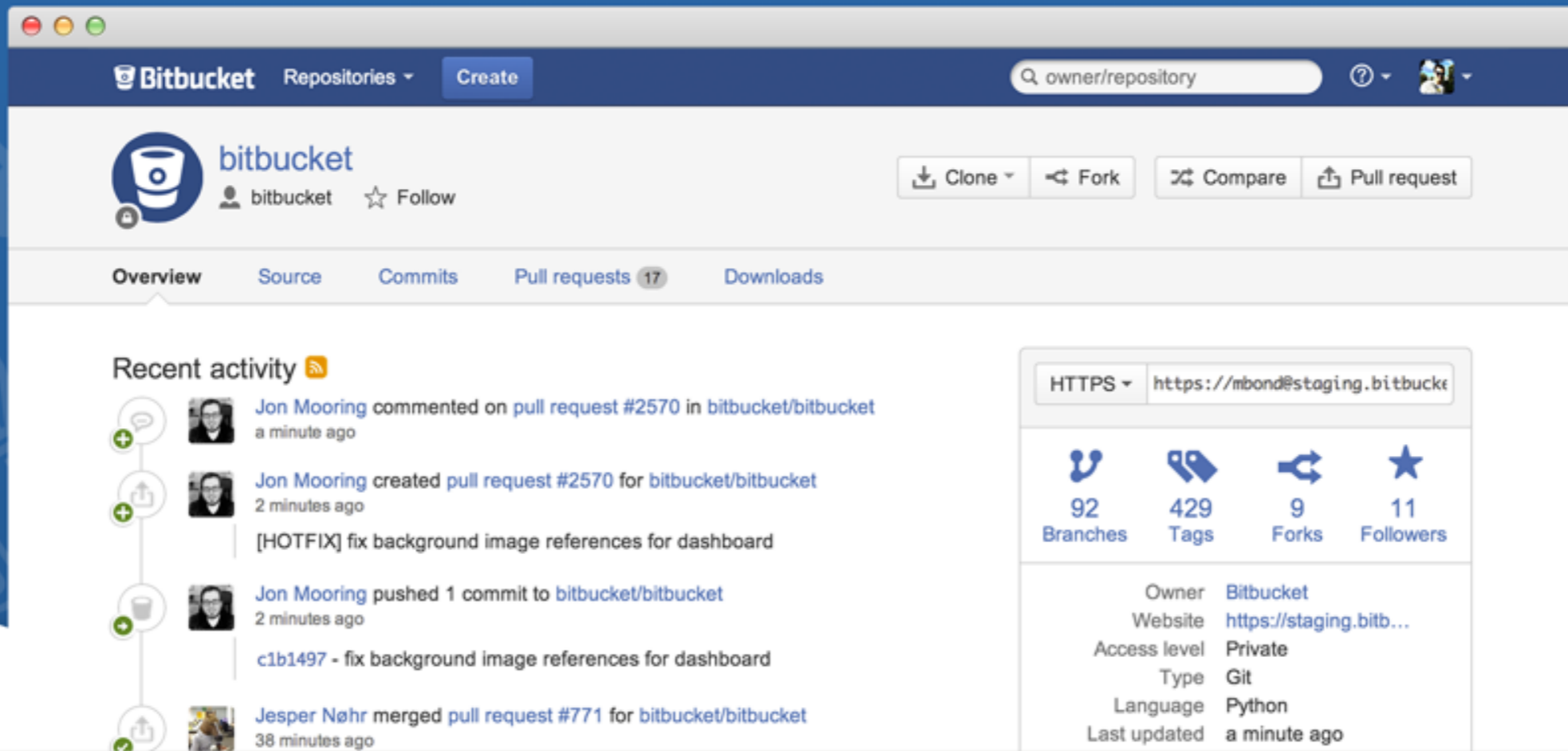
The EGit project is implementing Eclipse tooling on top of the **JGit** Java implementation of Git.



Unlimited private repositories

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

Sign up for free



The screenshot shows the Bitbucket interface for the repository 'bitbucket/bitbucket'. At the top, there's a navigation bar with 'Bitbucket', 'Repositories', and a 'Create' button. A search bar contains 'owner/repository'. Below the navigation, the repository name 'bitbucket' is displayed with a 'Follow' button. Action buttons for 'Clone', 'Fork', 'Compare', and 'Pull request' are visible. A tabbed interface shows 'Overview' selected, with other tabs for 'Source', 'Commits', 'Pull requests (17)', and 'Downloads'. The 'Recent activity' section lists several events: Jon Mooring commented on pull request #2570, Jon Mooring created pull request #2570 with the title '[HOTFIX] fix background image references for dashboard', Jon Mooring pushed 1 commit with the title 'c1b1497 - fix background image references for dashboard', and Jesper Nøhr merged pull request #771. On the right, a sidebar shows the repository's HTTPS URL and statistics: 92 Branches, 429 Tags, 9 Forks, and 11 Followers. Below the statistics, a table lists repository details: Owner (Bitbucket), Website (https://staging.bitb...), Access level (Private), Type (Git), Language (Python), and Last updated (a minute ago).



Repositories Lab - Step 1

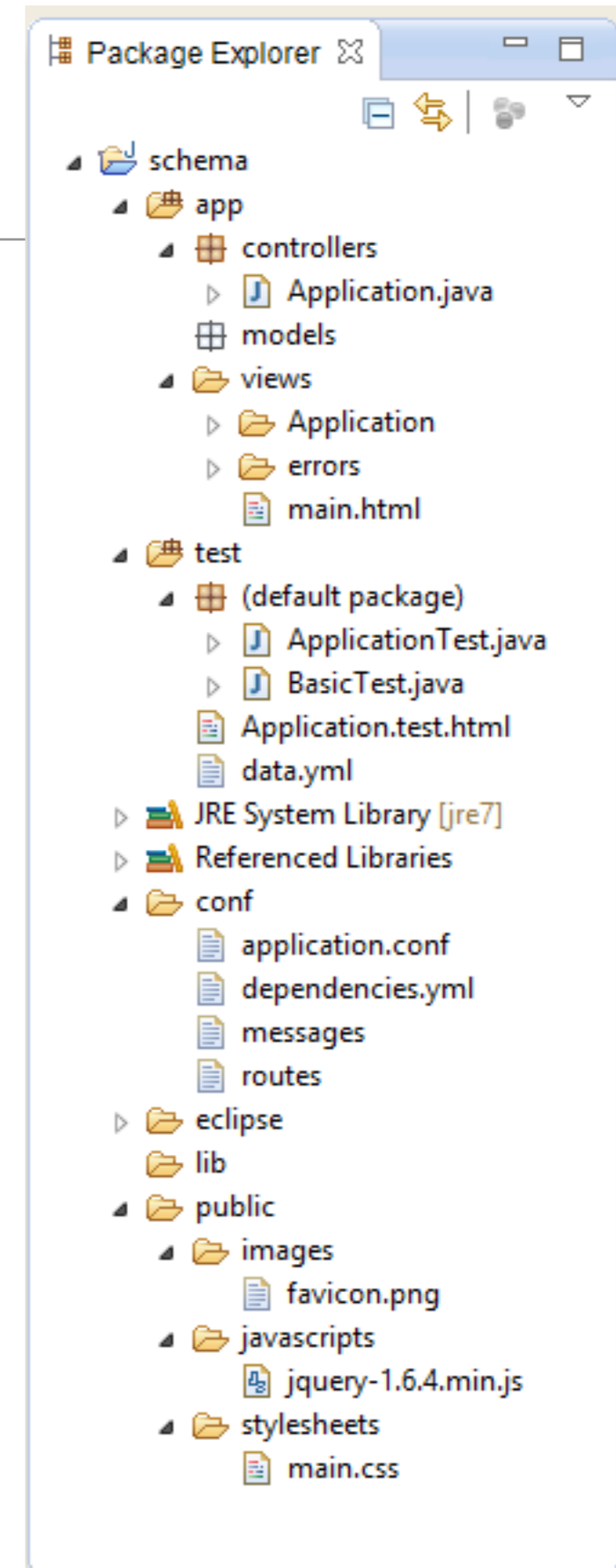
- Install git
- Install SourceTree
- Install Egit
- Create a free account on bitbucket.com

Step 2

- Create a new Play Project called “Schema”

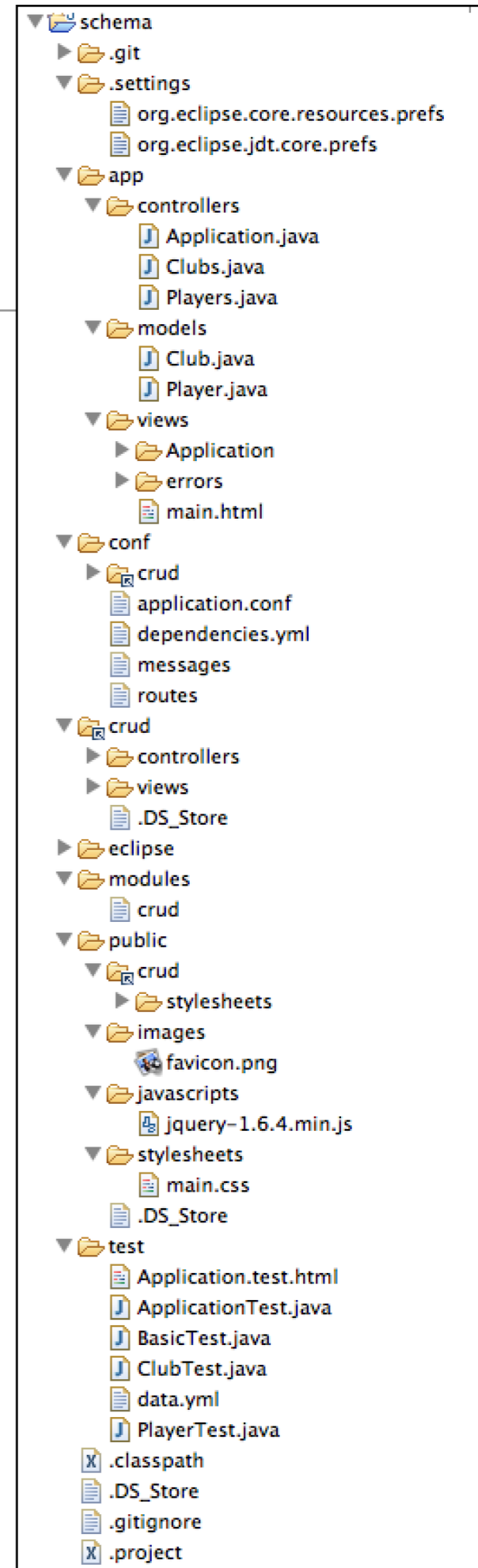
```
play new schema
```

```
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...)



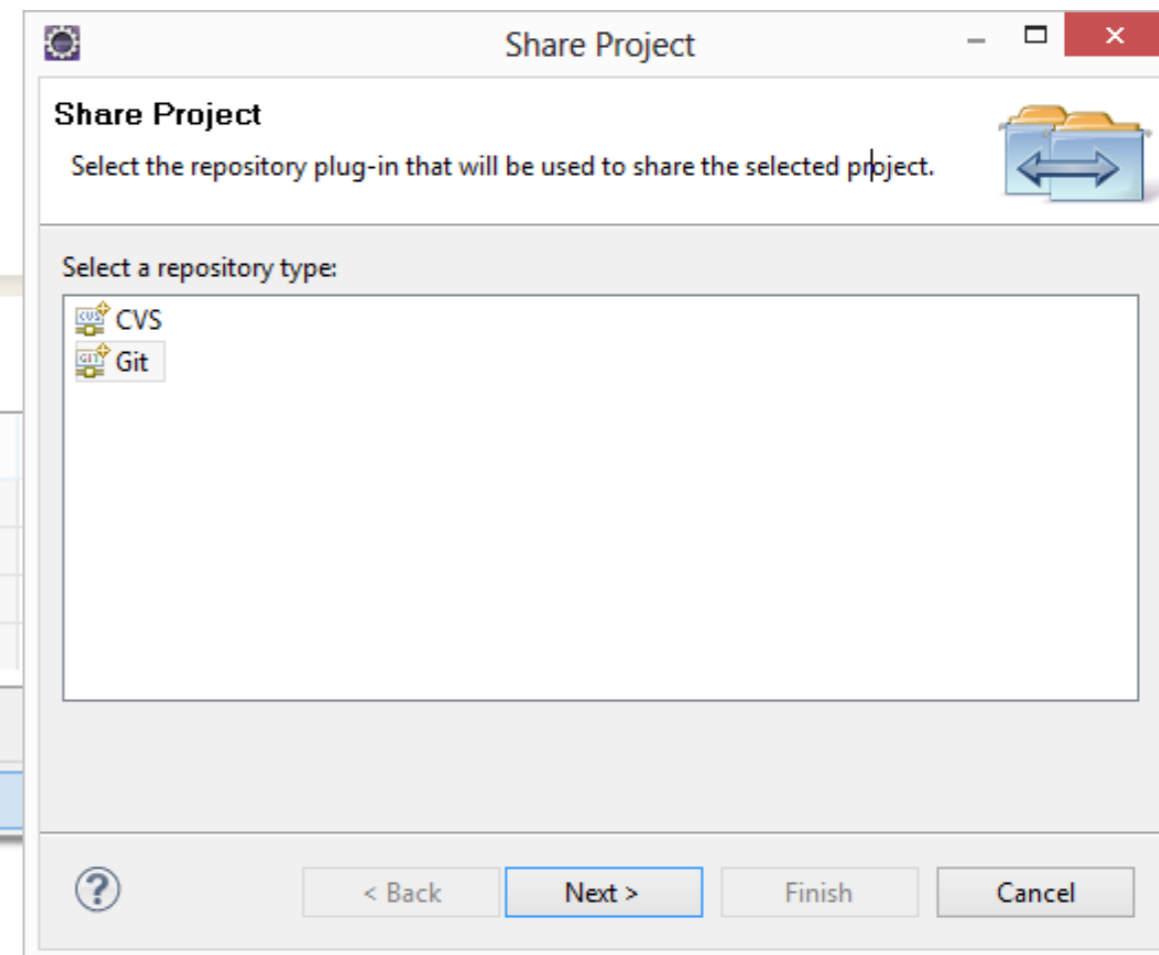
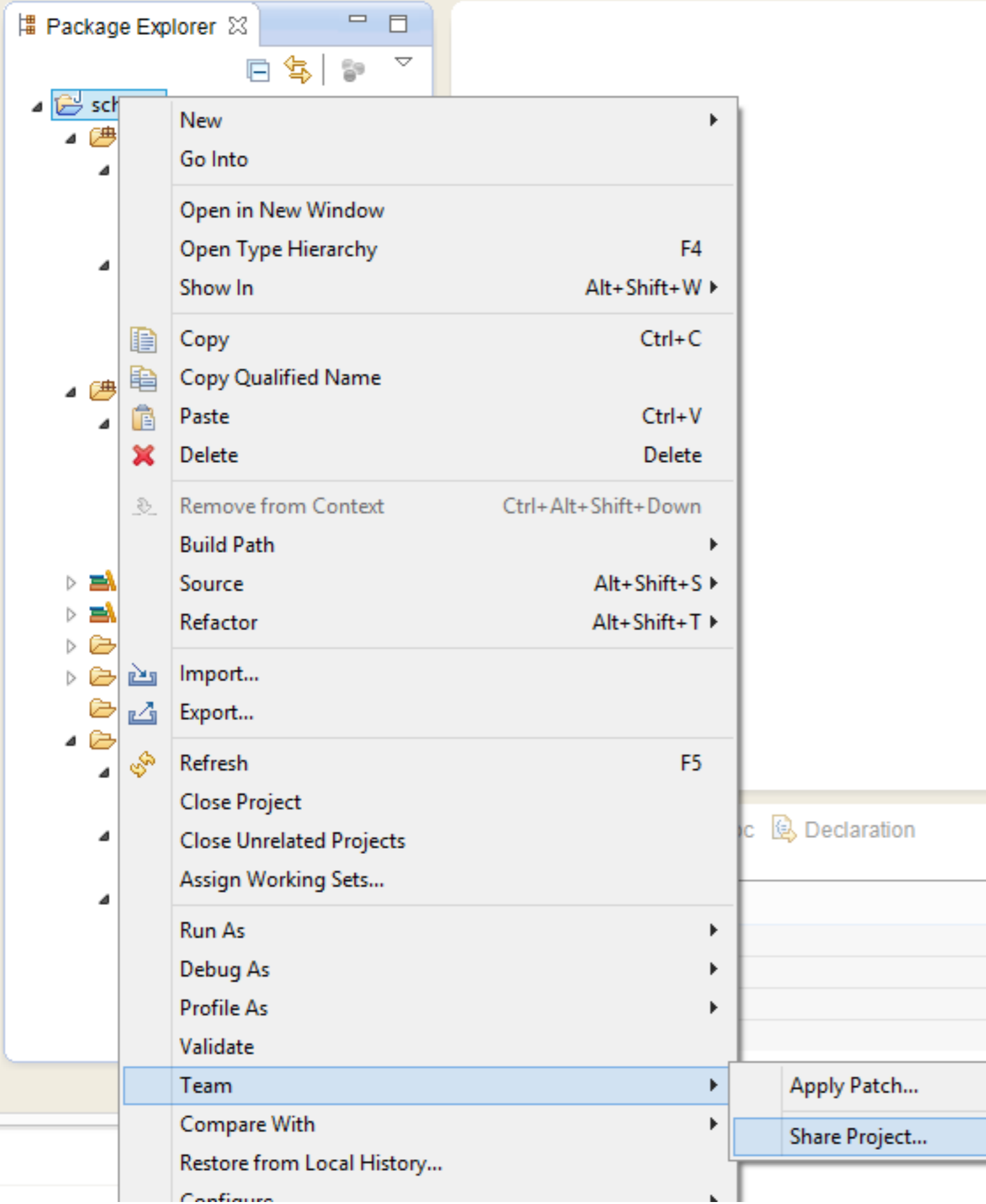
Step 3

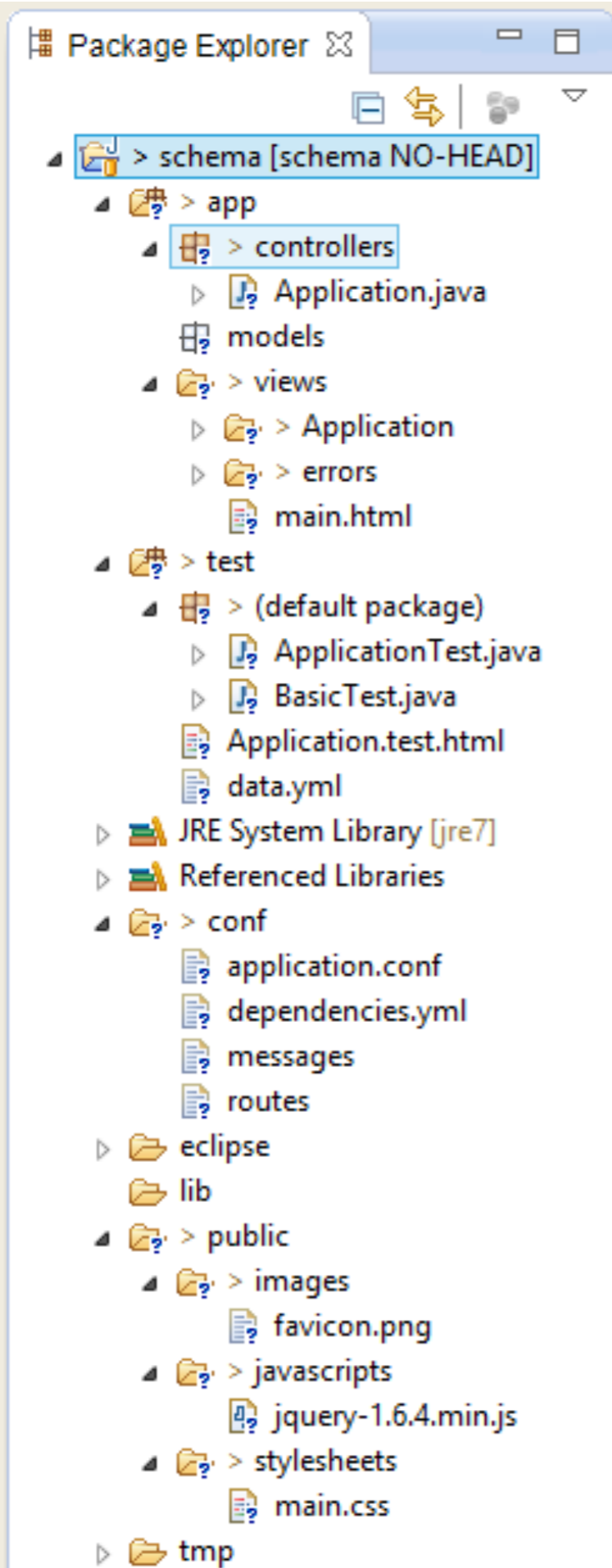
- A single file called ‘.gitignore’ contains patterns describing all of these files.
- Place this file in the root of the “repo”
- Any file in the repo that matches these patterns will be ignored by git

```
# 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

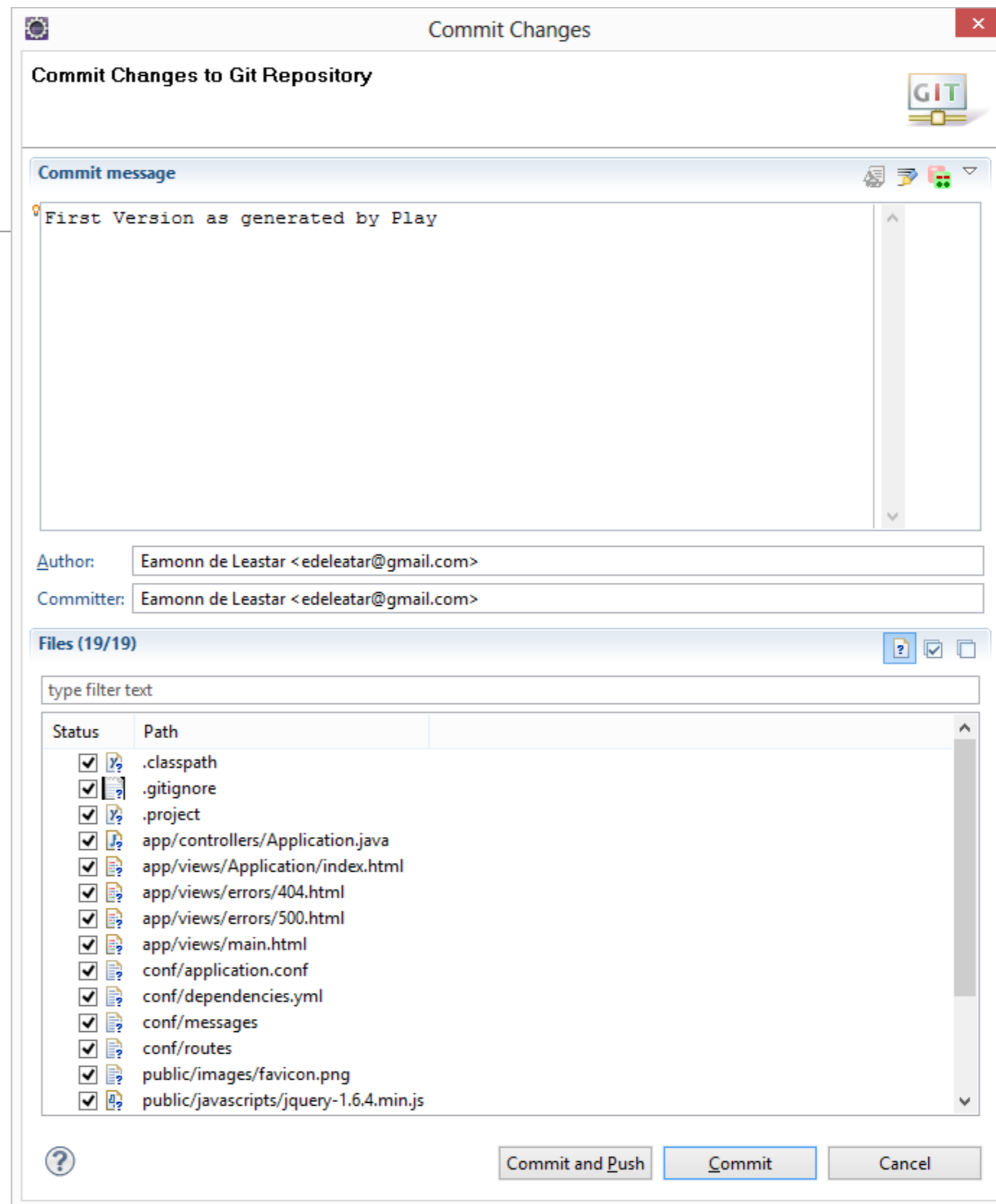
- Workspace is now ‘decorated’ with various icons.
- ‘?’ indicates the following:
 - the file is a candidate for version control (it will not be ‘.gitignored’)

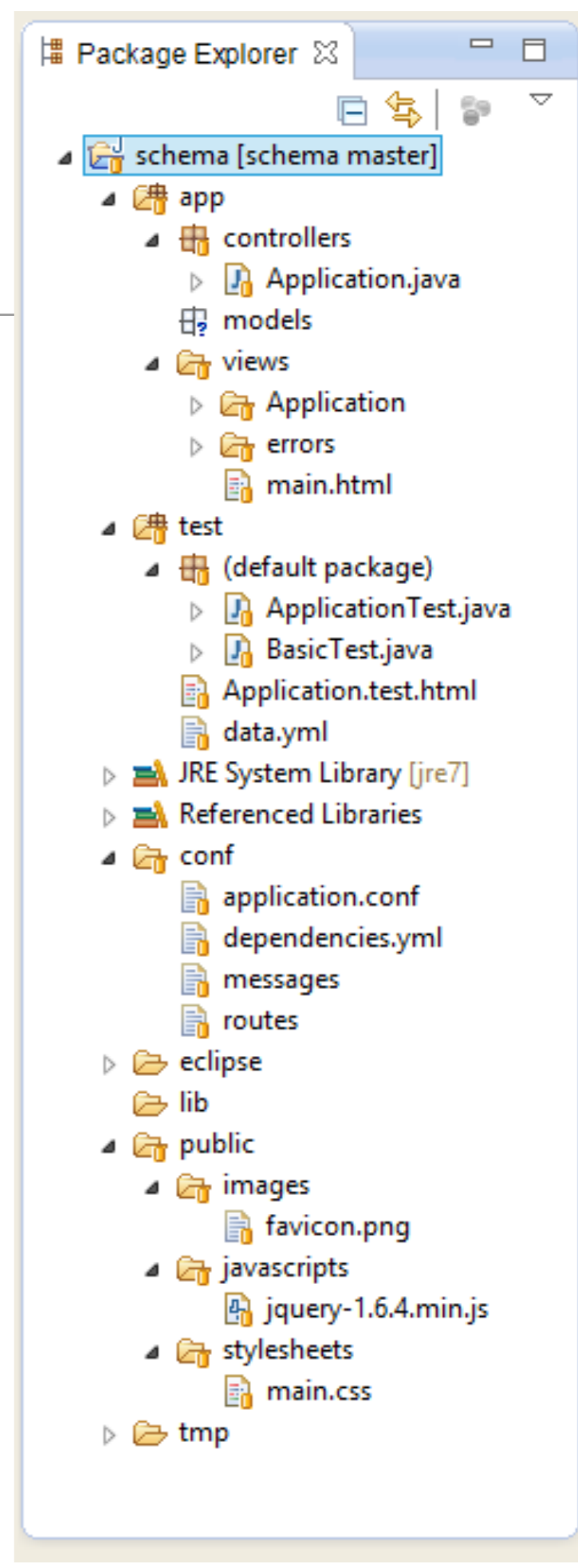
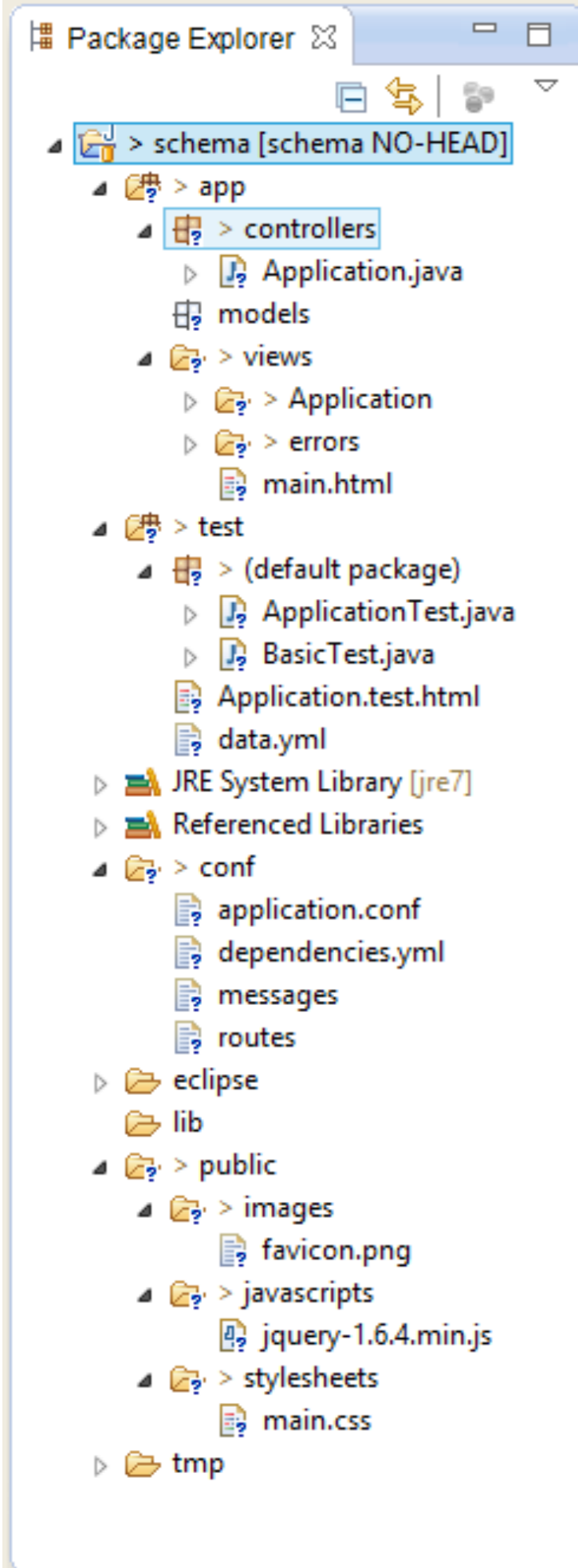
but

- 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"





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 standalone Player + Club classes
- 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 Player extends Model
{
    public String name;

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

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

```
package models;

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

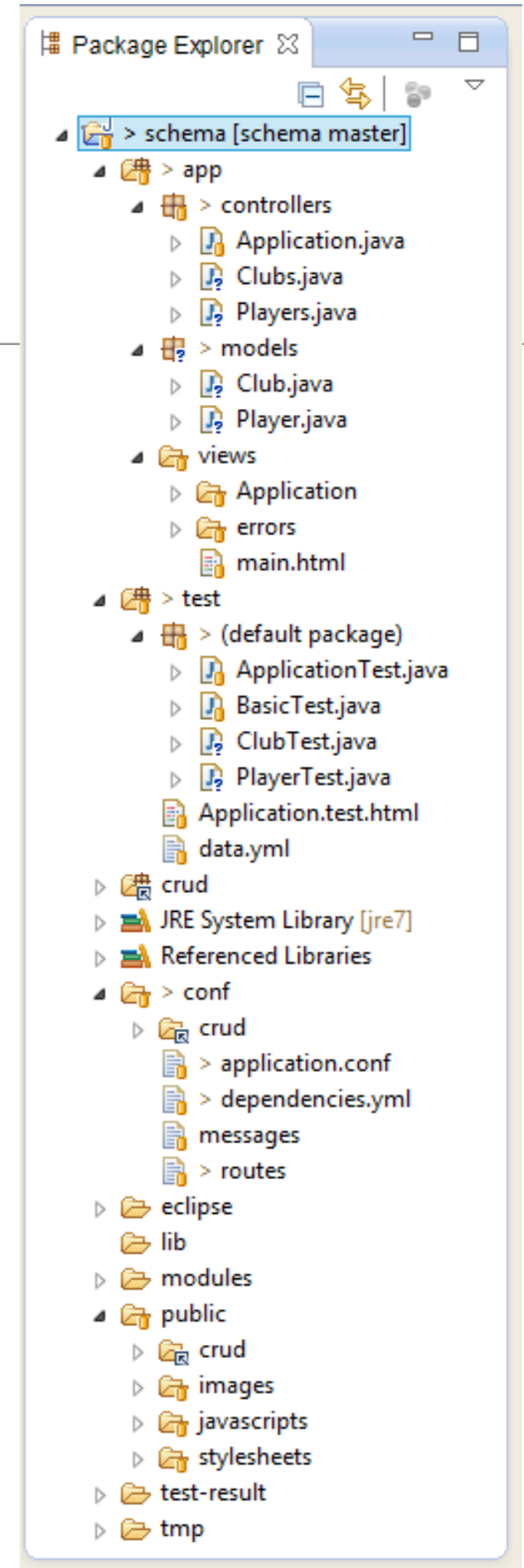
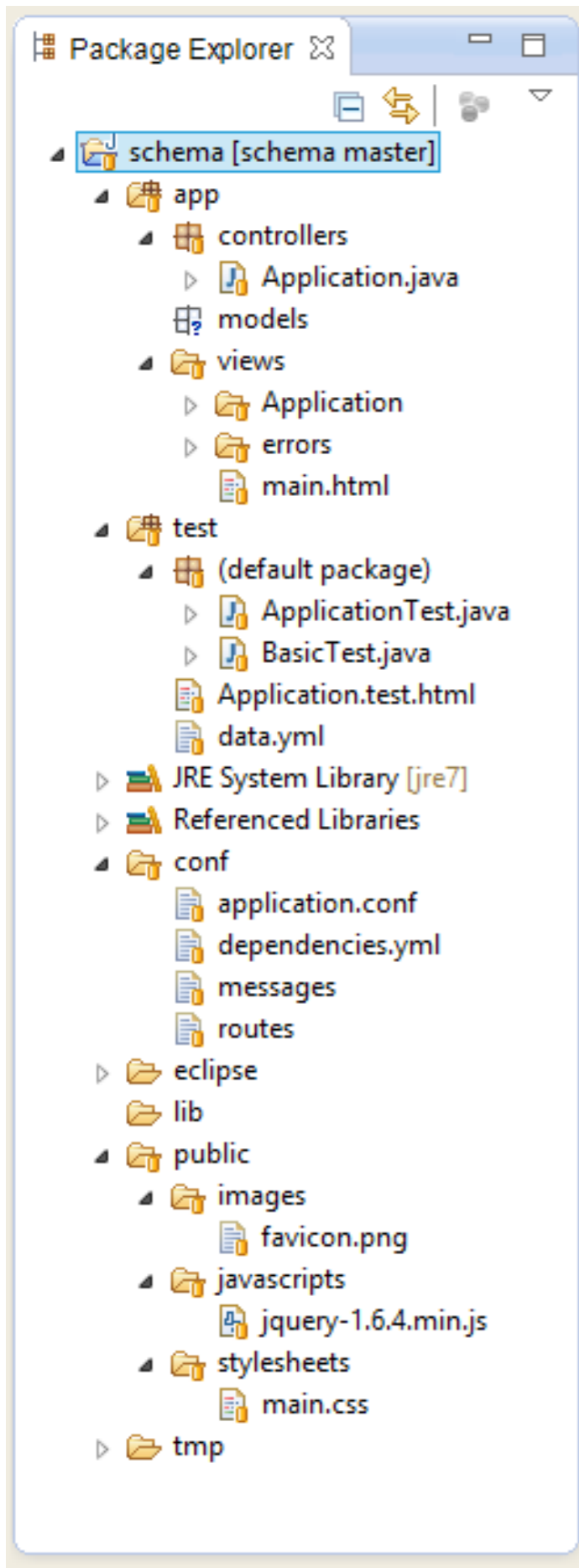
@Entity
public class Club extends Model
{
    public String name;

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

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

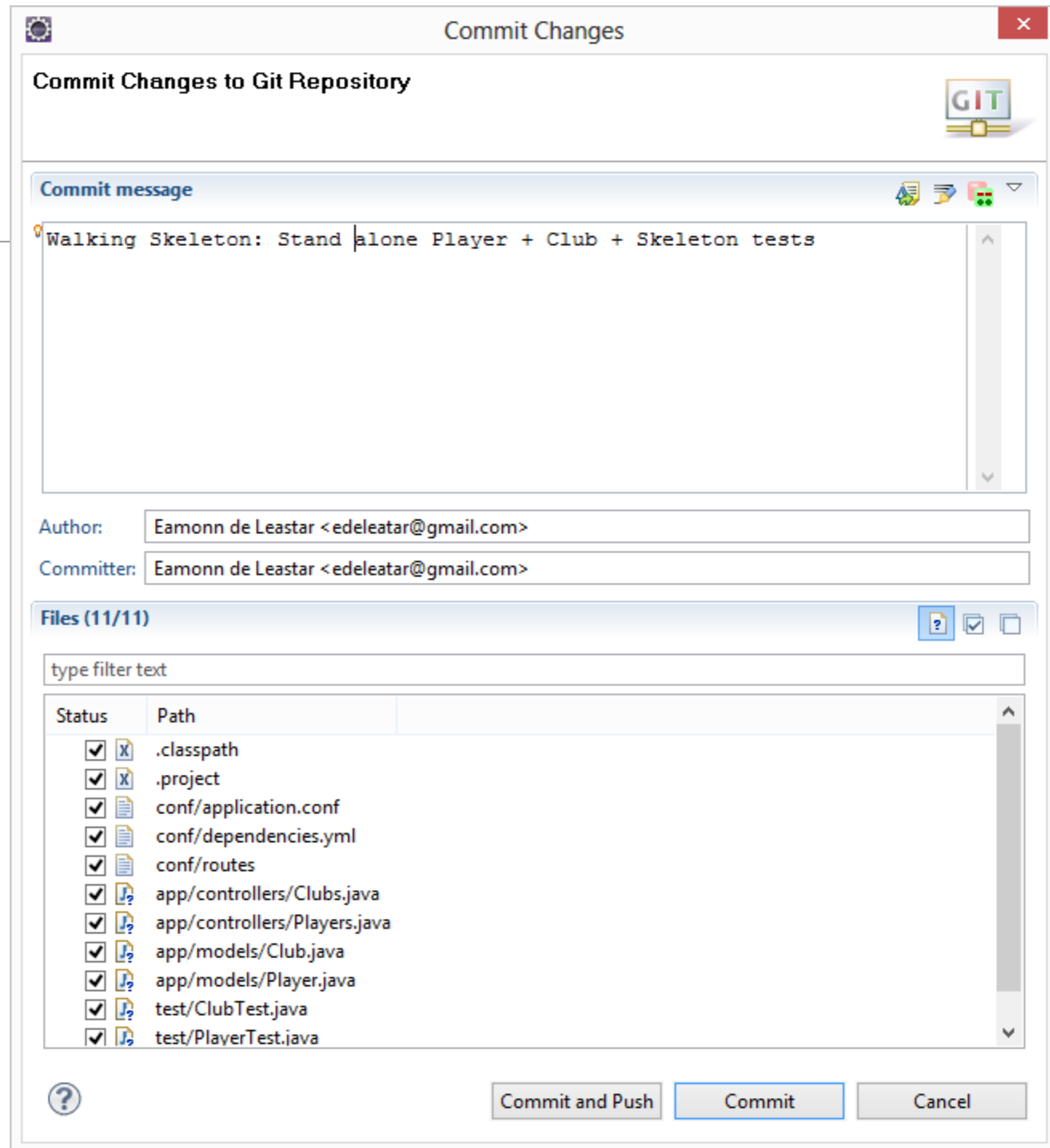
Step 5: Before Commit

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



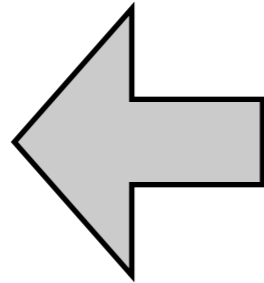
Step 5: Commit

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

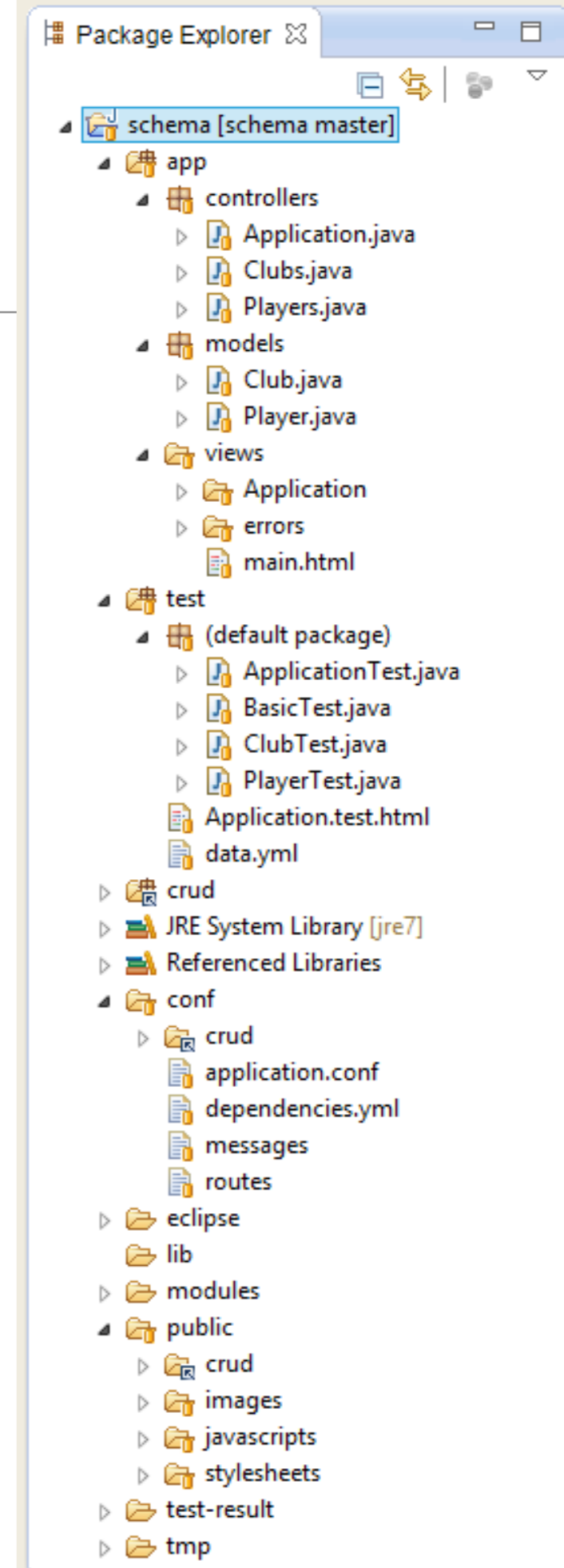
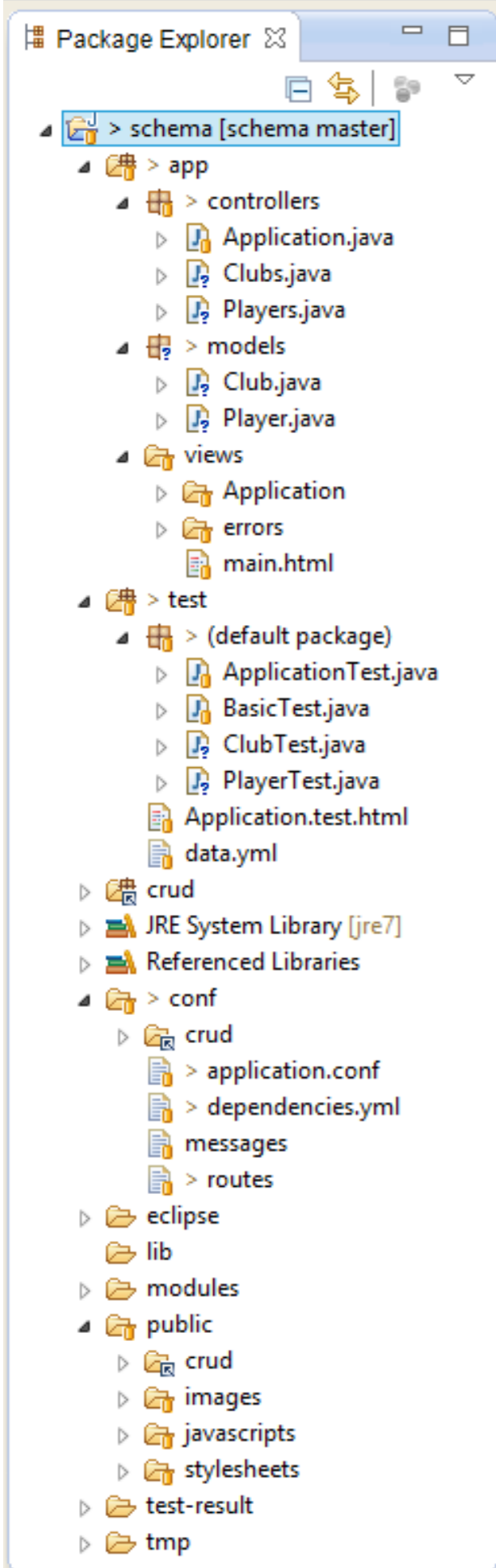
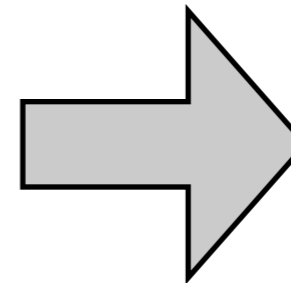


Step 5: After Commit

- Changes made but NOT committed



- Changes committed



Step 5: Repository History

The screenshot shows an IDE window titled 'Project: schema [schema]' with a 'History' tab selected. The history table lists two commits:

Id	Message	Author	Authored...	Committer	Committ...
637c707	master (HEAD) Walking Skeleton: Stand alone	Eamonn de Leastar <edelea...	11 minutes ago	Eamonn de Le...	11 minutes ago
e11590a	First Version as generated by Play	Eamonn de Leastar <edelea...	62 minutes ago	Eamonn de Le...	62 minutes ago

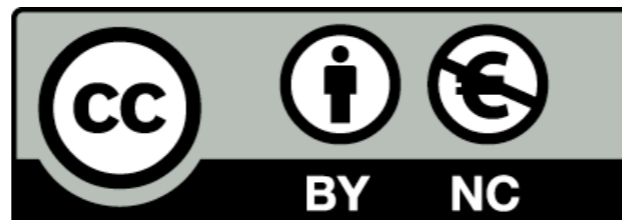
Below the table, the commit details for the selected commit (637c707) are displayed:

```
commit 637c7078a4baeb03944bfff991e38fd41f5266fe7
Author: Eamonn de Leastar <edeleatar@gmail.com> 2013-03-31
13:56:36
Committer: Eamonn de Leastar <edeleatar@gmail.com>
2013-03-31 13:56:36
Parent: e11590ae16e2e2acbd553474d16e0da7ba245fc4 (First
Version as generated by Play)
Branches: master

Walking Skeleton: Stand alone Player + Club + Skeleton
tests
```

On the right side of the IDE, a file explorer shows the project structure:

- .classpath
- .project
- app/controllers/Clubs.java
- app/controllers/Players.java
- app/models/Club.java
- app/models/Player.java
- conf/application.conf
- conf/dependencies.yml



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/>

