

Pro Dev Session



Pro Dev Session

You may want to write this down.

Today's Professional Development sessions will blend technical skills and soft skills. We'll be learning how to be better developers and better teammates. This involves utilizing best practices, employing clear communication in our code, and understanding technical aspects of our craft.

Today we'll discuss annotations.



Pro Dev Session

You may want to write this down.

Annotations are begin with the @ symbol and are somewhere between a comment and code.

Annotations do not impact how compiled code runs but can impact how code is compiled.

Annotations are essentially metadata about code

This is all very theoretical and vague. Let's look at some concrete examples to make sense of this.

Pro Dev Session

You may want to write this down.

```
@Override
public String add(int num1, int num2){
    return "The sum is " + (num1 + num2);
}
```

The `@Override` annotations tells the compiler that this method should override another method. If it doesn't, there will be a compile error. What mistakes could this save us from?

Pro Dev Session

You may want to write this down.

What mistakes could this save us from?

- Typos in the method name
- Misidentifying the class that is being extended
- Incorrectly declaring the parameter type(s) so that the method is actually overloading rather than overriding an inherited method.

Annotation Practice

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:

- Google the @Deprecated and @SuppressWarnings annotations. List examples of when you might use each?
- List 2-3 other common annotations.



**15
minutes!**

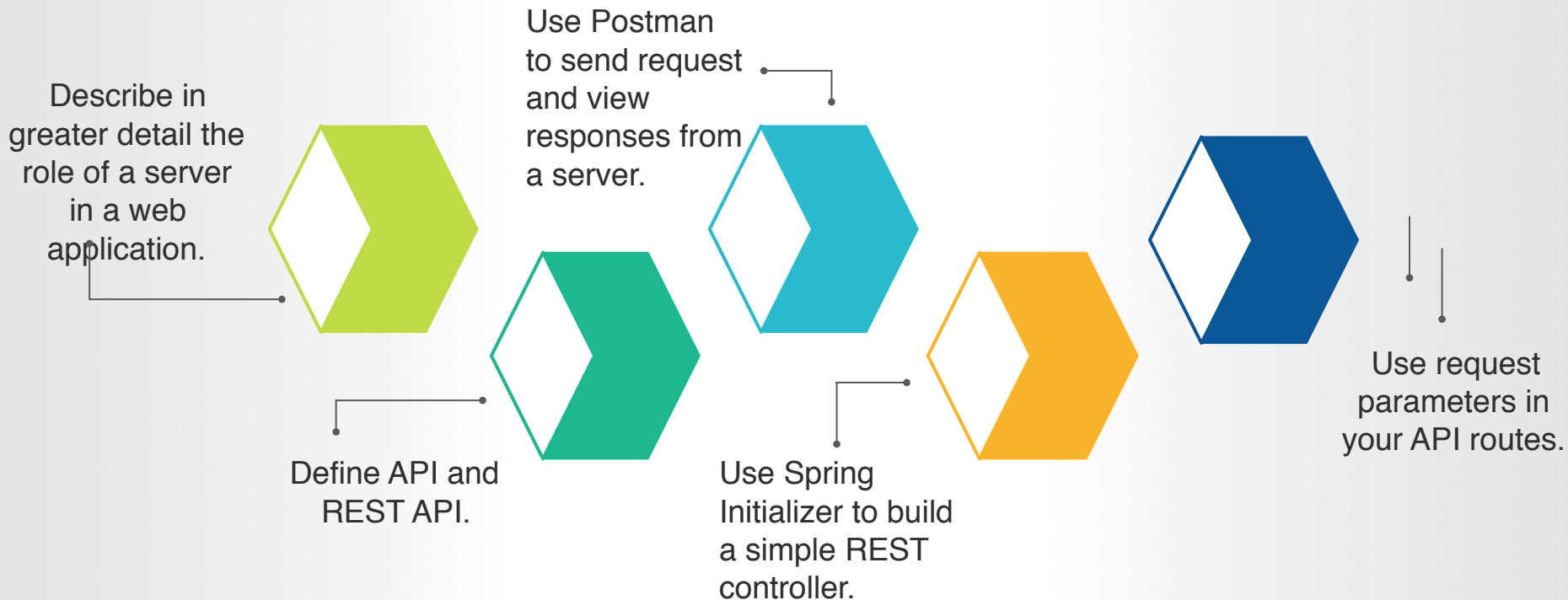
Stand Up!



Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:



Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:



Let's Do This!

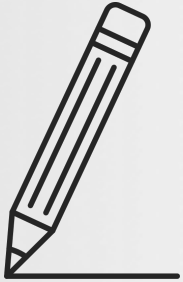


Web Applications



Web Applications

REVIEWING HOW WEB APPS WORK



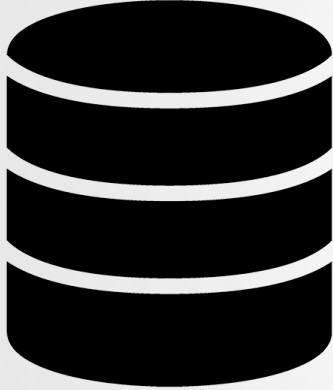
Notebooks Ready? It's time for a short lecture.



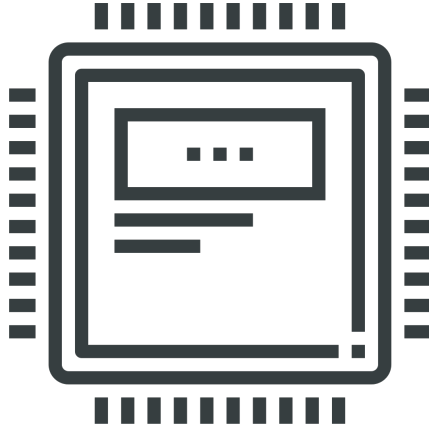
Web Applications

REVIEWING HOW WEB APPS WORK

Web applications are composed of 3 major components



Database



Server

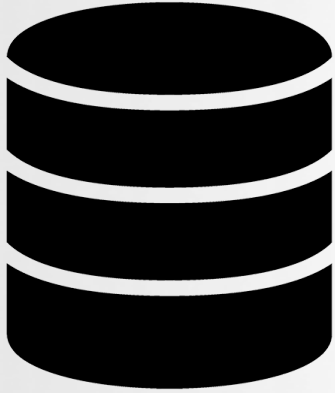


Client

Web Applications

REVIEWING HOW WEB APPS WORK

Databases store data. On a social media site, this might be your username, password, friends, birthday, posts, etc.



Database



Server



Client

Web Applications

REVIEWING HOW WEB APPS WORK

Clients are the computers rendering the web app. Like your computer. My computer. The computer of that person sitting next to you.



Database



Server



Client

Web Applications

REVIEWING HOW WEB APPS WORK

Servers manage the flow of communication between the client and the server. Servers are how the data from the database gets to page you are viewing.



Database



Server



Client

How the Web Works

EXAMPLE WEB APPLICATION

A simplified movie watching app. Let's call it Netflips. On Netflips, users can:

- Watch movies
- Watch the same movie repeatedly (because who doesn't watch Snakes on a Plane over and over?)
- Rate movies
- See a list of recommended movies



How the Web Works

EXAMPLE WEB APPLICATION

Vickie frequents Netflix. All of her data will be stored and used to inform her recommended movie list.



Vickie binge watching all the all the Austin Powers movies.

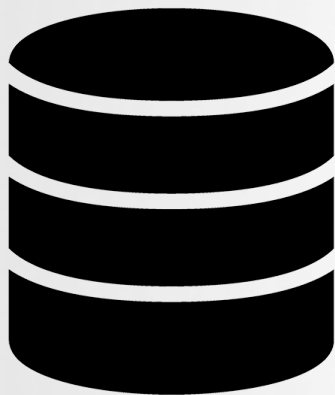
Why, Vickie?
Whyyyyyyyy???!?



How the Web Works

EXAMPLE WEB APPLICATION

Databases are like complex spreadsheets.



Database

Vickie's data

A curved arrow pointing from the text 'Vickie's data' to the first row of the table.

MOVIE	GENRE	REPEAT	RATIN
Old School	comedy	true	G 4.9
Jurassic Park	action	false	2.7
Bill and Ted	comedy	false	4.8
Mall Cop	comedy	true	5.0
It	horror	false	0.2

How the Web Works

EXAMPLE WEB APPLICATION

When Vickie goes to the recommended movie page, her computer asks the server which movies it should show in the list



Server

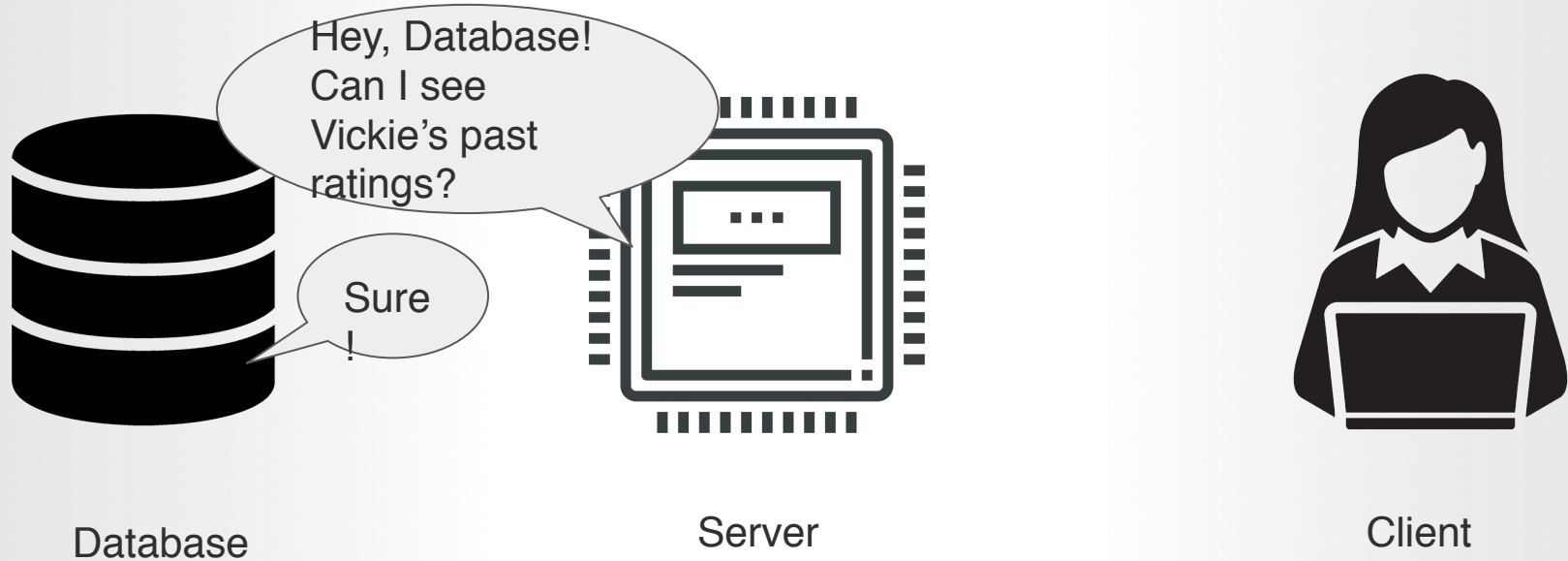


Client

How the Web Works

EXAMPLE WEB APPLICATION

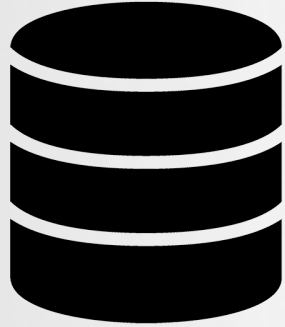
The server requests Vickie's data from the database



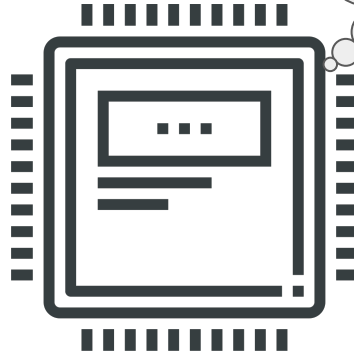
How the Web Works

EXAMPLE WEB APPLICATION

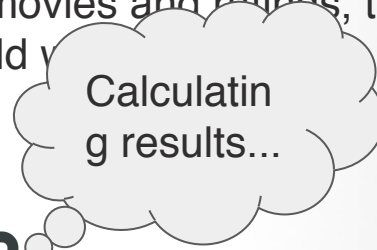
After the database sends a list of Vickie's previous movies and ratings, the server does some complex calculations to determine what Vickie should y



Database



Server

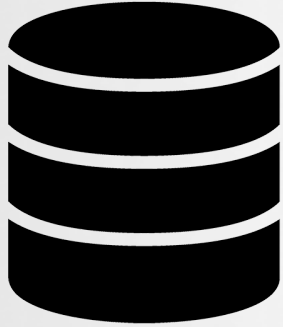


Client

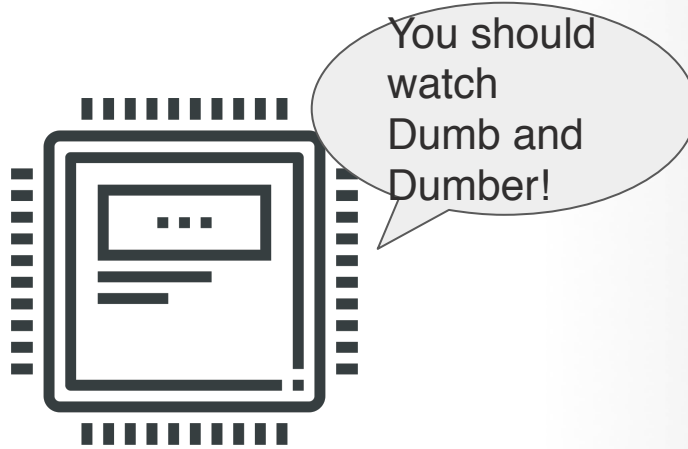
How the Web Works

EXAMPLE WEB APPLICATION

After the server determines what should be on the movie list, it sends it to Vickie's computer. Vickie's computer renders the data in the browser in an easy to read way.



Database



Server



Client

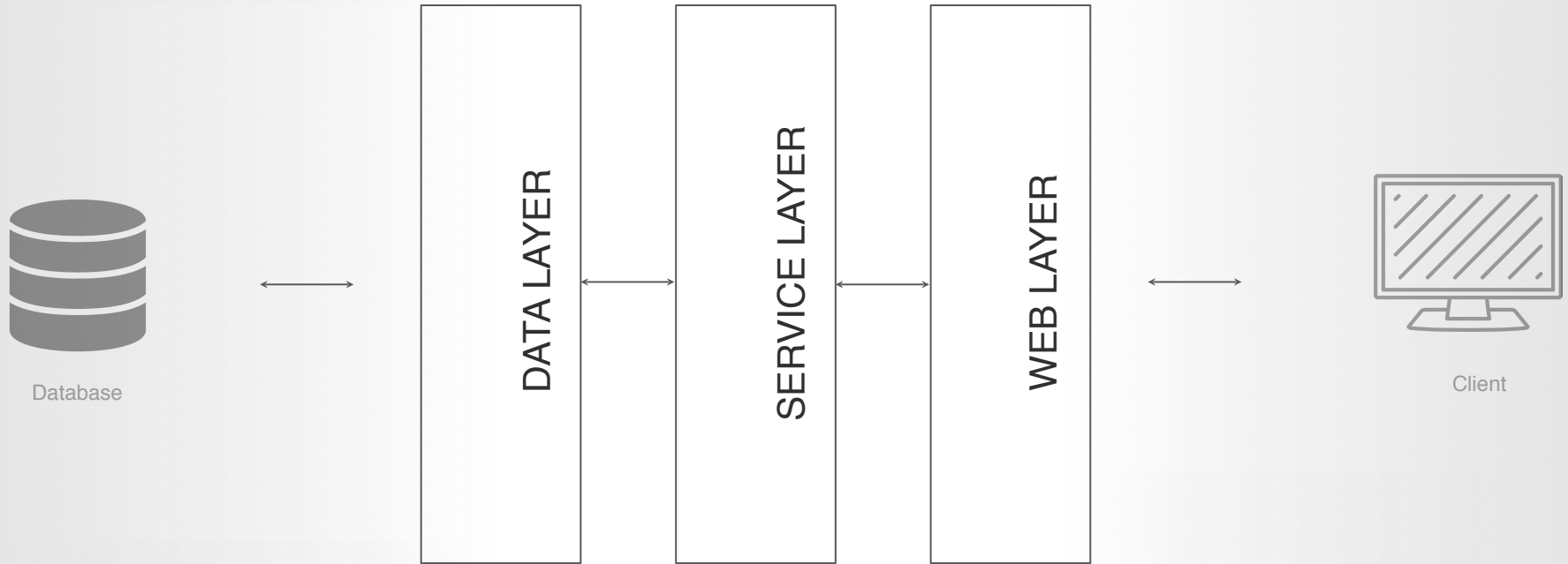
Let's dive a little deeper into the server piece ...



Server Structure

ANATOMY OF A SERVER

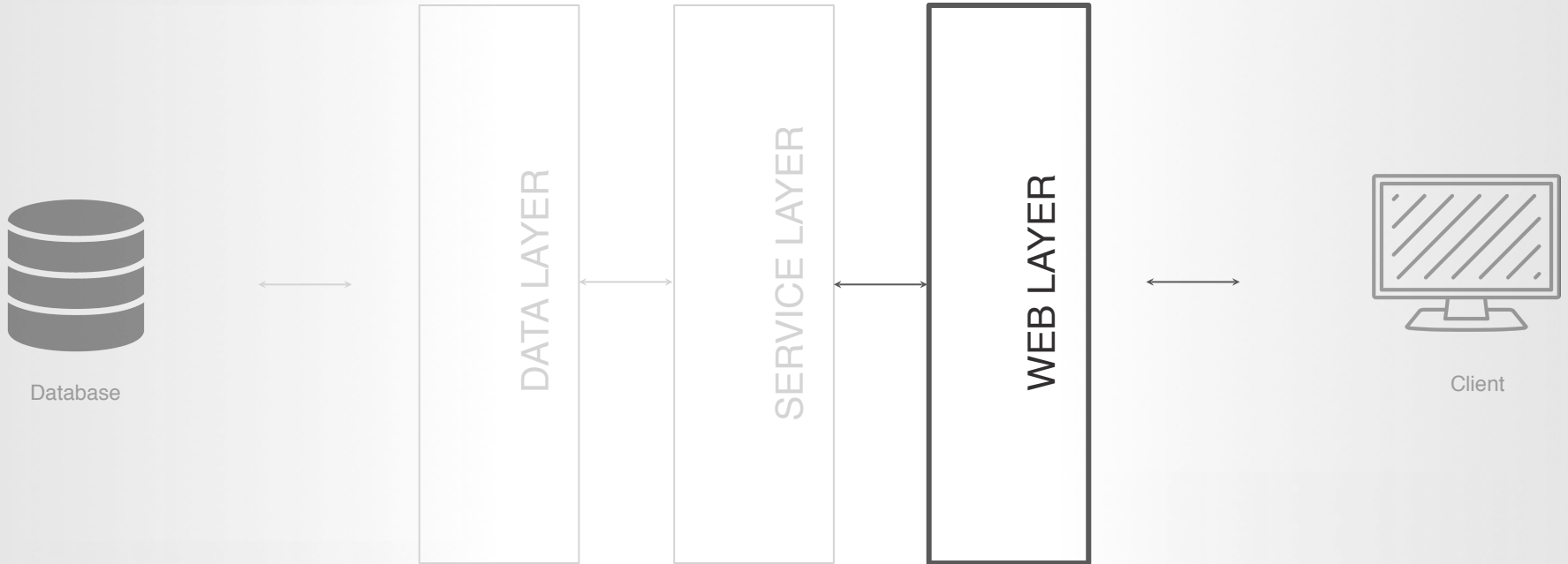
As Java developers, we're focussed on building servers. Spring servers have 3 parts:



Server Structure

WEB LAYER

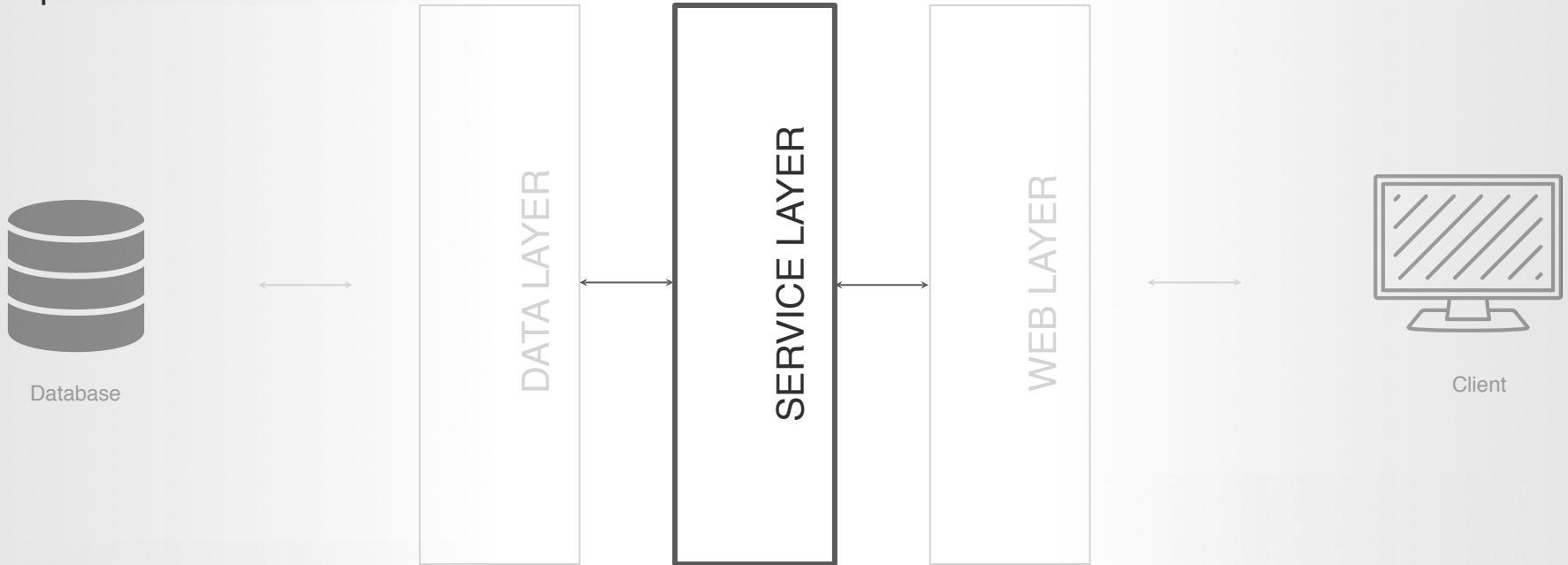
The web layer is responsible for processing requests from the client and sending responses.



Server Structure

SERVICE LAYER

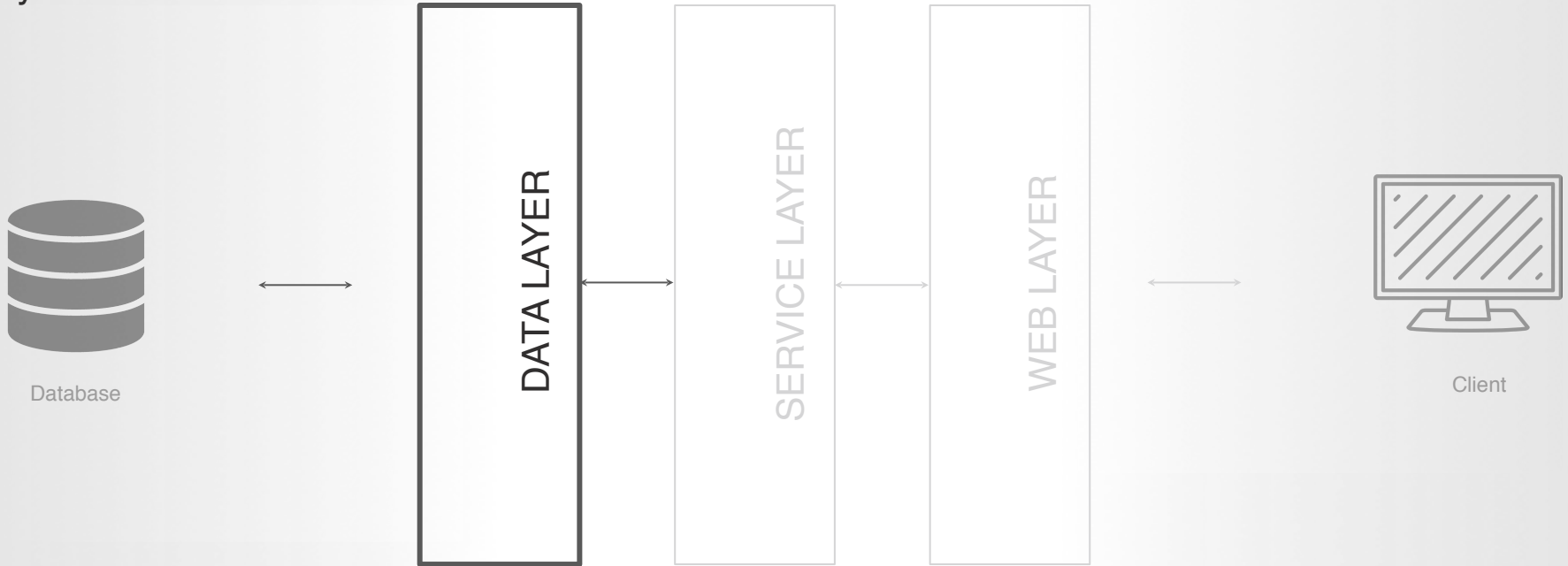
The service layer is responsible for any logic that needs to be performed on the request or the response.



Server Structure

DATA LAYER

The data layer is responsible for getting data from the database and sending it to the service layer.

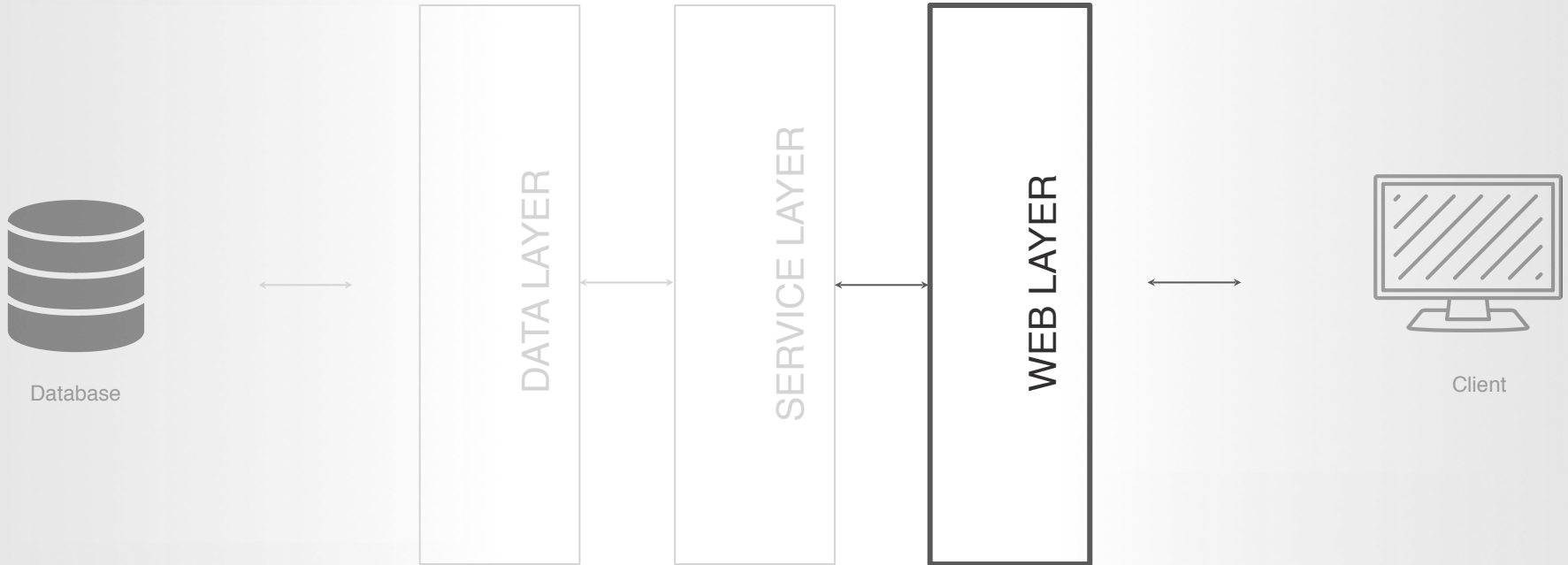


Let's look at a more concrete example. Suppose we are back in Netflix and Vickie clicks on the recommendations page



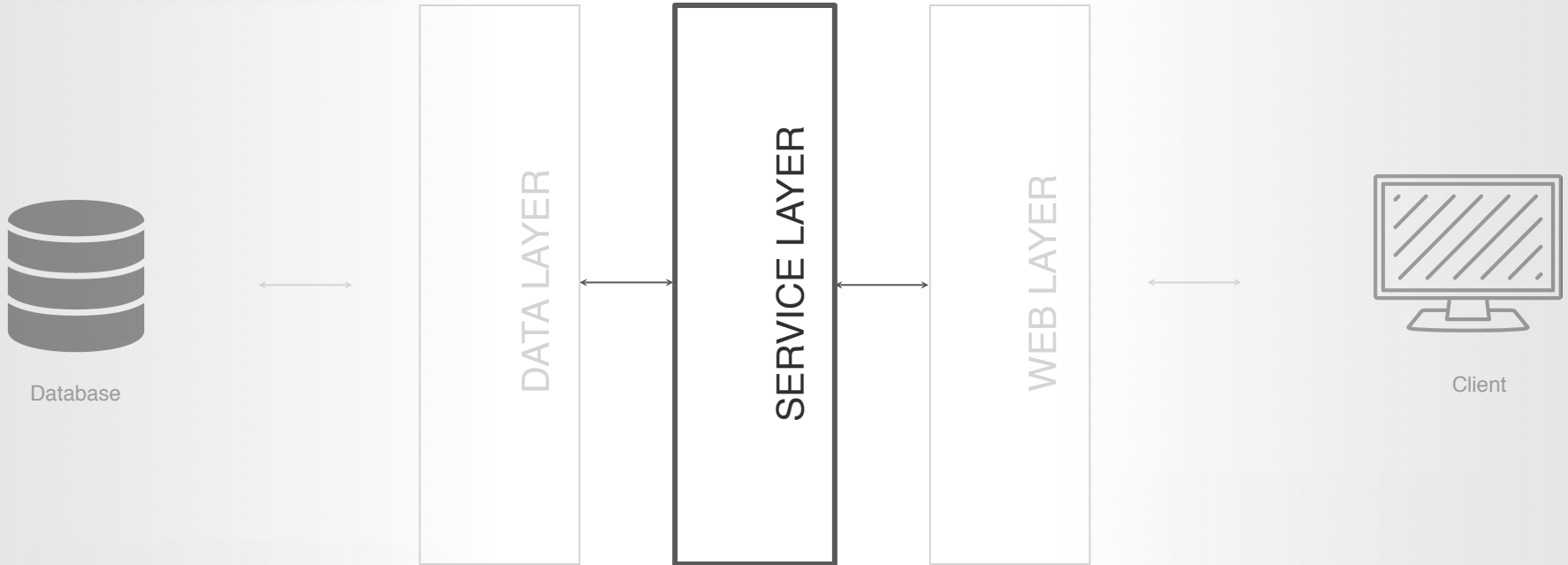
Server Structure

The web layer is responsible for receiving the request for recommended movies and calling the proper method in the service layer, then it waits to get a response from the service layer and sends this response back to Vickie's computer (client).



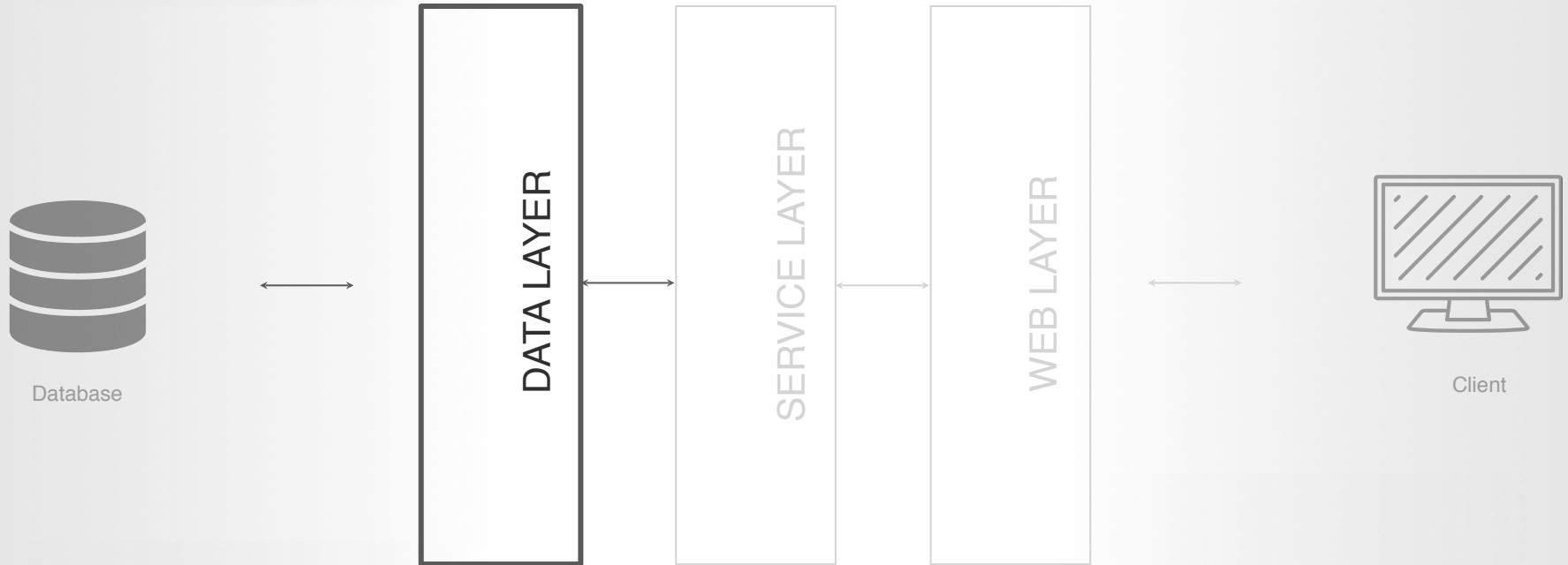
Server Structure

The service layer has a method that was called by the web layer. It's first job is to call a method from the data layer that gets all of Vickie's previous ratings.



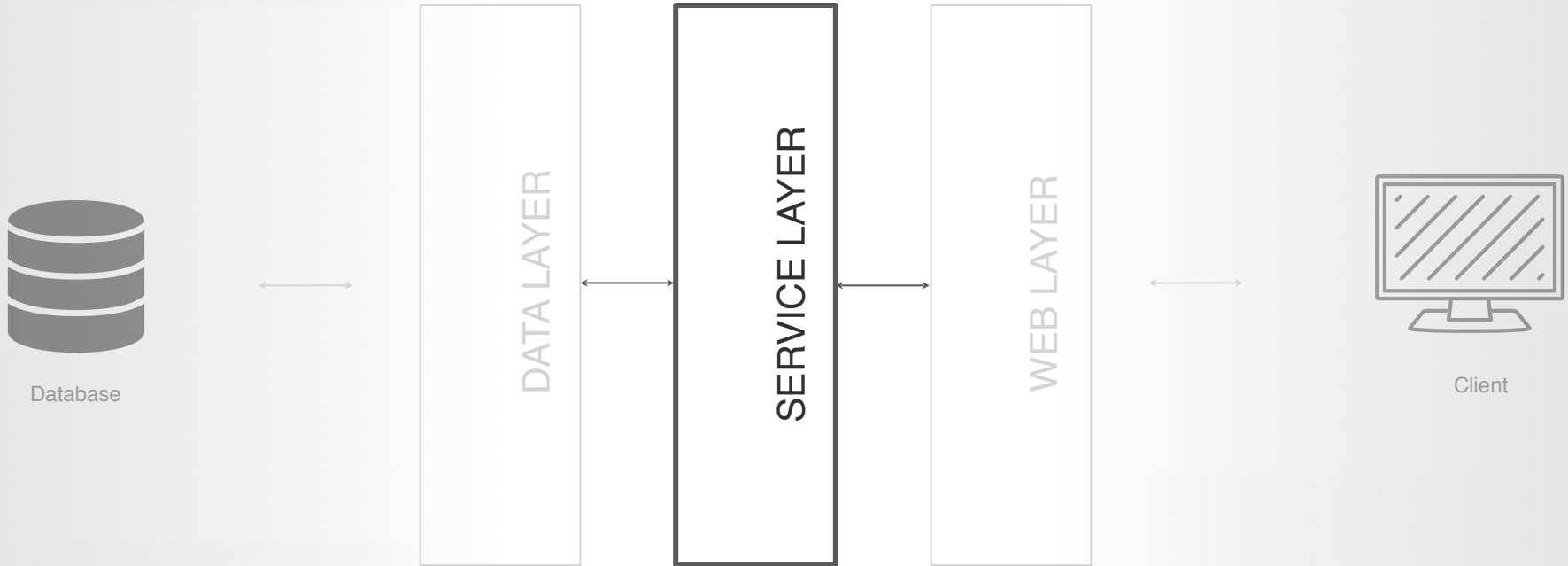
Server Structure

The data layer has a method that was called by the service layer. It's job is to ask the database for Vickie's data. It then returns this data to the service layer.



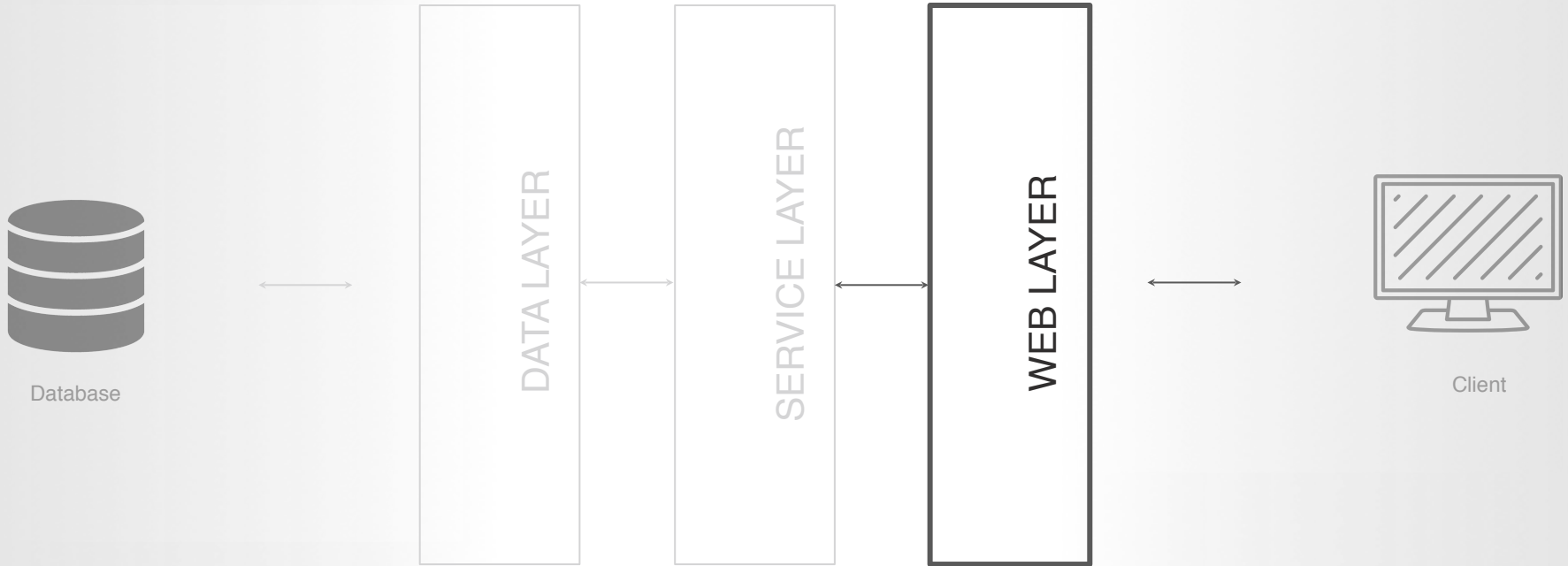
Server Structure

Now that the service layer has Vickie's data, it uses some complex algorithm to figure out which movie to recommend next. Once it knows the new movie list, it returns this list to the web layer. The service layer is kind of the brains of the operation.



Server Structure

The web layer now has the recommended movies and sends this list to Vickie's computer to render visually, so that Vickie can select a movie to watch.





Check-in Time

- On the Amazon website, which layer handles querying the database retrieve all items in the “Lawn Care” category?
- Which layer calls the proper method from the service layer when a client computer requests items in “Lawn Care”?
- Suppose you want to ensure that you never return a blank result page, where would you write the logic that fetches and returns related items if “Lawn Care” returns no results?



How the Web Works

EXAMPLE WEB APPLICATION

These concepts are very important to understand, so let's discuss a couple more examples:

[Pinterest](#)

[NASA API](#)



How the Web Works

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:

- Choose a web application
- Write an example of a task that the data, service, and web layers would each manage for this web application.

Note: This isn't always as black and white as we are making it seem. There will always be judgement calls that developers or teams have to make. Can you think of a task that could go in either the data or service layer? Where would you put it? Why?



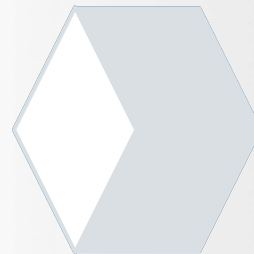
**10
minutes!**

Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:

Describe in greater detail the role of a server in a web application.





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 APIs

REST APIs

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:

- Spend 15 minutes learning as much as you possibly can about REST APIs with your partner.
- Come back ready to discuss your findings.

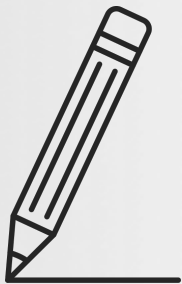
Note: Try to start with the least technical articles and videos that you can find and then move on to more technical content.



**15
minutes!**

REST APIs

A SET OF ARCHITECTURAL RULES



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



REST APIs

A SET OF ARCHITECTURAL RULES

An API is an interface that allows 2 systems to interact.

- Nest thermostats have an API that allow your phone application to interact with your thermostat.
- Instagram and Facebook allow you to cross post between platforms. That's possible because Facebook and Instagram have APIs that allow them to communicate.
- Servers have APIs that allow clients to get information from them and display them to end users.



REST APIs

A SET OF ARCHITECTURAL RULES

APIs control what access other systems have to the functionality of your system.

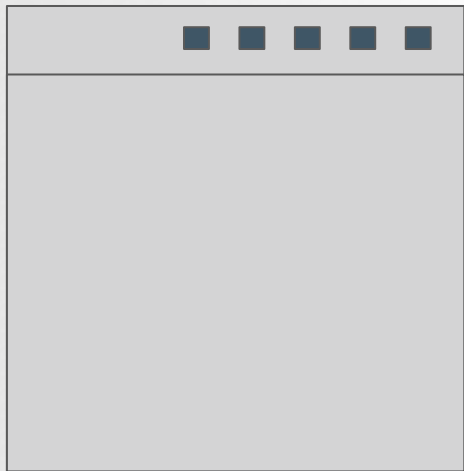
Think about the API on your banks server. It needs to control what functionality your computer has access to. For example you can withdraw money from your account, but not from anyone else's account.

Similarly, although the bank system undoubtedly has a way to delete accounts, you as a user don't have access to that functionality.



REST APIs

A SET OF ARCHITECTURAL RULES



When thinking of APIs consider your dishwasher. Your dishwasher has a lot of functionality.

It can set the water temperature between 100 and 160 degrees.

It can release the soap from its compartment.

It can set the water pressure from 20-120 PSI.

But you as a user can only interact with these settings through certain predefined methods (normal, heavy, pots & pans, etc)

Like an API, the button panel provides limited access to some of the dishwasher functionality.

REST APIs

A SET OF ARCHITECTURAL RULES

A client (someone using the API) makes a request to the API. The API returns some response.

With Nest, I can send a request from my phone app (client) to the Nest to drop the temperature. The Nest sends a response notifying the app that the temperature has now been dropped to the specified amount. The app then displays the new temperature to me.

With a bank app, I can send a request to logout. The server then responds that my logout was successful, and my browser notifies me by displaying a logout screen.



REST APIs

A SET OF ARCHITECTURAL RULES

There are loads of different kinds of APIs.

REST APIs are APIs used to transmit data over the internet from a server to a client using a particular set of rules.

We'll talk more about this in later lessons, but for now let's take a high level look at the rules of REST APIs.



REST APIs

A SET OF ARCHITECTURAL RULES

Somewhat formally, these are the rules of REST:

1. Resources are indicated by URIs
2. Actions are indicated by HTTP methods
3. Data is transferred via media types in the request and response bodies (usually JSON)



REST APIs

A SET OF ARCHITECTURAL RULES

Somewhat formally, these are the rules of REST:

1. Resources are indicated by URIs
2. Actions are indicated by HTTP methods
3. Data is transferred via media types in the request and response bodies (usually JSON)


What this practically means is that requests use a verb like GET (which is used to retrieve data) and a URI (a unique string) to request certain information

For example:

At your bank, GET `"/balance"` might be used to get your balance.

On facebook, GET `"/friends"` might be used to retrieve all your friends.

On YouTube, GET `"/video/dogs%20on%20skateboards"` might be used to get all videos about dogs on skateboards.



REST APIs

A SET OF ARCHITECTURAL RULES

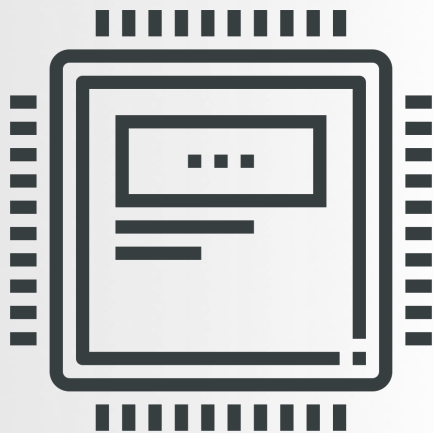
Forgetting the database component for now, the client makes a request:



REST APIs

A SET OF ARCHITECTURAL RULES

...and the server sends a response. This response is usually in JSON format.



Server

Some JSON response



Client

REST APIs

A SET OF ARCHITECTURAL RULES

Before we dive in, the HTTP verbs (or methods) you should know are:

GET - used to retrieve data (browsers make GET requests when you type a URL in the address bar).

POST - used to add new data to a database.

PUT - used to update an existing database entry (like changing a students name).

DELETE - used to remove an entry from the database entirely.





Check-in Time

- Which HTTP verb would you use to when an social media user changes their handle?
- When a user removes a post?
- When a user wants to see their friend list?
- When a user writes a new post?



POSTMAN

CODE-A-LONG

Open your laptop. Code with me. Don't jump ahead.

Download Postman and use each of the 4 http verbs from the previous slide given the documentation below:

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

Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:

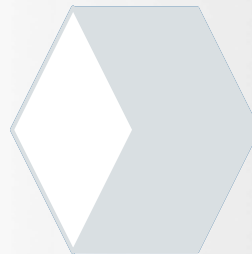
Describe in greater detail the role of a server in a web application.



Use Postman to send request and view responses from a server.



Define API and REST API.



lunch.

Spring_INITIALIZER

We've learned a lot of new terminology and concepts today. Let's slow down and make sure everyone understands what we're about to do.



SPRING INITIALIZER

CODE-A-LONG

Open your laptop. Code with me. Don't jump ahead.

Follow along as we use the Spring Initializer to create our first server.





KeyPoint

We just built a very small REST API. This controller is our Web Layer. Later we will add in a Service Layer and lastly a Data Layer and actual database integration.



Request Parameters

WATCH & LEARN

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

```
@RestController
public class EchoServiceController {

    @RequestMapping(value = "/echo/{input}", method = RequestMethod.GET)
    public String getEcho(@PathVariable String input) {
        return input;
    }
}
```

Request Parameters

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:

- Copying the example seen on the previous slide, start with the Spring_INITIALIZER and build a 'name' controller.
- In this controller, you should have a method called getFullName.
- This method should be mapped to the GET method with a URI value of “/name/{first}/{last}”
- Return the string “<firstname> <lastname>” with a space between them.



**15
minutes!**



Stay Seated & Take 3 Deep Breaths.

RELAX.

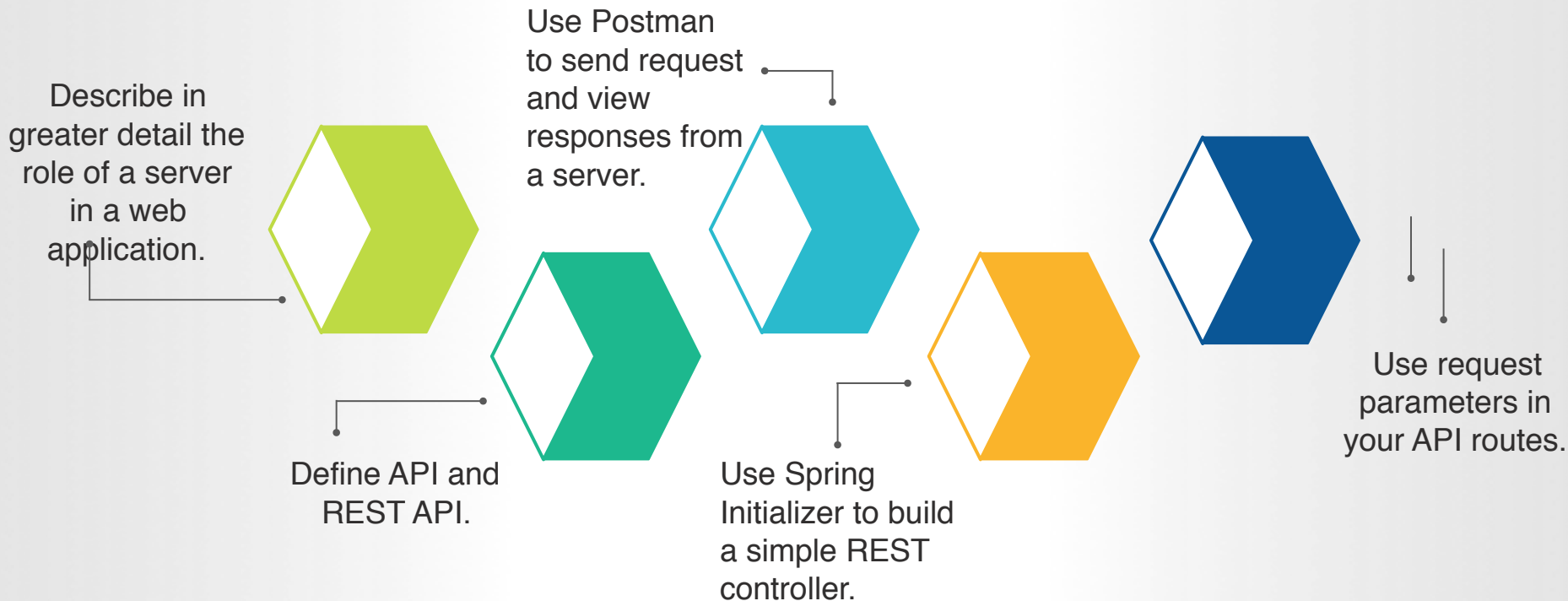
Now take a short walk. Clear your head. After a few minutes break, quickly review your notes.
We'll start back in a few minutes.



Objectives & Key Outcomes

THE TAKEAWAYS FROM THIS CLASS

By the end of class today, you will be able to:



Challenge



Request Parameters

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 class called Motorcycle with the properties String model, int maxSpeed, String model.
- Build a Motorcycle controller.
- Create a class variable to hold a List of Motorcycles
- Create a constructor for the controller.
- Inside the constructor build an ArrayList of Motorcycles (add at least 5).
- In this controller, you should have a method called getMotorcycleById.
- This method should be mapped to the GET method with a URI value of “/motorcycle/{id}”
- This method should return back the Motorcycle object at the index specified by the id parameter.



35
minutes!

Sneak Peek

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



Post Routes

WATCH & LEARN

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

```
@RestController
public class WordController {

    List<String> words = new ArrayList<>();

    @RequestMapping(value = "/words", method = RequestMethod.POST)
    @ResponseStatus(value = HttpStatus.CREATED)
    public String createWord(@RequestBody String word) {
        words.add(word);
        return word;
    }
}
```

Wrap Up

Module 3 Lesson 1

HOMEWORK

You don't have to submit your nightly homework, but you are expected to complete it.

- Go to <https://reqres.in/>, read the documentation and use Postman to make a request to each endpoint
- Read this article on HTTP status codes: <https://www.lifewire.com/http-error-and-status-codes-explained-817986>



No Daily Assessment Today

There is no daily assessment today. Please use this time to ask questions and get individualized help from the instructional staff.

