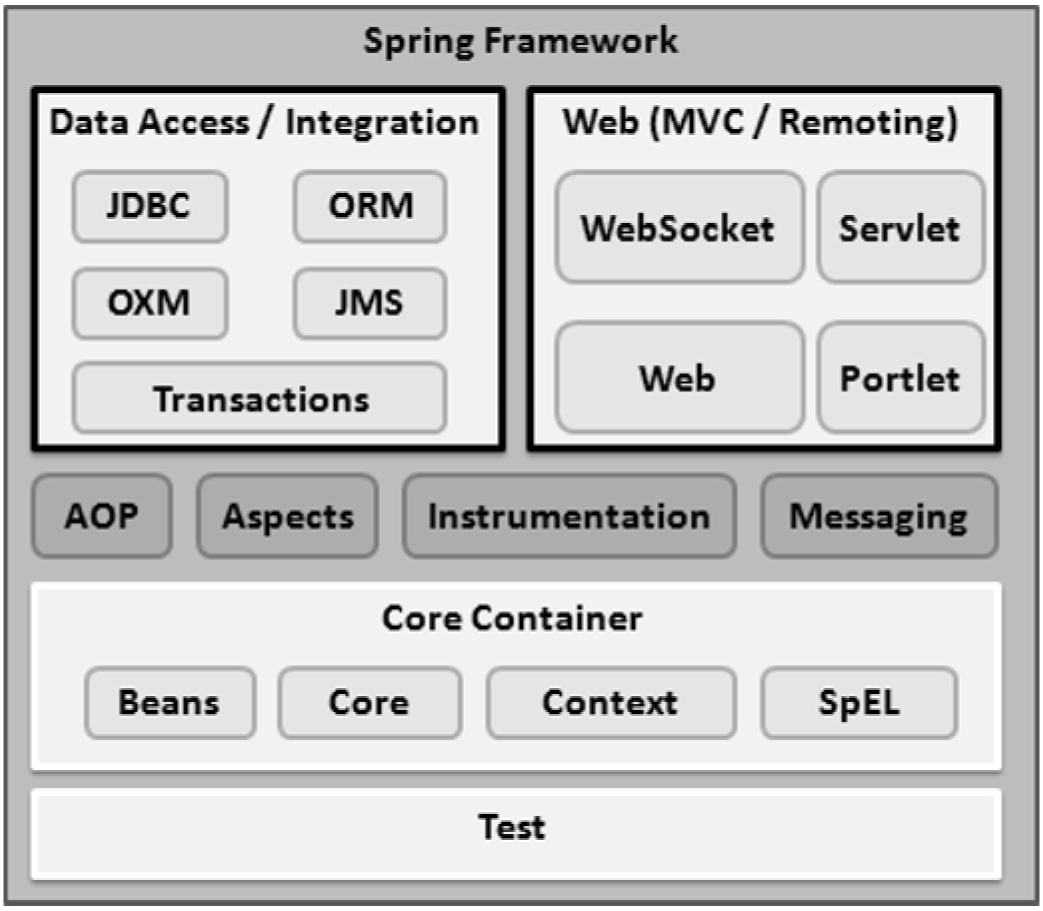
S.N.R.Likhitha(AF0312909)

1.Explain the architecture of Spring Framework.

The Spring Framework is a widely used open-source Java-based framework that provides comprehensive infrastructure support for developing Java applications. It is designed to simplify the development of complex and large-scale applications by offering a modular and layered architecture. The framework promotes the use of best practices, such as dependency injection (DI) and aspect-oriented programming (AOP), to achieve loose coupling and better maintainability.



The architecture of the Spring Framework is based on several key components, each of which plays a specific role in the application development process. Here's an overview of the main components and their roles:

Core Container: The core container is responsible for providing the fundamental building blocks of the Spring Framework. It includes the following components:

- **Beans**: The Beans module deals with the instantiation, configuration, and management of Java objects (beans). Beans are managed by the Spring container and are often configured through XML or annotations.
- **Core:** This module provides the core functionalities of the Spring Framework, including the IoC (Inversion of Control) container. The IoC container manages the lifecycle and dependencies of beans. Instead of having objects create their own dependencies, the container injects dependencies into objects, promoting loose coupling and easier testing.
- **Context:** The Context module builds on top of the Core module and provides a more advanced way to access application objects. It includes features such as internationalization, event propagation, resource loading, and application lifecycle events.
- **Expression Language (SpEL):** SpEL is a powerful expression language used for querying and manipulating objects at runtime. It is often used in configuration files to define dynamic values.

Data Access/Integration: This layer of the Spring Framework provides support for working with databases and other data sources. It includes the following modules:

- **JDBC (Java Database Connectivity):** This module simplifies database access using JDBC. It provides features like error handling, connection management, and simplified SQL execution.
- **ORM (Object-Relational Mapping):** The ORM module supports integration with popular ORM frameworks like Hibernate, JPA, and JDO. It helps
  manage the mapping between object-oriented models and relational databases.
- **Transaction:** The Transaction module provides support for programmatic and declarative transaction management. It abstracts away the complexities of handling transactions in a consistent manner.

Web: The Web layer is designed to simplify the development of web applications. It includes modules for various web-related tasks:

- **Web:** This module provides features for handling web-related tasks, such as multipart file uploads, initialization parameters, and programmatic registration of servlets and filters.

- **Web MVC:** The Web MVC module implements the Model-View-Controller (MVC) design pattern for building web applications. It offers a powerful and flexible way to structure and manage web application components.

AOP (Aspect-Oriented Programming): The AOP module allows developers to implement cross-cutting concerns, such as logging, security, and transaction management, separately from the main application logic. AOP achieves this by defining aspects that are applied to specific points in the application's execution.

Instrumentation: This module provides support for class instrumentation and profiling.

Messaging: The Messaging module simplifies the integration of messaging patterns and technologies within the Spring application.

Test: The Test module provides support for unit testing Spring components using JUnit or TestNG.

The Spring Framework's architecture encourages modularity, flexibility, and best practices, making it a popular choice for developing a wide range of Java applications, from small-scale projects to large enterprise systems.

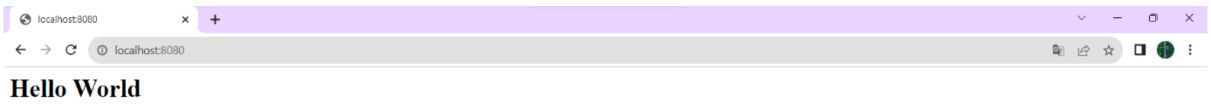
2. Create a Simple spring boot application to print hello world message to user in browser when the spring spoot app is up on your server.

Program:

```
package com.demo.HelloWorld;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@SpringBootApplication
public class HelloWorldApplication {

     @RequestMapping(value= "/")

     public String index() {
          return "<h1> Hello World<h1>";
```

```
      }

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

}
```

Output:



Hello World