**Spring MVC**

**I. Spring là gì?**

***1. Khái niệm và đặc điểm:***

- Spring là một Framework phát triển các ứng dụng Java được sử dụng bởi hàng triệu lập trình viên. Nó giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, sử dụng lại code…

Spring nhẹ và trong suốt (nhẹ: kích thước nhỏ, version cơ bản chỉ khoảng 2MB; trong suốt: hoạt động một cách trong suốt với lập trình viên)

Spring là một mã nguồn mở, được phát triển, chia sẻ và có cộng đồng người dùng rất lơn.

Spring Framework được xây dựng dựa trên 2 nguyên tắc design chính là: Dependency Injection và Aspect Oriented Programming.

Những tính năng core (cốt lõi) của Spring có thể được sử dụng để phát triển Java Desktop, ứng dụng mobile, Java Web. Mục tiêu chính của Spring là giúp phát triển các ứng dụng J2EE một cách dễ dàng hơn dựa trên mô hình sử dụng POJO (Plain Old Java Object)

***2. Kiến trúc của Java Spring:***



- Test: Tầng này cung cấp các khả năng hỗ trợ kiểm thử như: JUnit, TestNG.

- Spring Core Container:

+ Core và Bean cung cấp các tính năng IOC và Dependence Injection.

+ Context: hỗ trợ đa ngôn ngữ và các tính năng như EJB, JMX.

+ SpEL: Expression Language được mở rộng từ Expresion Language trong JSP. Nó cung cấp hỗ trợ việc setting/getting giá trị, các method cải tiến cho phép truy cập collections, index, các toán tử logic…

- AOP, Aspects and Instrumentation: Những module này hỗ trợ cài đặt lập trình hướng khía cạnh (Aspect Oriented Programming), hỗ trợ tích hợp với AspectJ.

- Data Access / Integration: Nhóm này bao gồm JDBC, ORM, OXM, JMS và module Transaction. Những module này cung cấp khả năng giao tiếp với database.

- Web: Hay còn gọi là Spring MVC Nhóm này gồm Web, Web-Servlet… hỗ trợ việc tạo ứng dụng web.

**3. Inversion of Control và Dependence Injection trong Spring:**

\* Project minh hoạ gồm có các class và interface như sau:

- Language là một interface gồm 2 phương thức:

public interface Language {

    public String getGreeting();

    public String getBye();

}

- Và 2 class implement interface trên:

English.java

public class English  implements Language {

  @Override

  public String getGreeting() {

      return "Hello";

  }

  @Override

  public String getBye() {

      return "Bye bye";

  }

}

Vietnamese.java

public class Vietnamese implements Language {

  @Override

  public String getGreeting() {

      return "Xin Chao";

  }

  @Override

  public String getBye() {

      return "Tam Biet";

  }

}

- Các class khác:

GreertingService.java

@Service

public class GreetingService {

   @Autowired

   private Language language;

   public GreetingService() {

   }

   public void sayGreeting() {

       String greeting = language.getGreeting();

       System.out.println("Greeting: " + greeting);

   }

}

*@Service là một annotation chú thích trên một class để cho Spring biết rằng nó là một Spring BEAN.*

*@Autowired là một annotation chú thích trên một trường cho Spring rằng nó là một trường cần được tiêm giá trị vào (hãy tiêm giá trị vào trường đó).*

MyRepository.java

@Repository

public class MyRepository {

    public String getAppName() {

        return "Hello Spring App";

    }

    public Date getSystemDateTime() {

        return new Date();

    }

}

-  *@Repository là một annotation chú thích trên một class để cho Spring biết rằng nó là một Spring BEAN.*

MyComponent.java

@Component

public class MyComponent {

    @Autowired

    private MyRepository repository;

    public void showAppInfo() {

        System.out.println("Now is: "+ repository.getSystemDateTime());

        System.out.println("App Name: "+ repository.getAppName());

    }

}

*@Component là một annotation chú thích trên một class để cho Spring biết rằng nó là một Spring BEAN.*

*Không có sự khác nhau giữa cách dùng các annotation @Repository, @Component và @Service, sử dụng tuỳ theo hoàn cảnh và ý nghĩa.*

AppConfiguration.java

@Configuration

@ComponentScan({"org.o7planning.spring.bean"})

public class AppConfiguration {

    @Bean(name ="language")

    public Language getLanguage() {

        return new Vietnamese();

    }

}

*@Configuration là một annotation trên một class, class náy sẽ dùng để định nghĩa các Spring BEAN.*

*@ComponentScan({“package.package”}) nói cho Spring các package dùng để tìm kiếm các Spring BEAN khác, Spring sẽ quét các package đó để tìm kiếm.*

***\* Các Spring BEAN được tạo sẽ được quản lý trong Spring IoC Container.***

MainProgram.java

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainProgram {

   public static void main(String[] args) {

       // Tạo ra một đối tượng ApplicationContext bằng cách đọc cấu hỉnh

       // trong class AppConfiguration

       ApplicationContext context = new AnnotationConfigApplicationContext(AppConfiguration.class);

       System.out.println("----------");

       Language language = (Language) context.getBean("language");

       System.out.println("Bean Language: "+ language);

       System.out.println("Call language.sayBye(): "+ language.getBye());

       System.out.println("----------");

       GreetingService service = (GreetingService) context.getBean("greetingService");

       service.sayGreeting();

       System.out.println("----------");

       MyComponent myComponent = (MyComponent) context.getBean("myComponent");

       myComponent.showAppInfo();

   }

}

\* Nguyên tắc hoạt động của Spring:

- Tạo ra một đối tượng ApplicationContext bằng cách đọc các cấu hình trong class được đánh annotation @Configuration (ở đây là class AppConfiguration):

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfiguration.class);

- Spring tạo ra một BEAN có tên language và được gán giá trị là Vietnamese.

- Tiếp theo, Spring sẽ tìm kiếm trong package được đánh dấu @ComponentScan … để tạo ra các Spring BEAN khác.

- Sau đó, các trường của Spring BEAN có chưa @Autowired sẽ được tiêm giá trị vào:

+ language trong greetingService sẽ được gán là Vietnamese.

+ myRepository trong myComponent sẽ được gán là myRepository.

***=> Theo cách truyền thống một đối tượng được tạo ra từ một class, các trường (field) của nó sẽ được gán giá trị từ chính bên trong class đó. Spring đã làm ngược lại với cách truyền thống, các đối tượng được tạo ra và một vài trường của nó được tiêm giá trị từ bên ngoài vào bởi một cái được gọi là IoC.   
IoC viết tắt của "Inversion of Control" - Có nghĩa là "Đảo ngược của sự điều khiển".   
IoC Container là bộ chứa tất cả các Spring BEAN được sử dụng trong ứng dụng.***

**II. Spring MVC:**

***\*Mở đầu:***

-

***1. Spring MVC là gì?***

- Spring MVC là một framework thực hiện mô hình MVC trong các ứng dụng web.

- Các include cần thiết trong Spring MVC:

JUnit

SpringFramework

JavaX.Servlet

***2. Controller trong Spring MVC:***

- Controller là nơi nhận request từ người dùng, xử lý request, xây dựng dữ liệu cho view (model) và chọn view để trả lại kết quả của cho người dùng.

-