

Spring, SpringBoot, JPA, Hibernate

Zero To Hero

Agenda of the course

November 2020

WU
BYTES

Introduction Spring Framework

- What is Spring?
- Spring for Java EE
- Features of Spring
- Spring module structure
- Different ways to install Spring

Spring Core

- Beans and ApplicationContext
- Configuration of beans via XML
- Container management and bean lifecycle
- Annotations: Bean classes
- Spring's IoC container: Dependency Injection

Spring MVC

- Introduction to the MVC pattern
- Overview of Spring MVC
- How does Spring MVC work?
- Spring MVC architecture
- How to work with Spring MVC: Spring MVC controller

Agenda of the course

push
bytes

1 September

- What is a vector space and how can we characterise it using linear combinations of vectors
- The vector space of all polynomials of degree at most n

Spring Break

- Why Spring Break?
- A review of linear algebra
- Some applications of linear algebra
- The vector space of all polynomials of degree at most n
- The vector space of all functions
- The vector space of all continuous functions

Spring Semester

- The vector space of all functions
- The vector space of all functions of degree at most n
- The vector space of all functions of degree at most n
- The vector space of all functions of degree at most n
- The vector space of all functions of degree at most n
- The vector space of all functions of degree at most n

Agenda of the course

Introduction

Push
Bytes

Spring 2020

- 1. Introduction to the course
- 2. Introduction to the course
- 3. Introduction to the course
- 4. Introduction to the course
- 5. Introduction to the course
- 6. Introduction to the course
- 7. Introduction to the course
- 8. Introduction to the course

Spring 2021

- 1. Introduction to the course
- 2. Introduction to the course
- 3. Introduction to the course
- 4. Introduction to the course
- 5. Introduction to the course
- 6. Introduction to the course
- 7. Introduction to the course
- 8. Introduction to the course

Spring 2022

- 1. Introduction to the course
- 2. Introduction to the course
- 3. Introduction to the course
- 4. Introduction to the course
- 5. Introduction to the course
- 6. Introduction to the course
- 7. Introduction to the course
- 8. Introduction to the course

Agenda of the course

WU
BYTES

1. INTRODUCTION

- Why we need it
- What is it and how it works
- How to use it
- How to install it

2. INSTALLATION & CONFIGURATION

- How to install it
- How to configure it
- How to use it
- How to install it
- How to use it
- How to install it

3. USAGE

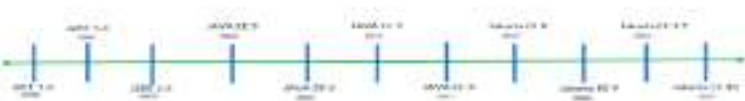
- How to use it
- How to use it
- How to use it
- How to use it
- How to use it
- How to use it



**DEPLOYING
SPRINGBOOT
WEBAPP INTO
CLOUD (AWS)**



JAVA EE RELEASE TIMELINE



- [illegible]

SPRING RELEASE TIMELINE

2024
Release Dates

1.0

2024

2.0

2024

3.0

2024

4.0

2024

5.0

2024

6.0

2024

Spring Release Timeline

- 1. Review current product line and identify areas for improvement. This includes gathering feedback from users, analyzing sales data, and identifying areas for innovation.
- 2. Develop a roadmap for the next 12 months, outlining key features, milestones, and release dates. This roadmap should be flexible and adaptable to changes in the market or technology.
- 3. Prioritize features and projects based on their strategic importance and user demand. Focus on delivering high-quality, reliable products that meet the needs of your target audience.
- 4. Allocate resources and assign team members to specific tasks. Ensure that everyone is clear on their responsibilities and deadlines.
- 5. Implement a regular communication schedule, such as weekly team meetings or bi-weekly status reports, to keep everyone informed and on track.
- 6. Monitor progress and adjust the timeline as needed. Be prepared to pivot if necessary, and celebrate successes along the way.

SPRING CORE

2024
English



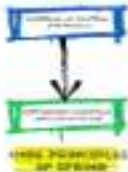
Writing: Analyze the American Dream as it appears in literature, history, and popular culture. Evaluate the impact of the American Dream on American society.

Reading: Read and analyze literary texts, historical documents, and popular culture. Evaluate the impact of the American Dream on American society.

Speaking: Present your findings and conclusions to the class. Engage in a class discussion about the American Dream.

- ✓ Unit 1: The American Dream
- ✓ Unit 2: The American Dream
- ✓ Unit 3: The American Dream
- ✓ Unit 4: The American Dream
- ✓ Unit 5: The American Dream
- ✓ Unit 6: The American Dream

INVERSION OF CONTROL & DEPENDENCY INJECTION



1. Inversion of Control (IoC) is a software design principle, implementation or paradigm which allows to externally control the objects to be created, instead of the way in which objects are being created.
2. IoC is the opposite: control the creation time of a program is handled instead of the programmer controlling the flow of a program. The framework or container takes control of the program flow.
3. Inversion of control is the opposite through which control is turned upside down.
4. Through Inversion, instead of the application or module object is created, it sends down the application for the type of a container. It takes care of creating the object and returns it to the module, the control is then in the container.

ADVANTAGES OF IOC & DI

NOTES
by *ayyappan*

• **Testability**
• **Flexibility**
• **Modularity**

• **Reduced coupling**
• **Increased cohesion**
• **Improved maintainability**

• **Enhanced scalability**
• **Improved performance**
• **Increased security**

• **Reduced development time**
• **Improved code quality**
• **Increased code reuse**

• **Reduced risk of errors**
• **Improved code readability**
• **Increased code maintainability**

• **Reduced code complexity**
• **Improved code maintainability**
• **Increased code reusability**



TIGHT COUPLING
EXAMPLE



LOOSE COUPLING
EXAMPLE

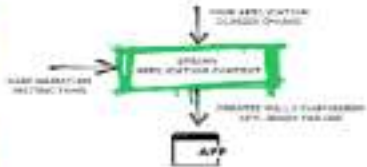
REAL LIFE LOOSE COUPLING EXAMPLE

Spring IOC Container

- The **Spring IOC container** is responsible
 - o to instantiate the application class.
 - o to manage the beans.
 - o to provide the dependencies between the objects.
- There are two types of test containers. They are:
 - o `org.springframework.test.context.junit4.SpringJUnit4ClassRunner`
 - o `org.springframework.test.context.junit5.SpringJUnit5ClassRunner`
- The **Spring container** is an environment where the **Spring** manages the components objects that making up an application.

SPRING IOC CONTAINER

Spring
Engine



MAVEN

ADDING NEW BLANS TO SPRING CONTEXT

NOTES
English

THE NEW BLANS TO THE LEFT ARE THE BLANS THAT ARE
BEING ADDED TO THE SPRING CONTEXT

THE BLANS TO THE RIGHT ARE THE BLANS THAT ARE
BEING REMOVED FROM THE SPRING CONTEXT

SPRING CONTEXT



SPRING CONTEXT



```
1. Add new blans to the spring context  
2. Remove old blans from the spring context  
3. Update the spring context  
4. Save the spring context
```

No Unique Definition & Exception

NOT
A
SYSTEM

When we have a system, we have a set of components that are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.

System components are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.

System components are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.

System components are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.

System components are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.



System components are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.

System components are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.



System components are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.

System components are not unique and are not defined. This is the only way to have a system that is not unique and is not defined.

No Unique and Definition & Exception

NOTU
English

1. **QUESTION** (What is the question being asked?)
2. **ANSWER** (What is the answer to the question?)
3. **EXPLANATION** (Why is this the answer?)
4. **DEFINITION** (What is the definition of the term?)
5. **EXCEPTION** (Are there any exceptions to the rule?)

QUESTION
What is the question being asked?
What is the answer to the question?

ANSWER
What is the answer to the question?
What is the definition of the term?

EXPLANATION
Why is this the answer?
What is the definition of the term?

QUESTION



ANSWER

What is the answer to the question?
What is the definition of the term?



EXPLANATION

Why is this the answer?
What is the definition of the term?

DIFFERENT WAYS TO NAME A BEAN

NOTES
English

In English, beans are commonly referred to by several names, and you might wonder how it all fits together. Here's a breakdown of the different names and how they relate to each other:

Common Bean Names:

- Black Beans:** Also known as **kidney beans** or **frijoles negros**.
- Red Beans:** Often referred to as **haricots rouges**.
- White Beans:** Sometimes called **cannellini beans** or **navy beans**.

Other Bean Names:

- Green Beans:** Also known as **string beans** or **snap beans**.
- Yellow Beans:** Sometimes referred to as **lima beans** or **butter beans**.

Bean Varieties:

- Black Beans:** Common in Mexican and Caribbean cuisine.
- Red Beans:** Often used in soups and stews.
- White Beans:** Popular in Italian and French dishes.



Beans are a staple food in many cultures, and they come in many different colors and shapes. The most common colors are black, red, and white, but there are also green, yellow, and purple beans. Beans are a good source of protein and fiber, and they are easy to cook and eat.

Bean Varieties

There are many different varieties of beans, each with its own unique flavor and texture. Some of the most popular varieties include black beans, red beans, white beans, green beans, yellow beans, and purple beans.

@Primary Annotation

NOTES
English

When you create a new document in the system, you are given a choice of three different document types. The first type is a standard document, which is a document that is used to create a new document. The second type is a document that is used to create a new document. The third type is a document that is used to create a new document.

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document



Document type: Standard document



Document type: Standard document



Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

Document type: Standard document

@Component Annotation



When added to one of the following interfaces, the annotated class is automatically detected by the Spring container and registered as a bean. This is useful for classes that are not managed by the Spring container, but are still managed by the container. This is useful for classes that are not managed by the Spring container, but are still managed by the container.

When the container starts, it scans the classpath for classes annotated with @Component. It then registers these classes as beans in the container.

Example:

```
import org.springframework.stereotype.Component;

@Component
public class MyComponent {
    // ...
}
```

Annotation



Annotation

The Spring container automatically detects and registers the annotated class as a bean. This is useful for classes that are not managed by the Spring container, but are still managed by the container.

Annotation

When the container starts, it scans the classpath for classes annotated with @Component. It then registers these classes as beans in the container.

```
import org.springframework.stereotype.Component;

@Component
public class MyComponent {
    // ...
}
```

Spring Stereotype Annotations

Spring
Annotations

- The `@Component` annotation is used to mark a class as a Spring component.
- The `@Service` annotation is used to mark a class as a Spring service.



@Component is a stereotype annotation that is used to mark a class as a Spring component.

@Service is a stereotype annotation that is used to mark a class as a Spring service.

@Repository is a stereotype annotation that is used to mark a class as a Spring repository.

@Controller is a stereotype annotation that is used to mark a class as a Spring controller.

Model Vs Component

Model
Component



@PreDestroy Annotation



- The `@PreDestroy` annotation can be used to declare a method that should be called before the bean is destroyed. It is used to perform cleanup tasks before the bean is destroyed.
- The `@PreDestroy` annotation is used to declare a method that should be called before the bean is destroyed. It is used to perform cleanup tasks before the bean is destroyed.
- Spring provides the `@PreDestroy` annotation with the `org.springframework.stereotype.annotation` package.



The `@PreDestroy` annotation is used to declare a method that should be called before the bean is destroyed. It is used to perform cleanup tasks before the bean is destroyed.

Example:

Code:

Example of a bean that is destroyed by Spring. The `@PreDestroy` annotation is used to declare a method that should be called before the bean is destroyed.

ADDING NEW BEANS PROGRAMMATICALLY

Spring
Engine

Sometimes we want to register new instances of an object and add them into the Spring context based on a condition. For the sake of this Spring 5 tutorial, a new instance is proposed to create the beans programmatically by passing the method passed inside the context object.

context.registerBean

The registerBean method is used to register a new bean into the Spring context.

context.registerBean("car", Vehicle.class, () -> new Car(), "carSupplier");

The registerBean method is used to register a new bean into the Spring context.

The registerBean method is used to register a new bean into the Spring context.

ADDING NEW BEANS PROGRAMMATICALLY

Spring
Engine

```
getTransactionManager() {  
    return old  
}  
context.registerBeanDefinition("myBean",  
    new GenericBeanDefinition() {  
        {  
            setBeanClass(MyBean.class);  
        }  
    }  
);  
context.refresh();  
context.getBean("myBean");  
}
```



ADDING NEW MEANS USING XML CONFIGS

NOTES
English

In order to add new means of testing, the user must create a new test plan, add the new means to the plan, and then save the plan. The new test plan will be added to the test plan list in the test plan list. The new test plan will be added to the test plan list in the test plan list.

The user must create a new test plan, add the new means to the plan, and then save the plan. The new test plan will be added to the test plan list in the test plan list. The new test plan will be added to the test plan list in the test plan list.



The user must create a new test plan, add the new means to the plan, and then save the plan. The new test plan will be added to the test plan list in the test plan list. The new test plan will be added to the test plan list in the test plan list.

The user must create a new test plan, add the new means to the plan, and then save the plan. The new test plan will be added to the test plan list in the test plan list. The new test plan will be added to the test plan list in the test plan list.

BEHIND THE SCENES OF A WEB APP

NOTES
English



WHY SHOULD WE USE FRAMEWORKS?

NOTES
English



IDEAS COME

They can easily come up and disappear.
But when you think about it, it's not that easy.



They can be hard to find.



They can be hard to find.



They can be hard to find.



They can be hard to find.



They can be hard to find.



IDEAS COME

They can be hard to find and disappear.
But when you think about it, it's not that easy.



They can be hard to find.



They can be hard to find.



They can be hard to find.



They can be hard to find.



They can be hard to find.

WHY SHOULD WE USE FRAMEWORKS?

NOTES



DEV SIDE

How fast you can build and improve
the better the framework, the better it is



1. Faster development and deployment



2. Consistent code structure



3. Easier to learn and use



4. Better performance



5. Easier to maintain



USER SIDE

How fast you can build and improve
the better the framework, the better it is



1. Faster development and deployment



2. Consistent code structure



3. Easier to learn and use



4. Better performance



5. Easier to maintain

SPRING PROJECTS

WUOLAH
WUOLAH



SPRING PROJECTS

WASH STATE



WASH STATE



SPRING PROJECTS



SPRING PROJECTS



SPRING PROJECTS



SPRING PROJECTS



SPRING PROJECTS



SPRING PROJECTS



SPRING PROJECTS

WASH STATE

INTRODUCTION TO BEANS WIRING INSIDE SPRING

NOTES
by [illegible]

1. This document is intended to provide a high-level overview of the Spring framework's internal wiring process. It is not a comprehensive guide to the framework's internals, but rather a starting point for those interested in understanding how the framework works under the hood.

2. The Spring framework is a powerful and flexible tool for building Java applications. It provides a wide range of features, including dependency injection, transaction management, and caching. The framework's internal wiring process is a complex and intricate process, but understanding it can help you to better utilize the framework's features and to troubleshoot any issues that may arise.



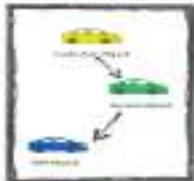
INTRODUCTION TO BEANS WIRING INSIDE SPRING

NOTES
English

SPRING TESTED WITH
WIRING



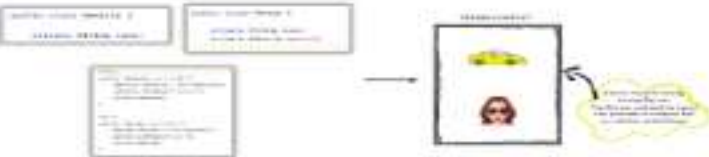
SPRING TESTED WITH
WIRING & AIR



NO WIRING SCENARIO INSIDE SPRING

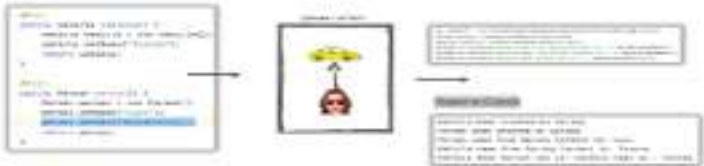
NOT
A
WIRING

They also can be used to create a new scenario. The scenario can be created by using the scenario wizard or by creating a new scenario from scratch. The scenario wizard is a tool that helps you create a new scenario. It is a wizard that guides you through the process of creating a new scenario. It is a wizard that guides you through the process of creating a new scenario. It is a wizard that guides you through the process of creating a new scenario.



WIRING BEANS USING METHOD CALL

- When the bean is created, it can be wired by using the constructor, a setter method, or a method call. In this example, we will use the method call to wire the bean.
- Let's see how to wire the bean using the method call.



WIRING BEANS USING METHOD PARAMETERS

SPRING
5/24/20

- Once the application runs, you can print details of loaded beans using `ApplicationContext.getBeanNames()` method.
- Spring supports the use of `Method` parameter for wiring beans.
- Spring will create beans using `Method` parameter if the bean is annotated with `@Autowired` and `@Qualifier`.

```
import org.springframework.context.annotation.*;
import org.springframework.stereotype.*;

@Component
public class Person {
    String name;
    int age;
    String address;
}

@Component
public class PersonService {
    Person person;
    PersonService(Person person) {
        this.person = person;
    }
}
```



```
import org.springframework.context.annotation.*;
import org.springframework.stereotype.*;

@Component
public class PersonService {
    Person person;
    PersonService(Person person) {
        this.person = person;
    }
}
```

PersonService

```
import org.springframework.context.annotation.*;
import org.springframework.stereotype.*;

@Component
public class PersonService {
    Person person;
    PersonService(Person person) {
        this.person = person;
    }
}
```

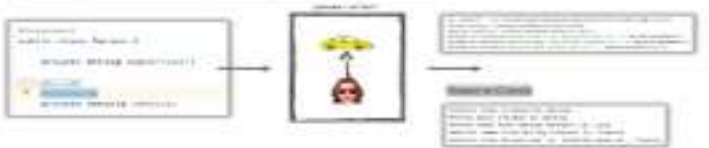

Inject beans using @Autowired on class fields

spring
oxygen

The first bean is the bean that is injected into the second bean. It is the bean that is injected into the second bean. It is the bean that is injected into the second bean.

The second bean is the bean that is injected into the first bean. It is the bean that is injected into the first bean. It is the bean that is injected into the first bean.

The third bean is the bean that is injected into the first bean. It is the bean that is injected into the first bean. It is the bean that is injected into the first bean.



It is the bean that is injected into the first bean. It is the bean that is injected into the first bean. It is the bean that is injected into the first bean.

inject beans using @Autowired on setter method



- 1. We will demonstrate how to inject beans using the `@Autowired` annotation on a setter method.
- 2. We will create a `Person` class and a `PersonService` class.
- 3. We will create a `Person` class and a `PersonService` class.

```
PersonService.java
import org.springframework.stereotype.Service;

@Service
public class PersonService {

    @Autowired
    private Person person;

    // ...
}
```



```
Person.java
import org.springframework.stereotype.Component;

@Component
public class Person {

    // ...
}
```

```
PersonService.java
import org.springframework.stereotype.Service;

@Service
public class PersonService {

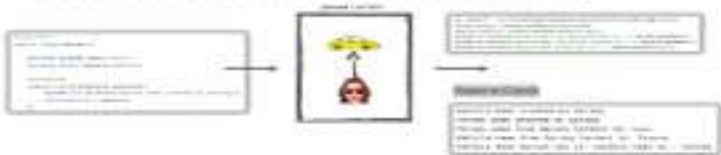
    @Autowired
    private Person person;

    // ...
}
```

Inject beans using @Autowired with constructor



- 1. We will create a simple Spring application using Maven and we will use the Spring Boot framework to create a simple application.
- 2. We will create a simple Spring application using Maven and we will use the Spring Boot framework to create a simple application.
- 3. We will create a simple Spring application using Maven and we will use the Spring Boot framework to create a simple application.



How AutoPilot works with multiple beams of same type

max
bytes

1. By default, AutoPilot works with one type of beam (e.g. one type of sensor) to detect objects in the environment.
2. If the beam type is changed, the system will automatically detect the change and will not work properly (e.g. it will not detect objects).
3. In the case of multiple beams of the same type, the system will work properly and will not detect objects in the environment.

STEP 1:

Step 1: Create a beam

beam = Beam(beam_type=1, beam_id=1, beam_name='beam_1')

beam = Beam(beam_type=1, beam_id=1, beam_name='beam_1')

beam = Beam(beam_type=1, beam_id=1, beam_name='beam_1')

beam = Beam(beam_type=1, beam_id=1, beam_name='beam_1')

STEP 1

beam_id=1



How Autowiring works with multiple beans of same type

step 0/100

- 1. In this example, we have two beans of type `Person` in the container. The first bean is named `person1` and the second bean is named `person2`.
- 2. We have a `Person` interface and a `Person` class. The `Person` class has a `name` attribute and a `getAge` method. The `Person` interface has a `getAge` method.

Person.java

```
public class Person {
```

```
    private String name;
    private int age;
```

```
    public Person {
```

```
        // constructor
        // constructor
        // constructor
    }
}
```



Person.java



STEP 2

How Autowiring works with multiple beans of same type

many
bytes

- When autowiring, Spring looks for all beans of the required type in the container. If there are multiple beans of the same type, Spring will throw an exception, stating that there are multiple beans of the same type.
- Spring will throw an exception, stating that there are multiple beans of the same type.

Example:

ApplicationContext (Context)

Context (Context)

Context

Context (Context) Context (Context)
Context (Context) Context (Context)
Context (Context) Context (Context)



Context (Context)



STEP 3

Indexing & Archiving (journal departments)



1. **Self-organizing systems:** In the **beginning**, I looked at self-organizing systems as a way of thinking about complex systems that are not designed.
2. **Complexity and self-organization:** In the **beginning**, I looked at self-organizing systems as a way of thinking about complex systems that are not designed.
3. **Self-organizing systems:** In the **beginning**, I looked at self-organizing systems as a way of thinking about complex systems that are not designed.



© 2000 Blackwell Science Ltd *Journal of Internal Medicine* 247: 399–406

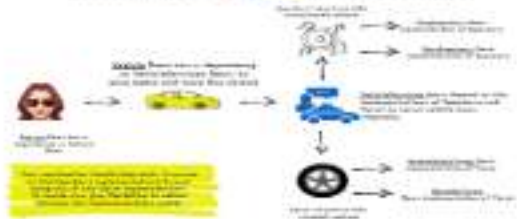
breaking point

Copyright © 2007 by Humana Press
All rights reserved. No part of this work may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or by any information storage or retrieval system, without permission in writing from Humana Press.



ASSIGNMENT RELATED TO BEANS AUTOMOBILES, DE

BRUNN
DUKE



Bean Scopes inside Spring

1 Singleton

2 Prototype

3 Session

4 Asynchronous

5 Custom



SINGLETON BEAN SCOPE

SSM
DJ143

- 1. Whenever the container creates or retrieves a bean, it first checks for a single bean instance of that class in its cache. If it finds one, it returns that instance. Otherwise, it creates a new instance and stores it in the cache.
- 2. Unlike Prototype beans, which are created each time they are requested, a Singleton bean is created only once. Subsequent requests for the same bean return the same instance. This is useful for beans that are expensive to create or that should be shared across the application.



RACE CONDITION

DATA
TYPE

When two or more processes access shared resources at the same time, the race condition can occur. The outcome depends on the sequence of the execution of the processes. This is a problem because the outcome is not predictable. In the worst case, the value can be 0, which is not the intended result. The value of the shared resource can be changed by the processes. This is a problem because the value of the shared resource can be changed by the processes.

```
1 // Shared resource (e.g., counter)
2 int shared_resource = 0;
3
4 // Process 1
5 void process1() {
6     // Increment shared resource
7     shared_resource++;
8 }
9
10 // Process 2
11 void process2() {
12     // Decrement shared resource
13     shared_resource--;
14 }
```



Singleton Pattern and Issues

- Ensuring there is only one instance of a Singleton class is the main purpose of the Singleton pattern. It is also possible that there is more than one instance.
- This design is often used in programming languages that do not support a Singleton pattern.

2

Building a Singleton class is a common design pattern, and it is used in many applications. It is a design pattern that ensures that there is only one instance of a class.

1

It is a design pattern that ensures that there is only one instance of a class. It is a design pattern that ensures that there is only one instance of a class.

1

It is a design pattern that ensures that there is only one instance of a class. It is a design pattern that ensures that there is only one instance of a class.

EAGER & LATE INSTANTIATION

BRUNN
DUTCH

- **Eager** loading will process all the dependencies of any module before the module is ever used in your code. This is done **immediately**.
- **Late** loading will instead defer the loading of the module until it is actually used in your code. This approach is **lazy**.

Example of eager loading:

```
class User {  
  // ...  
  static class Profile {  
    // ...  
  }  
}
```



Example of late loading:

```
class User {  
  // ...  
  static class Profile {  
    // ...  
  }  
}
```



```
class Profile {  
  // ...  
  static class User {  
    // ...  
  }  
}
```

Eager Vs Lazy

Memory Management

Eager Garbage Collection



Garbage collection is performed at a specific time, regardless of whether there is any garbage to be collected.



Garbage collection is performed at a specific time, regardless of whether there is any garbage to be collected.



Garbage collection is performed at a specific time, regardless of whether there is any garbage to be collected.



Garbage collection is performed at a specific time, regardless of whether there is any garbage to be collected.



Garbage collection is performed at a specific time, regardless of whether there is any garbage to be collected.



Lazy Garbage Collection



Garbage collection is performed only when there is garbage to be collected.



Garbage collection is performed only when there is garbage to be collected.



Garbage collection is performed only when there is garbage to be collected.



Garbage collection is performed only when there is garbage to be collected.



Garbage collection is performed only when there is garbage to be collected.

Singleton Vs Prototype

Design
Patterns

Singleton



There is a single instance of the class.



Provides global access to the instance.



Ensures that only one instance of the class is created.



Used when a class has a static state.



Used in many cases.



Prototype



Creates a new instance of the class.



Provides a way to create a new instance of the class.



Used when a class has a dynamic state.



Used in many cases.



Used in many cases.

Aspect-Oriented Programming

- Aspects are **cross-cutting concerns** that affect the behavior of an application across multiple modules or components.
- They are used to **encapsulate** cross-cutting concerns, such as logging, security, and transaction management, into reusable modules that can be applied to different parts of the application.

2

AOP provides the ability to dynamically join the crosscutting concerns (aspects) to the original code at runtime, without modifying the original code.

1

AOP allows you to modularize crosscutting concerns, such as logging, security, and transaction management.

3

AOP is a programming paradigm that enables you to modularize crosscutting concerns by defining them separately and applying them to specific parts of the application without modifying the original code.

Figure 1. Schematic representation of the experimental design. The figure shows a timeline of the experiment. The timeline starts with a baseline period (BL) and is divided into three main phases: Pre-Test, Test, and Post-Test. The Pre-Test phase includes a familiarization session (FAM) and a pre-test session (PRE). The Test phase includes a test session (TEST) and a post-test session (POST). The Post-Test phase includes a post-test session (POST) and a follow-up session (FOLLOW-UP). The timeline also shows the duration of each session and the time intervals between sessions.

AOP Design

- **Aspects** are modules that capture the logic for **crosscut concerns** (e.g., logging, security, etc.)
- **Aspects** are **advised** to **advise** **advice** (e.g., logging, security, etc.)
- **Aspects** are **advised** to **advise** **advice** (e.g., logging, security, etc.)
- **Aspects** are **advised** to **advise** **advice** (e.g., logging, security, etc.)

2

Aspects are **advised** to **advise** **advice** (e.g., logging, security, etc.)

1

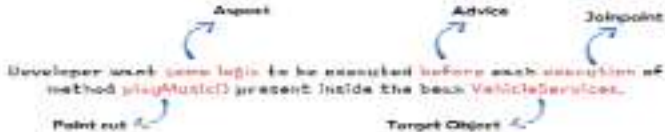
Aspects are **advised** to **advise** **advice** (e.g., logging, security, etc.)

3

Aspects are **advised** to **advise** **advice** (e.g., logging, security, etc.)

- **Aspects** are **advised** to **advise** **advice** (e.g., logging, security, etc.)
- **Aspects** are **advised** to **advise** **advice** (e.g., logging, security, etc.)

Typical Scenarios of AOP implementation



RELAYING INSIDE ACP

WASH
D.C. 20540

Relay only with a valid ID number to be used for
relaying (see the below)



Relay only with a valid ID number to be used for
relaying (see the below) to be used for relaying (see the below)
Relay only with a valid ID number to be used for relaying (see the below)

ADVICE TYPES INSIDE AOP

spring
aop

Type of Advice in Spring AOP

- `@Before`
- `@AfterReturning`
- `@AfterThrowing`
- `@After`
- `@Around`

1

Before advice runs before a method execution.

2

After returning advice runs after a method execution returns normally.

3

After throwing advice runs after a method execution returns by throwing an exception.

4

After advice runs after the method execution and returns its value.

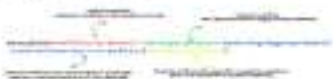
5

Around advice runs before & after a method execution. It can do anything before the method execution and after the method runs and is returning, return data and even it can skip the execution path, to do or not.

CONFIGURING SERVICES INSIDE AOP

WASH
03/03

- The default AOP configuration is based on the default settings of the AOP configuration tool. The default settings are:
 - The AOP configuration tool is installed on the AOP configuration tool.
 - The AOP configuration tool is installed on the AOP configuration tool.



WASH
03/03

- The AOP configuration tool is used to configure the AOP configuration tool. The AOP configuration tool is used to configure the AOP configuration tool.

WASH
03/03

The AOP configuration tool is used to configure the AOP configuration tool. The AOP configuration tool is used to configure the AOP configuration tool.

WASH
03/03

The AOP configuration tool is used to configure the AOP configuration tool. The AOP configuration tool is used to configure the AOP configuration tool.

CONFIGURING SERVICES INSIDE AOP

main
01/04/20

- **Service's configuration is defined in a separate file inside the AOP. The service's configuration is written in a format:**

```
service {
  type string "my-service"
  type string "my-service"
  type string "my-service"
}
```

Step 1: Configuration



```
service {
  type string "my-service"
  type string "my-service"
  type string "my-service"
}
```

Step 2: Service's configuration is written in a separate file inside the AOP. The service's configuration is written in a format:

```
service {
  type string "my-service"
  type string "my-service"
  type string "my-service"
}
```

Step 3: The service's configuration is written in a separate file inside the AOP. The service's configuration is written in a format:

OVERVIEW OF A WEB APP

main
types



1. The Web Client (Browser, Mobile, Tablet) sends a Request to the Web Server.

1. The Web Client sends a Request to the Web Server.
2. The Web Server sends a Response to the Web Client.
3. The Web Server sends a Response to the Web Client.
4. The Web Server sends a Response to the Web Client.

ROLE OF SERVLET INSIDE WEB APP



Servlet Life Cycle

1. When Spring container runs at startup it scan various annotations and configurations in the classes and creates them as servlets. Life cycle
2. When the client sends a request, it invokes public method of the servlet associated with the given url client request.
3. The servlet uses the request on the resource and builds the response that it sends back to the client.

Web Service

1. When Spring it defines a service called **DispatcherServlet** which manages all the requests coming from a web application.
2. This servlet intercepts every request that comes for any url in request, passing the request to change the request and the response. This way Spring internally does all the work for the request with out the need of defining the service inside a web app.



EVOLUTION OF WEB APPS INSIDE JAVA ECO SYSTEM

www.dynabiz.com



approach 1

- Easy to get started with all the tools of the J2EE stack (JSP and EJBs)
- Easy to get started with all the tools of the J2EE stack (JSP and EJBs)
- Spring Framework (Spring Core, Spring MVC, Spring ORM, Spring AOP, Spring Security, Spring Test)



approach 2

- Easy to get started with all the tools of the J2EE stack (JSP and EJBs)
- Easy to get started with all the tools of the J2EE stack (JSP and EJBs)
- Spring Framework (Spring Core, Spring MVC, Spring ORM, Spring AOP, Spring Security, Spring Test)

SPRINGBOOT: THE HERO OF SPRING FRAMEWORKS



1

Spring Boot was introduced in April 2014 to reduce some of the burdens while developing a Java web application.

2

Before SpringBoot, developers had to configure a number of tedious modules like, like Tomcat and DispatcherServlet, defined lots of classes, and a lot of dependencies.

3

But with Spring Boot, you can create your Spring MVC web or services in as little as 2-3 lines of code, without configuring all the configurations we needed in the

4

Spring Boot to reduce some of the most complicated problems in the Spring ecosystem. It helps to bootstrap Spring applications with easily added features in the business world.

5

SpringBoot is a revolutionary step taken due to the fact that developers had to use the components like DispatcherServlet, Tomcat, and ServletContext, that had lots

BEFORE & AFTER SPENDING OUT

Small
Dykes



Providence's ~~Massive~~ Thriftly project
cannot be implemented properly



Implementation schedule kept in
congruence with ~~business~~ business
needs (not met)



Delays in the early application delivery
system, leaving the team in the
lurch



Overly complex ~~complex~~ complex including
untested, unproven processes
and rolling back



Spring 2007 ~~unannounced~~
configuration ~~the~~ improvements
implemented as a Spring
application



Spring 2007 ~~unannounced~~ ~~announced~~ a
new version of the web-based
module as a Spring application
release



Spring 2007 ~~unannounced~~ ~~announced~~ a
new version of the web-based
module as a Spring application
release

SpringBoot Important features

SpringBoot Starter

SpringBoot provides a set of starter dependencies that can be used to quickly bootstrap a new application. These starters are designed to be used as a starting point for building a new application, and they provide a set of default configurations that can be customized as needed.

Example: SpringBoot Starter Web

Autoconfiguration

SpringBoot provides a set of autoconfiguration classes that can be used to automatically configure the application. These classes are designed to be used as a starting point for building a new application, and they provide a set of default configurations that can be customized as needed.

Example: SpringBoot Starter Web

Actuator & Dev Tools

SpringBoot provides a set of actuator endpoints that can be used to monitor the application. These endpoints are designed to be used as a starting point for building a new application, and they provide a set of default configurations that can be customized as needed.

Example: SpringBoot Starter Web

Quick Recap

- 1. **Application profile** is the primary object that we want to generate test scenarios. It is based on the statements required for an application.
- 2. We can identify the Spring Boot main classes by looking for `@SpringBootApplication`.
- 3. A single `@SpringBootApplication` class is associated with the root context of the Spring Boot application. It is:

 - **Configuration class** because Spring Boot is auto-configuration framework.
 - **Component** because `@SpringBootApplication` is a stereotype annotation. The application is based.
 - **Configuration class** because registration of properties in the context of the Spring Boot application is done. For example, Spring is using `@Configuration` annotation.



INTRODUCTION TO THYMELEAF

THYME
0.1.0

Thymeleaf is a modern, lightweight, and powerful templating engine for Java web applications. It is designed to be easy to use, fast, and secure.

Thymeleaf has a rich ecosystem of extensions, including Thymeleaf Spring, Thymeleaf Maven, and Thymeleaf Gradle.

This document is a quick start guide for Thymeleaf. It covers the basic concepts, installation, and usage of Thymeleaf. For more detailed information, please refer to the Thymeleaf documentation.

THYMELEAF

THYMELEAF

THYMELEAF

```
1. THYMELEAF is a modern, lightweight, and powerful templating engine for Java web applications. It is designed to be easy to use, fast, and secure.
```

THYMELEAF

THYMELEAF

THYMELEAF

THYMELEAF is a modern, lightweight, and powerful templating engine for Java web applications. It is designed to be easy to use, fast, and secure.

```
2. THYMELEAF has a rich ecosystem of extensions, including Thymeleaf Spring, Thymeleaf Maven, and Thymeleaf Gradle.
```

```
3. THYMELEAF is a modern, lightweight, and powerful templating engine for Java web applications. It is designed to be easy to use, fast, and secure.
```

THYMELEAF

THYMELEAF

THYMELEAF



Thymeleaf is a modern, lightweight, and powerful templating engine for Java web applications. It is designed to be easy to use, fast, and secure.

SPRINGBOOT DEVTOOLS

spring
boot

Spring Boot simplifies the development of Spring applications by providing a set of default configurations and conventions that allow you to get up and running quickly.

It also provides a set of tools and utilities that make it easier to develop and test your applications.

Dependencies

Spring Boot simplifies the management of dependencies by providing a set of default configurations and conventions that allow you to get up and running quickly.

Configuration

Spring Boot simplifies the configuration of your application by providing a set of default configurations and conventions that allow you to get up and running quickly.

Spring Boot simplifies the management of dependencies by providing a set of default configurations and conventions that allow you to get up and running quickly.

Spring Boot simplifies the configuration of your application by providing a set of default configurations and conventions that allow you to get up and running quickly.

Spring Boot simplifies the management of dependencies by providing a set of default configurations and conventions that allow you to get up and running quickly.

Spring Boot simplifies the configuration of your application by providing a set of default configurations and conventions that allow you to get up and running quickly.

INTRODUCTION TO MVC PATTERN

www.dytel.com

Controller

Coordinates flow & directs
other components to
perform a request
and response action

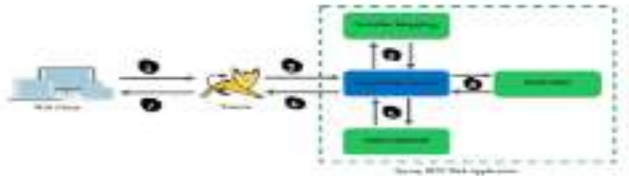
View

Responsible for displaying
data, controls and controls
to be displayed in the
presentation

Model

Responsible for the
business logic, data,
validation, etc. of the
application





SPRING MVC ARCHITECTURE & INTERNAL FLOW

SPRING
5/2/2020

- 1. This class is called **WebFlux** component.
- 2. The **WebFlux** component is used to create the **WebFlux** component in the **Spring** framework. It is **WebFlux** component.
- 3. The **WebFlux** component is used to create the **WebFlux** component in the **Spring** framework. It is **WebFlux** component.
- 4. The **WebFlux** component is used to create the **WebFlux** component in the **Spring** framework. It is **WebFlux** component.
- 5. The **WebFlux** component is used to create the **WebFlux** component in the **Spring** framework. It is **WebFlux** component.
- 6. The **WebFlux** component is used to create the **WebFlux** component in the **Spring** framework. It is **WebFlux** component.
- 7. The **WebFlux** component is used to create the **WebFlux** component in the **Spring** framework. It is **WebFlux** component.

Some data might be lost (e.g., if the data is not saved to disk or if the data is not saved to a file). This is a common problem for any program that uses data.

Instructions

Write a program that reads a file and prints the contents of the file.

```
1 # Read a file and print the contents of the file.
2 # The file is named "data.txt" and is located in the
3 # same directory as the program.
4 # The program should print the contents of the
5 # file to the standard output.
```



REDUCING BOILERPLATE CODE WITH LAMBDA

main
bytes

1. **Goal:** Reduce boilerplate code in the `main` method of the `Calculator` class.

2. **Approach:** Use lambda expressions to simplify the code and make it more concise.

3. **Result:** The code is more readable and easier to maintain.

Calculator.java

```
import java.util.function.BiFunction;
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.UnaryOperator;
```

1. **Goal:** Reduce boilerplate code in the `main` method of the `Calculator` class.

2. **Approach:** Use lambda expressions to simplify the code and make it more concise.

```
import java.util.function.BiFunction;
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.UnaryOperator;
```

3. **Result:** The code is more readable and easier to maintain.

```
import java.util.function.BiFunction;
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.UnaryOperator;
```


@RequestForm Annotation

spring
bytes

Spring - Controller - Controller - **Controller**

```
@RequestMapping("/users")
@Controller
public class UserController {

    @RequestMapping("/add")
    public void add() {
        // ...
    }
}
```

The **@RequestMapping** annotation is used to map a URL to a controller method. It is used to map a URL to a controller method.

The **@RequestMapping** annotation is used to map a URL to a controller method. It is used to map a URL to a controller method.

The **@RequestMapping** annotation is used to map a URL to a controller method. It is used to map a URL to a controller method.

@PathVariable Annotation

Spring
Day 63

- The `@PathVariable` annotation is used to extract the value from the URI of a REST endpoint and bind it to a method parameter. The URI template variable is used to identify the variable in the URI.
- The `@PathVariable` annotation is used to bind the value of the URI variable to the method parameter.

```
@RequestMapping("/books/{id}")  
public ResponseEntity<Book> getBook(@PathVariable("id") String id)  
{  
    // ...  
}
```

URI
Variable

```
@RequestMapping("/books/{id}")  
public ResponseEntity<Book> getBook(@PathVariable("id") String id)  
{  
    // ...  
    return ResponseEntity.ok(book);  
}
```

1

The `@PathVariable` annotation is used to bind the value of the URI variable to the method parameter. The `@PathVariable` annotation is used to bind the value of the URI variable to the method parameter.

VALIDATION WITH SPRING BOOT

spring
boot

- 1. **Spring Validation** is a framework for validating objects in a Spring application.
- 2. **Spring Validation** is a framework for validating objects in a Spring application.

```
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

@Validated
@RestController
public class UserController {

    @GetMapping("/users/{id}")
    public User getUser(@PathVariable Long id) {
        // ...
    }
}
```

Spring Validation is a framework for validating objects in a Spring application.

Spring Validation is a framework for validating objects in a Spring application.

Spring Validation is a framework for validating objects in a Spring application.

Spring Validation is a framework for validating objects in a Spring application.

Important Validation Annotations

Validating Strings

```
String username = "johndoe";  
@NotBlank  
@Email  
@Size(min = 5, max = 30)
```

Annotations for validating strings

Validating Numbers

```
Integer age = 25;  
@Positive  
@Min(value = 18)  
@Max(value = 120)
```

VALIDATION WITH SPRING BOOT

```

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}

```


Bean Scopes inside Spring

1 Singleton

2 Prototype

3 Application

4 Session

5 Web

6 Async

7 Custom

8 ...

How Spring works

- 1. `ApplicationContext`
- 2. `WebApplicationContext`
- 3. `AnnotationConfigWebApplicationContext`

1

ApplicationContext - Abstract interface and base class of all other contexts. It has the contract for all the beans. The beans are defined with the help of annotations. It is the base class.

2

WebApplicationContext - It is a sub-interface of `ApplicationContext`. It is used to define the beans that are used in the web application. It is a sub-interface of `ApplicationContext`.

3

AnnotationConfigWebApplicationContext - This class is used to define the beans that are used in the web application. It is a sub-interface of `WebApplicationContext`.

Key points of Spring Web scopes

Request Scope

- Spring associates an **instance** of the object to the **current request** for every request. The object is destroyed at the end of the request.
- When Spring creates a set of instances, it automatically registers them with the **request-scoped** scope. When creating the scope.
- Use the `request-scoped` scope to create objects that live inside the request object, such as `pageController`.

Session Scope

- An **instance** of the object is created when the user starts the request and lives until the end of the session.
- When creating the scope, it automatically registers the objects with the `session-scoped` scope. When creating the scope.
- Use the `session-scoped` scope to create objects that live inside the session object, such as `sessionController`.

Application Scope

- An **instance** of the object is created when the application starts and lives until the application ends.
- When creating the scope, it automatically registers the objects with the `application-scoped` scope. When creating the scope.
- Use the `application-scoped` scope to create objects that live inside the application object, such as `applicationController`.

- 1. Spring Security is a framework for securing applications using Spring. It is an open-source project that provides a comprehensive set of tools for securing applications.
- 2. It is a cross-platform framework that supports a wide range of applications, including web, mobile, and desktop.

Installation

```

<!-- https://mvnrepository.com/artifact/org.springframework.security/spring-security-core -->
<dependency>
<groupId>org.springframework.security</groupId>
<artifactId>spring-security-core</artifactId>
<version>5.3.0</version>
</dependency>

```

Configuration

Spring Security is a framework for securing applications using Spring. It is an open-source project that provides a comprehensive set of tools for securing applications.

Spring Security is a framework for securing applications using Spring. It is an open-source project that provides a comprehensive set of tools for securing applications.

Spring Security is a framework for securing applications using Spring. It is an open-source project that provides a comprehensive set of tools for securing applications.

Spring Security is a framework for securing applications using Spring. It is an open-source project that provides a comprehensive set of tools for securing applications.



Heute lernen:

- die verschiedenen Bedeutungen von *begeistert* (begeistert ist ein Adjektiv, das meistens als ein Adjektiv oder Substantiv in der 3. Person Singular verwendet wird)
- das Adjektiv *begeistert* (begeistert) und das Substantiv *Begeisterter* (Begeisterter) und die Substantive *Begeisterung* (Begeisterung) und *Begeisterung* (Begeisterung)
- die verschiedenen Bedeutungen von *begeistert* (begeistert ist ein Adjektiv, das meistens als ein Adjektiv oder Substantiv in der 3. Person Singular verwendet wird)

Begeistert ist ein Adjektiv, das meistens als ein Adjektiv oder Substantiv in der 3. Person Singular verwendet wird.



CONFIGURE `HashMap` WITH SPRING SECURITY

spring
bytes

- 1. `HashMap` is a mutable map that stores key-value pairs. It is a part of the `java.util` package.
- 2. `HashMap` is a mutable map that stores key-value pairs. It is a part of the `java.util` package.

```
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        Map<String, String> map = new HashMap<>();
        map.put("key", "value");
        System.out.println(map);
    }
}
```

- 1. `HashMap` is a mutable map that stores key-value pairs. It is a part of the `java.util` package.
- 2. `HashMap` is a mutable map that stores key-value pairs. It is a part of the `java.util` package.

CONFIGURE denyALL WITH SPENDING SECURITY

SPENDING
03/04/23

- 1. **denyALL** is a security policy that denies all traffic to and from the network. It is a common security measure to prevent unauthorized access to the network.
- 2. **denyALL** is a security policy that denies all traffic to and from the network. It is a common security measure to prevent unauthorized access to the network.

```
denyALL {  
  log {  
    enabled true;  
    level debug;  
  }  
  action {  
    deny;  
  }  
}
```

denyALL is a security policy that denies all traffic to and from the network. It is a common security measure to prevent unauthorized access to the network.

CONFIGURE CUSTOM SECURITY CONFIG & CERT CHAIN

WAF
09/03

- 1. Review application security configuration. Review for any requirements for security specific items
- 2. Determine what the application security and controls and active/passive security controls are (e.g., SSL/TLS)
- 3. Determine what security controls are in place for the application. This can include: SSL/TLS, HTTP, HTTPS, etc. (e.g., SSL/TLS will be required with certificate and private key)
- 4. Review the application security and controls and the application security controls and determine the security controls
- 5. Review the application security controls and the application security controls and determine the security controls



CONFIGURING LOGIN & LOGOUT PAGE

www.dynatrace.com

1. Configure the login page in the `login.html` file in the `src/main/resources` directory.
2. Configure the logout page in the `logout.html` file in the `src/main/resources` directory.
3. Configure the login and logout pages in the `login.html` and `logout.html` files.



QUICK TIP

many
bytes

Be sure to use the correct file extension with every file. Otherwise, you'll be unable to open the file when you need it.

Step 1: Choose the correct extension for your file

- **Documents:**
 - Save as .txt or .rtf if you're saving a plain text document
 - Use .doc or .docx for Microsoft Word documents, .xls or .xlsx for Excel spreadsheets

Step 2: Add the correct icon to your file name to make it easier to find

- **Icons:** Use the correct icon to represent the file type. For example, use a document icon for text files, a spreadsheet icon for Excel files, and a presentation icon for PowerPoint files.

Step 3: Double-check the file name and extension before saving

Before saving your file, double-check the file name and extension to make sure they are correct. This will help you avoid any confusion when you need to find the file later.



- **ExceptionHandler** is an interface that defines the methods that a handler must implement. It is used to handle exceptions that are thrown by the application. You can use it to handle exceptions that are thrown by the application, or you can use it to handle exceptions that are thrown by the framework.
- **ExceptionHandler** is an interface that defines the methods that a handler must implement. It is used to handle exceptions that are thrown by the application, or you can use it to handle exceptions that are thrown by the framework.
- **ExceptionHandler** is an interface that defines the methods that a handler must implement. It is used to handle exceptions that are thrown by the application, or you can use it to handle exceptions that are thrown by the framework.

```
import org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerMethod;
import org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerMethod;

@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler({ExceptionHandler.class})
    public void handle(Exception ex) throws IOException {
        // Handle the exception
        // ...
    }
}
```

ExceptionHandler is an interface that defines the methods that a handler must implement. It is used to handle exceptions that are thrown by the application, or you can use it to handle exceptions that are thrown by the framework.

Homework

1. Write a paragraph about a **technology** you think will be **important** in the future. Use the **writing process** to plan and write your paragraph. Be sure to include a **topic sentence**, **supporting details**, and a **concluding sentence**.
2. Write a paragraph about a **technology** you think will be **important** in the future. Use the **writing process** to plan and write your paragraph. Be sure to include a **topic sentence**, **supporting details**, and a **concluding sentence**.
3. Write a paragraph about a **technology** you think will be **important** in the future. Use the **writing process** to plan and write your paragraph. Be sure to include a **topic sentence**, **supporting details**, and a **concluding sentence**.

Write a paragraph about a **technology** you think will be **important** in the future. Use the **writing process** to plan and write your paragraph. Be sure to include a **topic sentence**, **supporting details**, and a **concluding sentence**.

Write a paragraph about a **technology** you think will be **important** in the future. Use the **writing process** to plan and write your paragraph. Be sure to include a **topic sentence**, **supporting details**, and a **concluding sentence**.

Write a paragraph about a **technology** you think will be **important** in the future. Use the **writing process** to plan and write your paragraph. Be sure to include a **topic sentence**, **supporting details**, and a **concluding sentence**.



CROSS-SITE REQUEST FORGERY (CSRF)

OWASP
DYWID

- 1. A user is logged in to a web application and has a session cookie set.
- 2. The user is redirected to a malicious website that has a CSRF attack.



The user is redirected to a malicious website that has a CSRF attack.

The user is redirected to a malicious website that has a CSRF attack.



The user is redirected to a malicious website that has a CSRF attack.

The user is redirected to a malicious website that has a CSRF attack.



SOLUTION TO CBRP

BRUNN
DUH

1. The first step is to identify the problem. In this case, the problem is that the company is not meeting its goals. The second step is to identify the causes of the problem. The third step is to identify the solutions. The fourth step is to implement the solutions. The fifth step is to evaluate the results.
2. The second step is to identify the causes of the problem. The third step is to identify the solutions. The fourth step is to implement the solutions. The fifth step is to evaluate the results.



- The first step is to identify the problem. In this case, the problem is that the company is not meeting its goals. The second step is to identify the causes of the problem. The third step is to identify the solutions. The fourth step is to implement the solutions. The fifth step is to evaluate the results.
- The second step is to identify the causes of the problem. The third step is to identify the solutions. The fourth step is to implement the solutions. The fifth step is to evaluate the results.



- The first step is to identify the problem. In this case, the problem is that the company is not meeting its goals. The second step is to identify the causes of the problem. The third step is to identify the solutions. The fourth step is to implement the solutions. The fifth step is to evaluate the results.
- The second step is to identify the causes of the problem. The third step is to identify the solutions. The fourth step is to implement the solutions. The fifth step is to evaluate the results.



Given that the job is done, you must find a solution that can
~~be implemented in the future~~
 It is a solution that is practical, feasible, effective, efficient, ethical, and equitable
 (Strongly) I like the solution proposed (Strongly)



The CSRP solution will be better for the community because it is a solution that is practical, feasible, effective, efficient, ethical, and equitable.
 The CSRP solution is better for the community because it is a solution that is practical, feasible, effective, efficient, ethical, and equitable.
 The CSRP solution is better for the community because it is a solution that is practical, feasible, effective, efficient, ethical, and equitable.

Efficiently

- Do not use string formatting (e.g., `str.format()`) to build strings. Instead, use f-strings (e.g., `f'string format'`).
- Using string formatting (e.g., `str.format()`) is slower than f-strings. It also requires an additional argument to the `str.format()` method, which is not needed for f-strings.
- `str.format()` is not thread-safe.
- `str.format()` is not pickleable (cannot be pickled).
- `str.format()` has a large overhead & requires a lot of memory. It also requires a lot of memory. We can avoid this by using the `str.format()` method. It also requires a lot of memory. We can avoid this by using the `str.format()` method.

Example: "Hello" + "World" is slower than f'Hello World' (f-string) by a factor of 10.



SPRING BOOT & H2 DATABASE

spring
boot

1. **SPRING BOOT** is a framework for building microservices, web applications, and mobile applications. It is built on top of the **SPRING** framework.

2. **H2 DATABASE** is an in-memory database that can be used for development, testing, and production environments.

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
```

1. **SPRING BOOT** is a framework for building microservices, web applications, and mobile applications. It is built on top of the **SPRING** framework.

2. **H2 DATABASE** is an in-memory database that can be used for development, testing, and production environments.

3. **SPRING BOOT** is a framework for building microservices, web applications, and mobile applications. It is built on top of the **SPRING** framework.

Key points of JDBC

What is JDBC

- JDBC is Java Database Connectivity, a specification from Java that provides standard approach for connecting to various databases.
- JDBC API along with the database drivers is capable of providing database.
- JDBC is a Java framework or standard for connecting any database, using only JNDI, DriverManager.

Steps to JDBC to access DB

We need to know the below steps to access DB using JDBC.

- 1) Load Driver Class
- 2) Obtain a DB connection
- 3) Making a statement using connection object
- 4) Execute the query
- 5) Process the results set
- 6) Close the connection

Problems with JDBC

- It requires you to know the database and the right connection to get the data and get response with DB using driver of the database connection using JDBC.
- The database needs to handle the database operations that will return the data.
- JDBC is database dependent.



Example of a problem with JIBC is that it is not a good tool for tracking the performance of individual employees. This is because it is not designed to track individual performance, but rather to track the performance of the organization as a whole.



- Spring JdbcTemplate provides a simple way to interact with a database that uses JDBC. It handles the boilerplate code of JDBC and provides a simple API to interact with the database.
- It is a part of the Spring framework and is used to interact with a database that uses JDBC.

```

<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>5.3.12</version>
</dependency>

```

Spring JdbcTemplate provides a simple way to interact with a database that uses JDBC. It handles the boilerplate code of JDBC and provides a simple API to interact with the database.

Spring JdbcTemplate provides a simple way to interact with a database that uses JDBC. It handles the boilerplate code of JDBC and provides a simple API to interact with the database.

Spring JdbcTemplate provides a simple way to interact with a database that uses JDBC. It handles the boilerplate code of JDBC and provides a simple API to interact with the database.

Spring JABC - who does what?

Activity	Typing time	Responsible
Introductory session		■
Open house	■	
Workshop on the topic		■
Workshop on the topic		■
Workshop on the topic	■	
Workshop on the topic	■	
Workshop on the topic	■	
Workshop on the topic		■
Workshop on the topic	■	
Workshop on the topic	■	
Workshop on the topic	■	

Unit 1 - Job Description

Unit 1
Page 1

The purpose of this document is to provide a clear and concise description of the job responsibilities and requirements for the position of **Software Engineer**. This document is intended to be used as a reference for the hiring process and to provide a clear understanding of the role to the candidate.

The job responsibilities of a Software Engineer include the design, development, testing, and deployment of software applications. The candidate should have a strong background in computer science and programming languages such as Java, Python, and JavaScript.

The candidate should also have experience with database management systems and cloud computing technologies. The job requires a strong attention to detail and the ability to work in a fast-paced environment.

```
1. Job Title: Software Engineer  
2. Job Description:  
The Software Engineer is responsible for the design, development, testing, and deployment of software applications. The candidate should have a strong background in computer science and programming languages such as Java, Python, and JavaScript.  
3. Job Requirements:  
The candidate should have a Bachelor's degree in Computer Science or a related field. They should have at least 2 years of experience in software development. The candidate should also have experience with database management systems and cloud computing technologies.  
4. Job Responsibilities:  
The Software Engineer is responsible for the design, development, testing, and deployment of software applications. They should be able to work in a fast-paced environment and have a strong attention to detail.
```



```

// The following query gets all rows of a table
// named t this JdbcTemplate query method returns List<Object> from Object, Integer.class

```

```

// The following query gets a row from a table
// named t this JdbcTemplate query method returns
// Object[] from Object, Integer.class

```

```

// The following query gets a row from a table this JdbcTemplate query method returns
// Object[] from Object, Integer.class

```

3.2.2. Using JdbcTemplate to insert data

Let's start by inserting a new record in the `users` table:

```
1014 JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);  
1015 jdbcTemplate.update("insert into users (first_name, last_name) values (?, ?)",  
1016     "John", "Doe");
```

Let's also `delete` a record in the `users` table:

```
1017 jdbcTemplate.update("delete from users where first_name = ? and last_name = ?",  
1018     "John", "Doe");
```

Let's also `update` a record in the `users` table:

```
1019 jdbcTemplate.update("update users set first_name = ? where last_name = ?",  
1020     "Jane", "Doe");
```

```
    // Note: you may find jdbcTemplate() at method 32 - you may not require this. Alternatively, the method is often  
    // just used from the constructor:  
    this(jdbcTemplate, dataSource, entityManager, entityManagerFactory, sessionFactory());
```

```
    // If you're using Spring 4.2+ you can also use JdbcTemplateBuilder to build the JdbcTemplate. This is useful if you want to  
    // build the JdbcTemplate in a more complex way, for example, you can use JdbcTemplateBuilder to build the JdbcTemplate  
    // and then use JdbcTemplateBuilder to build the JdbcTemplate.
```

USING ROOMMAPPER



- RoomMapper is a library that allows you to map your Room objects to a database table and vice versa.
- RoomMapper is a library that allows you to map your Room objects to a database table and vice versa.
- RoomMapper is a library that allows you to map your Room objects to a database table and vice versa.

```
private void onCreate() {  
    // Create a new Room object  
    Room room = new Room();  
    room.setName("Room 1");  
    room.setCapacity(10);  
    room.setPrice(100);  
    // Save the room to the database  
    roomMapper.save(room);  
}
```

```
public void deleteRoom(int roomId) {  
    // Delete the room from the database  
    roomMapper.delete(roomId);  
}
```

The first part of the presentation is a video that shows how to use a simple data visualization tool to create a dashboard. The second part is a quiz that tests your understanding of the concepts covered in the video.

The third part of the presentation is a video that shows how to use a simple data visualization tool to create a dashboard. The fourth part is a quiz that tests your understanding of the concepts covered in the video.

part of the presentation is a video that shows how to use a simple data visualization tool to create a dashboard.

The first part of the presentation is a video that shows how to use a simple data visualization tool to create a dashboard.

The second part of the presentation is a video that shows how to use a simple data visualization tool to create a dashboard.

The third part of the presentation is a video that shows how to use a simple data visualization tool to create a dashboard.



1. Given below are Spring Boot settings with placeholders. Fill in the blanks. Spring Boot uses configuration placeholders and properties such as `spring.datasource.url` and `spring.datasource.username` to connect to the database. Fill in the blanks with the appropriate values.

2. The code snippet below is a Spring Boot application. Fill in the blanks with the appropriate values.

3. The code snippet below is a Spring Boot application. Fill in the blanks with the appropriate values.

4. The code snippet below is a Spring Boot application. Fill in the blanks with the appropriate values.

```

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}

```



INTRO TO SPRING DATA

SPRING
DATA

Spring Data is a Spring framework project that simplifies the use of data in applications by providing an abstraction layer over various data stores (relational databases, NoSQL databases, etc.) and provides a consistent API for accessing data. It also provides a consistent API for managing data (CRUD operations).

Spring Application Layer

Spring Framework

Spring
Data
JPA

Spring
Data
MongoDB

Spring
Data
Cassandra

Spring
Data
Redis

Spring Data is a Spring framework project that simplifies the use of data in applications by providing an abstraction layer over various data stores (relational databases, NoSQL databases, etc.) and provides a consistent API for accessing data. It also provides a consistent API for managing data (CRUD operations).

- 1. **Database** (relational database, NoSQL database, etc.)
- 2. **ORM** (Object-Relational Mapping) (e.g. Hibernate, JPA, etc.)
- 3. **Spring Data** (e.g. Spring Data JPA, Spring Data MongoDB, etc.)

Spring Data is a framework that provides a common API for accessing data. It is designed to be used with a variety of databases, including relational databases, NoSQL databases, and cloud databases. Spring Data JPA is a module that provides a common API for accessing data using JPA (Java Persistence API).

Spring Data JPA is a module that provides a common API for accessing data using JPA. It is designed to be used with a variety of databases, including relational databases, NoSQL databases, and cloud databases. Spring Data JPA provides a common API for accessing data using JPA, which is a standard for managing data in a database.

Spring Data JPA provides a common API for accessing data using JPA. It is designed to be used with a variety of databases, including relational databases, NoSQL databases, and cloud databases. Spring Data JPA provides a common API for accessing data using JPA, which is a standard for managing data in a database.

The diagram shows how Spring Data integrates with Spring Boot and other Spring Framework components. The top row shows Spring Data, Spring Boot, and Spring Framework. The bottom row shows Spring Data, Spring Boot, and Spring Framework. The middle row shows Spring Data, Spring Boot, and Spring Framework. The diagram illustrates the integration of Spring Data with Spring Boot and other Spring Framework components.



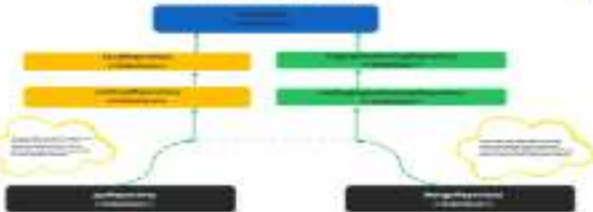
Key points:

- We describe a trading strategy of identifying a transition into Spring Price Expansion markets.
- Spring Price Expansion markets tend to last one month or following the principle called *one-for-one-growth*. This helps you understand how they are formed or when forming it.
- Considering the volume and growth of the market, we suggest the following: the market is likely to be Spring Price Expansion, and the market is likely to be Spring Price Expansion, and the market is likely to be Spring Price Expansion.



INTRO TO SPRING DATA

SCM
Lecture



INTRO TO SPRING DATA JPA

ROOM
10104

- Spring Data JPA is an extension to Spring Data Commons. It adds the JPA support to the Spring Data Commons infrastructure. It is a part of the Spring Data Commons project.
- It is a part of the Spring Data Commons project. It is a part of the Spring Data Commons project. It is a part of the Spring Data Commons project.

Example 1: Spring Data JPA

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface DepartmentRepository extends JpaRepository<Department, Long> {
    // ...
}
```

Example 2: Spring Data JPA

Example 3: Spring Data JPA

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

public interface DepartmentRepository extends JpaRepository<Department, Long> {
    // ...
}
```


EntityManager

`public interface EntityManager {
 // ...
}`

...

`EntityManagerFactory`

`EntityManagerFactory {
 EntityManager getEntityManager();
 // ...
}`

Derived Query Methods in Spring Data JPA



- Any query that JPA does not support natively is called a derived query and writing such queries for your database is tedious.
- To overcome this issue, Spring Data JPA provides a way to write queries in a declarative way, instead of writing queries in JPQL or SQL.
- **Annotations**

```
@Query("select p from Person p where p.name = :name")  
List<Person> findByname(String name);
```

```
@Query("select p from Person p where p.name = :name")  
Person findById(String id);
```

```
@Query("select p from Person p where p.name = :name and p.age = :age")  
Person findByIdAndAge(String id, String age);
```

My first book is about getting my first name. But I don't like my name. It's not what I want.

- 1. The problem (what the problem is) is that I don't like my name. I want a name that is better. I like the name my because it is better than my name. I like the name my because it is better than my name.
- 2. The solution (the answer) is that I want a name. The name my is a better name than my name. I like the name my because it is better than my name. I like the name my because it is better than my name.

My first book is about getting my first name. But I don't like my name. It's not what I want. I like the name my because it is better than my name. I like the name my because it is better than my name.



Important keywords to use in method names

Keyword	Example method name	Example query that flows to JPA
findBy	findByLastName(String name)	select * from user where last_name = :name
find	findByName(String name)	select * from user where name = :name
get	getByName(String name)	select * from user where name = :name
findAll	findAll()	select * from user
findAllBy	findAllByLastName(String name)	select * from user where last_name = :name
findAllBy	findAllByAge(int age)	select * from user where age = :age
findAllBy	findAllByAgeAndLastName(int age, String name)	select * from user where age = :age and last_name = :name
findAllBy	findAllByAgeBetween(int minAge, int maxAge)	select * from user where age between :minAge and :maxAge
findAllBy	findAllByAgeBetweenAndLastName(int minAge, int maxAge, String name)	select * from user where age between :minAge and :maxAge and last_name = :name
findAllBy	findAllByAgeBetweenAndLastNameAndName(int minAge, int maxAge, String name, String name2)	select * from user where age between :minAge and :maxAge and last_name = :name and name = :name2
findAllBy	findAllByAgeBetweenAndLastNameAndNameAndName2(int minAge, int maxAge, String name, String name2, String name3)	select * from user where age between :minAge and :maxAge and last_name = :name and name = :name2 and name2 = :name3
findAllBy	findAllByAgeBetweenAndLastNameAndNameAndName2AndName3(int minAge, int maxAge, String name, String name2, String name3, String name4)	select * from user where age between :minAge and :maxAge and last_name = :name and name = :name2 and name2 = :name3 and name3 = :name4

Important keywords to use in method names

Keyword	Example method name	Example query that flows to JPA
findBy	findByLastName()	select * from user where last_name = ?
findByExample	findByExample(user)	select * from user where last_name = ? and first_name = ?
findBy	findByLastNameAndFirstName()	select * from user where last_name = ? and first_name = ?
findBy	findByLastNameAndFirstNameAndAge()	select * from user where last_name = ? and first_name = ? and age = ?
findBy	findByLastNameAndFirstNameAndAgeAndGender()	select * from user where last_name = ? and first_name = ? and age = ? and gender = ?
findBy	findByLastNameAndFirstNameAndAgeAndGenderAndCity()	select * from user where last_name = ? and first_name = ? and age = ? and gender = ? and city = ?
findBy	findByLastNameAndFirstNameAndAgeAndGenderAndCityAndCountry()	select * from user where last_name = ? and first_name = ? and age = ? and gender = ? and city = ? and country = ?
findBy	findByLastNameAndFirstNameAndAgeAndGenderAndCityAndCountryAndState()	select * from user where last_name = ? and first_name = ? and age = ? and gender = ? and city = ? and country = ? and state = ?
findBy	findByLastNameAndFirstNameAndAgeAndGenderAndCityAndCountryAndStateAndZipCode()	select * from user where last_name = ? and first_name = ? and age = ? and gender = ? and city = ? and country = ? and state = ? and zip_code = ?
findBy	findByLastNameAndFirstNameAndAgeAndGenderAndCityAndCountryAndStateAndZipCodeAndEmail()	select * from user where last_name = ? and first_name = ? and age = ? and gender = ? and city = ? and country = ? and state = ? and zip_code = ? and email = ?
findBy	findByLastNameAndFirstNameAndAgeAndGenderAndCityAndCountryAndStateAndZipCodeAndEmailAndPhone()	select * from user where last_name = ? and first_name = ? and age = ? and gender = ? and city = ? and country = ? and state = ? and zip_code = ? and email = ? and phone = ?

Important keywords to the method names

Keyword	Sample method name	Sample query that flow by JPA
find	<code>findBy...</code>	<code>select * from ... where ...</code>
count	<code>countBy...</code>	<code>select count(*) from ... where ...</code>
exists	<code>existsBy...</code>	<code>select 1 from ... where ...</code>

Spring Data JPA uses a naming convention to generate the query. It is a simple way to set up a query. The query is generated by the JPA. The query is generated by the JPA. The query is generated by the JPA.

For the following example we will use a simple entity representing a person. In addition to a name the class also has a date of birth and a gender attribute. The following code shows the entity class:

```

@Entity
public class Person {
    private String firstName;
    private String lastName;
    private Date dateOfBirth;
    private Gender gender;

    // ... getters and setters ...
}

```

For the following example we will use a simple entity representing a person. In addition to a name the class also has a date of birth and a gender attribute. The following code shows the entity class:

```

@Entity
public class Person {
    private String firstName;
    private String lastName;
    private Date dateOfBirth;
    private Gender gender;

    // ... getters and setters ...
}

```

The next figure shows what the problem that we're trying to solve will resemble by Spring 2000. With the modeling that follows, you'll see.

spring 2000

spring 2000

- the next figure will show the problem that we're trying to solve in the summer of 2000. It will show the problem in a similar format.
- the next figure will show the problem that we're trying to solve in the summer of 2000. It will show the problem in a similar format.




```

// Custom validation logic
public boolean isValid(Long id) {
    // Custom validation logic
    return true;
}

// Custom validation logic
public boolean isValid(Long id) {
    // Custom validation logic
    return true;
}
    
```

```

// Custom validation logic
public boolean isValid(Long id) {
    // Custom validation logic
    return true;
}
    
```

One to One Relationship inside JPA

ORM
JPA

- What is a one-to-one relationship? It is a relationship where a single instance of one class is associated with exactly one instance of the other.
- How can this be implemented in the database?



© 2000 Blackwell Science Ltd *Journal of Internal Medicine* 247: 111–118

[illegible]

Key points of Cascade Types

What is Cascade

When a data is not propagated until its next change then it is called as Cascade operation. This concept is called cascading in SQL.

When we do any type of change we are getting all data of Cascade Type.

Cascade Types

The cascade type supported by SQL are as follows:

- 1) Cascade Type: FULL
- 2) Cascade Type: SETNULL
- 3) Cascade Type: SETDEFAULT
- 4) Cascade Type: NO ACTION
- 5) Cascade Type: NO CASCADE
- 6) Cascade Type: RESTRICT

What is Full Cascade

When we do any change with the Primary table then all data of the Primary table is also changed by the cascading operation. This cascading type is called as Full Cascade and it is not used and not recommended.

There is no default cascade type in SQL. So, default operation is RESTRICT.

CASCADE TYPES

NOTES
English

4.3.1

Concurrent cascades: parallel cascades, each cascade
operates independently of the other cascades.

4.3.2

Concurrent cascades: parallel cascades, each cascade
operates independently of the other cascades.

4.3.3

Concurrent cascades: parallel cascades, each cascade
operates independently of the other cascades.

4.3.4

Concurrent cascades: parallel cascades, each cascade
operates independently of the other cascades.

4.3.5

Concurrent cascades: parallel cascades, each cascade
operates independently of the other cascades.

4.3.6

Concurrent cascades: parallel cascades, each cascade
operates independently of the other cascades.

Spring Security AuthenticationProvider

SSS
Lecture

- We can have our own **AuthenticationProvider** that will implement the **authenticate()** method and return a **Authentication** object.
- The **AuthenticationProvider** interface defines the **authenticate()** method that will be called by the **AuthenticationManager** to authenticate the user.

```
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.password.PasswordEncoder;

public class MyAuthenticationProvider implements AuthenticationProvider {

    private UserDetailsService userDetailsService;
    private PasswordEncoder passwordEncoder;

    public MyAuthenticationProvider(UserDetailsService userDetailsService, PasswordEncoder passwordEncoder) {
        this.userDetailsService = userDetailsService;
        this.passwordEncoder = passwordEncoder;
    }

    @Override
    public Authentication authenticate(Authentication authentication) throws AuthenticationException {
        String username = authentication.getName();
        String password = authentication.getCredentials().toString();

        UserDetails userDetails = userDetailsService.loadUserByUsername(username);

        if (passwordEncoder.matches(password, userDetails.getPassword())) {
            return new UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
        } else {
            throw new BadCredentialsException("Invalid credentials");
        }
    }

    @Override
    public boolean supports(Class authentication) {
        return authentication.equals(UsernamePasswordAuthenticationToken.class);
    }
}
```


Different ways of Pwd management

Everything

- Everything is stored in a file (the presence of everything from those who stores to password itself has resulted in the whole compromise)
- It is vulnerable to access and compromise of the database
- Everything can be accessed, including data. Hence, can the system protect itself from compromise and the resulting

all access: plaintext, encrypted

Everything

- Everything is stored in the presence of everything, data is stored in a way that prevents compromise
- The system itself doesn't store anything, everything is stored in a way that is encrypted, but the user can't see it
- Everything can be accessed by using the system with the help of the "key" or the "key" is available, everything can be accessed in a way

Nothing

- Everything is stored in the presence of everything, data is stored in a way that prevents compromise
- There is no data stored in the system, everything is stored in a way that is encrypted, but the user can't see it
- Everything can be accessed by using the system with the help of the "key" or the "key" is available, everything can be accessed in a way

The old Spring Equinox Festival was the origin of the custom of hanging long strings of beads on the second day of the festival.

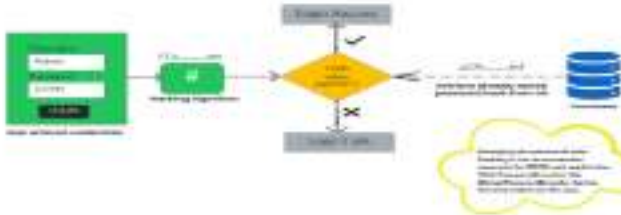
Which representation of Hamlet's death is pointed to by Spring Festival?

- A. Spring Festival is the festival of the dead.
- B. Spring Festival is the festival of the living.
- C. Spring Festival is the festival of the dead.
- D. Spring Festival is the festival of the living.
- E. Spring Festival is the festival of the dead.



How Passwords Validated With hashing & PasswordEncoder

Security
System



Do you know all our favorite and most loved authors for the Spring term? It's time to test your knowledge.

QUESTION
Which of the following is not a children's author?

ANSWER
A. J.K. Rowling



One to Many & Many to One Relationship inside JPA

- **One-to-Many relationship** refers to the relationship between two classes where one class is associated with a collection of instances of another class.
- **Many-to-One relationship** refers to the relationship between two classes where a collection of instances of one class is associated with a single instance of another class.
- **One-to-One relationship** refers to the relationship between two classes where one instance of one class is associated with one instance of another class.



One to Many
Relationship



Many to One
Relationship



One to Many & Many to One Relationship inside JPA

ORM
JPA

- Using JPA, the code developer is just describing a relationship between the entities with these annotations:
`@OneToOne` & `@OneToMany` & `@ManyToOne` & `@ManyToMany`

```
package org.hibernate.tutorial.jpa;  
  
import javax.persistence.*;  
  
@Entity  
@Table(name = "users")  
public class User {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String name;  
  
    // ...  
  
}
```

```
package org.hibernate.tutorial.jpa;  
  
import javax.persistence.*;  
  
@Entity  
@Table(name = "posts")  
public class Post {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String title;  
  
    // ...  
  
}
```

```
package org.hibernate.tutorial.jpa;  
  
import javax.persistence.*;  
  
@Entity  
@Table(name = "posts")  
public class Post {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String title;  
  
    // ...  
  
}
```

```
package org.hibernate.tutorial.jpa;  
  
import javax.persistence.*;  
  
@Entity  
@Table(name = "posts")  
public class Post {  
  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
  
    private String title;  
  
    // ...  
  
}
```

Money to Money Inside JPA

2024
Lecture

- Students may be asked to do the following exercise (or other exercise) to help to understand the concept of money to money.
- This exercise can be used to help to understand the concept of money to money.

CHARGE



STUDENT



Money to Money



DEBIT



PROFESSOR



Money to Money



Many to Many inside JPA

- Many to many relationship is implemented by a join table. The join table has two foreign keys, one to each entity, and a unique constraint on the combination of the two foreign keys.
- The join table is managed by the EntityManager. The join table is created and managed by the EntityManager. The join table is created and managed by the EntityManager.

id	name	age
1	John	25
2	Jane	22
3	Bob	30

Addressable Entity

EntityManager
EntityManager

Join Table

id	name	age
1	John	25
2	Jane	22
3	Bob	30

Addressable Entity

EntityManager
EntityManager

Join Table

id	name	age
1	John	25
2	Jane	22
3	Bob	30

Addressable Entity

EntityManager
EntityManager

Sorting

Introduction

- Spring Data JPA supports sorting in query results with some limitations
- Spring Data JPA provides default implementations of sorting with the help of `Pageable` and `Sort` interfaces
- There are two ways to perform sorting in Spring Data JPA

- 1) Native Sorting
- 2) Abstract Sorting

Native Sorting

- Native sorting refers to the mechanism where the returned data is directly sorted by specified column and direction. The column and sort direction is defined in the annotation level and cannot be changed at runtime.
- Entity is the manager of native sorting

[Entity Manager Sorting \(Native Sort\)](#)

Abstract Sorting

- By using dynamic sorting you can obtain the results related with changing conditions in runtime easily.
- Abstract sorting provides some facilities for querying with column and direction with the help of `Sort` provided in `Pageable` interface. The `Sort` class is just a utility class that provides sorting against the direction specified. Below is an example.

[Sort and Pageable \(Native\) Sorting](#)
with `Sort` and `Pageable`

Pagination

Specifications

- Spring Data JPA supports **Pagination** via its **Pageable** interface to allow users to view all data in various page-size in the application.
- And also the user can have pagination on basis of the dynamic criteria. Spring Data JPA supports another specific interface called **Pageable** to implement the pagination.
- Moreover, Spring Data JPA also has **Pageable** working with the help of **Pageable**.

How to Implement

- However, we need to apply pagination in query results, all required to do is to use the **Pageable** parameter in the query method definition such as the following **Pageable** like below.

Pageable pageable = Pageable.unpaged(); **And last Pageable**

Pageable pageable = Pageable.unpaged(); **Pageable pageable**

Pageable pageable = Pageable.unpaged(); **Pageable pageable**

Custom Queries with JPA

Introduction

- Database queries are great, working in that you can compare the the number of rows returned from a query to the number of rows processed and know that it worked or there's a problem.
- The query language used in database applications, Spring Data JPA, allows developers to write their own queries while the help of native queries.

• Query

• NativeQuery

• NativeViewQuery

Native Query

- The original, unprocessed query is passed through an SQL parser, which then gives you back a query object following the database's native conventions.
- When the native query is processed, it is run with the query in the form of string in native SQL query.
- When you are not writing a native SQL query, you can use the `nativeQuery` in the form of `nativeQuery` in the form of `nativeQuery`.

Native Query

- The native query is processed and then the query is passed to the database. The application is passed the results of the query in the form of a list of objects. The query is passed to the database in the form of a list of objects.
- The native query is processed and then the query is passed to the database. The application is passed the results of the query in the form of a list of objects.

• `nativeQuery` - I can use native SQL query.

• `nativeQuery` - I can use native SQL query.

JPQL

Specifications

- The Java Persistence Query Language (JPQL) is a platform-independent query language designed as part of the Java Persistence API (JPA).
- JPQL is used to create queries against entities stored in a relational database. It is loosely inspired by SQL, but the queries are written using classes as tables, and objects as rows. The syntax follows rules that derive from database tables.
- The Java Database Connectivity (JDBC) is used to implement a subset of the JPQL specified. This is why not for a given driver the complete query.

JPQL Examples

- Below is an example of JPQL. The `entityManager` variable refers to the entity manager and `Book` represents entity as defined in Spring Data and Hibernate entity.

```
entityManager.createQuery("SELECT b FROM Book b WHERE b.title = 'Huckleberry Finn'").getResultList();
```

For details see: <http://manybytes.org/blog/spring-data-jpa>

Sorting Examples

sort

Sort array of numbers in ascending order

```
let arr = [5, 2, 8, 1, 9];  
arr.sort();
```

Sort array of strings in ascending order

```
let arr = ['apple', 'banana', 'orange', 'grape'];  
arr.sort();
```

Sort array of objects in ascending order

```
const arr = [ {name: 'John', age: 30}, {name: 'Jane', age: 25}, {name: 'Bob', age: 35} ];  
arr.sort((a, b) => a.age - b.age);
```

Sorting Examples

sort

sort array of integers in ascending order

```
int main() {
    int arr[10] = {5, 4, 3, 2, 1, 6, 7, 8, 9, 0};
    sort(arr, arr + 10);
    for (int i = 0; i < 10; i++) {
        cout << arr[i] << " ";
    }
    return 0;
}
```

sort array of integers in descending order

```
sort(arr, arr + 10, greater<int>());
// or
sort(arr, arr + 10, greater<int>());
```

sort array of integers in ascending order

Example 1: Selecting all columns

```
SELECT * FROM customers WHERE address IS NOT NULL -- is NULL is a reserved word
(AddQuotes) -- (AddQuotes) is a reserved word (AddQuotes) is a reserved word, ...
```

Example 2: Selecting specific columns

```
SELECT name, address FROM customers WHERE address IS NOT NULL -- is NULL is a reserved word
(AddQuotes) -- (AddQuotes) is a reserved word (AddQuotes) is a reserved word, ...
```

query examples

SQL
Syntax

SELECT * FROM table;

SELECT * FROM table;

SELECT * FROM table WHERE column = value;

SELECT * FROM table ORDER BY column;

SQL is a query language that allows you to interact with a database. It is used to retrieve, insert, update, and delete data from a database. SQL is a standard language for relational databases.

SQL is a query language that allows you to interact with a database. It is used to retrieve, insert, update, and delete data from a database. SQL is a standard language for relational databases.

getNamedQuery & getNamedQueryExamples

SQL
JPA

```
EntityManager em = ...  
getNamedQuery("findByName", ...)  
getNamedQuery("findByName", ...).setParameter("name", "John")  
getNamedQuery("findByName", ...).setParameter("name", "John").getResultList()
```

```
EntityManager em = ...  
getNamedQuery("findByName", ...).setParameter("name", "John")  
getNamedQuery("findByName", ...).setParameter("name", "John").getResultList()  
getNamedQuery("findByName", ...).setParameter("name", "John").getResultList()
```

The above code is just an example. It is not recommended to use the above code in a real application. It is better to use the `EntityManager` interface to get the `EntityManager` object and then use the `getNamedQuery` method to get the `Query` object. This is the recommended way to use the `EntityManager` interface.

the provided `addEmployee` method cannot insert and return values using the `PreparedStatement` and `addBatch` methods. Below is the reason of the error.

`PreparedStatement`

`addBatch()` method is used, gives 4111

`addBatch()` method is used, gives 4111

`PreparedStatement`

`addBatch()` method

`PreparedStatement`

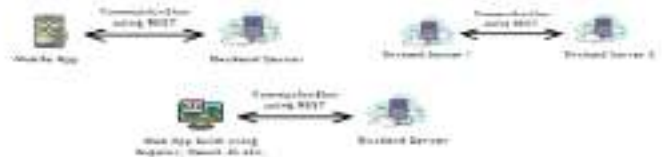
`addBatch()` method is used, gives 4111, inserted data is 11

`addBatch()` method is used, gives 4111, inserted data is 11



Implementing REST Services

- REST services are implemented using Hypertext Transfer Protocol (HTTP) and are designed to be stateless, meaning that the client and server do not maintain any session state.
- REST services are implemented using a standard set of HTTP methods (GET, POST, PUT, DELETE) and a standard set of response codes (200, 400, 404, 500).
- REST services are implemented using a standard set of media types (application/json, application/xml, text/html).



Implementing REST services

REST API

```
def get_posts(self):
    """Get all posts"""
    # ...
    return posts

def get_post(self, post_id):
    """Get a post by ID"""
    # ...
    return post

def create_post(self, post):
    """Create a new post"""
    # ...
    return post

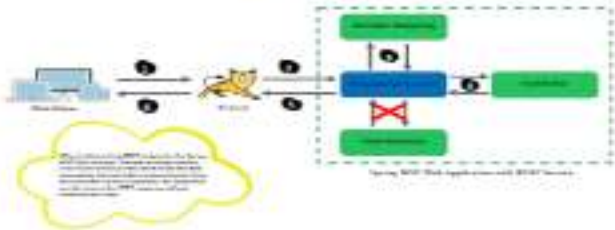
def update_post(self, post_id, post):
    """Update a post by ID"""
    # ...
    return post

def delete_post(self, post_id):
    """Delete a post by ID"""
    # ...
    return True
```

REST API is a set of rules for building networked applications. It is a standard for building web services that are easy to use and integrate.

SPRING MVC ARCHITECTURE WITH REST SERVICES

Spring
Day 60



```

@ExceptionHandler
public void handleException(Exception e) {

```

```

    // ...

```

```

    // ...

```

```

    // ...

```

```

    // ...

```

```

    // ...

```

```

}

```

```

    @ExceptionHandler
    public void handleException(Exception e) {

```

```

        // ...

```

```

        // ...

```

```

        // ...

```

```

        // ...

```

```

        // ...

```

```

    }

```

Different Annotation & Classes in HES1

HEAT
System

4.3.1

Annotation 4.3.1: The first annotation class in the HES1 system. It is used to mark the beginning of a new section.

4.3.2

Annotation 4.3.2: The second annotation class in the HES1 system. It is used to mark the end of a new section.

4.3.3

Annotation 4.3.3: The third annotation class in the HES1 system. It is used to mark the beginning of a new paragraph.

4.3.4

Annotation 4.3.4: The fourth annotation class in the HES1 system. It is used to mark the end of a new paragraph.

4.3.5

Annotation 4.3.5: The fifth annotation class in the HES1 system. It is used to mark the beginning of a new sentence.

4.3.6

Annotation 4.3.6: The sixth annotation class in the HES1 system. It is used to mark the end of a new sentence.

CROSS-ORIGIN RESOURCE SHARING (CORS)

W3C
2018

Web browsers implement the same-origin policy, which prevents scripts from making requests across domain boundaries. The policy is a security measure that limits the ability of scripts running in one domain to access data from another domain. This policy is implemented by the browser, not the server. The policy is implemented by the browser, not the server. The policy is implemented by the browser, not the server.

CORS is a standard that defines how to use the XMLHttpRequest API to request resources from a different domain.

- XMLHttpRequest
- XMLHttpRequest
- XMLHttpRequest



Client (browser)

Client (browser) sends request to server (server)



Server (server)

CROSS-ORIGIN RESOURCE SHARING (CORS)



It enables browser applications to access resources (APIs, scripts, stylesheets, etc.) from other domains, bypassing the browser's same-origin policy.

It allows a browser to request resources from a different domain, bypassing the same-origin policy.

It allows a browser to request resources from a different domain, bypassing the same-origin policy.

It allows a browser to request resources from a different domain, bypassing the same-origin policy.



Client (Browser)

Request (GET, POST, etc.)



Server (API, etc.)

We are not a Notary Public and do not represent or warrant the accuracy or completeness of the information provided on this website. Please use the information at your own risk.

© 2000 Blackwell Science Ltd *Journal of Internal Medicine* 247: 395–402

© 2004 Blackwell Publishing Ltd
Journal of Internal Medicine 255: 103–110

Journal (name) : I am completing this with you to add back the following amount: \$448.00. You agree to give the amount to me or to someone as designated by the program.

1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 26

This work was supported by the National Natural Science Foundation of China (Grant No. 81273055) and the National Key Research and Development Program of China (Grant No. 2016YFA0100900).



Consuming REST Services

REST
Style

- Representational Transfer: Transfer of representation of resources, not the resource itself. Every resource has a representation
- Representations are transferred over the network and are consumed by client programs

Overflight - is a protocol for transporting information over the network. It is a protocol for transporting information over the network.

FreeFlight - is a protocol for transporting information over the network. It is a protocol for transporting information over the network.

WebFlight - is a protocol for transporting information over the network. It is a protocol for transporting information over the network.

© 2015 OpenEgo. All rights reserved. OpenEgo is a registered trademark of OpenEgo. All other trademarks are the property of their respective owners.

OpenEgo is a REST client library that allows you to consume REST services using a simple and intuitive API. It is designed to be easy to use and to integrate with your existing code. OpenEgo is available for Java, JavaScript, and Python.

```

// Example 1: GET request
// Request: GET /api/v1/users/123
// Response: 200 OK
// Status: 200 OK
// Headers: { "Content-Type": "application/json" }
// Body: { "id": 123, "name": "John Doe", "email": "john.doe@example.com" }

// Example 2: POST request
// Request: POST /api/v1/users
// Response: 201 Created
// Status: 201 Created
// Headers: { "Content-Type": "application/json" }
// Body: { "id": 123, "name": "John Doe", "email": "john.doe@example.com" }
    
```

Example

```
public RestApiResponse<BookResponse> bookInfo(Integer bookId) {
    return restClient.getResponse(bookId, "bookInfo");
}
```

MyLibrary

DefaultPageController

```
private RestApiResponse<BookResponse>
public getBook(Integer bookId) {
    return restClient.getResponse(bookId, "bookInfo");
}
```

Consuming REST Services using RestTemplate



1. Create a Spring Boot application with the following dependencies:

2. Add the following code to the `src/main/java` directory:

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

3.

```

public class RestClient {
    public void testRestTemplate() {
        // Create a RestTemplate instance
        RestTemplate restTemplate = new RestTemplate();
        // Create a URI
        URI uri = URI.create("http://localhost:8080/api/users");
        // Create a HttpHeaders object
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json");
        // Create a HttpEntity object
        HttpEntity<String> entity = new HttpEntity<String>(headers);
        // Create a ResponseEntity object
        ResponseEntity<String> response = restTemplate.exchange(uri, HttpMethod.GET, entity, String.class);
        // Print the response
        System.out.println(response.getBody());
    }
}

```

Consuming REST Services using WebClient



It will consume the REST service and transform the response into a Java object.

The following code snippet shows a WebClient class that consumes the REST service and returns the response as a Java object.

```
import java.util.List;
import org.springframework.web.client.RestClientException;
import org.springframework.web.reactive.function.client.WebClient;

public class WebClientExample {
    public static void main(String[] args) {
        WebClient webClient = WebClient.builder().baseUrl("http://localhost:8080").build();
        List<String> response = webClient.get().uri("/api/strings").retrieve().bodyToFlux().collectList().block();
        System.out.println(response);
    }
}
```

```

@PostMapping
public Mono<String> post() {
    WebClient client = WebClient.builder()
        .baseUrl("http://localhost:8080")
        .build();
    return client.post()
        .uri("/api/v1/users")
        .bodyValue("{\"name\":\"John\", \"age\": 30}")
        .retrieve()
        .bodyToMono(String.class);
}

```

Spring Data REST

REST
Style

- Spring Data REST is a part of Spring Data ecosystem, which is a framework for building applications that use REST API.
- It is used to create REST API for Spring Data applications.
- It is used to create REST API for Spring Data applications.
- It is used to create REST API for Spring Data applications.

Example:

```
spring.data.rest.core.version=1.0.0  
spring.data.rest.core.uri=/api/v1  
spring.data.rest.core.uri=/api/v1
```

Example:

Example:

Example:

Example:

Example:

IoT Explorer

IoT Explorer

- The IoT Explorer interface provides a simple way to view and manage your IoT devices and data.
- The interface is divided into three main sections: a sidebar on the left for navigation, a main content area in the center, and a top header area.
- The sidebar contains links to various features, including a dashboard, a list of devices, and a data explorer.

IoT Explorer

IoT Explorer

IoT Explorer

IoT Explorer



IoT Explorer

The job done by the device that sends data is called by network users **sender** using the terms **transmission**.
The job done by the device that receives data is called by network users **receiver**.

transmission medium

How the data is sent from the sender to the receiver is called by network users **transmission medium**.
The job done by the device that sends data is called by network users **sender** using the terms **transmission**.

transmission medium

The job done by the device that sends data is called by network users **sender** using the terms **transmission**.
The job done by the device that receives data is called by network users **receiver**.

transmission medium



Logging

100

- *We cannot let our past be heavy, afraid begging it not
 let loose things that were part of our independence and
 begging them to stay with them.

- It is usually not hard to find the following types of language:

- ☐ FACTS (Who? What? When?)
- ☐ Evidence
- ☐ Details
- ☐ Answer
- ☐ Explanation
- ☐ Conclusion

1. *Journal of the American Medical Association*, 1997; 277: 1001-1005.

- [illegible]

Logging inside SpringBoot

DEV
LOGS

1. **Configure the logging framework to use Logback and add the logback-spring.xml file to the classpath.**



2. **Configure the logging framework to use Logback and add the logback-spring.xml file to the classpath.**

3. **Configure the logging framework to use Logback and add the logback-spring.xml file to the classpath.**

4. **Configure the logging framework to use Logback and add the logback-spring.xml file to the classpath.**

5. **Configure the logging framework to use Logback and add the logback-spring.xml file to the classpath.**



Logging inside SpringBoot

2024
Logika

1. **Configure Logging Framework**
Spring Boot uses the Logback logging framework by default. To configure logging, you can create a `logback.xml` file in the `src/main/resources` directory. This file defines the logging levels, appenders, and log file locations.

```

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} %n%n</pattern>
        </encoder>
    </appender>
    <root>
        <level>INFO</level>
        <appender-ref ref="STDOUT"/>
    </root>
</configuration>
```

2. **Use Logging in Application Code**
Once logging is configured, you can use the logging framework in your application code. Spring Boot provides the `Logger` interface, which is implemented by Logback. You can use the `Logger` interface to log messages at different levels (INFO, DEBUG, ERROR, etc.).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class MyApplication {
    private static final Logger logger = LoggerFactory.getLogger(MyApplication.class);

    public void start() {
        logger.info("Application started");
    }
}
```

Logging inside SpringBoot

Spring
Boot

Spring Boot uses Logback as the default logging framework.

Logback consists of three main components: **Context**, **Appender**, and **Logger**.
The **Context** is the root of the logging hierarchy. It contains a list of **Appenders** and a **Logger**.

The **Appender** is responsible for writing the log messages to the output.

The **Logger** is responsible for creating the log messages.

Spring Boot uses Logback to log the application's execution.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

Logback uses the **LogbackConfiguration** class to configure the logging framework.

The previous example was rather convenient to help students with finding their words. Finding answers takes time.

• **Let's re-examine the before and after of a word and try to make sense.**

Why?

What does it mean?

1

the person

What does it mean?

What does it mean? What does it mean? What does it mean? What does it mean? What does it mean?

2

What does it mean? What does it mean? What does it mean? What does it mean? What does it mean?



Configurations

Introduction

- Spring Boot has two externalized configuration options that interact with the Java application in different manners. The application starts up without configurations unless specific files are present in the classpath.
- By default, Spring Boot looks for the configurations in properties inside application resources (not present in the compiled binaries). But users have other options. One is to use just a file. Spring Boot reads from them.

<https://docs.spring.io/spring-boot/docs/2.1.1.RELEASE/reference/html/>

Config Properties Mechanism

- Spring Boot uses a very particular order that is designed to make easier resolution of values. Properties are resolved in the following order from left to right based on the overriding order needs.
- Properties from the file in application path
- OS Environment variables
- Java System properties (System.getProperties())
- Java command-line properties
- System environment variables
- Default values

Reading properties with g-value

Open
Logic

```
def getPropName, L, m, n, v, c:
```

```
    exprname, exprvalue
```

```
    return exprname, m, n, v, c
```

def getPropName, L, m, n, v, c:

```
def getPropName, L, m, n, v, c:
```

```
    exprname, exprvalue
```

```
    return L["Blockchain", "Blockchain"]
```

```
    return L["Blockchain", "Blockchain"]
```

```
    return L["Blockchain", "Blockchain"]
```

```
    return L["Blockchain", "Blockchain"]
```


1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

© 2004 Blackwell Publishing Ltd *Journal of Internal Medicine* 255: 105–112

[illegible]

Reading properties with `@ConfigurationProperties`



```
import org.springframework.boot.context.properties.bind.*;
import org.springframework.boot.context.properties.source.*;
import org.springframework.stereotype.Component;

@Component
public class ConfigurationProperties {
    // ...
}
```

Now I can access values in `env` from the `ConfigurationProperties` class:

`ConfigurationProperties`

```
1. @ConfigurationProperties(prefix = "myapp")
2. public class ConfigurationProperties {
3.     @Bind({ "myapp.name", "myapp.fullName" })
4.     private String name;
5.     @Bind({ "myapp.name", "myapp.fullName" })
6.     private String fullName;
7.     @Bind({ "myapp.name", "myapp.fullName" })
8.     private String email;
```


Reading properties with getPropertyValue

DOM
try it

```
const element = document.querySelector('div');
const value = element.getPropertyValue('color');
// value === 'red'
```

It's time to learn a new way of manipulating and retrieving data. We'll start by using `getComputedStyle()` to get the computed value of a property. This is the value that the browser uses to render the page. It's the value that you see in the browser's developer tools. It's the value that you see in the browser's console. It's the value that you see in the browser's network tab. It's the value that you see in the browser's storage tab. It's the value that you see in the browser's security tab. It's the value that you see in the browser's settings tab. It's the value that you see in the browser's about:blank page. It's the value that you see in the browser's about:config page. It's the value that you see in the browser's about:debugging page. It's the value that you see in the browser's about:networking page. It's the value that you see in the browser's about:permissions page. It's the value that you see in the browser's about:privacy page. It's the value that you see in the browser's about:search page. It's the value that you see in the browser's about:studies page. It's the value that you see in the browser's about:telemetry page. It's the value that you see in the browser's about:tracing page. It's the value that you see in the browser's about:welcome page. It's the value that you see in the browser's about:welcome page.

```
const element = document.querySelector('div');
const value = window.getComputedStyle(element).color;
// value === 'red'
```

It's time to learn a new way of manipulating and retrieving data. We'll start by using `getComputedStyle()` to get the computed value of a property. This is the value that the browser uses to render the page. It's the value that you see in the browser's developer tools. It's the value that you see in the browser's console. It's the value that you see in the browser's network tab. It's the value that you see in the browser's storage tab. It's the value that you see in the browser's security tab. It's the value that you see in the browser's settings tab. It's the value that you see in the browser's about:blank page. It's the value that you see in the browser's about:config page. It's the value that you see in the browser's about:debugging page. It's the value that you see in the browser's about:networking page. It's the value that you see in the browser's about:permissions page. It's the value that you see in the browser's about:privacy page. It's the value that you see in the browser's about:search page. It's the value that you see in the browser's about:studies page. It's the value that you see in the browser's about:telemetry page. It's the value that you see in the browser's about:tracing page. It's the value that you see in the browser's about:welcome page. It's the value that you see in the browser's about:welcome page.

It's time to learn a new way of manipulating and retrieving data. We'll start by using `getComputedStyle()` to get the computed value of a property. This is the value that the browser uses to render the page. It's the value that you see in the browser's developer tools. It's the value that you see in the browser's console. It's the value that you see in the browser's network tab. It's the value that you see in the browser's storage tab. It's the value that you see in the browser's security tab. It's the value that you see in the browser's settings tab. It's the value that you see in the browser's about:blank page. It's the value that you see in the browser's about:config page. It's the value that you see in the browser's about:debugging page. It's the value that you see in the browser's about:networking page. It's the value that you see in the browser's about:permissions page. It's the value that you see in the browser's about:privacy page. It's the value that you see in the browser's about:search page. It's the value that you see in the browser's about:studies page. It's the value that you see in the browser's about:telemetry page. It's the value that you see in the browser's about:tracing page. It's the value that you see in the browser's about:welcome page. It's the value that you see in the browser's about:welcome page.

```
const element = document.querySelector('div');
const value = window.getComputedStyle(element).color;
// value === 'red'
```

Profiles

Introduction

- Spring provides a great tool for grouping and managing configurations across an entire application. This tool provides a framework for creating a family of configurations based on the environment.
- Profiles are used to group configurations that are specific to a particular environment. This allows you to manage configurations for different environments, such as development, testing, and production, in a single place.
- The `Environment` is a profile that is used to manage configurations for different environments. This allows you to manage configurations for different environments, such as development, testing, and production, in a single place.

Configuring Profiles

- The default profile is `default`, which means that all properties in application.properties are for the default profile.
 - We can create custom profiles for managing properties like a file name.
- ```

spring.profiles.default=development
spring.profiles.active=development

```
- We can create a custom profile by using the `spring.profiles.active` property in the file.

spring.profiles.active=development

When you know there are many ways to achieve a particular feature, use the most appropriate one.

- ✓ Do everything you can to avoid nested `if` and `switch` statements.
- ✓ Using `return` inside a function helps reduce
  - `return` statements, `break` and `continue` statements
  - `return` for any function – `return` is `void` if you
- ✓ Using `goto` is often a good idea
  - `goto` is a good way to avoid nested `if` and `switch` statements
  - `goto` is a good way to avoid nested `if` and `switch` statements
- ✓ Avoiding `goto` is generally a good idea, but sometimes it is the only way to avoid nested `if` and `switch` statements.
- ✓ Using `do` and `while` loops is often a good idea
  - `do` and `while` loops
  - `do` and `while` loops



Example

```

@Component
@Profile("!prod")
public class DatabaseHelper {
 // ...
}

```

```

@Component
@Profile("prod")
public class DatabaseHelper {
 // ...
}

```



# Spring Meet Activities

2024  
Spring  
Meet

- 1. Students will be able to identify the different types of plants and animals that live in the spring.
- 2. Students will be able to identify the different types of plants and animals that live in the spring.

Students will be able to identify the different types of plants and animals that live in the spring.

Students will be able to identify the different types of plants and animals that live in the spring.

Students will be able to identify the different types of plants and animals that live in the spring.

Students will be able to identify the different types of plants and animals that live in the spring.

Students will be able to identify the different types of plants and animals that live in the spring.

## API paths provided by Actuator

| API Path     | Path     | Description                                                                                         |
|--------------|----------|-----------------------------------------------------------------------------------------------------|
| /health      | Health   | Indicates the overall health of the application and its dependencies.                               |
| /info        | Info     | Provides a summary of application information, including version, name, and environment.            |
| /metrics     | Metrics  | Provides a summary of application metrics, including memory usage, CPU usage, and network activity. |
| /env         | Env      | Provides a summary of application environment variables.                                            |
| /configprops | Config   | Provides a summary of application configuration properties.                                         |
| /refresh     | Refresh  | Refreshes the application configuration.                                                            |
| /restart     | Restart  | Restarts the application.                                                                           |
| /shutdown    | Shutdown | Shuts down the application.                                                                         |
| /loggers     | Loggers  | Provides a summary of application loggers.                                                          |
| /logfile     | Logfile  | Provides a summary of application log files.                                                        |
| /auditevents | Audit    | Provides a summary of application audit events.                                                     |
| /actuator    | Actuator | Provides a summary of application actuator endpoints.                                               |

# Spring Test Activities

Small  
bytes

## API paths provided by Articulate

| API Path                                                                          | Path                                                                              | Description                                                                                                                      |
|-----------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| GET /api/v1/courses                                                               | GET /api/v1/courses                                                               | Return a list of courses that are currently active at a given time. The response is paginated and you can filter by course type. |
| GET /api/v1/courses/{course_id}                                                   | GET /api/v1/courses/{course_id}                                                   | Return a single course by ID. The response is paginated and you can filter by course type.                                       |
| GET /api/v1/courses/{course_id}/sections                                          | GET /api/v1/courses/{course_id}/sections                                          | Return a list of sections for a given course.                                                                                    |
| GET /api/v1/courses/{course_id}/sections/{section_id}                             | GET /api/v1/courses/{course_id}/sections/{section_id}                             | Return a single section by ID.                                                                                                   |
| GET /api/v1/courses/{course_id}/sections/{section_id}/enrollments                 | GET /api/v1/courses/{course_id}/sections/{section_id}/enrollments                 | Return a list of enrollments for a given section.                                                                                |
| GET /api/v1/courses/{course_id}/sections/{section_id}/enrollments/{enrollment_id} | GET /api/v1/courses/{course_id}/sections/{section_id}/enrollments/{enrollment_id} | Return a single enrollment by ID.                                                                                                |

API paths provided by Articulate

## Deploying Spring Boot Applications



Spring Boot applications can be deployed to a variety of environments, including:

- Cloud providers (AWS, Azure, Google Cloud)
- Container orchestration (Kubernetes, Docker Swarm)
- Traditional servers (Tomcat, JBoss)
- Edge devices (IoT, embedded systems)

Spring Boot provides a variety of deployment options, including:

- **Executable JARs:** A single JAR file that can be run directly.
- **WARs:** Web Archive files that can be deployed to a web container.
- **Cloud Foundry:** A cloud-native platform for deploying applications.

Using **AWS** as a target for deployment, you can use the **AWS CLI** to upload your application to the **AWS S3** bucket.

For more information on deploying Spring Boot applications to AWS, see the **AWS documentation**.

# ★ AWS EC2 vs AWS Elastic Beanstalk ★

**ELASTIC BEANSTALK PROVIDES SIMPLE**  
 EC2 instances that automatically scale  
 according to the load and manage the  
 software stack to maintain the health of the  
 application. It is a managed service that  
 handles the underlying infrastructure and  
 allows you to focus on your application.

The service handles the scaling of the  
 application and the underlying infrastructure.  
 You can use the service to deploy your  
 application to a fleet of EC2 instances in  
 multiple Availability Zones for high availability.  
 Elastic Beanstalk also provides a console  
 and CLI for managing your application.

**Elastic Beanstalk is a managed service**  
 that automatically scales your application  
 based on the load. It handles the underlying  
 infrastructure and allows you to focus on  
 your application.

The service handles the scaling of the  
 application and the underlying infrastructure.  
 You can use the service to deploy your  
 application to a fleet of EC2 instances in  
 multiple Availability Zones for high availability.  
 Elastic Beanstalk also provides a console  
 and CLI for managing your application.





THANK  
YOU

See you next time!

WOW  
HUGS

