

1
2
3
4
5

PAW - Programowanie Aplikacji Webowych 06

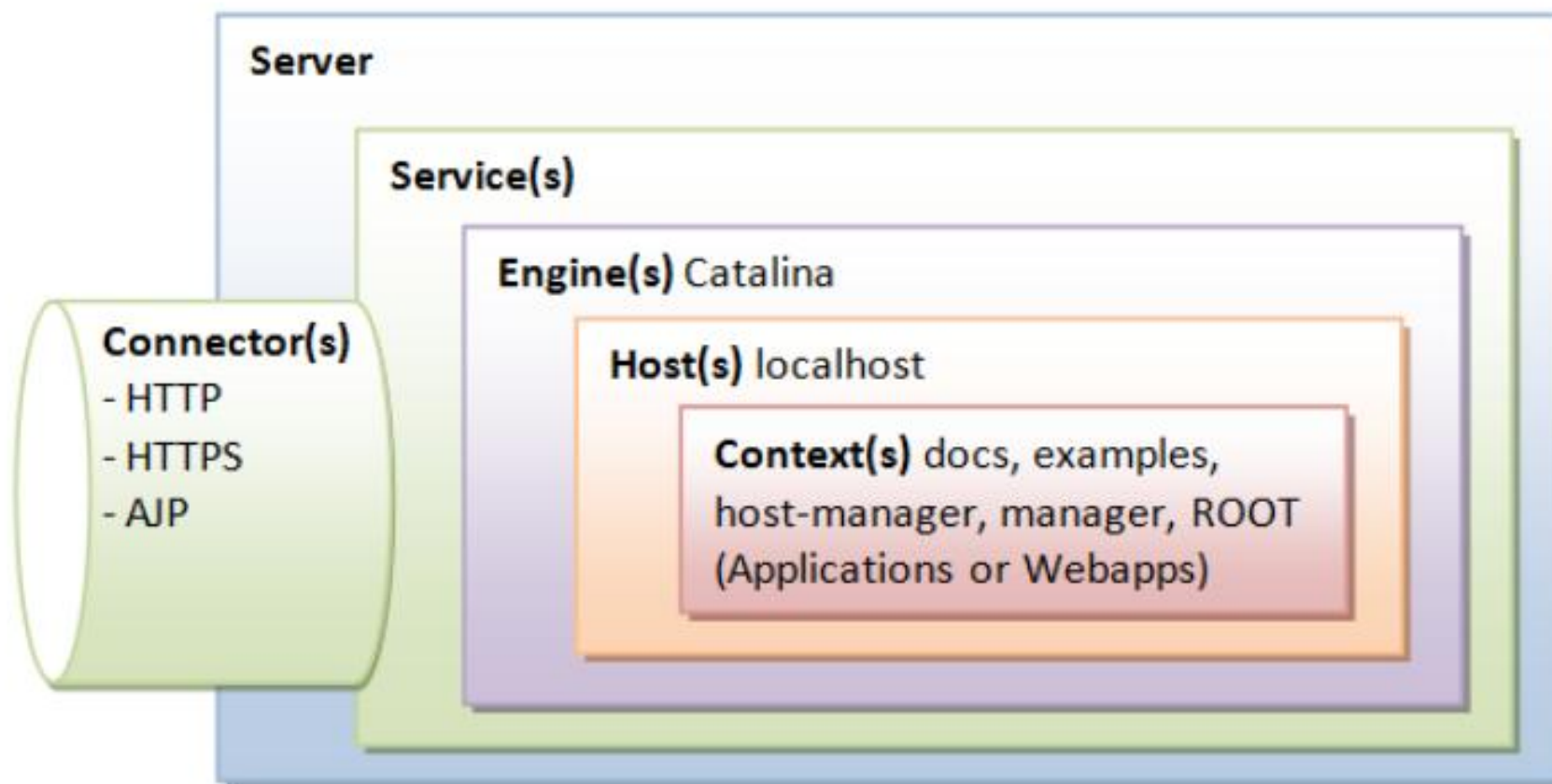
SpringBooks: SpringBoot + PostgreSQL

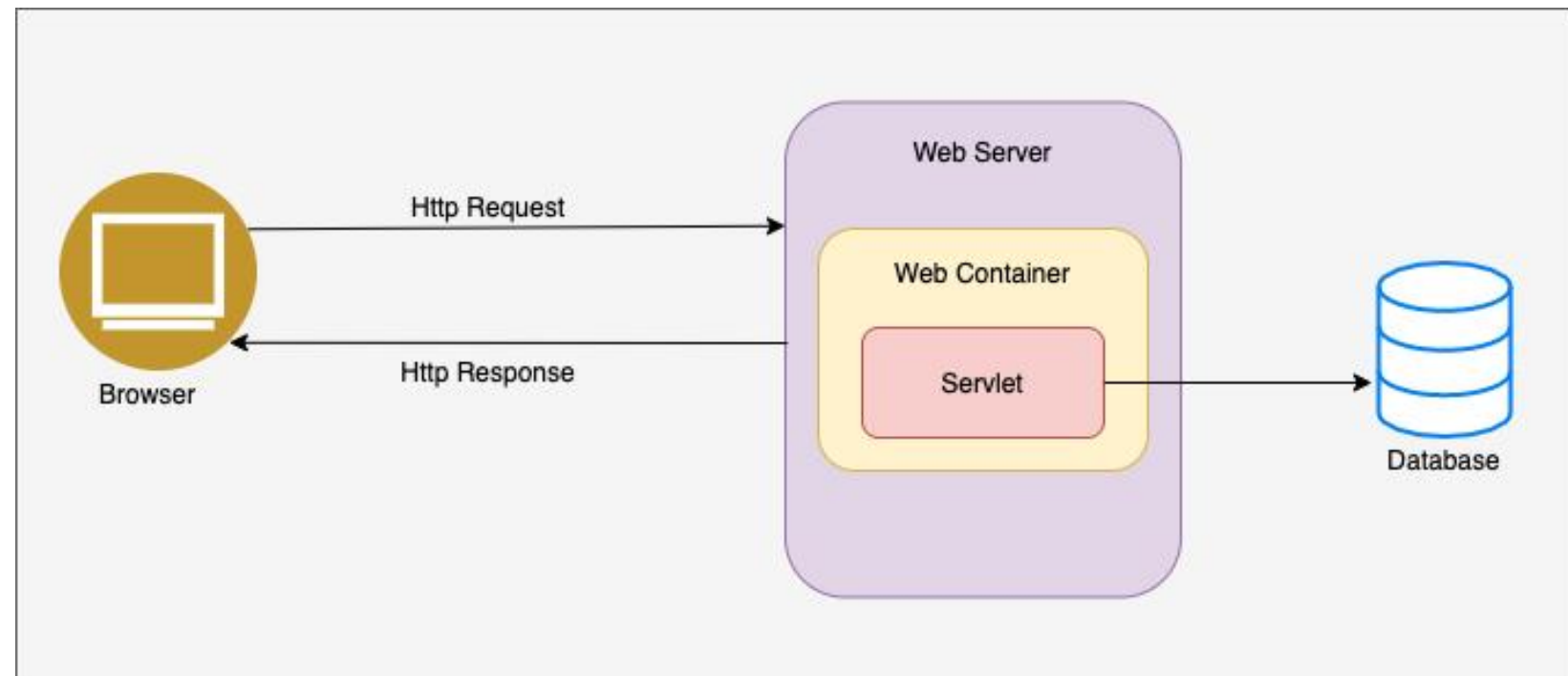
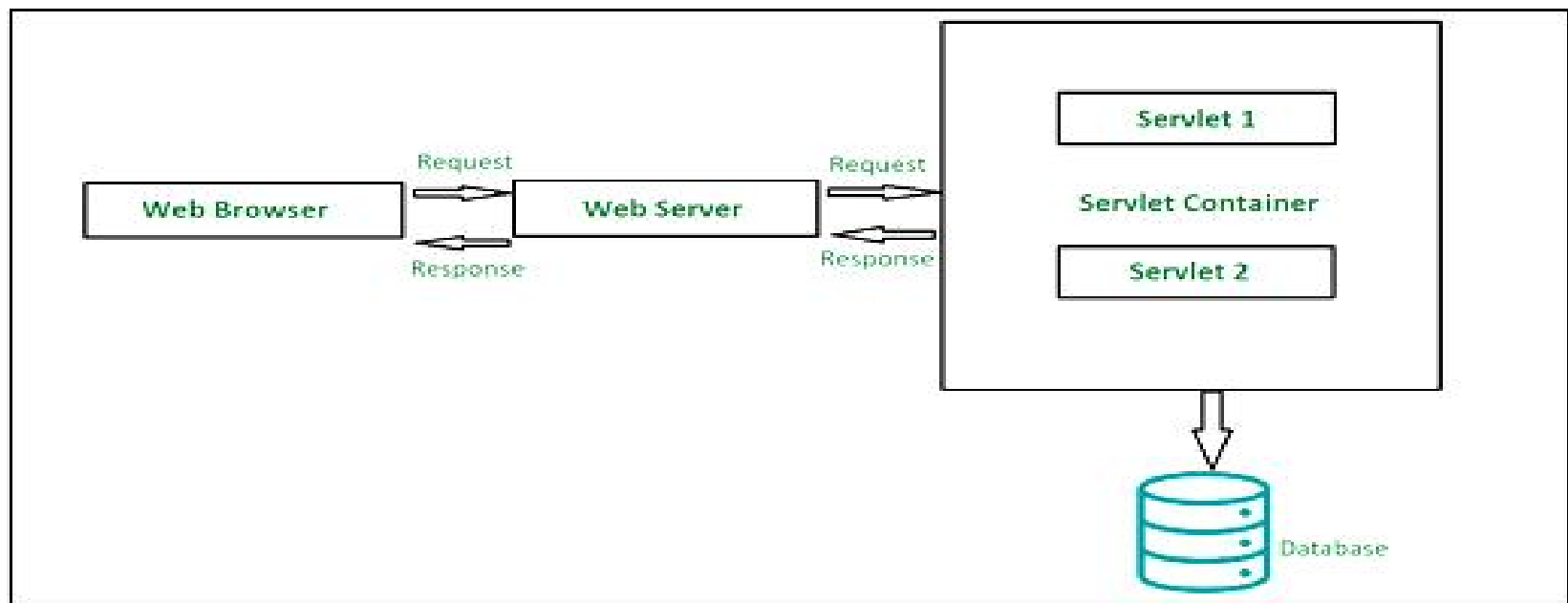
Inż. Juliusz Łosiński

Hello, World!



Architektura Apache Tomcat





```

<?xml version='1.0' encoding='utf-8'?>
<Server port="8005" shutdown="SHUTDOWN">
  <Listener className="org.apache.catalina.core.JasperListener" />
  <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on" />
  <Listener className="org.apache.catalina.core.JreMemoryLeakPreventionListener" />
  <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener" />
  <Listener className="org.apache.catalina.core.ThreadLocalLeakPreventionListener" />

  <GlobalNamingResources>
    <Resource name="UserDatabase" auth="Container"
      type="org.apache.catalina.UserDatabase"
      description="User database that can be updated and saved"
      factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
      pathname="conf/tomcat-users.xml" />
  </GlobalNamingResources>

  <Service name="Catalina">
    <Connector port="8080" protocol="HTTP/1.1"
      connectionTimeout="20000"
      redirectPort="8443" />
    <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />

    <Engine name="Catalina" defaultHost="localhost">

      <Realm className="org.apache.catalina.realm.LockOutRealm">
        <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
          resourceName="UserDatabase"/>
      </Realm>

      <Host name="localhost" appBase="webapps"
        unpackWARs="true" autoDeploy="true">
        <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
          prefix="localhost_access_log." suffix=".txt"
          pattern="%h %l %u %t &quot;%r&quot; %s %b" />
        </Host>
      </Engine>
    </Service>
  </Server>

```

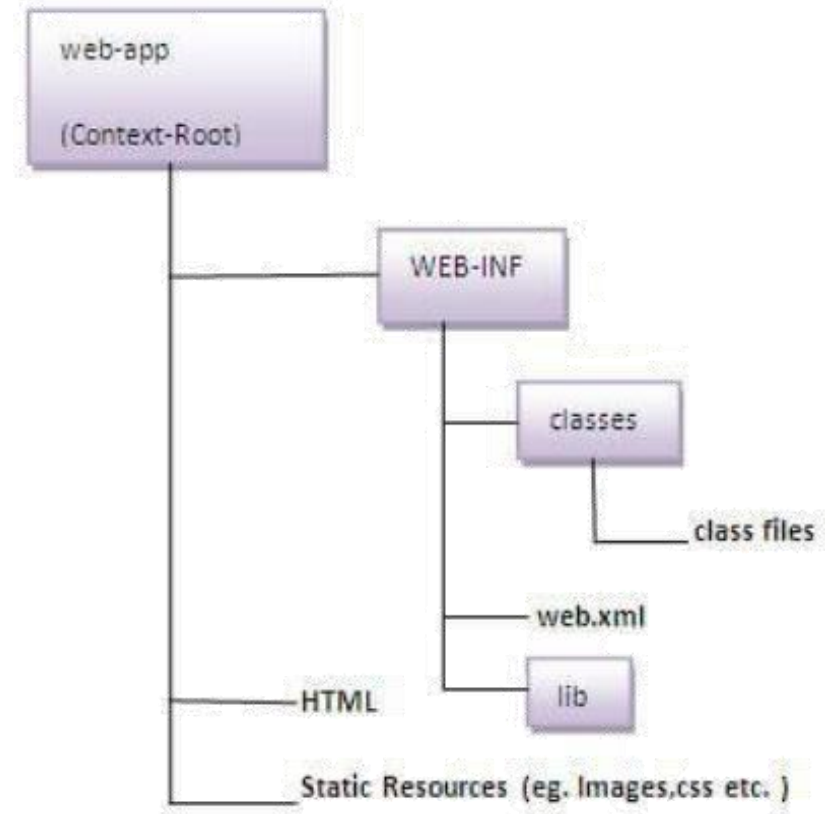
Plik konfiguracyjny *server.xml*

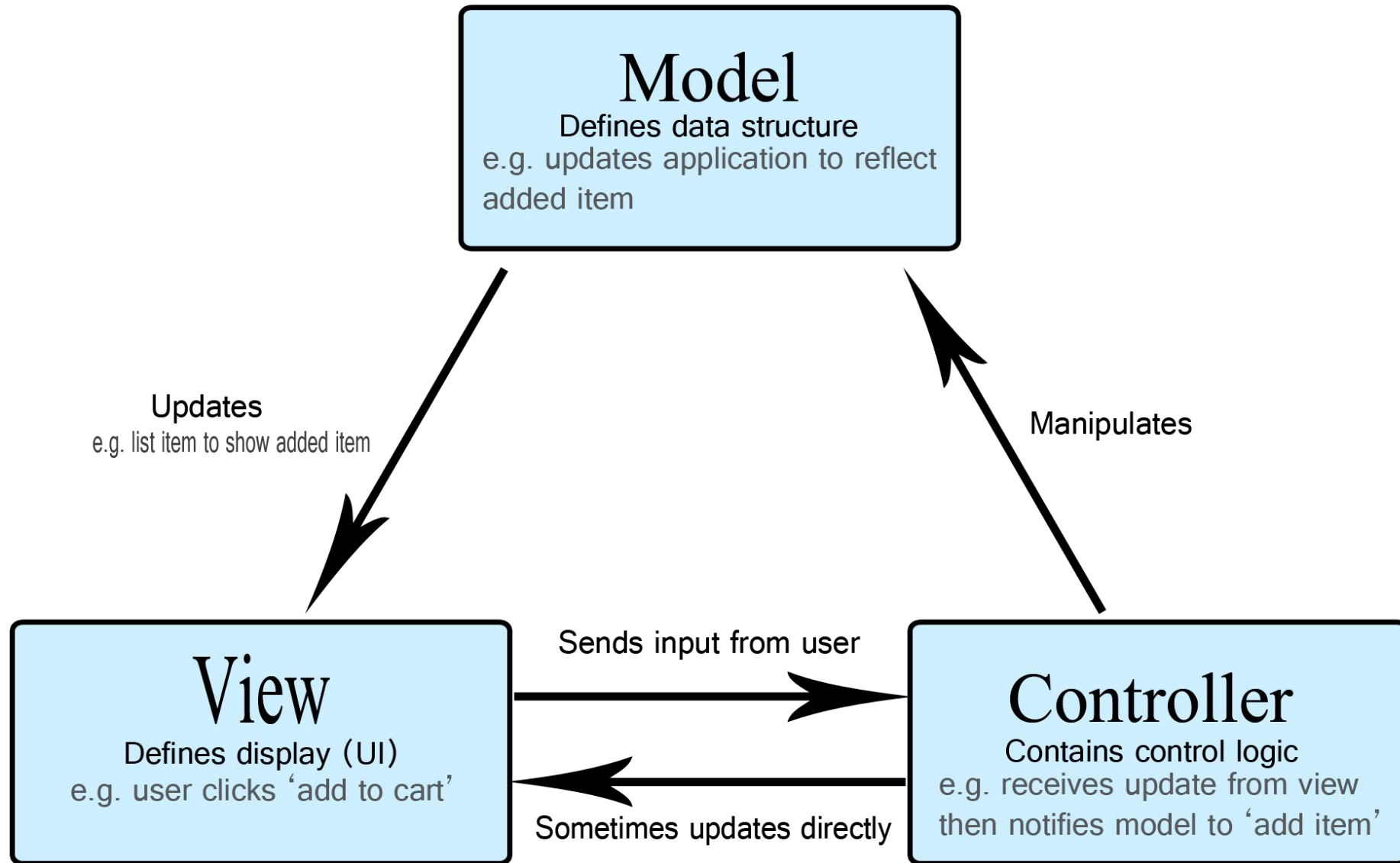
» Ten komputer » DATA (D:) » DevEnv » xampp » tomcat » conf

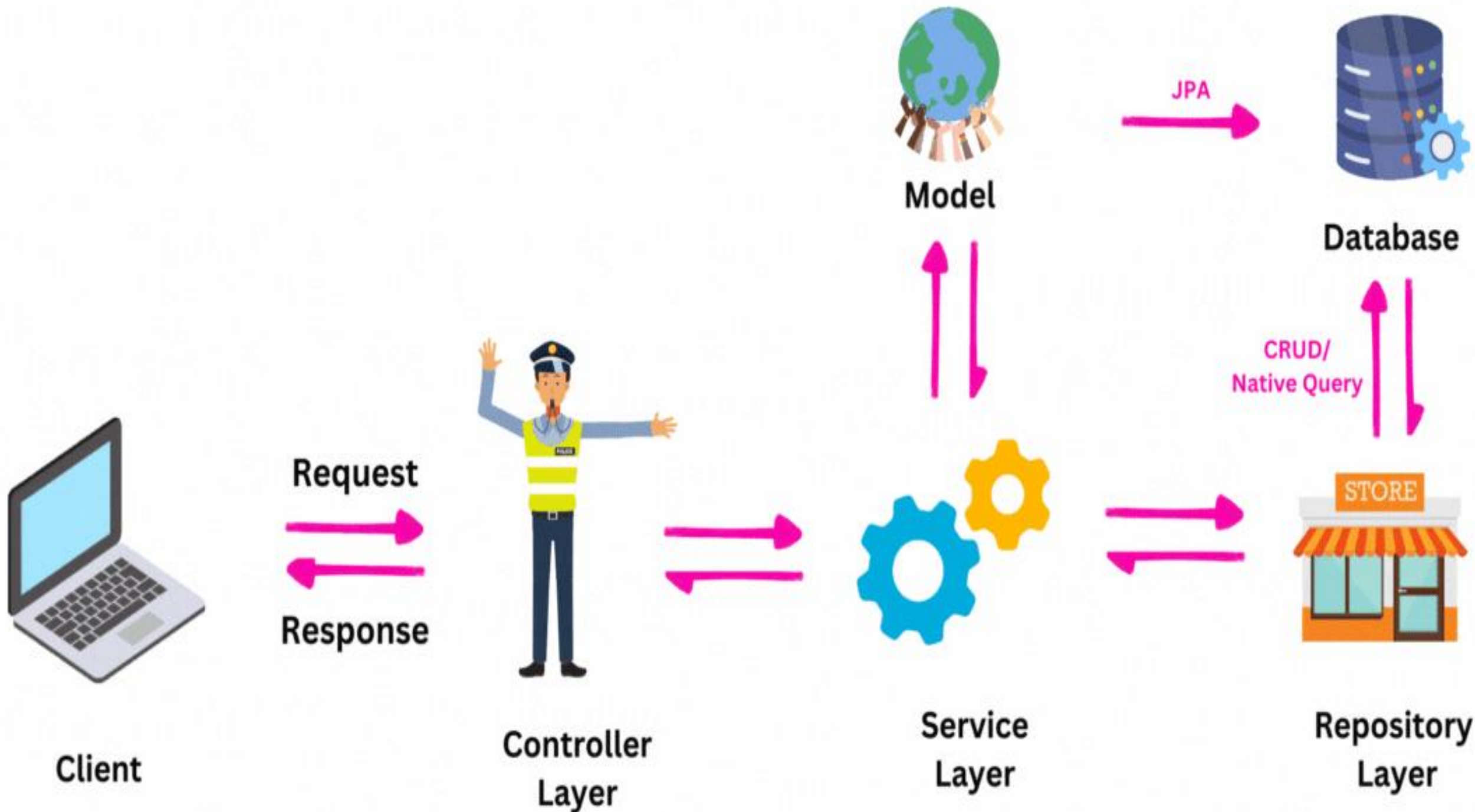
Nazwa	Data modyfikacji	Typ	Rozmiar
Catalina	29.09.2023 14:06	Folder plików	
catalina.policy	25.09.2023 22:53	Plik POLICY	13 KB
catalina	25.09.2023 22:53	Properties Source ...	8 KB
context	25.09.2023 22:53	Dokument XML	2 KB
jaspic-providers	25.09.2023 22:53	Dokument XML	2 KB
jaspic-providers.xsd	25.09.2023 22:53	XML Schema File	3 KB
logging	25.09.2023 22:53	Properties Source ...	5 KB
server	25.09.2023 22:53	Dokument XML	7 KB
tomcat-users	25.09.2023 22:53	Dokument XML	3 KB
tomcat-users.xsd	25.09.2023 22:53	XML Schema File	3 KB
web	25.09.2023 22:53	Dokument XML	173 KB

Proces tworzenia projektu (*Web Dynamic Project*)

1. **Utworzenie** struktury projektu (wygenerowanie przy użyciu odpowiedniego narzędzia np.: Eclipse).
2. **Utworzenie** Servlet'u ~ **.class**
3. **Kompilacja** Servlet'u ~ **.bytecode**
4. **Utworzenie** deskryptora wdrażania (*deployment descriptor*) ~ **Web.xml**
5. **Wdrożenie** rozwiązania na serwer, a następnie uruchomienie ~ **użycie pliku Server.xml**
6. **Wywołanie** punktu końcowego (*End-Point*) skojarzonego ze Servlet'em.









what is
spring
boot?

Spring Projects

Spring Cloud

Spring Boot

Spring
LDAP

Spring
Web
Services

Spring
Session

Spring
Integration

More ...

Spring
Data

Spring
Batch

Spring
Security

Spring
Social

Spring
Kafka

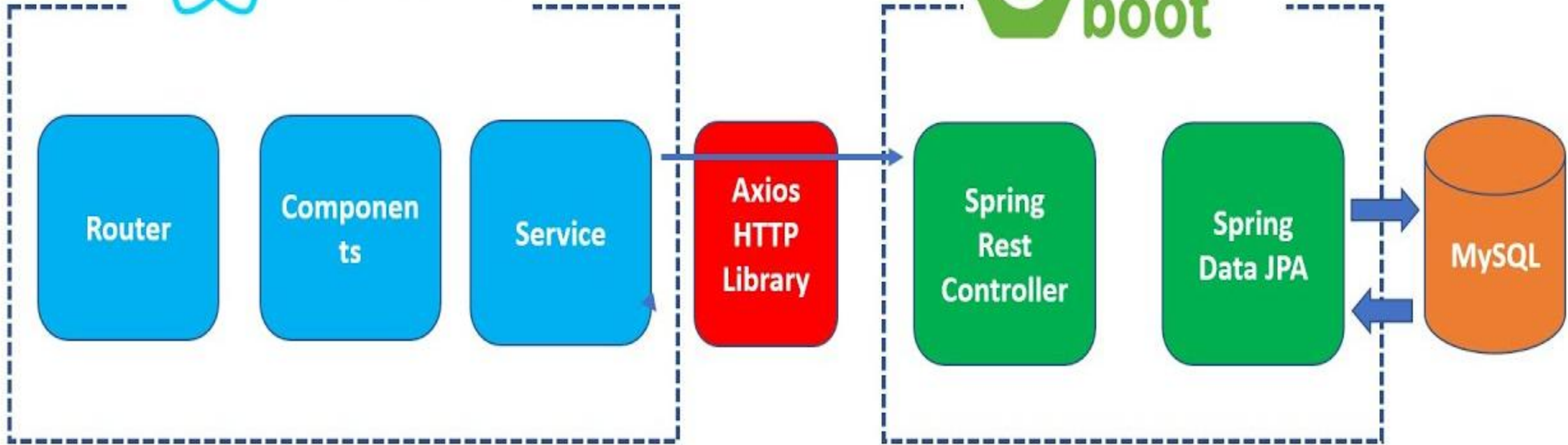
Web

Data

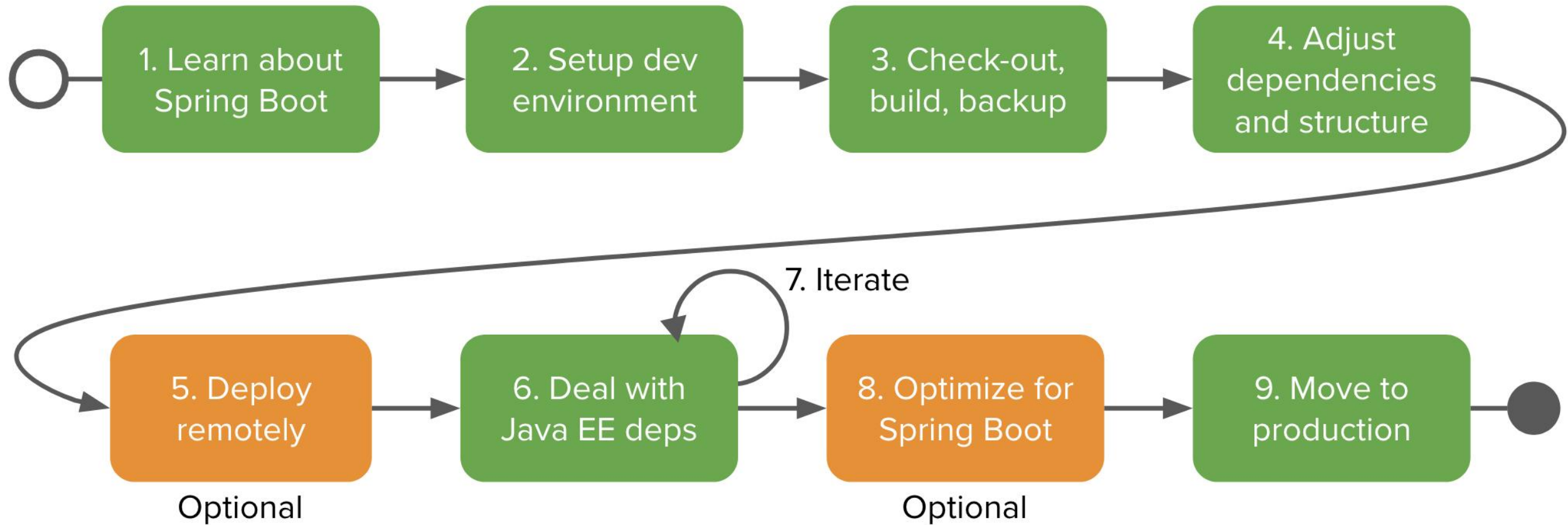
Spring Framework

AOP

Core



ReactJS + Spring Boot CRUD Full Stack App





Main Class	@SpringBootApplication	Spring Boot auto configuration
REST Endpoint	@RestController	Class with REST endpoints
	@RequestMapping	REST endpoint method
	@PathVariable	URI path parameter
	@RequestBody	HTTP request body
Periodic Tasks	@Scheduled	Method to run periodically
	@EnableScheduling	Enable Spring's task scheduling
Beans	@Configuration	A class containing Spring beans
	@Bean	Objects to be used by Spring IoC for dependency injection
Spring Managed Components	@Component	A candidate for dependency injection
	@Service	Like @Component
	@Repository	Like @Component, for data base access
Persistence	@Entity	A class which can be stored in the data base via ORM
	@Id	Primary key
	@GeneratedValue	Generation strategy of primary key
	@EnableJpaRepositories	Triggers the search for classes with @Repository annotation
	@EnableTransactionManagement	Enable Spring's DB transaction management through @Beans objects
Miscellaneous	@Autowired	Force dependency injection
	@ConfigurationProperties	Import settings from properties file
Testing	@SpringBootTest	Spring integration test
	@AutoConfigureMockMvc	Configure MockMvc object to test HTTP queries

Spring Boot and Web annotations

Use annotations to configure your web application.

T **@SpringBootApplication** - uses @Configuration, @EnableAutoConfiguration and @ComponentScan.

T **@EnableAutoConfiguration** - make Spring guess the configuration based on the classpath.

T **@Controller** - marks the class as web controller, capable of handling the requests. **T** **@RestController** - a convenience annotation of a @Controller and @ResponseBody.

M **T** **@ResponseBody** - makes Spring bind method's return value to the web response body.

M **T** **@RequestMapping** - specify on the method in the controller, to map a HTTP request to the URL to this method.

P **@RequestParam** - bind HTTP parameters into method arguments.

P **@PathVariable** - binds placeholder from the URI to the method parameter.

Spring Cloud annotations

Make your application work well in the cloud.

T **@EnableConfigServer** - turns your application into a server other apps can get their configuration from.

Use `spring.application.cloud.config.uri` in the client @SpringBootApplication to point to the config server.

T **@EnableEurekaServer** - makes your app an Eureka discovery service, other apps can locate services through it.

T **@EnableDiscoveryClient** - makes your app register in the service discovery server and discover other services through it.

T **@EnableCircuitBreaker** - configures Hystrix circuit breaker protocols.

M **@HystrixCommand(fallbackMethod = "fallbackMethodName")** - marks methods to fall back to another method if they cannot succeed normally.

Spring Framework annotations

Spring uses dependency injection to configure and bind your application together.

T **@ComponentScan** - make Spring scan the package for the @Configuration classes.

T **@Configuration** - mark a class as a source of bean definitions.

M **@Bean** - indicates that a method produces a bean to be managed by the Spring container.

T **@Component** - turns the class into a Spring bean at the auto-scan time. **T** **@Service** - specialization of the @Component, has no encapsulated state.

C **F** **M** **@Autowired** - Spring's dependency injection wires an appropriate bean into the marked class member.

T **M** **@Lazy** - makes @Bean or @Component be initialized on demand rather than eagerly.

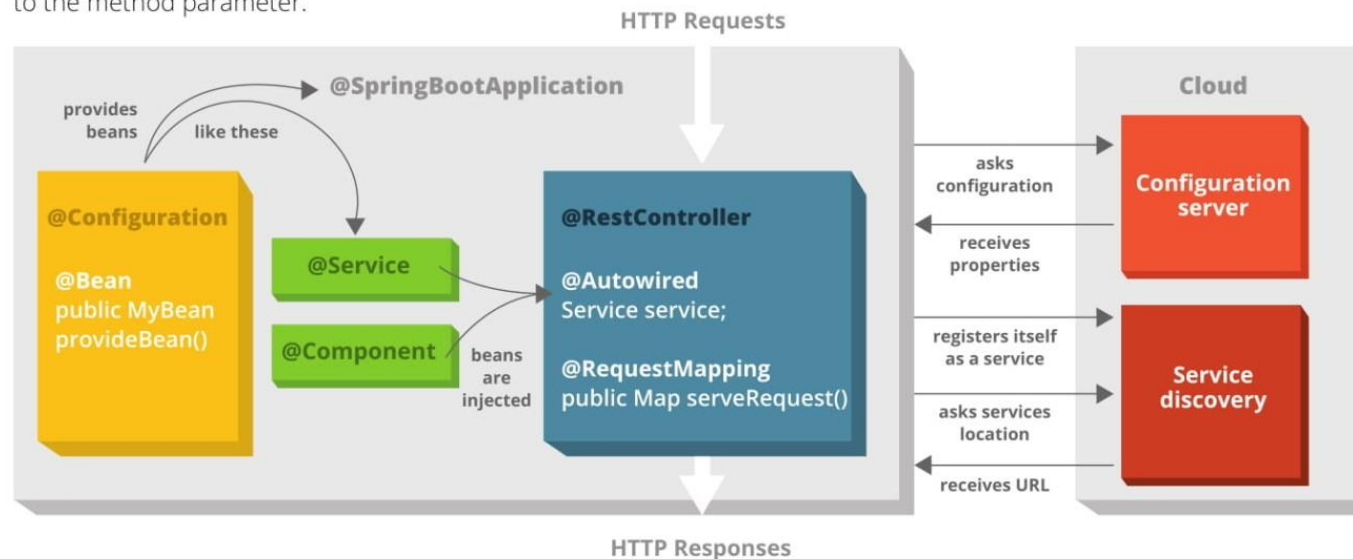
C **F** **M** **@Qualifier** - filters what beans should be used to @Autowire a field or parameter.

C **F** **M** **@Value** - indicates a default value expression for the field or parameter, typically something like `"#{systemProperties.myProp}"`

C **F** **M** **@Required** - fail the configuration, if the dependency cannot be injected.

Legend

T - class
F - field annotation
C - constructor annotation
M - method
P - parameter



Annotations Used In Spring/Spring Boot MVC

@Controller

@GetMapping

@PostMapping

@ModelAttribute

@RequestParam

@CrossOrigin



@RequestMapping



Annotations Used In Spring/Spring Boot REST

@RestController

@GetMapping

@PostMapping

@PatchMapping

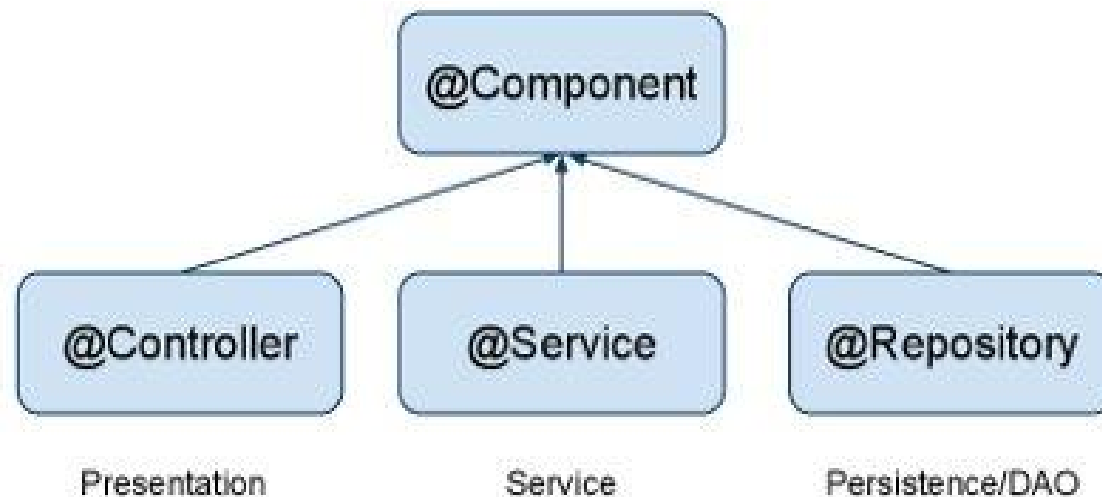
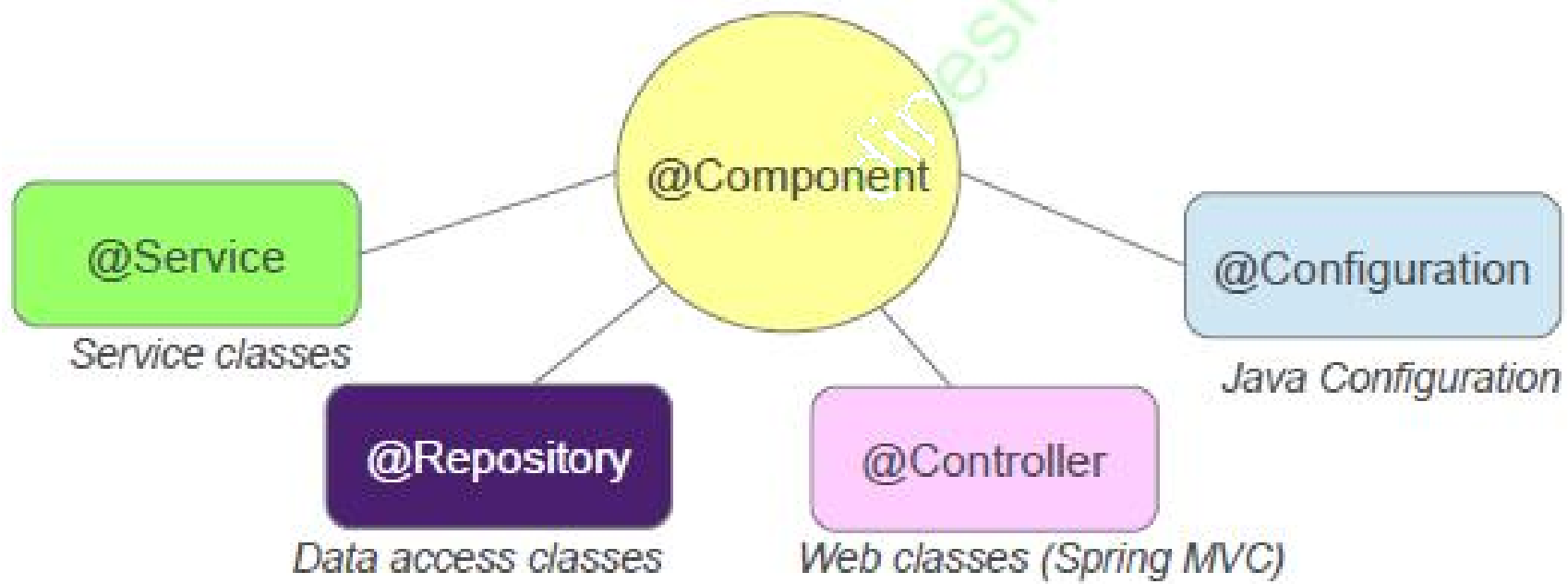
@PutMapping

@DeleteMapping

@PathVariable

@RequestBody

@ResponseBody



Tworzenie aplikacji - Spring Books

Eclipse Marketplace

Eclipse Marketplace

One solution selected for install

Search

Recent

Popular

Favorites

Installed

Research at the Eclipse


Find:

spring

All Markets

All Categories

Go



Spring Tools 4 (aka Spring Tool Suite 4) 4.20.1.RELEASE

Spring Tools 4 is the next generation of Spring Boot tooling for your favorite coding environment. Largely rebuilt from scratch, it provides world-class support... [more info](#)

by [VMware](#), [EPL](#)

[spring](#) [Spring IDE](#) [Cloud](#) [Spring Tool Suite](#) [STS](#)


★ 4053

Installs: 2,72M (24 210 last month)

Install Pending

One solution selected | [Deselect all](#)

Marketplaces



?

< Back

Install Now >

Finish

Cancel

Eclipse Marketplace

Confirm Selected Features

Press Confirm to continue with the installation. Or go back to choose more solutions to install.

☒ Spring Tools 4 (aka Spring Tool Suite 4) 4.20.1.RELEASE <https://download.springsource.com>

☒ Spring Boot Language Server Feature (required)

☒ Spring Tool Suite 4 Main Feature (required)

☐ Cloud Foundry Manifest Language Server Feature

☐ Concourse Pipeline Language Server Feature

☒ Spring IDE Boot Microservices Dash


?

< Install More


Confirm >

Finish

Cancel



New Spring Starter Project



Service URL

https://start.spring.io

Name

SpringBooks

☒ Use default location

Location

F:\workAllIM2223\SpringBooks

Browse

Type:

Gradle - Groovy

Packaging:

Jar

Java Version:

17

Language:

Java

Group

ksi

Artifact

SpringBooks

Version

0.0.1-SNAPSHOT

Description

Spring Books

Package

ksi.springbooks


Working sets

☐ Add project to working sets

New...

Working sets:

Select...




< Back


Next >

Finish

Cancel



New Spring Starter Project Dependencies



Spring Boot Version:

3.0.0

Available:

Type to search dependencies

Selected:

☒ Spring Data JPA

☒ PostgreSQL Driver

☒ Thymeleaf

☒ Spring Web

Spring Cloud Tools

Template Engines

☒ Thymeleaf

☐ Apache Freemarker

☐ Mustache

☐ Groovy Templates

Testing

VMware Tanzu Application Service

Web

☒ Spring Web

☐ Spring Reactive Web

☐ Spring for GraphQL

☐ Rest Repositories

☐ Spring Session

☐ Rest Repositories HAL Explorer

☐ Spring HATEOAS


☐ Spring Web Services

☐ Jersey

☐ Vaadin

Make Default

Clear Selection



< Back

Next >

Finish

Cancel



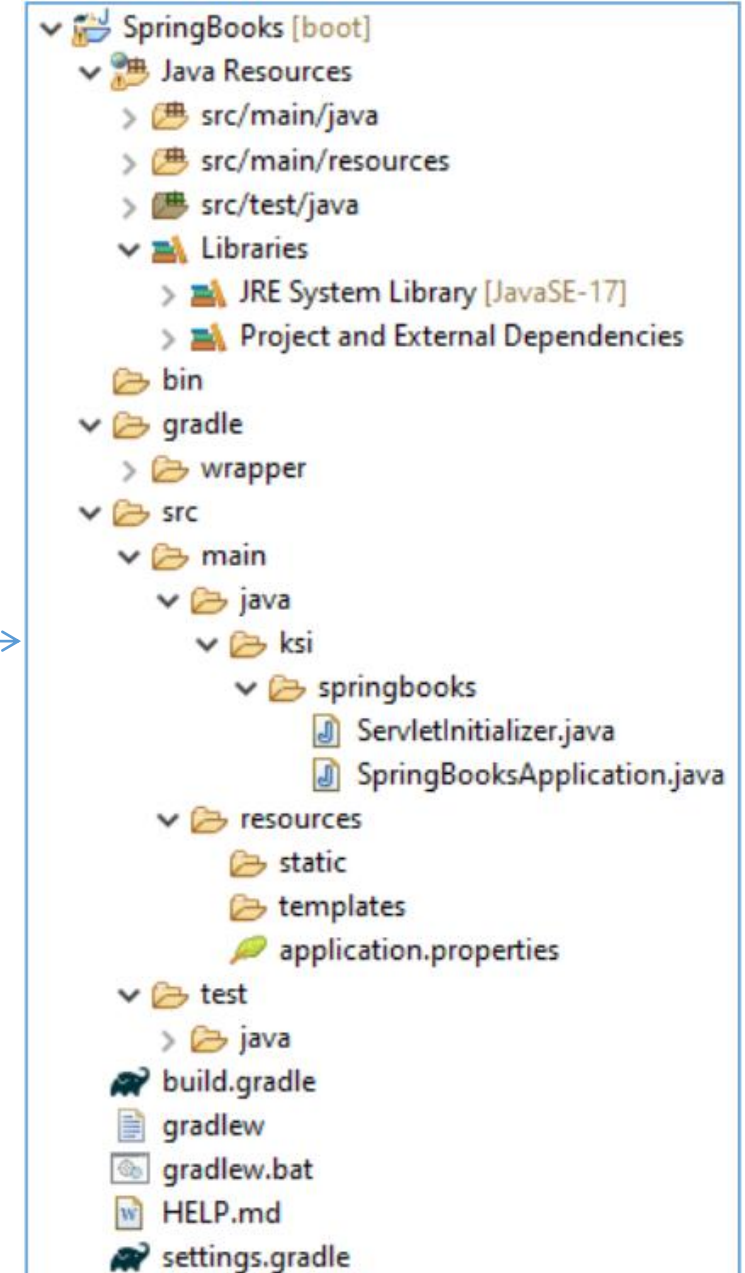
New Spring Starter Project

Site Info

Base Url:

Full Url:

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)





Project

- ☒ Gradle - Groovy
- ☐ Gradle - Kotlin
- ☐ Maven

Language

- ☒ Java
- ☐ Kotlin
- ☐ Groovy

Spring Boot

- ☐ 3.2.0 (SNAPSHOT)
- ☐ 3.2.0 (RC2)
- ☐ 3.1.6 (SNAPSHOT)
- ☒ 3.1.5
- ☐ 3.0.13 (SNAPSHOT)
- ☐ 3.0.12
- ☐ 2.7.18 (SNAPSHOT)
- ☐ 2.7.17

Project Metadata

Group com.example

Artifact demo

Name demo

Description Demo project for Spring Boot

Package name com.example.demo

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Web

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



GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...



```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    runtimeOnly 'org.postgresql:postgresql'  
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
}
```

Przeprowadź proste sprawdzenie możliwości uruchomienia aplikacji generując plik **index.html** z tytułem i nagłówkiem: Spring Books Application. W tym celu wygeneruj statyczny plik HTML 5 o nazwie **index.html** w folderze **src/main/resources/static**. Ustaw tytuł i nagłówek h1 z napisem Spring Books Application.

Główne pakiety:



- ***spring-boot-starter-web*** - dostarcza automatyczną deklarację, mapowanie oraz konfigurację servlet'u dyspozytora,
- ***spring-boot-starter-data-jpa*** - obsługują relacyjne bazy danych przy użyciu interfejsów Spring Data JPA oraz systemu **ORM** Hibernate,
- ***spring-boot-starter-thymeleaf*** - to implementacja silnika szablonów Thymeleaf, który umożliwia tworzenie elementów interfejsu użytkownika, zawiera takzwane dialekty: **Standard** oraz **SpringStandard**.



```
import org.springframework.boot.autoconfigure.*;
import org.springframework.boot.autoconfigure.jdbc.*;
import org.springframework.context.annotation.*;

@Configuration
@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})
public class MyConfiguration {
}
```

Wybierz projekt, a następnie z podręcznego menu Run As | Spring Boot App. Podczas uruchamiania aplikacji, w konsoli Eclipse pojawi się logo ASCII Spring Boot oraz seria komunikatów mówiąca o stanie uruchomienia aplikacji. Eclipse zgłosi błąd ponieważ dołączyliśmy do projektu sterownik bazy danych, natomiast nie mamy jeszcze skonfigurowanego połączenia. Żeby pominąć łączenie z bazą danych w pliku aplikacji dołącz do adnotacji *@SpringBootApplication* parametr wyłączający użycie konfiguracji dla danych. Powinna ona mieć postać:

```
@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
```


Problems Servers Javadoc Declaration Console X Progress

AppSpringBooks - AppSpringBooksApplication [Spring Boot App] [pid: 10576]

```

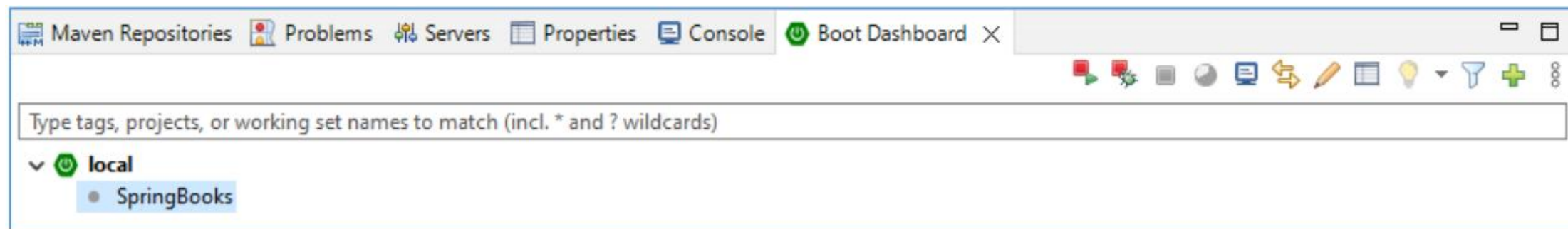
  ____ _
 / ___ \| | | |
/ /___ \| |_| |
\___)___|_____|
:: Spring Boot :: (v2.6.7)

```

```

2022-05-18 21:19:54.439 INFO 10576 --- [main] k.springbooks.AppSpringBooksApplication : Starting AppSpringBooksApplication using Java 17.0.2 on DESKTOP-INE6MJ8 with PID 10576 (F
2022-05-18 21:19:54.442 INFO 10576 --- [main] k.springbooks.AppSpringBooksApplication : No active profile set, falling back to 1 default profile: "default"
2022-05-18 21:19:55.404 INFO 10576 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-05-18 21:19:55.416 INFO 10576 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-05-18 21:19:55.417 INFO 10576 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.62]
2022-05-18 21:19:55.532 INFO 10576 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-05-18 21:19:55.532 INFO 10576 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1033 ms
2022-05-18 21:19:55.766 INFO 10576 --- [main] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding welcome page: ServletContext resource [/index.html]
2022-05-18 21:19:55.834 WARN 10576 --- [main] ion$DefaultTemplateResolverConfiguration : Cannot find template location: classpath:/templates/ (please add some templates or check
2022-05-18 21:19:55.924 INFO 10576 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-05-18 21:19:55.933 INFO 10576 --- [main] k.springbooks.AppSpringBooksApplication : Started AppSpringBooksApplication in 1.891 seconds (JVM running for 2.674)
2022-05-18 21:20:40.698 INFO 10576 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-05-18 21:20:40.698 INFO 10576 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-05-18 21:20:40.699 INFO 10576 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms

```



Konfiguracja pliku `application.properties`

```
# parametry poziomu logowania/raportowania aplikacji
logging.level.org.springframework=ERROR
spring.sql.init.mode=always

# parametry dla PostgreSQL
spring.sql.init.platform=postgres
spring.jpa.properties.hibernate.default_schema=books
spring.datasource.url=jdbc:postgresql://localhost:5432/<nazwa_bd>
spring.datasource.username=<nazwa_u>
spring.datasource.password=<haslo_u>

# parametry dla JPA
spring.jpa.hibernate.ddl-auto=update
```

```
@Entity
public class Book {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Long idb;
    private String title;
}
```

Użycie powyższych adnotacji umożliwia standard JPA ORM, z którego korzysta Spring. Znacznie ułatwia to obsługę bazy danych z poziomu aplikacji.

Struktura aplikacji Web składa się z łańcucha powiązanych obiektów/klas.

Podstawowe elementy takiego łańcucha to kontroler (controller), usługa (service) i interfejs do repozytorium (repository).

Na przykład dla części aplikacji obsługującej książki będzie to:

- **BookController** – klasa z adnotacją @Controller
- **BookService** – klasa z adnotacją (@Service), która zostanie wstrzyknięta do kontrolera
- **BookRepository** – interfejs rozszerzający JpaRepository z adnotacją @Repository, który
- zostanie wstrzyknięty do BookService.

Działanie aplikacji polega na tym, że odpowiednia metoda kontrolera (wybrana na podstawie zdefiniowanego mapowania) przejmuje żądanie. Następnie uruchamia ona metodę serwisu odpowiedzialną za przetworzenie danych. Ta metoda wywołuje następnie odpowiednią metodę repozytorium pobierającą, bądź zapisującą dane w bazie



C:\> Users \PanCh \Desktop \Asystent - Praca \PAW - 5 Semestr Informatyka - Alim \Lab06 > ldata.sql

```
1  insert into books.publisher(idp,name,address) values (1,'Helion','Gliwice, Polska');
2  insert into books.publisher(idp,name,address) values (2,'PWN','Warszawa, Polska');
3  insert into books.publisher(idp,name,address) values (3,'OREILLY','Boston, USA');
4
5  insert into books.category(idc,description) values (1,'WWW');
6  insert into books.category(idc,description) values (2,'HTML');
7  insert into books.category(idc,description) values (3,'JavaScript');
8  insert into books.category(idc,description) values (4,'Java');
9
10 insert into books.book(title,publisher_idp,category_idc) values ('Java. Podstawy',1,4);
11 insert into books.book(title,publisher_idp,category_idc) values ('Projektownie serwisów WWW. Standardy sieciowe',1,1);
12 insert into books.book(title,publisher_idp,category_idc) values ('Zrozumieć JavaScript',1,3);
13 insert into books.book(title,publisher_idp,category_idc) values ('Head first Java',3,4);
14 insert into books.book(title,publisher_idp,category_idc) values ('HTML5. Komponenty',2,2);
15 insert into books.book(title,publisher_idp,category_idc) values ('Wydajny JavaScript',2,3);
```

```
package ksi.springbooks.services;
```

```
import org.springframework.beans.factory.annotation.Autowired;
```

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

```
import ksi.springbooks.repositories.BookRepository;
```

```
@Service
```

```
public class BookService {
```

```
    @Autowired
```

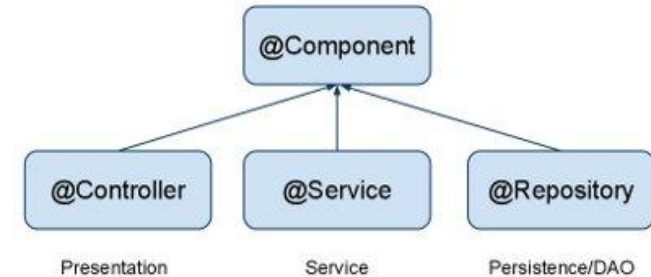
```
    private BookRepository repository;
```

```
    public BookService() {
```

```
        super();
```

```
    }
```

```
}
```



```
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.stereotype.Repository;
```

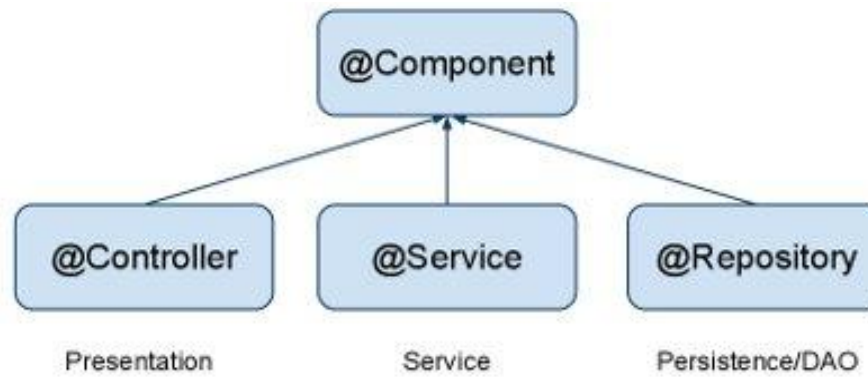
```
import ksi.springbooks.models.Book;
```

```
@Repository
```

```
public interface BookRepository extends JpaRepository<Book, Long> {
```

```
}
```





```
package ksi.springbooks.controllers;

import ksi.springbooks.models.Book;
import ksi.springbooks.services.BookService;

@Controller
public class BookController {

    @Autowired
    private BookService service;

    public BookController() {
    }
}
```



```
@RequestMapping("books_list")
public String viewBooksList(Model model){
    List<Book> lb=service.findAll();
    model.addAttribute("lb", lb);
    return "books_list";
}
```

```
@RequestMapping("/new_book")
public String showFormNewBook(Model model) {
    Book nb = new Book();
    model.addAttribute("book", nb);
    return "new_book";
}
```

```
@PostMapping(value="/save_book")
public String saveBook(@ModelAttribute("book") Book book) {
    service.save(book);
    return "redirect:/books_list";
}
```



Jak widać znaczniki Thymeleaf w szablonie będą się rozpoczynały od th:.

Books List

[Create New Book](#)

Book ID	Title	Publisher	Category		
1	Java. Podstawy	Helion	Java	Edit	Delete
2	Projektownie serwisów WWW. Standardy sieciowe	Helion	WWW	Edit	Delete
3	Zrozumieć JavaScript	Helion	JavaScript	Edit	Delete
4	Head first Java	OREILLY	Java	Edit	Delete
5	HTML5. Komponenty	PWN	HTML	Edit	Delete
6	Wydajny JavaScript	PWN	JavaScript	Edit	Delete





```
<div>
  <h2>Create new book</h2>
  <br />
  <form action="#" th:action="@{/save_book}" th:object="${book}" method="post">
    <table>
      <tr>
        <td>Title</td>
        <td><input type="text" th:field="*{title}" /></td>
      </tr>
      <tr>
        <td>Publisher id</td>
        <td><input type="text" th:field="*{publisher.idp}" /></td>
      </tr>
      <tr>
        <td>Category id</td>
        <td><input type="text" th:field="*{category.idc}" /></td>
      </tr>
    </table>
    <button type="submit">Save</button>
  </form>
</div>
```



Create new book

Title	<input type="text" value="Nowa książka"/>
Publisher id	<input type="text" value="1"/>
Category id	<input type="text" value="1"/>
<input type="button" value="Save"/>	



```
@RequestMapping("/edit_book/{idb}")
public ModelAndView showEditFormBook(@PathVariable(name = "idb") Long idb) {
    ModelAndView mav = new ModelAndView("edit_book");
    Optional<Book> eb = service.findById(idb);
    mav.addObject("book", eb);
    return mav;
}
```

```
<tr>
    <td>Book Id:</td>
    <td><input type="text" th:field="*{idb}" readonly="readonly" /></td>
</tr>
```



Posortuj listę książek według tytułów.

Sortowanie można wykonać na kilka sposobów. Jednym z prostych rozwiązań jest dodanie deklaracji metody o odpowiedniej nazwie w interfejsie BookRepository. Dodaj do interfejsu deklarację metody do sortowania malejąco według id:

```
List<Book> findByIdOrderByIdbDesc();
```

Wywołaj ją w metodzie findAll() klasy BookService, zamiast

```
return repository.findAll();
```

wprowadź:

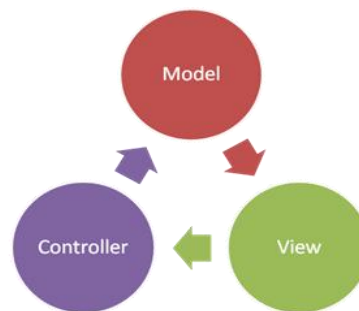
```
return repository.findByIdOrderByIdbDesc();
```

Przetestuj działanie metody uruchamiając ponownie aplikację. Sprawdź, czy książki zostały odpowiednio posortowane.

Posortuj książki według tytułów od A do Z. Zadeklaruj w interfejsie BookRepository dodatkową deklarację metody: List<Book> findByIdOrderByTitleAsc();

Dokonaj odpowiednich zmian w metodzie findAll() klasy BookService i **przetestuj** działanie aplikacji.





Powodzenia!



Źródła

- <https://www.infoworld.com/article/3336161/what-is-jsp-introduction-to-javascript-pages.html>,
- <https://www.javatpoint.com/steps-to-create-a-servlet-using-tomcat-server>,
- <https://spring.io/projects/spring-boot>,
- <https://www.baeldung.com/spring-core-annotations>,

