

# Pro Dev Session



# Pro Dev Session

You may want to write this down.

Collaboration is an essential part of being a great teammate and developer. To be a good collaborator, we must master our collaboration tools, namely git and GitHub.

Today we will get more practice with git.



# Pro Dev Session

You may want to write this down.

We previously looked at the `add`, `commit`, `log`, and `checkout` commands.

Today we will continue down in this learning journey by learning the `status`, `stash`, and `branch` commands.

# Pro Dev Session

You may want to write this down.

The `status` command allows us to see which files are being tracked, which changes have been committed, which branch we are on, and more.

This is a command you should use often, especially when you run into errors and you aren't sure what's wrong.

# Pro Dev Session

You may want to write this down.

The `stash` command allows you to temporarily save work that you don't want to save forever.

Remember when you used the `commit` command to permanently save your project exactly as it was at that moment in time? Well `stash` is like `commit`'s baby sister. It saves your work but more temporarily and without a record that's tied to your project.

# Pro Dev Session

You may want to write this down.

But why would you want to save something temporarily? To understand this, we first need to take a detour and understand branching.

Git allows us to create isolated areas to work on new features for our project. This allows developers to safely build new functionality without fear of messing up their master copy or interfering with other features that they may be working on.

These isolated areas are called branches. You can create one using the `branch` command.

# Pro Dev Session

You may want to write this down.

On an actual project, you'll often have many working branches on your machine at once. The clean working production ready code that mirrors what's on GitHub is in a branch named master. You will get to name all of the additional branches that you add.


# Pro Dev Session

You may want to write this down.

So why would you want to save something temporarily?

Git fundamentally does not want you to lose things, so it won't let you move from one branch to another without some form of saving. If you want to change branches but your work is in a messy state that you don't want to permanently save or you're on a branch that you would like to leave so you can delete it, you still have to save your work first.

The `stash` command allows you to save it in a non-permanent way.





# Pro Dev Session

You may want to write this down.

Sidebar: Git takes time to master. We'll be going back over the concepts from the last 2 days again in the next class before we add any significant new knowledge.

# Git Practice

## Goals:

- Create a new repository on GitHub and clone it to your local machine.
- Create a main method and add some print statements.
- Assume this is your working production ready code and your boss asks you to add a new chat feature.
- Type `git branch chat` to create a new branch called chat (This allows you to safely build the chat feature without risk of messing up the production ready code).
- The previous command created the new branch but you aren't on it. Type `git status`. This will show you the branch you are on and which files are unmonitored (not added) and which monitored changes are not committed.
- Type `git add -A` and `git commit -m "initial commit"`.
- Now that you have committed your changes, you can change branches. Type `git checkout chat`.
- You are now on the chat branch. Make some changes in the main method. Add and commit these changes.
- Make some more changes. Do not commit them.
- Suppose your boss decides to scrap the chat feature. Let's go back to master. Type `git checkout master`.
- You can't change branches. Use the `status` command to see what's going on? You have changes that have not been saved, but you don't want to permanently save them.
- Type `git stash` then `git checkout master`. You are now back in your master branch!

# Stand Up!



# Project Day

# Project Day

THIS IS GOING TO BE FUN

You will work in teams of 3-4, practicing paired programming for the entire activity.

Switch driver and navigator every 30 minutes.

You will essentially be working as 2 independent teams of 2 whose code needs to interact.  
This means coding to spec is very important.



# Project Day

## INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work in teams of 4 split into 2 pairs.

### Goals:

- Have a sprint planning meeting to assign features to pairs (guess at what you can complete in the
- Work for 30 minutes
- Have a scrum and adjust expectations if needed.
- Work for 30 minutes
- Rinse and Repeat



**All Day**

# Project Day

THIS IS GOING TO BE FUN

In this activity, you will design and build a REST API based on the specification that follows. This specification is intentionally vague—this activity will force you to think about the design of the application and to ask good questions about the requirements.



# Project Day

THIS IS GOING TO BE FUN

This application must keep track of city information.

It must track the following data about cities:

- Name
- State
- Population
- Whether or not the city is a state capital





# Project Day

THIS IS GOING TO BE FUN

The API must allow the user to do the following:

Add a city

- **URI:** /city
- **Method:** POST
- **Request Body:** City
- **Response Body:** City

Delete a city by name

- **URI:** /city/{name}
- **Method:** DELETE
- **Request Body:** None
- **Response Body:** None

Retrieve all the cities

- **URI:** /city
- **Method:** GET
- **Request Body:** None
- **Response Body:** City List

Retrieve one city by name

- **URI:** /city/{name}
- **Method:** GET
- **Request Body:** None
- **Response Body:** City

Retrieve all the cities with a population greater than the supplied request parameter

- **URI:** /city/population/{population}
- **Method:** GET
- **Request Body:** None
- **Response Body:** City List

Retrieve all cities by state

- **URI:** /city/state/{state}
- **Method:** GET
- **Request Body:** None
- **Response Body:** City List

# Project Day

THIS IS GOING TO BE FUN

This application must:

- Store cities in an ArrayList.
- Use JSR 303 Validation to validate all request bodies as thoroughly as is reasonable.
- Add at least 1 validation not using JSR 303 such as validating that each State and Capital begin with a capital letter.
- Respond with a 422 Status Code for any invalid request body.



# Project Day

## INDEPENDENT PRACTICE

It's time to fly. Focus. Work hard. Ask for help when you need it.

Work with your team.

### Goals:

- Have a retrospective
- What went well?
- Did you complete everything you planned?
- How did you share code?
- Could it have gone better?
- What would you do differently next time?



10 min



**IT'S TIME FOR OUR WEEKLY RETROSPECTIVE**

This is your chance to drive your own education. Let's work together to make our team stronger.

# Wrap Up

# Module 3 Lesson 5

## SUMMATIVE HOMEWORK

This homework is due in 1 week.

- Summative Assessment: Spring Boot and REST due July 23 at 11:59pm