WELCOME BACK
&
THANK YOU

Spring/Spring Boot Crash Course

For TD Bank

MEET YOUR CRASH COURSE TEAM

TANGY F.
CEO

HAL M.
DEVELOPER

WILLIAM
DEVELOPER

AGENDA

DAY 2

1. Recap Day one:
>>Tools to use
>>Rapid review of day one

2. Database Connection
&
CRUD Operations on DB with API's

3. Question and Answer session

Creative Devs. Software

DAY 2

A REMINDER
TOOLS * RESOURCES

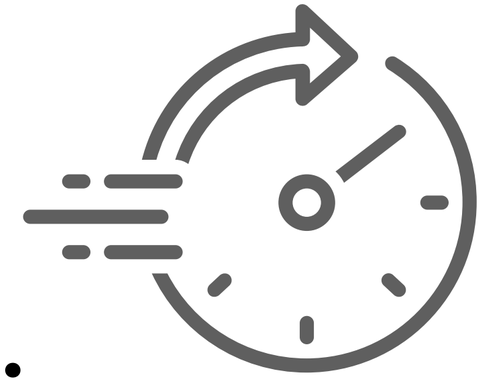
1. We are building a bite size Movie Review app
2. Pre assessment ~ Sent
3. Confluence Page
4. Teams Chat
5. Bitbucket

Creative Devs. Software

Rapid Review

TRIVIA

LESSON 1

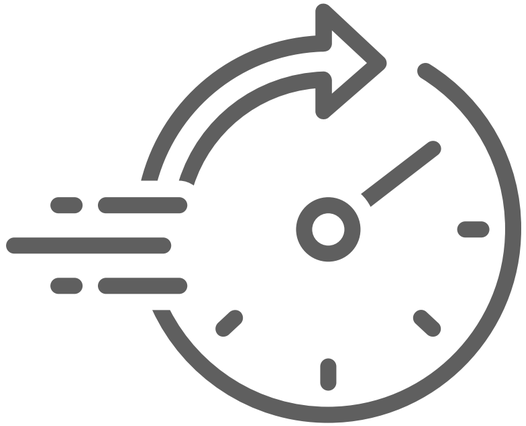


Dependency Injection
REST API
Rapid
Trivia

What is an advantage of using a Dependency Injector over a traditional Dependency Model?

LESSON 1

Rapid Review



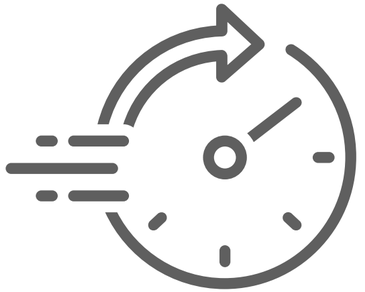
Rapid Review

Dependency Injection REST API

LESSON 1

Rapid Review

Rapid Review



@Autowired is the Spring annotation used for dependency injection.

It allows Spring to handle the

- ✓ **Instantiation**
- ✓ **Loading**
- ✓ **Injection of objects at runtime**
- ✓ **Relieving developers from manually creating & passing objects around.**

By using @Autowired, you can significantly reduce the need to specify properties or constructor arguments explicitly.

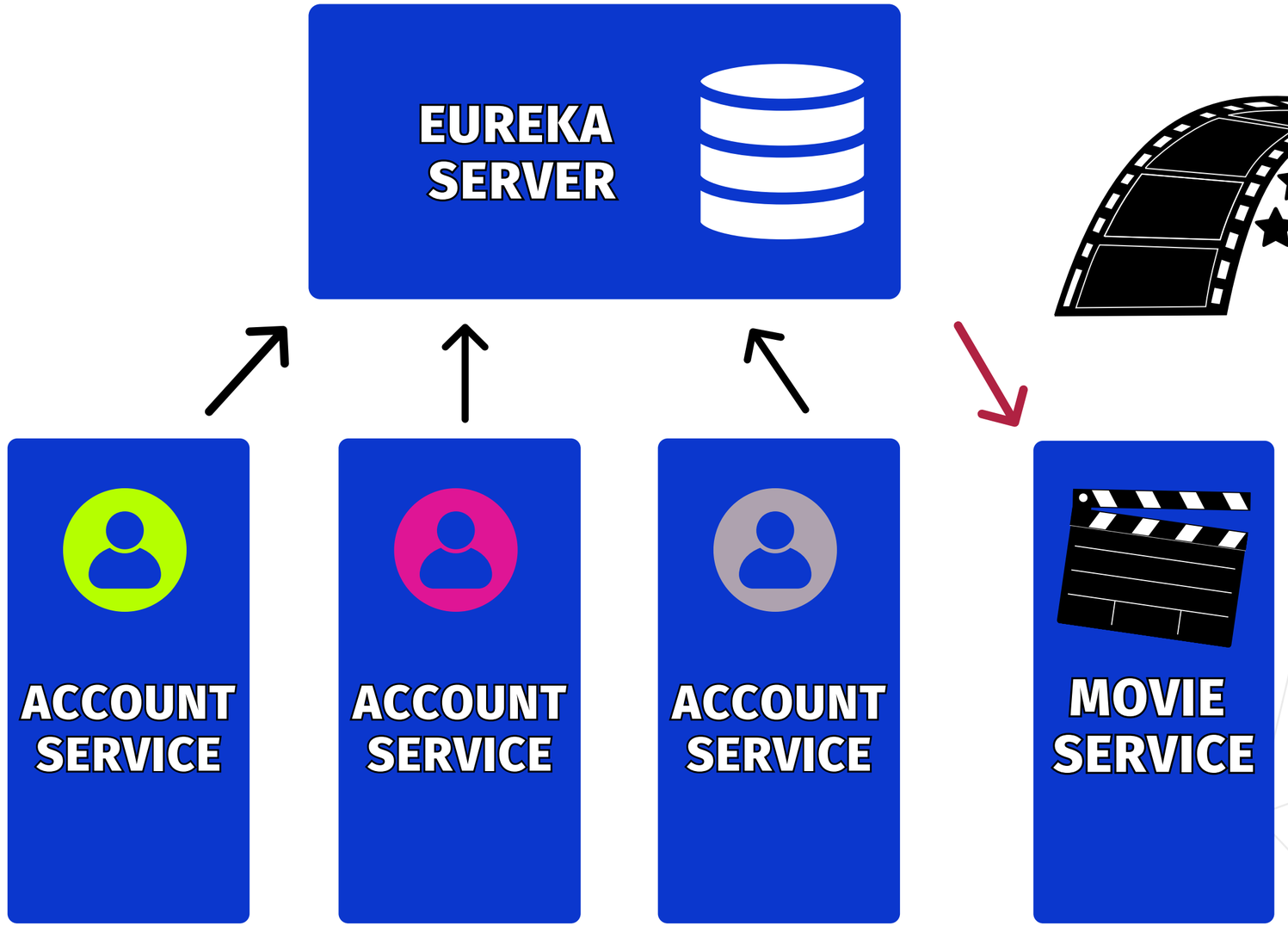
LESSON 2

LET'S GET STARTRED

Database Connection & CRUD Operations on DB with API's

WE ARE BUILDING

BITE SIZE
MOVIE REVIEW APP



Creative Devs. Software

12-FACTOR INTRODUCTION

12 Factor is a set of guiding principles or best practices for building modern, scalable, and maintainable web applications.

These principles serve as a set of guidelines when building cloud-native applications.

By adhering to these principles, developers can create applications that are easier to scale, maintain, and deploy on modern cloud platforms.

12-FACTOR TO WATCH FOR

Factor 3 Config

Watch for where we store our config information, and consider how we could share this code to a repository without exposing sensitive information.

Store config in the environment with configuration files but not in the code

Factor 4 Backing Services

Take note of all the locations that we reference MySQL in the application config.

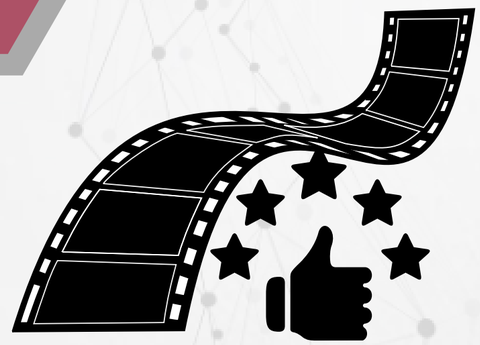
Treat backing services; such as databases, message queues, caches, or third-party APIs, as attached resources

Factor 6 Processes

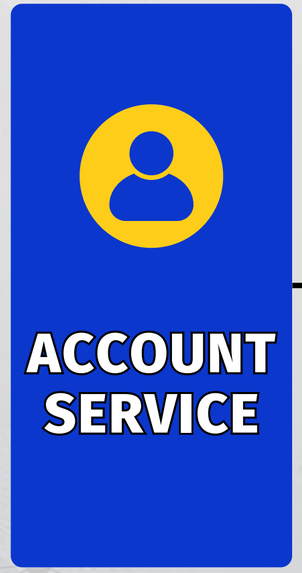
Take note of how the Account entity is stored at the end of each function's processing.

Execute the app as one or more stateless processes, as a stand-alone script without sharing anything. Any data that need to be shared across sessions need to be stored in a stateful backing service like a database.

12 FACTOR IN LESSON 1



BITE SIZE
MOVIE REVIEW APP



Factor 2: Dependencies

Watch for how we declare dependencies, and how we don't require any dependencies to be installed system-wide.

DESIGN PATTERNS INTRODUCED

Design patterns are established, reusable solutions to common software design problems.

They are blueprints that guide developers in structuring their code to achieve flexibility, maintainability, and scalability.

These patterns encapsulate best practices, allowing developers to address recurring challenges in various contexts effectively.

DESIGN PATTERNS USED FOR LESSON 1

BITE SIZE
MOVIE REVIEW APP

BITE-SIZE MOVIE REVIEW APP

Singleton Pattern

Using @Bean and @Autowired annotations ensures that we only have one ModelMapper instance.

Private Class Data Pattern

Prevents the data of our DTOs and entity from being modified after they are created & mapped to.

DESIGN PATTERNS USED IN LESSON 2

Singleton Pattern

Ensures that a class has only one instance and provides a global point of access to that instance.

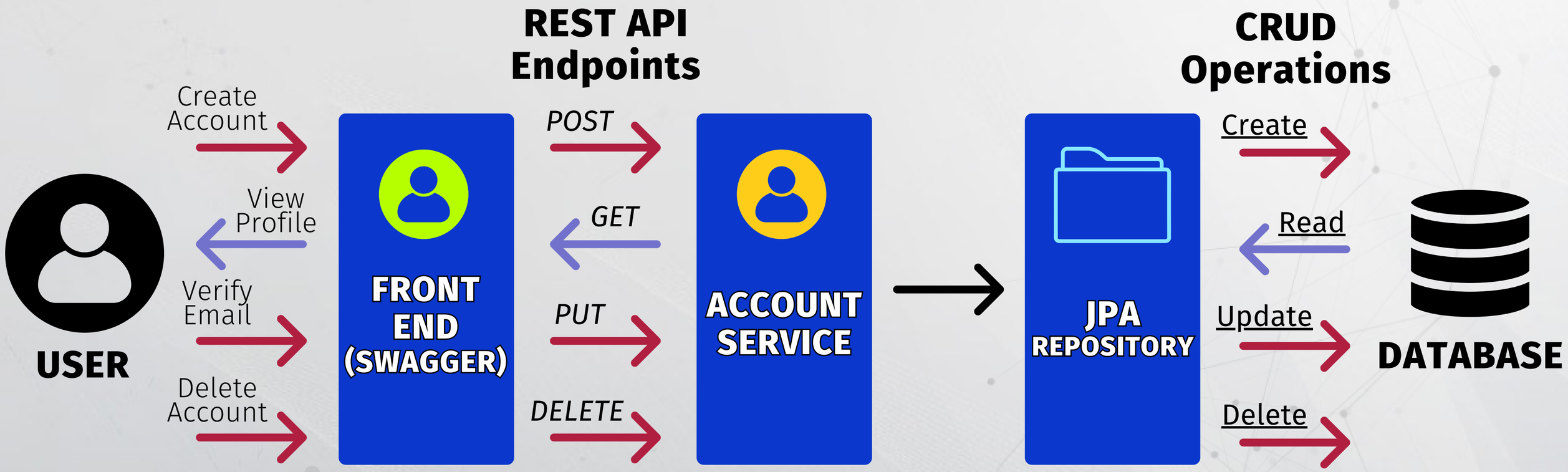
It guarantees that no matter how many times the Singleton class is instantiated, there will always be just one unique instance throughout the application's lifetime.

Chain of Responsibility Pattern

Allows multiple objects to handle a request without knowing which one will ultimately fulfill it. It creates a chain of interconnected handlers, where each handler processes the request and decides whether to pass it to the next handler in the chain or handle it itself.

The pattern promotes loose coupling between the sender and receiver of a request, as the sender only needs to know about the first handler in the chain.

CRUD OPERATIONS WITH JPA



TIME TO CODING #1

Complete the below method stub so that an Account is retrieved from the database, and returned as an AccountResponseDto object. Remember that AccountRepository extends JpaRepository, using Account and Long as type parameters.

```
@Autowired
ModelMapper modelMapper;

public AccountResponseDto getPerson(long id){
    //Code Goes Here
}
```

DB CONNECTION & CRUD OPERATIONS USING REST API

LESSON 2

CRUD Operation

Create (Insert)
Read (Select)
Update
Delete

API Methods

GET
POST
PUT
DELETE
PATCH

Database Accessing

JDBC
JPA
Hibernate

TIME TO CODING #2

Below is the code for our AccountRepository. AccountRepository is an extension of JpaRepository, and we have added a function to find an Account entity in the database using JpaRepository's "find by email" functionality.

The code does not work. What is missing from the below code?

```
@Repository
public interface AccountRepository extends JpaRepository<Account,Long> {
    Account findByEmail(String email);

}
```

THANK YOU

Crash Course

We will see you tomorrow