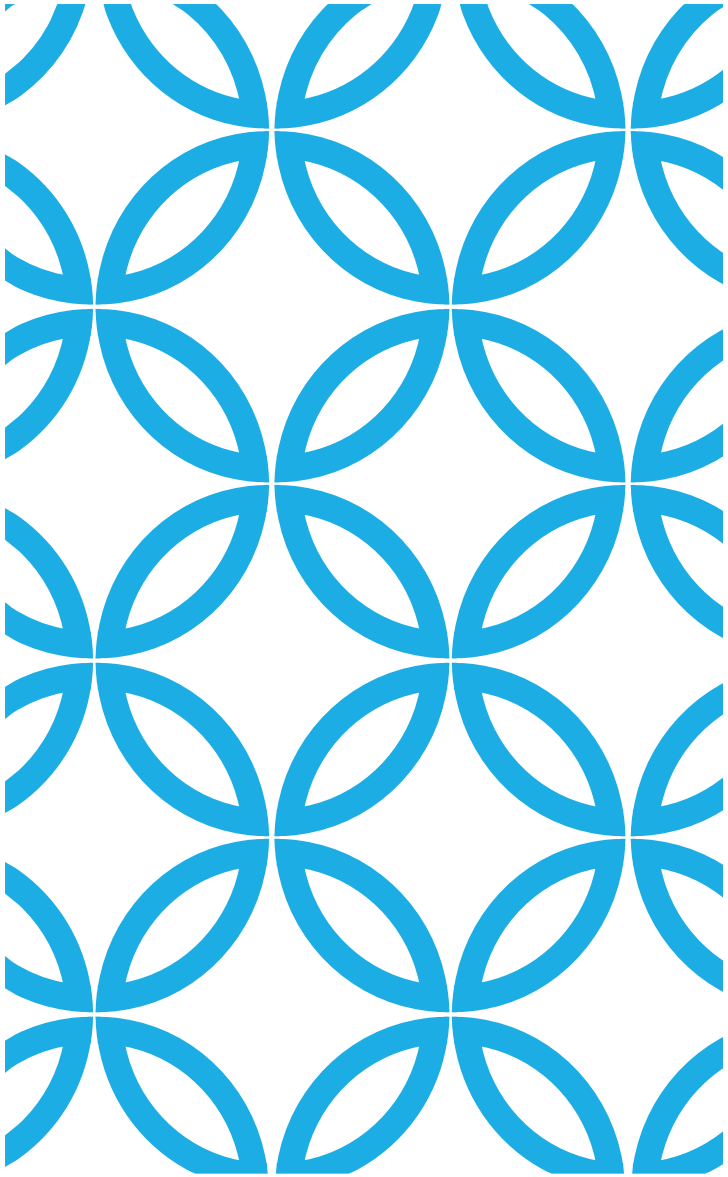


SPRING BOOT

Mentor : Rahul Dixit

GETTING STARTED WITH SPRING BOOT

WHY Spring Boot?

- You can build web apps & REST API WITHOUT Spring Boot
- Then What is the need for Spring Boot?

WHAT are the goals of Spring Boot?

HOW does Spring Boot work?

COMPARE Spring Boot vs Spring MVC vs Spring

GETTING STARTED WITH SPRING BOOT - APPROACH

1. Understand the world before Spring Boot.

2: Create a Spring Boot Project.

3: Build a simple REST API using Spring Boot.

4: Understand the MAGIC of Spring Boot.

- Spring Initializr
- Starter Projects
- Auto Configuration
- Developer Tools
- Actuator ...

WORLD BEFORE SPRING BOOT!

Setting up Spring Projects before Spring Boot was NOT easy!

We needed to configure a lot of things before we have a production-ready application

DEPENDENCY MANAGEMENT

1. Dependencies refer to external libraries or modules that an application requires to function correctly.
2. Dependency management is the process of handling and organizing these external libraries or modules.
3. It ensures that the application has the necessary dependencies and versions to function correctly.
4. In Spring Boot, dependency management is handled by build tools like
   - Maven
   - Gradle.s

MAVEN DEPENDENCY

1. Maven is a popular build tool for managing dependencies in Spring Boot applications.
2. Maven uses an XML file called the "pom.xml" to configure and manage dependencies.

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

DEPENDENCY SCOPES

- Scope determines where the dependency is available in the project.
  - **Compile:** This scope is the default scope. The dependency is available in all Proprietary content. All rights reserved. Unauthorized use or distribution prohibited. phases of the project.
  - **Provided:** The dependency is provided by the container (e.g., Tomcat) during runtime, and it is not included in the packaged application.
  - **Runtime:** The dependency is required during runtime, but it is not required for compilation.
  - **Test:** The dependency is only required for testing purposes and is not included in the packaged application.

WORLD BEFORE SPRING BOOT - 1 - DEPENDENCY MANAGEMENT

Manage frameworks and versions

- REST API - Spring framework, Spring MVC framework, JSON binding framework, ..
- Unit Tests - Spring Test, Mockito, JUnit, ...

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>6.2.2.RELEASE</version>
</dependency>
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.13.3</version>
</dependency>
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

WORLD BEFORE SPRING BOOT - 2 - WEB.XML

```
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/todo-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/*</url-pattern>
</servlet-mapping>
```

- **Example**: Configure **DispatcherServlet** for Spring MVC

WORLD BEFORE SPRING BOOT - 3 - SPRING CONFIGURATION

```
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix">
        <value>/WEB-INF/views/</value>
    </property>
    <property name="suffix">
        <value>.jsp</value>
    </property>
</bean>
```

Define your Spring Configuration

- Component Scan
- View Resolver ....

WORLD BEFORE SPRING BOOT - 4 - NFRS

```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.2</version>
    <configuration>
        <path>/</path>
        <contextReloadable>true</contextReloadable>
    </configuration>
</plugin>

<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
</dependency>
```

Logging

Error Handling

Monitoring

WORLD BEFORE SPRING BOOT!

Setting up Spring Projects before Spring Boot was NOT easy!

1: Dependency Management (pom.xml)

2: Define Web App Configuration (web.xml)

3: Manage Spring Beans (context.xml)

4: Implement Non-Functional Requirements (NFRs)

AND repeat this for every new project!

Typically takes a few days to setup for each project (and countless hours to maintain)

UNDERSTANDING POWER OF SPRING BOOT


```
[
  {
    "courseId": 1,
    "courseName": "SpringFrameWork",
    "courseAuthor": "Rahul"
  },
  {
    "courseId": 2,
    "courseName": "SpringBoot",
    "courseAuthor": "Rahul"
  },
  {
    "courseId": 3,
    "courseName": "Azure",
    "courseAuthor": "Rahul"
  }
]
```


1: Create a Spring Boot Project

2: Build a simple REST API using Spring Boot

WHAT'S THE MOST IMPORTANT GOAL OF SPRING BOOT?

Help you build PRODUCTION-READY apps QUICKLY

- Build QUICKLY
  - Spring Initializr
  - Spring Boot Starter Projects
  - Spring Boot Auto Configuration
  - Spring Boot DevTools
- Be PRODUCTION-READY
  - Logging
  - Different Configuration for Different Environments
    - Profiles, ConfigurationProperties
  - Monitoring (Spring Boot Actuator) ...

EXPLORING SPRING BOOT STARTER PROJECTS

I need a lot of frameworks to build application features:

- Build a REST API: I need Spring, Spring MVC, Tomcat, JSON conversion...
- Write Unit Tests: I need Spring Test, JUnit, Mockito, ...

How can I group them and make it easy to build applications?

- Starters: Convenient dependency descriptors for diff. features

Spring Boot provides variety of starter projects:

- Web Application & REST API - Spring Boot Starter Web (spring-webmvc, springweb, spring-boot-starter-tomcat, spring-boot-starter-json)
- Unit Tests - Spring Boot Starter Test
- Talk to database using JPA - Spring Boot Starter Data JPA
- Talk to database using JDBC - Spring Boot Starter JDBC
- Secure your web application or REST API - Spring Boot Starter Security

(REMEMBER) Starters: Define all application dependencies

EXPLORING SPRING BOOT AUTO CONFIGURATION

I need lot of configuration to build Spring app:

- Component Scan, DispatcherServlet, Data Sources, JSON Conversion, ...

How can I simplify this?

- Auto Configuration: Automated configuration for your app
  - Decided based on:
    - Which frameworks are in the Class Path?
    - What is the existing configuration (Annotations etc)?

Example: Spring Boot Starter Web

- Dispatcher Servlet (DispatcherServletAutoConfiguration)
- Embedded Servlet Container - Tomcat is the default (EmbeddedWebServerFactoryCustomizerAutoConfiguration)
- Default Error Pages (ErrorMvcAutoConfiguration)
- Bean<->JSON (JacksonHttpMessageConvertersConfiguration)

UNDERSTANDING THE GLUE - @SPRINGBOOTAPPLICATION

Questions:

- Who is launching the Spring Context?
- Who is triggering the component scan?
- Who is enabling auto configuration?

Answer: @SpringBootApplication

1: @SpringBootConfiguration: Indicates that a class provides Spring Boot application @Configuration.

2: @EnableAutoConfiguration: Enable auto-configuration of the Spring Application Context,

3: @ComponentScan: Enable component scan (for current package, by default)

BUILD FASTER WITH SPRING BOOT DEVTOOLS

Increase developer productivity

Why do you need to restart the server manually for every code change?

Remember: For pom.xml dependency changes, you will need to restart server manually

Not able to handle change in pom.xml

Add Dependency DevTools In pom.xml

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

MANAGING APP. CONFIGURATION USING PROFILES

Applications have different environments: Dev, QA, Stage, Prod, ...

Different environments need different configuration:

- Different Databases
- Different Web Services

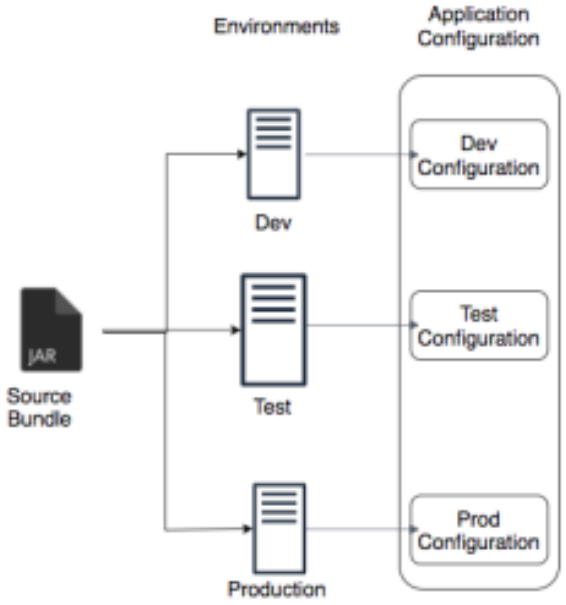
How can you provide different configuration for different environments?

- Profiles: Environment specific configuration

How can you define externalized configuration for your application?

- ConfigurationProperites: Define externalized configuration

(Create Separate Profile and Configuration w.r.t to Environment)



i.e. dev : - trace level profile

- (application-dev.properties)->logging.level.org.springframework=debug

i.e prod :- info level profile

- (application-prod.properties)->logging.level.org.springframework=info

Trace - Everything

debug – more than info

Info – all things which are logged @info level

Warning – little more than error

Error – only error

Off – off the log

(How to create profile go to Example com.rahul.boot.SpringWithRest CurrencyServiceConfiguration)

MONITOR APPLICATIONS USING SPRING BOOT ACTUATOR

Monitor and manage your application in your production

Provides a number of endpoints:

- beans - Complete list of Spring beans in your app
- health - Application health information
- metrics – Application metrics
- mappings - Details around Request Mappings

Beans-"http://localhost:8080/actuator/beans"

caches-cache

-http://localhost:8080/actuator/caches/{cache}

Caches-

http://localhost:8080/actuator/caches

Health-

http://localhost:8080/actuator/health

health-path

-http://localhost:8080/actuator/health/{*path}

Info-http://localhost:8080/actuator/info

Conditions-http://localhost:8080/actuator/conditions

Configprops-http://localhost:8080/actuator/configprops

configprops-prefix-http://localhost:8080/actuator/configprops/{prefix}

Env-http://localhost:8080/actuator/env

env-toMatch

-http://localhost:8080/actuator/env/{toMatch}

Loggers-http://localhost:8080/actuator/loggers

loggers-name

-"http://localhost:8080/actuator/loggers/{name}"

Heapdump-

"http://localhost:8080/actuator/heapdump"

Threaddump

-"http://localhost:8080/actuator/threaddump"

metrics-requiredMetricName

-http://localhost:8080/actuator/metrics/{requiredMetricName}

Metrics-

http://localhost:8080/actuator/metrics

Scheduledtasks

-http://localhost:8080/actuator/scheduledtasks

Mappings

-"http://localhost:8080/actuator/mappings"

UNDERSTANDING SPRING BOOT VS SPRING MVC VS SPRING

Spring Boot vs Spring MVC vs Spring: What's in it?

- Spring Framework: Dependency Injection
  - @Component, @Autowired, Component Scan etc..
  - Just Dependency Injection is NOT sufficient (You need other frameworks to build apps)
    - Spring Modules and Spring Projects: Extend Spring Eco System
      - Provide good integration with other frameworks (Hibernate/JPA, JUnit & Mockito for Unit Testing)
- Spring MVC (Spring Module): Simplify building web apps and REST API
  - Building web applications with Struts was very complex
  - @Controller, @RestController, @RequestMapping("/courses")
- Spring Boot (Spring Project): Build PRODUCTION-READY apps QUICKLY
  - Starter Projects - Make it easy to build variety of applications
  - Auto configuration - Eliminate configuration to setup Spring, Spring MVC and other frameworks!
  - Enable non functional requirements (NFRs):
    - Actuator: Enables Advanced Monitoring of applications
    - Embedded Server: No need for separate application servers!
    - Logging and Error Handling
    - Profiles and ConfigurationProperties

DOUBTS

APPROACH

Create a Spring Boot Project with H2

Create COURSE table

Use Spring JDBC to play with COURSE table

Use JPA and Hibernate to play with COURSE table

Use Spring Data JPA to play with COURSE table

SPRING BOOT AUTO CONFIGURATION

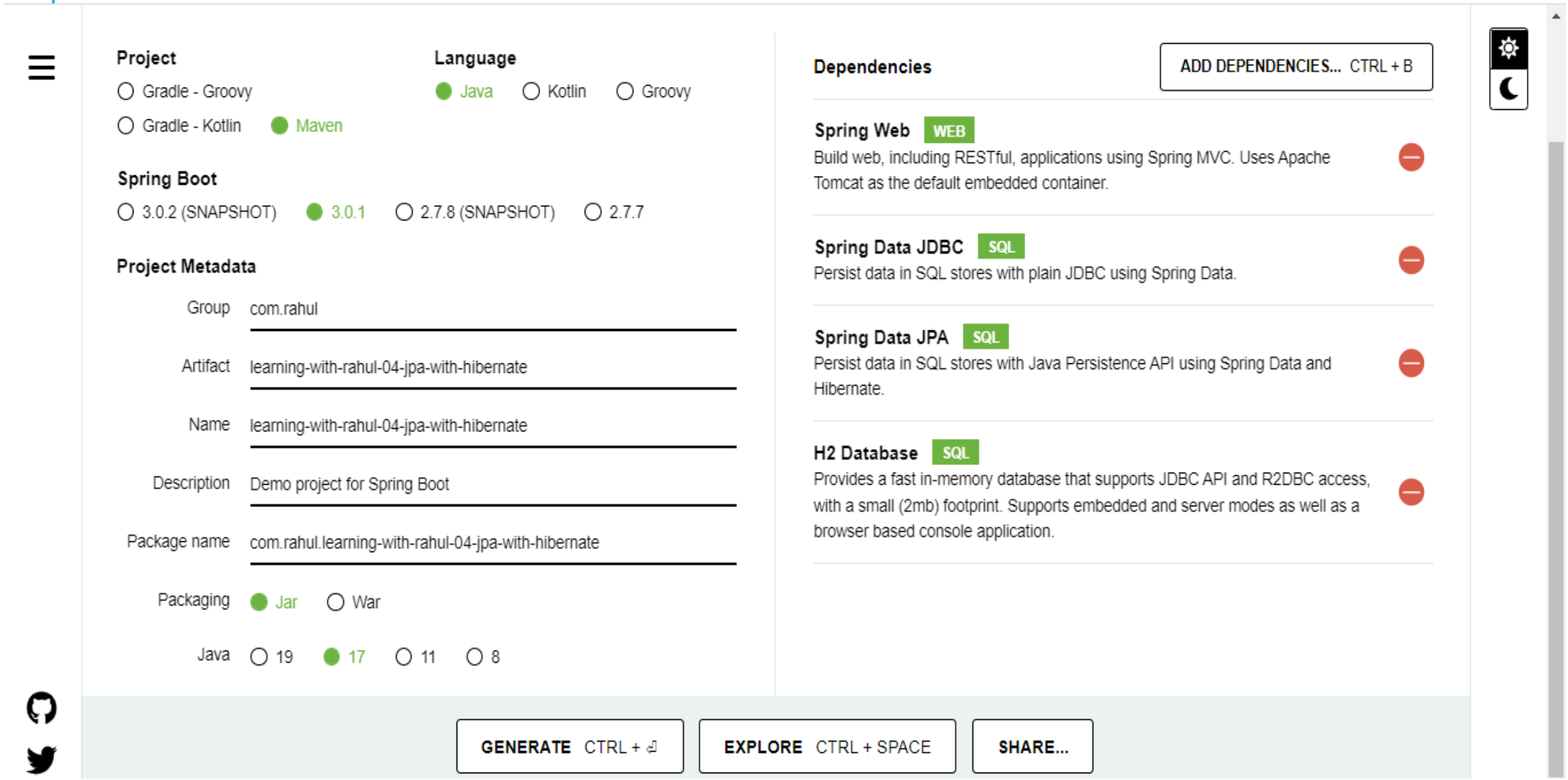
We added Data JPA and H2 dependencies:

- Spring Boot Auto Configuration does some magic:
  - Initialize JPA and Spring Data JPA frameworks
  - Launch an in memory database (H2)
  - Setup connection from App to in-memory database
  - Launch a few scripts at startup (example: data.sql, schema.sql)

Remember - H2 is in memory database

- Does NOT persist data
- Great for learning
- BUT NOT so great for production

BOOT + JPA + HIBERNATE



JDBC TO SPRING JDBC TO JPA TO SPRING DATA JPA

JDBC

- Write a lot of SQL queries! (delete from todo where id=?)
- And write a lot of Java code

Spring JDBC (E.g :- learning-with-rahul-04-jdbcTemplate-with-hibernate)

- Write a lot of SQL queries (delete from todo where id=?)
- BUT lesser Java code

JPA (E.g.learning-with-rahul-04-jpa-with-hibernate)

- Do NOT worry about queries
- Just Map Entities to Tables!

Spring Data JPA (E.g. learning-with-rahul-04-springdatajpa-with-hibernate)

- Let's make JPA even more simple!
- I will take care of everything!

JDBC TO SPRING JDBC

JDBC example:

```
public void deleteTodo(int id) {
PreparedStatement st = null;
try {
st = db.conn.prepareStatement("delete from todo where id=?");
st.setInt(1, id);
st.execute();
} catch (SQLException e) {
logger.fatal("Query Failed : ", e);
} finally {
if (st != null) {
try {st.close();}
catch (SQLException e) {}}}}
```

SPRING JDBC EXAMPLE

```
public void deleteTodo(int id) {
jdbcTemplate.update("delete from todo where id=?", id);
}
```

JPA EXAMPLE

```
@Repository
public class PersonJpaRepository {
@PersistenceContext
EntityManager entityManager;
public Person findById(int id) {
return entityManager.find(Person.class, id);}
public Person update(Person person) {
return entityManager.merge(person);}
public Person insert(Person person) {
return entityManager.merge(person);}
public void deleteById(int id) {.....
```

SPRING DATA JPA EXAMPLE

```
public interface TodoRepository extends JpaRepository{}
```

HIBERNATE VS JPA

1. JPA defines the specification. It is an API.
   - How do you define entities?
   - How do you map attributes?
   - Who manages the entities?
2. Hibernate is one of the popular implementations of JPA
3. Using Hibernate directly would result in a lock in to Hibernate
   - There are other JPA implementations (Toplink, for example)

THANKS