[**Chương 1: Giới thiệu tổng quan** 3](#_Toc447450024)

[1.1 Giới thiệu về ngôn ngữ Java và công nghệ web java. 3](#_Toc447450025)

[1.1.1 Khái niệm ngôn ngữ lập trình java 3](#_Toc447450026)

[1.1.2 Lịch sử phát triển 3](#_Toc447450027)

[1.1.3 Java servelet 4](#_Toc447450028)

[1.1.4 Vòng đời của một Servlet 5](#_Toc447450029)

[1.1.5 Java server page(JSP) 6](#_Toc447450030)

[**Chương 2: Tìm hiểu Struts framework** 7](#_Toc447450031)

[2.1 Mô hình MVC 7](#_Toc447450032)

[2.1.1 Mô hình lập trình MVC 7](#_Toc447450033)

[2.1.2 Cấu trúc mô hình MVC 8](#_Toc447450034)

[2.1.3 Định nghĩa từng thành phần trong MVC 9](#_Toc447450035)

[2.2 Struts framework 11](#_Toc447450036)

[2.2.1 Giới thiệu Struts framework 11](#_Toc447450037)

[2.2.1.1 Khái niệm Struts framework 11](#_Toc447450038)

[2.2.1.3 Tại sao phải cần một framework 11](#_Toc447450039)

[2.2.1.4 Nhược điểm của Framework 12](#_Toc447450040)

[2.2.3 Struts 1 Framework 12](#_Toc447450041)

[2.2.3.1 Sơ lược về Struts 1 12](#_Toc447450042)

[2.2.3.2 Kiến trúc struts 1 13](#_Toc447450043)

[2.2.3.3 Các thành phần trong Struts 1 14](#_Toc447450044)

[2.2.3.4 Xây dựng các thành phần Struts 1 16](#_Toc447450045)

[2.2.4 Struts 2 FrameWork 29](#_Toc447450046)

[2.2.4.1 Sơ lược về Struts 2 29](#_Toc447450047)

[2.2.4.3 Mô hình làm việc và cơ chế hoạt động của Struts 2 Framework 35](#_Toc447450049)

[2.2.4.4 Cấu trúc Struts 2 38](#_Toc447450050)

[2.3 So sánh khác nhau của Struts 1 Framework và Struts 2 Framework 40](#_Toc447450053)

[**Chương 3: Ứng dụng Struts 1 Framework vào xây dựng hệ thống quán lý**  42](#_Toc447450054)

KHÓA LUẬN TỐT NGHIỆP

1. Lời cảm ơn
2. Mục lục
3. Ký hiệu chữ viết tắt
4. MỞ ĐẦU
5. Lý do chọn đề tài
6. Đối tượng và phạm vi nghiên cứu
7. Mục đích và nhiệm vụ nghiên cứu
8. Lịch sử nghiên cứu
9. Nguồn dữ liệu và phương pháp nghiên cứu
10. Đóng góp của khóa luận
11. Bố cục khóa luận

**Chương 1: Giới thiệu tổng quan**

**1.1 Giới thiệu về ngôn ngữ Java và công nghệ web java.**

**1.1.1 Khái niệm ngôn ngữ lập trình java**

Java là một ngôn ngữ lập trình do hãng Sun phát triển từ đầu những năm 90 đã trở thành một ngôn ngữ lập trình hướng đối tượng rất được ưa chuộng trong những năm gần đây. Java được sử dụng phổ biến trên toàn thế giới nhằm đáp ứng các yêu cầu phát triển ứng dụng trên mạng phục vụ cho nhiều người dùng với những môi trường thực thi phần mềm khác nhau.

Java là một ngôn ngữ lập trình hoàn chỉnh được thiết kế theo hướng đối tượng với đầy đủ các tính chất quan trọng của ngôn ngữ lập trình hướng đối tượng:

* Kế thừa.
* Đa hình.
* Đóng gói.
* Trừu tượng.

Ngày nay, khi nhắc đến Java người ta không còn chỉ nhắc đến Java như là một ngôn ngữ mà nhắc đến Java như là một công nghệ hay một nền tảng phát triển. Java đánh dấu sự trưởng thành của mô hình lập trình hướng đối tượng, nó được coi là một nền tảng mang tính cách mạng trong ngành phần mềm

**1.1.2 Lịch sử phát triển**

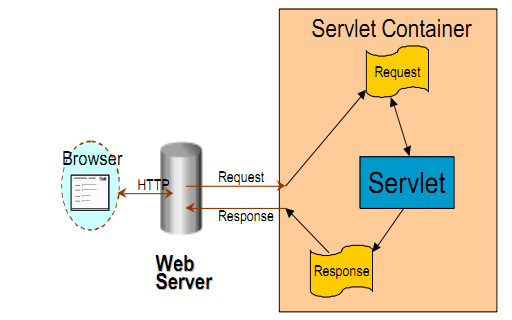
Lần đầu tiên xuất hiện vào năm 1992 là một ngôn ngữ dùng trong nội bộ tập đoàn Sun Microsystems để xây dựng ứng dụng điều khiển bộ xử lý bên trong máy điện thoại cầm tay, lò vi sóng, các thiết bị dân dụng khác . Không chỉ là ngôn ngữ java còn là một nền tảng phát triển và triển khai ứng dụng trong đó máy ảo java, bộ thông dịch có vai trò trung tâm Java servlet

### **1.1.3 Java servelet**

Servlet được tạo ra vào năm 1997, Servlet là các thành phần đối tượng nhúng trên trình chủ Web server thực hiện xử lý yêu cầu và sinh ra các trang Web động trả về máy khách.

Java Servlet là công nghệ nền tảng cho các công nghệ khác để tạo ra các ứng dụng Web có tính tương tác cao mà giao diện người dùng được sinh ra ở phía server. Một servlet là một lớp của Java xử lý động các yêu cầu và xây dựng các trả lời. Mặc dù servlet có thể dùng để trả lời bất kỳ một yêu cầu nào nhưng chúng thường dùng để hỗ trợ cho các ứng dụng Web.

Servlet là một thành phần ứng dụng Web không phụ thuộc vào hệ nền. Về bản chất thì Servlet là một chương trình Java (chỉ cần nạp một lần vào máy ảo) thường được sử dụng để chạy nền trên các máy ảo Java. Khi có một yêu cầu từ trình duyệt (Browser), máy chủ (server) sẽ phát lệnh thực hiện Servlet tương ứng, kết quả trả về của Servlet sẽ được máy chủ xử lý và trả về cho trình duyệt dưới dạng các trang HTML.

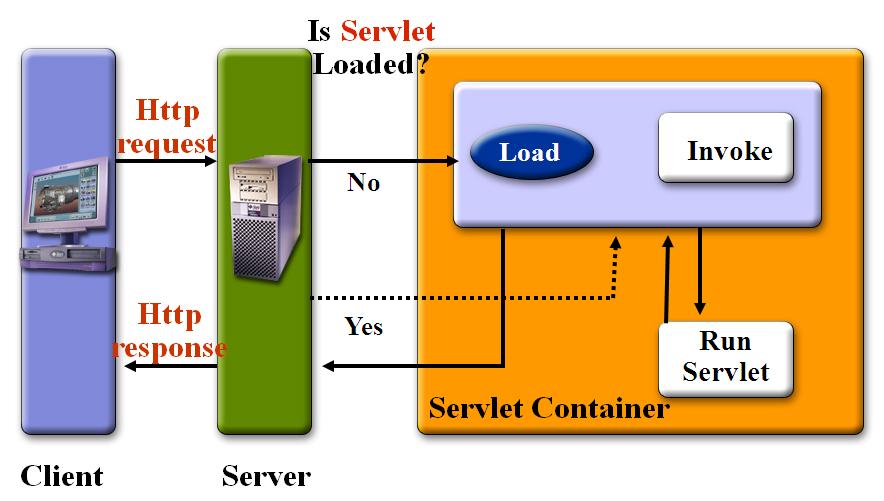


Hình 1: Mô hình Servlet Response and Request

Nhiệm vụ của Servlet:

* Nhận Client Request (hầu hết ở dạng HTTP Request).
* Trích xuất thông tin từ Request.
* Xử lý nghiệp vụ (Truy cập cơ sở dữ liệu, …).
* Tạo và gửi trả Response cho Client (hầu hết ở dạng HTTP Response) hoặc Forward cho Servlet hoặc trang web khác.
* Request là những thông tin được gửi từ Client đến Server.
* Response là những thông tin được gửi từ Server đến Client.

### **1.1.4 Vòng đời của một Servlet**



*Hình 2: Vòng đời Servlet*

**1.1.5 Java server page(JSP)**

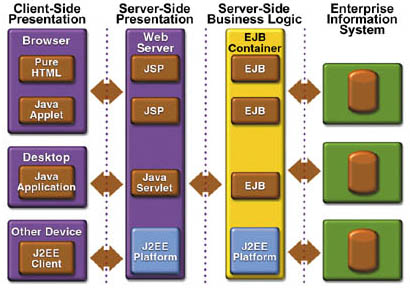
Công nghệ JSP kế thừa tất cả các kỹ thuật động của công nghệ Java Servlet, nhưng nó lại cung cấp một cách tiếp cận tự nhiên hơn trong việc tạo ra các thành phần Web tĩnh, JSP có thể kết hợp giữa công nghệ sinh trang HTML tĩnh với công nghệ động sinh ra các trang HTML. JSP là công nghệ lập trình Web phía máy chủ của Java, JSP cho phép truy xuất, xử lý đơn giản, giúp cho lập trình viên xử lý dễ dàng.

Đặc trưng của công nghệ JSP:

* Cho phép trộn giữa mã Java và các thẻ HTML.
* Xây dựng các đối tượng phía máy chủ.
* JSP sử dụng các truy xuất và xử lý đơn giản như : trình bày giao diện, định dạng trang HTML, triệu gọi các Javabean, servlet khác …

Nguyên tắc:

* JSP script được viết lồng vào các trang HTML.
* Khi có yêu cầu client, JSP Engine sẽ dịch JSP script thành mã Java.
* Java Compiler dịch Java code thành bytecode.
* Bytecode chạy trên máy ảo Java.



*Hình 3: JSP trong kiến trúc J2EE*

Với những lợi thế và những đặc điểm của mình dựa trên hướng mã nguồn mở và nhất là với sự ra đời của các framework nổi tiếng như : Struts, Hibernate, Spring, JSF, AJAX dựa trên nền tảng của công nghệ Java đã khuấy động lên một cuộc cách mạng trong lĩnh vực lập trình ứng dụng Web : “Chỉ trong một tương lai không xa, tất cả các ứng dụng phần mềm đều là ứng dụng Web”.

Công nghệ JSP được sử dụng để tạo ra các ứng dụng web. Nó tập trung nhiều hơn về mảng trình bày, giao diện của dứng dụng web, cách trình bày code của JSP khác với Servlets thêm code java trong hml. Trang JSP được chuyển đổi thành một Servlets nhờ container web. Container dịch một trang JSP vào tập tin class source.(java) và sau đó biên dịch thành một lớp Java Servlet.

JSP là cách đơn giản hóa hơn cho Servlet. Nếu như đối với Servlet phải viết mã Java và biên dịch bằng tay trước khi sử dụng với trình chủ Web server thì JSP không cần điều này. JSP viết mã Java tương tự Servlet nhưng cho phép trộn lẫn Java với các thẻ HTML để tạo ra Servlet xử lý các yêu cầu mà trình khác gửi đến trang.

JSP đơn giản và dễ sử dụng hơn Servlet mặc dù sau khi biên dịch và thực thi chúng chỉ là một.

# **Chương 2: Tìm hiểu Struts framework**

## **2.1 Mô hình MVC**

### **2.1.1 Mô hình lập trình MVC**

Đối với một trang JSP trước kia, bây giờ người ta tách làm ba phần Mô hình 1: Trước kia, khi lập trình một trang jsp, php hay asp. Do vậy có những khó khăn có thể gặp phải:

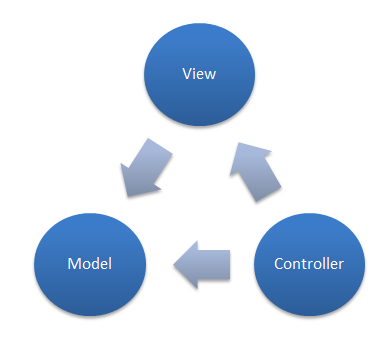
Người thiết kế giao diện cũng cần phải biết các ngôn ngữ lập trình hoặc bản thân chúng ta phải tự làm.

Việc bảo trì rất khó khăn vì một phần các mã chương trình lẫn lộn với các mã html

Khi có lỗi xảy ra, việc tìm ra lỗi và định vị lỗi cũng là một vấn đề khó khăn. Việc lập trình theo mô hình 1 còn được gọi là lập trình theo mô hình cổ điển từ đó mô hình MVC

Model-View-Control thay thế cho mô hình 1

Mô hình MVC (Model - View - Controller) là một kiến trúc phần mềm hay mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. Nó giúp cho các developer tách ứng dụng của họ ra 3 thành phần khác nhau Model, View và Controller. Mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác.



*Hình 4: Mô hình MVC*

Mô hình MVC tạo nên một nền tảng phát triển ứng dụng mạnh mẽ mà các công nghệ sau này đều ra đời trên nền tảng đó (Struts, Spring…)

**2.1.2 Cấu trúc mô hình MVC**



Mô hình cộng tác MVC



*Hình 5:* Mô hình cộng tác MVC

* Mô hình chia các conponent ứng dụng thành 3 loại khác nhau, Model-View-Controller. Mỗi componets đều đảm nhận một trách nhiệm nhất định.
* View: người dùng tương tác với ứng dụng thông qua components view. View có trách nhiệm gửi request tới controller sau đó nhận kết quả từ controller hiển thị cho người dùng html, jsp các thư viện tag, các file nguồn là các thành phần của components view.
* Controller: là trung gian giữa model và view. Controller có trách nhiệm nhận yêu cầu từ view. Sau khi nhân yêu cầu controller sẽ thực hiện business logic thích hợp từ model, sau đó điều khiển components View. Xuất dữ liệu cho User.ActionSevlet, Action, Actionform và Struts config.xml là các components của controller
* Model: Đảm nhận nhiệm vụ cung cấp dữ liệu từ cơ sở dữ liệu và lưu thông tin và lưu thông tin vào CSDL, tất cả các Business đều được thực thi ở Model

### **2.1.3 Định nghĩa từng thành phần trong MVC**

1. **Model:**

* Biểu diễn mô hình dữ liệu của ứng dụng và thực hiện xử lý tất cả các nghiệp vụ của hệ thống.
* Quán lý thông tin ứng dụng, thông tin này mô tả toàn bộ trạng thái của ứng dụng
* Model chứa các dữ liệu của ứng dụng và các chức năng làm nhiệm vụ xử lý những dữ liệu đó.
* Là một ánh xạ của thế giới thực, mô tả các thực thể mà hệ thống quản lý.
* Thực hiện các truy vấn trên cơ sở dữ liệu.
* Tính toán, xử lý nghiệp vụ hệ thống.
* Đóng gói miền Logic hệ thống mà độc lập hoàn toàn với việc trình bày trực quan.

**2. Controller:**

* Được thiết kế nhằm mục đích tạo nên các kết nối logic giữa những tương tác người dùng và xử lý nghiệp vụ của hệ thống.
* Nó nhận và vận chuyển các đầu vào là những thông tin nhập từ người dùng đến Model hoặc View.
* Những thông tin đầu vào và những chỉ thị người dùng được truyền tới View và Model nhờ việc thực hiện các Action.
* Controller cũng có trách nhiệm dựa vào những thông tin đầu vào và các kết quả xử lý nghiệp vụ hệ thống để quyết định đầu ra là trang nào trong View.
* Ánh xạ giữa các Action đến các tài nguyên của ứng dụng.

**3. View**

* Lấy dữ liệu từ Model và biểu diễn đến người dùng.
* Biểu diễn các đầu vào, đầu ra của ứng dụng.
* Hiển thị kết quả xử lý nghiệp vụ hệ thống.
* Có thể truy xuất tới Model. Đây là sự mềm dẻo của mô hình MVC.
* Việc truy xuất đến Model được thực hiện nhờ các câu lệnh truy vấn.

**Mối quan hệ giữa View-Controller-Model**

View và Controller : Controller có nhiệm vụ tạo ra hoặc lựa chọn View

Model và Controller : Controller dựa vào Model. Nếu Tại Model có một sự thay đổi thì có thể sẽ dẫn theo sự thay đổi ở Controller.

Model và View : View dựa vào Model. Nếu tại Model có một sự thay đổi thì có thể sẽ dẫn theo sự thay đổi ở View.

**2.1.4.Ưu điểm và hạn chế của MVC**

* + - * 1. Ưu điểm.

Giúp người sử dụng phân tách ứng dụng một cách rõ ràng

Mô hình đơn giản dễ dàng áp dụng và xử lý với các dự án nhỏ, mang đến yếu tố lâu dài cho người sử dụng.

Không phụ thuộc môi trường, nền tảng xây dựng hay ngôn ngữ lập trình phát triển.

* + - * 1. Hạn chế.

Nạp, load nhiều thư viện đồ sộ làm cho mô hình trở nên chậm chạp

Sử dụng mảng thành phần chính cho việc truy xuất dữ liệu. Nhất là active record để viết ứng dụng. Yêu cầu người sử dụng phải nắm vững mảng đa chiều.

Xây dựng cầu kì, mất thời gian để xây dựng thư viện, cấu trúc.

## **2.2 Struts framework**

### **2.2.1 Giới thiệu Struts framework**

### **2.2.1.1 Khái niệm Struts framework**

Struts là một framework phục vụ việc phát triển các ứng dụng Web trên Java. Sử dụng mẫu thiết kế Model-View-Controller (MVC), Struts giải quyết rất nhiều các vấn đề liên quan đến các ứng dụng Web hướng business đòi hỏi hiệu năng cao sử dụng Java servlet và JSP. Struts cơ bản định hình lại cách các Web programmer nghĩ về và cấu trúc một ứng dụng Web.  
Structs được phát triển bởi Craig McClanahan và được bão trợ bởi Apache, nhóm Jakarta

**2.2.1.2 Lịch sử phát triển**

Struts nguồn gốc được tạo ra bởi Craig R. McClanahan và sau đó được chuyển giao cho dự án Jakarta của Apache Software Foundation (AFS) vào năm 2000. Vào tháng 6 năm 2001, Struts đã phát hành phiên bản 1.0. Sau đó, có rất nhiều người đã tham gia đóng góp cả về mã nguồn và tài liệu cho dự án và Struts ngày càng trở nên hoàn thiện hơn.

Struts là một framework mã nguồn mở được phát triển trên nền tảng của ngôn ngữ Java, JSP nhằm mục đích tạo ra một thành phần chuẩn trong việc thiết kế ở tầng Web sử dụng MVC dựa trên nền tảng của J2EE. Với Struts, người lập trình sẽ không phải quan tâm đến việc xây dựng mô hình MVC như thế nào mà chỉ cần cấu hình và chạy tầng hiển thị của dự án, điều này cho phép giảm thiểu chi phí đào tạo, thời gian phát triển dự án nhanh hơn và cho phép hệ thống chạy ổn định trên các nền và hoặc module tích hợp khác nhau. Với Struts ta có thể tách làm 3 tầng làm việc riêng biệt một cách dễ dàng, thích hợp cho một dự án lớn, không bị nhập nhằng giữa việc tạo giao diện, xử lý và truy xuất cơ sở dữ liệu.

Struts hiện có hai phiên bản chính, đó là Struts 1. xx và 2.0x. Struts 1. xx được công nhận như một framework phổ biến nhất trong Java, Struts 1. xx đủ chín muồi và là một giải pháp tốt cho nhóm phát triển để giải quyết các vấn đề chung nhất. Struts 2.0x, kế thừa nền tảng của webwork framework, được tổ chức như một nền làm việc chuyên nghiệp với web, nhằm giải quyết các vấn đề phức tạp, mang tính nghiệp vụ cao.

**2.2.1.3 Tại sao phải cần một framework**

* Model hóa: có khả năng phân thành các thành phần khác nhau, tạo ra một khung sườn làm việc
* Dễ mở rộng: đầu tiên phát triển trên web sau đó tiếp tục phát triển trên mobile
* Giảm thiếu sự ràng buộc giữa các thành phần

**2.2.1.4 Nhược điểm của Framework**

* Ứng dụng có dung lượng nặng do chứa các class cũng như interface không dùng đến nhưng vẫn có trong ứng dụng.

**2.2.3 Struts 1 Framework**

**2.2.3.1 Sơ lược về Struts 1**

Struts 1 được thiết kế dựa trên mô hình MVC. Dùng để xây dựng các ứng dụng web sử dụng sử dụng Java Servlet và Jsp. Ứng dụng Struts chạy được trên nhiều nền tảng, dễ học, dễ sử dụng, có kèm các thư viện thẻ cho việc phát triển ứng dụng nhanh chóng hơn.

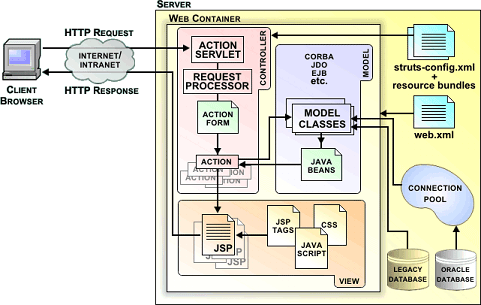
Struts cho phép các lập trình viên:

* Phát triển ứng dụng web Jsp/Servlet theo mô hình MVC
* Sử dụng các đối tượng có sẵn của Framework trong file cấu hình Xml
* Sử dụng các mẫu thiết kế xây dựng sẵn của Framework
* Sử dụng nhiều tính năng nâng cao (Validation, …)

Những thuận lợi khi sử dụng:

* Struts đảm nhiệm những phần phức tạp nhất khi xây dựng ứng dụng MVC
* Dễ học, dễ sử dụng
* Nhiều tính năng được hỗ trợ
* Nhiều công cu hỗ trợ của bên thứ 3
* Linh động và dễ mở rộng
* Có một cộng đồng đông đảo
* Ổn định và không ngừng phát triển
* Mã nguồn mở hoàn toàn
* Tích hợp tốt với J2EE
* Hỗ trợ một thư viện thẻ cực kỳ hữu dụng
* Tích hợp với Tiles Framework

**2.2.3.2 Kiến trúc struts 1**

****

*Hình 6: Kiến trúc Struts 1*



*Hình 7: Sơ đồ làm việc cộng tác của Struts 1*

Controller là trung tâm điều hành của luồng ứng dụng, nó ủy quyền cho các bộ xứ lý thích hợp gọi là các Action.

Các Action có thể sử dụng các thành phần của Model.

Model đóng gói các trạng thái hoặc logic nghiệp vụ.

Điều khiển được chuyển tiếp cho Controller hoặc View phù hợp.

Struts bao gồm ba thành phần chính:

* Servlet controller (Controller)
* Jsp hay mọi công nghệ hiển thị nào khác (View)
* Application Business Logic ở một dạng bất kỳ phù hợp với ứng dụng (Model)

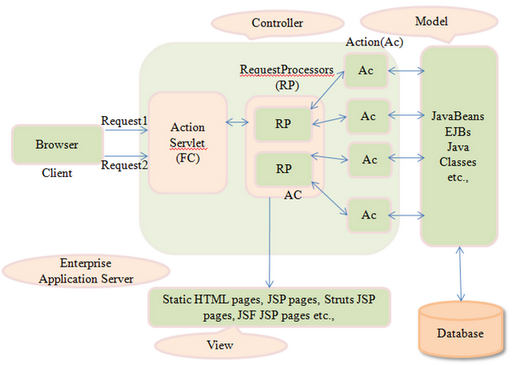
Tuy nhiên, Struts tập trung chủ yếu vào phần Controller. Có sự độc lập giữa Model và View. Người phát triển có thể sử dụng bất kỳ công nghệ Model và View nào.

File cấu hình của Struts (struts-config.xml) chứa các Action mapping. Ánh xạ URL với các Action. Controller sử dụng các mapping này cho hai mục đích chính: trả về các HTTP Request cho Action của ứng dụng, xác định điều hướng trang.

**2.2.3.3 Các thành phần trong Struts 1**

2.2.2.3.1 Controller

Giống như một Switch trong kiến trúc MVC. Mọi Request đều phải đi qua Controller. Chịu trách nhiệm điều khiển luồng cho các xử lý Request (action mapping) bằng cách đọc file cấu hình của ứng dụng để xác định luồng điều khiển.



*Hình 8: Các thành phần của Controller*

Trong đó, Struts đã cung cấp sẵn thành phần Action Servlet (org.apache.struts.action.ActionServlet). Cần phải cấu hình Servlet mapping cho Servlet này trong file web.xml.

Các cấu hình liên quan đến Struts khác được tạo lập trong file struts-config.xml.

File này làm nhiệm vụ mapping URL của Request đến Action cụ thể.

Như vậy nhiệm vụ của lập trình viên là:

* Viết lớp Action (kế thừa từ lớp org.apache.struts.action.Action có sẵn trong Struts Framework) cho mỗi Request có thể nhận được. Trong đó phải ghi đè phương thức execute().
* Viết Action mapping trong file cấu hình ứng dụng struts-config.xml.
* Cập nhật file web.xml, nhằm thực hiện mapping cho ActionServlet.

2.2.2.3.2 Model

Model được chia làm hai thành phần:

* Internal state: Trạng thái trong của hệ thống.
* Business logic: Các phương pháp thay đổi các trạng thái đó.

Internal state của Model có thể được biểu diễn bằng:

* Javabeans
* Enterprise Javabeans
* POJO’s
* JDO
* JDBC

2.2.2.3.3 View

Thành phần View chứa các file Jsp cho từng ứng dụng. Để dễ dàng cho quá trình phát triển Strust có sẵn một thư viện Jsp custom tags.

Chúng ta cũng có thể tạo nên các Resource file trong View cho nhiều mục đích khác nhau chẳng hạn tạo ứng dụng đa ngôn ngữ.

Nhiệm vụ của thành phần View là tạo form cho các ứng dụng, phối hợp hoạt động với Controller Servlet

Đối tượng ActionForward sẽ thông báo với Servlet Controller trang Jsp nào sẽ được trả về.

Các trang Jsp sử dụng ActionForm để lấy dữ liệu output từ Model để hiển thị.

Như đã trình bày ở trên Struts có một bộ thư viện thẻ riêng điều đó khiến các lập trình viên dễ phát triển các nội dung động hơn, đơn giản cho việc trao đổi giữa Html Designer và lập trình viên.

**2.2.3.4 Xây dựng các thành phần Struts 1**

2.2.2.4.1 Xây dựng thành phần Model

2.2.2.4.1.1 JavaBeans và phạm vi

Trong một ứng dụng web. Các JavaBeans có thể được lưu trữ và truy cập từ các phạm vi khác nhau. Mỗi phạm vi đều có những quy tắc riêng cho vòng đời của nó cũng như khả năng có thể truy cập của các bean được lưu trữ tại đây.

Java Server Page (Jsp) định nghĩa các phạm vi của nó bằng những từ như sau: (Những khái niệm tương ứng được định nghĩa trong Servlet trình bày trong dấu ngoặc)

* Page: Các bean có thể được truy cập trong một trang, trong vòng đời của Request hiện tại (Các biến cục bộ trong phương thức service()).
* Request: Các bean có thể được truy cập trong một trang Jsp cũng như bất kỳ trang hoặc Servlet nào được included trong trang Jsp này, hoặc được Forward từ nó (Thuộc tính Request).
* Session: Các bean có thể được truy cập trong các trang Jsp và các Servlet trong session của một User, trên một hoặc nhiều Request (Thuộc tính session).
* Application: Các bean có thể được truy cập trong tất cả các trang Jsp hoặc các Servlet trong cả ứng dụng web (Thuộc tính Servlet context).

Một điều rất quan trọng là các trang Jsp và Servlet trong cùng ứng dụng web chia sẻ cùng một tập các beans.

2.2.2.4.2 ActionFormBeans

ActionForm là một tính năng quan trọng và hữu ích trong Struts Framework, đây cũng là một thành phần dễ gây nên sự hiểu nhầm cho người sử dụng Strust Framework. Thông thường, ActionForm thường là một thành phần của Controller nhưng vì nó được sử dụng để mô hình dữ liệu, nên có những nét đồng nhất với Model.

Khi xây dựng một form hiển thị tới người dùng, Strust yêu cầu phải xây dựng một ActionForm đi kèm form đó. Chính vì vậy mà nó mới có tên gọi là ActionForm. Tuy nhiên, một ActionForm có thể được sử dụng cho nhiều form, hoặc toàn ứng dụng. Khi ActionForm được định nghĩa trong file cấu hình hệ thống struts-config.xml thì Controller sẽ tự động thực hiện các dịch vụ như sau trước khi gọi đối tượng Action thích hợp:

* Kiểm tra một thể hiện của bean được định nghĩa kèm với form gửi Request tới Controller trong phạm vi thích hợp.
* Nếu như không tồn tại bean thỏa mãn, Controller tự động tạo ra một bean theo đúng định nghĩa của lập trình viên và thêm vào phạm vi thích hợp.
* Với tất cả các tham số trong Request gửi tới Controller mà có tên tương ứng với tên của một thuộc tính trong bean, phương thức setter tương ứng sẽ được gọi.
* Sau khi được cập nhật, các ActionForm bean sẽ được truyền đến phương thức execute() của lớp Action. Do đó các giá trị này có thể được sử dụng để tác động lên trang thái trong của hệ thống hoặc phục vụ cho các xử lý logic nghiệp vụ nào đó.

Phải lưu ý rằng một form ở đây không đơn giản chỉ tương ứng với một trang Jsp duy nhất trong giao diện người dùng. Một cách sử dụng rất phổ biến là chỉ một form mà sử dụng với nhiều trang Jsp khác nhau. Điều này có những lợi ích lớn trong những trường hợp đặc biệt như tạo một wizard cho việc cài đặt một ứng dụng mới. Struts khuyến khích các lập trình viên nên tận dụng được một ActionForm bean cho nhiều trang jsp. Điều đó giúp chương trình đơn giản hơn, dễ kiểm soát hơn. Khi nhà thiết kế muốn thay đổi lại về các trường dữ liệu hiển thị thì cũng không quá phức tạp và khó khăn cho việc sửa code bên trong.

Đối với các ứng dụng nhỏ thì chỉ cần một ActionForm cho mọi form nhập vào của người dùng, cũng có người muốn sử dụng một ActionForm cho một số form chính riêng của họ, có người lại muốn sử dụng nhiều ActionForm, mỗi ActionForm tương ứng với một form nhập vào. Số lượng ActionForm hoàn toàn chỉ phụ thuộc vào lập trình viên.

2.2.2.4.3 System State Beans

Trạng thái thực tế của một hệ thống thường được biểu diễn như là một hoặc nhiều Java Bean mà có các thuộc tính định nghĩa trạng thái hiện thời của hệ thống.

Với các hệ thống quy mô nhỏ, hoặc những thông tin của hệ thống mà không cần thiết phải lưu trữ trong một thời gian dài, một tập hợp các bean có thể được sử dụng để lưu trữ các thông tin chi tiết đó. Với các hệ thống quy mô lớn hơn, hoặc trường hợp các thông tin hệ thống mà bắt buộc phải lưu trữ trong một thời gian dài. Thì giải pháp là nên lưu trữ lại trong cơ sở dữ liệu. Và vẫn cần thiết phải có một tập các bean dùng để đại diện cho các thông tin đó.

2.2.2.4.4 Business Logic Beans

Chúng ta nên đóng gói các chức năng logic của ứng dụng như các phương thức gọi trên các JavaBean phục vụ cho mục đích này. Các phương thức này có thể là một phần của các lớp định nghĩa trạng thái hệ thống hoặc là thành phần của các lớp riêng biệt được thiết kế riêng cho mục đích xử lý logic nghiệp vụ hệ thống. Trong trường hợp thứ hai, các trạng thái hệ thống luôn luôn phải được truyền vào làm tham số cho các phương thức thực hiện xử lý logic nghiệp vụ.

Nhằm mục đích có thể sử dụng lại tối đa mã nguồn. Business Logic Bean nên được thiết kế và thực thi sao cho người lập trình viên của Model cũng không biết được họ đang thực hiện trên một môi trường ứng dụng web. Sau khi lớp Action của chúng ta với một phần vai trò điều khiển dịch tất cả các thông tin cần thiết từ Http Request đã được gửi tới nó, gọi Model xử lý logic nghiệp vụ và gọi setter để lưu lại các thông tin trạng thái hệ thống. Một lớp business logic này có thể được sử dụng lại trong một môi trường khác, không chỉ trong pham vi ứng dụng web mà chúng ta đang xây dựng nên.

2.2.2.4.5 DynaBeans

DynaBeans là một sự mở rộng của JavaBeans với nguyên lý ánh xạ. Việc định nghĩa một JavaBeans cho dù đơn giản nhất gồm những yêu cầu: tạo một lớp mới, mã hóa cho các thuộc tính bằng hai phương thức getter, setter. Đối với DynaBeans việc định nghĩa nó có thể thông qua một mô tả Xml mà vẫn đầy đủ những tính chất của JavaBeans thông thường. Các thuộc tính của một DynaBeans không thể được gọi bằng các phương thức Java chuẩn cho Bean.

Trong ứng dụng web Struts, người ta có thể sử dụng DynaBeans cho việc mô tả form Html. Chiến lược này sẽ tránh được việc phải viết một lớp JavaBeans để lưu trữ những thuộc tính đơn giản.

2.2.2.5 Xây dựng thành phần View

2.2.2.5.1 Sự tương tác Form và FormBean

Thành phần View thông thường sử dụng công nghệ Jsp để trình bày thông tin đến người dùng, tuy nhiên bên cạnh đó người ta hoàn toàn có thể sử dụng các công nghệ khác cho mục đích này mà có thể kết xuất thông qua JavaServlet.

2.2.2.5.2 Form tự động tương tác bean

Nói đến một ứng dụng web, tại một lúc này hay lúc khác, hầu hết các nhà phát triển web xây dựng các form bằng các thẻ chuẩn của Html (<input>). Người sử dụng luôn mong đợi tương tác với các ứng dụng web để có được những hành vi nhất định.

Một trong những sự kỳ vọng của người dùng là về điều khiển lỗi. Khi người dùng gây ra một lỗi nào đó, ứng dụng cần cho họ sửa chữa lại nhưng chỉ sửa lại những thông tin bị sai mà không cần phải nhập lại những thông tin còn lại trên form hay trên trang.

Để thực hiện điều này đối với những chuẩn Html và Jsp thông thường cực kỳ cồng kềnh và khó chịu (phải nhúng thẻ <% %> của Jsp vào mã Html). Bởi các nhà thiết kế giao diện có thể không biết tới lập trình, không biết tới Framework, khi đó vấn đề này trở nên khó khăn. Vì thế Struts đưa ra một thư viện các thẻ mới riêng của Struts dựa trên Custom Tag Library Jsp. Các thư viện thẻ này giải quyết được các vấn đề giữa nhà phát triển và nhà thiết kế như nêu trên một cách tự động hoàn toàn.

2.2.2.5.3 Form tự động validate

Ngoài sự tương tác động giữa form và bean đã được mô tả ở trên, Framework còn cung cấp một cơ chế cho phép xác nhận các thông tin đầu vào được người dùng cung cấp.

Để sử dụng tính năng này, tại ActionForm cần phải Override phương thức validate(ActionMapping mapping, HttpServletRequest request); Phương thức này được gọi bởi Controller Servlet sau khi sự tương tác giữa form và bean đã hoàn thành, trước khi phương thức execute() của Action được gọi.

Phương thức validate gồm các tùy chọn sau:

* Thực hiện kiểm chứng một cách phù hợp để tìm ra các vấn đề do người dùng nhập thông tin. Trả về hoặc null hoặc một thể hiện của ActionErrors chứa chuỗi các số 0. Controller sẽ điều khiển để gọi execute() của Action một cách phù hợp.
* Thực hiện kiểm chứng một cách phù hợp để tìm ra các vấn đề do người dùng nhập thông tin. Trả về một thể hiện của ActionErrors chứa các ActionMessages (đó là các lớp có chứa các thông báo lỗi được lưu trong một tập tin tài nguyên của ứng dụng). Servlet điều khiển lưu trữ mảng này như một thuộc tính của Request được sử dụng bởi thẻ  <html:errors> và sau đó sẽ được chuyển tiếp đến form nhập thông tin của người dùng.

Việc sử dụng tính năng này là hoàn toàn tùy chọn, mặc định Controller hiểu phương thức validate này trả về giá trị null. Và Controller coi tất cả công việc validate dữ liệu nhập vào bởi người dùng sẽ được thực hiện ở Action.

2.2.2.5.4 The Struts Validator

Ngoài các cách validate bằng javascript hay validate bằng ActionForm, bằng Action, Struts còn cung cấp một cách khác để validate một cách nhanh chóng và thuận lợi. Đó là phương pháp cấu hình Validator.

Việc cấu hình có những quy tắc cần phải tuân thủ như sau:

* ActionForm Bean phải kế thừa lớp ValidatorForm
* Html form phải bao gồm các thẻ <html:javascript> để validate phía client.
* Lập trình viên phải định nghĩa các quy tắc validate trong một file xml.
* Cuối cùng, lập trình viên phải kích hoạt ValidatorPlugin trong file cấu hình hệ thống struts-config.xml.

Chú ý nếu thuộc tính trên form mà là một đối tượng đại diện cho một kiểu dữ liệu nguyên thủy (chẳng hạn: int) thì lập trình viên cần phải thiết lập ActionServlet chuyển đổi giá trị null khi khởi tạo. Nếu không validate trên trường này có thể không được thực hiện bởi nó luôn có một giá trị mặc định (chẳng hạn với kiểu int, giá trị mặc định là : 0).

2.2.2.5.5 Bố cục trang với Tiles

Tiles là một thư viện có sẵn hỗ trợ mạnh mẽ cho việc phát triển thành phần View của ứng dụng Struts. Tiles cho phép nhà phát triển xây dựng View bằng cách kết hợp nhiều tiles với nhau. Tiles là một Framework mã nguồn mở cho phép tạo trang trình diễn dữ liệu đến người dùng một cách đơn giản và tiện lợi. Nó giúp giảm thiểu công việc cho lập trình viên, tránh tình trạng phải viết những đoạn mã lặp đi lặp lại nhiều lần. Tiles là thực sự hữu ích cho việc xây dựng các trang với giao diện “Look and feel”. Giúp dễ dàng sửa đổi và duy trì giao diện trang web.

Tiles được xây dựng dựa trên tính năng “Include” được cung cấp bởi Java Server Pages. Như vậy chúng ta có thể dễ dàng xây dựng các trang có thể sử dụng lại. Nó cung cấp đầy đủ các tính năng và sự mạnh mẽ để lắp ghép thành trang trình bày dữ liệu từ các thành phần riêng lẻ. Mỗi phần (Tiles) có thể được tái sử dụng tùy ý theo sự cần thiết của thành phần đó trong suốt ứng dụng Struts. Điều này làm giảm số lượng thẻ Html cần phải duy trì và làm cho trang web dễ dàng thay đổi giao diện.

Một bố cục tiles là một trang Jsp mà cho phép các tiles có thể được đặt vào. Bố cục tiles xác định rõ nơi nào mà tiles có thể được đặt vào. Tiles Framework sử dụng một tập tin cấu hình xml để tổ chức các tiles.

2.2.2.5.6 Bộ thư viện thẻ xây dựng sẵn

Bộ thư viện này được chia thành ba loại chính:

* Các thẻ với tiếp đầu ngữ là Html:

html:form

html:img

html:image

html:errors

html:link

html:multibox

html:radio

html:checkbox

html:text

html:select

html:options

* Các thẻ với tiếp đầu ngữ Bean:

bean:write

bean:define

* Các thẻ với tiếp đầu ngữ Logic

logic:iterator

logic:match

logic:equal

logic:notEqual

2.2.2.6 Xây dựng thành phần Controller

2.2.2.6.1 Tổng quan

Thành phần chính của Controller là một Servlet thực hiện các chức năng chính của việc ánh xạ các yêu cầu qua các URL với những Action tương ứng.

Nhiệm vụ chính của một lập trình viên liên quan đến thành phần Controller này là:

* Viết một lớp ActionForm làm trung gian giữa Model và View
* Viết một lớp Action cho mỗi yêu cầu xử lý
* Cấu hình AcionMapping (trong file cấu hình ứng dụng struts-config.xml) cho mỗi yêu cầu có thể nhận được.
* Cập nhật tập tin web.xml. Tập tin mô tả triển khai ứng dụng

2.2.2.6.2 ActionServlet

ActionServlet là thành phần quan trọng nhất trong Controller. Nó đóng vai trò là một bộ điều khiển. Công việc của ActionServlet như sau:

Nhiệm vụ của ActionServlet là:

* Thực hiện các vai trò của Controller như: xử lý các user request, xác nhận xem user cần gì thông qua request nhận được, lấy dữ liệu từ Model nếu cần thiết lấy ra View phù hợp trả về cho người dùng.
* Thực tế ActionServlet ủy nhiệm hầu hết các công việc này cho Action.
* Ngoài ra ActionServlet còn chịu trách nhiệm khởi tạo và dọn dẹp tài nguyên: như load các cấu hình ứng dụng trong file web.xml, duyệt danh sách init-param, đánh dấu các modun …

2.2.2.6.3 Request Processor

Là nơi xảy ra hầu hết những xử lý cốt lõi cho mỗi Request. Để xử lý một Request được gửi tới cho Controller, Request đó phải được đi qua từng bước gọi là một chuỗi các xử lý (chain).

Mỗi bước tương ứng với một lệnh áp dụng lên Request. Các lệnh đó là các phương thức trong các lớp thuộc gói org.apache.struts.chain.commands hoặc org.apache.struts.chain.commands.servlet.

Xem xét từng lệnh một trong chuỗi xử lý đó:

* SelectLocale: Chọn một Locale cho request nếu Locale chưa có và đặt nó vào Request.
* SetOriginalURI: Lưu trữ URI của Request vào Request.
* RequestNoCache: Thiết lập tiêu đề cho phản hồi (giao thức HTTP): "Pragma", "Cache-Control", and "Expires".
* RemoveCachedMessages: Loại bỏ đi bất kỳ đối tượng ActionMessages nào được lưu trữ trong Session dưới Globals.MESSAGE\_KEY và Globals.ERROR\_KEY nếu phương thức isAccessed của Messages trả về true.
* SetContentType: Thiết lập kiểu nội dung mặc định cho mọi Response của Request.
* SelectAction: Xác định ActionMapping của đường dẫn này.
* AuthorizeAction: Nếu mapping có một vai trò liên kết đến nó, phương thức này đảm bảo Request Client có vai trò đó. Nếu không, đưa ra thông báo lỗi và dừng quá trình xử lý Request tại đây.
* CreateActionForm: Khởi tạo (nếu cần thiết) một ActionForm được liên kết đến Mapping này (nếu có) và đặt nó vào phạm vi thích hợp.
* PopulateActionForm: Thực hiện tương tác giữa form và ActionForm nếu có.
* ValidateActionForm: Thực hiện validate (nếu được yêu cầu) tại ActionForm được liên kết đến Request này (nếu có).
* SelectInput: Nếu validate không thành công, chọn liên kết ForwardConfig thích hợp để trở lại trang đầu vào.
* ExecuteCommand: Tra cứu và thực thi một chuỗi lệnh nếu ActionConfig được cấu hình.
* SelectForward: Nếu mapping này đại diện cho một Forward, forward đến đường dẫn được chỉ định bởi mapping.
* SelectInclude: Chọn URI thêm (nếu có) cho ActionMapping hiện tại.
* PerformInclude: Thêm vào các kết quả của việc gọi đường dẫn trong Request.
* CreateAction: Khởi tạo một thể hiện của lơp được chỉ định bởi ActionMapping hiện tại (nếu cần thiết).
* ExecuteAction: Đây là điểm mà phương thức execute của Action sẽ được thực thi.
* ExecuteForwardCommand: Tra cứu và thực thi một chuỗi lệnh nếu ForwardConfig được cấu hình.
* PerformForward: Cuối cùng, phương thức này của RequetsProcessor sẽ lấy ActionForward được trả về bởi phương thức execute của lớp Action và sử dụng nó để chọn tài nguyên tiếp theo (nếu có).

2.2.2.6.4 Các lớp trong ActionForm

Một ActionForm đại diện cho một form Html mà người dùng tương tác với một hoặc nhiều trang. Lập trình viên cần cung cấp cho ActionForm các thuộc tính để lưu trữ lại trạng thái của trang với các phương thức setter và getter của chúng. ActionForm có thể được lưu trữ trong phạm vi session (mặc định) hay request. Nếu chúng ở trong session thì rất cần thiết phải ghi đè phương thức reset để khởi tạo lại form mỗi lần sử dụng. Framework thiết lập các thuộc tính của ActionForm từ các thông tin trong request và sau đó gửi form đã được validate tới phương thức execute của Action phù hợp được liên kết với nó trong ActionMapping.

Các nguyên tắc cần tuân thủ khi tạo ActionForm Bean:

* Lớp ActionForm không yêu cầu phải tạo ra các phương thức thực thi cho nó. ActionForm bean chỉ bao gồm các phương thức setter và getter, mà không liên quan đến quá trình xử lý logic.
* Các đối tượng ActionForm cũng cung cấp một cơ chế validate chuẩn, nếu chúng ta ghi đè một phương thức có sẵn của Framework và cung cấp các tin nhắn báo lỗi trong tài nguyên chuẩn của ứng dụng. Khi đó Framework sẽ tự động validate các thông tin đầu vào của form bằng cách sử dụng phương thức của chúng ta. Và tất nhiên chúng ta có thể bỏ qua cách xác nhận này và cung cấp cách cách thức khác như: dùng Javascript hay sử dụng lớp Action.
* ActionForm định nghĩa các thuộc tính được liên kết đến các phương thức setter và getter cho mỗi trường được đại diện trên form. Tên trường cũng như tên các thuộc tính phải phù hợp với quy tắc của JavaBean thông thường.
* Các nút cũng như các điều khiển khác đều có thể định nghĩa như các thuộc tính. Điều này có thể giúp xác định các nút nào hoặc các điều khiển nào trên form đã được lựa chọn khi form này được submit.
* Có thể coi ActionForm như một bức tường lửa giữa HTTP và Action. Tại tường lửa này cần sử dụng phương thức validate để chắc chắn rằng tất cả các thuộc tính được yêu cầu là tồn tại và các giá trị của chúng là hợp lý. Một ActionForm phát hiện lỗi trong quá trình validate sẽ không được tiếp tục gửi tới Action nữa.
* Chúng ta có thể đặt một ActionForm bean trên form và sau đó sử dụng các tham chiếu thuộc tính lồng. Để làm được điều này chúng ta phải sử dụng thư viện thẻ của Struts.

2.2.2.6.5 ***.*** DynaActionForm

Việc duy trì một ActionForm riêng biệt cụ thể cho mỗi form của chúng ta là rất tốn thời gian. Ngoài ra chương trình sẽ ngày càng phức tạp, cồng kềnh, khó kiểm soát khi quá nhiều ActionForm được tạo ra. Struts cung cấp cho chúng ta một giải pháp khác để tạo ra các ActionForm mà không cần phải tạo lớp mới kế thừa từ ActionForm và các phương thức setter, getter rườm rà mà chỉ cần cấu hình trong file cấu hình ứng dụng struts-config.xml.

Danh sách các kiểu dữ liệu được hỗ trợ cho DynaActionForm

* java.math.BigDecimal
* java.math.BigInteger
* boolean và java.lang.Boolean
* byte và java.lang.Byte
* char và java.lang.Character
* java.lang.Class
* double và java.lang.Double
* float và java.lang.Float
* int và java.lang.Integer
* long và java.lang.Long
* short và java.lang.Short
* java.lang.String
* java.sql.Date
* java.sql.Time
* java.sql.Timestamp

Ngoài các kiểu dữ liệu liệt kê trên, chúng ta cũng có thể sử dụng các kiểu dữ liệu mảng hoặc cũng có thể chỉ định inplements các interface Map như java.util.HashMap hoặc inplements các List như java.util.ArrayList.

Nếu chúng ta không cung cấp giá trị khởi tạo cho các thuộc tính này, mặc định giá trị khởi tạo của các kiểu số sẽ là 0, và kiểu đối tượng là null.

Ở trang Jsp bằng cách sử dụng bộ thư viên thẻ Struts, các thuộc tính của DynaActionForm cũng được tham chiếu tới như các ActionForm thông thường. Ở bất cứ nơi nào một thẻ struts với một thuộc tính “property” sẽ tự động sử dụng các thuộc tính của DynaActionForm. Thậm chí còn có thể sử dụng thẻ <bean:define> để xác định các thuộc tính của một DynaActionForm một cách trực quan.

Các thuộc tính map là một thuộc tính của DynaActionForm mà đại diện cho một HashMap có chứa các thuộc tính của DynaActionForm.

DynaActionForm có ý nghĩa như là một giải pháp dễ dàng cho một vấn đề chung: ActionForm chứa các thuộc tính và phương pháp để validate dữ liệu chuẩn.

DynaActionForm không phải là một sự thay thế cho ActionForm. Nếu muốn truy cập đến các thuộc tính của DynaActionForm chúng ta cần sử dụng map-style accessor. Nếu muốn sử dụng các thuộc tính một cách chủ động hơn thì chúng ta phải sử dụng ActionForm thông thường.

Chúng ta có thể mở rộng thêm các phương thức validate và reset nếu cần thiết.

Để sử dụng DynaActionForm với Validator Struts, phải chỉ định org.apache.struts.validator.ValidatorActionForm như là lớp form-bean.

Ngoài các loại ActionForm trên, Framework còn cung cấp một số loại ActionForm khác như: LazyActionForm, ListBackActionForm, MapBackActionForm tuy nhiên thực tế những loại ActionForm này ít được sử dụng.

2.2.2.6.6 Các lớp Action

Lớp Action định nghĩa hai phương thức quan trọng, là hai phương thức nạp chồng:

public ActionForward execute(ActionMapping mapping,

ActionForm form,

ServletRequest request,

ServletResponse response)

throws Exception;

public ActionForward execute(ActionMapping mapping,

ActionForm form,

HttpServletRequest request,

HttpServletResponse response)

throws Exception;

Nhưng hầu hết các dự án sẽ sử dụng phiên bản HttpServletRequest. Phương thức non-HTTP chỉ được sử dụng cho các ứng dụng không sử dụng giao thức Http.

Mục đích của một lớp Action là để xử lý Request thông qua phương thức execute của nó và trả về một đối tượng ActionForward mà định danh nơi điều khiển sẽ được chuyển đến, cung cấp một phản hồi phù hợp. Một lớp Action thông thường sẽ thực hiện những công việc sau trong phương thức execute của nó:

* Xác nhận trạng thái hiện tại của phiên làm việc. Ví dụ: kiểm tra người dùng đã đăng nhập thành công chưa. Nếu như lớp Action tìm thấy tài khoản đăng nhập có thể không tồn tại, yêu cầu có thể được chuyển tiếp đến trang trình bày hiển thị yêu cầu nhập lại tên đăng nhập và mật khẩu. Điều này xảy ra khi một người dùng chưa đăng nhập hệ thống mà cố gắng truy nhập vào một trang nào đó của ứng dụng mà trang này yêu cầu người dùng phải đăng nhập mới có thể vào được, hoặc session của người dùng đã hết hạn và Servlet container đã tạo ra một session mới.
* Thực hiện các xử lý cần thiết để đáp ứng cho yêu cầu của người dùng. Điều này có thể thực hiện được bằng cách nhúng các mã xử lý logic vào phương thức execute của Action. Tuy nhiên, nhận thấy rằng, các mã xử lý logic thuộc phần Model còn lớp Action thuộc phần Controller do vậy, không nên nhúng trực tiếp các mã xử lý logic mà chỉ đặt lời gọi tới xử lý logic trong Model vào phương thức execute của lớp Action.
* Cập nhật lại các đối tượng phía Server sẽ được sử dụng để tạo nên giao diện người dùng cho trang tiếp theo sẽ hiển thị tới người dùng. Thường là các đối tượng Request hay Session. Tùy thuộc vào thời gian mà người lập trình viên muốn giữ các đối tượng đó sẵn sàng.
* Trả về một đối tượng ActionForward thích hợp xác định trang tiếp theo sẽ trình bày dữ liệu đến người dùng, dựa trên các bean được cập nhật mới nhất. Thông thường ta sẽ có được một tham chiếu đến một đối tượng bằng cách gọi phương thức findForward của đối tượng ActionMapping mà đã nhận được trước đó nếu ta đang sử dụng một tên cục bộ trên mapping này hoặc trên chính Controller Servlet của chính nó nếu ta đang sử dụng một tên toàn cục trên toàn ứng dụng.

2.2.2.6.7 ActionMapping

Để có thể hoạt động được thành công, Controller Servlet cần phải biết được làm thế nào mỗi Request của người dùng có thể được ánh xạ đến lớp Action thích hợp. Do vậy Struts đã đóng gói mọi thông tin cần thiết trong một lớp java có tên là ActionMapping các thuộc tính quan trọng nhất của lớp này như sau:

* Type: Tên đầy đủ của lớp java kế thừa từ lớp Action của Framework mà được sử dụng bởi ActionMapping.
* Name: Tên của form bean được định nghĩa trong file config mà Action này sẽ sử dụng.
* Path: đường dẫn Request URI mà được khớp với mapping này.
* Unknown: Thiết lập true nếu Action này nên được cấu hình mặc định cho ứng dụng này, để điều khiển mọi Request không được điều khiển bởi Action khác. Chỉ một Action duy nhất có thể được cấu hình làm Action mặc định cho một ứng dụng.
* Validate: Thiết lập true nếu phương thức validate của Action được liên kết với mapping nên được gọi.
* Forward: đường dẫn Request URI mà điều khiển được chuyển đến khi mapping này đã được gọi

2.2.2.6.8 Cấu trúc ActionMapping

Nhiệm vụ của nhà phát triển là tạo một file cấu hình ứng dụng struts-config.xml và đặt nó vào thư mục WEB-INF của ứng dụng.

Các phần tử của file xml này bao gồm:

* Phần tử <struts-config>: là phần tử lớn nhất bao gồm tất cả các phần tử khác bên trong nó (ba phần tử quan trọng nhất cấu hình nên ứng dụng web với công nghệ Struts).
* Phần tử <form-bean>.
* Phần tử <global-forwards>.
* Phần tử <action-mappings>.

## **2.2.4 Struts 2 FrameWork**

### **2.2.4.1 Sơ lược về Struts 2**

Struts 2 là một Framework được phát triển bởi sự kết hợp của Struts và webwork. Đây là một Framework hỗ trợ thiết kế và xây dựng ứng dụng web theo mô hình MVC thông qua việc cung cấp hàng loạt các thư viện và tiện ích để hỗ trợ người dùng phát triển ứng dụng Web theo mô hình MVC một cách nhanh chóng và dễ dàng.

Struts 2 Framework hỗ trợ việc xây dựng ứng dụng Web theo mô hình MVC nhưng đơn giản hơn mô hình Struts 1 Framework như là rút gọn tập tin cấu hình hay sử dụng annotation thay thế cho tập tin cấu hình.

Struts 2 Framework được sử dụng để phát triển ứng dụng web dựa trên MVC. Struts Framework đầu tiên được tạo bởi Craig McClanahan và được bảo trợ bởi Apache Foundation vào tháng 5/2000 và Struts 1.0 được công bố vào tháng 6/2001. Phiên bản hiện tại là Struts 2.3.16.1 được công bố 2/3/2014.

* Struts 2 : là một open resource được phát triển với Craig McClanahan và được hỗ trợ The Apache Software Foundation
* Struts 2 : là một framework theo mô hình MVC. Dựa trên Struts 1 và là cải thiện của struts 1. Struts 2 việc định nghĩa các thành phần trở nên đơn giản và sáng sủa hơn.

Vào thàng 12/2005, Struts và webwork framework đã được kết hợp với nhau để phát triển nên Struts 2 Framework. Struts 2 rất thích hợp và dễ dàng mở rộng để phát triển các ứng dụng web với các mức độ khác nhau

* Đặc điểm: Cung cấp nhiều đặc điểm mà không có trong Struts 1

Các thành phần MVC có thể cấu hình

Các Action dựa trên POJO

Hỗ trợ AJAX

Dễ dàng tích hợp

Kiểu kết quả đa dạng

Hỗ trợ các tag

Hõ trở them và template

* Các thành phần MVC có thể cầu hình

Struts 2 Framework cung cấp tất cả các thành phần (action và các thành phần view) trong struts.xml file. Nếu bạn cần thay đổi bất cứ thông tin nào, bạn có thể dễ dàng thay đổi nó trong xml file.

* Vào tháng 12/2005,Struts và webwork framework đã được kết hợp với nhau để  
  phát triển nên Struts 2 Framework. Struts 2 Framework rất thích hợp và dễ dàng mở rộng để phát triển các ứng dụng web với mức độ khác nhau
* Struts 2.0x ra đời kế thừa nền tảng của webwork framework, được tổ chức  
  như một nền làm việc chuyên nghiệp với web, nhằm giải quyết các vấn đề phức tạp, mang tính nghiệp vụ cao, khắc phục các nhược điểm của struts 1.  
  Struts 2 Framework đơn giản hóa hơn mô hình Struts 1 Framework và hỗ trợ việc xây dựng ứng dụng web theo mô hình MVC2 với những tiện ích   
  dựng ứng dụng Web theo mô hình MVC2 với những tiện ích hơn phiên bản trước.
* **Một số đặc tính cải tiến của Struts 2**

JavaBeans được sử dụng thay thế ActionForm và có phương thức chỉ định để kích hoạt thực thi (mặc định là phương thức execute không tham số truyền và kiểu trả về là String) nhằm tăng khả năng tái sử dụng của các object trong các ứng dụng và các framework khác. Đặc biệt, chúng dễ dàng thuận tiện cho testing từng thành phần chức năng và thành phần.

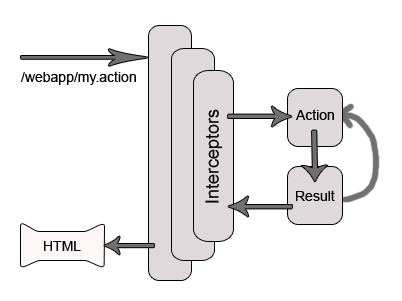
Sử dụng cả annotation và tập tin cấu hình XML rút gọn.

Sử dụng ngôn ngữ mới Object Graphic Notation Language (ONGL) thay thế cho EL của JSP.

Sử dụng bộ taglib dung nhất thay cho 4-5 bộ taglib trong Struts 1 Framework và JSTL 1.1 trong JSP.

### **2.2.4.2 Kiến trúc Struts 2**

Vòng đời của một request trong ứng dụng Struts 2



Người dùng gửi yêu cầu đến server.

Bộ điều khiển lọc FilterDispatcher xem xet yêu cầu và sau đó xác định một số Action thích hợp.

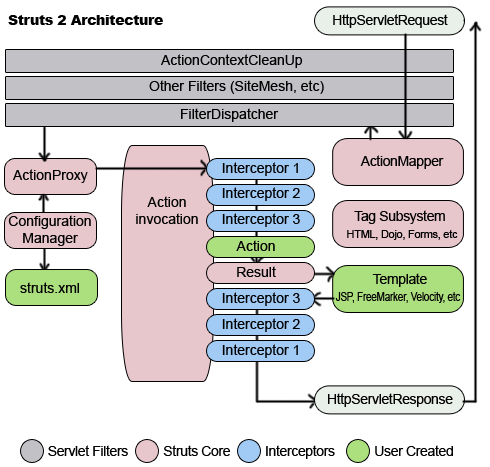
Các bộ chặn Interceptors được ứng dụng: cấu hình bộ chặn áp dụng cho các chức năng chung như luồng công việc, thẩm định, tải file … được tự động ứng dụng vào yêu cầu.

Thực thi Action: sau khi phương thức action được thực thi để thực hiện các hoạt động liên quan đến cơ sở dữ liệu như sắp xếp hoặc nhận dữ liệu từ cơ sở dữ liệu.

Sinh kết quả, sau đó kết quả yêu cầu được trả về thông qua bộ chặn theo chiều ngược lại. Kết quả trả về cho phép chúng ta thực hiện thu dọn hoặc thêm quá trình xử lý.

Hiển thị kết quả cho người sử dụng: việc điều khiển cuối cùng được trả về cho trình chứa servlet, nó sẽ gửi kết quả trả về cho trình duyệt của người sử dụng.

**Kiến trúc Struts 2**



Action ContextCleanUp filter: là một tùy chọn và nó được sử dụng khi tích hợp với các công nghệ khác như SiteMash Plugin

FilterDispatcher: Khi FilterDispatcher được gọi, nó quay lại sử dụng ActionMapper để xác định Action nào được triệu gọi. Nếu Action được yêu cầu được triệu gọi thì FilterDispatcher ủy nhiệm việc điều khiển lại cho ActionProxy.

ActionProxy: Nhận sự giúp đỡ từ bộ quản lý cấu hình file (Configuration Files Manager), được khởi tạo từ file struts.xml. Sau đó ActionProxy atoj ra một ActionInvocation, thực thi từ mẫu lệnh. ActionInvocation xử lý triệu gọi Interceptors nếu nó được cấu hình và sau đó triệu gọi Action. ActionInvocation tìm kiếm kết quả thích hợp, sau đó kết quả được thực thi, nó triệu gọi việc sinh ra các trang JSP hoặc các mẫu.

**2.2.3.3 Các thành phần chính yếu và thường dùng của Struts 2 Framework FilterDispatcher**

* FilterDispatcher được mapping string theo web.xml như sau:

<filter>

<filter-name>struts2</filter-name>

<filter-class>**org.apache.struts2.dispatcher.FilterDispatcher**</servlet-class>

</filter>

<filter-mapping>

<filter-name>struts2</servlet-name>

<url-pattern**>/\***</url-pattern>

</filter-mapping>

* **Action Class hay ActionSupport Class**

Chứa đựng các thuộc tính mapping trên form và các thành phần xử lý của Object (business logic) và phương thức chỉ định thực thi.

Struts 2 Framework cung cấp cơ chế tự động mapping các tham số trong request trong form đến các thuộc tính trong Action với các thuộc tính cùng tên và có phương thức setXxx.

Trong quá trình mapping, Struts 2 Framework sử dụng OGNL thay thế EL để ép kiểu tự động từ form đến kiểu dữ liệu của Action property. Do vậy, không nhất thiết lúc nào cũng khai báo kiểu dữ liệu là String và thực hiện ép kiểu bằng tay.

* **Struts config (Struts.xml)**

<struts>

[<include file=”xml File”/>]

<package name=”default” namespace=”/” extends=”struts-default” [method=”method\_name”]>

<action name=”action\_Name” class=”package.className”>

<result name=”stringValue”>/pageView</result>

…

</action>

</package>

</struts>

Trong đó:

- package: nhóm thông tin cấu hình của những thành phần có chung thuộc tính

. name: xác định tên của của package.

. namespace: phân biệt giữa các package khác tránh xung đột giữa các action với nhau.

. extends: xác định class cha tương tự như khái niệm kế thừa.

. abstract: bằng true nếu là lớp kế thừa.

- action: ánh xạ một action với một class action

. name: xác định tên action

. class: xác định java class để xử lý action

. method: xác định phương thức được gọi khi request được chuyển tới. Nếu tham số này không tồn tại thì phương thức execute mặc định được tìm kiếm và thực thi.

- result: xác định tên result hay page kết xuất của một action

. name: xác định tên result. Nếu không có thì mặc định là success.

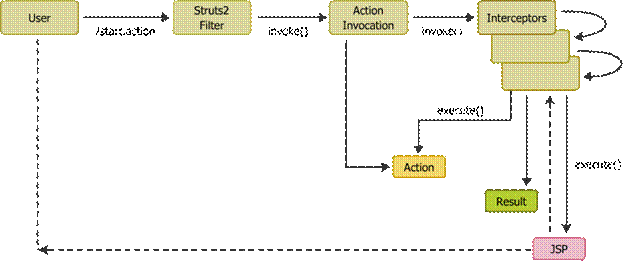
. type: xác định loại result. Nếu không có thì mặc định là FilterDispatcher thực hiện forward đến trang kết quả.

### **2.2.4.3 Mô hình làm việc và cơ chế hoạt động của Struts 2 Framework**

**2.2.3.3.1 Cơ chế hoạt động Struts 2 Framework**

Struts 2 framework hoạt động tương tự như mô hình MVC của Struts 1 Framework nhưng điểm khác biệt là pull-framework nghĩa là dữ liệu được lấy trực tiếp từ action đưa lên view

* Controller: FilterDispatcher là servlet Filter có nhiệm vụ đón nhận request và dựa trên cấu hình để xác định action cụ thể để đón nhận request cho xử lý.
* Model: Action là một java class đảm bảo đặc tính của một object trong mô hình hướng đối tượng và thỏa tính chất của JavaBean.
* View: Result là một kết xuất hay một dạng xác định action hay trang chuyển về và trình bày trên Web Browser.



Khi người dùng gửi yêu cầu truy xuất ứng dụng web, request được chuyển đến FilterDispatcher, đây là Filter được tạo sẵn trong Framework để làm chức năng như Controller.

FilterDispatcher dựa trên trập tin cấu hình xác định xem action có được ứng dụng Interceptor hay không.

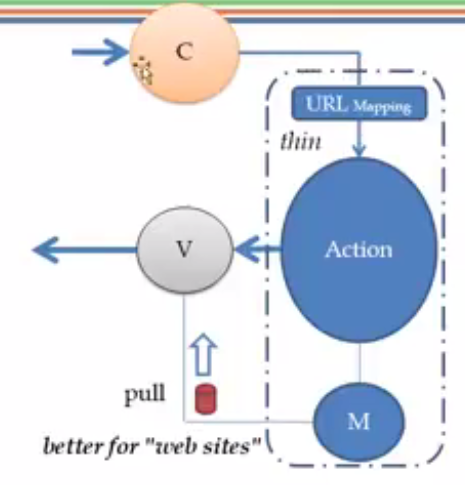
Nếu có, framework sẽ gọi Interceptor theo tuần tự. khi gọi xong một Interceptor, framework sẽ checking tiếp Interceptor trong stack cho đến Interceptor cuối cùng.

Nếu không có áp dụng Interceptor cuối cùng được thực thi thì framework sẽ gọi action.

Action đón nhận dữ liệu, thực hiện xử lý và kích hoạt phương thức chỉ định thực thi. Sau khi phương thức đó xử lý hoàn tất, Result được trả về. Tại bước này, nếu ứng dụng có áp dụng interceptor là Result sẽ được apply theo chiều ngược chiều gọi ban đầu để trả kết quả trở về hay chuyển đến Action tiếp theo.

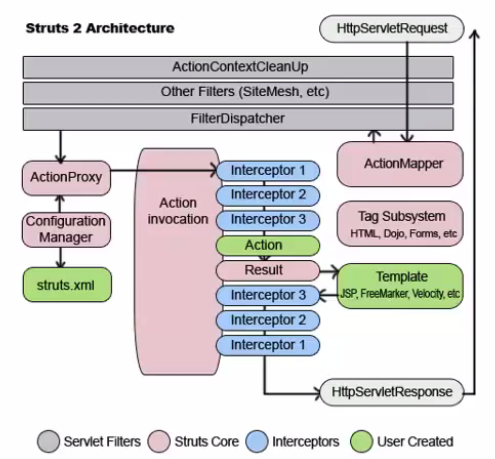
Điểm khác biệt giữa cơ chế hoạt động của Struts 2 so với Struts 1 là tất cả mọi thứ phải thông qua file struts.xml hay annotation và một phần nào đó Struts 2 che dấu FilterDispatcher đối với người lập trình để tạo nên tính linh hoạt trong việc xây dựng và bảo trì ứng dụng web, đặc biệt là việc mapping kết xuất giữa các trang không còn lệ thuộc và đặt trong code servlet dẫn đến việc mapping trở nên linh hoạt và uyển chuyển hơn.

**2.2.3.3.2 Mô hình làm việc Struts 2 Framework**

****

* Trong struts 2 là chỉ bao gồm 1 controller và controller này có thể chứa các controller khác.
* Sau khi nhận request của người dùng, controller sẽ xác định xem sẽ gọi đến Action nào.Sau đó dữ liệu truyền vào model.
* Sau đó Action sẽ chuyển hướng đến View và View sẽ lấy dữ liệu từ model và trả về response cho người dùng.

Mô hình chi tiết hơn

****

- Khi người dùng có 1 request lên server. Thì request sẽ chuyển đến ActionMapper và sau đó nó sẽ đi qua các filter.  
           - Sau đó nó sẽ đọc file struts.xml, tùy thuộc vào config trong file struts.xml. ActionProxy sẽ chuyển đến Action tương ứng. Trước khi gọi Action thì nó sẽ các gọi các interceptor nếu có. Sau đó nó gọi đến Action thật sự và nó trả về resuilt.  
          - Sau khi có Resuilt nó sẽ định hướng đến View, sau đó nó tiếp tục đi qua các interceptor và trả về Response cho người dùng.

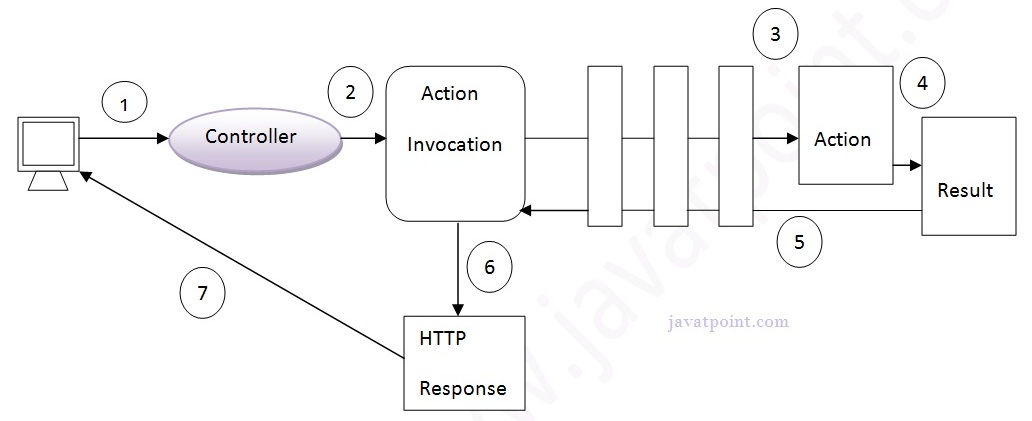
### **2.2.4.4 Cấu trúc Struts 2**

Struts 2 là một MVC 2 Framework, được tổ hợp bởi rất nhiều thành phần. Mẫu MVC trong Struts 2 bao gồm 5 thành phần cốt lõi, đó là:

* **Action**
* **Interceptor**
* **Value Stack / OGNL**
* **Result type**
* **Các công nghệ View**

Từ các thành phần cốt lõi trên, phần dưới sẽ trình bày cho các bạn về Struts Flow theo hai cách, đó là Basic Flow và Standard Flow. Qua đó giúp bạn có cái nhìn khái quát về vòng đời của Struts.

**Struts 2 Basic Flow**

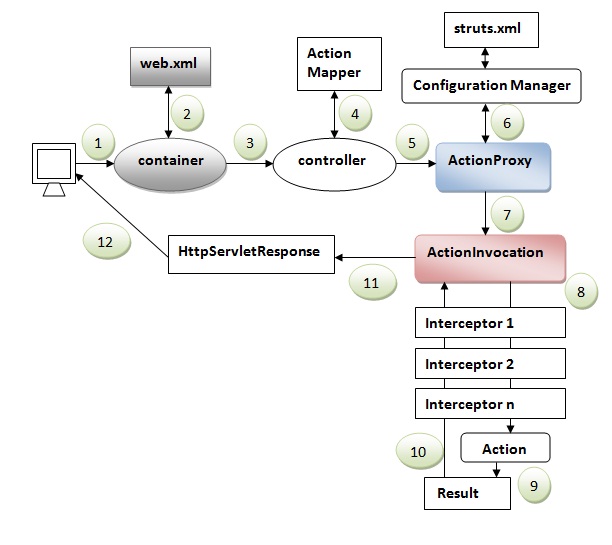


Các bước trên có thể diễn giải như sau:

* + Người dùng gửi một yêu cầu (request) cho Action.
  + Controller triệu hồi ActionInvocation
  + ActionInvocation triệu hồi mỗi Interceptor và Action
  + Một kết quả được tạo ra
  + Kết quả được gửi lại ActionInvocation
  + Một HttpServletResponse được tạo ra
  + Phản hồi (response) được gửi tới người dùng

**Struts 2 Standard Flow (Cấu trúc Struts 2)**

Bạn theo dõi sơ đồ sau:



* 1. Người dùng gửi một yêu cầu (request) cho Action
  2. Container ánh xạ request này trong web.xml file và lấy tên lớp của Controller
  3. Container triệu hồi Controller (StrutsPrepareAndExecuteFilter hoặc FilterDispatcher). Bắt đầu từ Struts 2.1, đó là lớp StrutsPrepareAndExecuteFilter
  4. Controller lấy thông tin cho Action từ ActionMapper
  5. Controller triệu hồi ActionProxy
  6. ActionProxy lấy thông tin của Action và Interceptor Stack từ Configuration Manager mà lấy thông tin từ struts.xml file
  7. ActionProxy chuyển tiếp request tới ActionInvocation
  8. ActionInvocation triệu hồi mỗi Interceptor và Action
  9. Một kết quả được tạo ra
  10. Một kết quả được tạo ra
  11. Một HttpServletResponse được tạo ra
  12. Phản hồi (response) được gửi tới người dùng

**2.3 So sánh khác nhau của Struts 1 Framework và Struts 2 Framework**

|  |  |
| --- | --- |
| Struts 1 | Struts 2 |
| Sử dụng ActionServlet làm Controller | Sử dụng FilterDispatcher làm Controller |
| Dùng HTML form kết hợp với ActionForm object để đón nhận giá trị nhập và xử lý validation nếu cần | Dùng các thuộc tính trong Action class để đón nhận giá trị nhập từ form và thực hiện xử lý cùng với validation nếu cần |
| Action bắt buộc implement Action interface | Action class không bắt buộc implement Action interface |
| Duy nhất một instrance của Action đón nhận để xử lý mọi request. Cơ chế để đảm bảo xử lý đồng bộ đòi hỏi phức tạp hơn | Một request sẽ có một instance đón nhận xử lý |
| Cung cấp nhiều taglib gây khó khăn cho người dùng khi tiếp cận và sử dụng | Cung cấp duy nhất một taglib duy nhất đảm bảo đầy đủ các thành phần hỗ trợ xử lý từ đơn giản đến nâng cao, kể cả JSTL |
| Sử dụng EL và JSTL | Sử dụng ONGL để xử lý |
| Dùng cơ chế biên dịch JSP để kết nối các thành phần trong xử lý | Sử dụng ValueStack để cho phép taglib truy cập giá trị trong quá trình xử lý |
| Chia thành module để chuyển đổi thao tác và đòi hỏi kết hợp của switchAction để tạo sự kết hợp giữa các thành phần khi làm việc theo nhóm | Sử dụng cơ chế interceptor để đảm bảo tích hợp nhiều thành phần và nâng cấp ứng dụng một cách uyển chuyển linh hoạt, đặc biệt là tích hợp mà không làm ảnh hưởng các thành phần có sẵn khi làm việc theo nhóm. |

# **Chương 3: Ứng dụng Struts 1 Framework vào xây dựng hệ thống quán lý cửa hàng bán vật tư y tế.**

## Giới thiệu về ứng dụng demo sẽ xây dựng

### Tính cấp thiết của đề tài

Trường Đại học Khoa học hiện nay đang đào tạo đại học hệ chính quy và hệ không chính quy. Việc tuyển sinh đại học hệ chính quy đang sử dụng phần mềm chung của Bộ Giáo dục và Đào tạo áp dụng cho một số lượng lớn thí sinh tham dự và đòi hỏi một cơ sở hạ tầng khá phức tạp. Với đặc điểm tuyển sinh đại học hệ không chính quy là số lượng thí sinh tham dự ít, hình thức tổ chức đơn giản nên việc áp dụng phần mềm do Bộ Giáo dục và Đào tạo cung cấp không phù hợp với thực tế của nhà trường. Phòng Đào tạo Đại học, Trường Đại học Khoa học đã xây dựng một chương trình nhỏ bằng Excel để phục vụ cho công tác tuyển sinh đại học hệ không chính quy. Chương trình này còn nhiều công đoạn thực hiện thủ công nên đòi hỏi thời gian thao tác nhiều, người sử dụng phải am hiểu sâu về Excel. Do đó, phần mềm này không phù hợp với thực trạng hiện nay của Phòng Đào tạo Đại học. Vì vậy, việc xây dựng một chương trình quản lý tuyển sinh đại học hệ không chính quy là một nhu cầu bức thiết của Phòng và góp phần vào việc phát triển công tác đào tạo đại học hệ không chính quy.

### 3.1.2 Mục đích

Việc xây dựng chương trình quản lý tuyển sinh đại học hệ không chính quy với mục tiêu:

* Quản lý thông tin thí sinh dự thi như: họ tên, địa chỉ, số báo danh, số phòng, điểm thi,…; quản lý nghiệp vụ thi, chấm thi bao gồm: phân bố phòng thi, đánh số báo danh, số phách, điểm thi, xác định điểm chuẩn.
* Giúp chuyên viên Phòng Đào tạo Đại học trong việc nâng cao hiệu suất làm việc trong công tác tuyển sinh đại học hệ không chính quy của nhà trường, hạn chế sai sót, tăng cường tính chuyên nghiệp.

### Phạm vi

Hệ thống hỗ trợ trong công tác tuyển sinh đại học hệ không chính quy của nhà trường, áp dụng trong khuôn viên trường Đại Học Khoa Học – Huế.

# 3.2 Mô tả bài toán

Chương trình quản lý tuyển sinh đại học hệ không chính quy được tổ chức như sau:

a/ Tổ chức thi:

+ Quản lý thông tin kỳ thi

+ Quản lý thí sinh dự thi

+ Đánh số báo danh

+ Quản lý phòng thi

b/ Tổ chức chấm thi:

+ Xử lý bài thi

+ Dồn túi phách

+ Nhập điểm thi

+ Xác định điểm chuẩn

c/ Thống kê – báo cáo:

+ Thống kê bảng điểm môn thi

+ Thống kê bảng điểm tổng hợp

+ Thống kê danh sách thí sinh trúng tuyển

+ Thống kê điểm thi tuyển

+ Thống kê điểm trúng tuyển

+ Thống kê điểm thi tuyển theo ưu tiên

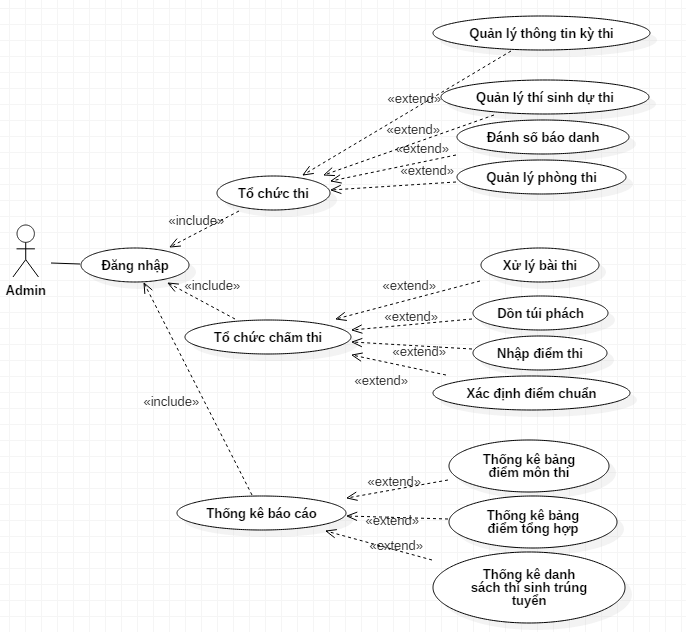
...

# 3.3 Phân tích thiết kế hệ thống

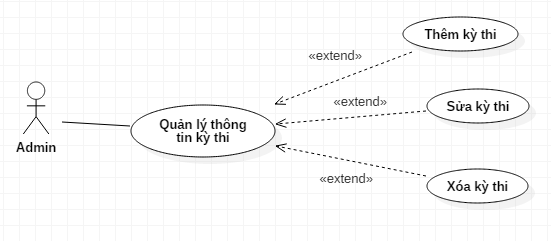
## 3.3.1 Tác nhân

|  |  |
| --- | --- |
| **Tác nhân** | **Mô tả** |
| Admin | Quản lý thông tin kỳ thi, thí sinh, phòng thi; đánh số báo danh, phân phòng thi; xử lý bài thi, dồn túi phách, nhập điểm thi, xác định điểm chuẩn; thực hiện thống kê – báo cáo. |

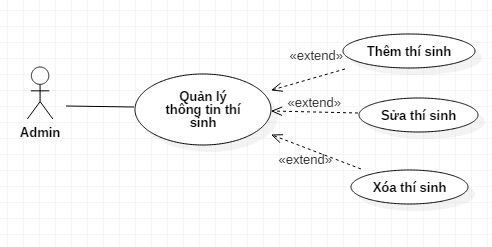
## 3.3.2 Biểu đồ use case



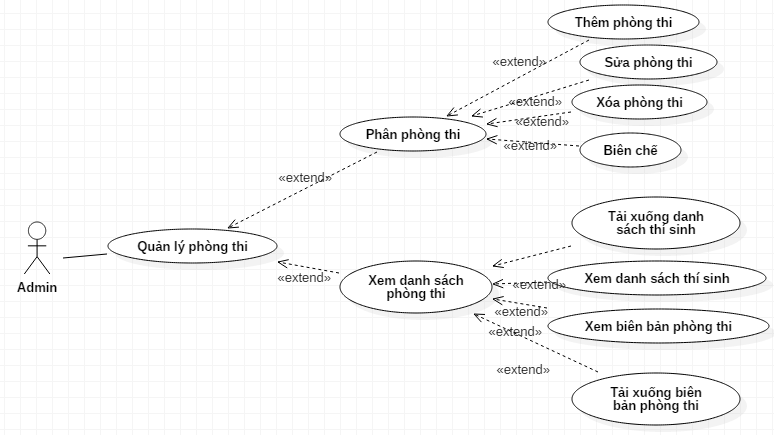
## 3.3.3 Chức năng Quản lý thông tin kỳ thi



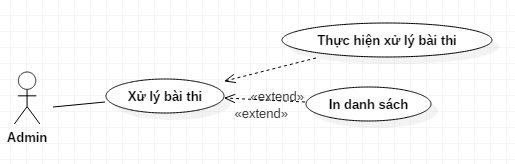
## 3.3.4 Chức năng Quản lý thông tin thí thí sinh



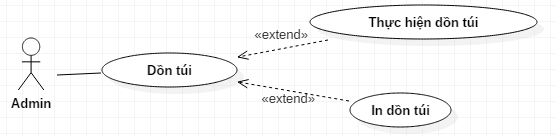
## 3.3.5 Chức năng Quản lý phòng thi



## 3.3.6 Chức năng Xử lý bài thi



## 3.3.7 Chức năng Dồn túi phách



## 3.3.8 Chức năng Nhập điểm thi

