Spring MVC

Mục lục

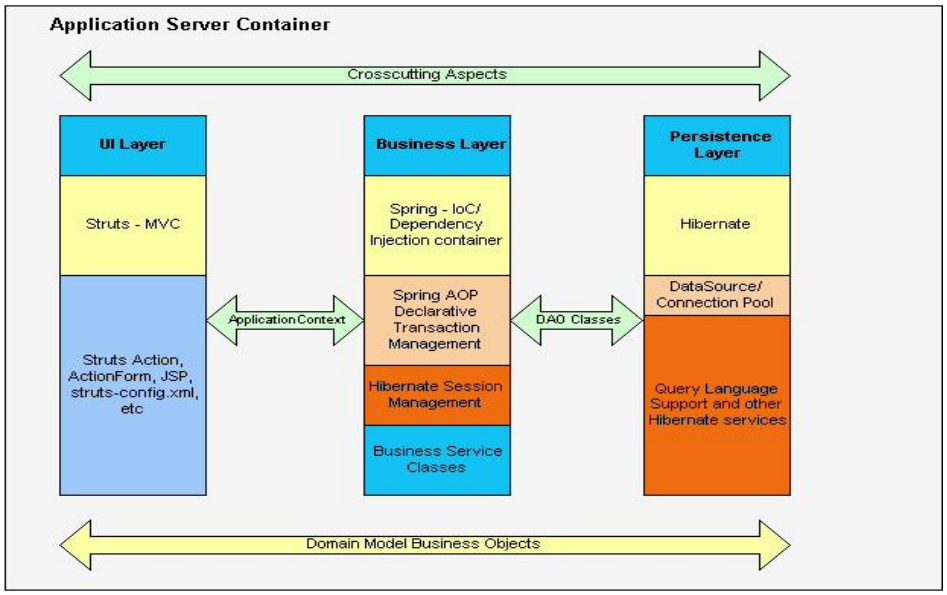
1 Spring Framework.

1.1 Giới thiệu.

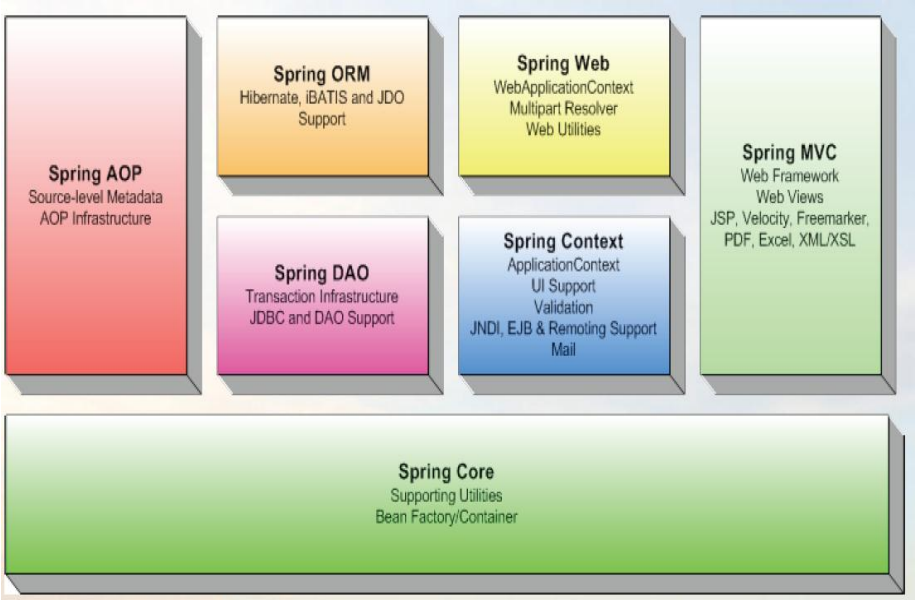
- Spring Framework, hay ngắn hơn là *Spring*, là một framework mã nguồn mở được viết bằng Java. Nó được xem như là một giải pháp kiến trúc tốt nhất của Java EE hiện nay.
- Theo thiết kế, bộ khung hình này giải phóng lập trình viên dùng Java, cho phép họ nhiều quyền tự do hơn.
- Đồng thời cung cấp một giải pháp tiện lợi, đầy đủ dẫn chứng bằng tài liệu, dễ dàng sử dụng, phù hợp với những thực hành thông dụng trong công nghệ phần mềm.

1.2 Các module chính.

- Spring đóng vai trò là tầng business.



- Spring được tổ chức thành 7 modules :



1.2.1 Core Container.

- Core package là phần cơ bản nhất của framework, cung cấp những đặc tính IoC (*Inversion of Control: Khung chứa đảo ngược khống chế*) và Dependency Injection.
- Khái niệm cơ bản là BeanFactory - cài đặt factory pattern cho phép bạn móc nối sự phụ thuộc giữa các đối tượng trong file cấu hình.

1.2.2 Spring Context/Application Context.

- Phía trên của Core package là Context package - cung cấp cách để truy cập đối tượng.
- Context package kết thừa các đặc tính từ bean package và thêm vào chức năng đa ngôn ngữ (I18N), truyền sự kiện, resource-loading,...

1.2.3 Spring AOP (Aspect Oriented Programming).

- Spring AOP module tích hợp chức năng lập trình hướng khía cạnh vào Spring framework thông qua cấu hình của nó. Spring AOP module cung cấp các dịch vụ quản lý giao dịch cho các đối tượng trong bất kỳ ứng dụng nào sử dụng Spring. Với

Spring AOP chúng ta có thể tích hợp declarative transaction management vào trong ứng dụng mà không cần dựa vào EJB component.

- Spring AOP module cũng đưa lập trình metadata vào trong Spring. Sử dụng cái này chúng ta có thể thêm annotation (chú thích) vào source code để hướng dẫn Spring và làm thế nào để liên hệ với aspect (Khía cạnh khác).

1.2.4 Spring DAO.

- DAO package cung cấp cho tầng JDBC, bỏ bớt những coding dài dòng của JDBC và chuyển đổi mã lỗi được xác định bởi database vendor. JDBC package cung cấp cách lập trình tốt như declarative transaction management.
- Tầng JDBC và DAO đưa ra một cây phân cấp exception để quản lý kết nối đến database, điều khiển exception và thông báo lỗi được ném bởi vendor của database. Tầng exception đơn giản điều khiển lỗi và giảm khối lượng code mà chúng ta cần viết như mở và đóng kết nối. Module này cũng cung cấp các dịch vụ quản lý giao dịch cho các đối tượng trong ứng dụng Spring.

1.2.5 Spring ORM.

- ORM package cung cấp tầng tích hợp với object-relational mapping API bao gồm: JDO, Hibernate, iBatis.
- Sử dụng ORM package bạn có thể sử dụng tất cả các object-relational mapping đó kết hợp với tất cả các đặc tính của Spring như declarative transaction management.

1.2.6 Spring Web module.

- Spring Web package cung cấp đặc tính của web như: chức năng file-upload, khởi tạo IoC container sử dụng trình lắng nghe serlvet và web-oriented application context.
- Nằm trên application context module, cung cấp context cho các ứng dụng web. Spring cũng hỗ trợ tích hợp với Struts, JSF và Webwork. Web module cũng làm giảm bớt các

công việc điều khiển nhiều request và gắn các tham số của request vào các đối tượng domain.

1.2.7 Spring MVC Framework.

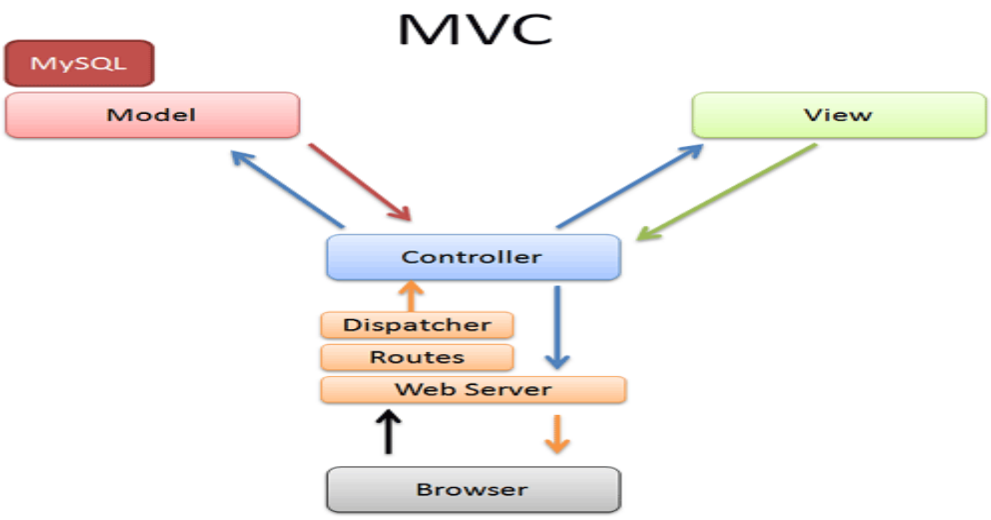
- Spring Framework là một ứng dụng mã nguồn mở phổ biến cho Java EE phát triển dễ dàng hơn. Nó bao gồm một container, một framework cho các thành phần quản lý, và một bộ các snap-in dịch vụ cho các giao diện người dùng web, giao dịch,…. Một phần của Spring Framework là Spring Web MVC, một mô hình MVC mở rộng để tạo các ứng dụng web.
- MVC Framework thì cài đặt đầy đủ đặc tính của MVC pattern để xây dựng các ứng dụng Web. MVC framework thì cấu hình thông qua giao diện và chứa được một số kỹ thuật view bao gồm: JSP, Velocity, Tiles và generation of PDF và Excel file. . Spring MVC framework cung cấp sự phân biệt rõ ràng giữa domain model và web form.

1.3 Lý do sử dụng.

- Tất cả các framework đã được tích hợp rất tốt vào Spring.
- Hoạt động rất tốt khi áp dụng theo kiến trúc MVC.
- Sử dụng cơ chế plug-in.
- Kết hợp rất tốt với các O/R (object-relational) Mapping frameworks như là Hibernate.
- Dễ Testing ứng dụng.
- Ít phức tạp hơn so với các framework khác.
- Cộng đồng người sử dụng rất nhiều, nhiều sách mới được xuất bản.

2 Spring Web MVC.

2.1 Mô hình MVC.



2.1.1 Model (Mô hình).

- Mô hình gồm các lớp java có nhiệm vụ:
  - Biểu diễn data và cho phép truy cập tới để get và set data trong (JAVABEAN), Thường thì phần layer này mô phỏng 1 cách đầy đủ đối tượng từ thế giới thực.
  - Nhận các yêu cầu từ khung nhìn
  - Thi hành các yêu cầu đó (tính toán, kết nối CSDL …)
  - Trả về các giá trị tính toán theo yêu cầu của Controller

2.1.2 View (khung nhìn) :

- Bao gồm các mã tương tự như JSP,_HTML, CSS, XML, Javascript, JSON… để hiển thị giao diện người dùng, các dữ liệu trả về từ Model thông qua Controller…

2.1.3 Controller (Bộ điều khiển) :

- Đồng bộ hoá giữa Khung nhìn và Mô hình. Tức là với một trang JSP này thì sẽ tương ứng với lớp java nào để xử lý nó và ngược lại, kết quả sẽ trả về trang jsp nào.Nó đóng vai trò điều tiết giữa View và Model.
- Như vậy, chúng ta có thể tách biệt được các mã java ra khỏi mã html. Do vậy, nó đã giải quyết được các khó khăn đã nêu ra trong Mô hình 1. Người thiết kế giao diện và người lập trình java có thể mang tính chất độc lập tương đối.

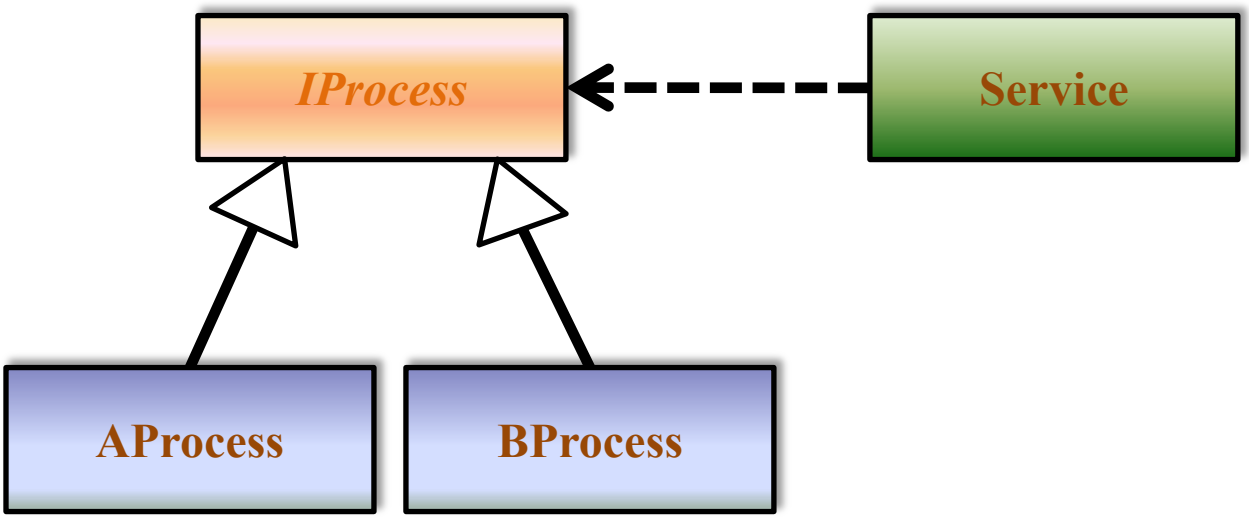
- Việc debug hay bảo trì sẽ dễ dàng hơn, việc thay đổi các theme của trang web cũng dễ dàng hơn …

2.2 Spring MVC.

- Framework Spring như thùng chứa nhẹ và hỗ trợ nhiều framework và các thư viện. Nó cho phép các nhà phát triển trộn và kết hợp nhiều framework khi phát triển và triển khai các ứng dụng. Các Spring MVC có những lợi thế từ Spring framework và cung cấp một lần tốt nhất cho framework và hiệu quả cho sự phát triển của các ứng dụng web.
- Các MVC Spring là một framework phát triển web dựa trên các mẫu thiết kế MVC (Model View Controller). Các tính năng của framework Spring MVC là Pluggable công nghệ View và Injection dịch vụ vào điều khiển.

2.3 Các khái niệm liên quan.

2.3.1 Inversion of Control Container (IoC container) :



- Hai package: org.springframework.beans và

org.springframework.context cung cấp IoC container cho Spring framework. Giao tiếp BeanFactory cung cấp kỹ thuật cấu hình nâng cao, có khả năng quản lý các đối tượng.

- Giao tiếp ApplicationContext kế thừa từ BeanFactory và thêm một vài chức năng khác như tích hợp với đặc tính Spring AOP, truyền sự kiện, application context như WebApplicationContext sử dụng trong ứng dụng web.
- Hiệu chỉnh các components của chương trình và quản lý vòng đời (lifecycle) của các đối tượng Java.

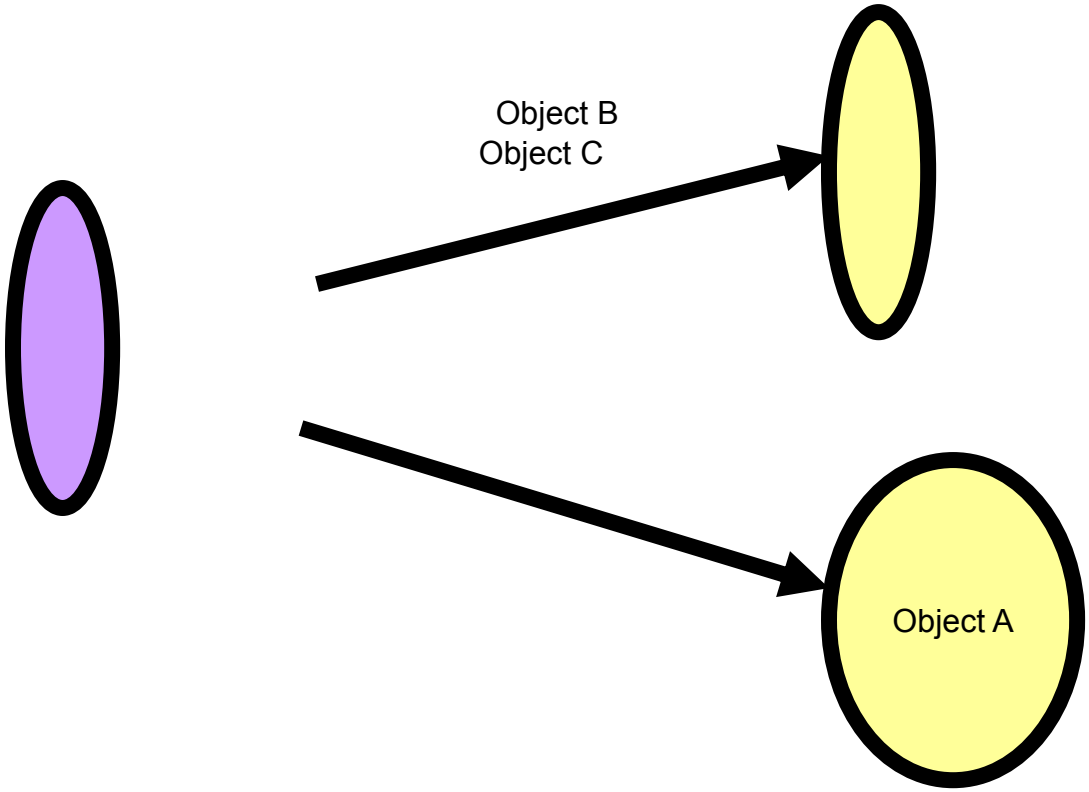
2.3.2 Bean :

- Các đối tượng trong ứng dụng được quản lý bởi IoC container được gọi là các bean. Một bean thì đơn giản là một đối tượng được khởi tạo, phân phát và được quản lý bởi IoC container. Sự phụ thuộc giữa chúng được phản ánh trong configuration metadata được sử dụng bởi container.
- Giao tiếp org.springframework.beans.factory.BeanFactory là IoC container chịu trách nhiệm chứa, khởi tạo, quản lý và móc nối sự phụ thuộc giữa các bean.
- Một cài đặt (kế thừa) của BeanFactory là XmlBeanFactory - nhận XML configuration metadata và sử dụng nó để cấu hình cho ứng dụng.
- Configuration metadata có định dạng XML (hoặc properties), được sử dụng để viết các *bean definitions* cho các bean mà bạn muốn IoC container quản lý.
- Ở mức cơ bản nhất thì cấu hình IoC container phải chứa ít nhất một bean mà container đó phải quản lý. Khi sử dụng XML-based configuration metadata, các bean này được cấu hình như một hoặc nhiều element bên trong element. Các bean này tương ứng với các đối tượng được tạo trong ứng dụng.

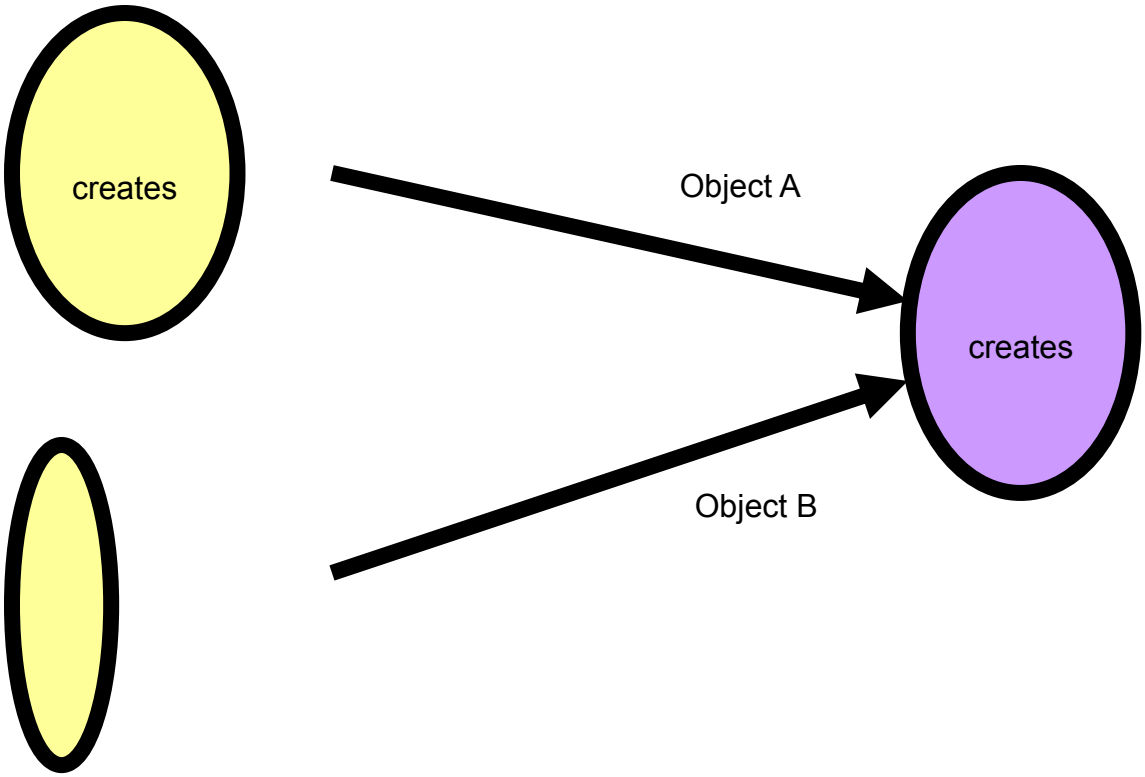
2.3.3 Dependency Injection (DI) :

- Kiểm soát ứng dụng, Liên kết các đối tượng lại với nhau.

- Tất cả thông tin ứng dụng được cấu hình trong một file xml do dó dễ dàng thay đổi , mở rộng ứng dụng như là một kiến trúc plug-in.



- Đối tượng A tạo ra đối tượng B,C phụ thuộc vào A nhưng không liên kết chặt chẽ với A.



- Một đối tượng A chứa các phương thức setter *accept interfaces to objects B and C*. Điều này có thể cũng đã đạt được với các hàm trong đối tượng A chấp nhận các đối tượng B và C
- Là các đối tượng định nghĩa sự phụ thuộc của chúng thông qua tham số của phương thức khởi tạo (constructor) hoặc các thuộc tính được thiết lập trên thể hiện (instance) sau khi nó được khởi tạo hoặc trả về từ phương thức factory. Sau đó là công việc của container, nó sẽ đảo ngược sự phụ thuộc đó khi nó tạo bean. Do đó nó có tên là Inversion of Control (IoC).
- Có hai biến thể chính của DI là: Setter Injection và Constructor Injection.

2.3.3.1 Setter Injection:

- Cho đoạn code sau :

```
<beans>
 <bean name="person" class="examples.spring.Person">
    <property name="email">
```

```
        <value>my@email.address</value>
    </property>
  </bean>
</beans>
```

- Đoạn code trên tương ứng với đoạn code sau :

```
Person person = new Person();
person.setEmail(“my@email.address”);
```

- Khởi tạo đối tượng Person và gán giá trị email cho nó. Được nhận ra bởi phương thức setter trên bean sau khi triệu gọi phương thức khởi tạo không tham số hoặc phương thức static factory không tham số để khởi tạo bean.

2.3.3.2 Constuctor injection:

- Được nhận ra bằng cách triệu gọi một phương thức khởi tạo với một số tham số.
- Một biến thể khác thay thế cho constructor là sử dụng phương thức static factory để trả về thể hiện của đối tượng.

```
<beans>
  <bean name="fileDataProcessor“  class="examples.spring.DataProcessor"
    singleton="true">
    <constructor-arg>
            <ref bean="fileDataReader"/>
    </constructor-arg>
```

```
 </bean>

 <bean name="fileDataReader" class="examples.spring.FileDataReader"
singleton="true">

    <constructor-arg>

            <value>/data/file1.data</value>

    </constructor-arg>

 </bean>

</beans>
```

- Giải thích đoạn code trên :

```
FileDataReader fileDataReader = new FileDataReader("/data/file1.data");

DataProcessor fileDataProcessor = new DataProcessor(fileDataReader);
```

2.3.4 Application Context :

- Trong khi Bean Factory được sử dụng cho các ứng dụng đơn giản, thì Application Context là một container nâng cao của Spring. Giống như BeanFactory, nó có thể được sử dụng để load các định nghĩa bean, gắn các bean với nhau và phân phát các bean theo yêu cầu.
- Nó cũng cung cấp:
  - giải quyết text message, bao gồm hỗ trợ internationlization.
  - cách chung để load file resource.
  - các sự kiện để bean được đăng ký như các trình lắng nghe.
- Có 3 cách sử dụng cài đặt Application Context:

2.3.4.1 ClassPathXmlApplicationContext:

- Nó load định nghĩa context từ XML file được đặt trong classpath, xem các định nghĩa context như các tài nguyên của classpath. Application context thì được load từ classpath của ứng dụng như sau:

```
ApplicationContext   context = new ClassPathXmlApplicationContext("bean.xml");
```

2.3.4.2 FileSystemXmlApplicationContext:

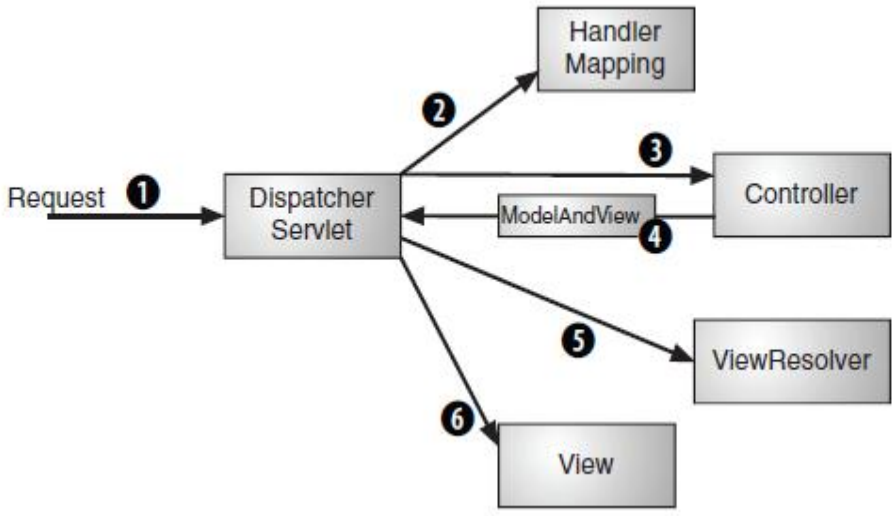
- Nó load định nghĩa context từ XML file trong từ hệ thống file. Application Context được load từ hệ thống file như sau:

```
ApplicationContext context=new FileSystemXmlApplicationContext("bean.xml");
```

2.3.4.3 XmlWebApplicationContext:

Nó sẽ load các định nghĩa context từ XML file trong ứng dụng web.

2.4 Cơ chế xử lý Request-Respone.



- Cũng giống như các java-base MVC FrameWork khác Spring MVC cũng phải requests thông qua một front controller servlet. Một bộ front controller servlet đại diện duy nhất chịu trách nhiệm về yêu cầu các thành phần khác của một ứng dụng để

thực hiện việc xử lý thực tế. Trong trường hợp của Spring MVC, DispatcherServlet là bộ điều khiển phía trước.

- Một Request được gửi bởi DispatcherServlet đến điều khiển (được chọn thông qua một bản đồ xử lý). Một khi điều khiển kết thúc, yêu cầu sau đó được gửi để xem (đó là lựa chọn thông qua ViewResolver) để làm cho đầu ra.

2.4.1 Giải thích sơ đồ luồng dữ liệu :

1. Request được gửi đến DispatcherServlet .
2. DispatcherServlet gửi yêu cầu đến Handler Mapping (Một bản đồ cấu hình URL) để xác định controller nào sẻ xử lý yêu cầu này.
3. DispatcherServlet gửi yêu cầu đến Controller sau khi biết được Controller nào sẻ xử lý yêu cầu. Nếu yêu cầu đó cần truy xuất cơ sở dữ liệu thì Controller sẻ ủy nhiệm cho một business logic hay nhiều hơn một service Objects (MODEL) để lấy thông tin và gủi dữ liệu về cho Controller lúc này Controller đóng gói mô hình dử liệu và tên của một view sẻ được tải lên thành đối tượng ModelAndView.
4. Gói ModelAndView được gửi trả về DispatcherServlet.
5. DispatcherServlet gửi gói ModelAndView cho ViewResolver để tìm xem trang web (JSP) nào sẻ được load lên. DispatcherServlet load trang web đó lên cùng với dữ liệu của nó.

2.4.2 Configuring DispatcherServlet :

- DispatcherServlet đóng vai trò như trái tim của Spring MVC. Một servlet có chức năng giống như Spring MVC’s front controller. Giống như bất kỳ servlet nào, DispatcherServlet phải dược cấu hình trong web.xml file.

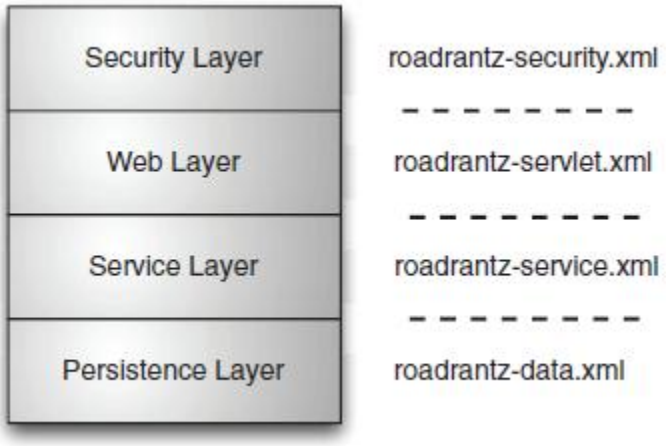
```
<servlet>
<servlet-name>roadrantz</servlet-name>
<servlet-class>
org.springframework.web.servlet.DispatcherServlet
```

```
</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>
```

- Theo mặc định, khi DispatcherServlet được nạp, nó sẽ tải các ứng dụng Spring context từ file XML có tên dựa vào tên của servlet.
- Trong trường hợp này, vì servlet có tên roadrantz, DispatcherServlet sẽ cố gắng để tải các ứng dụng context từ một file có tên roadrantz-servlet.xml.
  - Sau đó, bạn phải chỉ ra những gì các URL sẽ được xử lý bởi các DispatcherServlet. Thêm <servlet-mapping> sau đây vào web.xml để cho DispatcherServlet xử lý tất cả các URL mà kết thúc bằng ".html":

```
<servlet-mapping>
<servlet-name>roadrantz</servlet-name>
<url-pattern>*.htm</url-pattern>
    </servlet-mapping>
```

  - Trong thực tế, nên chia ngữ cảnh ứng dụng của bạn qua lớp ứng dụng, như thể hiện trong hình sau.



    - Phân chia một ứng dụng thành các lớp riêng biệt để sạch sẽ phân chia trách nhiệm. lớp An ninh (Security Layer) đóng chặt những ứng dụng, lớp web-layer

tập trung vào tương tác người dùng, Service Layer tập trung vào business logic, và Persistence Layer có liên quan đến cơ sở dữ liệu.

2.4.3 Configuring a context loader :

- Để đảm bảo rằng tất cả các file cấu hình được nạp, bạn sẽ cần phải cấu hình một Context Loader trong tập tin web.xml của bạn. Một bộ Context Loader cấu hình Ngoài các tập tin trong DispatcherServlet. Việc phổ biến nhất Context Loader được sử dụng là một servlet listener gọi là servlet ContextLoaderListener được cấu hình trong web.xml như sau:

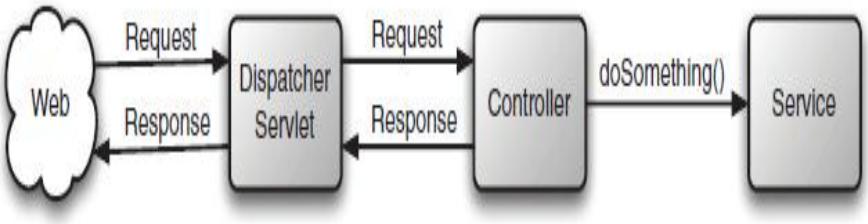
```
<listener>
<listener-class>org.springframework.
web.context.ContextLoaderListener</listener-class>
</listener>
```

- Với ContextLoaderListener khi cấu hình, bạn sẽ cần phải nói cho nó vị trí của Spring cấu hình files để tải. Nếu không quy định khác, các bối cảnh bộ nạp sẽ tìm một file cấu hình Spring ở / WEB-INF/applicationContext.xml. Nhưng vị trí này không thích hợp để chia nhỏ các ngữ cảnh ứng dụng trên lớp ứng dụng, do đó, có thể bạn sẽ muốn ghi đè mặc định này.
Bạn có thể chỉ định một hoặc nhiều file cấu hình Spring cho bộ nạp ngữ cảnh để tải , bằng cách thiết lập các thông số contextConfigLocation trong bối cảnh servlet:

```
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>
/WEB-INF/roadrantz-service.xml
/WEB-INF/roadrantz-data.xml
/WEB-INF/roadrantz-security.xml
</param-value>
</context-param>
```

- Các tham số contextConfigLocation được quy định như một danh sách các đường dẫn (liên quan đến ứng dụng web gốc). Khi cấu hình ở đây, bộ tải bối cảnh sẽ sử dụng contextConfigLocation để tải tập tin cấu hình ba bối cảnh , một cho Security Layer, một cho các tầng dịch vụ(Service Layer), và một cho các lớp dữ liệu (data layer).
- DispatcherServlet bây giờ là cấu hình và sẵn sàng để gửi yêu cầu đến lớp ứng dụng web của bạn. Nhưng lớp web đã chưa được xây dựng ! Đừng băn khoăn. Chúng tôi sẽ xây dựng nhiều lớp web trong chương này. Hãy bắt đầu bằng cách cảm nhận được một cái nhìn tổng quanlàm thế nào tất cả các mảnh của Spring MVC được lắp ráp để sản xuất các chức năng web.

2.4.4 Building the controller :



- HomePageController lấy một danh sách các rants gần đây để hiển thị trên trang chủ

```
package com.roadrantz.mvc;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.AbstractController;
import com.roadrantz.service.RantService;

public class HomePageController extends AbstractController {
  public HomePageController() {}

  protected ModelAndView handleRequestInternal(
      HttpServletRequest request, HttpServletResponse response)
      throws Exception {

    List recentRants = rantService.getRecentRants();

    return new ModelAndView("home",
        "rants", recentRants);
  }

  private RantService rantService;
  public void setRantService(RantService rantService) {
    this.rantService = rantService;
  }
}
```

Retrieves list of rants

Goes to "home" view

Returns rants in model

Injects RantService

- Trường hợp Spring MVC điều khiển khác nhau từ một servlet hay một hành động Struts là nó cấu hình như một JavaBean chỉ trong bối cảnh ứng dụng Spring. Điều này có nghĩa là bạn có thể tận dụng đầy đủ các dependency injection (DI) và Spring AOP với một controller class giống như bạn làm với bất kỳ Bean khác.
- Trong trường hợp của HomePageController, DI được sử dụng để bao bọc trong một RantService . HomePageController delegates trách nhiệm để lấy danh sách các rants gần đây các RantService.

2.4.5 Introducing ModelAndView :

- Một khi các logic nghiệp vụ đã được hoàn thành bởi các đối tượng dịch vụ, đó là thời gian cho bộ điều khiển để gửi kết quả trở lại trình duyệt. Các ModelAndView lớp đại diện cho một khái niệm quan trọng trong Spring MVC. Trong thực tế, mọi điều khiển thực hiện phương thức phải trả về ModelAndView.

- Các đối tượng ModelAndView được xây dựng như sau:

```
new ModelAndView("home", "rants", recentRants);
```

Hai thông số "home" và recentRants sẽ được chuyển choView . Hai thông số tạo thành tên của mô hình đối tượng một tham số thứ ba. Trong trường hợp này, danh sách rants trong biến recentRants sẽ được chuyển đến View với một tên là "rants".

2.4.6 Configuring the controller bean:

- Bây giờ HomePageController đã được viết, đó là thời gian để cấu hình nó trong DispatcherServlet contex của tập tin cấu hình (đó là roadrant-servlet.xml cho các ứng dụng RoadRantz).
- Các đoạn sau đây của XML cấu hình cho HomePageController.

```
<bean name="/home.htm"
class="com.roadrantz.mvc.HomePageController">
<property name="rantService" ref="rantService" />
</bean>
```

2.5 Truy xuất dữ liệu trong Spring MVC.

- Spring cung cấp một template JDBC để bạn có thể quản lý những kết nối của bạn.
- Một ví dụ đơn giản để bạn có thể kết nối đến một datasource :
- ProductManagerDaoJdbc implements ProductManagerDao {

```
public void setDataSource(DataSource ds)
{
this.ds = ds;
}
```

- Không cần thay đổi code java khi thay đổi datasource :

```
<beans>
    <bean name="dataSource" class="com.mysql.jdbc.jdbc2.optional.MysqlDataSource" destroy-method="close">
    <property name="url">
        <value>jdbc:mysql://localhost/test</value>
    </property>
<beans>
<bean id="prodManDao" class="db.ProductManagerDaoJdbc">
    <property name="dataSource">
        <ref bean="dataSource"/>
    </property>
</bean>
```