Deloitte.

Spring Boot

USI CBO CR LAUNCHPAD TRAINING PROGRAM

Spring Boot

Context, Objectives, Agenda

Context

- Spring Boot is a Spring module that provides the Rapid Application Development features to the Spring framework.
- Spring Boot makes it easy to create stand-alone, production-grade spring based applications that the user can “just run”. It does not generate code, so there is no requirement for XML configuration.
- Most Spring Boot applications need minimal Spring configuration.

Objectives

- You will learn :
  - Spring Boot Features
  - Spring Boot Components and its implementation
  - Developing Spring Boot application that performs CRUD operations

Agenda

Topic	Content
Spring Boot - Introduction	• Introduction, Advantages, Features, Goals, Pre-Requisites, Spring vs Spring Boot vs Spring MVC, Spring Boot Architecture
Spring Initializer & Spring Boot Example	• Spring Initializer : Overview and Starter Project
Project Components : Annotations & Starters	• Annotations, Dependency Management, Application Properties, Starters, Starter Parent, Starter Data JPA, Starter Web
Spring Boot - Restful	• Initialize - Rest Web Service, Auto Configuration, Approach : REST API Creation, Example – REST API
Actuator	• Overview, Features, Enabling Spring Boot Actuator, Actuator Properties, Execute different Actuator Rest Endpoints
Spring Boot – CRUD Operations	• SQL vs HTTP Verbs vs REST, CRUD Repository & JPA Repository

Spring Boot - Introduction

Spring Boot – Introduction

Spring Boot – What & Why?

01

What is Spring Boot ?

- Spring Boot is an open-source framework and its main goal is to reduce overall development time and increase efficiency by having a default setup for unit and integration tests.
- It is focused on shortening the code length and providing you with an easy way to run Spring application.
- Spring Boot is a project that is built on the top of the **Spring Framework**.
- It provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can **just run**.

02

Why Spring Boot ?

- Spring Boot is an amazing tool that helps you to get enterprise-grade applications up and running quickly without having to worry about configuring your application correctly and safely.
- Spring Boot makes it easier for developers to create and test Java-based applications by autoconfiguring all components **with embedded HTTP servers** to test web applications.
- It reduces development time and increases the overall productivity of the development team. It is compatible to easily connect with database and queue services.

03

New Features

- Support constructor binding for property nested inside a JavaBean.
- Add config property for CodecConfigurer.maxInMemorySize in **WebFlux.**
- Make test slices' type exclude filters public.
- Support amqps:// URIs in **spring.rabbitmq.addresses.**

Spring Boot – Introduction

Spring Boot – Advantages, Features, Goals, Pre-Requisites

Advantages

- It avoids writing lots of Boilerplate code, Annotations and XML configuration
- Embedded servers like Tomcat or Jetty to reduce complexity in deployment
- Creates stand-alone Spring applications

Features

- Opinionated ‘starter’ dependencies to simplify build and application configuration
- Metrics, Health check and externalization configuration
- Automatic config for Spring functionality – whenever possible

Goals

- To avoid complex XML configuration in Spring
- To develop production ready Spring applications in an easier way
- To reduce development time and run the application independently

Pre-Requisites

- Java 1.8
- Maven 3.0+
- Spring Framework 5.0.0.BUILD-SNAPSHOT
- An IDE(Spring Tool Suite) is recommended

Spring Boot – Introduction

Spring vs Spring Boot vs Spring MVC

Spring

- The developer writes a lot of code(boilerplate code) to do the minimal task.
- Spring Framework is a widely used Java EE development for building applications.
- The primary feature of Spring Framework is dependency injection.
- To test Spring project, we need to set up the server explicitly.
- The development time is more compared to Spring Boot and Spring MVC.
- It allows loose coupling and easy testability.

Spring MVC

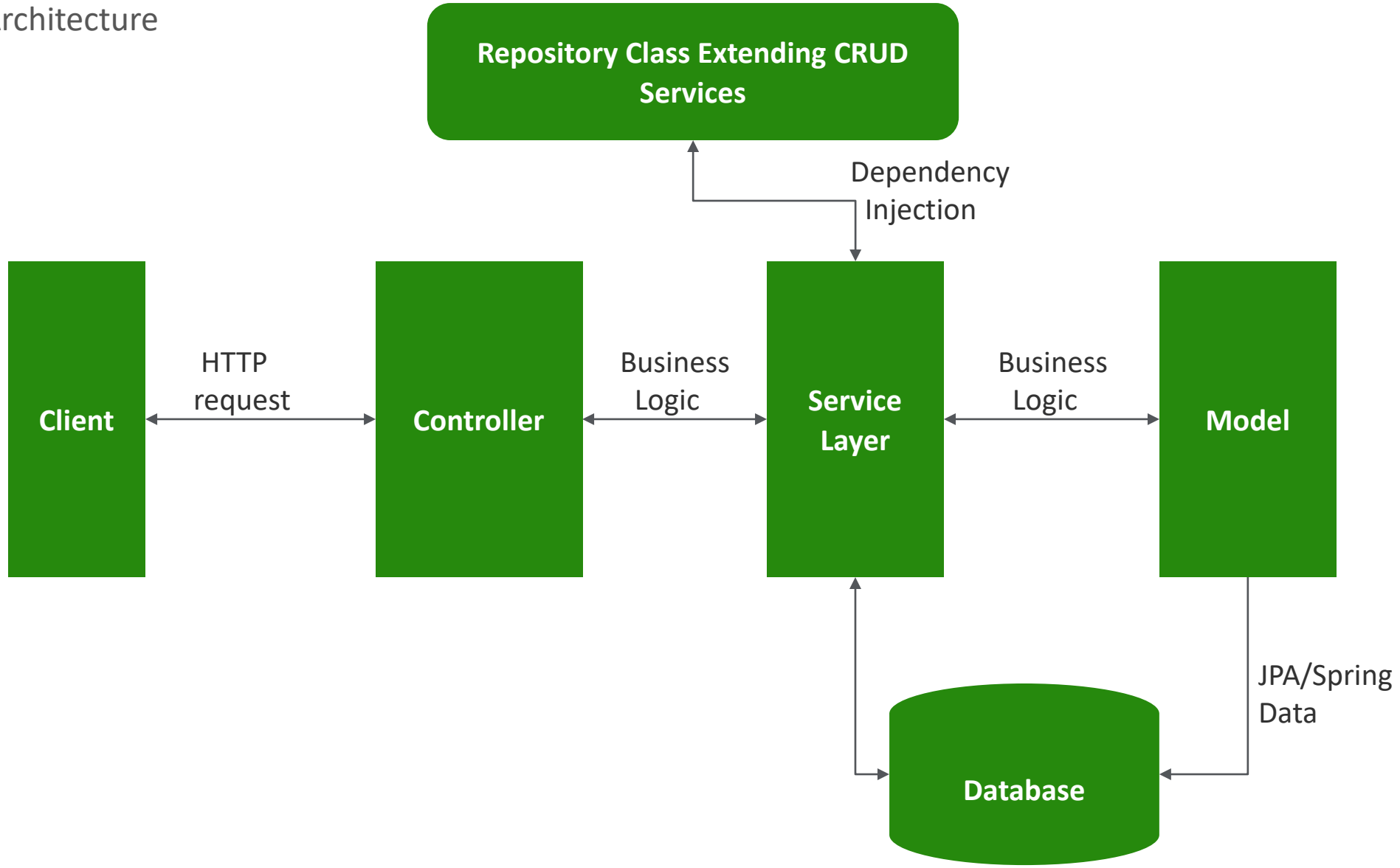
- Spring MVC is a Model View Controller-based web framework under the Spring framework.
- It provides ready to use features for building a web application.
- It specifies each dependency separately.
- A Deployment descriptor is required.
- It takes more time to achieve the same.
- It requires build configuration manually.
- We need to configure component scan, dispatcher servlet, a view resolver, web jars among other things.

Spring Boot

- It avoids boilerplate code and wraps dependencies together in single unit.
- Spring Boot provides default configurations to build Spring-powered framework.
- The primary feature of Spring Boot is Auto Configuration.
- Spring Boot offers embedded servers such as Jetty and Tomcat, etc.
- It reduces development time and increases productivity.
- Metrics, Health check, and externalized configuration.

Spring Boot – Introduction

Spring Boot Architecture



Knowledge Check

1. Which of the following is Spring Boot feature?

- ○ Boilerplate code
- ○ Need External server
- ○ Auto Configuration
- ○ None of the above

2. Which of the following statement is used to create Spring Boot application?

- ○ Angular CLI
- ○ Command Prompt
- ○ Spring Initializr
- ○ None of the above

3. Which of the following is the embedded server in Spring Boot?

- ○ Tomcat
- ○ Wild Fly
- ○ Web logic
- ○ None of the above

4. Which of the following are pre-requisites for Spring Boot?

- ○ IDE
- ○ JDK
- ○ Maven
- ○ All of the above

Knowledge Check

5. Spring Boot deploys in the form of?

- War
- JAR
- RAR
- Both A and B

6. Which of the following is true about Spring Boot?

- Deployment descriptor is required
- Build configuration manually
- Boilerplate code
- None of the above

7. Which of the following can be considered as external configuration in Spring Boot?

- Application.properties
- Adding JARS externally
- POM.xml
- None of the above

8. Using Spring Boot we can create stand alone applications?

- True
- False

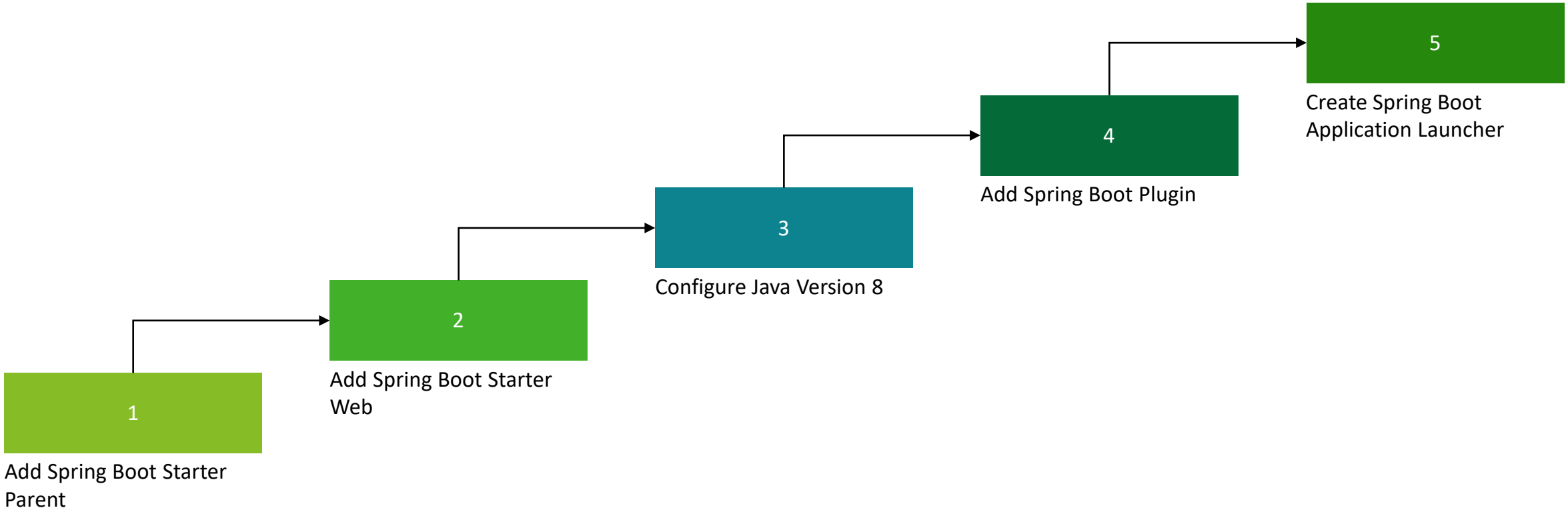
Spring Initializer

Spring Initializer

Spring Initializer : Overview and Starter Project

- The Initializer offers a fast way to pull in all the dependencies you need for an application and does a lot of the setup for you.
- Spring Initializer provides an extensible API to generate JVM-based projects and to inspect the metadata used to generate projects, for instance to list the available dependencies and versions.

Steps to create your First Spring Boot Application:



Spring Boot Project Components - Annotations & Starters

Spring Boot Project Components : Annotations

Spring Boot Annotations : It is a form of meta data that provides data about a program.

@EnableAutoConfiguration :

- It auto-configures the bean that is present in the class path and configures it to run the methods.
- The use of this annotation is reduced in Spring Boot 1.2.0 release because developers provided an alternative of the annotation, i.e. @SpringBootApplication.

@SpringBootApplication

It is the combination of three annotations :

- @EnableAutoConfiguration
- @ComponentScan
- @Configuration

@ComponentScan:

- Enable @Component scan on the package where the application is located

@Configuration:

- Allow to register extra beans in the context or import additional configuration classes

Example :

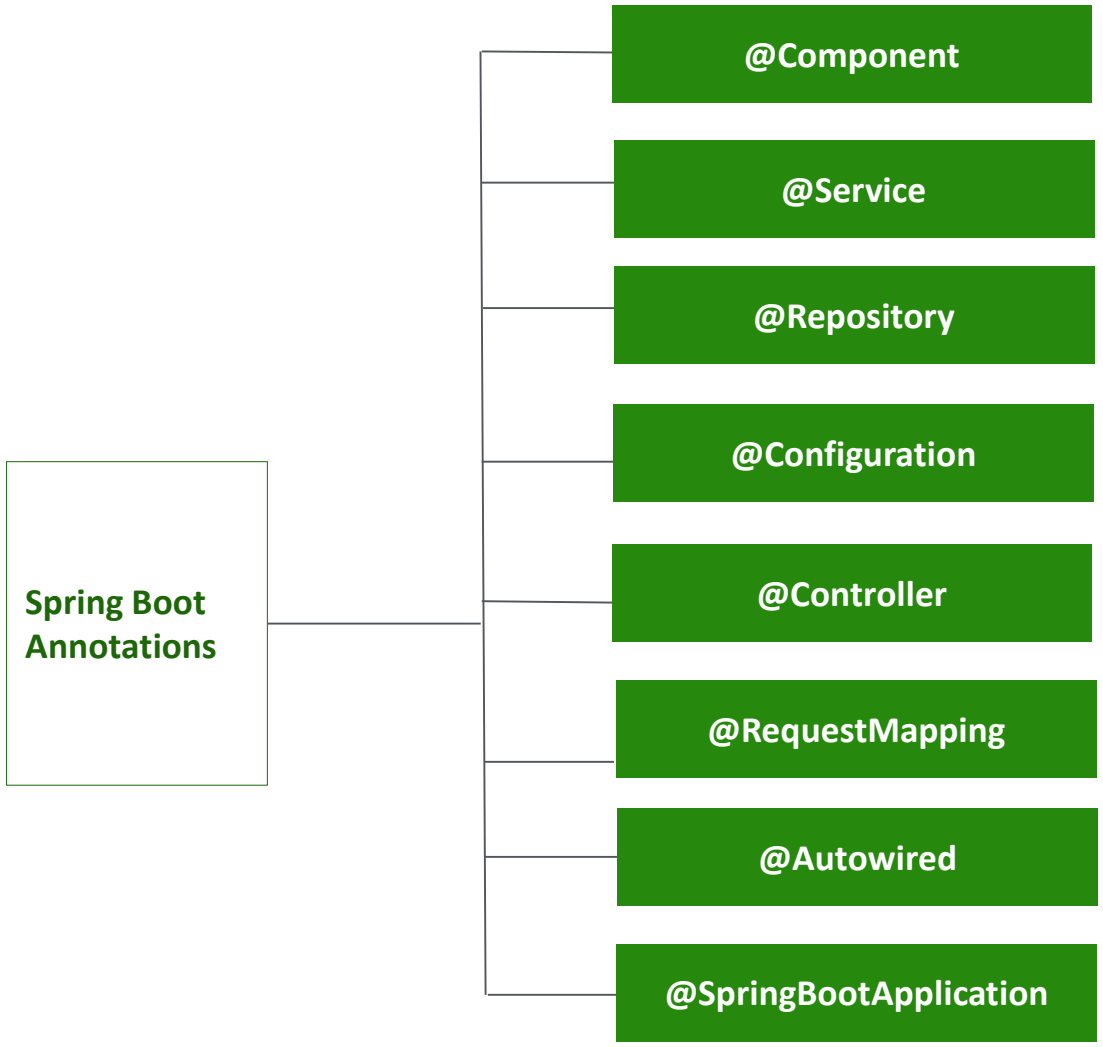
```
package com.example.myapplication;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication // same as @Configuration
@EnableAutoConfiguration @ComponentScan
public class Application {

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

Spring Boot Project Components : Annotations

Basic Examples of Spring Boot Annotations



- A Java class annotated with @Component is found during the class path. The Spring Framework pick it up and configure it in the application context as a Spring Bean.
- Indicates that an annotated class is a service class.
- Indicates that an annotated class is a repository, which is an abstraction of data access and storage.
- Indicates that a class is a configuration class that may contain bean definitions.
- Marks the class as web controller, capable of handling the requests.
- Maps HTTP request with a path to a controller method.
- Marks a constructor, field, or setter method to be autowired by Spring dependency injection.
- Enables Spring Boot autoconfiguration and component scanning.

Spring Boot Project Components : Annotations

Spring Boot Dependency Management - Features and Advantages

Features

- ✓ Spring Boot manages dependencies and configuration automatically.
- ✓ Each release of Spring Boot provides a list of dependencies that it supports.
- ✓ The list of dependencies is available as a part of the Bills of Materials (spring-boot-dependencies) that can be used with Maven.
- ✓ So, we need not to specify the version of the dependencies in our configuration. Spring Boot manages itself.
- ✓ Spring Boot upgrades all dependencies automatically in a consistent way when we update the Spring Boot version.
- ✓ Maven Dependency inherits a **Dependency Section** from the spring-boot-dependency-pom. It manages the version of common dependencies.
- ✓ Dependencies, inherited from the spring-boot-dependencies POM

Advantages

- It provides the centralization of dependency information by specifying the Spring Boot version in one place.
- It helps when we switch from one version to another.
- It avoids mismatch of different versions of Spring Boot libraries.
- We only need to write a library name with specifying the version. It is helpful in multi-module projects.
- When you upgrade Spring Boot itself, these dependencies are upgraded as well in a consistent way.
- Sensible **resource filtering**.
- Sensible **plugin configuration**.
- The default **Java compiler version**.
- **UTF-8** source encoding.

Spring Boot Project Components : Annotations

Spring Boot Dependency Management - Features and Advantages

Configuration :

- Spring Boot Framework comes with a built-in mechanism for application configuration using a file called **application.properties**.
- It is located inside the **src/main/resources** folder, as shown in the following figure.
- Spring Boot also allows us to define our own property if required. The application.properties file allows us to run an application in a different environment.

Features :

We can use the application.properties file to:

- Configure the Spring Boot framework.
- Define our application custom configuration properties.
- Spring Boot provides another file to configure the properties is called yml file.
- All the database related configurations are mentioned in application properties.

Example of application.properties:

```
#configuring application name
spring.application.name = demoApplication
#configuring port
server.port = 8081
```

Example of application.yml:

```
spring:
application:
name: demoApplication
server:
port: 8081
```

Spring Boot Project Components : Starters

Spring Boot Starters - Dependencies and its Uses

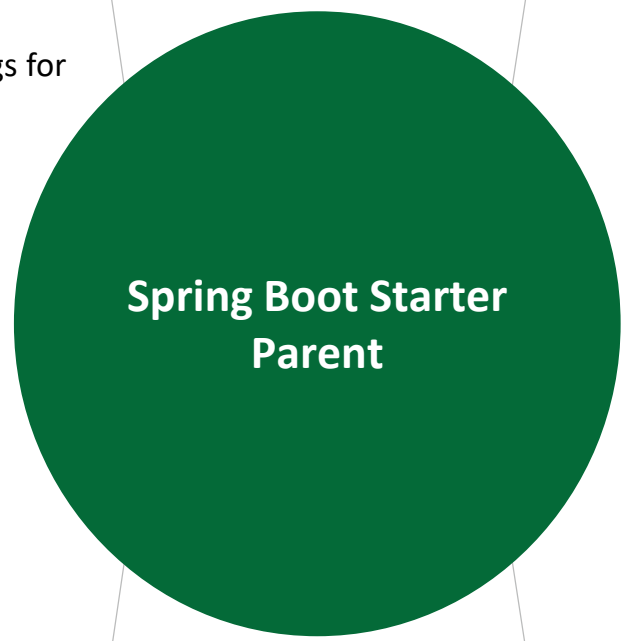
There are many starters available. Here are just five of them to get a glimpse.

Starter	Transitive dependency	Dependency
✓ spring-boot-starter	❑ spring-boot, spring-boot-autoconfigure, spring-boot-starter-logging, spring-core, snakeyaml	❑ Core starter, autoconfiguration support, logging, YAML
✓ spring-boot-starter-data-jpa	❑ Spring-boot-starter, spring-boot-starter-aop, spring-boot-starter-jdbc, hibernate-entitymanager, javax.transaction-api, spring-data-jpa, spring-aspects	❑ Starter for using spring data JPA with Hibernate
✓ spring-boot-starter-test	❑ Junit, mockito-core, hamcrest-core, hamcrest-library, spring-core, spring-test	❑ Starter for testing using libraries Junit, Hamcrest, Mockito
✓ spring-boot-starter-web	❑ Spring-boot-starter, spring-boot-starter-tomcat, spring-boot-starter-validation, jackson-databind, spring-web, spring-webmvc	❑ Starter for building Web application using Spring MVC, REST, Tomcat as a default embedded container
✓ spring-boot-starter-data-mongodb	❑ Spring-boot-starter, mongo-java-driver, spring-data-mongodb	❑ Starter for using MongoDB and Spring data MongoDB

Spring Boot Project Components : Starters

Spring Boot Starter Parent

- The spring-boot-starter-parent is a project starter. It provides default configurations for our applications. It is used internally by all dependencies.
- All Spring Boot projects use spring-boot-starter-parent as a parent in pom.xml file.
- Parent Poms allow us to manage the following things for multiple child projects and modules:
- **Configuration:**
  - It allows us to maintain consistency of Java Version and other related properties.
- **Dependency Management:**
  - It controls the versions of dependencies to avoid conflict
  - Source encoding
  - Default Java Version
  - Resource filtering
  - It also controls the default plugin configuration
- The spring-boot-starter-parent inherits dependency management from spring-boot-dependencies.



Example :

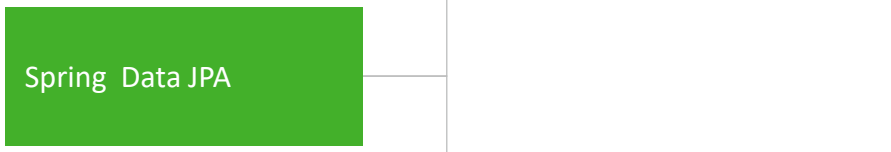
```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>1.4.0.RELEASE</version>
</parent>
```

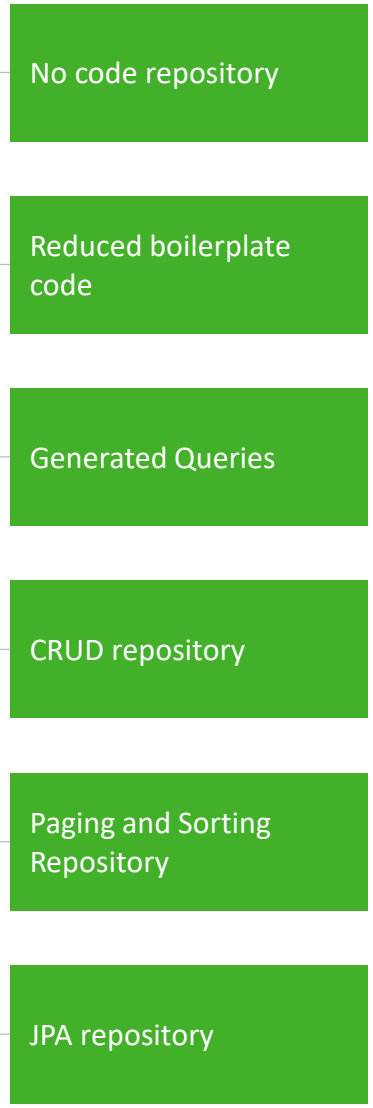
Default parent POM :

```
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-dependencies</artifactId>
<version>1.6.0.RELEASE</version>
<relativePath>../../spring-boot-dependencies</>
</parent>
```

Spring Boot Project Components : Starters

Spring Boot Starter Data JPA





Description

- It is the most popular persistence-related pattern.
- It enables us to implement our business code on a higher abstraction level.

- It provides the default implementation for each method by its repository interfaces.
- It means that there is no longer need to implement read and write operations.

- Generation of database queries based on the method name.
- If the query is not too complex, we need to define a method on our repository interface with the name that starts with findBy

- It offers standard create, read, update and delete .
- It contains methods like findOne(), findAll(), save(), delete(), etc.

- It extends the CrudRepository and adds the findAll methods.
- It allows us to sort and retrieve the data in a paginated way.

- It extends the both repository CrudRepository and PagingAndSortingRepository.
- It adds the JPA-specific methods, like flush() to trigger a flush on the persistence context.

Spring Boot Project Components : Starters

Spring Boot Starter Web - Features, Embedded Server, Configuration

Features

- Compatible for web development
- Auto configuration
- Default embedded server
- Reduces build dependency count
- Tomcat server dependency by default

Auto configures the following :

- Dispatcher Servlet
- Error Page
- Web JARs
- Embedded servlet container

Spring Boot Embedded Web Server

- Embedded as part of deployable application
- Do not require pre–installed server
- Tomcat is default embedded server
- Also supports Jetty & Undertow server

Example:

```
dependency>

<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>

<exclusions>
 <exclusion>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-
tomcat</artifactId>
 </exclusion>
</exclusions>

</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

Knowledge Check

1. Which annotation is used to convert normal Java class as Spring Boot application?

- ○ @SpringBootApplication
- ○ @Autowired
- ○ @Component
- ○ @Controller

2. Which annotation maps HTTP request with a path to controller method?

- ○ @RequestMapping
- ○ @Service
- ○ @Entity
- ○ None of the above

3. Which of the following is Spring Boot Repository?

- ○ PagingAndSortingRepository
- ○ JPA Repository
- ○ CrudRepository
- ○ All of the above

4. Which of the following can be autoconfigured by Spring Boot?

- ○ Error Page
- ○ Web JARS
- ○ Dispatcher Servlet
- ○ All of the above

Knowledge Check

5. Which file is used for external configuration?

- Web.xml
- Application.jsp
- Application.properties
- None of the above

6. Which annotation indicates that a class is a configuration class that may contain bean definitions?

- @Controller
- @Component
- @Configuration
- None of the above

7. In which file the dependencies are declared?

- Application.properties
- WEB.xml
- POM.xml
- None of the above

8. Which file contains database related configuration?

- WEB.jsp
- Application.properties
- POM.xml
- All of the above

Hands On

Spring Initializer Annotations & Starters

Approach

- Launch Spring Initializr and choose the following
  - Choose **com.deloitte.springboot** as Group
  - Choose **cr-services** as Artifact
- Choose from the following dependencies:
  - Web
  - Actuator
  - DevTools
  - Click Generate Project
- This would download a ZIP file to your local machine. Unzip the zip file and extract to a folder.
- In Eclipse, Click File > Import > Existing Maven Project as shown below.

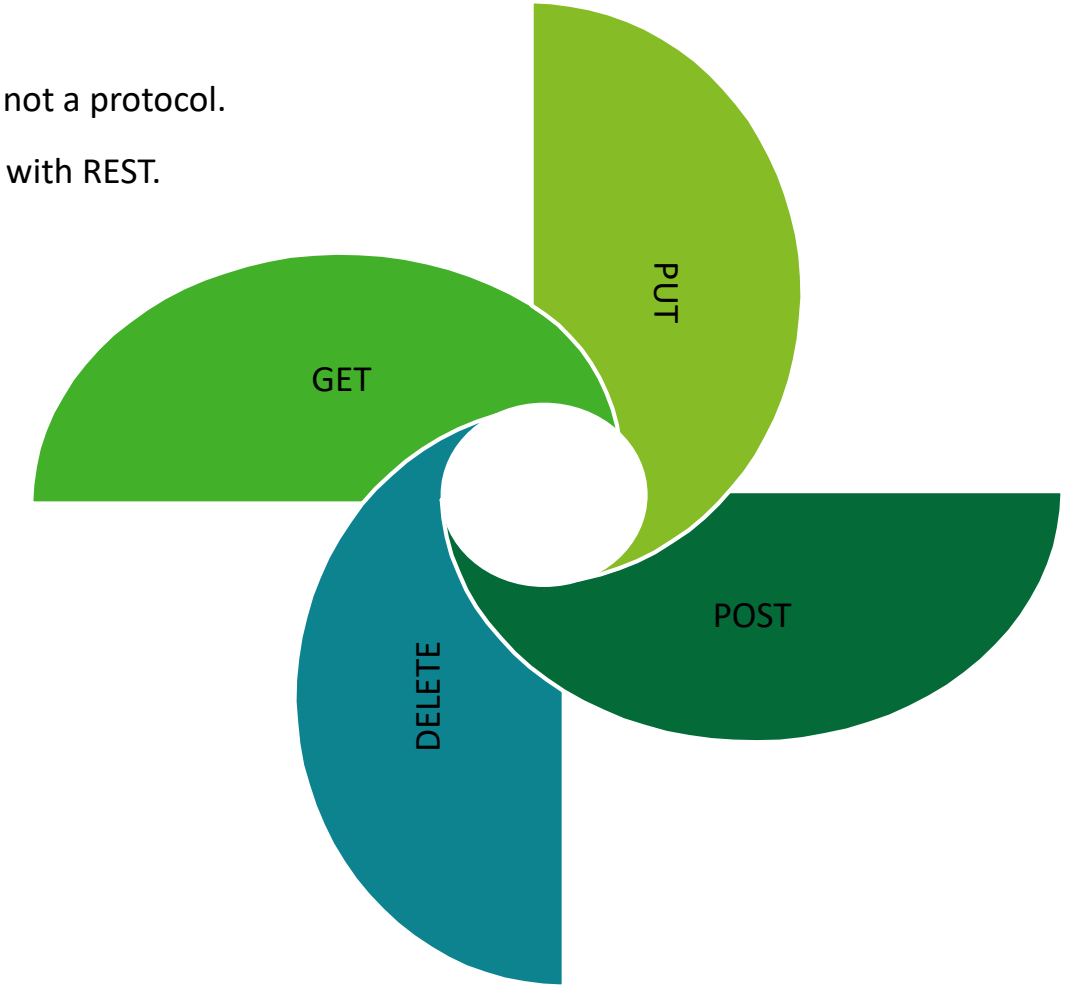
Break – 15 min

Spring Boot - Restful

Spring Boot - Restful

Introduction to Restful Web Services With Spring Boot : Overview

- REST stands for Representational State Transfer. REST is an architectural approach, not a protocol.
- We can build REST services with both XML and JSON. JSON is more popular format with REST.
  - GET: It reads a resource.
  - PUT: It updates an existing resource.
  - POST: It creates a new resource.
  - DELETE: It deletes the resource.

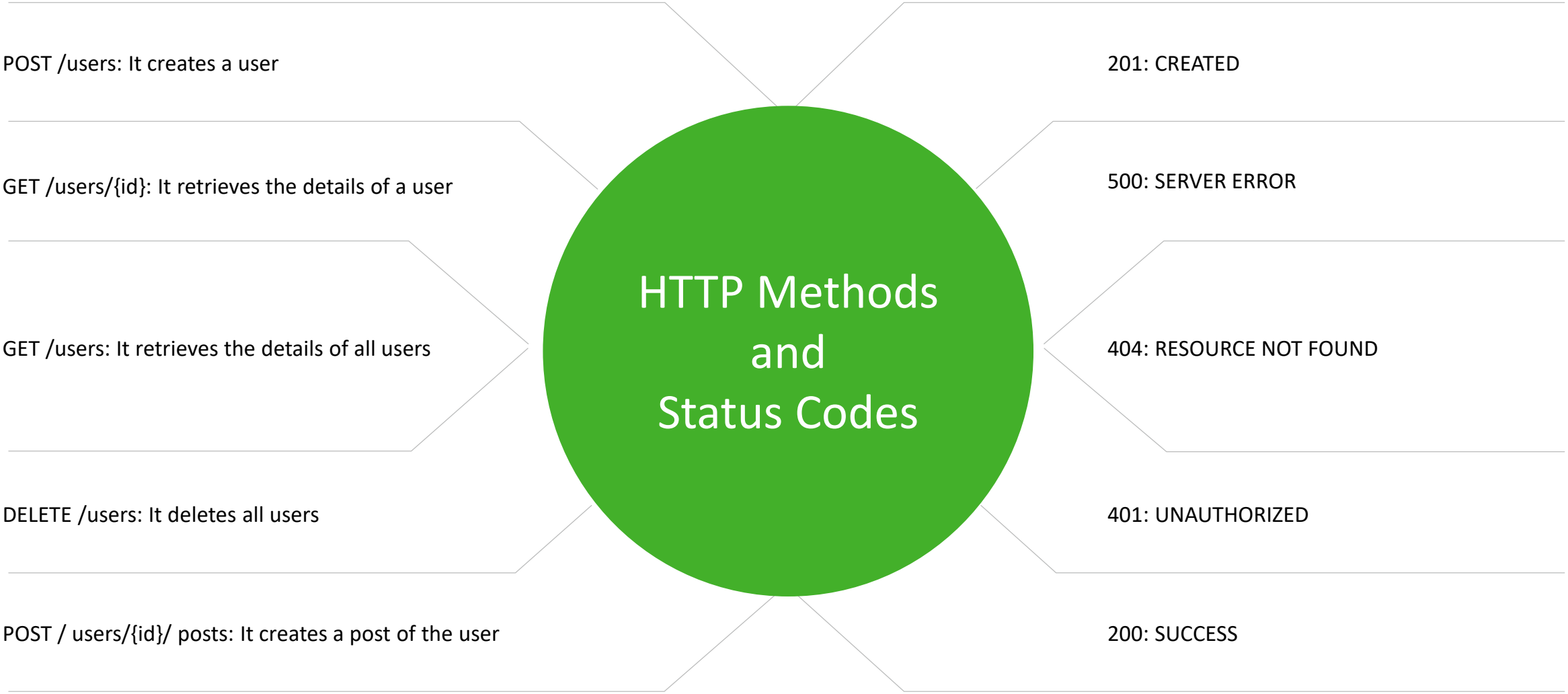


Features

- ✓ RESTful web services are platform-independent.
- ✓ It can be written in any programming language and can be executed on any platform.
- ✓ It provides different data format like JSON, text, HTML, and XML.
- ✓ It is fast in comparison to SOAP because there is no strict specification like SOAP.
- ✓ These are reusable and language neutral.

Spring Boot - Restful

HTTP Methods And Standard Status Codes - *These are the important methods in URI implementation of REST call.*



Spring Boot - Restful

REST Annotation

@RestController

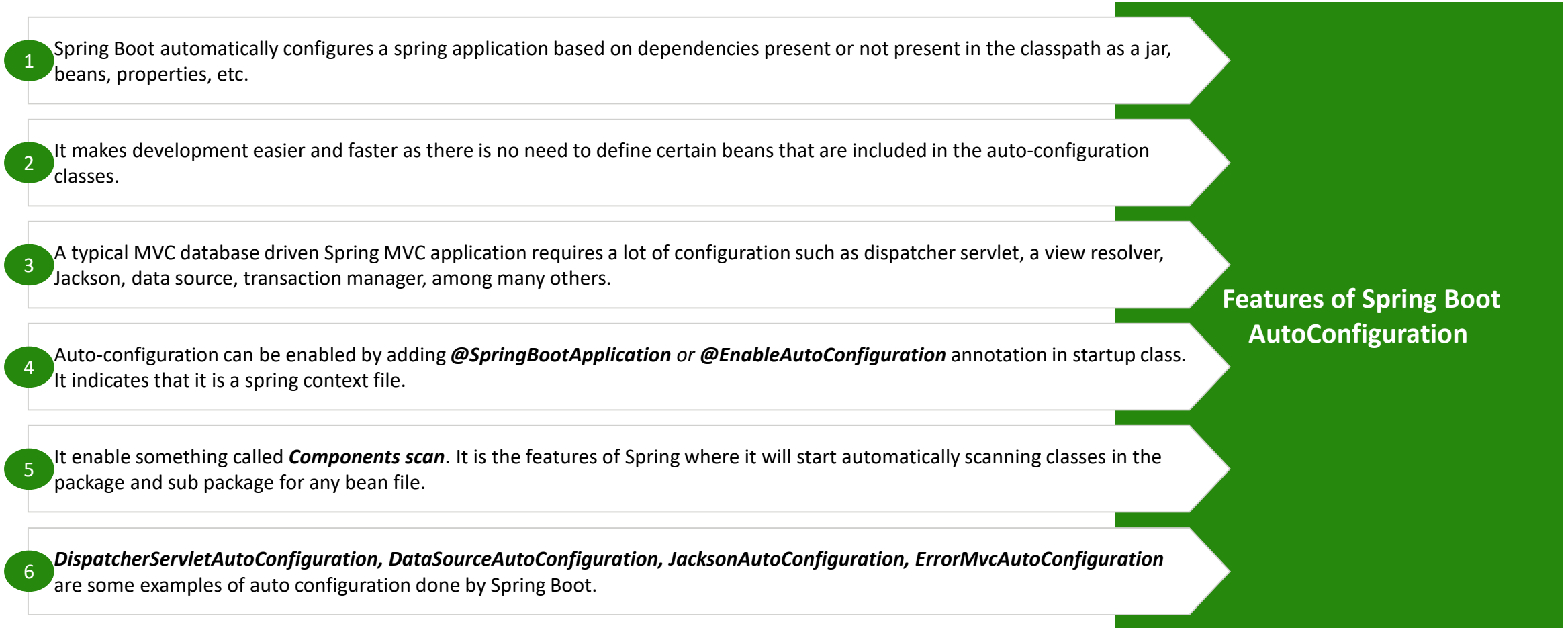
1. This annotation is used at the class level.
2. The **@RestController** annotation marks the class as a controller where every method returns a domain object instead of a view.
3. By annotating a class with this annotation, you no longer need to add **@ResponseBody** to all the **RequestMapping** methods.
4. It means that you no long use view-resolvers or send HTML in response. You just send the domain object as an HTTP response in the format that is understood by the consumers, like JSON.
5. @RestController is a convenient annotation that combines **@Controller** and **@ResponseBody**.
6. Spring RestController annotation is used to create RESTful web services using Spring Boot.
7. Spring RestController takes care of mapping request data to the defined request handler method.
8. Once response body is generated from the handler method, it converts it to JSON or XML response.

Example :

```
@RestController
@RequestMapping("books-rest")
public class SimpleBookRestController {
@GetMapping("/{id}", produces = "application/json")
public Book getBook(@PathVariable int id) {
return findBookById(id);
}
```

Spring Boot - Restful

Spring Boot Auto Configuration - Overview



Spring Boot - Restful

Example – REST API

Steps to understand concept of REST API

- Create a Maven project using Spring Initializr.
- Create Controller using annotation **@RestController**. This annotation informs Spring Boot framework that the class contains methods that will be invoked through a web-based resource URL.
- Add **@RequestMapping** annotations to methods. These annotations define the HTTP method used along with the structure of the resource URLs that will be used to invoke them.
- We need to generate a JSON-based response for the client.
- There are excellent frameworks like Jackson and GSON, which you should use in larger projects. We can create simple JAVA string manipulation to generate JSON.
- Run the Spring Boot Application class and invoke URLs using **POSTMAN** .

Code Example :

```
restTemplate.getForObject(REST_SERVICE_URI + "getDetails", Class.class)

restTemplate.put(REST_SERVICE_URI , object, params);

restTemplate.delete(REST_SERVICE_URI , params);
```

Knowledge Check

1. Which annotation enables autoconfiguration?

- ○ @SpringBootApplication
- ○ @EnableAutoConfiguration
- ○ Both A and B
- ○ @Component

2. Which HTTP method is used to fetch resources?

- ○ GET
- ○ POST
- ○ PUT
- ○ DELETE

3. Which annotation indicates that methods in class is invoked by web URLs?

- ○ @RestController
- ○ @Controller
- ○ @Component Scan
- ○ None of the above

4. Which status is displayed when response is success?

- ○ 200
- ○ 404
- ○ 500
- ○ All of the above

Knowledge Check

5. The data formats supported by REST?

- ○ HTML
- ○ XML
- ○ JSON
- ○ All of the above

6. Which status code is displayed for internal server error?

- ○ 500
- ○ 503
- ○ 404
- ○ 200

7. Which HTTP method updates existing resource?

- ○ DELETE
- ○ GET
- ○ PUT
- ○ None of the above

8. REST is architectural approach?

- ○ TRUE
- ○ FLASE

Hands On

Spring Boot RESTful

Approach

- Create a Maven project using Spring Initializer
- Add spring-boot-starter-parent, spring-boot-starter-web as per requirement
- Create REST API Controller with following methods. Feel free to add more methods
  - HTTP GET /employees – Returns list of the employees.
  - HTTP POST /employees – Add an employee in the employees collection.
- Create Model class with following details :
  - Class name : Employee
  - Fields : id, firstname, lastname, email
- Create DAO class with static list to store Employee data
- Run the application. Once the server is Up, access API using some rest client.

Lunch Break – 45 min.

Spring Boot - Actuator

Spring Boot - Actuator

Overview And Features

1 Endpoint

- The actuator endpoints allows us to monitor and interact with the application.
- Spring Boot provides a number of built-in endpoints. We can also create our own endpoint.
- We can enable and disable each endpoint individually.

2 Metrics

- Spring Boot Actuator provides dimensional metrics by integrating with the micrometer.
- The micrometer is integrated into Spring Boot.
- It is the instrumentation library powering the delivery of application metrics from Spring.

3 Audit

- Spring Boot provides a flexible audit framework that publishes events to an AuditEventRepository.
- It automatically publishes the authentication events if spring-security is in execution.

Spring Boot - Actuator

Properties and Enabling

Spring Boot Actuator Properties

- Spring Boot enables security for all actuator endpoints.
- It uses form-based authentication that provides user Id as the user and a randomly generated password.
- We can also access actuator-restricted endpoints by customizing basic auth security to the endpoints.
- We need to override this configuration by management.security.roles property.

For example:

```
management.security.enabled=true
management.security.roles=ADMIN
security.basic.enabled=true
security.user.name=admin
security.user.passowrd=admin
```

Enabling Spring Boot Actuator

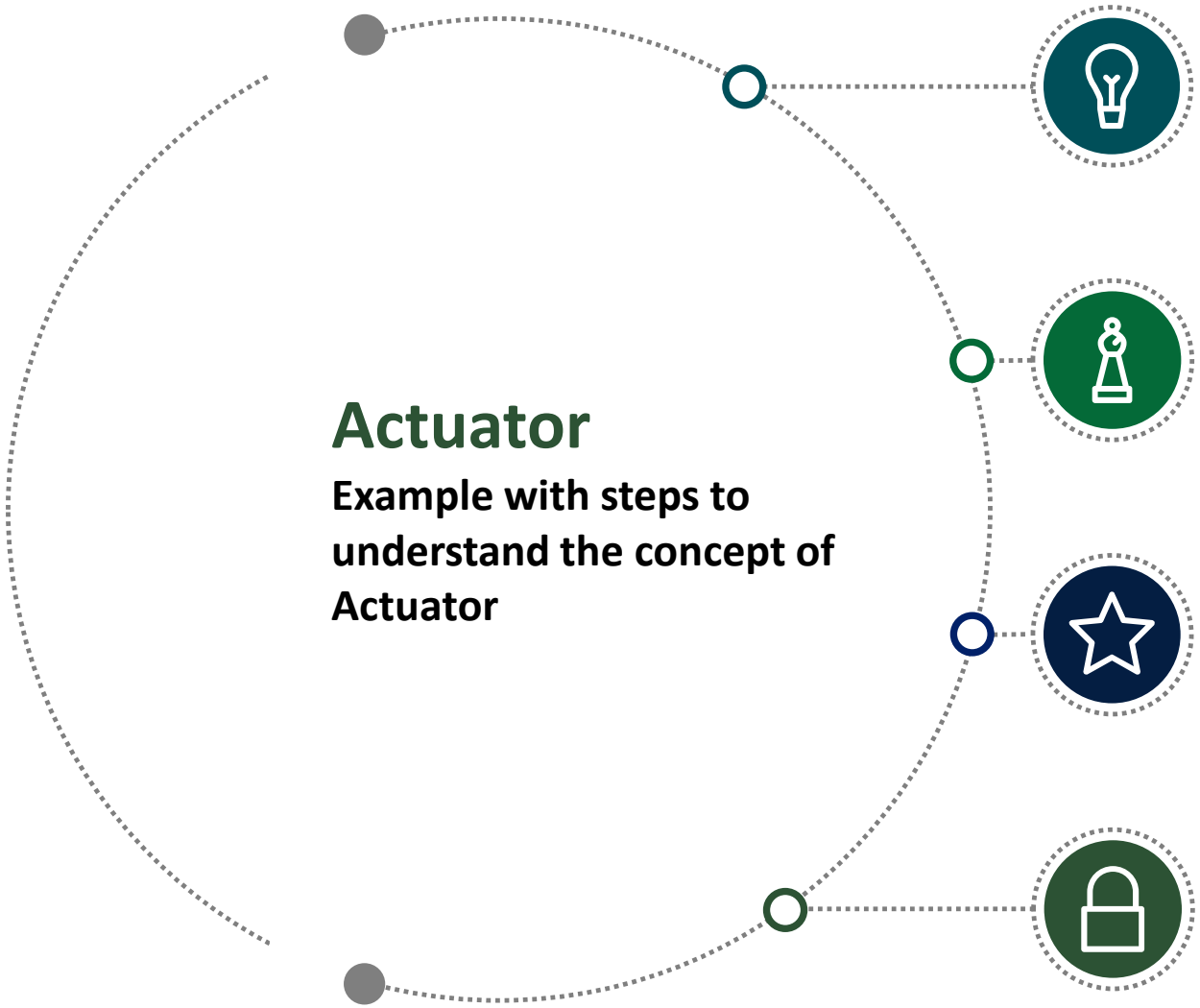
- We can enable actuator by injecting the dependency spring-boot-starter-actuator in the pom.xml file.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
    <version>2.2.2.RELEASE</version>
</dependency>
```

- The actuator endpoints allow us to monitor and interact with our Spring Boot application.
- Spring Boot includes number of built-in endpoints and we can also add custom endpoints in Spring Boot application.
- Some of the widely used Endpoints :
  - -actuator
  - -health
  - -info
  - -metrics

Spring Boot - Actuator

Example



Create Maven Project

- Open Spring Initializr **https://start.spring.io/** and create a Maven project
- Provide the Group name

Add Dependencies

- Spring Web
- Spring Boot Starter Actuator
- Spring Data Rest HAL Browser

Add Security Feature

- Add "**management.security.enabled=false**" to disable the security feature of the actuator in application.properties

Invoke Actuator

- Open the browser and invoke the URL **http://localhost:8080/actuator.**
- The application runs on port 8080 by default.
- Once the actuator has started, we can see the list of all the endpoints exposed over HTTP.

Knowledge Check

1. Which of the following is Actuator Feature?

- ○ End Point
- ○ Audit
- ○ Metrics
- ○ All of the above

2. Which condition should be added to disable the security feature of the actuator?

- ○ spring.h2.console.enabled=true
- ○ management.security.enabled=false
- ○ server.port = 8090
- ○ None of the above

3. Which dependency enables actuator?

- ○ spring-boot-starter-web
- ○ spring-boot-starter-actuator
- ○ spring-boot-starter-test
- ○ spring-boot-starter

4. Which feature shows application health information?

- ○ Health
- ○ Info
- ○ Metrics
- ○ Trace

Knowledge Check

5. It displays trace information for last few HTTP requests?

- PORT
- Enable
- TRACE
- All of the above

6. It displays arbitrary application information?

- trace
- metrics
- health
- info

7. Use of Spring Boot Actuator?

- Build
- Deploying
- Monitoring
- None of the above

8. Metrics gives all metrics related information for the current application?

- TRUE
- FLASE

Hands On

Spring Boot Actuator

Approach

- Create a Maven project using Spring Initializer
- Add spring-boot-starter-parent, spring-boot-starter-web, **spring-boot-starter-actuator**
- Create REST API Controller with **@GetMapping**
- Add following security related properties in application.properties
    - management.security.enabled = true
    - management.security.roles = ADMIN
    - security.basic.enabled = true
    - security.user.name = admin
    - security.user.password = admin
    - endpoints.cors.allowed-origins = http://example.com
    - endpoints.cors.allowed-methods = GET,POST
- Configure **Spring Security** to allow you to access the endpoints.
- Run the application. Once the server is Up, access API using following endpoints:
    - http://localhost:8080/env
    - http://localhost:8080/health
    - http://localhost:8080/metrics

Break – 15 min.

Spring Boot - CRUD Operations

Spring Boot - CRUD Operations

Introduction

What is CRUD OPERATION ?

- The CRUD stands for Create, Read/Retrieve, Update, and Delete. These are the four basic functions of the persistence storage.
- The CRUD operation can be defined as user interface conventions that allow view, search, and modify information through computer-based forms and reports.
- CRUD is data-oriented and the standardized use of HTTP action verbs. HTTP has a few important verbs.
- Within a database, each of these operations maps directly to a series of commands. However, their relationship with a RESTful API is slightly more complex.

Standard CRUD Operation

- CREATE
- READ
- UPDATE
- DELETE

Function

- It performs the Insert statement to create a new record.
- It Reads table records based on the input parameter.
- It executes an Update statement on the table. It is based on the input parameter.
- It Deletes a specified row in the table. It is also based on the input parameter.

Spring Boot - CRUD Operations

SQL vs HTTP Verbs vs REST - How CRUD Operations Works?

- CRUD operations are at the foundation of the most dynamic websites. Therefore, we should differentiate CRUD from the HTTP action verbs.
- Suppose we want to create a new record, we should use HTTP action verb POST.
- To update a record, use the PUT verb. Similarly, to delete a record, use the DELETE verb.
- Through CRUD operations, users and administrators have the right to retrieve, create, edit, and delete records online.
- We have many options for executing CRUD operations. One of the most efficient choices is to create a set of stored procedures in SQL to execute operations.

The CRUD operations are the major functions that are implemented in relational database applications.

Operation	SQL	HTTP verbs	RESTful Web Service
CREATE	INSERT	PUT/POST	POST
READ	SELECT	GET	GET
UPDATE	UPDATE	PUT/POST/PATCH	PUT
DELETE	DELETE	DELETE	DELETE

Spring Boot – CRUD Operations

Crud Repository vs. JPA Repository

CrudRepository

- CrudRepository does not provide any method for pagination and sorting.
- It is defined in the package **org.springframework.data.repository**. It extends the Spring Data Repository interface.
- It provides CRUD function only. It provides generic Crud operation on a repository
- It is used when we do not need the functions provided by JpaRepository and PagingAndSortingRepository.

Syntax :

```
public interface CrudRepository<T,ID> extends Repository<T,ID>
```

JpaRepository

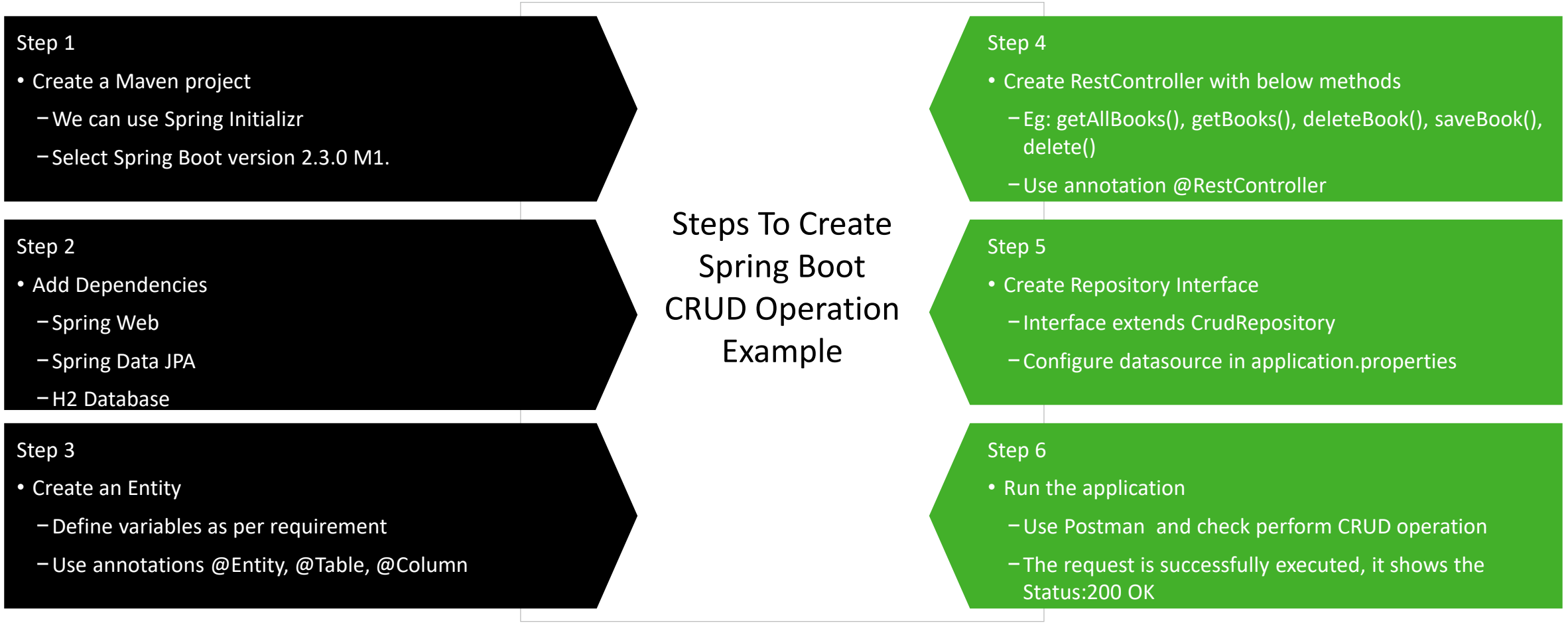
- JpaRepository extends PagingAndSortingRepository. It provides all the methods for implementing the pagination.
- JpaRepository extends both CrudRepository and PagingAndSortingRepository.
- It provides some extra methods along with the method of PagingAndSortingRepository and CrudRepository.
- It is used when we want to implement pagination and sorting functionality in an application.

Syntax :

```
public interface CrudRepository<T,ID> extends JpaRepository<T,ID>
```

Spring Boot - CRUD Operations

Example and Code Setup



Spring Boot – CRUD Operations

Demo and Walkthrough/Code setup

Model Class

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;
//mark class as an Entity
@Entity
//defining class name as Table name
@Table
public class Books
{
//Defining book id as primary key
@Id
@Column
private int bookid;
//  Getters and Setters
```

Controller

```
@RestController
public class BooksController
{
//autowire the BooksService class
@Autowired
BooksService booksService;
//creating a get mapping that retrieves all the books detail from the database
@GetMapping("/book")
private List<Books> getAllBooks()
{
return booksService.getAllBooks();
}
 // similarly try put,delete
```

Service

```
@Service
public class BooksService
{
@Autowired
BooksRepository
//getting all books record by using the method findaAll() of CrudRepository
public List<Books> getAllBooks()
{
List<Books> books = new ArrayList<Books>();
booksRepository.findAll().forEach(books1 -> books.add(books1));
return books;
}
//similarly update,delete
```

Application.properties

```
spring.datasource.url=jdbc:h2:mem:books_data
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
#enabling the H2 console
spring.h2.console.enabled=true
```

SpringBootCrudOperationApplication.java

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

Knowledge Check

1. How to implement Pagination and Sorting with Spring Boot?

- Spring Data JPA
- Spring Actuator
- Spring Cache
- All of the above

2. Annotation that makes class as table?

- @Row
- @Column
- @Table
- None of the above

3. Annotation used to retrieve the details?

- @GetMapping
- @RequestMapping
- @PutMapping
- None of the above

4. Annotation used to declare field as primary key?

- @Column
- @Row
- @Primary
- @Id

Knowledge Check

5. Repositories extended by JPA Repository?

- PagingAndSorting
- CrudRepository
- Both A and B
- SQL Repository

6. HTTP verb used to create new record?

- PUT
- GET
- POST
- DELETE

7. Dependencies required to perform CRUD operations?

- Spring Web
- Spring Data JPA
- H2 database
- All of the above

8. We can override versions in dependency?

- TRUE
- FLASE

Hands On

Spring Boot CRUD Operations

Approach

- Create a Maven project using Spring Initializer
- Add spring-boot-starter-parent, spring-boot-starter-web, spring-boot-starter-data-jpa
- Create REST API Controller with following methods. Feel free to add more methods
  - HTTP GET /employees – Returns list of the employees.
  - HTTP POST /employees – Add an employee in the employees collection.
- Create Model class and JPA Entity class with following details :
  - Class name : Employee
  - Fields : id, firstname, lastname, email
- Create Service class and JPA Repository
- Run the application. Once the server is up, access API using some rest client.

Any Questions ?

Thank You

Deloitte.