T3H
IT-Institute

Lập trình Java
Spring framework

Ths. Vũ Duy Khương

Nội dung

1. Spring framework là gì ?
2. Thành phần chính trong Spring
3. Tính chất của Spring framework
4. Các annotation trong Spring

Spring framework là gì?

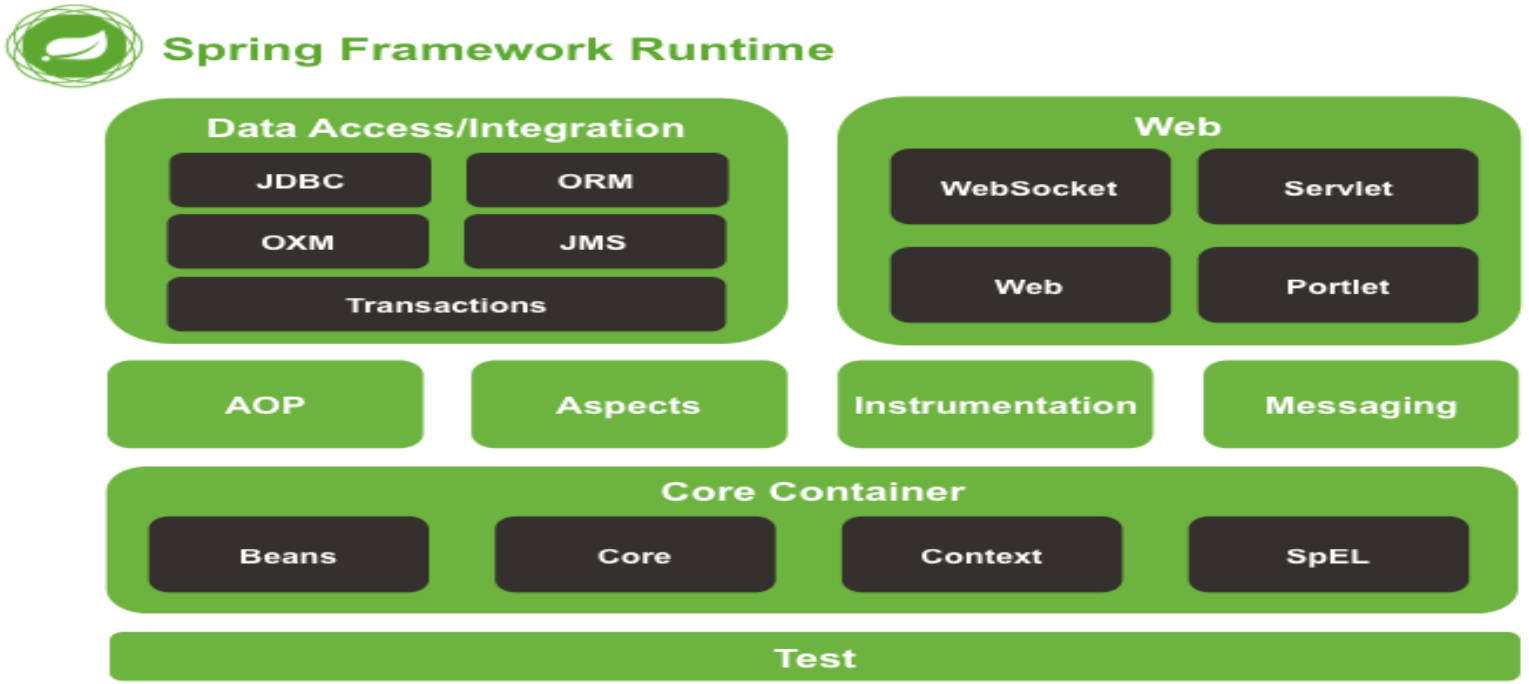
- Mục tiêu chính của Spring là giúp phát triển các ứng dụng J2EE(Java Platform Enterprise Edition) một cách dễ dàng hơn dựa trên mô hình sử dụng POJO (Plain Old Java Object)
- Spring Framework được xây dựng dựa trên 2 nguyên tắc design chính là: **Dependency Injection** và **Aspect Oriented Programming.**

Spring framework là gì?

- 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 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**.

Spring framework là gì?

Kiến trúc của Spring Framework:



Dependency Injection là gì?

Dependency Injection là gì?

Dependency Inject là 1 kỹ thuật, 1 design pattern cho phép xóa bỏ sự phụ thuộc hard-code và làm cho ứng dụng của bạn dễ mở rộng và maintain hơn.

Ví dụ:

1 ứng dụng gọi tới object của class MySQLDAO(class MySQLDAO chuyên thực hiện truy vấn với cơ sở dữ liệu MySQL của ứng dụng)

Dependency Injection là gì?

Dependency Injection là gì(tiếp)?

Bây giờ bạn muốn truy vấn tới cơ sở dữ liệu postgre. Bạn phải xóa khai báo MySQLDAO trong ứng dụng và thay bằng PostgreDAO, sau đó muốn dùng lại MySQLDAO bạn lại làm ngược lại… rõ ràng code sẽ phải sửa lại và test nhiều lần.

Dependency Injection là gì?

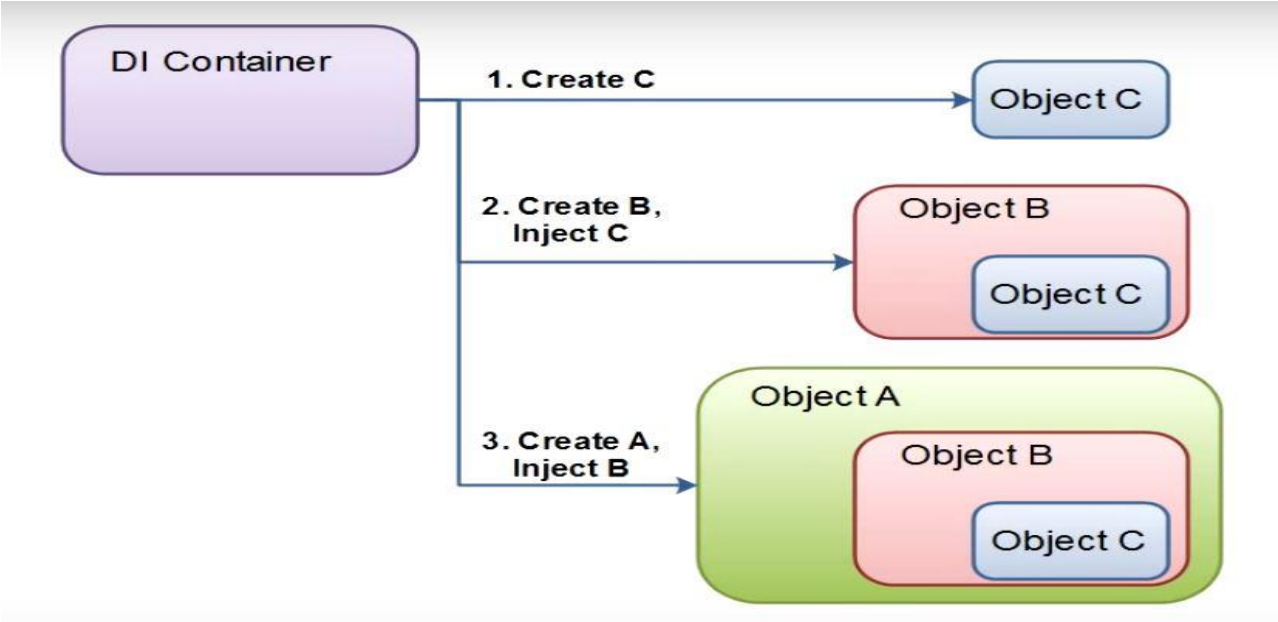
Dependency Injection là gì(tiếp)?

Dependency Inject chính là để giải quyết cho trường hợp như thế này. Trong ví dụ trên ta tạo 1 interface AbstractDAO và cho các class DAO kia thừa kế AbstractDAO. Bây giờ trong các class sử dụng DAO ta khai báo AbstractDAO, tùy theo điều kiện tương ứng AbstractDAO có thể là MySQLDAO hoặc PostgreDAO.

Dependency Injection là gì?

Dependency Injection là gì(tiếp)?

Việc thay thế AbstractDAO bằng MySQLDAO/PostgreDAO được gọi là injection.



Dependency Injection là gì?

Các phương pháp thực hiện Dependency Injection.

- **Constructor Injection**:

  Các dependency sẽ được truyền vào (inject vào) 1 class thông qua constructor của class đó. Đây là cách thông dụng nhất. (ví dụ trên mình dùng theo cách này)

- **Setter Injection**:

  Các dependency sẽ được truyền vào 1 class thông qua các hàm Setter/Getter

Inversion of Control

Inversion of Control

- Inversion of Control dịch là đảo ngược điều khiển (hơi khó hiểu)
- Ý của nó là làm thay đổi luồng điều khiển của ứng dụng. ví dụ như ở trên việc thay đổi thông tin trong file config.properties đã làm thay đổi luồng chạy của ứng dụng.

Thành phần chính trong Spring

DI Container

- DI Container là chỉ những thành phần tạo và quản lý module/object con được Inject, ví dụ ở trên là FactoryDAO.
- Hiện tại có rất nhiều Framework và các thư viện hỗ trợ làm DI như CDI, Spring DI, JSF…

Thành phần chính trong Spring

Spring IoC

IoC Container là thành phần thực hiện IoC.

Trong Spring, Spring Container (IoC Container) sẽ tạo các đối tượng, lắp rắp chúng lại với nhau, cấu hình các đối tượng và quản lý vòng đời của chúng từ lúc tạo ra cho đến lúc bị hủy.

Thành phần chính trong Spring

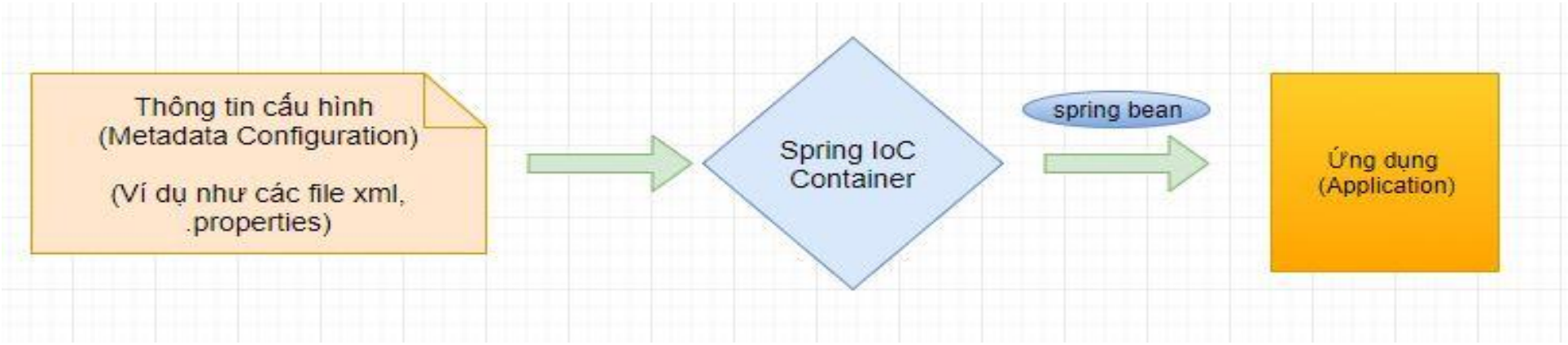
Spring IoC(tiếp)

Spring container sử dụng DI để quản lý các thành phần, đối tượng để tạo nên 1 ứng dụng. Các thành phần, đối tượng này gọi là Spring Bean (mình sẽ nói về Spring Bean trong các bài sau)

Để tạo đối tượng, cấu hình, lắp rắp chúng, Spring Container sẽ đọc thông tin từ các file xml và thực thi chúng.

Thành phần chính trong Spring

Spring IoC



IoC Container trong Spring có 2 kiểu là:

BeanFactory

ApplicationContext

Tính chất của Spring framework

Spring Bean, Các scope trong Spring, Spring Bean Scope

Spring Bean là gì?

Spring Bean là các object trong Spring Framework, được khởi tạo thông qua Spring Container. Bất kỳ class Java POJO nào cũng có thể là Spring Bean nếu nó được cấu hình và khởi tạo thông qua container bằng việc cung cấp các thông tin cấu hình (các file config .xml, .properties..)

Tính chất của Spring framework

Spring Bean, Các scope trong Spring, Spring Bean Scope

Các Bean Scope trong Spring?

Có 5 scope được định nghĩa cho Spring Bean:

Singleton: Chỉ duy nhất một thể hiện của bean sẽ được tạo cho mỗi container. Đây là scope mặc định cho spring bean. Khi sử dụng scope này cần chắc chắn rằng các bean không có các biến/thuộc tính được share.

Tính chất của Spring framework

Có 5 scope được định nghĩa cho Spring Bean (tiếp):

Prototype: Một thể hiện của bean sẽ được tạo cho mỗi lần được yêu cầu(request)

Request: giống với prototype scope, tuy nhiên nó dùng cho ứng dụng web, một thể hiện của bean sẽ được tạo cho mỗi HTTP request.

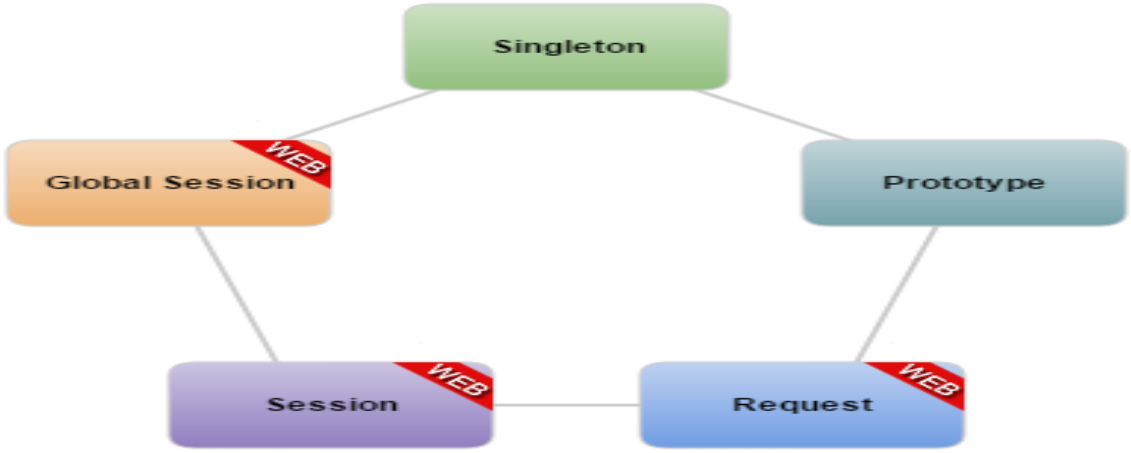
Session: Mỗi thể hiện của bean sẽ được tạo cho mỗi HTTP Session

Tính chất của Spring framework

Có 5 scope được định nghĩa cho Spring Bean (tiếp):

Global-Session: Được sử dụng để tạo global sesion bean cho các ứng dụng Portlet.

Trong 5 scope trên thì 3 scope cuối chỉ dùng trong ứng dụng web.



Tính chất của Spring framework

Spring DI với Object (Dependent Object)

Trường hợp mối quan hệ giữa các class là has-a (1 đối tượng chứa 1 đối tượng khác) chúng ta sẽ tạo bean cho đối tượng bên trong và truyền nó vào hàm khởi tạo hoặc setter.

Tính chất của Spring framework

Spring DI với Object (Dependent Object)

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

  <!-- Inject by setter -->
  <bean id="person" class="stackjava.com.springdiobject.demo.Person">
    <property name="name" value="stackjava.com"></property>
    <property name="age" value="25"></property>
    <property name="address" ref="address"></property>
  </bean>

  <!-- Inject by constructor -->
  <bean id="person2" class="stackjava.com.springdiobject.demo.Person">
    <constructor-arg name="name" type="String" value="spring"></constructor-arg>
    <constructor-arg name="age" type="int" value="30"></constructor-arg>
    <constructor-arg name="address" ref="address"></constructor-arg>
  </bean>

  <bean id="address" class="stackjava.com.springdiobject.demo.Address">
    <property name="country" value="Viet Nam"></property>
    <property name="province" value="Ha Noi"></property>
    <property name="district" value="Thanh Xuan"></property>
  </bean>

</beans>
```

Các annotation trong Spring

Spring Auto Component Scanning:

Thông thường, chúng ta khai báo tất cả các bean hoặc component trong file XML để Spring container có thể tìm và quản lý các bean.

Thực tế, Spring có khả năng tự động tìm, dò và tạo thể hiện của bean từ các định nghĩa ban đầu ở package, class mà không cần phải khai báo chúng trong file XML.

Spring Auto Component Scanning: Sử dụng XML tạo bean

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:p="http://www.springframework.org/schema/p"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

  <bean id="userService" class="stackjava.com.autoscancomponent.demo.UserService">
    <property name="userDAO" ref="userDAO" />
  </bean>
  <bean id="userDAO" class="stackjava.com.autoscancomponent.demo.UserDAO">
  </bean>

</beans>
```

Các annotation trong Spring

Spring Auto Component Scanning: Sử dụng Anotation tạo bean

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
  http://www.springframework.org/schema/context
  http://www.springframework.org/schema/context/spring-context-2.5.xsd">

  <context:component-scan base-package="stackjava.com.autoscancomponent.demo" />

</beans>
```

Các annotation trong Spring

Các annotation trong Spring:

@Component – biểu thị đây là một component được tự động scan.

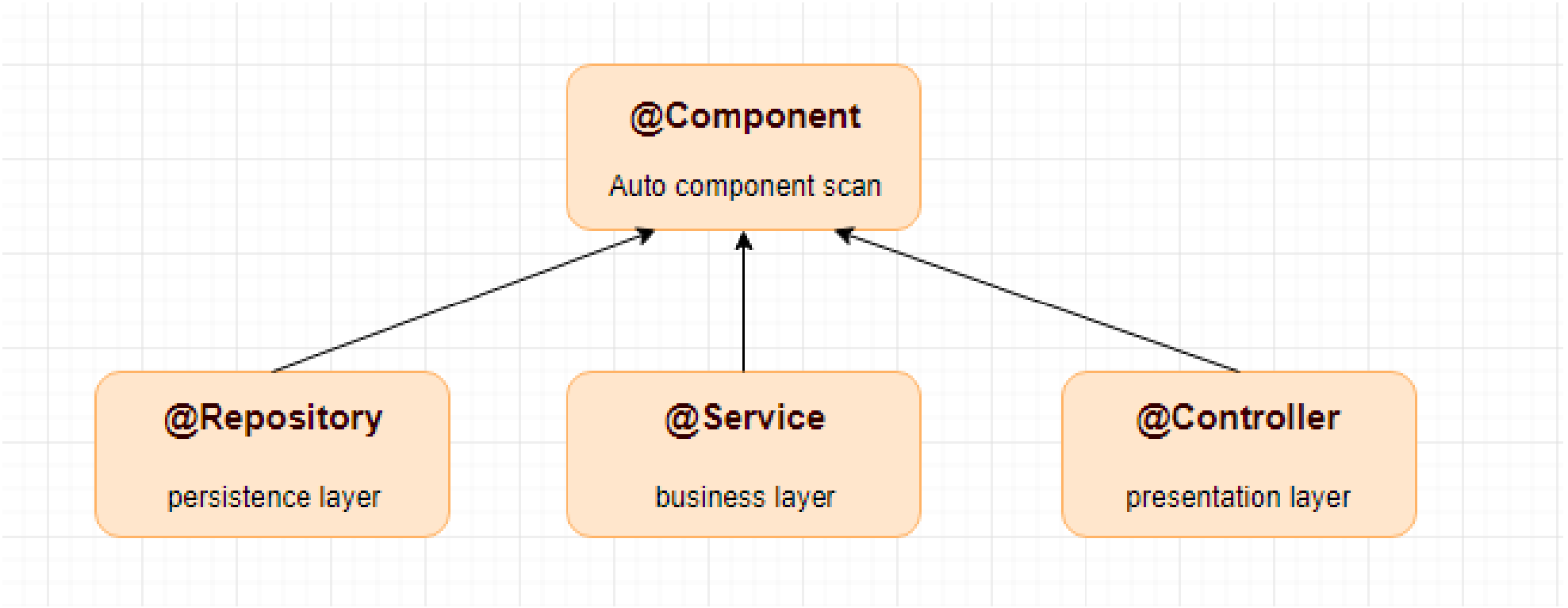
@Repository – biểu thị đây là một DAO component trong tầng persistence.

@Service – biểu thị đây là một Service component trong tầng business.

@Controller – biểu thị đây là một Controller component trong tầng presentation

Các annotation trong Spring

Các annotation trong Spring:



Các annotation trong Spring

Vậy khi nào dùng @Repository,@Service hay @Controller?

Thực ra cả 4 annotation này chỉ dùng với mục đích đánh đấu là auto component scan, bạn có thể dùng chúng lẫn lộn, hoặc nếu không rõ class đang ở tầng persistence, business hay presentation thì cứ dùng @Component nó vẫn hoạt động.

Lưu ý, mặc định bean được tạo từ auto scan component có tên là tên của class với chữ đầu tiên viết thường.

Ví dụ: UserService -> userService, UserDAO -> userDAO

Q&A

THANK YOU