**ngJava Practice:** [**https://practiceit.cs.washington.edu/problem/view/bjp5/chapter3/s5-Weird**](https://practiceit.cs.washington.edu/problem/view/bjp5/chapter3/s5-Weird)

[**https://www.javaguides.net/p/java-collections-tutorial.html**](https://www.javaguides.net/p/java-collections-tutorial.html)

[**https://www.codegrepper.com/code-examples/javascript/button+click+navigate+to+another+page+angular**](https://www.codegrepper.com/code-examples/javascript/button+click+navigate+to+another+page+angular)

[**https://www.geeksforgeeks.org/html-course-building-main-content-section-3/**](https://www.geeksforgeeks.org/html-course-building-main-content-section-3/)

[**https://www.geeksforgeeks.org/sql-tutorial/?ref=lbp#sql-server**](https://www.geeksforgeeks.org/sql-tutorial/?ref=lbp#sql-server)

[**https://java.meritcampus.com/core-java-topics**](https://java.meritcampus.com/core-java-topics)

[**https://www.java67.com/search/label/core%20java?&max-results=3**](https://www.java67.com/search/label/core%20java?&max-results=3)

[**https://www.javatpoint.com/java-program-to-left-rotate-the-elements-of-an-array**](https://www.javatpoint.com/java-program-to-left-rotate-the-elements-of-an-array)

[**https://www.softwaretestinghelp.com/core-java-interview-questions/**](https://www.softwaretestinghelp.com/core-java-interview-questions/)

**Free Project: https://frontendfun.com/free?page=9**

**Model** in Spring boot- represent a Java object carrying data

The **View** visualize the data that the model contains.

The **Controller** manages the data flaw into model object and updates the view whenever data changes. It keeps view and model separate.

@Entity – is a lightweight persistence domain object that represents a table in a relational database and each entity instance corresponds to a row in that table.

toString() method returns the string representation of the object. If you print any object, java complier internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be state of object etc. depends on your implementation.

@Id annotation specifies the primary key of an entity

@GenerateValue provides for the specification of generation strategies for the value of primary keys

@Column annotation is used to specify the mapped column for a persistent property or field. If no Column annotation is specified, the default value will be applied

@Repository – is a Spring annotation that indicates that the decorated class is a repository. A repository is a mechanism for encapsulating storage, retrieval, and search behavior which emulate the collection of objects.

**JpaRepository** is JPA specific extension of Repository that contains the full API of CrudRepository and PagingAndSortingRepository. So it contains API for basic CRUD operations and also API for pagination and sorting. The goal of Spring Data repository abstraction is to significantly reduce the amount of boilerplate code required to implement data access layers for various persistence stores

**@Service** component are the class file which contains @Service annotation. These class files are used to write business logic in a different layer, separated from @RestController class file

**@Autowired** enables to inject the object dependency implicitly. It internally uses setter or constructor injection. Autowiring can’t be used to inject primitive and string values.

@RestController – RestController annotation in order to simplify the creation of RESTful web services. It’s a convenient annotation that combines @Controller and @ResponseBody, which eliminates the need to annotate every request handling method of the controller class with the @ResponseBody annotation.

@RequestMapping – This annotation maps HTTP requests to handler methods of MVC and REST controller

@RequestParam – is used to read the HTML form data provide by a user and bind it to the request parameter. The model contains the request data and provides it to view page.

@GetMapping annotation maps HTTP GET request onto specific handler methods. It is a composed annotation that acts as a shortcut for @RequestMapping (method = RequestMethod.GET)

**Angular:**

Observables provide support for passing messages between parts of application. It is a technique for event handling, asynchronous programming and handling multiple values

**Observable** are declarative-that is you define a function for publishing values but it does not execute until a consumer subscribes to it. The subscribed consumer then receives notification until the function completes, or until they unsubscribe

**Decorator** that marks as class as an Angular component and provides configuration metadata that determines how the component should be processed, instantiated, and used at runtime.

Components are the most basic UI building block of an Angular app. An Angular app contains a tree of Angular components.

Angular components are a subset of directives, always associated with a template, Unlike other directives, only one component can be instantiated for a given element is a template.

A component must belong to an NgModule in order for it to be available to another component or application. To make it a member of an NgModule, list it in the declarations field of the NgModule metadata.

\*ngFor directive is used to repeat a portion of HTML template one per each item from an iterable list(Collection). The ngFor is an Angular structural directive. Let user creates a local variable that will be available in the template

**Angular Service** – to share the data among various components in an Angular app. To make the Testing and Debugging simple

So a component can delegate certain task to service, such as fetching data from the server, validating user input, or logging directly to the console

To define a class as a service in Angular, use the @Injectable() decorator to provide the metadata that allows Angular to inject it into a component as a dependency.

**Pipe:** Transform bound properties before display. Build in pipes: date, number, decimal, percent, currency, json etc

Example: {{ product.productCode | currency | lowercase}}

**Data Binding:**

* **Data binding makes it easy to display component properties and set DOM element properties from component to better control the view**
* **The component can listen for and respond to events, such as button click**
* **Two way binding we can process user entry for an interactive experience**
* **FourTypes of Data binding**
* **1. Interpolation: {{pageTitle}}🡨 Interpolation** inserts interpolated strings into the text between HTML elements or assigns element properties
* **2. Property Binding: <img [src] = ‘product.imageUrl’> 🡨** Property binding set an HTML element property to the value of the template expression.
* **3. Event Binding: <button (Click> = ‘toggleImage()’> 🡪** Event binding listen the events from the user interface and executes a component method when the event occurs
* **4. Two-Way Binding: <input [(ngModel)] = ‘listFilter’> 🡨🡪** Two way binding displays a component property and updates that property when the user makes a change in an input element.

**Interface:** An interface is a specification identifying a related set of properties and methods. Interface is the implementation of properties and methods we need for a specific feature.

Error: Cross Origin problem : chrome.exe --user-data-dir="C://Chrome dev session" --disable-web-security

Solved: <https://stackoverflow.com/questions/56328474/origin-http-localhost4200-has-been-blocked-by-cors-policy-in-angular7>

if You use spring boot , you should add origin link in the @CrossOrigin annotation

@CrossOrigin(origins = "http://localhost:4200")

@GetMapping("/yourPath")

@CrossOrigin(origins = "http://localhost:4200")

<https://www.youtube.com/watch?v=G46fjVzQ7BQ>

Two Way Binding

Graphical user interface, diagram, application

Description automatically generated

https://spring.io/guides/gs/rest-service-cors/

**Install Json server in Angular:**

npm install -g json-server

npm install --save json-server

**Run Json Server in Angular:**

json-server --wathch db.json

Response Json Sever as following:

Resources

<http://localhost:3000/employee>

Create boiler component command

ng g s services/tutorial

ng g c components/add-tutorial

ng g c components/tutorial-details

ng g c components/tutorials-list

ng g class models/tutorial --type=model

Home

http://localhost:3000

employee.service.ts

Employee-list.component.ts

<https://medium.com/analytics-vidhya/spring-boot-angular-8-crud-example-8aeafd47b54>

<https://getbootstrap.com/docs/4.0/components/card/>

Server

1. Web Container (Container)- Tomcat, GlashFish, Axis

2. Applications Server(Box)- Weblogic, JBoss, Websphear

A picture containing table

Description automatically generated

A picture containing table

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Diagram

Description automatically generated

You can learn about Node.js (learning Node.js is looking extremely promising for the future industry) here: <http://www.nodebeginner.org/>

there are excellent resources at Codecademy to Learn JavaScript (

<https://www.codecademy.com/learn/learn-javascript>

) where you can also learn about HTML, CSS (!important). However you won’t get info about databases, frameworks like React.js and Node.js, deployment services like GoDaddy or Heroku and others which are all integral parts of fullstack Javascript web development.

User Login

Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated