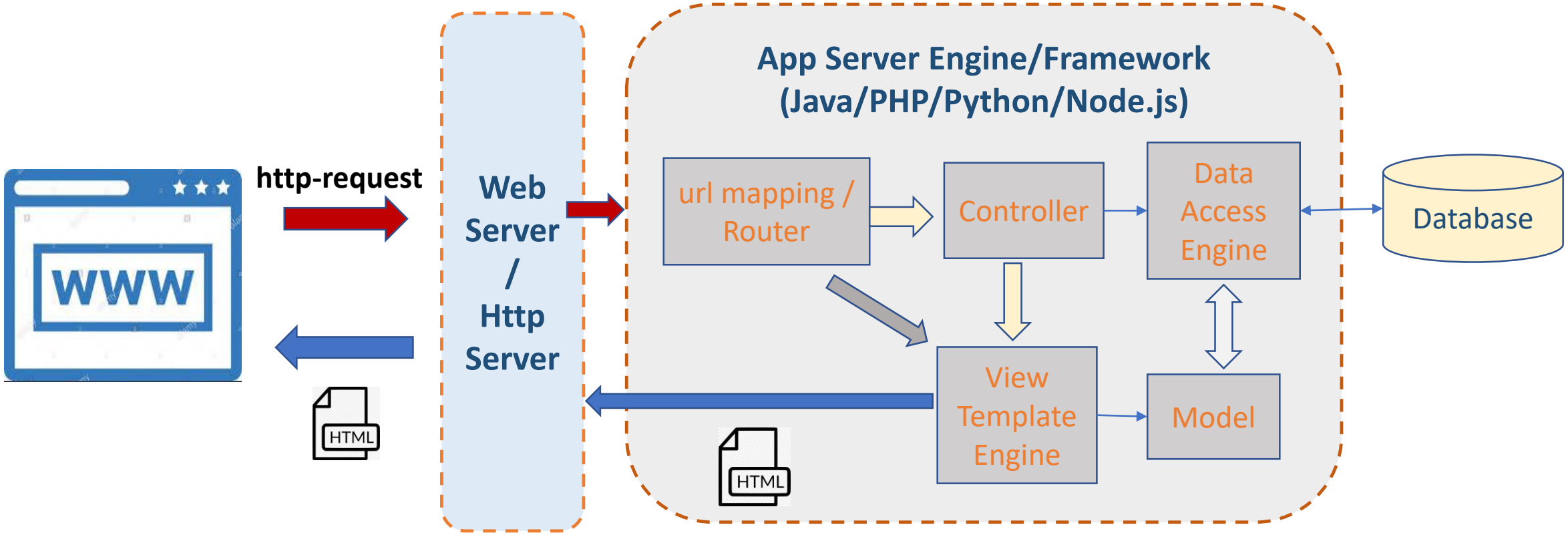
Topics

- Introduction to Spring boot
  - Week 1, 2:
    - MVC
    - Spring JPA
  - Week 3:
    - Rest API + Vue.js
- Week 4:
  - Introduction to PHP Laravel / Adonis Node.js / Python Django
    - Installing
    - MVC
    - Rest API

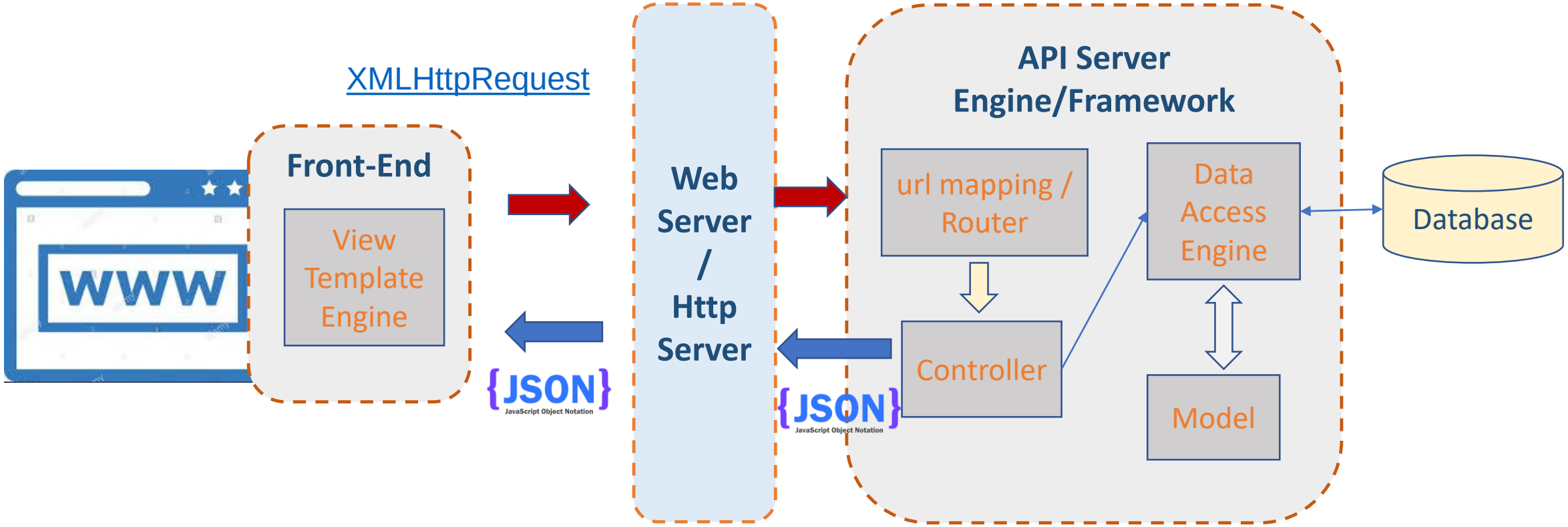
Evaluation

- Final Exam 20%
- Lab Exam 25%
- Assignment 5%
  - Programming tricks and Technics
    - Issue
    - Case study
    - Coding example
    - Output

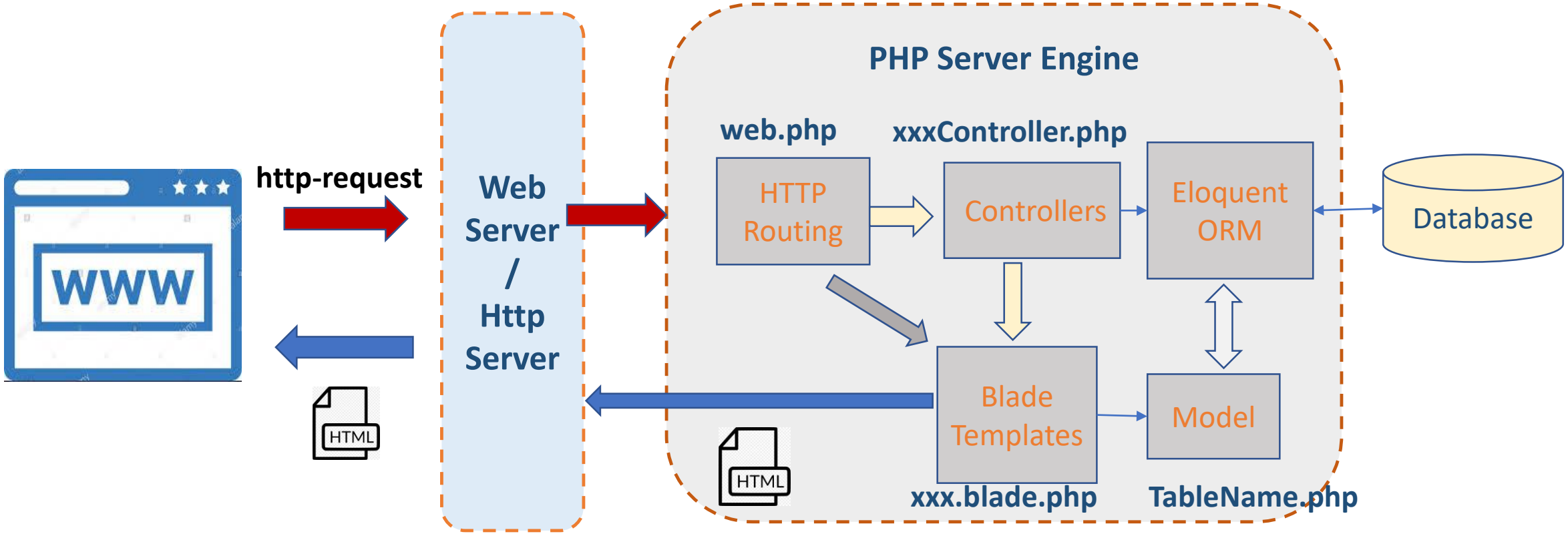
MVC Web Application Architectures (classic)



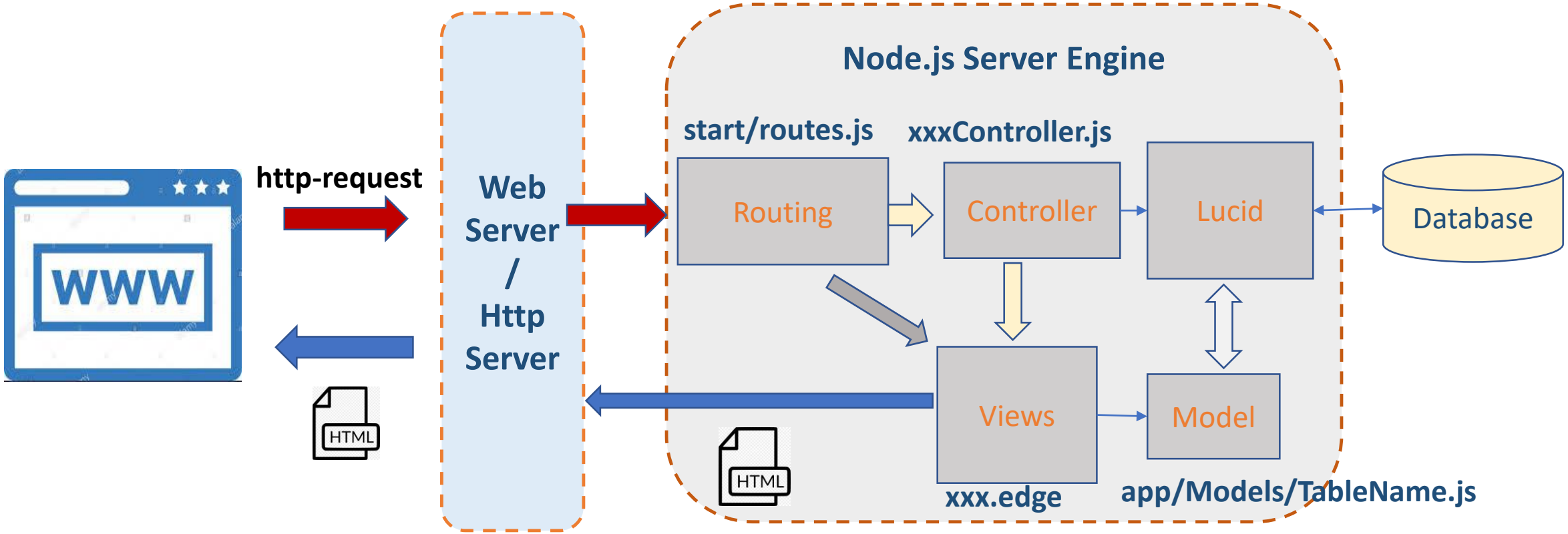
MVC Web Application Architectures (modern/spa)



Laravel MVC Framework



Adonis MVC Framework



Spring Framework

- Spring at its core is a dependency injection container that provides flexibility to configure beans in multiple ways, such as XML, Annotations, and JavaConfig.
- Spring is the most popular Java-based framework for building enterprise applications.
- Provides a rich ecosystem of projects to address modern application needs
  - Security
  - Simplified access to relational and NoSQL datastores
  - Batch processing
  - etc.
- Spring is a very flexible and customizable framework, **there are multiple ways to configure the application**.
- It can be overwhelming to the beginners.

Spring Boot

- Spring Boot addresses this “Spring applications need complex configuration” problem by using its powerful autoconfiguration mechanism.
- Spring Boot is an opinionated framework following the “Convention Over Configuration” approach
- Which helps build Spring-based applications quickly and easily.
- The main goal of Spring Boot is to quickly create Spring-based applications without requiring the developers to write the same boilerplate configuration again and again.

Spring sub-projects

- Spring Data:
  - Simplifies data access from relational and NoSQL datastores.
- Spring Batch:
  - Provides a powerful batch-processing framework.
- Spring Security:
  - Robust security framework to secure applications.

- https://spring.io/projects

Spring Configuration Styles (1/3)

- XML-Based Configuration

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
destroymethod="close">
    <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
    <property name="url" value="jdbc:mysql://localhost:3306/test"/>
    <property name="username" value="root"/>
    <property name="password" value="secret"/>
</bean>

<!-- DSL based configuration : domain-specific language -->
<beans>
    <jee:jndi-lookup id="entityManagerFactory" jndi-name="persistence/defaultPU"/>
</beans>
```

Spring Configuration Styles (2/3)

- Annotation-Based Configuration

```
@Repository
public class JdbcUserDao {
    private DataSource dataSource;
    @Autowired
    public JdbcUserDao(DataSource dataSource) {
        this.dataSource = dataSource;
    }
    ...
    ...
}
```

Spring Configuration Styles (3/3)

- JavaConfig-Based Configuration

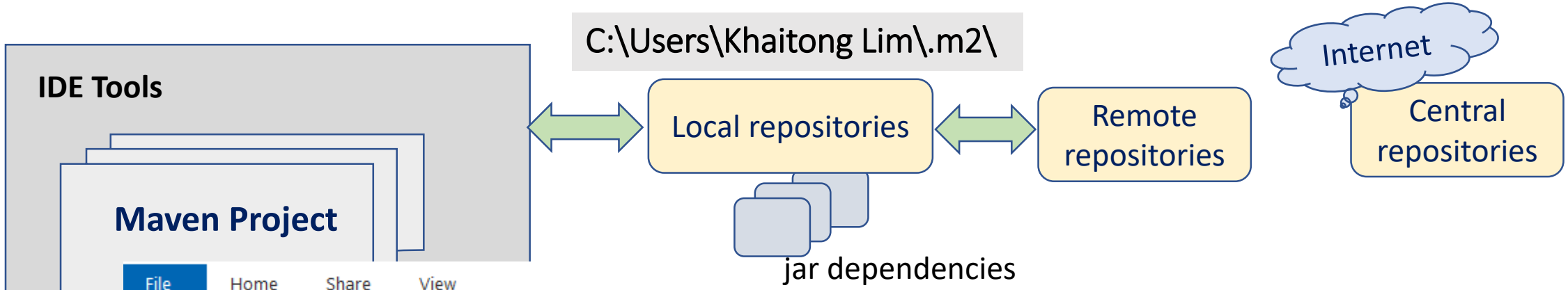
```
@Configuration
public class AppConfig {
    @Bean
     public UserDao userDao(DataSource dataSource){
        return new JdbcUserDao(dataSource);
     }
    @Bean
    public DataSource dataSource() {
        BasicDataSource dataSource = new BasicDataSource();
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");
        dataSource.setUrl("jdbc:mysql://localhost:3306/test");
        dataSource.setUsername("root");
        dataSource.setPassword("secret");
        return dataSource;
     }
}
```

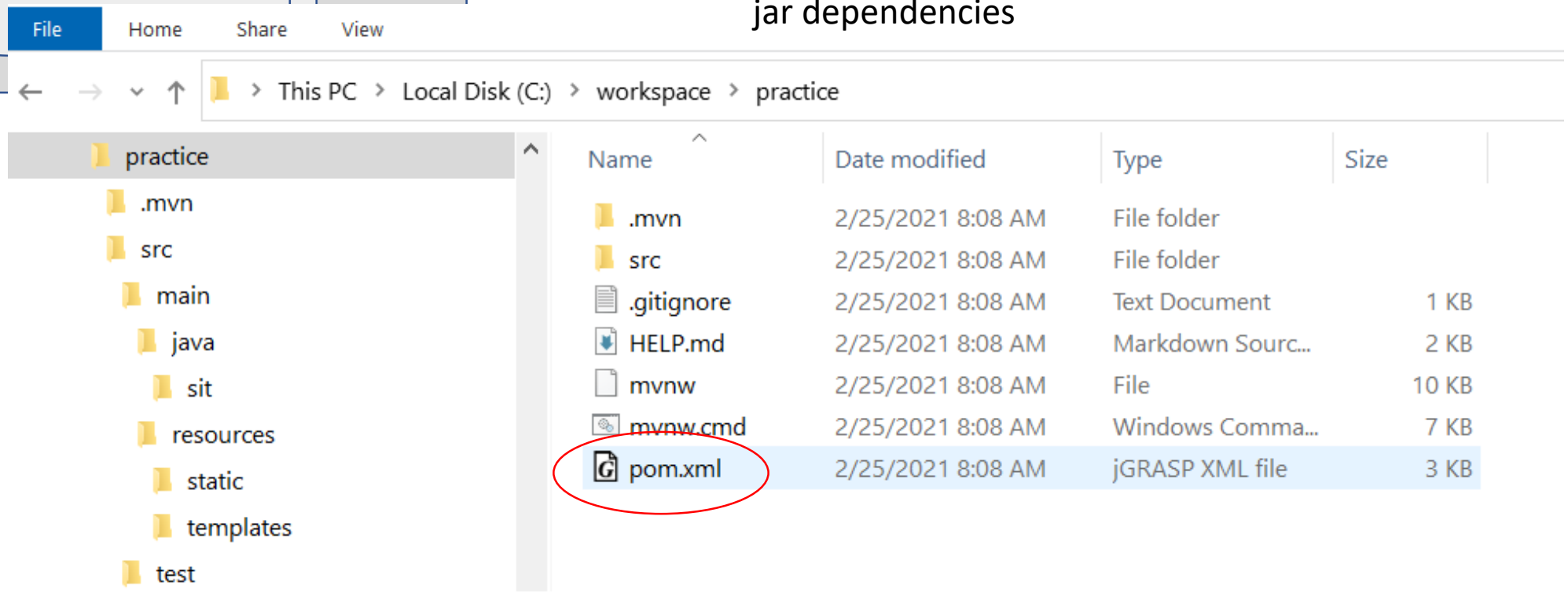
Java Build Tools

- Apache ANT: Another Neat Tool
  - Using XML to write build scripts
  - Base on Apache Ivy (local dependency management)
  - extremely flexible and does not impose coding conventions or directory layouts
- Apache Maven
  - Improve Ant (use XML in different structure)
  - Impose coding conventions or directory layouts
  - Download dependencies over the network
- Gradle
  - DSL: Domain-Specific Language base on Apache Gloovy (JVM Language)

- https://technologyconversations.com/2014/06/18/build-tools/
- https://gradle.org/maven-vs-gradle/

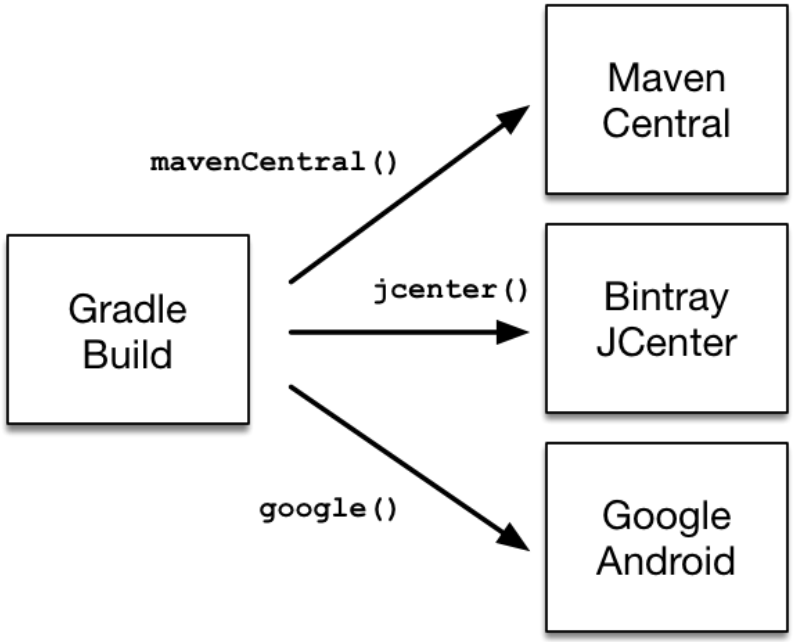
https://repo1.maven.org/maven2/



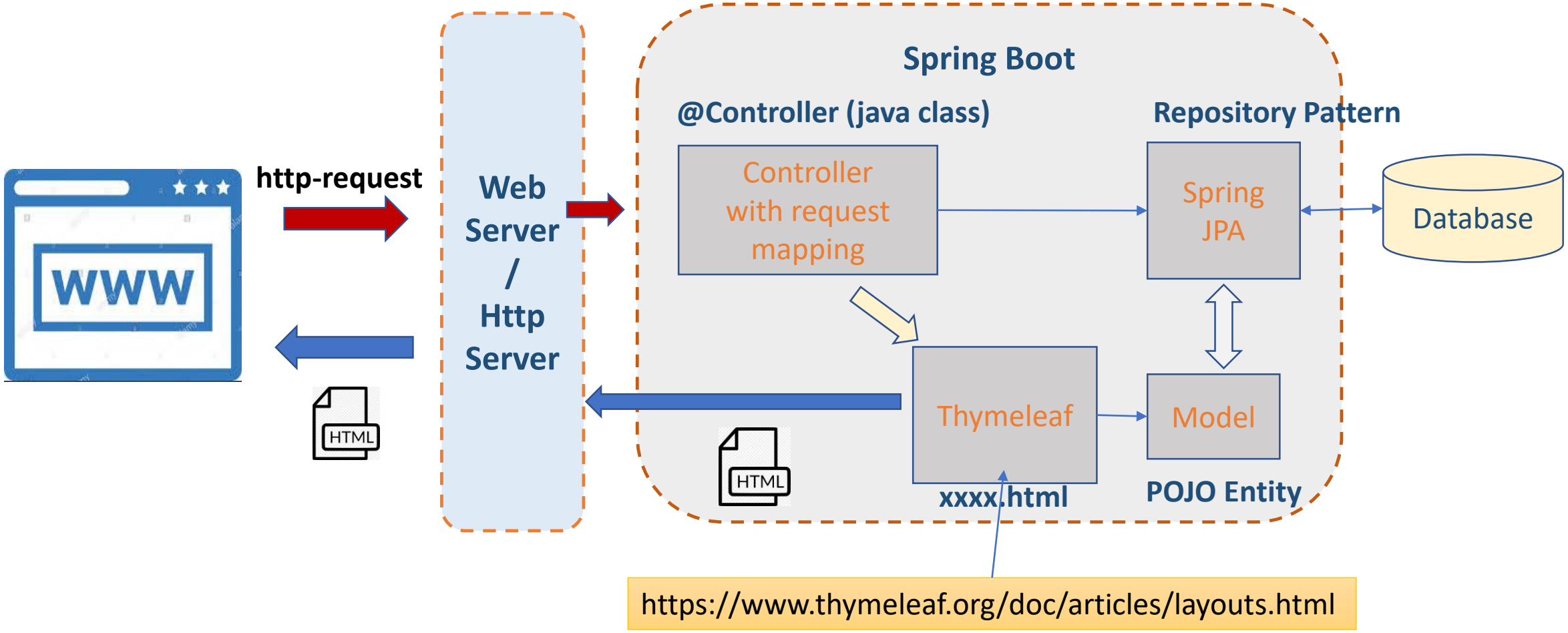


Gradle

```
.
├── build.gradle
└── src
    └── main
        ├── java
        │   └── HelloWorld.java
        └── kotlin
            └── Utils.kt
```

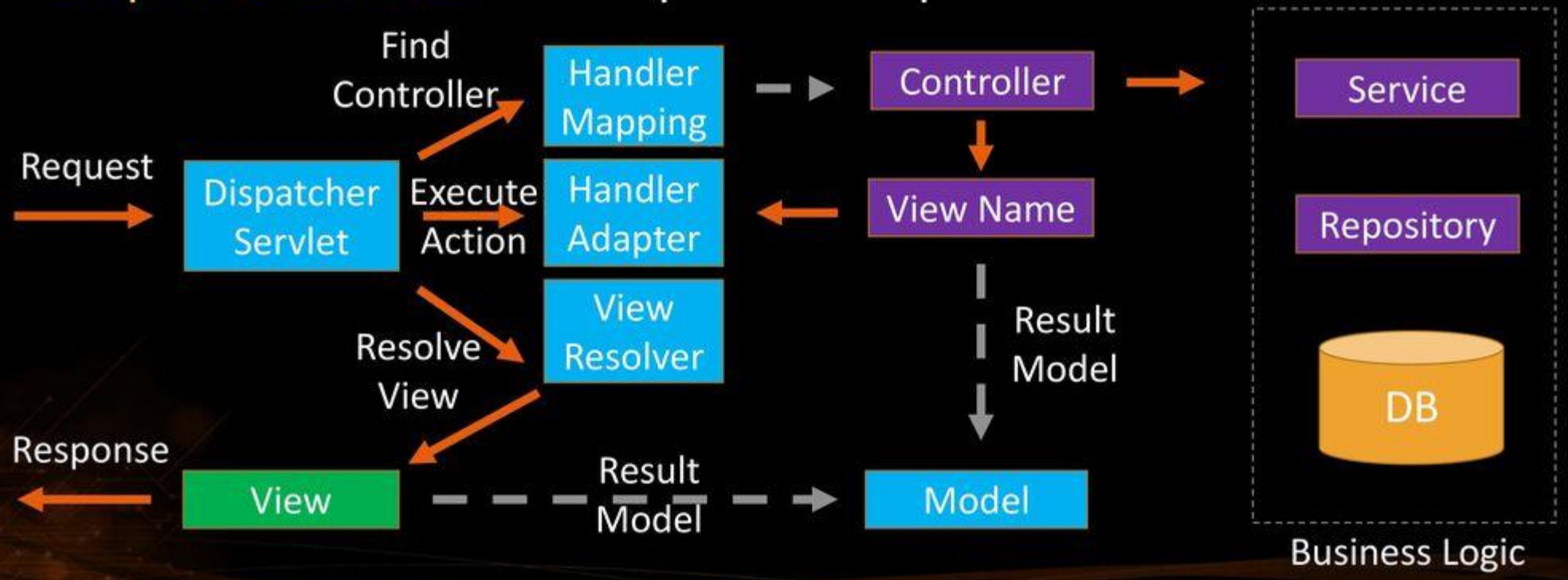


Spring MVC (classic)

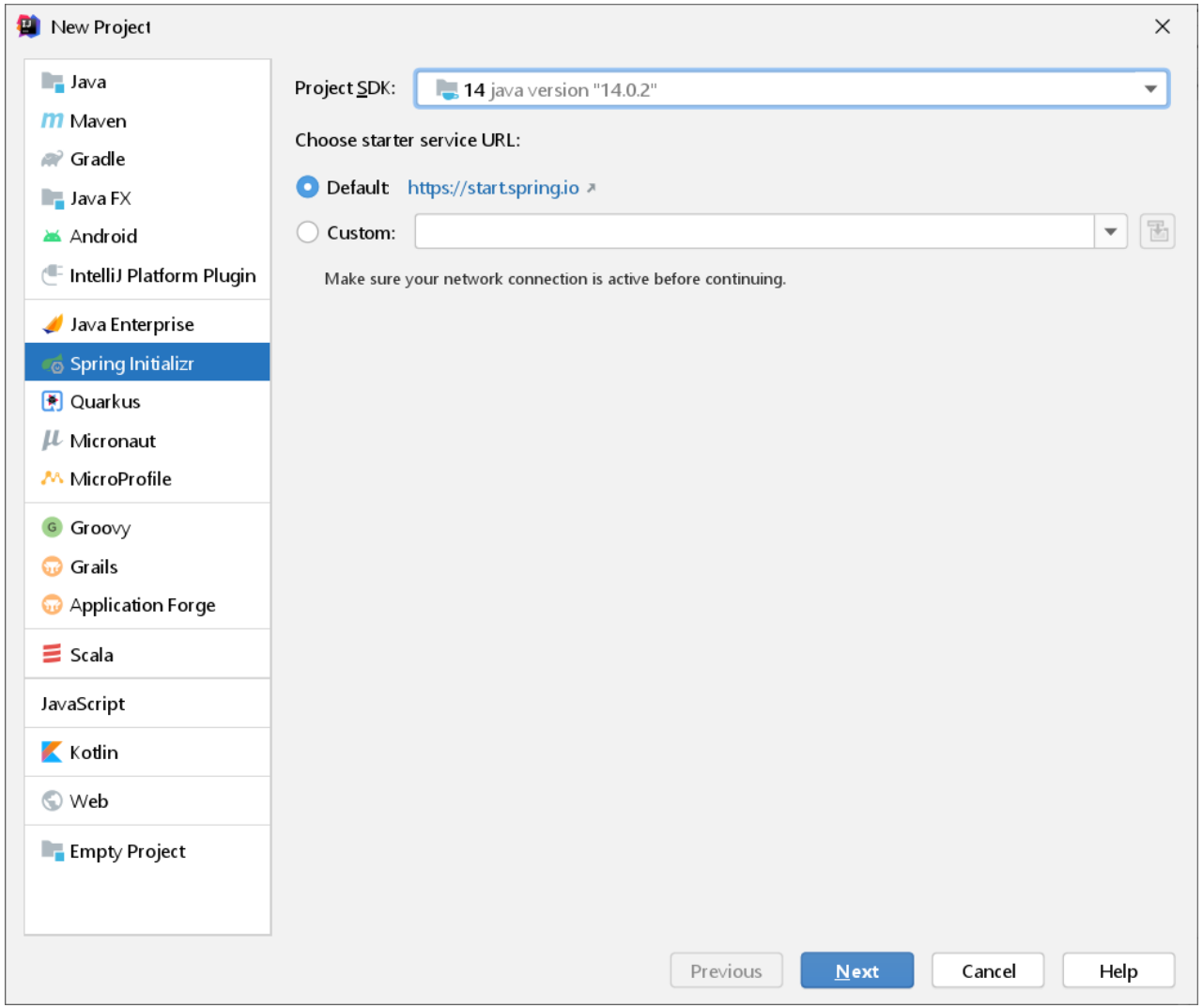


What is Spring MVC?

- Model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers



Spring Boot Initializer (using IDE plugin)



Spring Boot Initializer web site: https://start.spring.io

spring initializr

Project
- Maven Project
- Gradle Project

Language
- Java
- Kotlin
- Groovy

Spring Boot
- 2.5.0 (SNAPSHOT)
- 2.5.0 (M2)
- 2.4.4 (SNAPSHOT)
- 2.4.3
- 2.3.10 (SNAPSHOT)
- 2.3.9

Project Metadata

Group	sit.int204
Artifact	practice
Name	practice
Description	Demo project for Spring Boot
Package name	sit.int204.practice
Packaging	Jar / War
Java	15 / 11 / 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL
MySQL JDBC and R2DBC driver.

Spring Configuration Processor DEVELOPER TOOLS
Generate metadata for developers to offer contextual help and "code completion" when

GENERATE CTRL + ⏎ | EXPLORE CTRL + SPACE | SHARE...

g.io

Maven Command Line

- mvnw –version
- Maven Phases
- Although hardly a comprehensive list, these are the most common default lifecycle phases executed.

validate: validate the project is correct and all necessary information is available

compile: compile the source code of the project

test: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed

package: take the compiled code and package it in its distributable format, such as a JAR.

integration-test: process and deploy the package if necessary into an environment where integration tests can be run

verify: run any checks to verify the package is valid and meets quality criteria

install: install the package into the local repository, for use as a dependency in other projects locally

deploy: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

JAVA PROJECTS
- practice
  - src/main/java
    - sit.int204.practice
      - PracticeApplication
    - sit.int204.practice.controller
    - sit.int204.practice.model
    - sit.int204.practice.repository
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-11]
  - Maven Dependencies
    - HikariCP-3.4.5.jar C:/Users/Khaitong Lim/.m2/...
    - accessors-smart-1.2.jar C:/Users/Khaitong Li...
    - android-json-0.0.20131108.vaadin1.jar C:/...
    - antlr-2.7.7.jar C:/Users/Khaitong Lim/.m2/rep...
    - apiguardian-api-1.1.0.jar C:/Users/Khaitong ...
    - asm-5.0.4.jar C:/Users/Khaitong Lim/.m2/repo...
    - aspectjweaver-1.9.6.jar C:/Users/Khaitong Li...
- MAVEN
- SPRING BOOT DASHBOARD

Visual Studio Code

Run | Debug

```
public static void main(String[] args)
    SpringApplication.run(PracticeAppl
}
```

14

TERMINAL PROBLEMS 2 OUTPUT DEBUG CONSOLE

```
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Khaitong Lim\eclipse-workspace\practice.zip_
g-0.31.0\scripts\launcher.bat" "C:\Program Files\Java
sers\KHAITO~1\AppData\Local\Temp\cp_b4dtxmuohdwepeykp

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.2)

2021-02-23 15:01:26.297  INFO 30428 --- [  restartedM
2 on DESKTOP-EP3MPPT with PID 30428 (C:\Users\Khaitor
```

master* ⊗ 0 ⚠ 2 Spring Boot-PracticeApplication<practice> (practice) ✓ java | ✓ PracticeApplication.java local

Project Explorer

- practice
  - src/main/java
    - sit.int204.practice
      - PracticeApplication.java
    - sit.int204.practice.controller
    - sit.int204.practice.model
    - sit.int204.practice.repository
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-11]
  - Maven Dependencies
  - src
  - target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
- Servers
- spring5
- TestSpringMvc

PracticeApplication.java

```java
package sit.int204.practice;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class PracticeApplication {

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

}
```

Tasks | Console | Servers | Coverage

PracticeApplication [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jr

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.2)

2021-02-23 15:41:05.661  INFO 27284 --- [  restartedMain] sit.int204.
```

practice
- Source Packages
  - sit.int204.practice
    - PracticeApplication.java
  - sit.int204.practice.controller
  - sit.int204.practice.model
  - sit.int204.practice.repository
- Test Packages
  - sit.int204.practice
    - PracticeApplicationTests.java
- Other Sources
  - src/main/resources
    - <default package>
    - static
    - static.css
    - templates
    - templates.layout
- Dependencies
- Runtime Dependencies
- Test Dependencies
- Java Dependencies
  - JDK 14 (Default)
- Project Files
  - pom.xml
  - settings.xml
- SimpleJavaProject
- SimpleRegisterWebApp-1.0-SNAPSHO

```
4   import org.springframework.boot.autoconfigure.SpringBootApp
5
6   @SpringBootApplication
7   public class PracticeApplication {
8
9       public static void main(String[] args) {
10          SpringApplication.run( primarySource: PracticeApplication.clas
11      }
12
13  }
```

Apache NetBeans IDE

Output - Run (practic...

```
--- exec-mave                                          practice -

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.2)

2021-02-23 15:55:52.224  INFO 3804 --- [  restartedMain] si
2021-02-23 15:55:52.227  INFO 3804 --- [  restartedMain] si
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help webmvc - ProductsRestController.jav

webmvc

Project

webmvc C:\Users\Khaitong Lim\IdeaProjects\web
- .idea
- .mvn
- src
  - main
    - java
      - sit.int204.webmvc
        - controllers
        - models
        - repositories
        - WebmvcApplication
    - resources
      - static
      - templates
      - application-default.properties
      - application-dev.properties
  - test
- target
- .gitignore
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml
- webmvc.iml

External Libraries

Scratches and Consoles

ProductsRestController.java

```
1  package sit.int204.webmvc.controllers;
2
3  import org.springframework.web.bind.annotation.*;
4  import sit.int204.webmvc.models.Products;
5  import sit.int204.webmvc.repositories.ProductsRep
6
7  import java.util.List;
8
9  @RestController
10 public class ProductsRestController {
11     private final ProductsRepository
12
13     public ProductsRestController(Pro
14         this.productsRepository = pr
15     }
16
17     @GetMapping("/products")
18     List<Products> all() {return produc
19
20     @GetMapping("/productsSearch")
21     public  List<Products> productList(@RequestPa
22         return productsRepository.findAllByProduc
23     }
24
25     @GetMapping("/show/{code}")
26     public Products show(@PathVariable String cod
29     @DeleteMapping("/remove/{code}")
```

Problems Terminal Profiler Endpoints Build TODO Java Enterprise Spring

What is Spring Data JPA ?

- Spring Data JPA is not a JPA provider. It is a library / framework that adds an extra layer of abstraction on the top of our JPA provider. If we decide to use Spring Data JPA, the repository layer of our application contains three layers that are described in the following:
  - Spring Data JPA provides support for creating JPA repositories by extending the Spring Data repository interfaces.
  - Spring Data Commons provides the infrastructure that is shared by the datastore specific Spring Data projects.
  - The JPA Provider implements the Java Persistence API.

CRUD using Spring JPA

- 1) Create Java Model or Entity Class

```
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String prodName;
    private String prodDesc;
    private String prodImage;
    private Double prodPrice;
    public Product() { }

    public Product(String prodName, String prodDesc, String prodImage, Double prodPrice) {
        this.prodName = prodName;
        this.prodDesc = prodDesc;
        this.prodImage = prodImage;
        this.prodPrice = prodPrice;
    }
```

2) Creating a repository

- Create an interface that extends the JpaRepository interface and add the required methods to the created interface.
- we have to provide two type parameters:
  - The type of the entity that is managed by our repository.
  - The type of the entity's id field.

```
public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

3) Create Spring MVC Controller

```
@Controller
public class ProductController {

    @Autowired
    ProductRepository productRepository;

    @RequestMapping("/product")
    public String product(Model model) {
        model.addAttribute("products", productRepository.findAll());
        return "product";
    }
```

4) Create Spring MVC Views (Master)

Default.html

```
<!DOCTYPE html>
<html lang="en"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
  <head>
    <meta charset="UTF-8"/>
    <title>Default title for my pages</title>
    <link rel="stylesheet" href="/webjars/bootstrap/3.3.7/css/bootstrap.min.css"/>
    <link rel="stylesheet" href="/webjars/bootstrap/3.3.7/css/bootstrap-
theme.min.css"/>
    <link rel="stylesheet" href="/css/style.css" />
  </head>
```

```
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
      <div class="container">
        <div class="navbar-header">
                |
                |
        <div id="navbar" class="collapse navbar-collapse">
          <ul class="nav navbar-nav">
            <li class="active"><a href="/product">Home</a></li>
          </ul>
        </div><!--/.nav-collapse -->
      </div>
    </nav>
    <div class="container">
        <div class="starter-template" layout:fragment="content"></div>
    </div><!-- /.container -->
    <script src="/webjars/jquery/1.11.1/jquery.min.js"></script>
    <script src="/webjars/bootstrap/3.3.7/js/bootstrap.min.js"></script>
</body>
```

5) Create Spring MVC Views page

```
!DOCTYPE HTML>
<html lang="en"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org"

xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
    layout:decorate="default">
  <head>
    <title>Show Product</title>
  </head>
  <body>
    <div layout:fragment="content" class="row">
      <div class="col-xs-8 col-md-8">
```

show.html

```
<h3>
    <a href="/product" class="btn btn-primary"><span class="glyphicon glyphicon-list"></span> Product</a>
 </h3>
    <h2 th:text="${product.prodName}"></h2>
    <h2><img th:src="${product.prodImage}" width="200" /></h2>
            <dl class="list">
              <dt>Product Description</dt>
              <dd th:text="${product.prodDesc}"></dd>
              <dt>Product Description</dt>
              <dd th:text="${product.prodPrice}"></dd>
            </dl>
```

Application Properties Injection

- http://localhost:8080/h2-console

- resource/application.properties

```
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
```