

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. We'll be building on this for the next several lessons.



Pro Dev Session

You may want to write this down.

Git is a version control software that runs on your computer. You can think of it like Microsoft Word.

When you work on a project in a repository on your computer, git is running in the background.

Git essentially just runs idley behind the scenes until you tell it to do something.

The command `git add -A` tells git to monitor all the files in your project for changes.

At this point git is just watching and making note of any changes. It's not saving them.

Pro Dev Session

You may want to write this down.

The command `git commit -m "<meaningful message>"` saves the project in its current state.

This is where the Microsoft Word metaphor breaks down. When you save a document in Word, you override any previous versions.

With git, the `commit` command is similar to save but it saves a totally new version of your project exactly as it is in that moment. Old versions are preserved and never automatically overridden.

Git Practice

Work individually but with the support of your classmates to complete all of the goals below.

Goals:

Let's see how powerful version control really is!

- Create a new repository on GitHub and clone it to your local machine, using the command `git clone <address>`
- In the new repository, create a new IntelliJ project that simply prints your name to the console.
- In the terminal type `git add -A` followed by `git commit -m "initial commit"`.
- Delete the print statement and replace it with 2 integers. Then print the sum of the integers.
- In the terminal type `git add -A` followed by `git commit -m "replaces name with sum"`.
- Type `git log`. You will see a list of all your commits. Copy the long identifier next to the word commit from your initial commit. This is called a commit hash, id, or SHA (yes, it has 3 names).
- Type `git checkout <SHA>`. Look in IntelliJ. All the code from your original commit is



20
minutes!



Check-in Time

- When might this need to checkout old versions of your code be useful?
- How often do you think you should commit your code?
- What should you do once you have committed your final change on a feature? Should you continue to only store it locally? How can you store it somewhere safer?



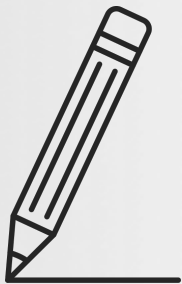
Stand Up!



REST APIs

REST APIs

FOUNDATIONAL API KNOWLEDGE



Notebooks Ready? It's time for a short refresher.



REST APIs

FOUNDATIONAL API KNOWLEDGE

Recall that APIs are an interface through which separate systems interact.



REST APIs

FOUNDATIONAL API KNOWLEDGE

Recall that REST is a set of rules around how 2 systems should communicate. Namely, that requests should be sent using an HTTP verb (GET, POST, PUT, DELETE) and a URI that identifies the resource being accessed.





Check-in Time

- If `GET /students` returns all students and `GET /students/{id}` returns the student with a specified id, what would you use to add a new student?



Recall this API documentation that we saw last class. This will be useful in the next exercise.

URI: <https://rest-users.herokuapp.com>

- GET /users
 - Returns all users
- GET /users/{id}
 - Returns User with specified id
- POST /users
 - Adds a new User
- PUT /users/{id}
 - Replaces User at specified id with the one provided in the request body
- DELETE /users/{id}
 - Deletes the User with the specified id

REST API

INDEPENDENT PRACTICE

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

Work in PAIRS to complete all of the goals below.

Goals:

Suppose you are building an application for a small elementary school.

- Using the previous slide as a guide, provide basic API documentation for an application that allows school staff to register new students, see a roster of all students, see information on an individual student based on id, update a student's status, and remove a student.



**10
minutes!**

REST API part II

INDEPENDENT PRACTICE

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

Work in PAIRS to complete all of the goals below.

Goals:

- Using Spring Initializer, build the GET and GET by id routes from the previous activity.
- You should begin by creating a student class with id, name, and isEnrolled.
- Next initialize an ArrayList of Students in the Rest Controller constructor and add 5 students.
- Lastly, set up your 2 routes.

*Think hard on how to get by id. We're not just getting by index this time. You'll need to check each student's id.

**20
minutes!**





Stay Seated & Take 3 Deep Breaths.

RELAX.

Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back in a few minutes.



REST Cont'd

REST API cont'd

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
@RequestMapping(value = "/students", method = RequestMethod.POST)
@ResponseStatus(value = HttpStatus.CREATED)
public Student createStudent(@RequestBody Student student) {
    student.setId(id++);
    stuList.add(student);
    return student;
}
```

REST API part III

INDEPENDENT PRACTICE

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

Work in PAIRS to complete all of the goals below.

Goals:

- Create a new REST API for a service that helps puppies get adopted.
- You will need a Puppy class with id, name, kennelNum, isAdopted.
- Start with an ArrayList of 5 puppies.
- Create a GET route to get all puppies.
- Create a GET by id route to get a single puppy by id.
- Create a POST route that adds new puppies to the ArrayList.
- Test your routes using Postman.



**20
minutes!**

REST API part IV

INDEPENDENT PRACTICE

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



Work in PAIRS to complete all of the goals below.

Goals:

- Add a DELETE route to the PuppyController. You've never seen this before, but you have all the tools you need to figure it out.
- Think through what the differences are between GET and POST routes. What should change in a DELETE route?
- What should you send back from your DELETE route (this is up to you but be ready to justify your answer)?



**15
minutes!**

lunch.

More REST

REST API cont'd

```
@RequestMapping(value = "/students/{id}", method = RequestMethod.PUT)
@ResponseStatus(value = HttpStatus.NO_CONTENT)
public void updateStudent(@PathVariable int id, @RequestBody Student student) {
    for (int i = 0; i < stuList.size(); i++){
        if(stuList.get(i).getId() == id){
            stuList.set(i, student);
        }
    }
}
```

REST API part V

INDEPENDENT PRACTICE

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

Work in PAIRS to complete all of the goals below.

Goals:

- Add a PUT route to your Puppy API
- Include the appropriate status code and return type



**10
minutes!**

REST API part VI

INDEPENDENT PRACTICE

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

Work in PAIRS to complete all of the goals below.

Goals:

- Create a new REST API for a service that tracks t-shirt inventory for an online store.
- You will need a TShirt class with id, color, stockCount, isAvailable. isAvailable should toggle to false when stock is 0.
- Start with an ArrayList of 5 TShirts.
- Create GET, POST, PUT, DELETE, and GET by id routes.
- Include appropriate return types and status codes for each route.
- Test your routes using Postman



**30
minutes!**

*If you finish early, as a challenge, Google PATCH and try to implement a PATCH route.



Stay Seated & Take 3 Deep Breaths.

RELAX.

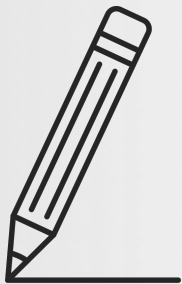
Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back in a few minutes.



Even More REST

REST APIs

EXPANDING OUR KNOWLEDGE



Notebooks Ready? It's time for a short lecture.



REST APIs

EXPANDING OUR KNOWLEDGE

It's important to step back and remember what we are doing. We are building the web layer of a REST API. In a real application (as we'll see in the next module), this data would not be stored in an ArrayList but rather in a database.

The ArrayList aspect will largely disappear next week, but the routes, methods, URIs, status codes, and other aspects of the REST controller will stay exactly the same.

The ArrayList is a crude attempt to mimic a database until we learn about databases.



REST APIs

EXPANDING OUR KNOWLEDGE

Validation is another important component of creating robust REST routes. The `@Valid` annotation can be used to ensure that incoming objects conform to the specifications set in the class.

Check out these [possible constraints](#) that can be added to properties in our class.



REST API cont'd

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
public class Student{

    @PositiveOrZero
    int id;

    @NotBlank
    String name;

    boolean isEnrolled;

    //getters and setters not shown for brevity
}
```

REST API cont'd

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
@RequestMapping(value = "/students", method = RequestMethod.POST)
@ResponseStatus(value = HttpStatus.CREATED)
public Student createStudent(@RequestBody @Valid Student student) {
    student.setId(id++);
    stuList.add(student);
    return student;
}
```


REST API part VII

INDEPENDENT PRACTICE

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

Work in PAIRS to complete all of the goals below.

Goals:


- Using the Javadoc documentation as a guide, add two different validation constraints to your Puppy model (Puppy class).
- Use the proper annotation to enforce those constraints in all routes that take a Request Body.



**10
minutes!**

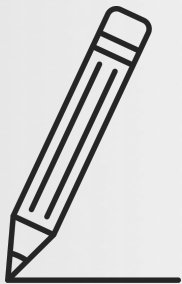
Sneak Peek

Note that next class will start with the sneak peek topic. You are NOT expected to master this today.



Error Handling

BUILDING ROBUST APPLICATIONS



Notebooks Ready? It's time for a short lecture.



Error Handling

BUILDING ROBUST APPLICATIONS

When we build methods, like our API routes, that rely on input from another system, we have little control over what comes into our method. This creates a potential for errors.



Error Handling

BUILDING ROBUST APPLICATIONS

When we build methods, like our API routes, that rely on input from another system, we have little control over what comes into our method. This creates a potential for errors and exceptions.

In the homework, you'll find some additional reading on errors and exceptions.

We'll learn how to handle errors and exceptions in the next couple of classes, but for now, let's just take a look at what they are.



Error Handling

WATCH & LEARN

Close your laptop. Eyes on my screen. Pay attention.

```
public class App {  
    public static void main(String[] args){  
  
        String num = "9";  
  
        int x = Integer.parseInt(num);  
  
        System.out.println(x);  
  
        System.out.println("After integer");  
  
    }  
}
```

```
public class App {  
    public static void main(String[] args){  
  
        String num = "me";  
  
        int x = Integer.parseInt(num);  
  
        System.out.println(x);  
  
        System.out.println("After integer");  
  
    }  
}
```

Wrap Up