Session 2

Aims

This session will give the basics of setting up a new project in Play. You'll be comfortable with some of the SBT commands to compile test and run the application. You'll create a project in Git and check the source in and push to a GitHub remote. Finally you'll be able to open the project in the IntelliJ IDE.

Prerequisites

You'll need to have all the prerequisites listed in the Outline.

In addition, you'll need to have setup SSH keys in github https://help.github.com/articles/connecting-to-github-with-ssh/. Don't worry if you've not done this, the instructors will be able to help.

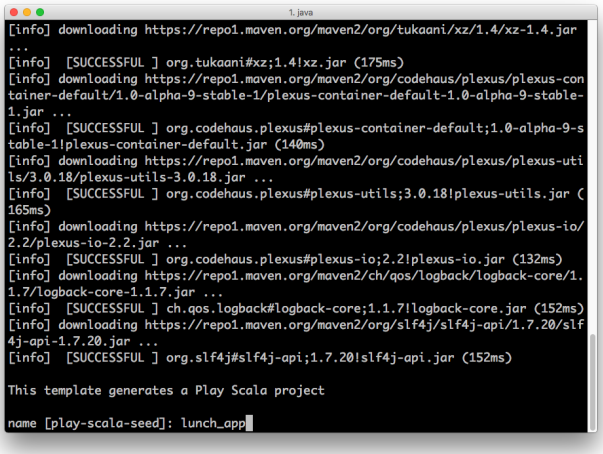
Lesson Steps

New project

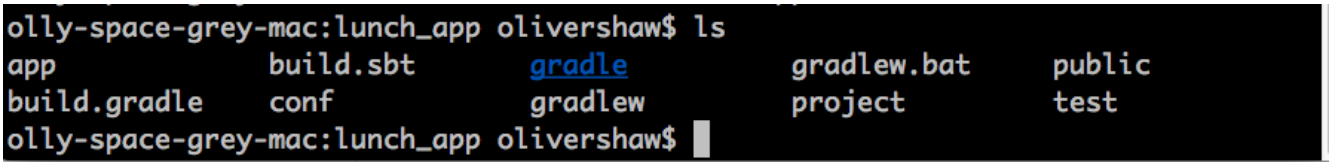
Starting with a bash terminal run the command. Note that you'll need sbt version 0.13.13 or later for this command to work.

```
sbt new playframework/play-scala-seed.g8
```

Fill in appropriate answers to the guided setup program. The defaults are fine, but you'll want to change the name.



You'll now have the new lunch app directory with the play application inside. Change (cd) into the directory and have a look:



SBT basics

There are some basic commands you can now run to interact with SBT. These are:

- test
- compile
- run
- test-quick

These commands can be run either by entering interactive mode or by running the command directly (batch mode). Interactive mode will prevent you paying the startup costs each time.

The result of running `sbt test` should be green (red, green, refactor). It should show you that there are some tests already...

```
1. bash
[info] downloading https://repo1.maven.org/maven2/org/scala-lang/scala-compiler/
2.11.11/scala-compiler-2.11.11.jar ...
[info]  [SUCCESSFUL ] org.scala-lang#scala-compiler;2.11.11!scala-compiler.jar (
1185ms)
[info] Done updating.
[info] Compiling 8 Scala sources and 1 Java source to /Users/olivershaw/projects
/hmrc/lunch_app/target/scala-2.11/classes...
[info] 'compiler-interface' not yet compiled for Scala 2.11.11. Compiling...
[info]   Compilation completed in 10.004 s
[info] Compiling 1 Scala source to /Users/olivershaw/projects/hmrc/lunch_app/tar
get/scala-2.11/test-classes...
[info] HomeControllerSpec:
[info] HomeController GET
[info] - should render the index page from a new instance of controller
[info] - should render the index page from the application
[info] - should render the index page from the router
[info] ScalaTest
[info] Run completed in 2 seconds, 954 milliseconds.
[info] Total number of tests run: 3
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 3, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: Total 3, Failed 0, Errors 0, Passed 3
[success] Total time: 23 s, completed 24-May-2017 14:58:27
olly-space-grey-mac:lunch_app olivershaw$
```

View the running project

The project can be run from the command line so you can see it in the browser.

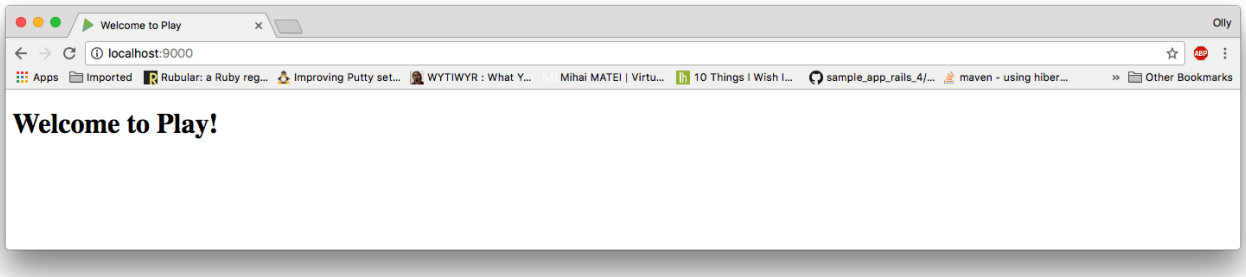
```
sbt run
```

```
1. java
[info] ScalaTest
[info] Run completed in 2 seconds, 954 milliseconds.
[info] Total number of tests run: 3
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 3, failed 0, canceled 0, ignored 0, pending 0
[info] All tests passed.
[info] Passed: Total 3, Failed 0, Errors 0, Passed 3
[success] Total time: 23 s, completed 24-May-2017 14:58:27
olly-space-grey-mac:lunch_app olivershaw$ sbt run
[warn] Executing in batch mode.
[warn]   For better performance, hit [ENTER] to switch to interactive mode, or
[warn]   consider launching sbt without any commands, or explicitly passing 'she
ll'
[info] Loading project definition from /Users/olivershaw/projects/hmrc/lunch_app
/project
[info] Set current project to lunch_app (in build file:/Users/olivershaw/project
s/hmrc/lunch_app/)

--- (Running the application, auto-reloading is enabled) ---

[info] p.c.s.NettyServer - Listening for HTTP on /0:0:0:0:0:0:0:0:9000

(Server started, use Ctrl+D to stop and go back to the console...)
```

You'll see that by default play uses port 9000. You can now have a look at your application in a browser by visiting

http://localhost:9000

The results are less than impressive. But you've got a fully functional play app!



Commit to GitHub

Now we have something worth committing, it's time to set up a new git repo and push to GitHub.

Creating a new git repo is easy. Just type the following in your lunch app directory.

```
git init
```

That's it!

One thing you'll notice is that there are a lot of files in this project. Some of them we want to commit to GitHub (conf, test etc) and some we don't (logs, target, .project). The play template has helpfully given us a file named `.gitignore`. This file lists the stuff that git should ignore. Unfortunately there's an issue in that the .g8 files are not in this .gitignore file. Let's add it in. Your .gitignore file should now look like the following:

```
olly-space-grey-mac:lunch_app olivershaw$ cat .gitignore
.g8
logs
target
/.idea
/.idea_modules
/.classpath
/.project
/.settings
/RUNNING_PID
```

Now this is ok, we can be aggressive and add everything to git safe in the knowledge that it will ignore files that should not be there.

```
git add --all
```

Let's add a commit.

```
git commit -m "initial commit"
```

check that the directory is clean by running

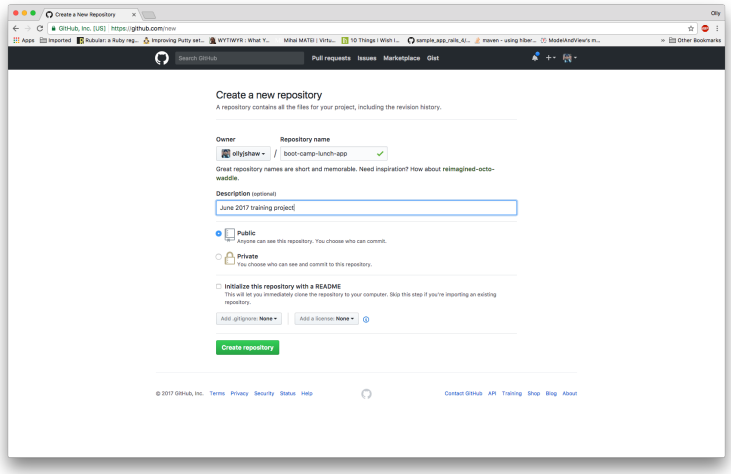
```
git status
```

```
olly-space-grey-mac:lunch_app olivershaw$ git status
On branch master
nothing to commit, working tree clean
```

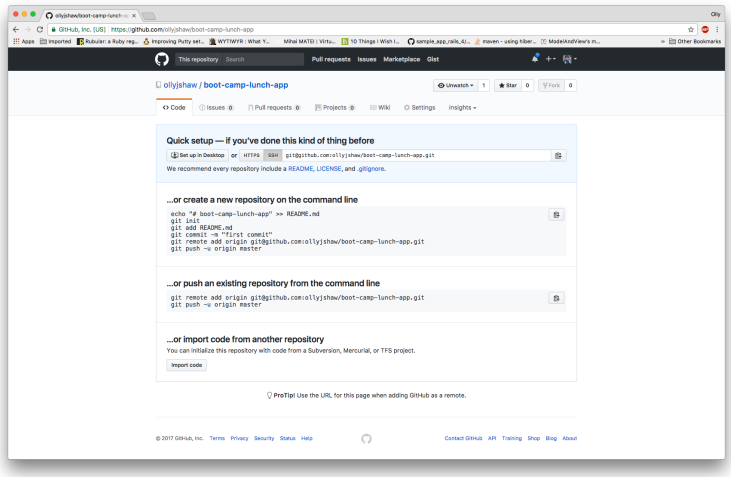
Creating the GitHub Remote

We've got a local repo, but we need to make sure the source is put somewhere safer and more accessible than our laptops. In Git we call this a 'remote'. There are lots of options for remotes in Git, however we'll use the well known GitHub.

In GitHub, make sure you're signed in and then create a new repository. Fill in some details, but in this case don't add a .gitignore



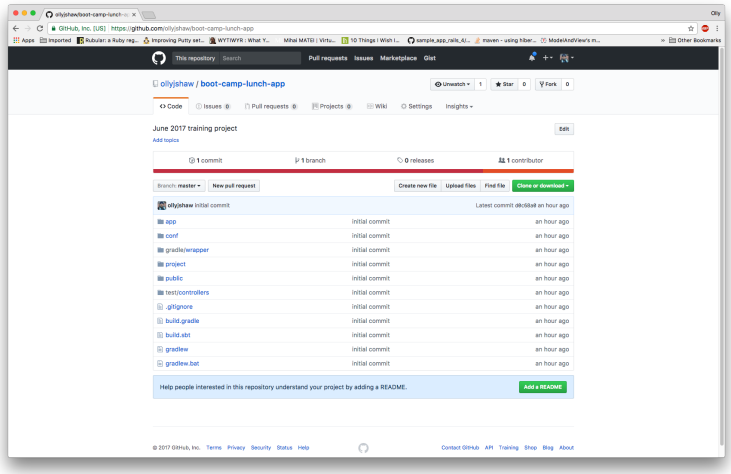
With this complete, you'll be presented with some options on what to do next.



In the second second section are exactly the commands you need to use

```
git push -u origin master
```

With this carried out, you should see your files in GitHub!



Open in IntelliJ

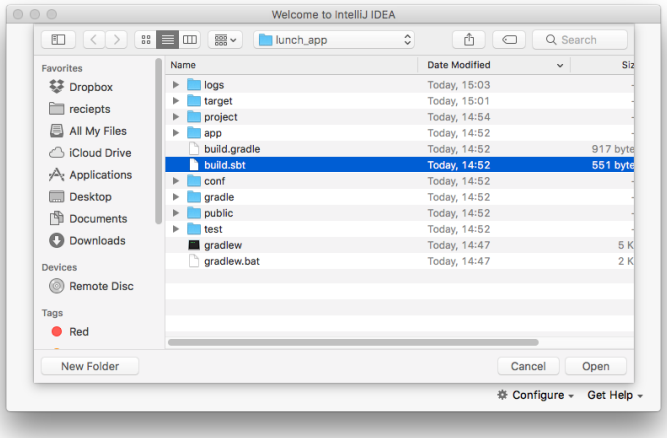
Up untill this point we have not used anything other than the command line, perhaps an editor. (Bonus points if you used vim and worked out how to close it. Millions have needed help with this (https://stackoverflow.blog/2017/05/23/stack-overflow-helping-one-million-developers-exit-vim/). It's crucial that you know how everything works on the command line. It's means you know what the program and utilities do. It's especially crucial with git. Indeed you'll seldom see a git veteran using anything other than the command line to commit pull and push.

Once you have an understanding of what is going on you may find that an Integrated Design Environment (IDE) is helpful to develop code. Here at HMRC lots of people like IntelliJ. It's not mandated, use the tool you're comfortable with. However we're sticking with it in this tutorial.

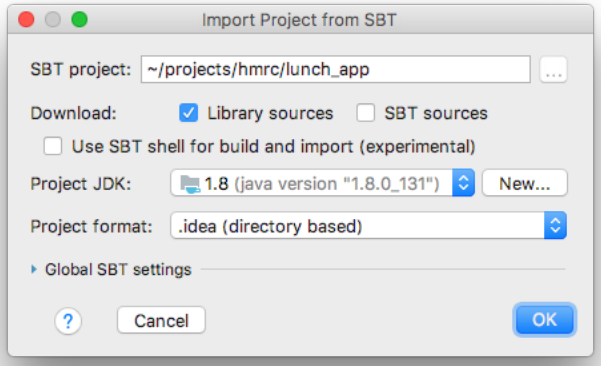
With that warning, let's open in IntelliJ

Select "Import project"

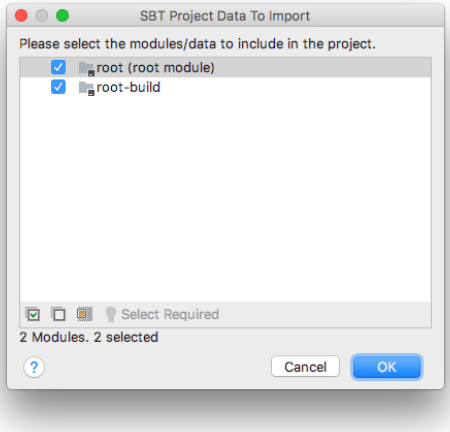
Find the build.sbt file (you know where your files are right? 😀)



The default options for the import will do



Select both modules



Done, give IntelliJ a few mins to download dependancies and index your code.

Final outcome

If you've followed along. Your machine should be ready to code. Your github repo should be at the same stage as this https://github.com/ollyjshaw/boot-camp-lunch-app/tree/session1

Homework