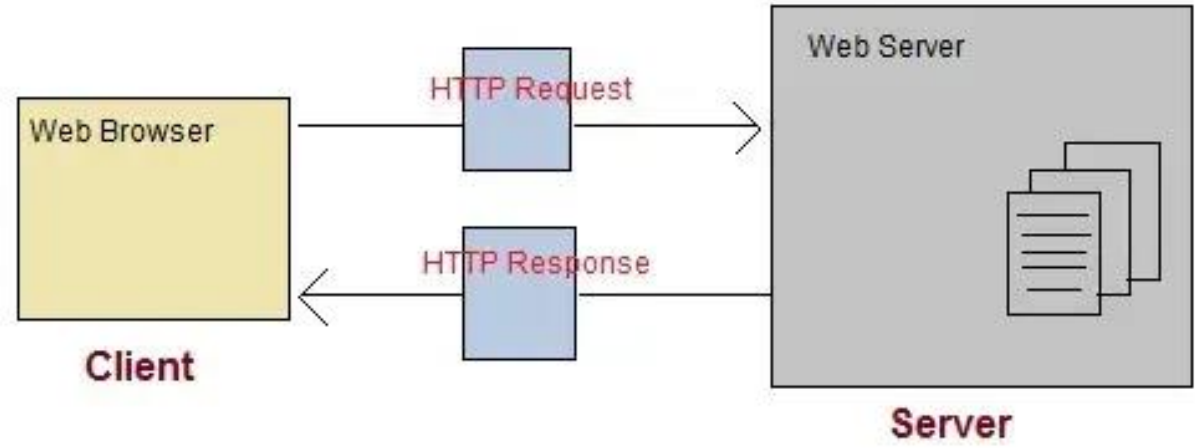
SPRING BOOT

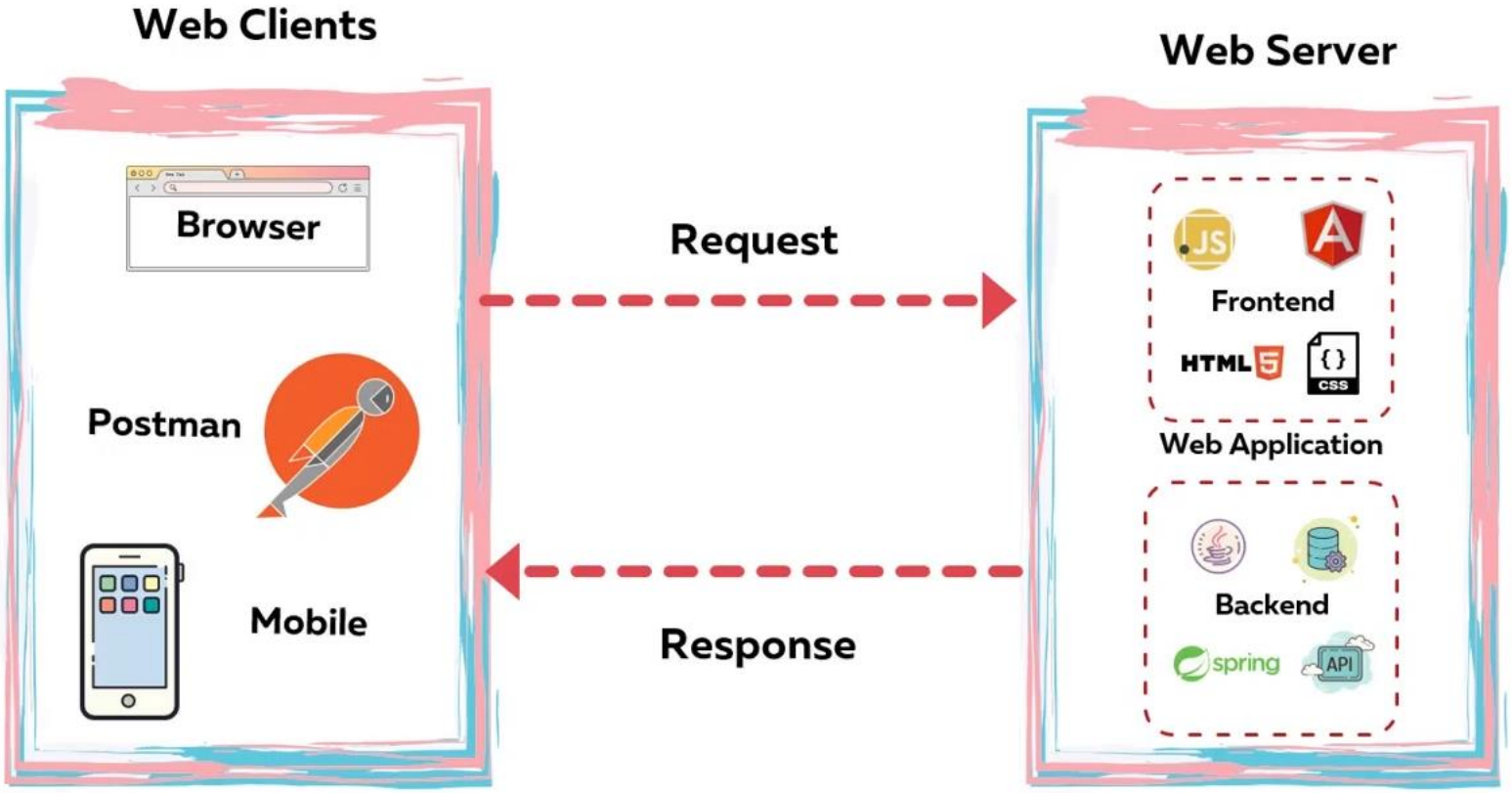
AMOL R PATIL - 9822291613

Web Application



The HTTP request can be made using a variety of methods, but the ones which we use widely are Get and Post. The method name itself tells the server the kind of request that is being made, and how the rest of the message will be formatted.

Web Application



1. The Web clients sends a request using protocols like HTTP to Web Application asking some data like list of images, videos, text etc.

2. The web server where web app is deployed receives the client requests and process the data it receives. Post that it will respond to the client's request in the format of HTML, JSON etc.

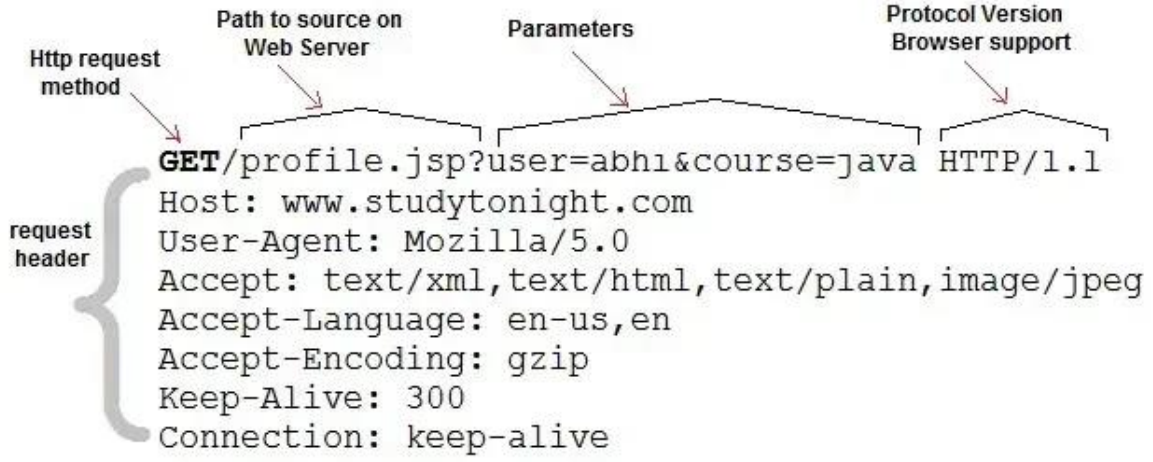
3. In Java web apps, Servlet Container (Web Server) takes care of translating the HTTP messages for Java code to understand. One of the mostly used servlet container is Apache Tomcat.

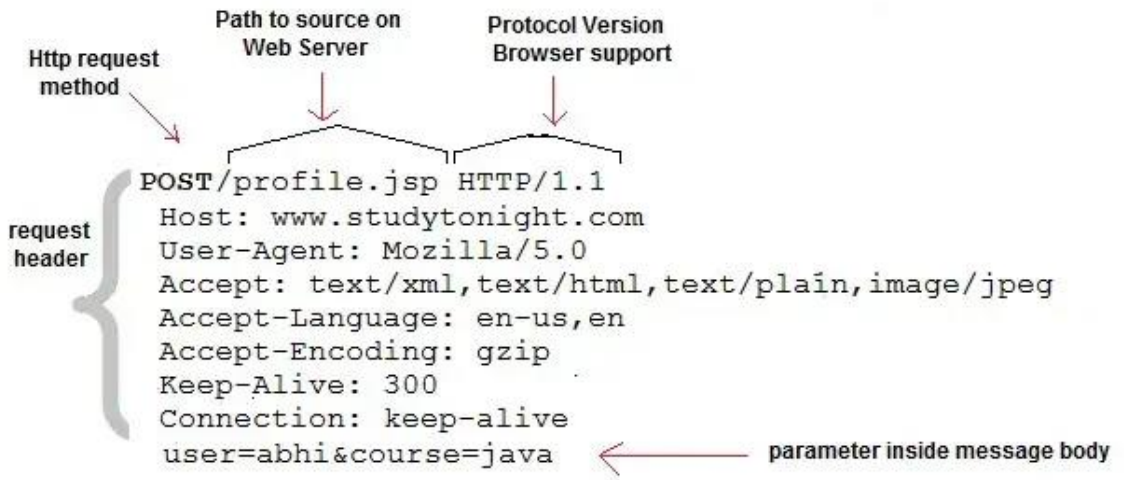
4. Servlet Container converts the HTTP messages into ServletRequest and hand over to Servlet method as a parameter. Similarly, ServletResponse returns as an output to Servlet Container from Servlet.

Usually Web Applications can be,
1) Only Frontend (Static Web Apps)
2) Only Backend (APIs)
3) Frontend + Backend (Ecommerce Apps)

HTTP Methods

Get	Post
1. Data is sent in the header body	1. Data is sent in the request body
2. Restricted to limited data transfer	2. Supports a large amount of data transfer
3. It is not secured	3. It is completely secured
4. It can be bookmarked	4. It cannot be bookmarked

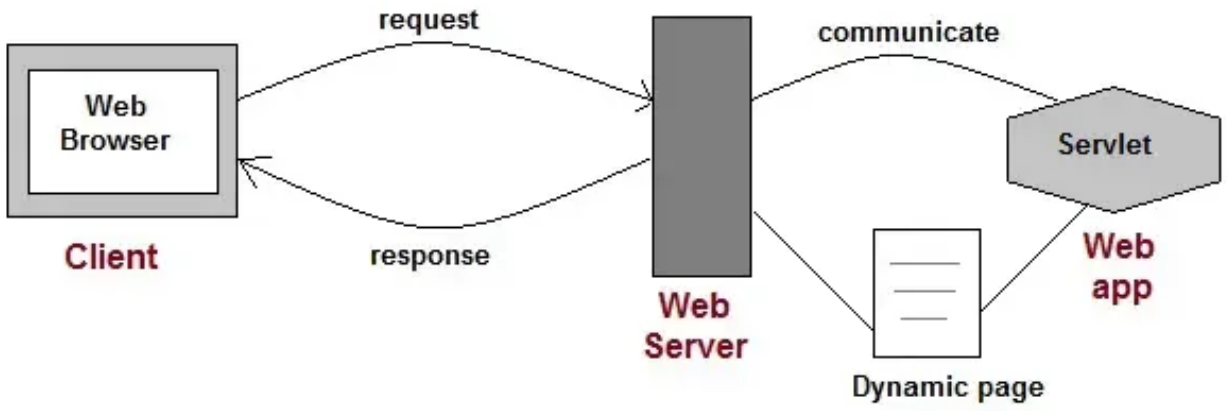




Servlets

A servlet is a Java Programming language class that is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model.

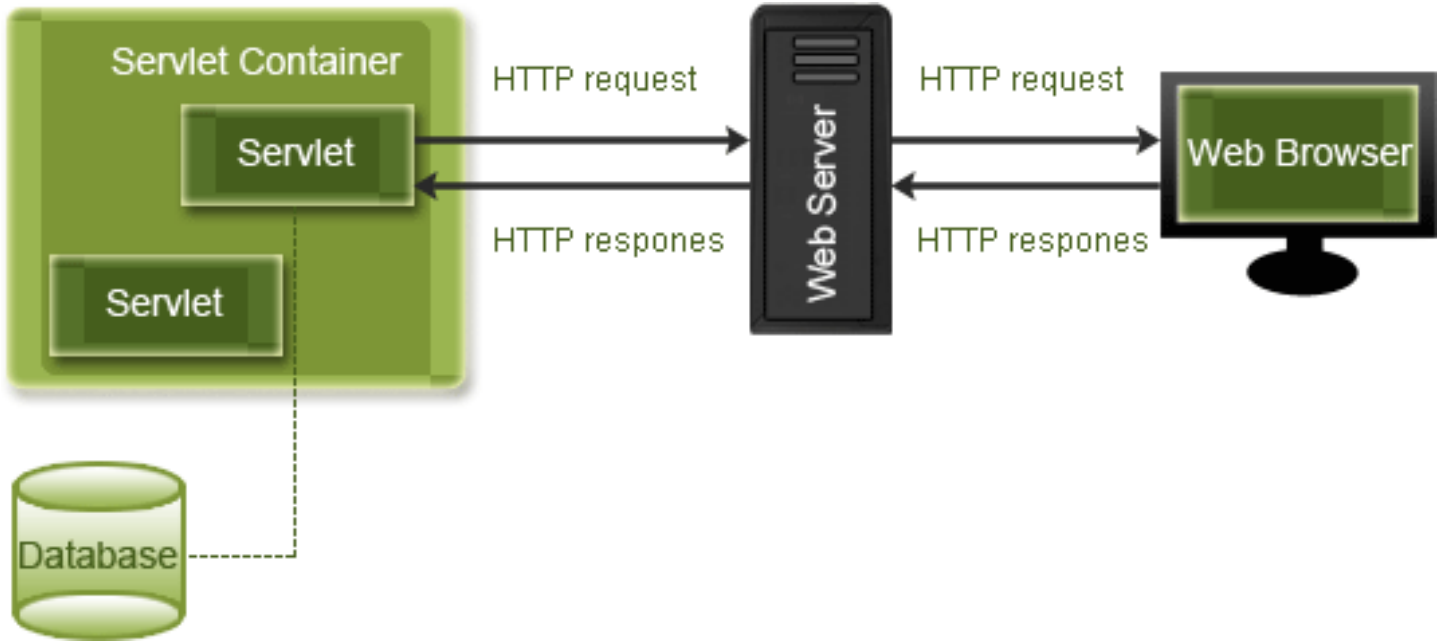
Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. It is also a web component that is deployed on the server to create a dynamic web page



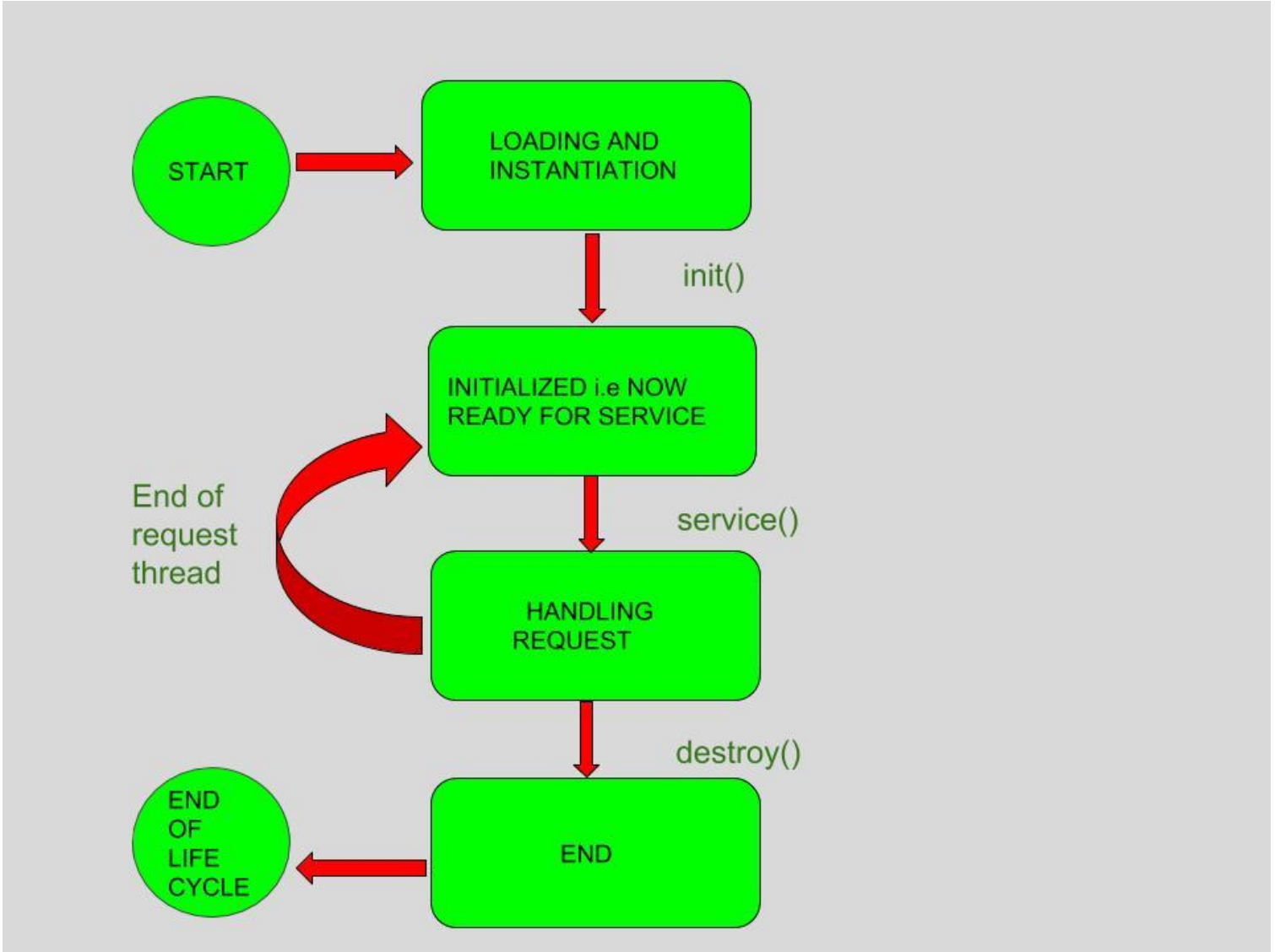
Servlet Container

In Java, Servlet container (also known as a Web container) generates dynamic web pages. So servlet container is the essential part of the web server that interacts with the java servlets. Servlet Container communicates between client Browsers and the servlets.

Servlet Container managing the life cycle of servlet. Servlet container loading the servlets into memory, initializing and invoking servlet methods and to destroy them. There are a lot of Servlet Containers like Jboss, Apache Tomcat, WebLogic etc.



Servlet Life Cycle



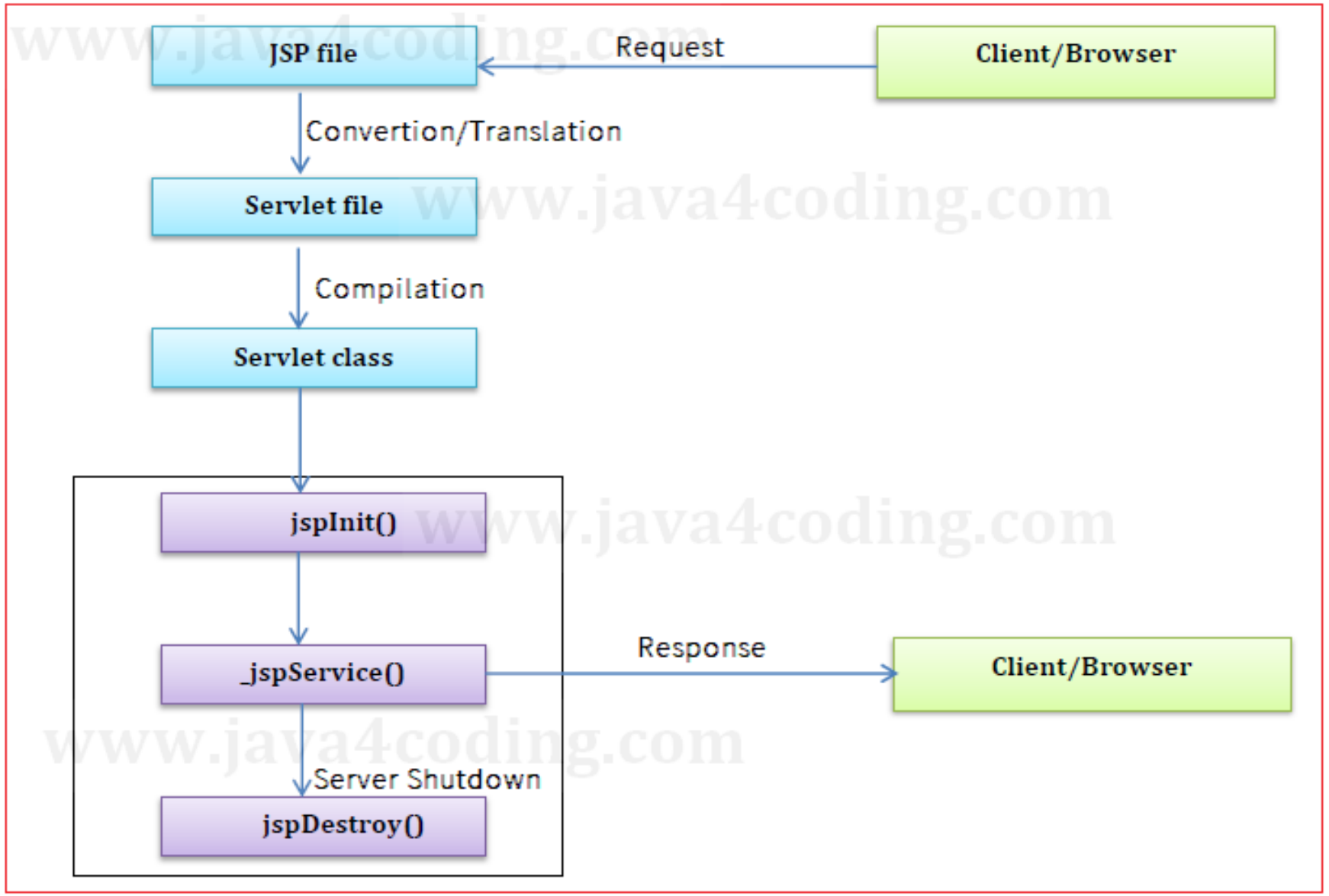
Java Server Pages - JSP

JavaServer Pages (JSP) technology allows you to easily create web content that has both static and dynamic components.

JSP technology makes available all the dynamic capabilities of Java Servlet technology but provides a more natural approach to creating static content.

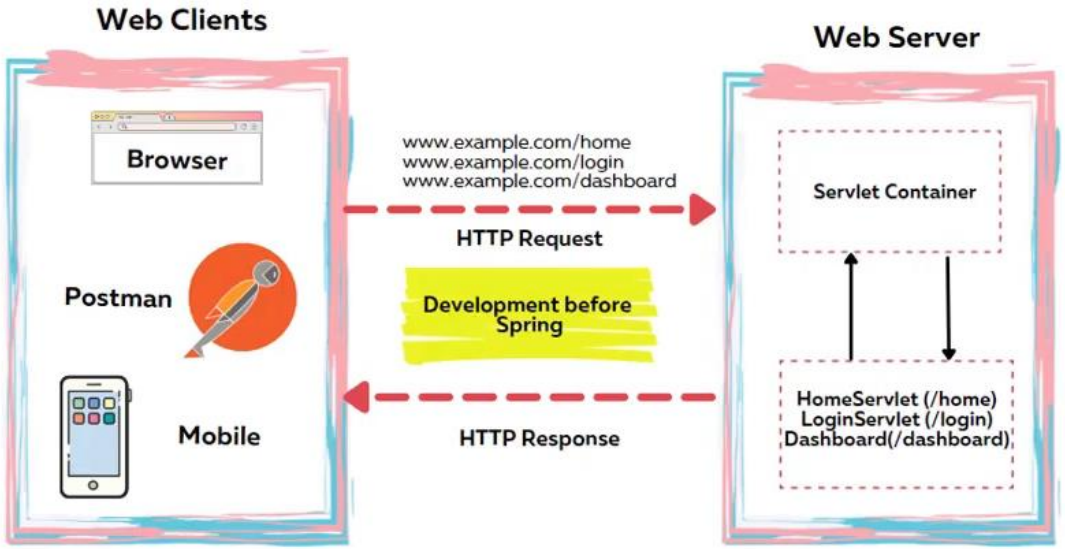
Java Server Pages (JSP) is a technology which is used to develop web pages by inserting Java code into the HTML pages by making special JSP tags. The JSP tags which allow java code to be included into it are <% —-java code—-%>.

How JSP Works ?



Spring MVC

- Spring MVC is a Java Framework **used to build Web Applications as well as RESTful web services**
- **It is build on top of Servlet API**
- It follows basic features of Spring IOC and DI
- A Spring MVC provides an elegant solution to use MVC in spring framework by the help of **DispatcherServlet**.
- DispatcherServlet is a class that receives the incoming request and maps it to the right resource such as controllers, models, and views.



With Spring

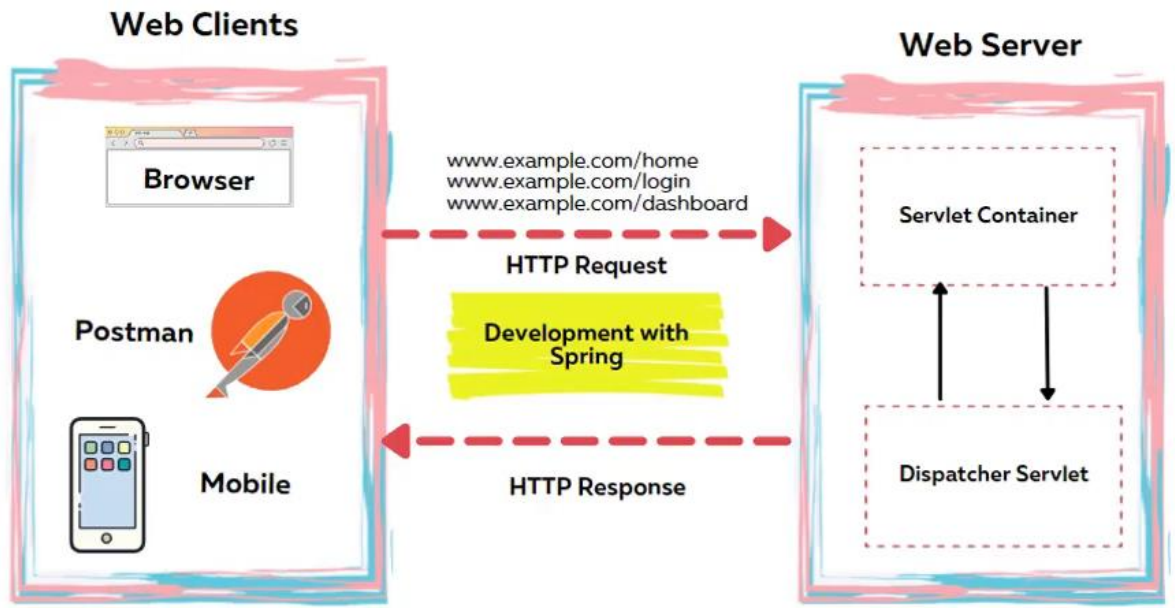
1. With Spring, it defines a servlet called Dispatcher Servlet which maintain all the URL mapping inside a web application.

2. The servlet container calls this Dispatcher Servlet for any client request, allowing the servlet to manage the request and the response. This way Spring internally does all the magic for Developers with out the need of defining the servlets inside a Web app.

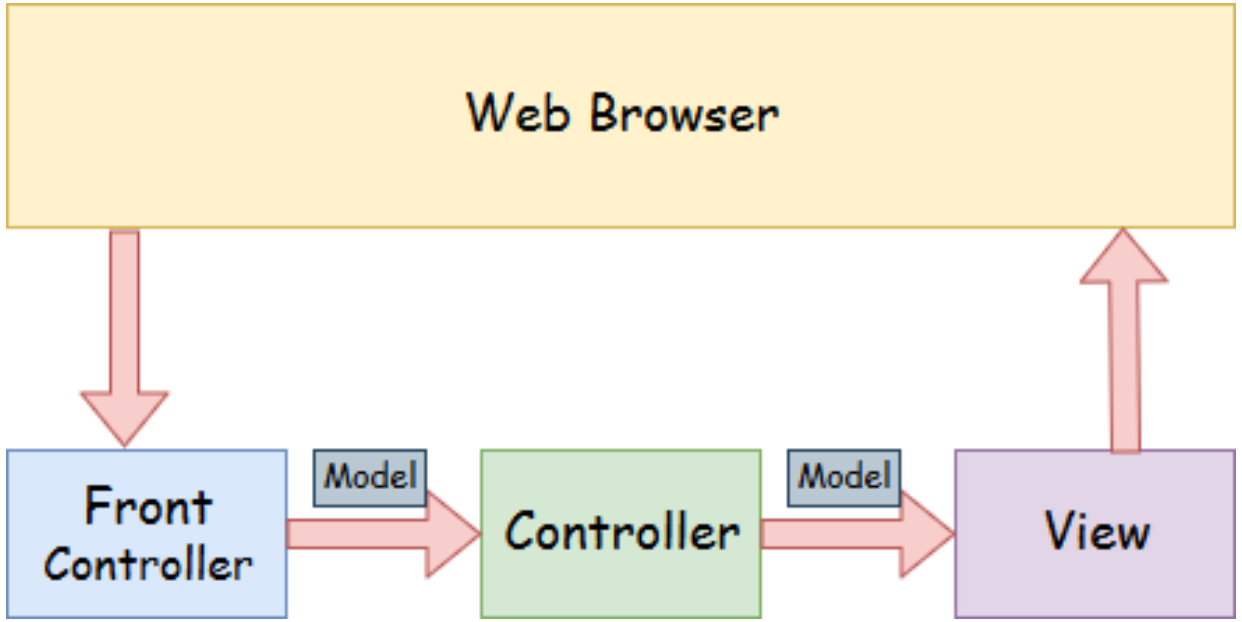
Before Spring

1. Before Spring, developer has to create a new servlet instance, configure it in the servlet container, and assign it to a specific URL path

2. When the client sends a request, Tomcat calls a method of the servlet associated with the path the client requested. The servlet gets the values on the request and builds the response that Tomcat sends back to the client.



Spring MVC

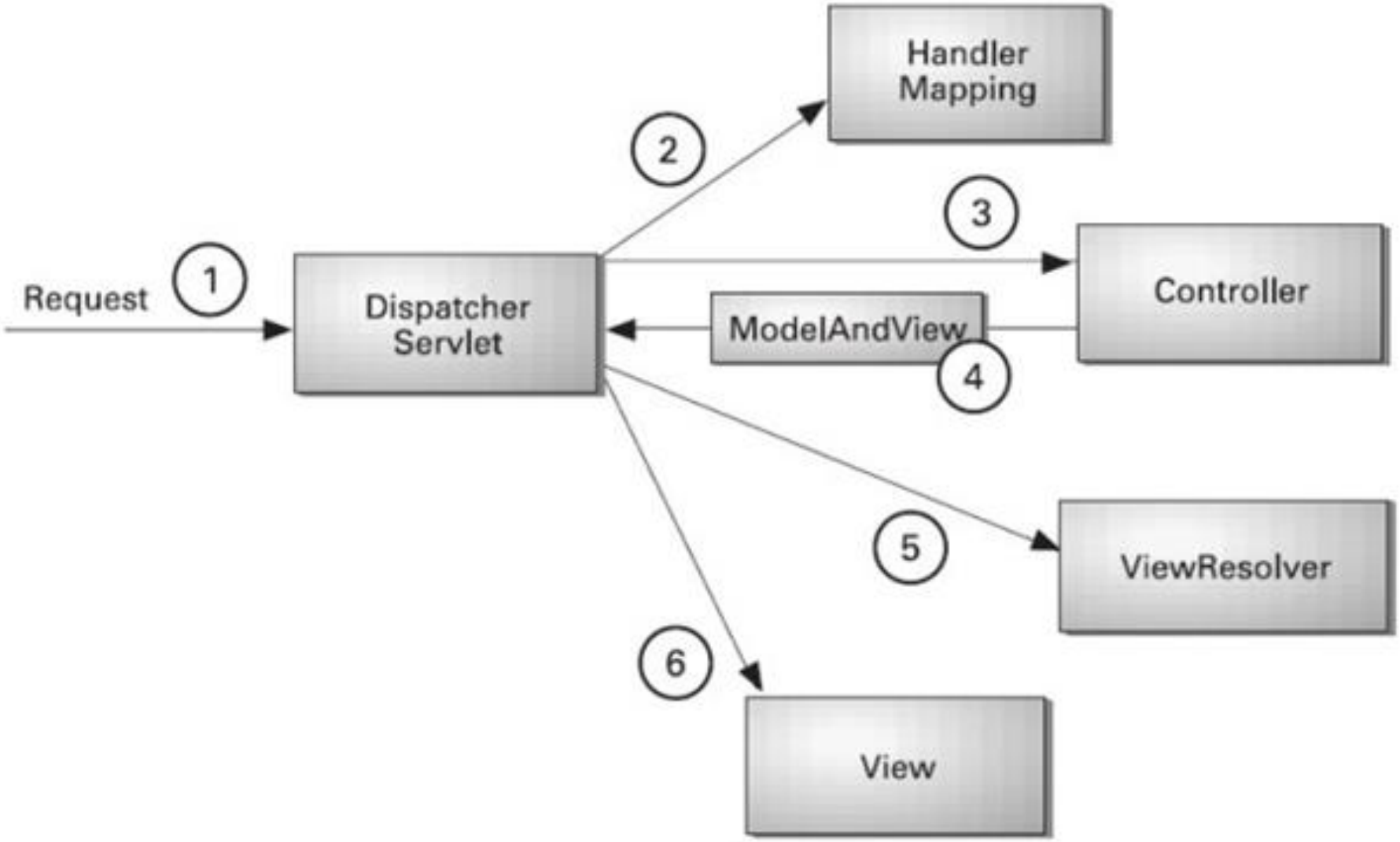


- Model-view-controller (MVC) is a well known design pattern for designing UI based applications.
- It mainly decouples business logic from UIs by separating the roles of model, view, and controller in an application.

Spring MVC

- Model - A model contains the data of the application. A data can be a single object or a collection of objects.
- Controller - A controller contains the business logic of an application. Here, the @Controller annotation is used to mark the class as the controller.
- View - A view represents the provided information in a particular format. Generally, JSP+JSTL is used to create a view page. Although spring also supports other view technologies such as Apache Velocity, Thymeleaf and FreeMarker.
- Front Controller - In Spring Web MVC, the DispatcherServlet class works as the front controller. It is responsible to manage the flow of the Spring MVC application.

Spring MVC Flow



Spring MVC Application Steps

1. Configure the dispatcher servlet in web.xml
2. Create Spring Configuration File
3. Configure View Resolver
4. Create Controller
5. Create a View to show(page)

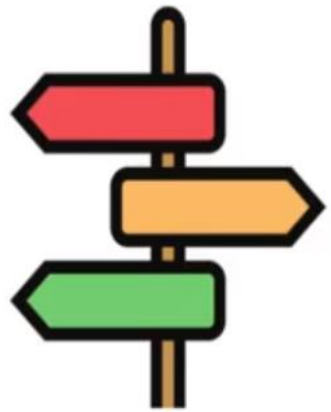
Spring MVC Advantages

1. Separate each role modal , view , controller etc
2. Powerful configuration.
3. It is sub framework of Spring Framework. Use of Spring core features like IOC etc.
4. Rapid Application Development.
5. Spring MVC is Flexible , easy to test and much features.

DEVELOPING WEB APPLICATIONS USING SPRING

APPROACH 1

- Web Apps which holds UI elements like HTML, CSS, JS and backend logic
- Here the App is responsible to fully prepare the view along with data in response to a client request
- Spring Core, Spring MVC, SpringBoot, Spring Data, Spring Rest, Spring Security will be used



Approaches to build web applications using Spring framework

APPROACH 2

- Web Apps which holds only backend logic. These Apps send data like JSON to separate UI Apps built based on Angular, React etc.
- Here the App is responsible to only process the request and respond with only data ignoring view.
- Spring Core, SpringBoot, Spring Data, Spring Rest, Spring Security will be used

Spring MVC is the key differentiator between these two approaches.

What is Spring Boot ?

Java Spring Boot (Spring Boot) is a tool that makes developing web application and microservices with Spring Framework faster and easier through three core capabilities:

1.Autoconfiguration

2.An opinionated approach to configuration

3.The ability to create standalone applications

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

2. Before SpringBoot, Developer need to configure a servlet container, establish link b/w Tomcat and Dispatcher servlet, deploy into a server, define lot of dependencies……..

3. But with SpringBoot, we can create Web Apps skeletons with in seconds or at least in 1-2 mins ☺. It helps eliminating all the configurations we need to do.

4. Spring Boot is now one of the most appreciated projects in the Spring ecosystem. It helps us to create Spring apps more efficiently and focus on the business code.

5. SpringBoot is a mandatory skill now due to the latest trends like Full Stack Development, Microservices, Serverless, Containers, Docker etc.

Why Spring Boot?

- Provide a radically faster and widely accessible getting-started experience for all Spring development
- Be opinionated out of the box but get out of the way quickly as requirements start to diverge from the defaults
- Provide a range of non-functional features that are common to large classes of projects
    - Embedded servers, metrics, health checks, externalized configuration, containerization, etc.

Spring Boot Starters

Spring Boot *"Starter"* Dependencies

- Easy way to bring in multiple coordinated dependencies
  - Including *"Transitive"* Dependencies

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

Version not needed!
Defined by parent

Resolves ~ 18 JARs!

spring-boot-.jar*
spring-core-.jar*
spring-context-.jar*
spring-aop-.jar*
spring-beans-.jar*
**-slf4j-*.jar*
...

Spring Boot Starters

Many Starters are available out of the box

- Not essential but *strongly* recommended for getting started
- Coordinated dependencies for common Java enterprise frameworks
  - Pick the starters you need in your project
- To name a few:
  - **`spring-boot-starter-jdbc`**
  - **`spring-boot-starter-data-jpa`**
  - **`spring-boot-starter-web`**
  - **`spring-boot-starter-batch`**

Spring Boot Auto Configuration

Auto-configuration enabled by @EnableAutoConfiguration

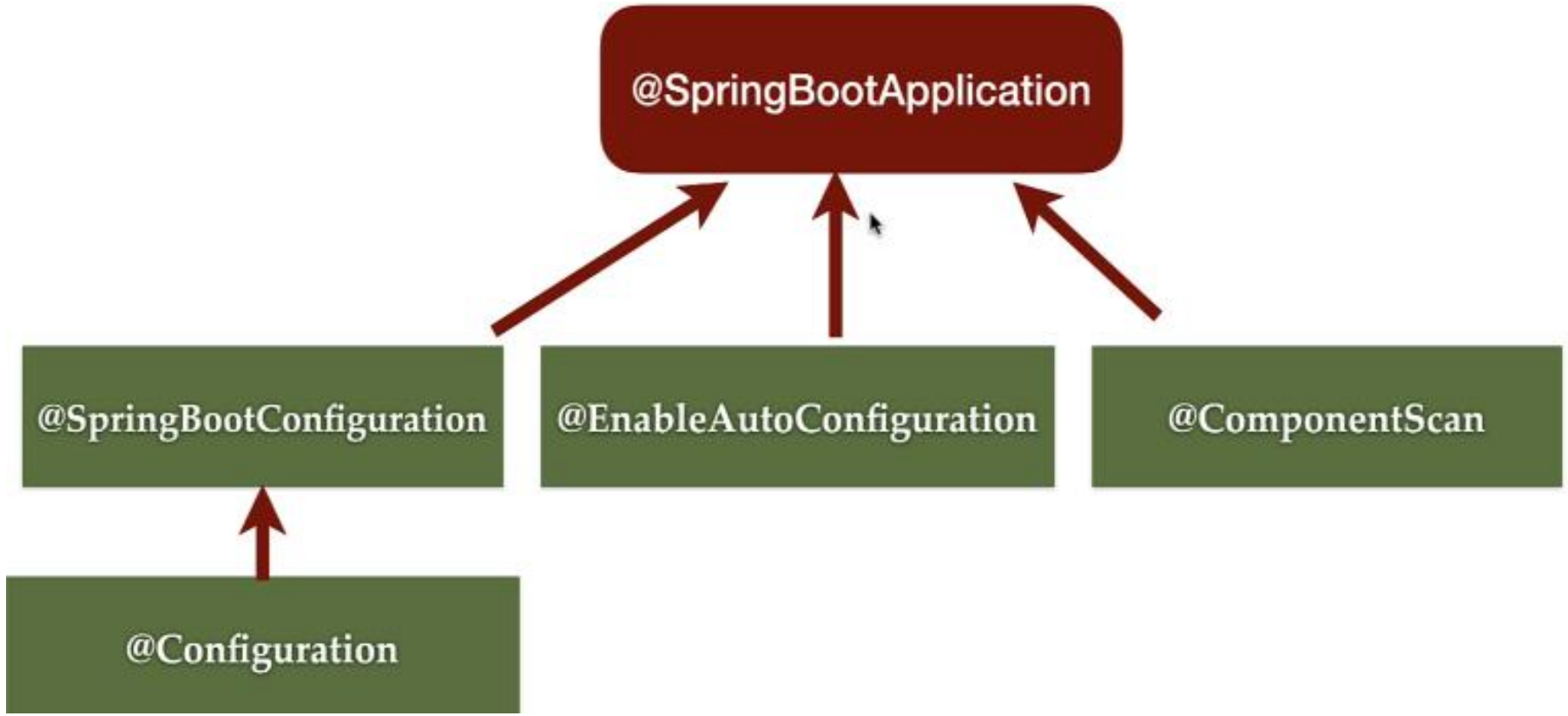
- Spring Boot automatically creates beans it thinks you need based on some conditions
- *`@EnableAutoConfiguration`* annotation on a Spring Java configuration class enables auto-configuration

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

`@SpringBootConfiguration` simply extends `@Configuration`

`SpringApplication` is actually a Spring Boot class

Spring Boot Auto Configuration



Spring Boot Auto Configuration

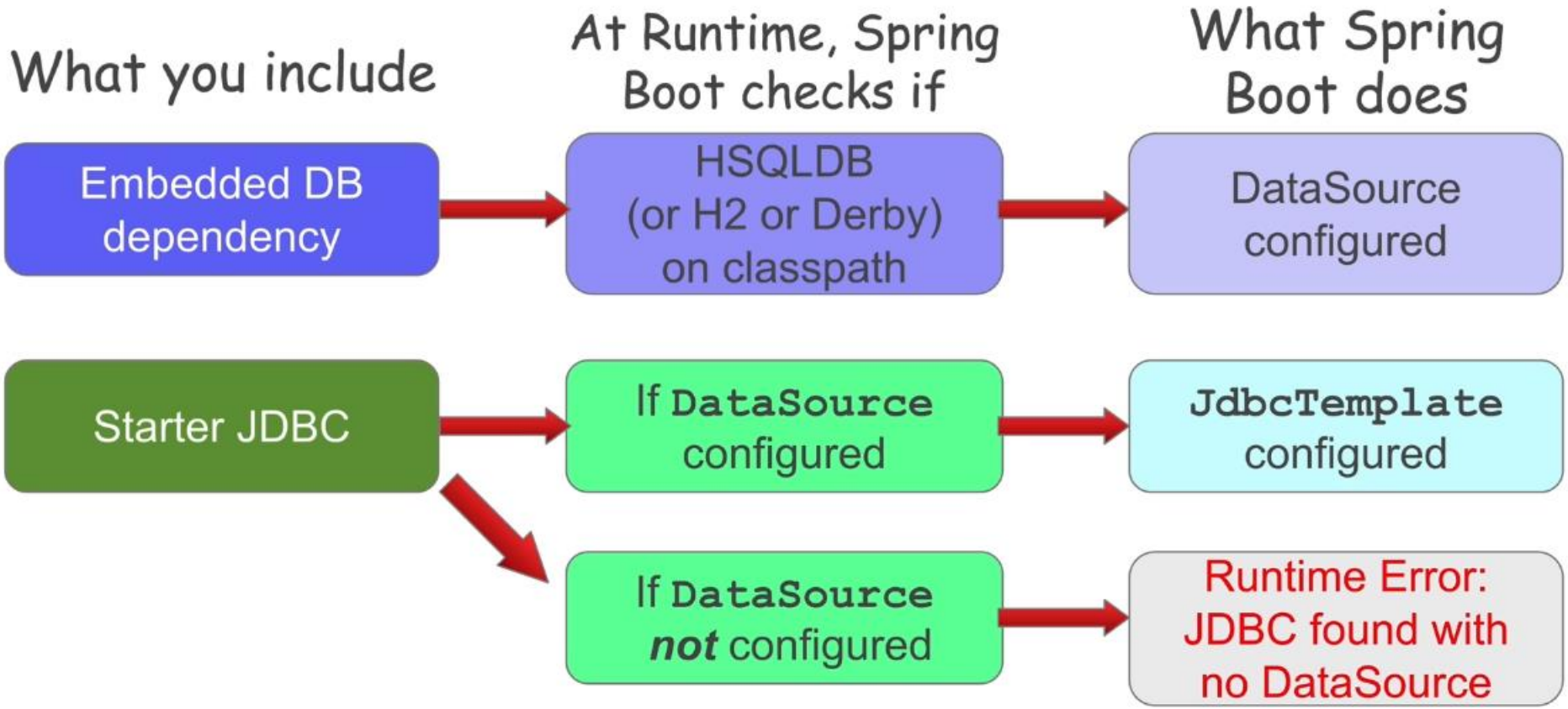
Example 1: if you add **spring-boot-starter-web** Jar dependency to your Spring boot application, Spring Boot assumes you are trying to build a SpringMVC-based web application and automatically tries to register Spring beans such as **DispatcherServlet, ViewResolver** if it is not already registered.

Example 2: If you have any embedded database drivers in the classpath, such as H2 or HSQL, and if you haven't configured a DataSource bean explicitly, then Spring Boot will automatically register a DataSource bean using in-memory database settings.

Example 3: If you add **spring-boot-starter-data-jpa** starter dependency then it assume that you are trying to use Hibernate to develop DAO layer so Spring boot automatically register the Spring beans such as **Data source, Entity manager factory/session factory, Transaction manager.**

Spring Boot auto-configuration attempts to automatically configure Spring application based on the jar dependencies that you have added to project.

Examples of Auto-configuration: DataSource, JdbcTemplate



-> Spring boot auto configuration is implemented in **spring-boot-autoconfigure.jar**

-> Example of Spring MVC:

- Configure DispatcherServlet (**DispatcherServletAutoConfiguration**)
- Configure Embedded Tomcat Server (**EmbeddedWebServerFactoryCustomizerAutoConfiguration**)
- Configure Default Error Page (**ErrorMvcAutoConfiguration**)
- Configure JSON to Java Conversion (**JacksonAutoConfiguration**)