JAVA FULLSTACK

What you will learn?

- Master everything required to build an end to end Enterprise Java Application
- Understand the different layers that make up a Enterprise Java Application
- Create a data access layer in two simple steps
- Create the Presentation and Services Layers for your application
- Develop Utility Classes
- Learn how to send emails form your applications
- Use third party libraries in your applications
- Create two end to end mini applications
- Create a AngularJS front end for the Java backend
- Learn how to enable logging
- Implement Security
- Learn the two different ways to deploy your application
- All in simple and easy steps

Requirements

- Knowledge of Spring Framework and Spring Boot
- Or Should have completed my Spring Framework in easy steps course

Where we Start with?

- Spring
- What are Micro Services?
- Layers in Java Application
- 10 Different Classes
- Technologies

Concepts we learn

- Creating Data Access Layer
- Spring Data
- Presentation Layer
- Spring MVC
- Data Access Layer
- Services
- CRUD
- Presentation Layer
- Utility Layer
- Spring Email
- Java Mail Sender
- Configuration
- Generate Reports
- Third Party Jars
- JFREE Charts
- Integration Layer
- Restful Layer
- Post Man

Mini project concepts

- Data Access Layer
- Service Layer
- Presentation Layer
- Utility Layer

Mini Projects

- Flight Reservation
- Flight Check-in

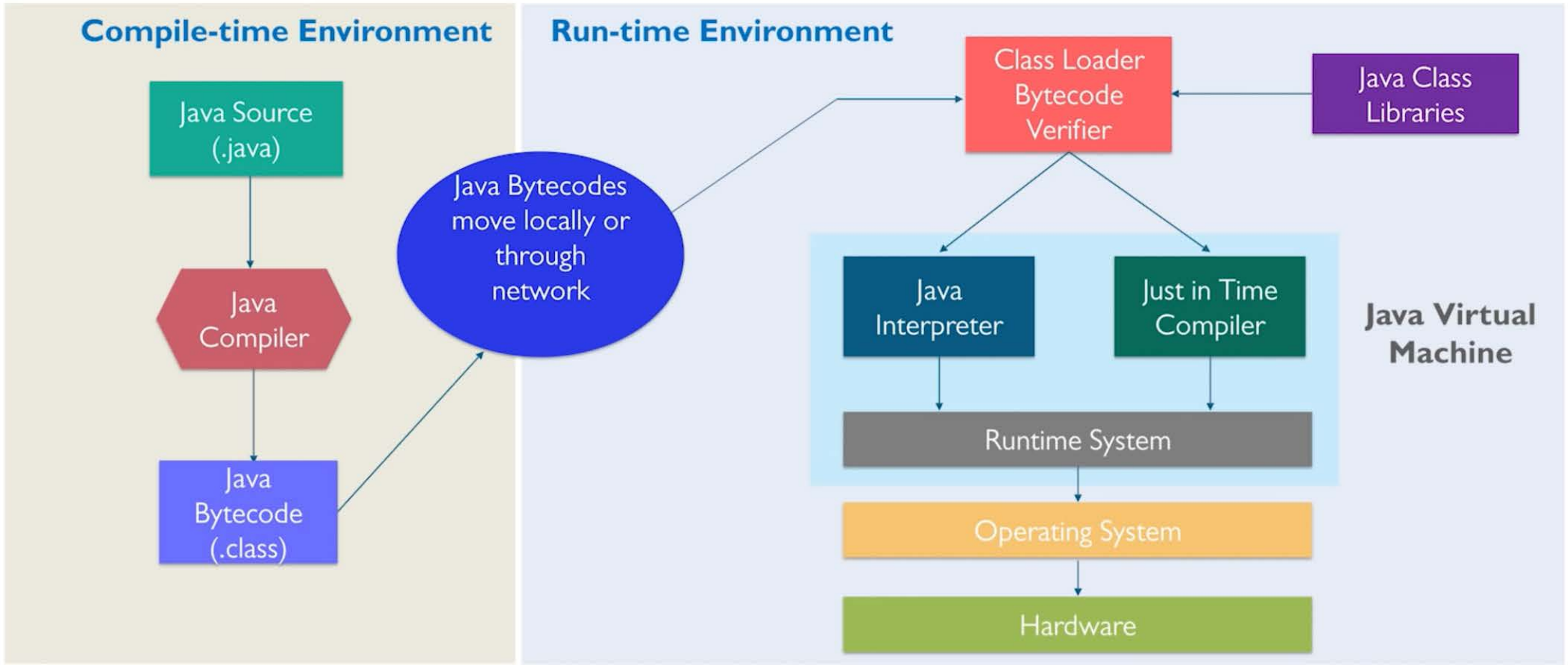
Functional Requirements

- User Registration and Login
- Search Flights
- Book flights
- Check in

Non Functional Requirements

- Security
- Logging
- Deployment etc

Java Working

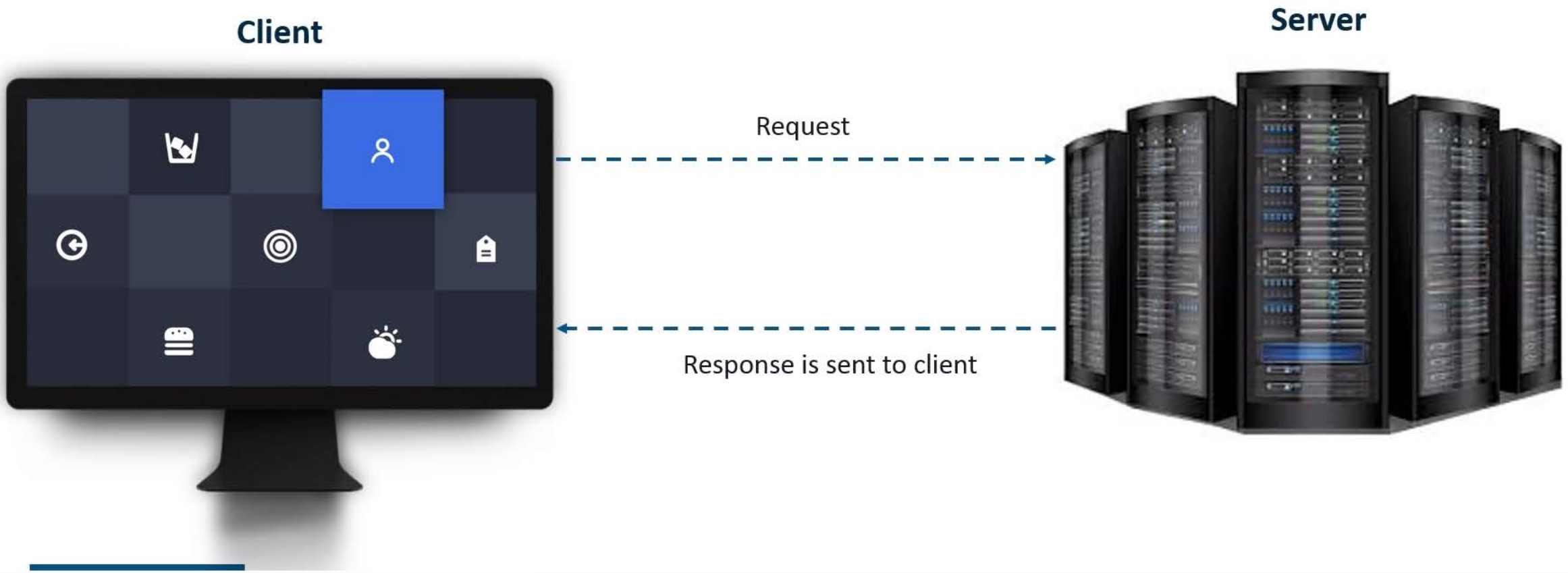


SERVLET


CGI vs Servlets
5
What are Servlets?
4
6
Servlet Architecture
INTRODUCTION TO SERVLETS
HTTP Request & Response
3
7
Generic Servlets
WEB & HTTP
2
8
Servlet Request & Response
Introduction to WEB
1
9
Servlet Classes & Interfaces

Introduction to Web

Web is basically a system of **Internet** servers that support specially formatted documents



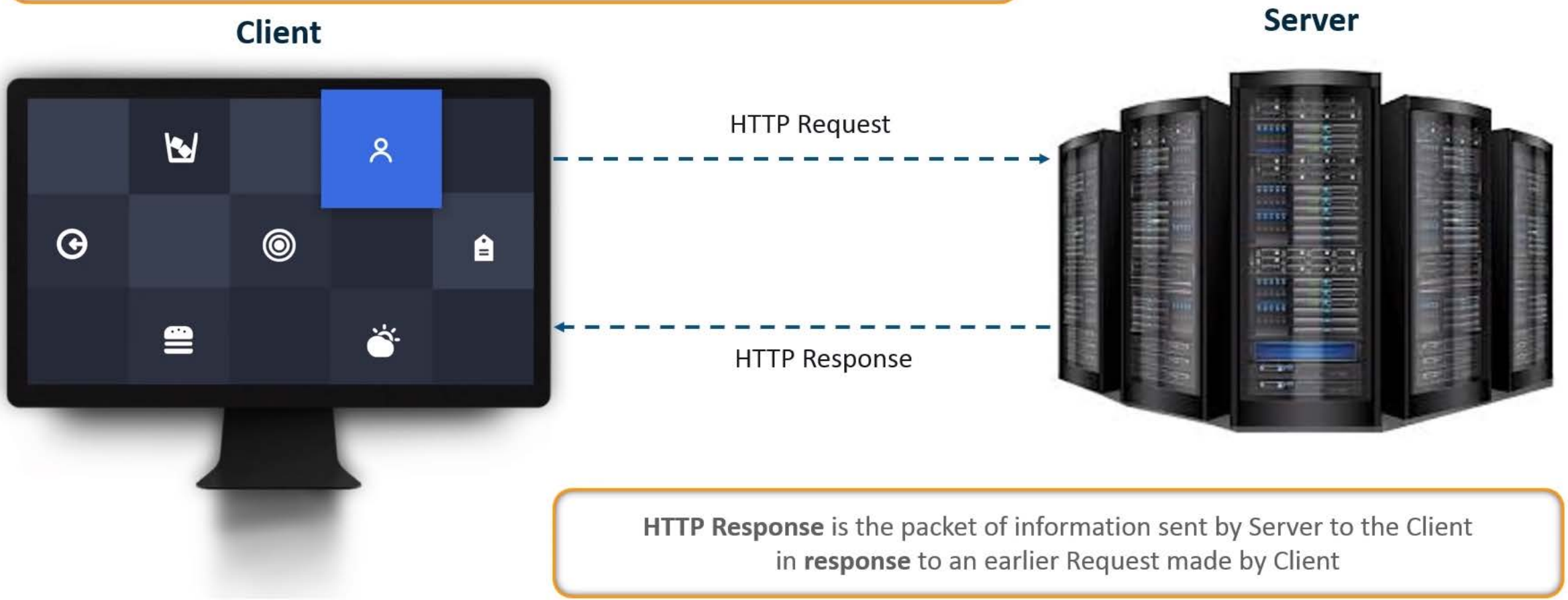
Web & HTTP

http://

HTTP is a protocol that clients and servers use on the web to communicate.

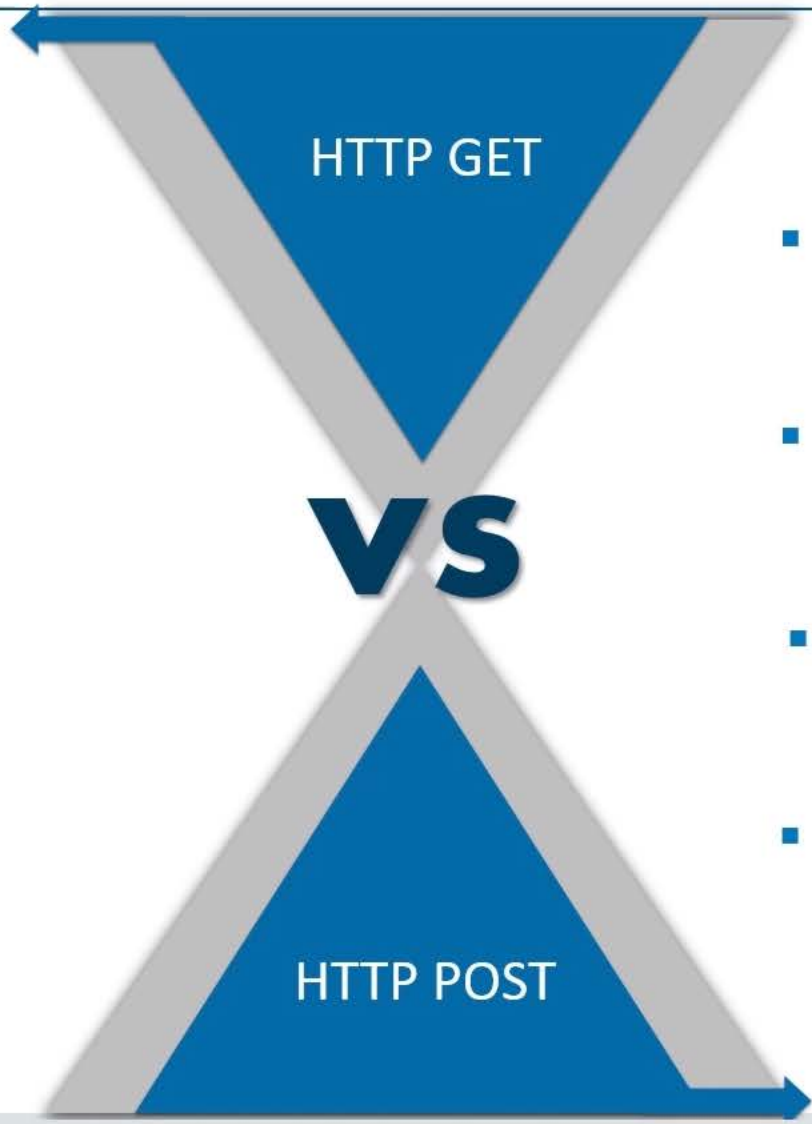
HTTP Request & Response

HTTP Request is a packet of Information that one computer sends to another computer to communicate something



HTTP Response is the packet of information sent by Server to the Client in **response** to an earlier Request made by Client

Difference between HTTP Get & Post



- Data is sent in header
- Limited data can be sent
- Not secured
- Can be bookmarked

- Data is sent in request body
- Large amount of data can be sent
- It is secured
- Cannot be bookmarked

INTRODUCTION TO SERVLETS

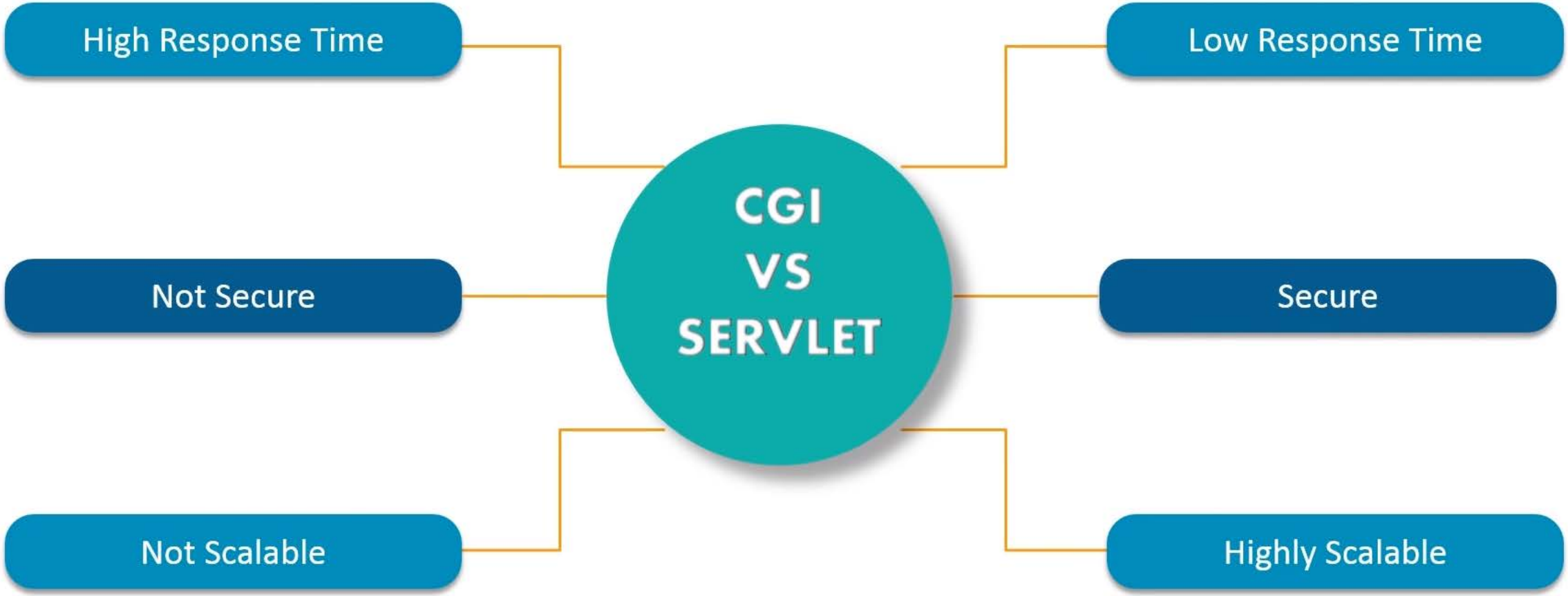
What are Servlets?

What are Servlets?

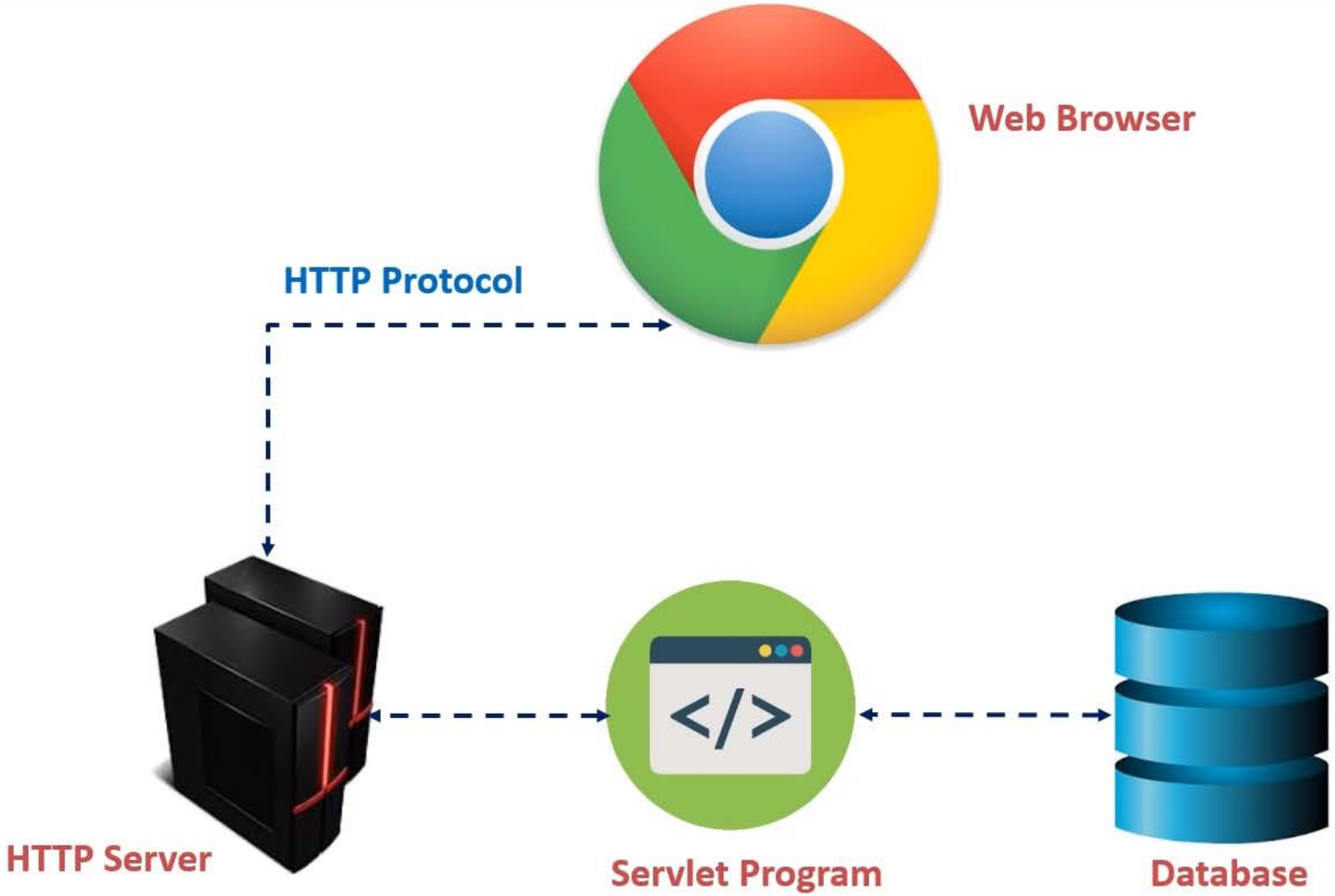
CGI VS SERVLETS

What is CGI?

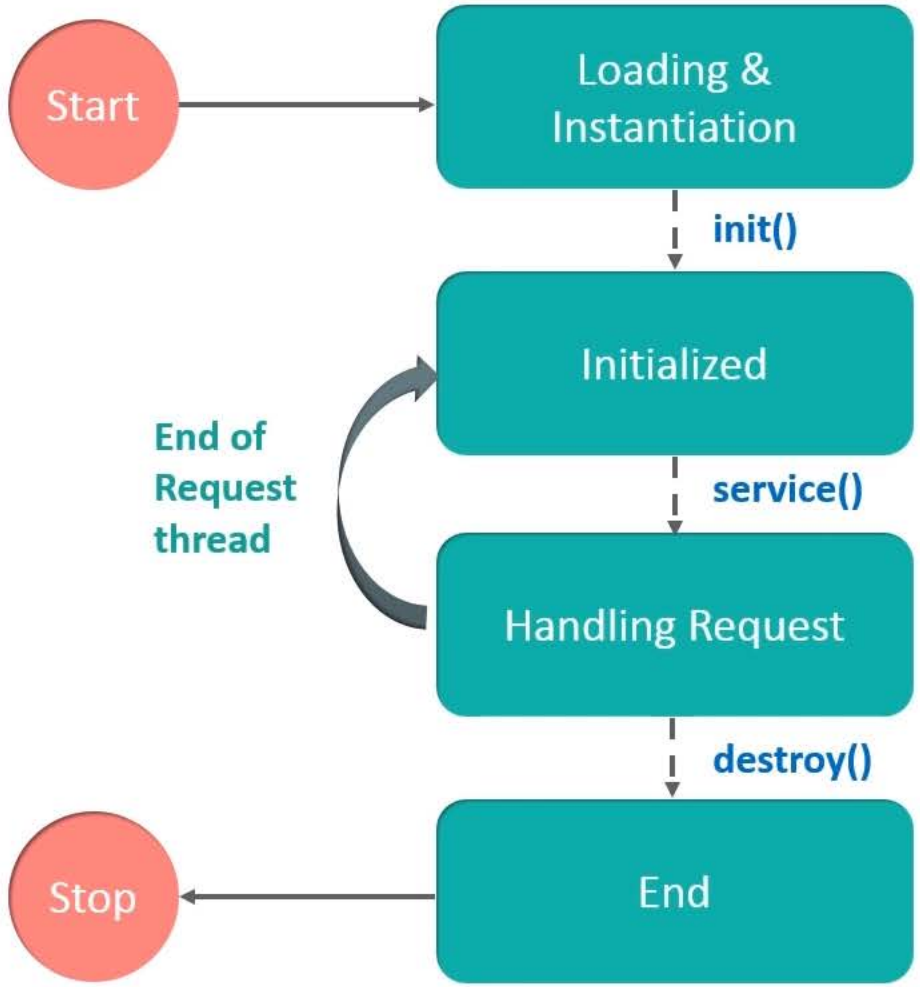
Common Gateway Interface (**CGI**) is a standard for writing programs that can interact through a Web server with a client running a Web browser.



Servlet Architecture



Servlet Life Cycle



Steps to create Simple Servlet

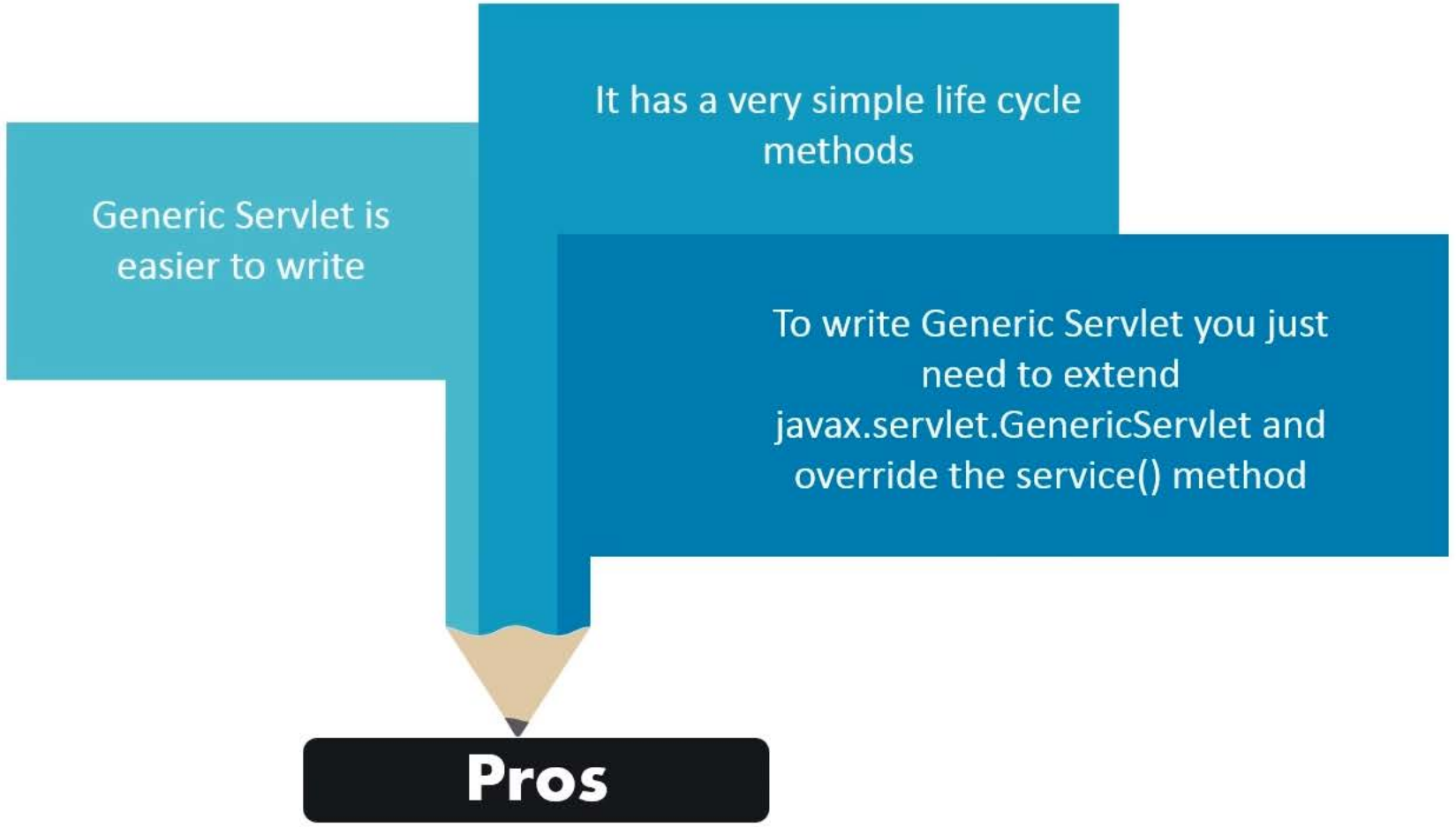
1. Create and compile simple servlet code
2. Add mappings to web.xml file
3. Start Apache Tomcat
4. Start Web Browser & Request Servlet

Servlets

GENERIC SERVLET

Generic Servlet

A *generic servlet* is a protocol independent Servlet that should always override the service() method to handle the client request..



Request & Response

Request & Response

Servlet Request

Servlet Response

An object of ServletRequest is used to provide the client request information to a servlet such as content type, content length, parameter names and values, header informations, attributes etc.

Request & Response

Servlet Request

Servlet Response

An object of ServletResponse is used to write the information back to the client.

doGet()	shall be used when small amount of data and insensitive data like a query has to be sent as a request.
doPost()	shall be used when comparatively large amount of sensitive data has to be sent. Examples are sending data after filling up a form or sending login id and password.

Servlet Classes & Interfaces

Servlet	Declares life cycle method of servlet
ServletConfig	Allows servlet to get Initialization methods
ServletContext	Enables servlet to log access and access information
ServletRequest	Used to read data from client request
ServletResponse	Used to write data to client response
GenericServlet	Implements servlet and Servlet.Config Interface
ServletInputStream	Provides input stream to read requests from the client
ServletOutputStream	Provides output stream to write responses to the client
ServletException	Indicates servlet error has occured
UnavailableException	Indicates servlet is not available
HttpServlet	Provides methods to handle Http Request and Response
HttpServletRequest	Enables servlets to read data from Http Request
HttpServletResponse	Enables servlets to write data to Http Response

MAVEN

Agenda

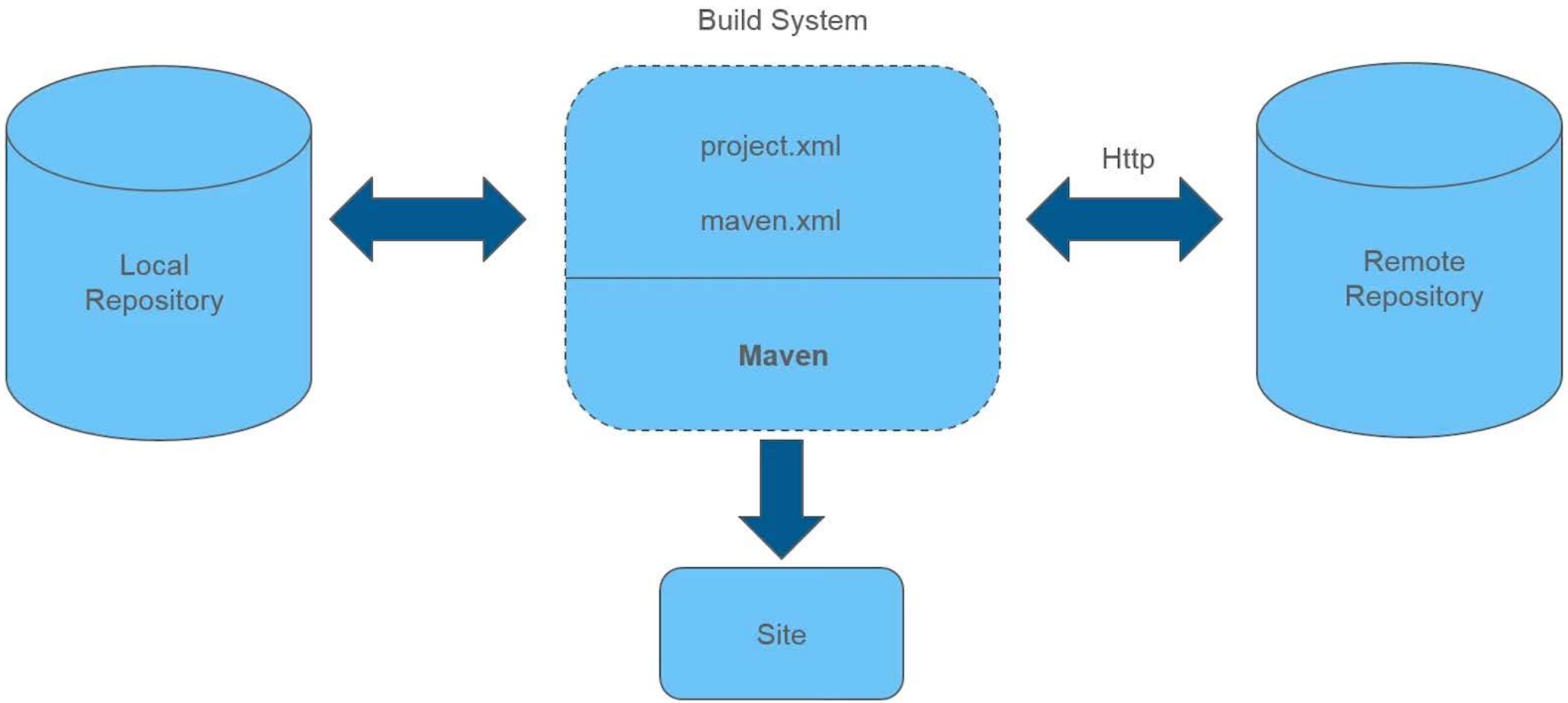
- Why do we need Maven?
- What is Maven?
- Maven Architecture?
- Maven LifeCycle, Phases and Goals
- Advantages of Maven
- Demo of Maven

WHY DO WE NEED MAVEN?

WHAT IS MAVEN?

- Project of Apache Software Foundation
- Build Automation tool
- Project Management tool
- Handles complete build process

MAVEN ARCHITECTURE



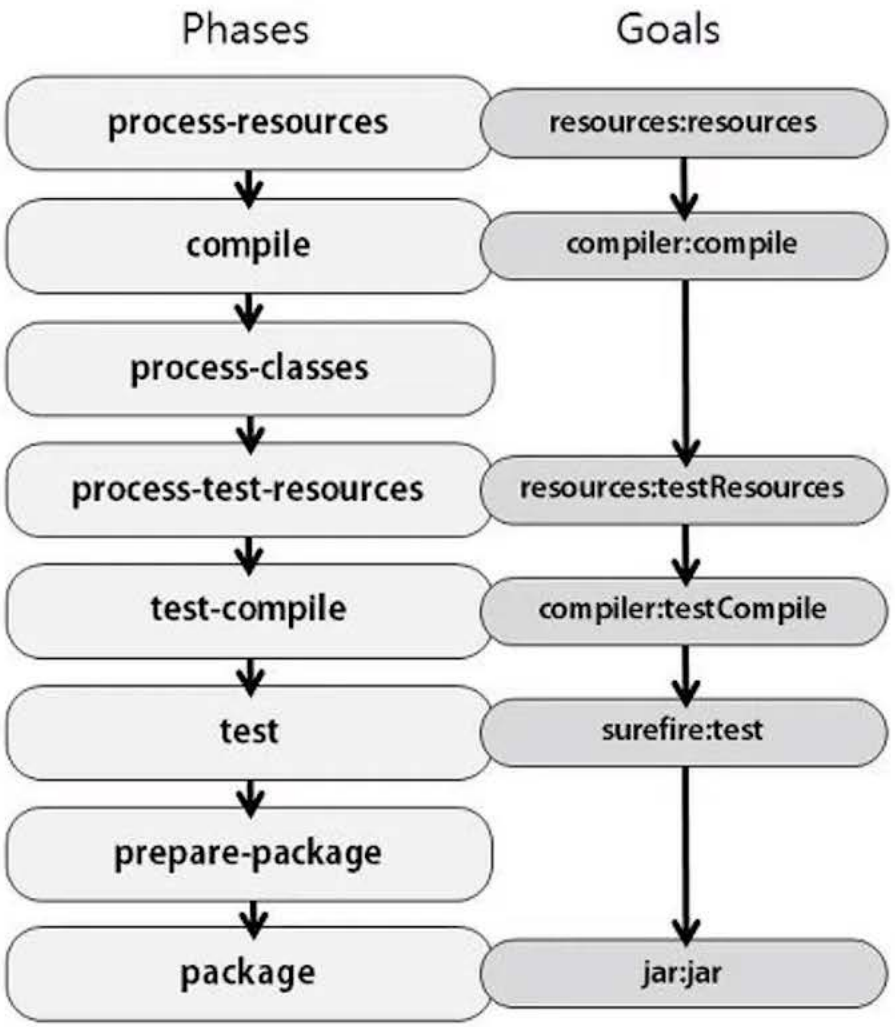
MAVEN LIFECYCLE

Clean Lifecyle
pre-clean
clean
post-clean

Default Lifecyle	
validate	test-compile
initialize	process-test-classes
generate-sources	test
process-sources	prepare-package
generate-resources	**package**
process-resources	pre-integration-test
compile	integration-test
process-classes	post-integration-test
generate-test-sources	verify
process-test-sources	**install**
generate-test-resources	**deploy**
processs-test-resources	

Site Lifecyle
pre-site
site
post-site
site-deploy

MAVEN PHASES AND GOALS



ADVANTAGES OF MAVEN

01 Making the build process easy

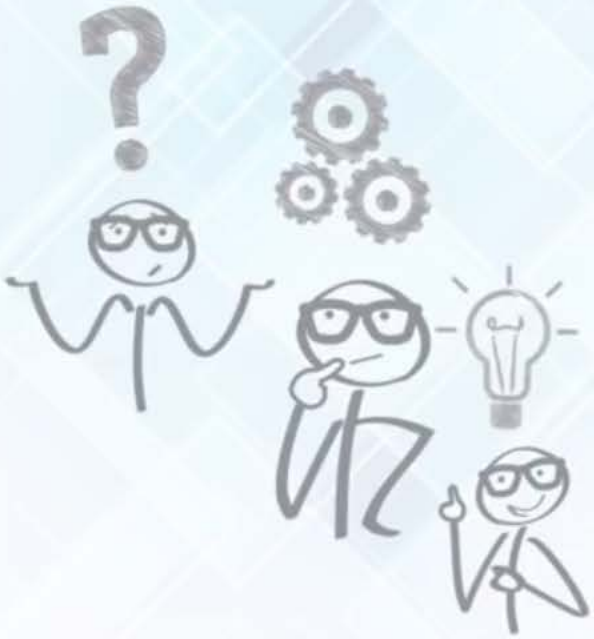
Provides a uniform build system 02

03 Making the build process easy

Provides guidelines for best practices 04

05 Allows transparent migration to new features

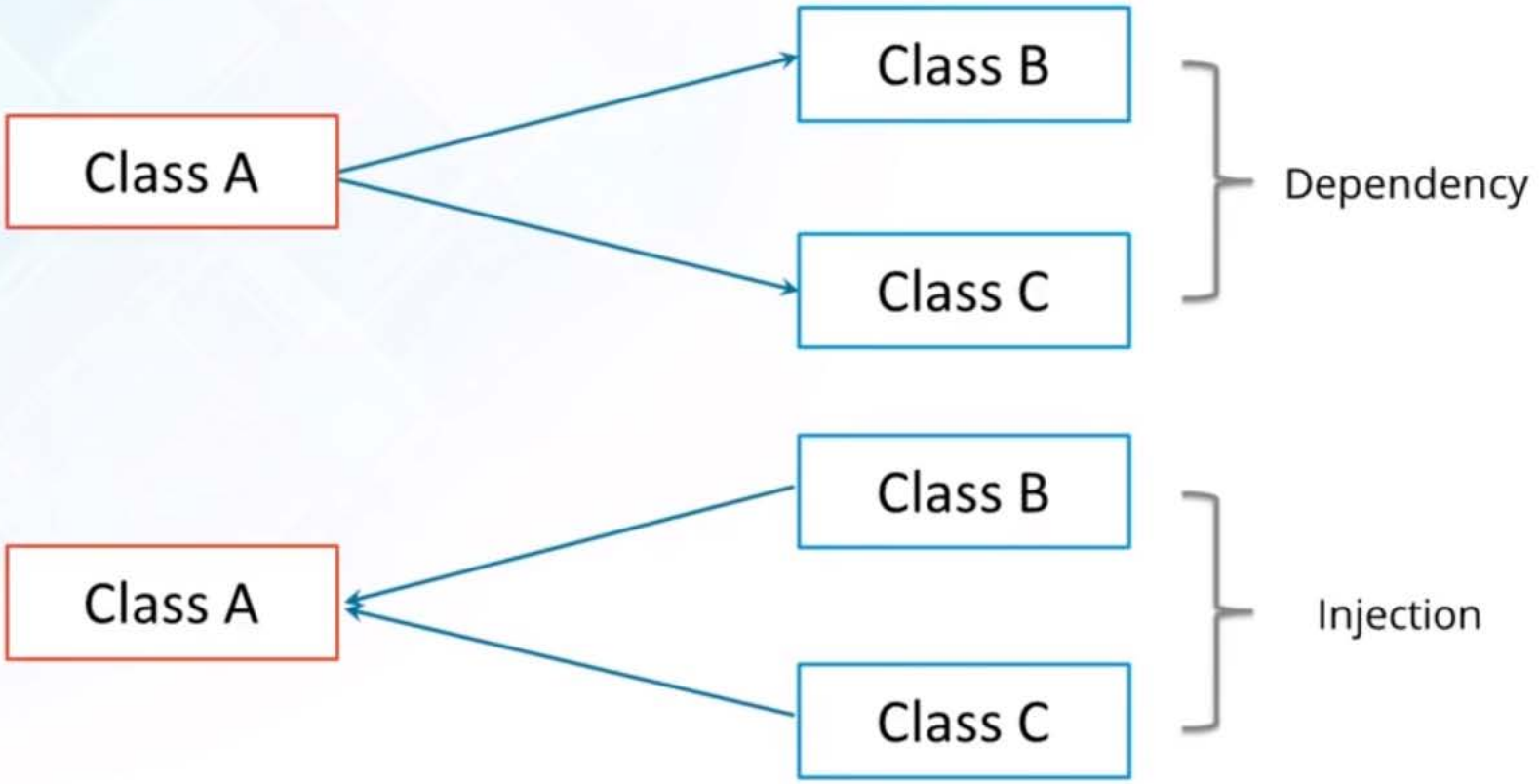
Java Frameworks



This fits with my code too !

Java Framework Principle

Abides **the Hollywood Principle** that is **"Don't call us, we'll call you".**
Also called Inversion of flow or Inversion of Control



Different Java Frameworks

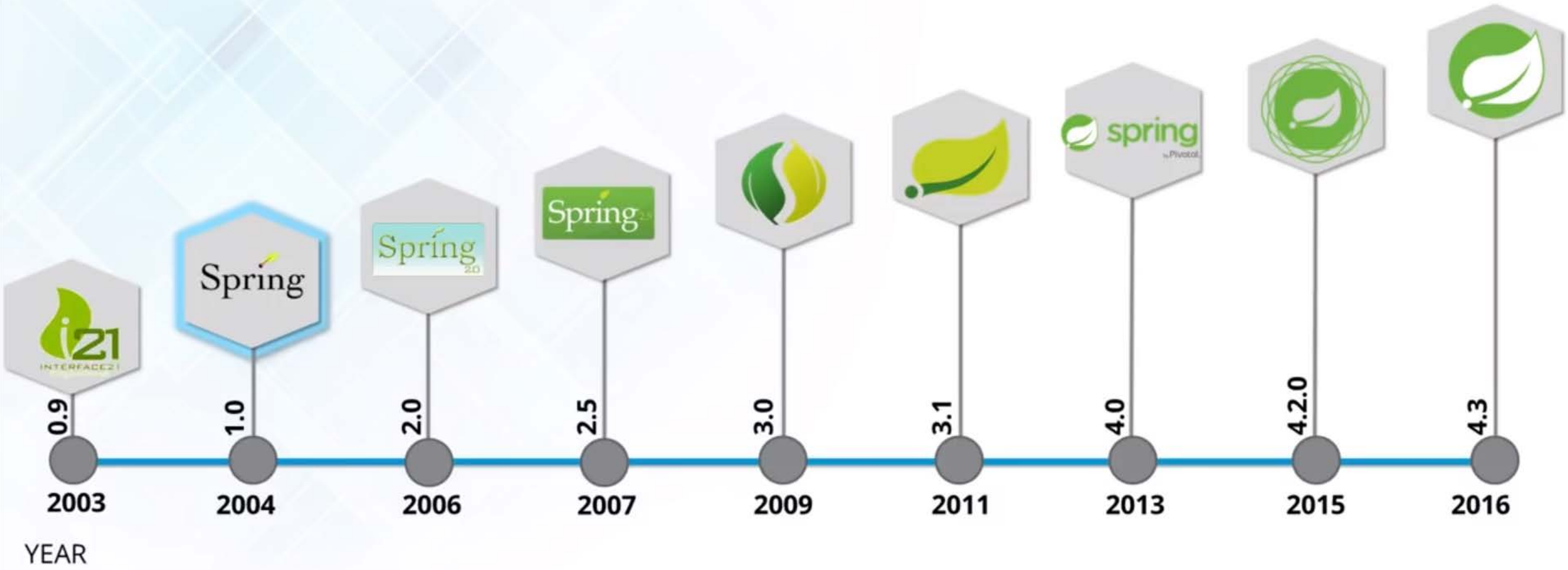
Spring Framework

Spring Origin

First version of Spring was first released on **February 2003**, By **Rod Johnson**.

Spring has been hosted on **SourceForge** since January 2003.

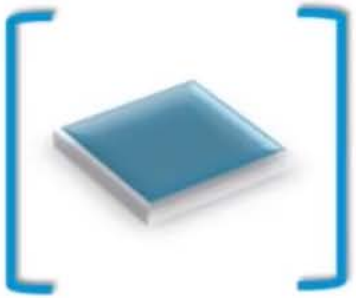
Spring History



Definition

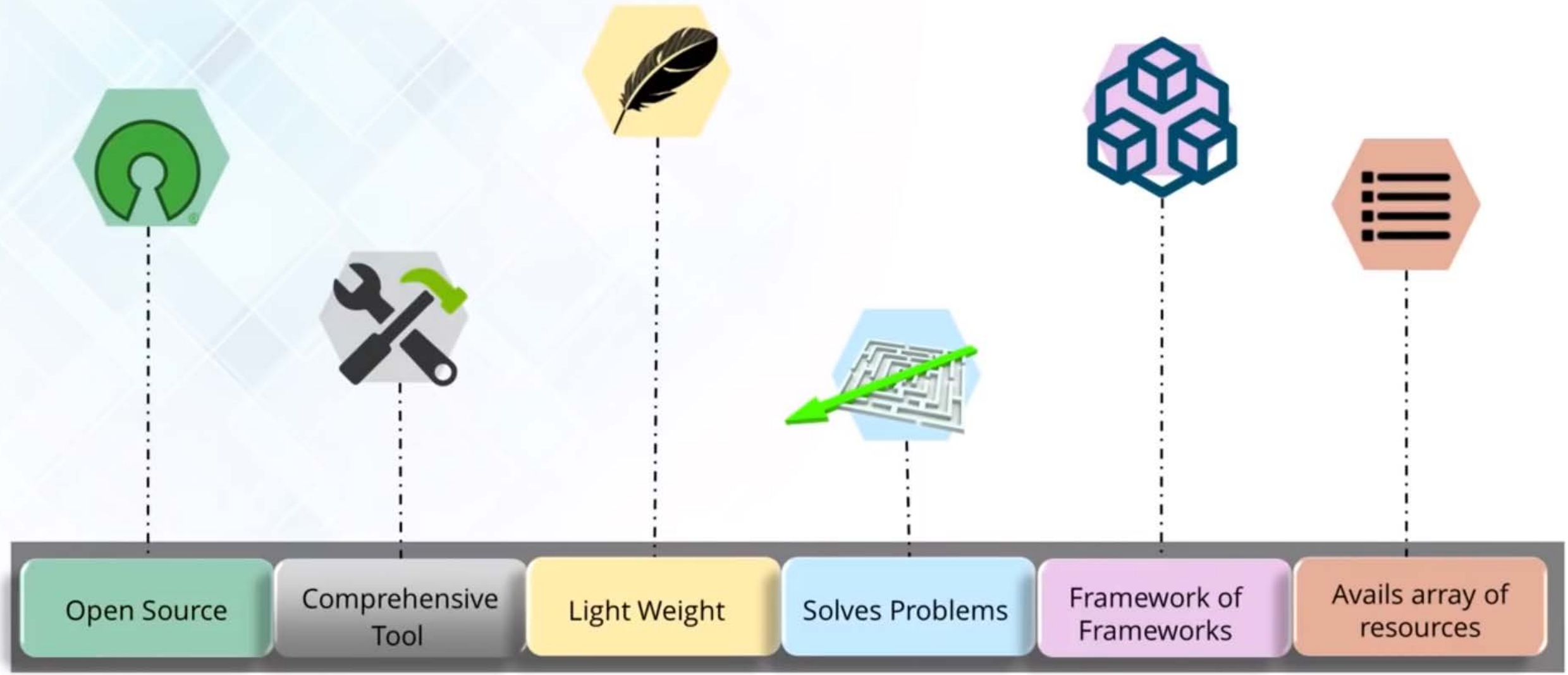
Spring is a complete and a modular framework for developing enterprise applications in java.

spring framework can be used for **all layer implementations** for a real time application



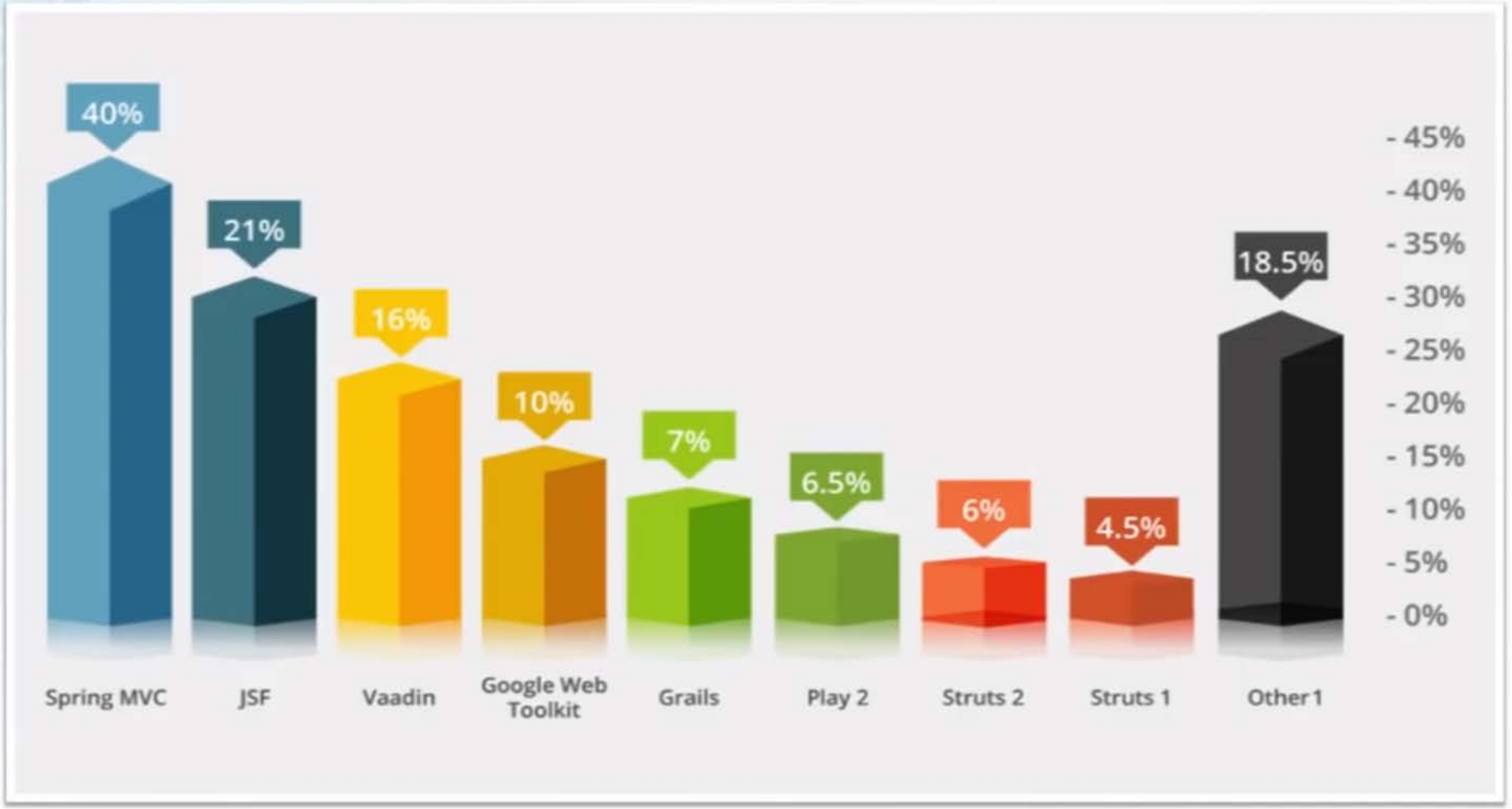
spring can be used for the development of **particular layer** of a real time application

Features Of Spring



Why Spring Framework ?

Uses Of Spring Over Other Frameworks

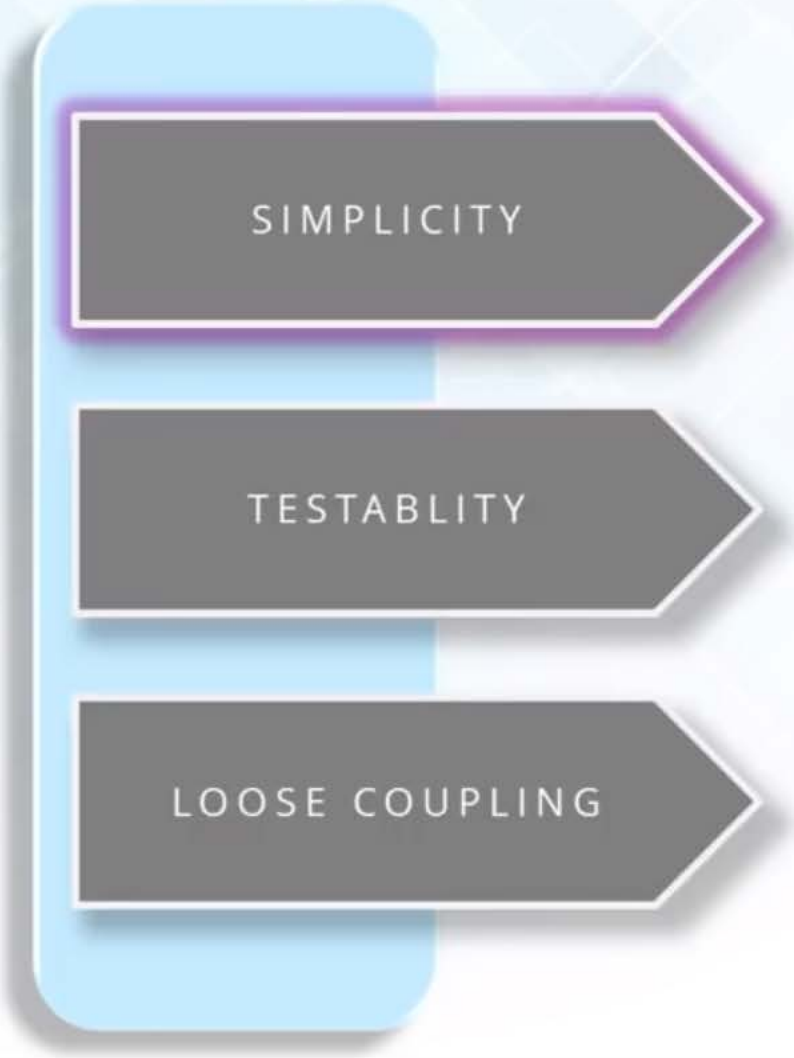


Why Spring Is So Popular ?

- SIMPLICITY
- TESTABLITY
- LOOSE COUPLING

These 3 are basically the main reasons for its popularity.
Lets now understand how they are exactly making it so

Simplicity



Spring framework is simple because as it is non-invasive.
It uses **POJO** and **POJI**

If a java class is not coupled with any technology (or) any framework then that java class is called **"POJO"** (plain old java class)

```
public class MyClass
  {   ...
      ...
      ...
  }
```

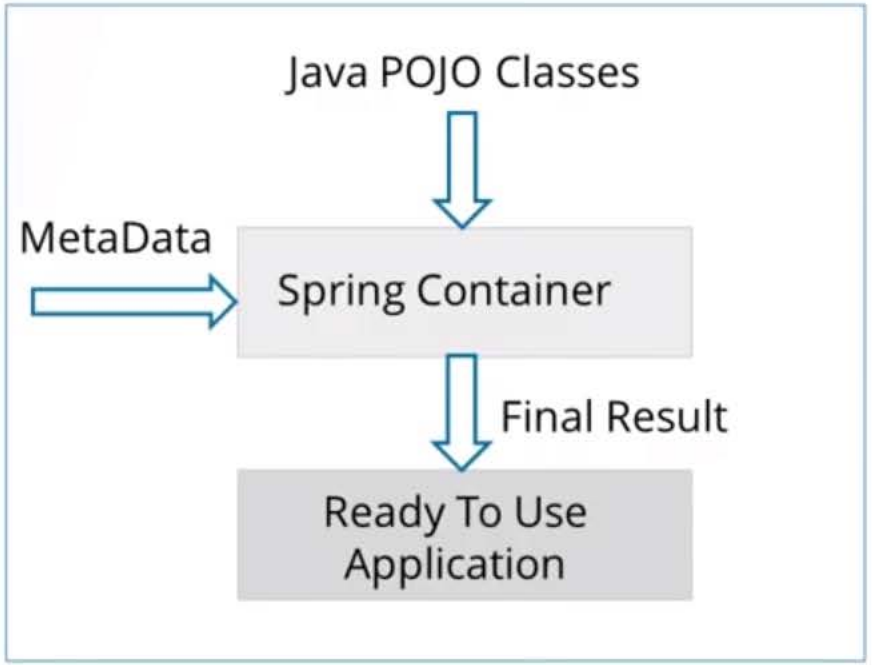
Testablity

SIMPLICITY

TESTABLITY

LOOSE COUPLING

Actually for writing the spring application, server [**Container**] is not mandatory because it has it own container to run the applications



Loose Coupling

SIMPLICITY

TESTABLITY

LOOSE COUPLING

Spring objects are loosely coupled, this is the core concept of spring framework

```
class Rider
{
   Bike b;
   public void setBike(Bike b)
   {
     this.b = b;
   }

   void ride()
   {
     b.start();
   }
}
```

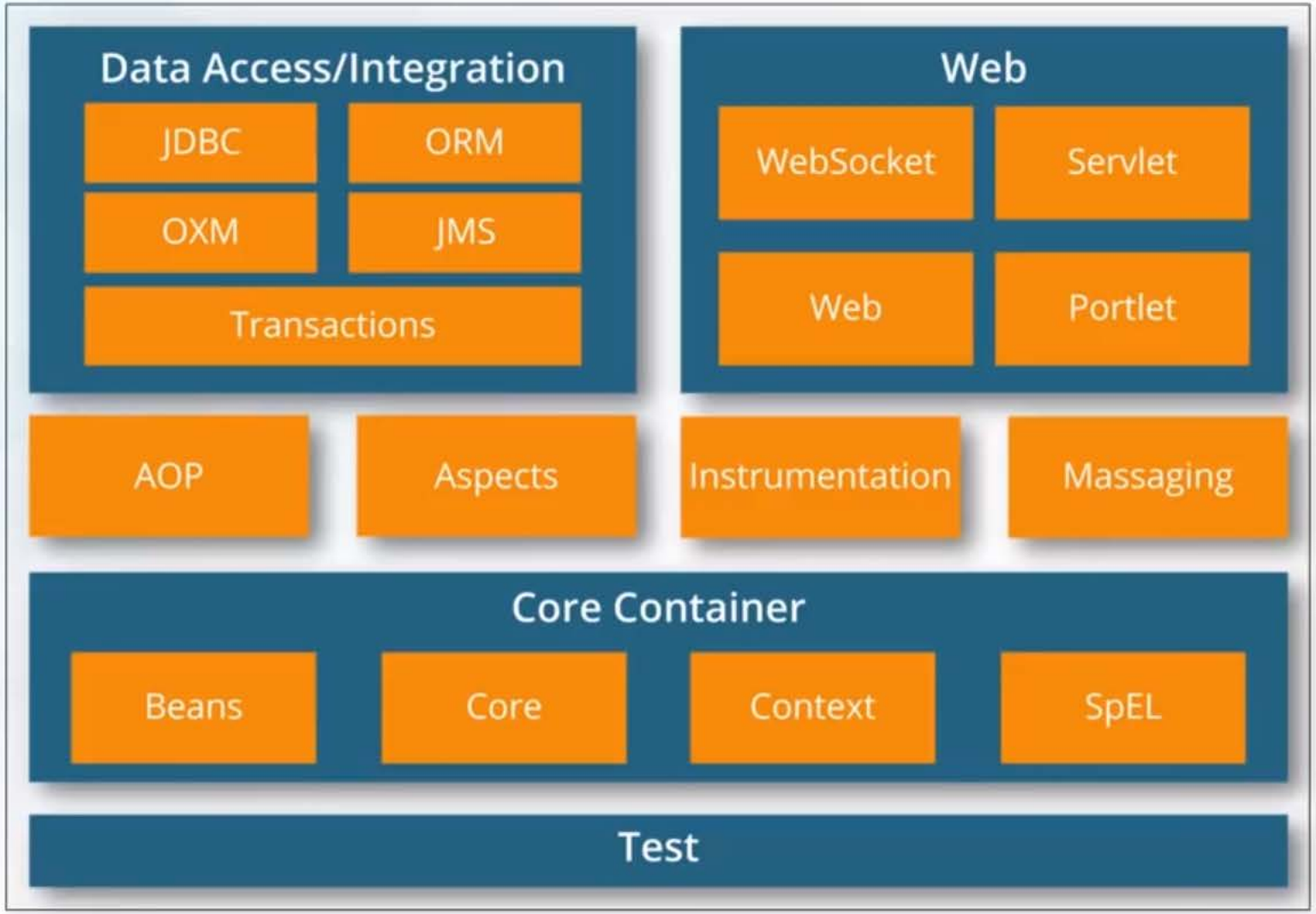
```
Interface Bike
{
   void start();
}
```

Honda | Bajaj | Yamaha

Classes

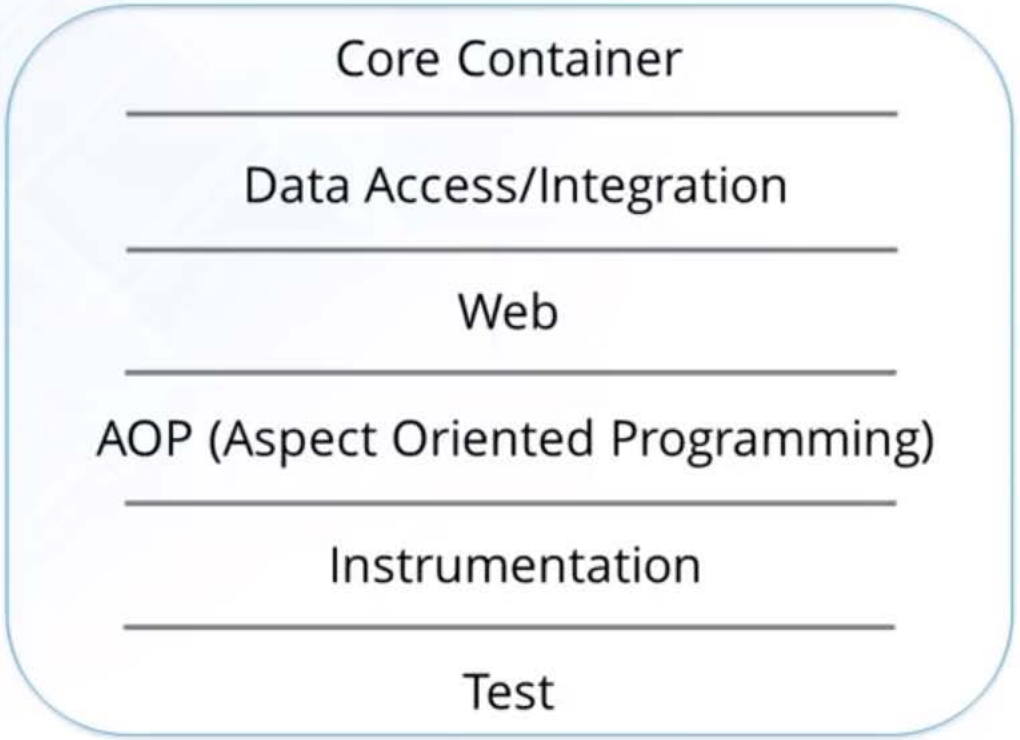
Spring container will inject either Honda object or Bajaj object or Yamaha object into the Rider class by calling setter method

Spring Architecture

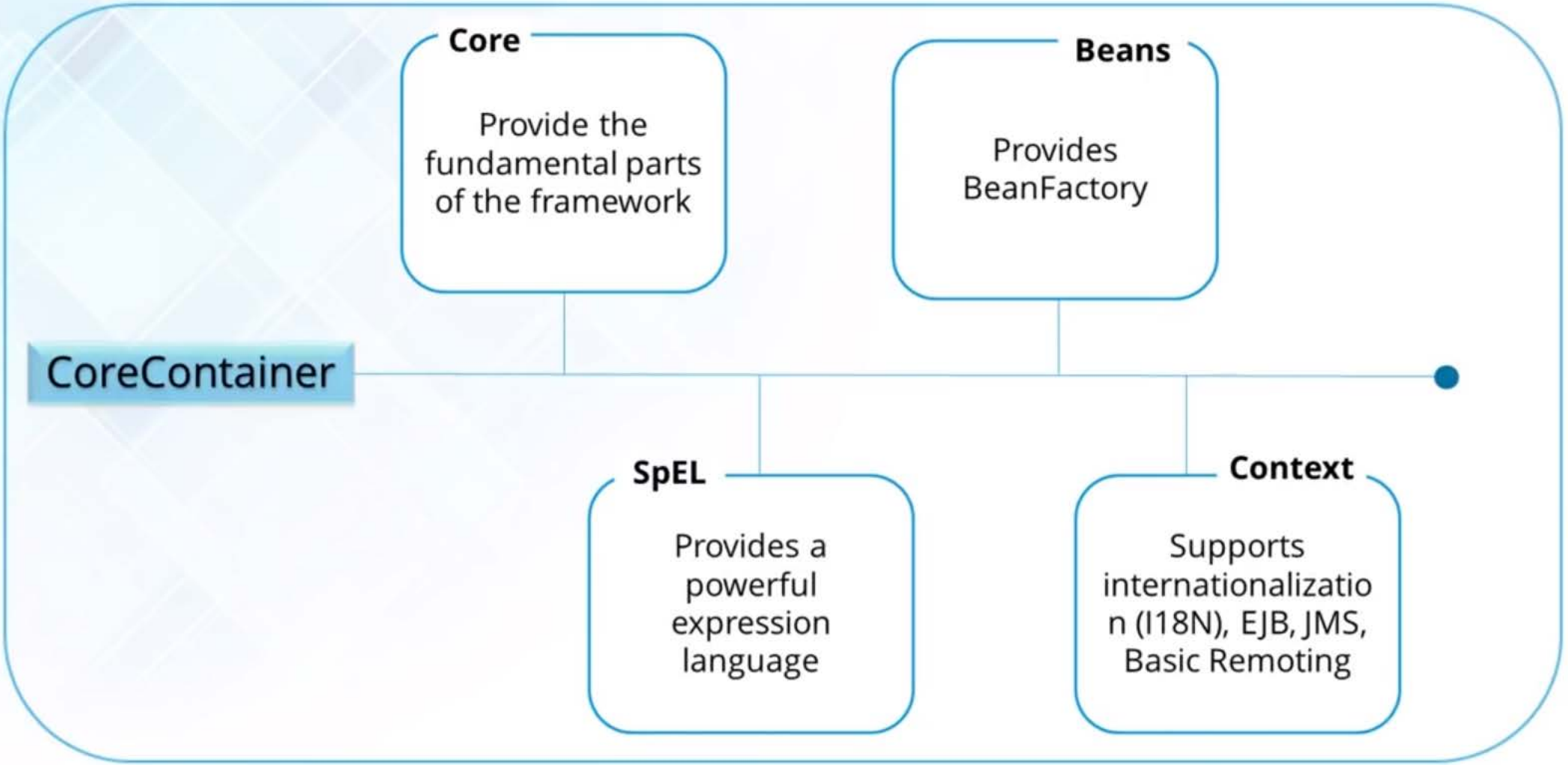


Spring Modules

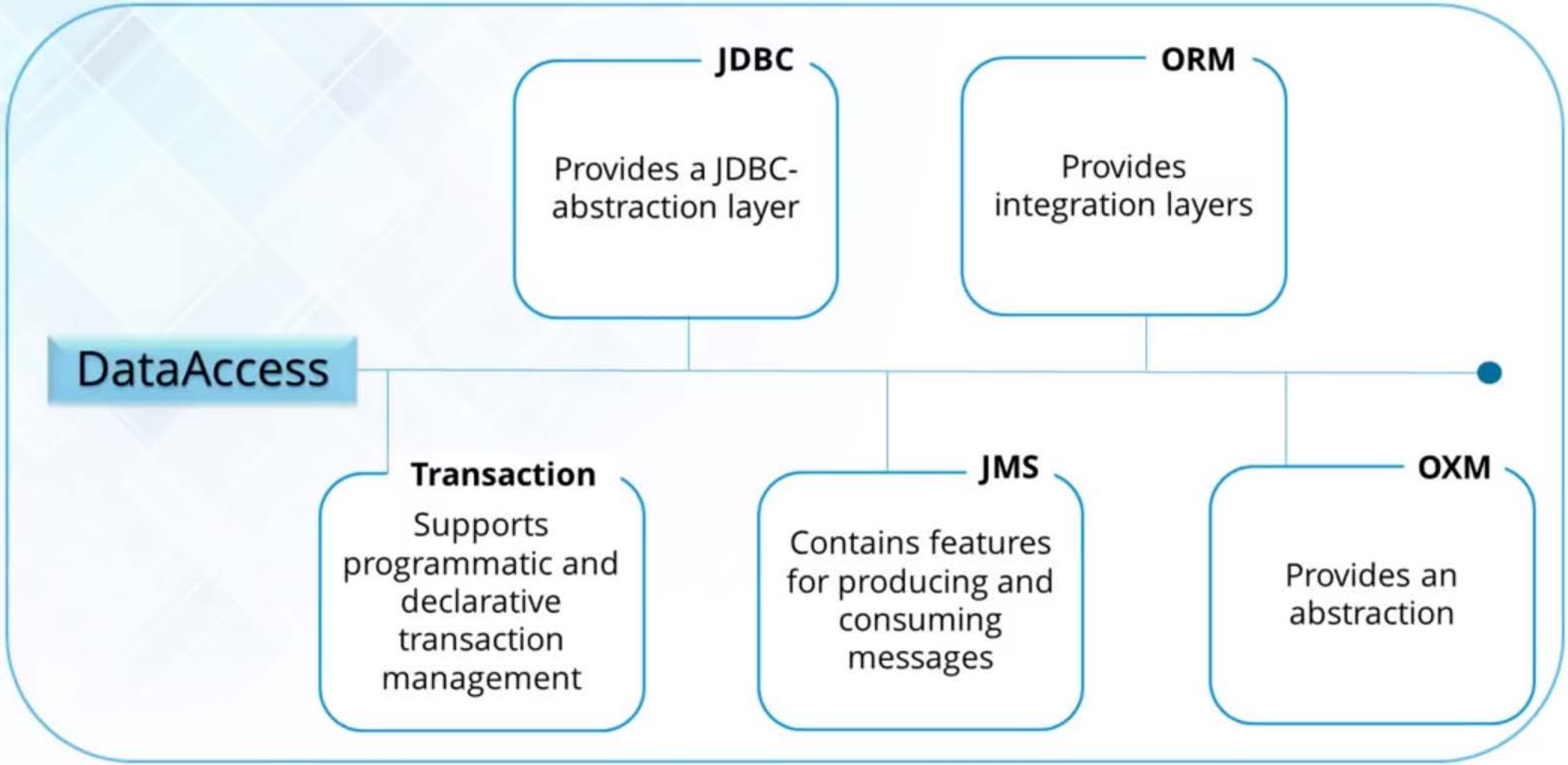
- The Spring Framework contains a lot of features, which are well-organized in about twenty modules.
- These modules can be grouped together based on their primary features into following:



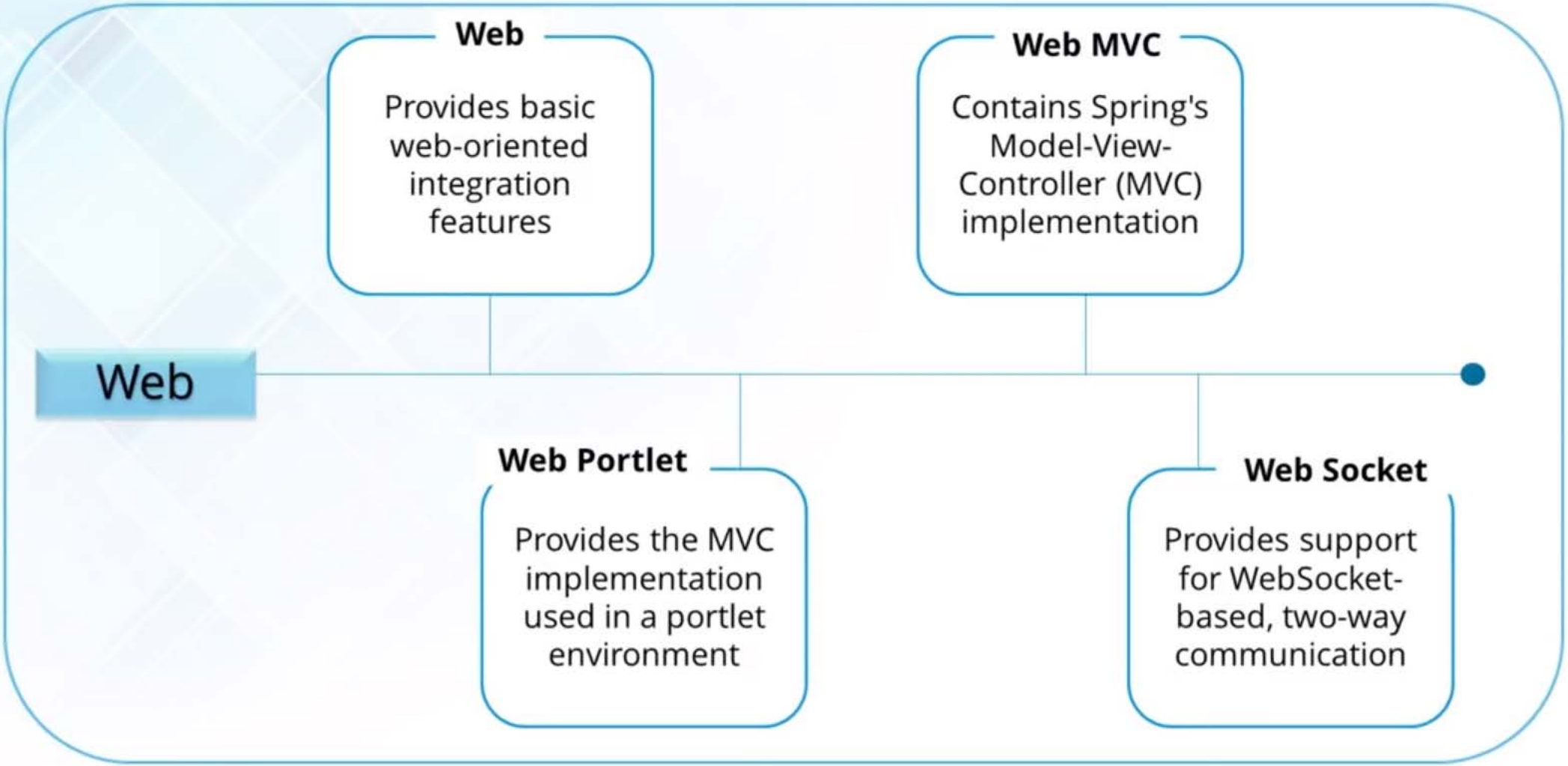
Modules-Core Container



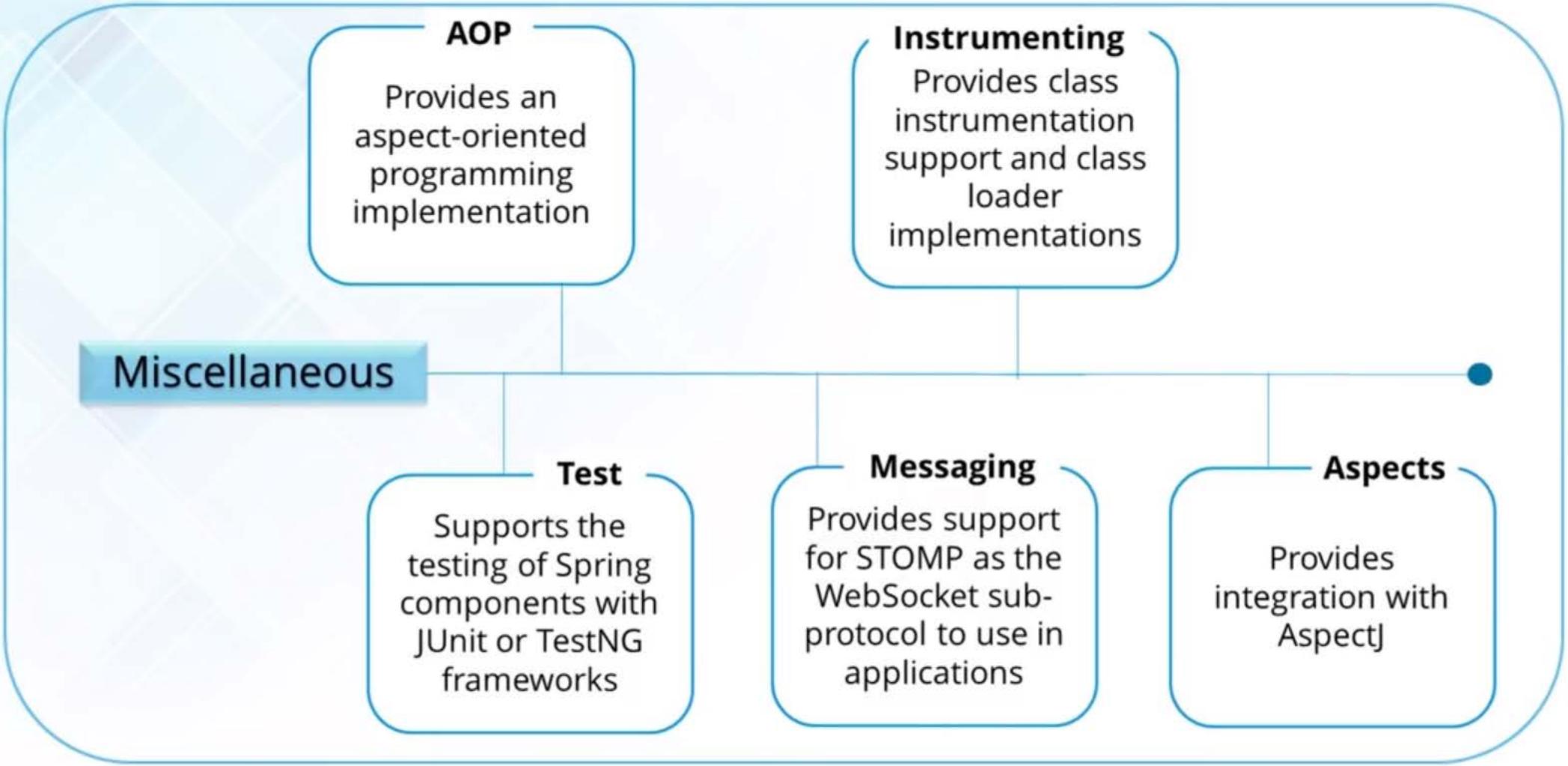
Modules-Data Access

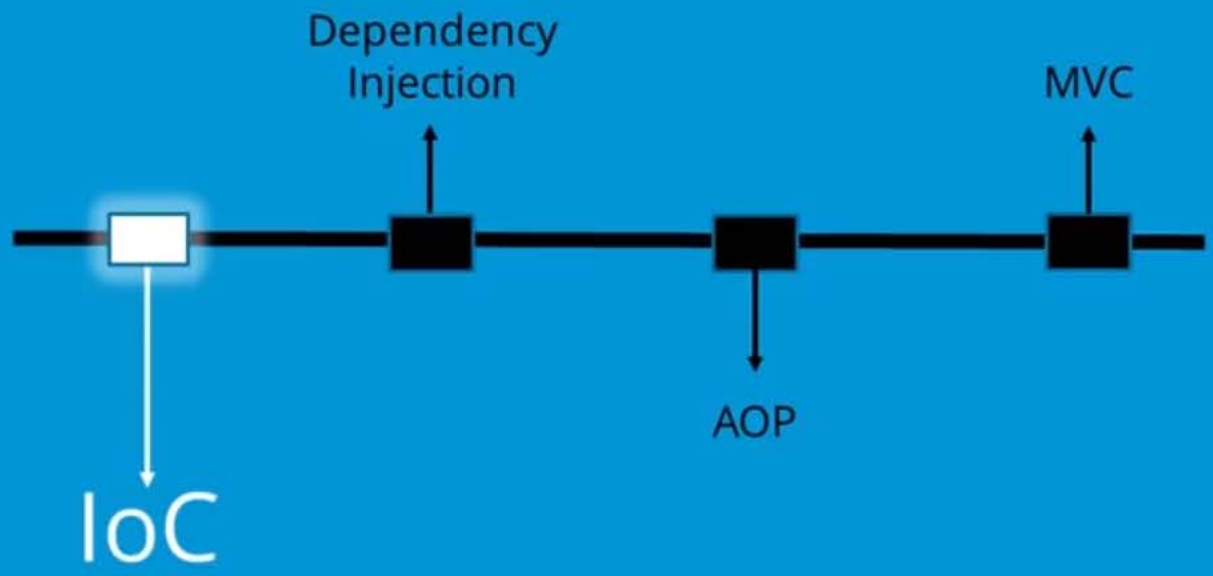


Modules-Web


Web
Web
Provides basic web-oriented integration features
Web MVC
Contains Spring's Model-View-Controller (MVC) implementation
Web Portlet
Provides the MVC implementation used in a portlet environment
Web Socket
Provides support for WebSocket-based, two-way communication

Modules-Miscellaneous




Dependency
Injection
MVC
AOP
IoC

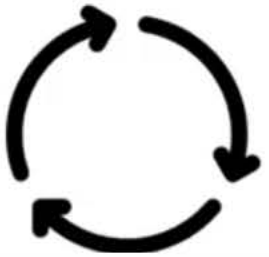
IoC Container Features

Creating The Object

Wiring Them Together

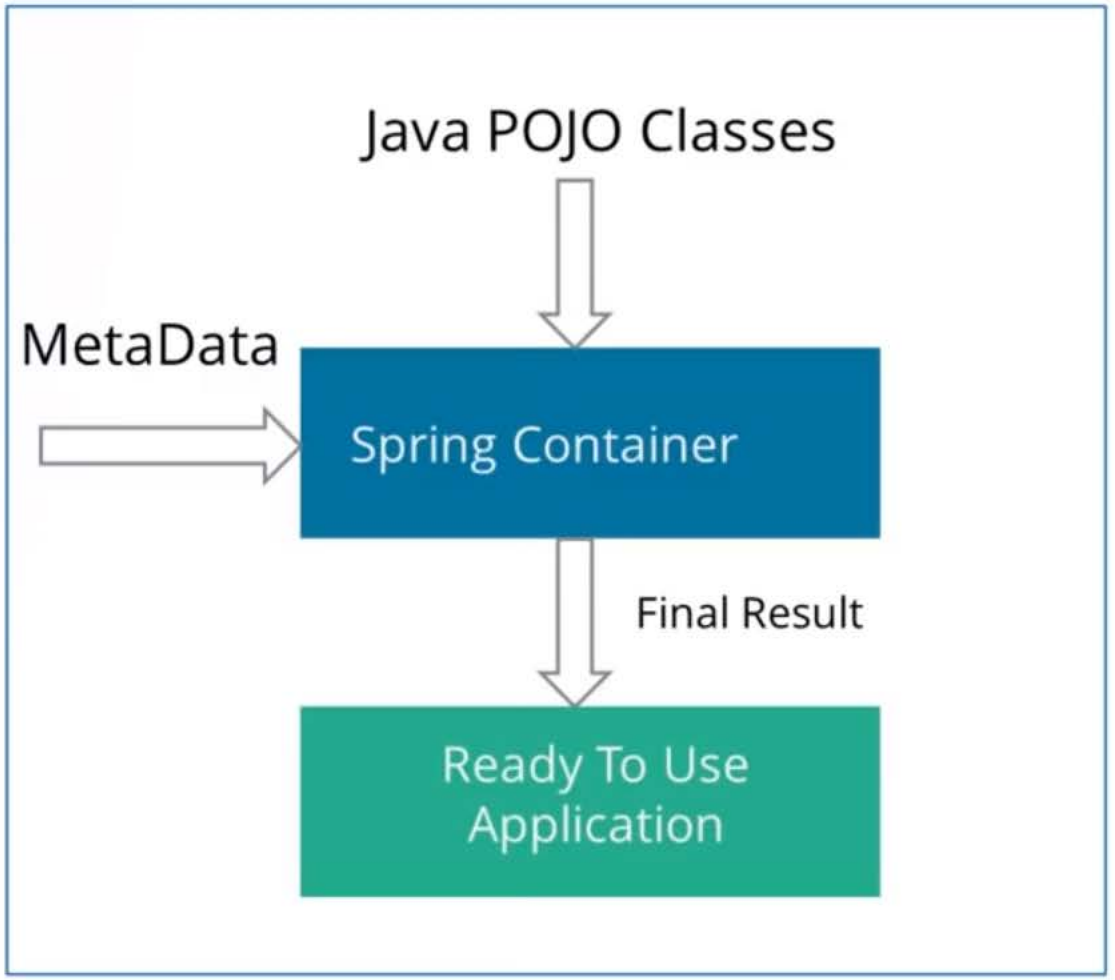
Configuring Them

Managing Their Complete Life Cycle



IoC Container

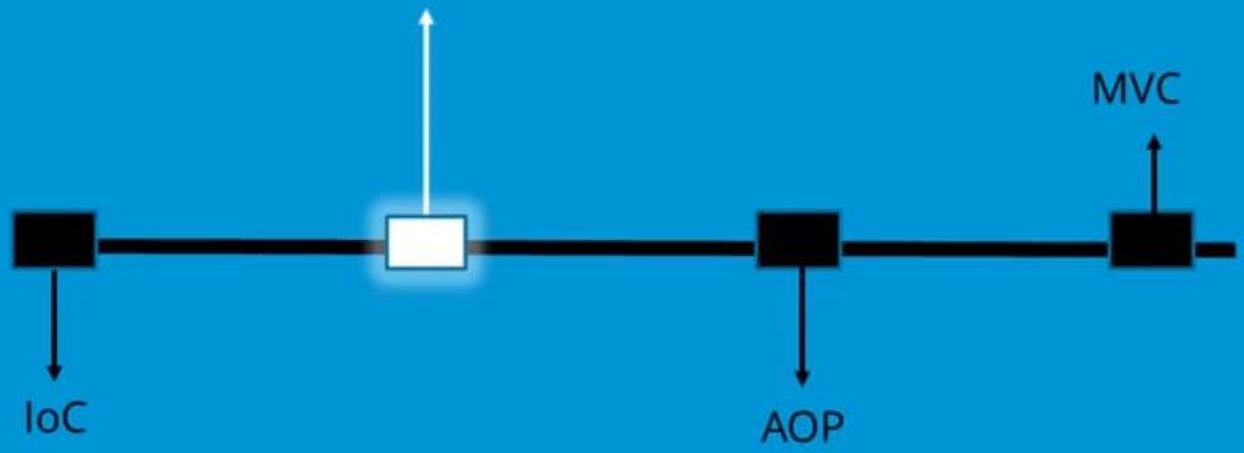
The Spring IoC container by using Java POJO classes and configuration metadata produces a fully configured and executable system or application.



IoC Container

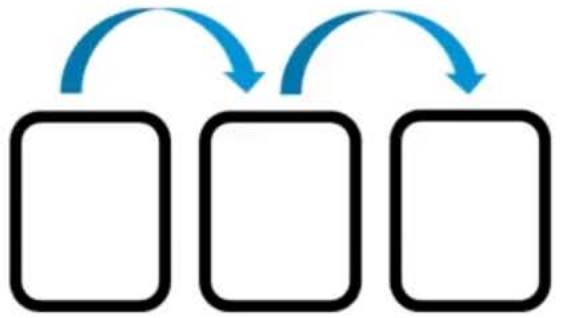
- Configured by loading the XML files or by detecting specific Java annotations on configuration classes.
- Two types of IoC containers :
  - BeanFactory
  - ApplicationContext

Dependency Injection



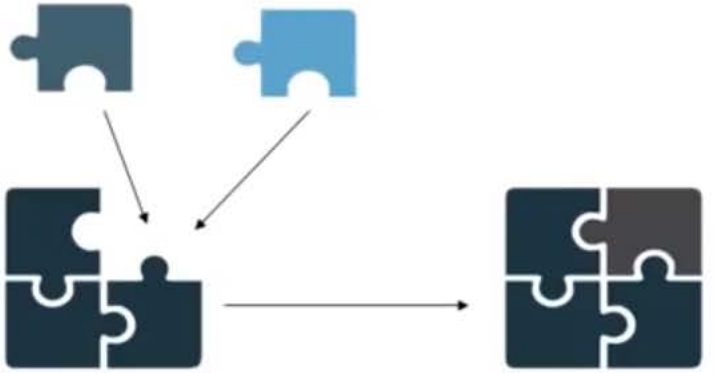
Dependency Injection

Removes the dependency from the programming code



Makes the Application easy to manage and test

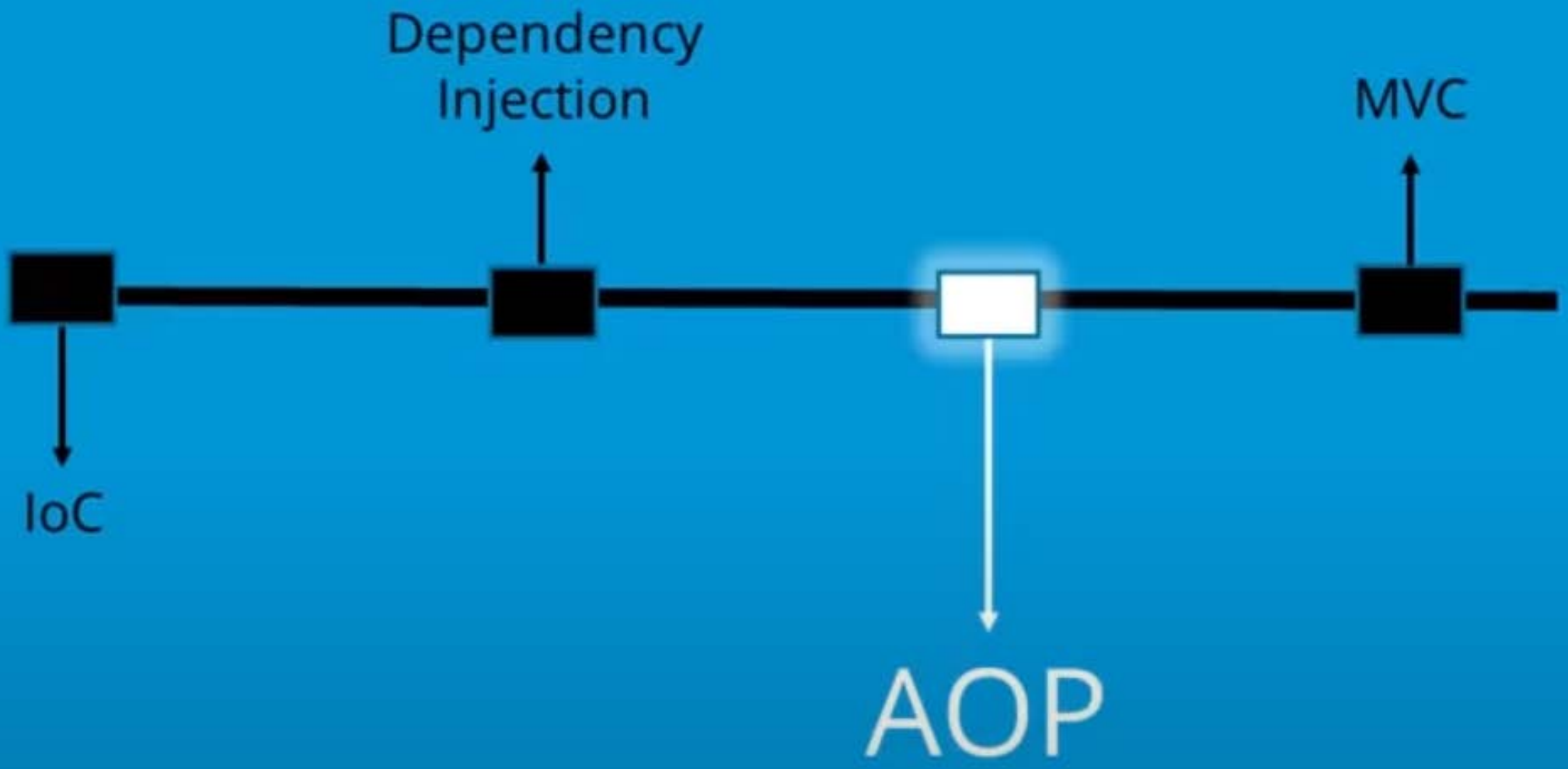
Makes our programming code loosely coupled



Types Of Dependency Injection

Spring framework avails two ways to inject dependency :

By Constructor	1	The **<constructor-arg>** subelement of **<bean>** is used for constructor injection
By Setter method	2	The **<property>** subelement of **<bean>** is used for setter injection

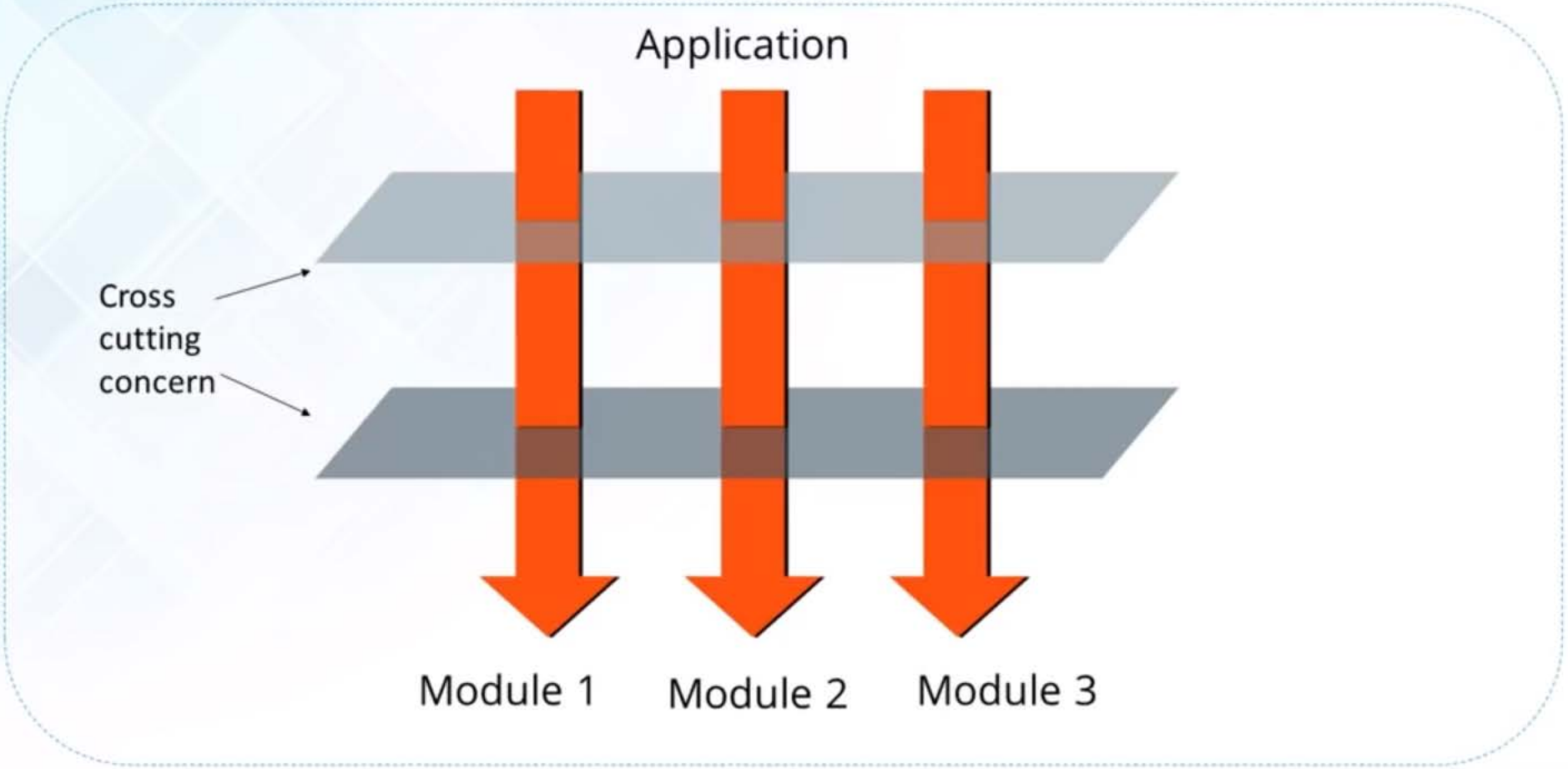

Dependency
Injection
MVC
IoC
AOP

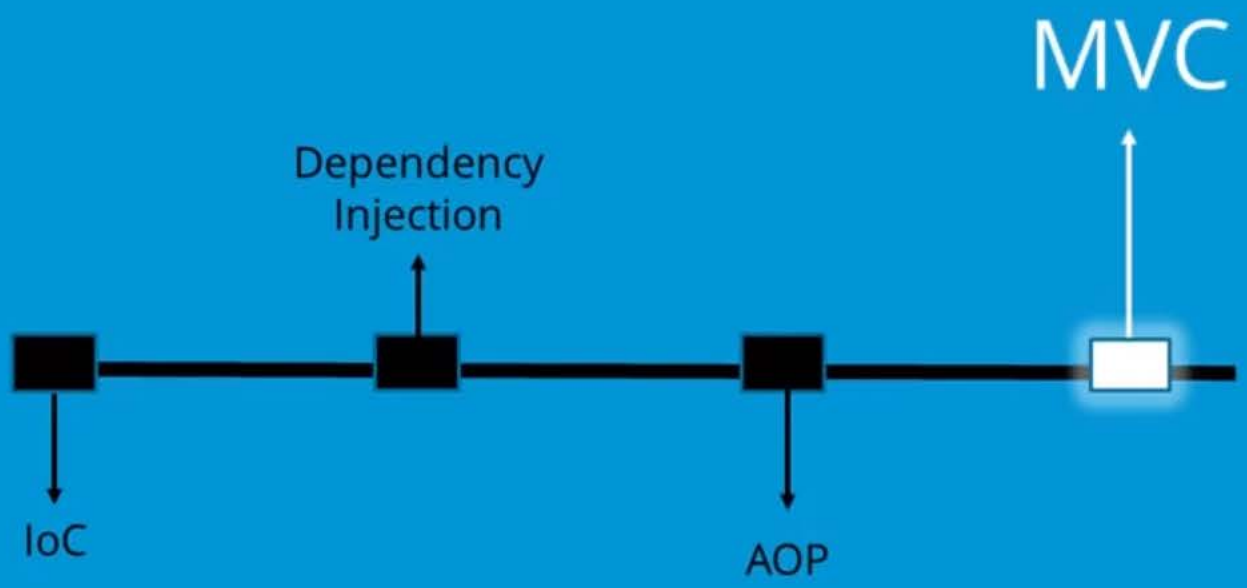
Aspect Oriented Programming

1. Providing modularity with aspect rather than class.
2. AOP breaks the program logic into distinct parts called concerns
3. Increases modularity by **cross-cutting concerns**
4. A **cross-cutting concern** is a concern which can affect the entire application

Aspect Oriented Programming

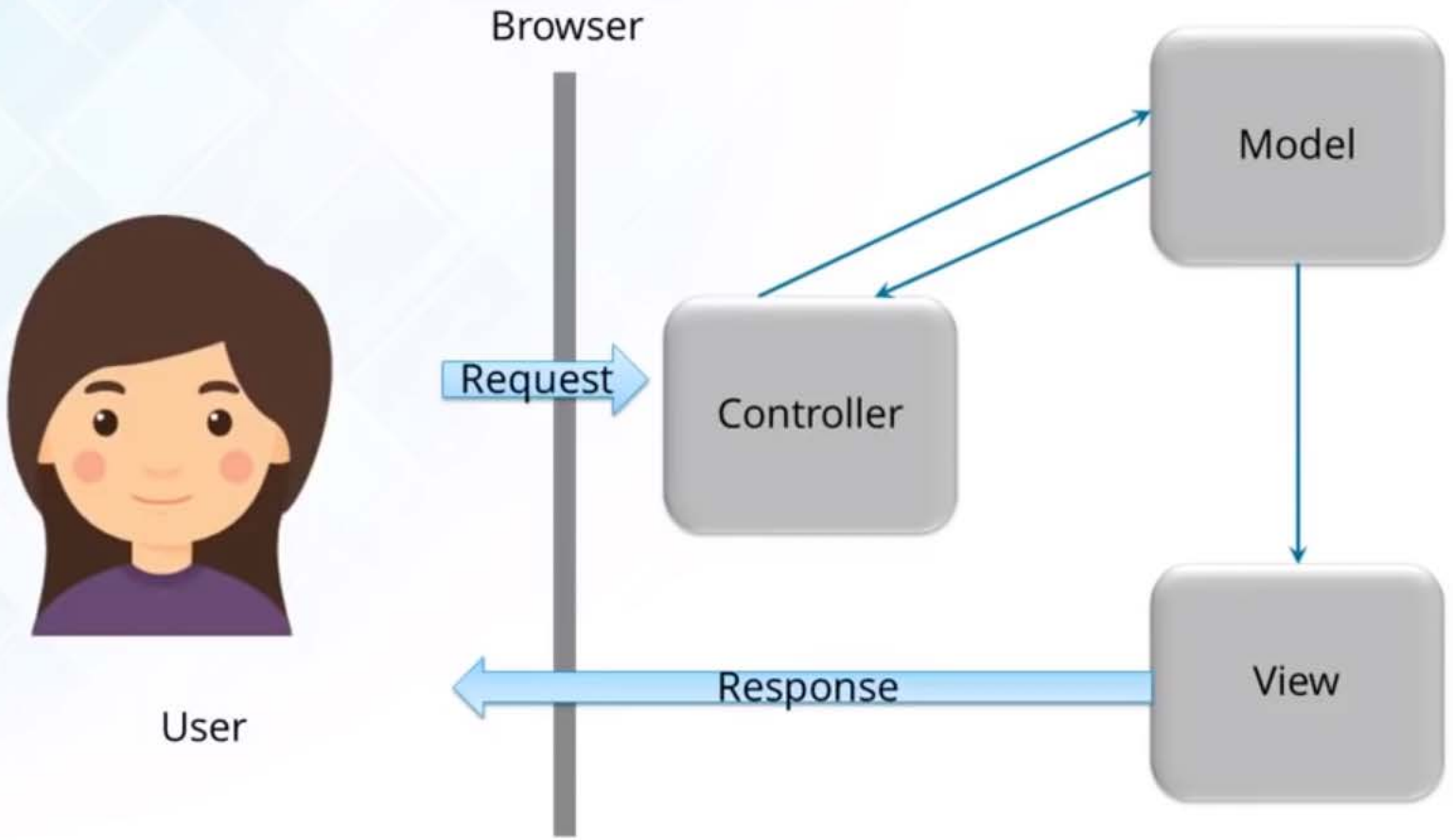
Crosscutting concern is applicable throughout the application and it affects the entire application modules




MVC
Dependency
Injection
IoC
AOP

Model View Controller

User input is interpreted by the controller and are transformed into a model which is represented to the user by the view.



What is Spring Boot?

What is Spring Boot?

3

Used to create stand-alone Java applications.

2

Takes an opinionated view of the Spring platform.

1

Makes it easy to create Spring-powered, production-grade applications.

Features of Spring Boot

Spring CLI

Spring Boot CLI allows you to Groovy for writing Spring boot application and avoids boilerplate code.

Starter Dependency

With the help of this feature, Spring Boot aggregates common dependencies together and eventually improves productivity

Spring Initializer

This is basically a web application, which can create an internal project structure for you.

Auto-configuration

The auto-configuration feature of Spring Boot helps in loading the default configurations according to the project you are working on

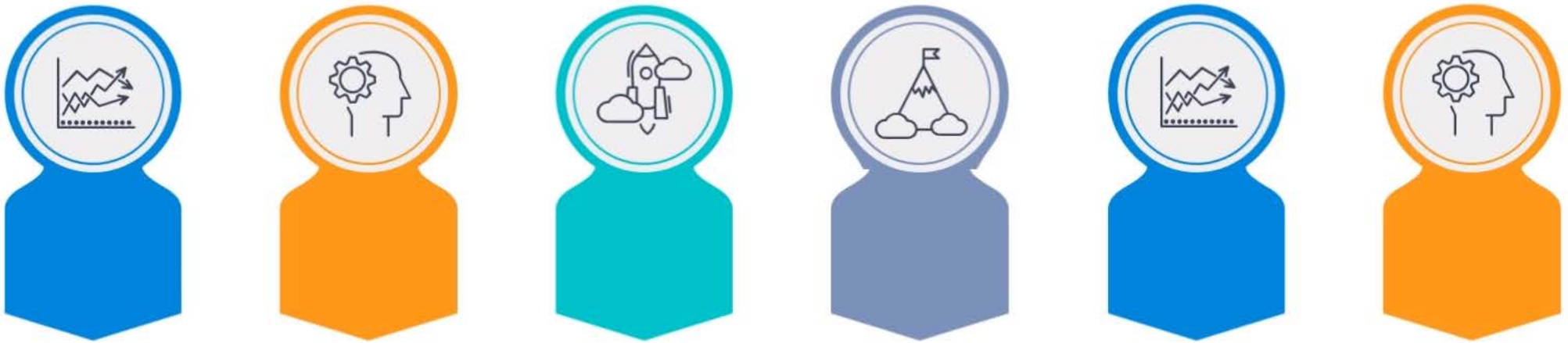
Spring Actuator

This feature provides help while running Spring Boot applications.

Logging and Security

This feature of Spring Boot, ensures that all the applications made using Spring Boot are properly secured without any hassle.

Why Do We Need Spring Boot



Stability

Based on JVM

Connectivity

Cloud-Native

Flexibility

Open Source

Market Trends of Spring Boot

Location

United States

Popular Jobs	Average Salary	Salary Distribution
Java Developer 6,914 salaries reported Java Developer Jobs	**$102,521** per year	$36,000 — $219,000
Senior Java Developer 3,413 salaries reported Senior Java Developer Jobs	**$119,529** per year	$36,000 — $219,000
Full Stack Developer 10,120 salaries reported Full Stack Developer Jobs	**$109,674** per year	$36,000 — $219,000

Spring vs Spring Boot

01 Takes time to have Spring application up and running

02 Manages life cycle of Java

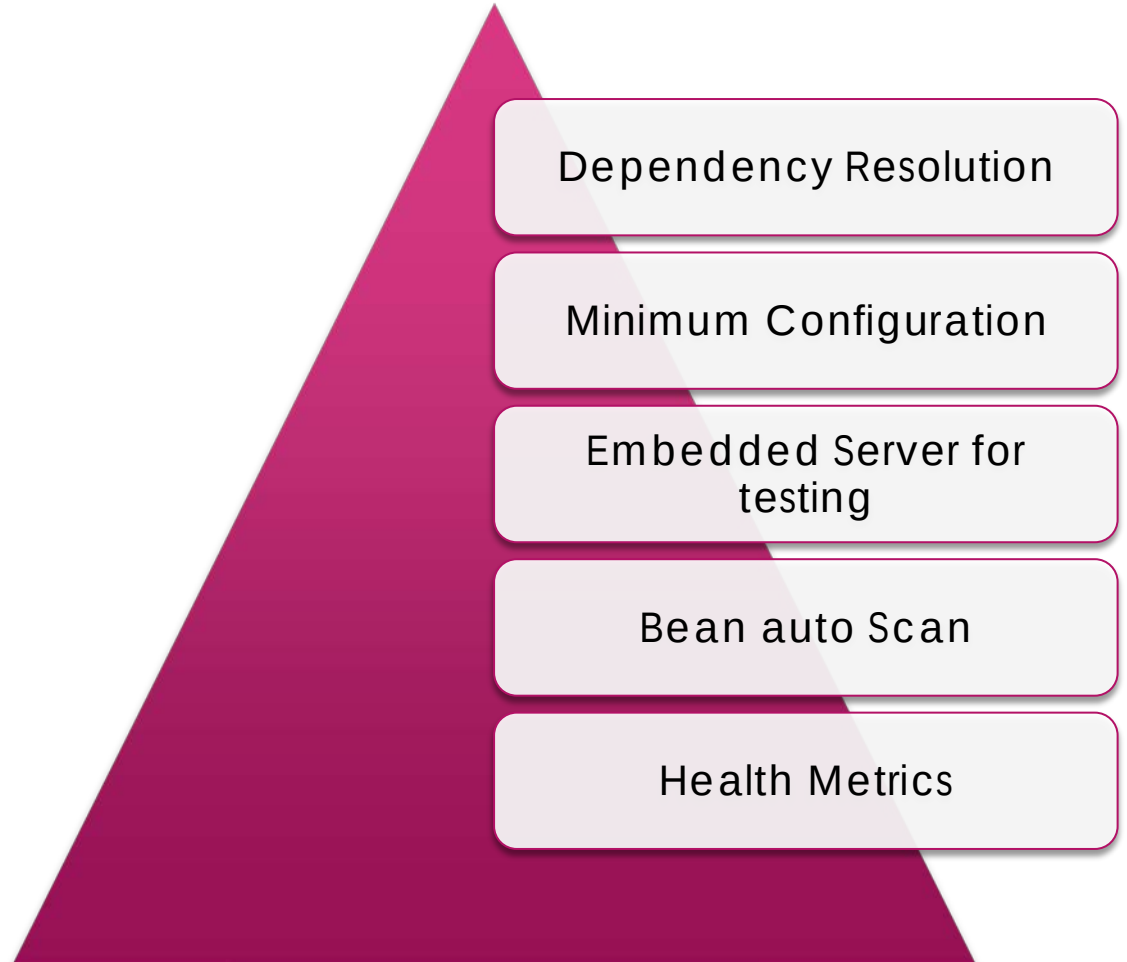
03 Dependency Injection Framework

01 Shortest way to run Spring application

02 Need not worry about configuring a data source

03 Pre-configured set of frameworks/technologies

Benefits of Spring Boot Over Spring



- Dependency Resolution
- Minimum Configuration
- Embedded Server for testing
- Bean auto Scan
- Health Metrics

Install & Set Up Spring Boot

There are 2 ways you can use Spring Boot

01 Spring Boot CLI

Spring Tool Suite (STS) 02

Install & Set Up Spring Boot

System Requirements

Spring Boot 2.1.7.Release requires

- ✓ Java 8 +
- ✓ Spring Framework 5.1.9 +

Explicit Build Support

- ✓ Maven 3.3+
- ✓ Gradle 4.4+

Servlet Container Support

- ✓ Tomcat 9.0 – Servlet Version 4.0
- ✓ Jetty 9.4 – Servlet Version 3.1
- ✓ Undertow 2.0 – Servlet Version 4.0

Install and Set Up Spring Boot CLI

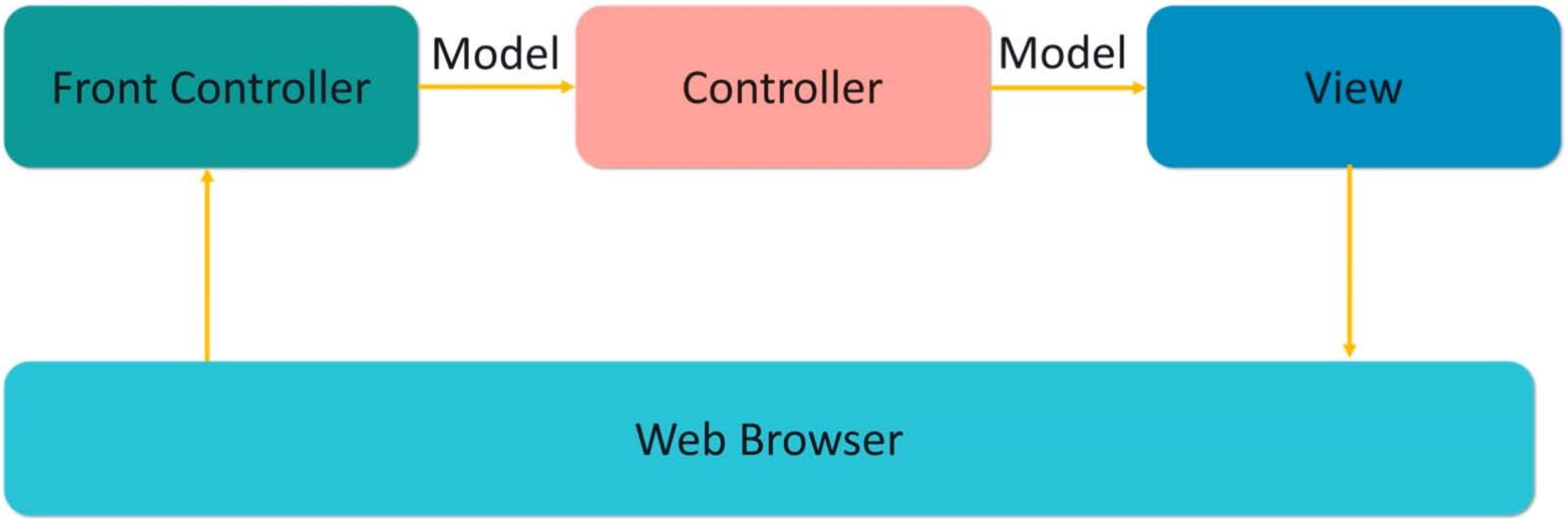
Model View Controller

What is MVC?

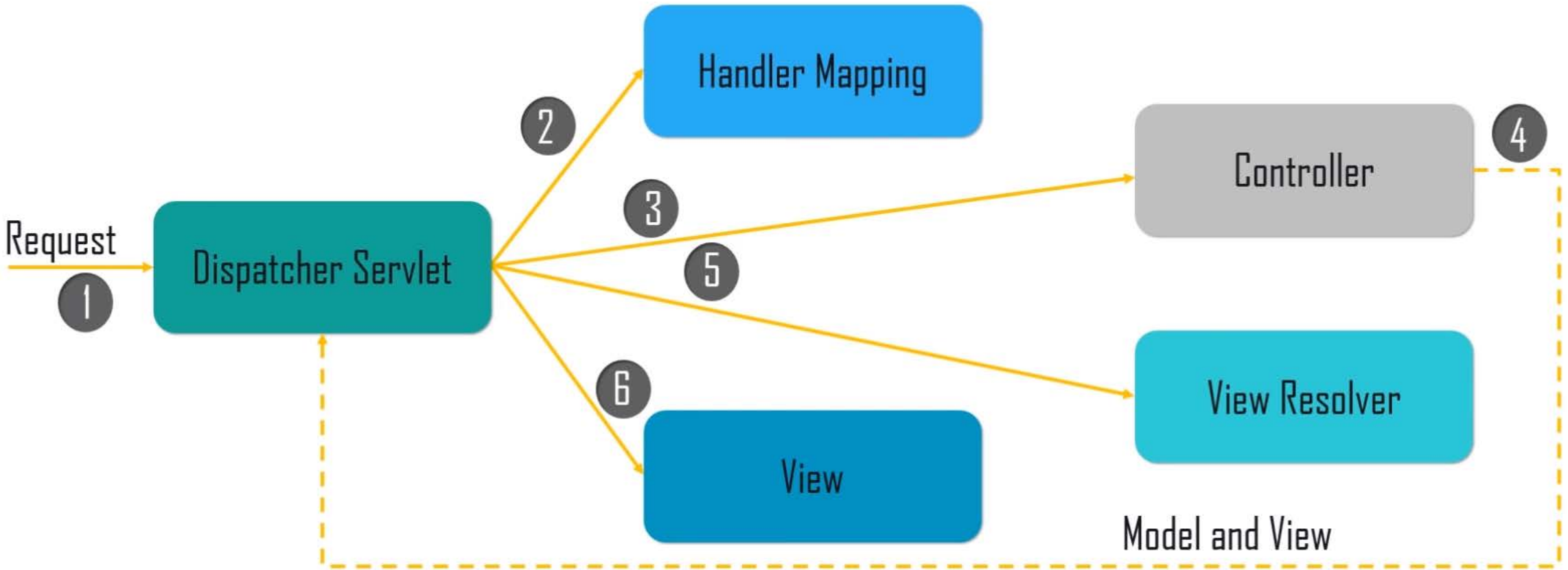
It is a **Java framework** which is used to build web applications. It follows the **Model-View-Controller** design pattern.

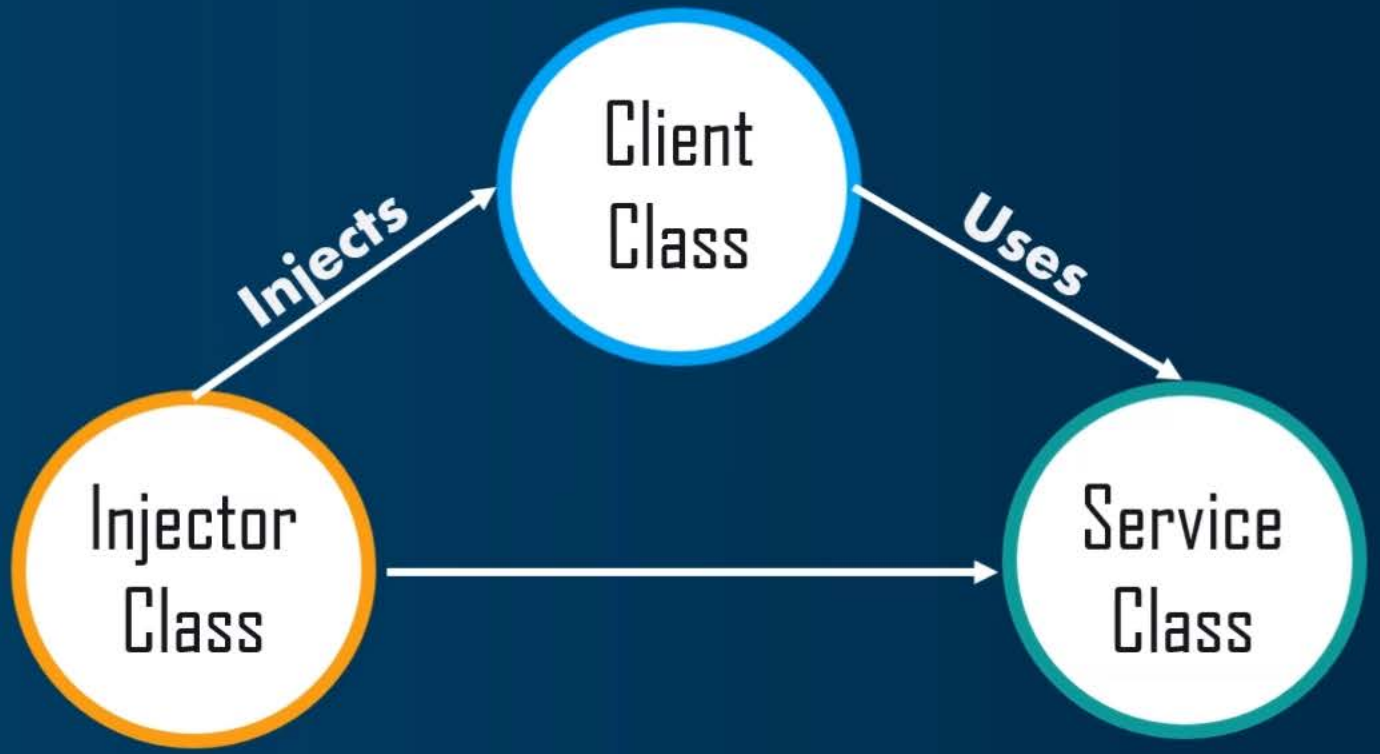
Not just that, it also implements all the basic features of a core **Spring framework** like Dependency Injection.

Model View Controller



Model View Controller Workflow





What is Dependency Injection(DI)?

Dependency Injection uses three types of classes. The injector class creates an object of the service class. Then, the injector class injects the object to a client object.

Inversion of Control

A class should not configure its dependencies statically but should be configured by some other class from outside.

A class should concentrate on fulfilling its responsibilities like the flow of an application, and not on creating objects

Concept Behind Dependency Injection

Types of Dependency Injection

Constructor	Setter	Interface
Dependencies are provided through a class constructor	Injector method injects the dependency to the setter method exposed by the client.	Injector uses Interface to provide the dependency to the client class.

Benefits of DI

Enables an easy way to interconnect the components

Application can be easily extended

Unit Testing is made much easier

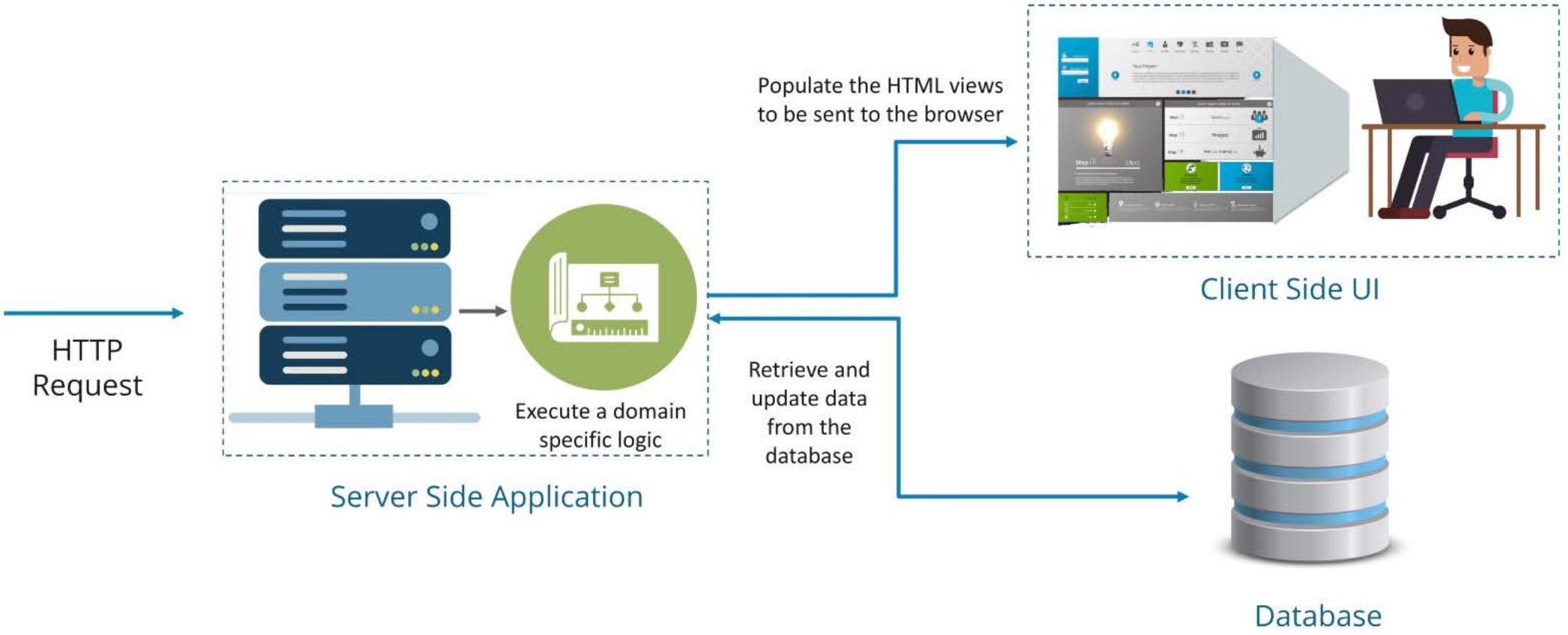
Reduction of Boiler plate code

“MICRO SERVICES”

Why Microservices?

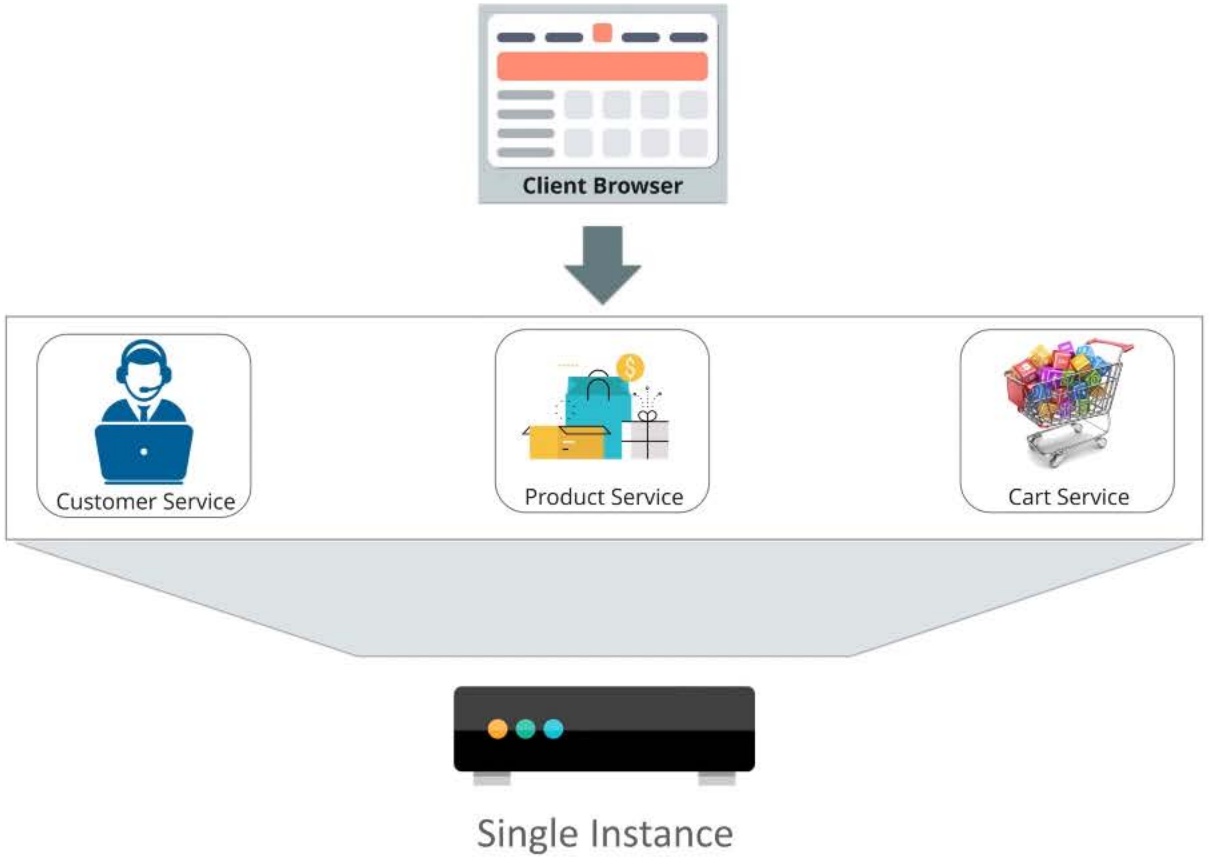
Before Microservices – Monolithic Architecture

Monolithic Architecture is like a big container wherein all the software components of an application are assembled together and tightly packaged

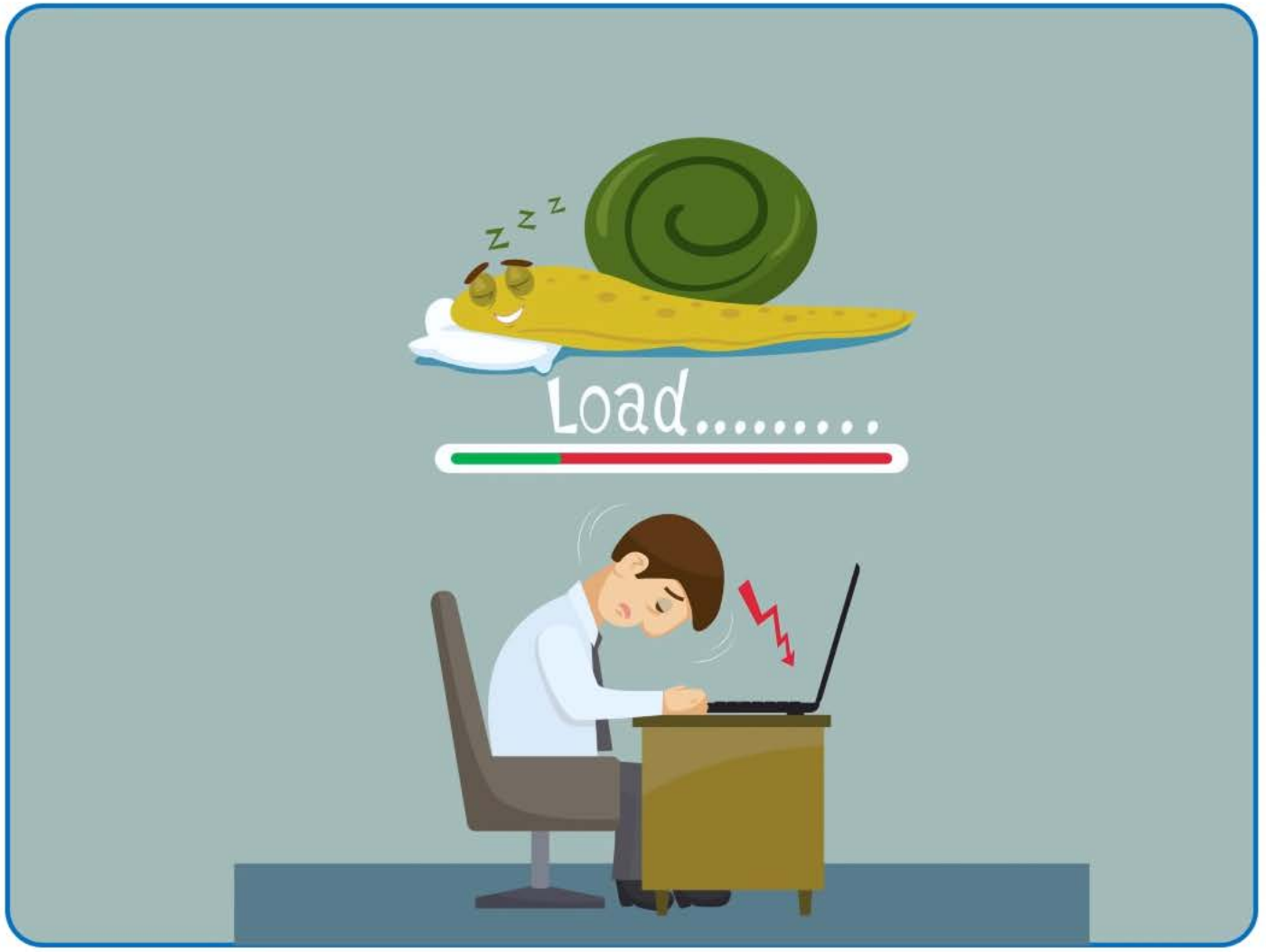


Monolithic Architecture - Example

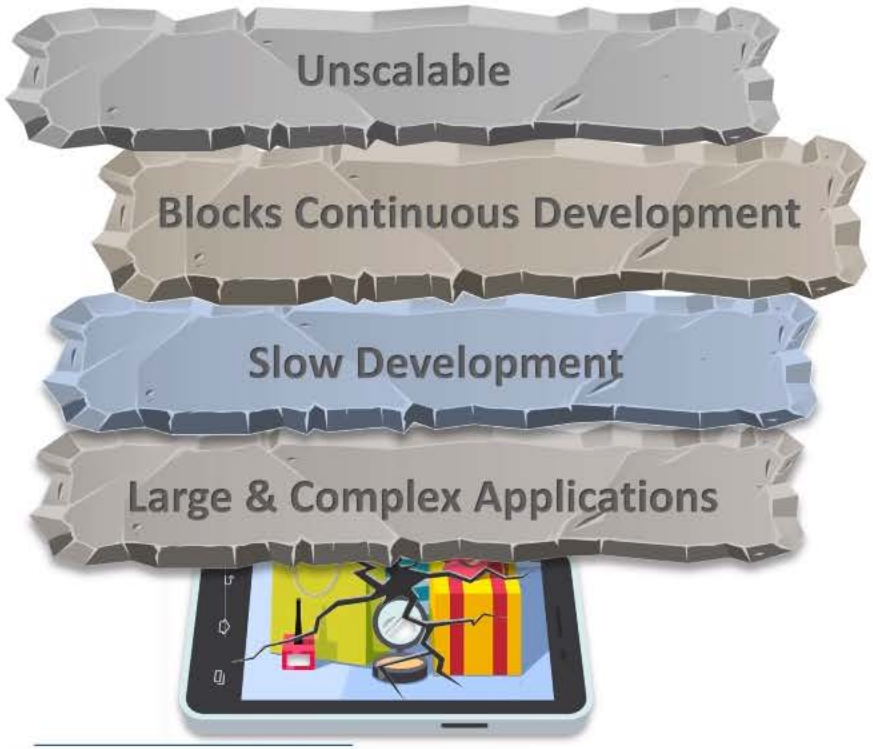
Let’s take a classic use case of an E-Commerce Application

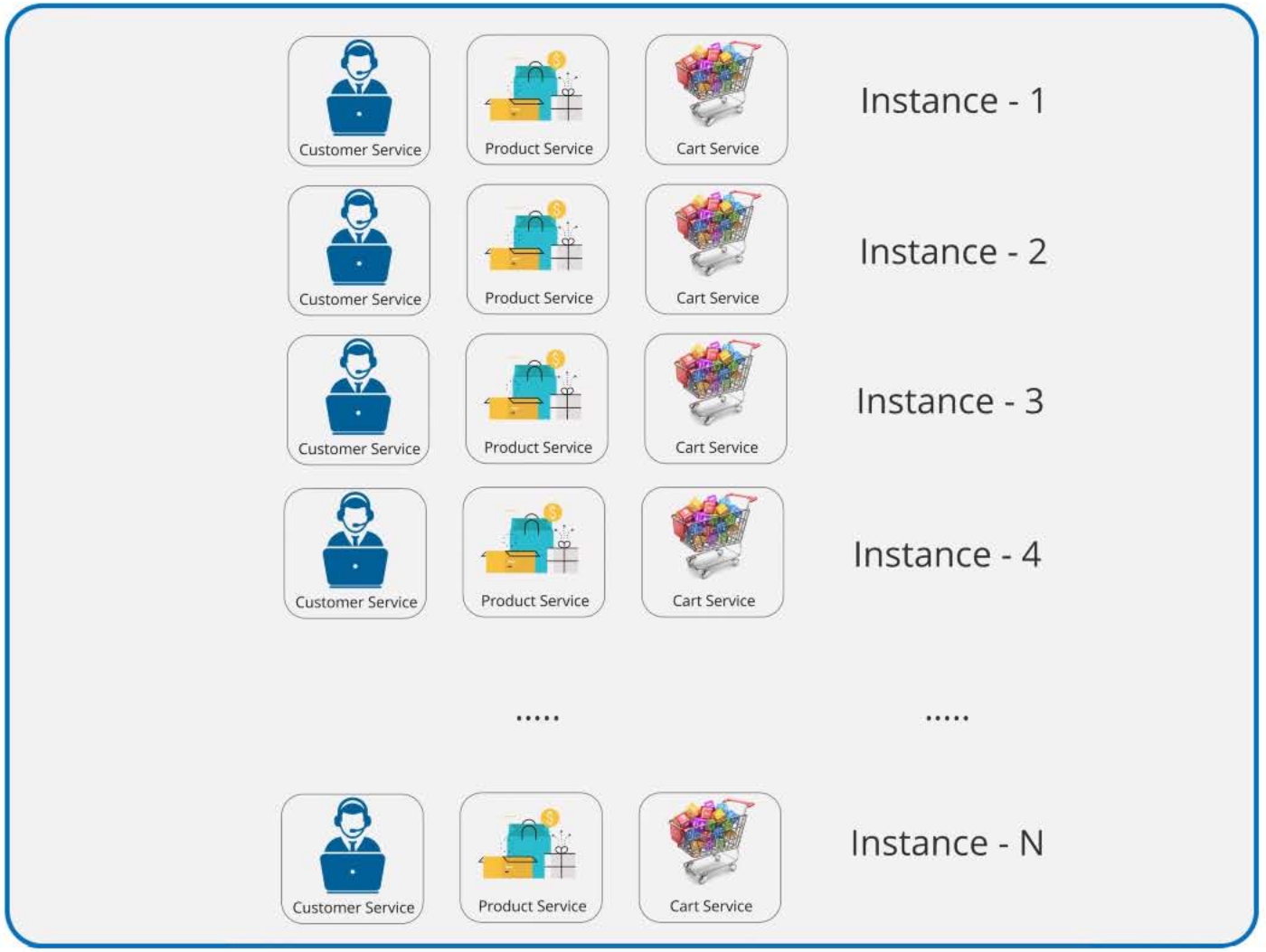


Monolithic Architecture - Challenges

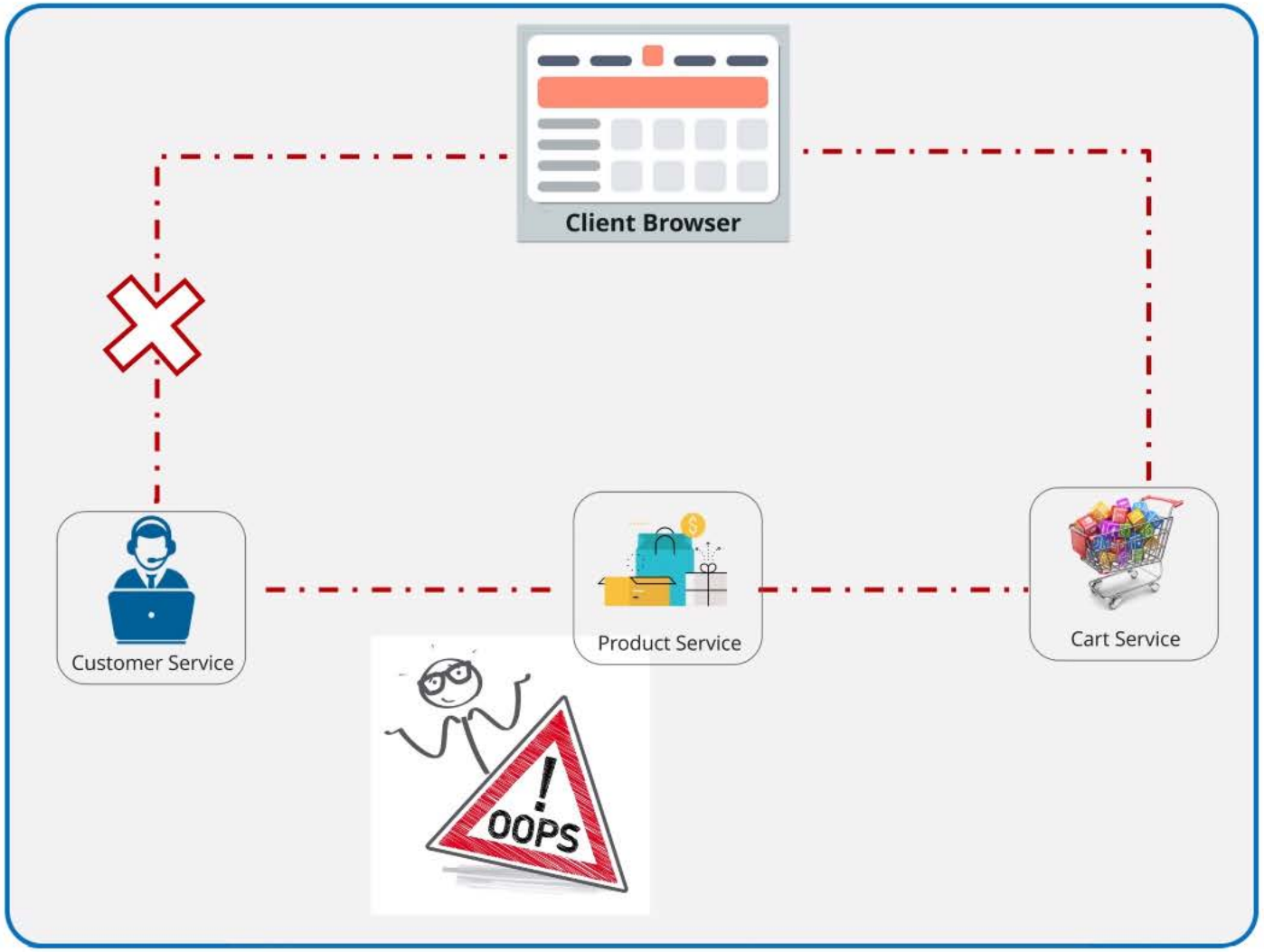


Monolithic Architecture - Challenges

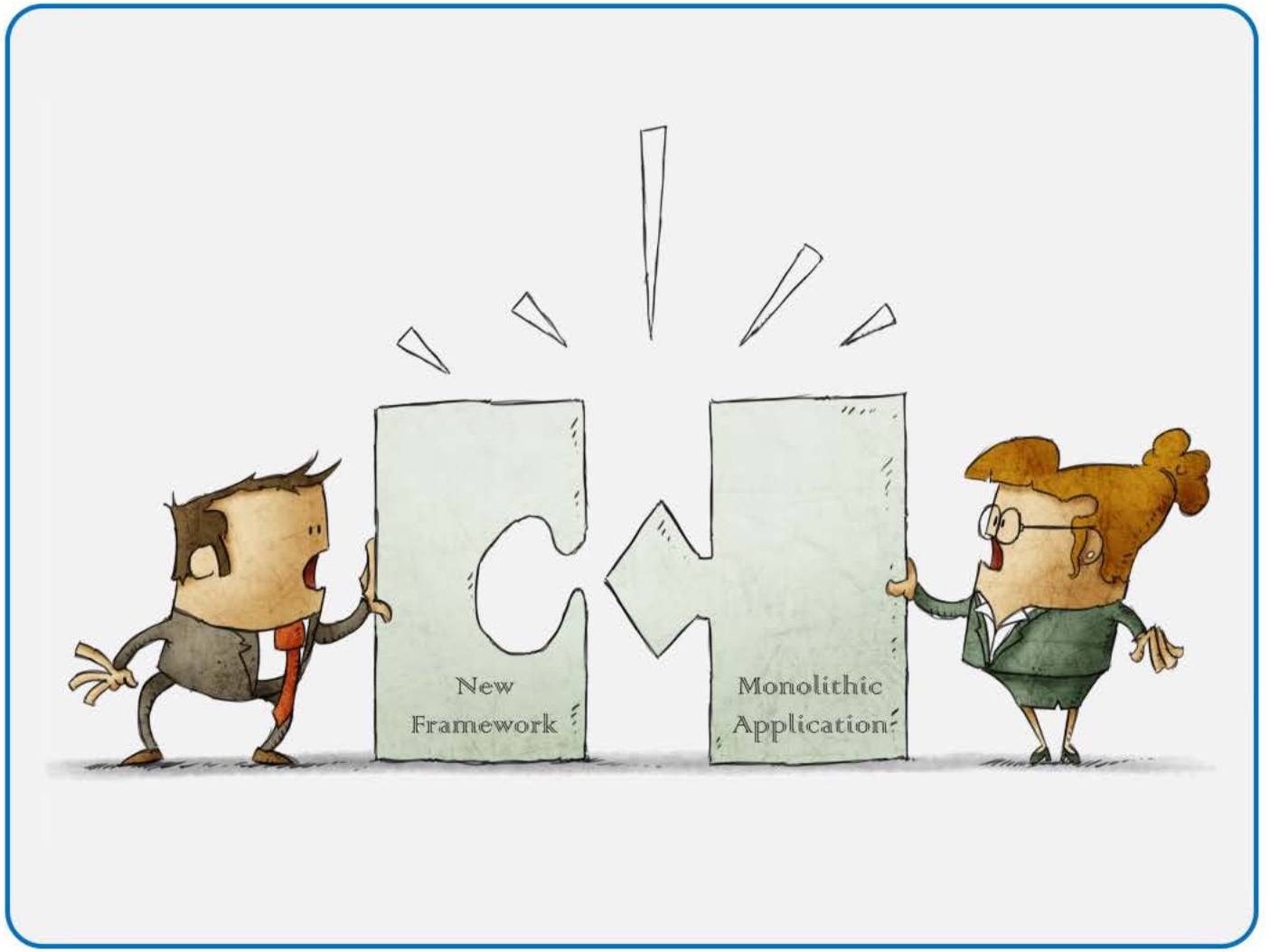




Monolithic Architecture - Challenges



Monolithic Architecture - Challenges

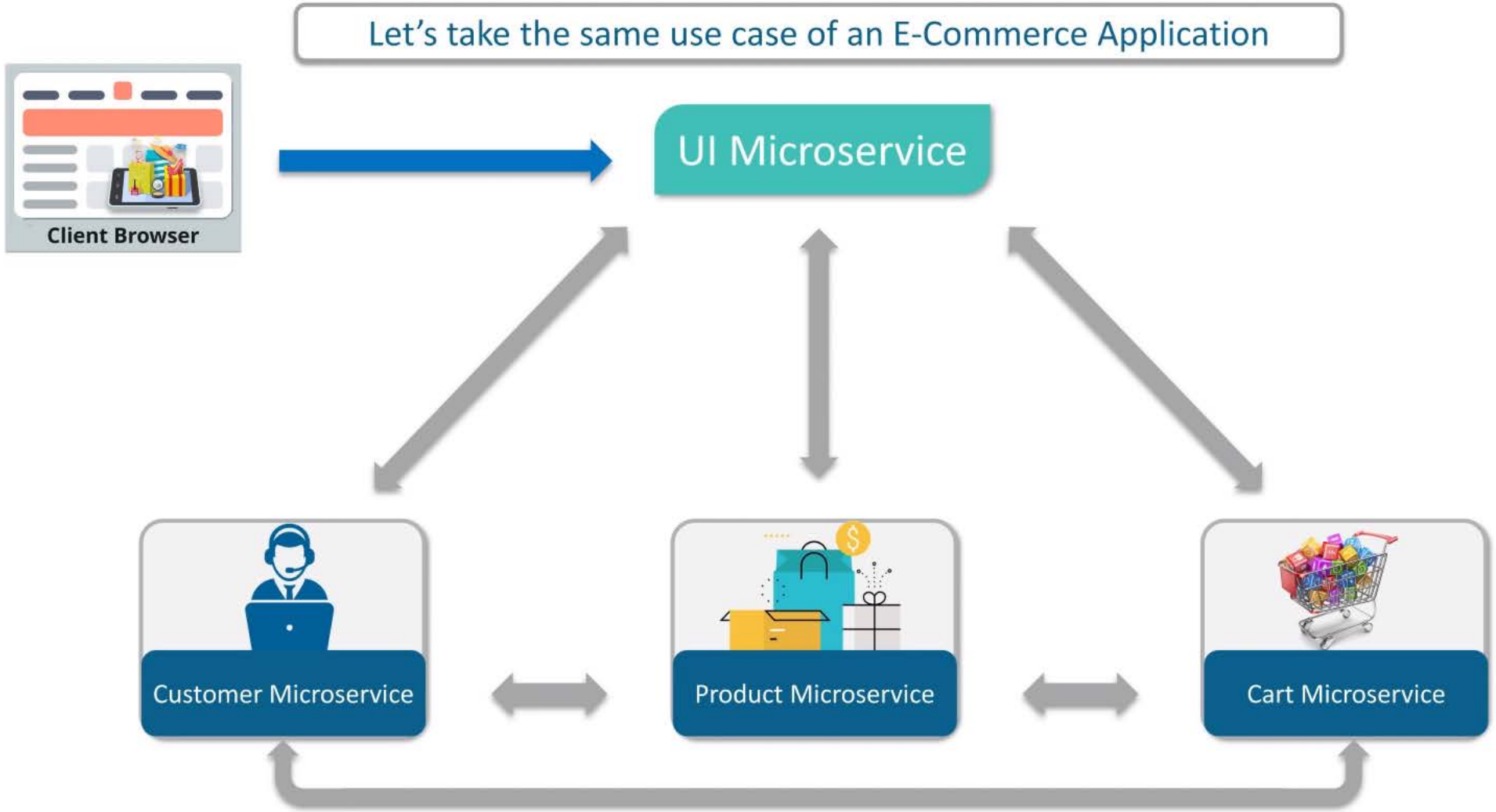


What Is Microservice Architecture?

Microservices, aka *Microservice Architecture*, is an **architectural style** that structures an application as a **collection of small autonomous services**, modelled around a **Business Domain**

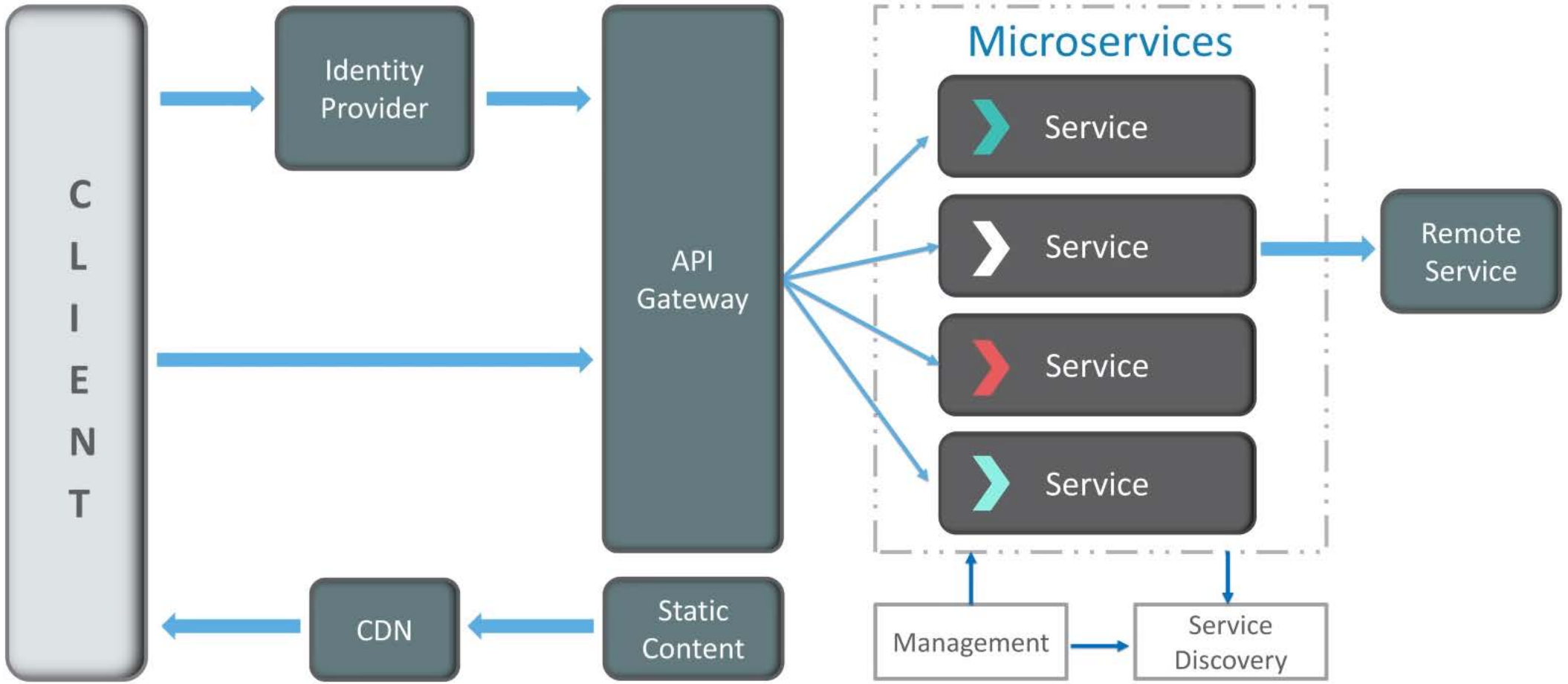
In Microservice Architecture, each service is **self-contained** and implements a **single Business capability**

Microservice Architecture - Example

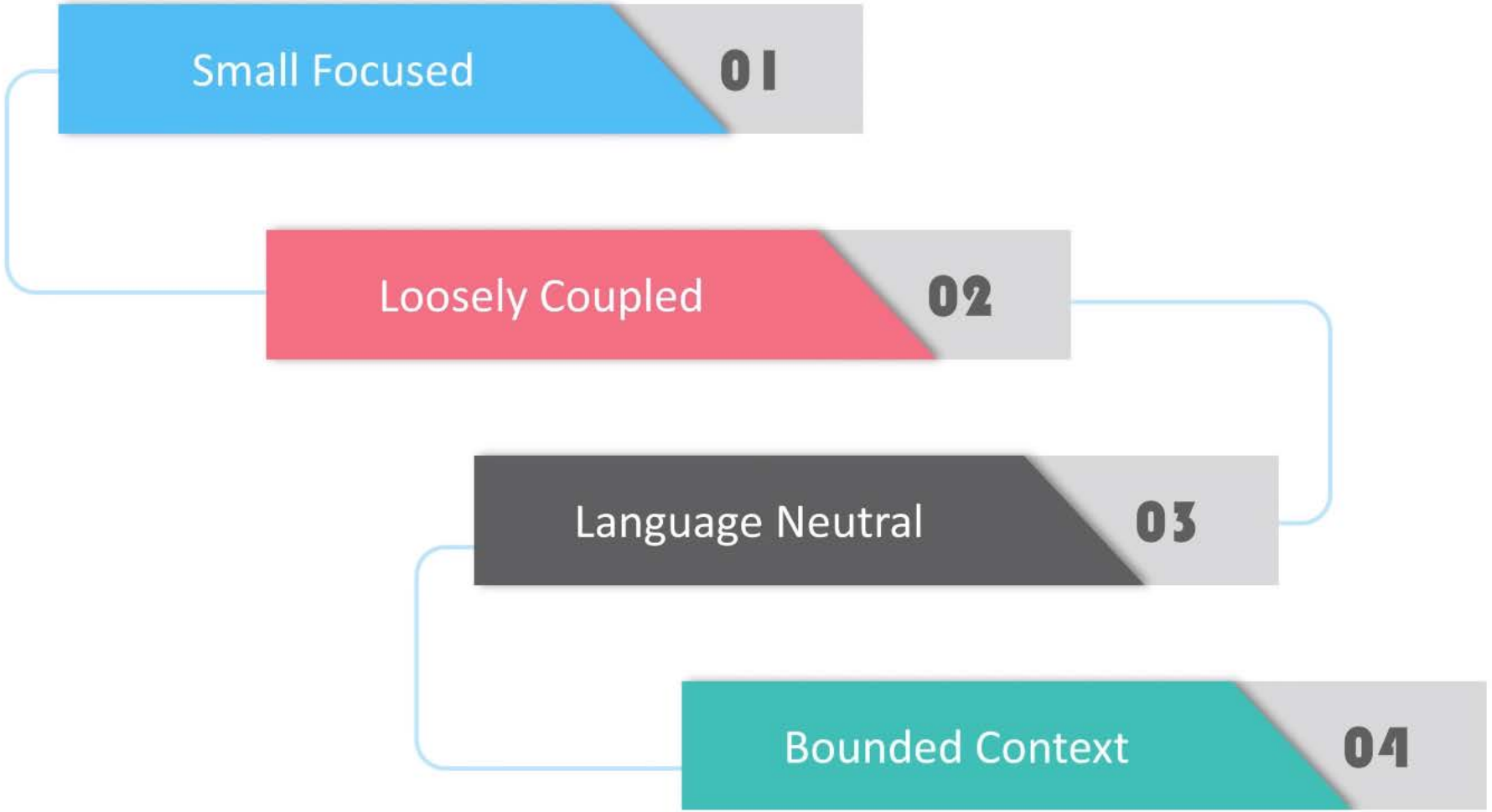


Microservice Architecture

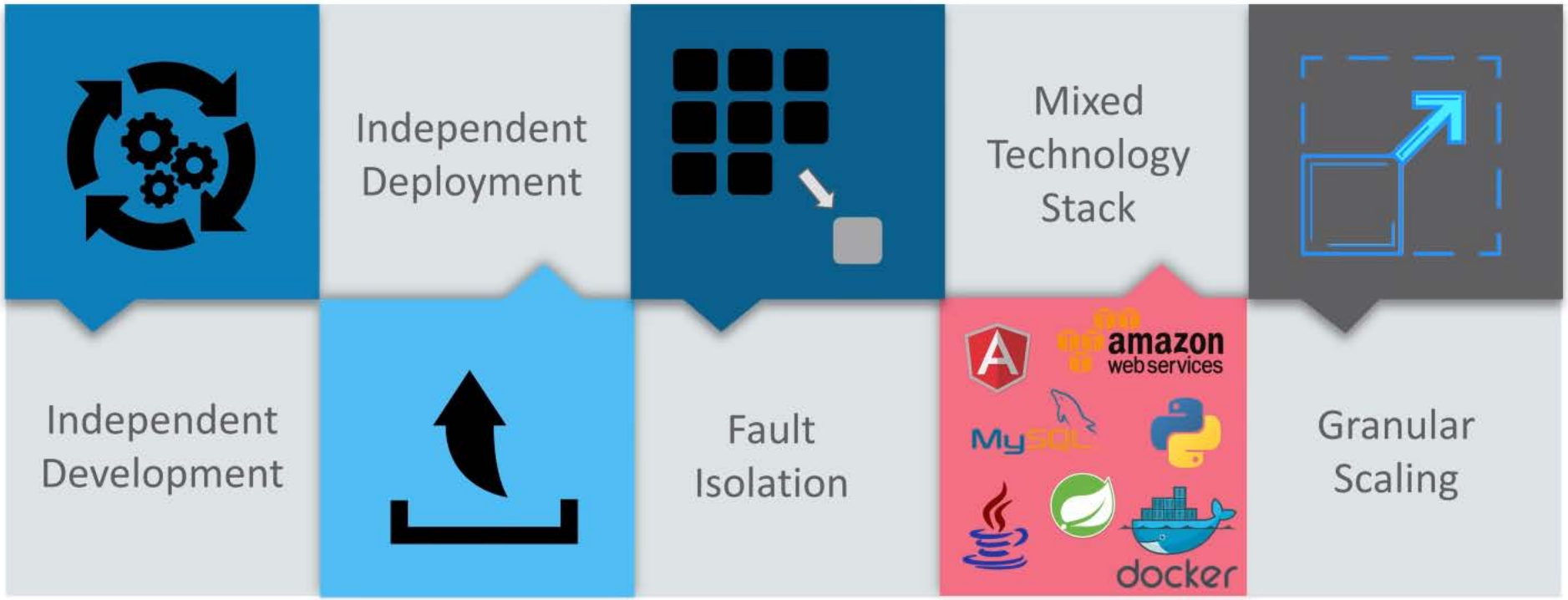
Microservice Architecture



Features Of Microservice Architecture



Advantages Of Microservice Architecture



Companies Using Microservices

SOFTWARES / TOOLS

- Java
- STS IDE
- MySQL WORKBENCH
- Postman

THANK YOU

Corporate Trainer & Motivational Speaker
9035351965