Capítulo 2. Marco Teórico

2.1. Frameworks para Aplicaciones Web en Java

Con el crecimiento exponencial de Internet en los últimos años, las aplicaciones Web se han convertido en una parte básica y común dentro del desarrollo de software; éstas han acaparado la atención no sólo de las empresas que desean formar parte de este nuevo mundo, sino también de aquellas que se han dedicado a las herramientas de desarrollo de software.

Dentro del desarrollo de este tipo de aplicaciones, Java juega un papel muy importante actualmente, ya que éste es uno de los usos más comunes que se le da a este lenguaje de programación, además de que representa uno de los mejores medios para construir dichas aplicaciones.

2.1.1. ¿Qué es un framework?

El término framework, o marco de trabajo, se ha popularizado en los últimos años dentro del ambiente de desarrollo de software. Es común encontrar dicho término en diversas circunstancias: leyendo un libro sobre algún lenguaje de programación, buscando información de interés en Internet sobre una nueva tecnología Web, etc.

Un framework, dentro del ambiente de desarrollo de software, es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, librerías y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un framework se considera como una arquitectura de software que modela las relaciones generales de los componentes del proyecto que lo implementa; provee una estructura y manera de trabajo la cual utilizan las aplicaciones del proyecto. La finalidad de los frameworks es facilitar el desarrollo de software, permitiéndoles a diseñadores y programadores concentrarse en los requerimientos del proyecto, reduciendo los posibles problemas con las tecnologías utilizadas, así como facilitando ciertas funcionalidades básicas y comunes.

Otro término también bastante popular hoy en día es el de patrones de diseño, que muchas veces se tiende a emplear en el mismo sentido que framework, pensando que hacen referencia a un mismo concepto; sin embargo, debe quedar claro que tienen significados diferentes, ya que un framework representa código implementado, mientras que un patrón de diseño representa conocimiento y experiencia.

2.1.2. Orígenes de los frameworks para aplicaciones Web en Java

El surgimiento de los servlets de Java resultó en un avance bastante productivo en relación con el estándar CGI, ya que resultaban más poderosos y rápidos, así como portables y fácilmente extensibles; sin embargo, el desplegar código HTML en el explorador a través del método *println()*, especialmente cuando se trataba de gran cantidad de líneas, resultaba problemático y agotador.

Ante esta situación, llegaron al mundo los JSP (*JavaServer Pages*), invirtiendo el concepto de los servlets, permitiendo insertar fácilmente código Java dentro de la página HTML. Con esta solución, las aplicaciones Web adoptaron a los JSP como figura central, lo que pronto traería como consecuencia problemas en el control del flujo, así como en el mantenimiento de páginas con demasiado código Java.

Ante esta nueva situación, se llegó a la idea de utilizar ambas tecnologías de manera conjunta, lo que representó una buena opción y satisfizo las necesidades de los desarrolladores, aunque sólo por un tiempo, ya que con la experimentación e implementación de dichas tecnologías, que además se presentaban como estándares, la comunidad que las había utilizado llegó a la conclusión de que lo que se les ofrecía, o bien no era suficiente para cumplir con los requerimientos de los diferentes proyectos, o lo realizaba de una manera no óptima o por debajo del nivel requerido.

Esto condujo a que los programadores desarrollaran sus propios medios para cumplir con sus necesidades, lo que con el tiempo traería como resultado la aparición de los primeros frameworks orientados a las aplicaciones Web.

2.1.3. Patrón de diseño "Modelo 2"

Cuando los programadores utilizaron los servlets junto con los JSP, los primeros para realizar el control de flujo y los segundos para presentar código HTML, surgió el Modelo 2, que a fin de cuentas consiste en una adaptación del famoso patrón de diseño Modelo-Vista-Controlador (MVC, *Model-View-Controller*).

El patrón de diseño MVC tiene por objetivo realizar la separación entre la lógica de la aplicación (Modelo) y la interfaz de usuario (Vista), como se muestra en la Figura 2.1.1.

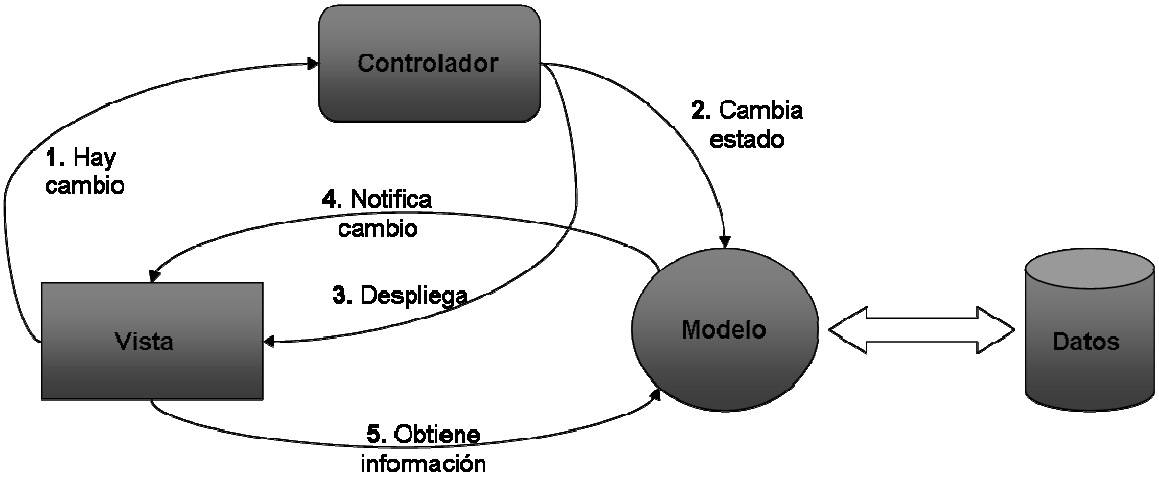

Figura 2.1.1 Arquitectura del Patrón de Diseño MVC

Como se puede observar en la Figura 2.1.1, este patrón tiene tres componentes fundamentales:

1. **Modelo:** es la representación de los datos de la aplicación y del código. Coordina la lógica de la aplicación, acceso a bases de datos y otras partes no visuales del sistema.
2. **Vista:** es la presentación visual de los datos. Se encarga de la interacción con el usuario.
3. **Controlador:** es el intermediario entre las otras dos partes. Es responsable de desplegar la vista adecuada al usuario.

Como se mencionó anteriormente, el patrón de diseño Modelo 2 sigue la estructura del MVC, sólo que orientado a aplicaciones Web, y por tal motivo los componentes son ahora representados por tecnologías específicas, como se muestra en la Figura 2.1.2.

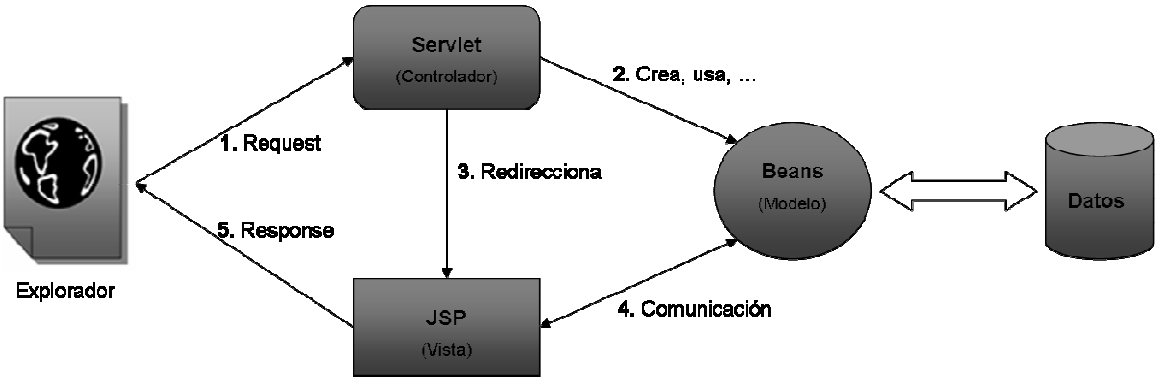

Figura 2.1.2 Arquitectura del Patrón de Diseño Modelo 2

Debido a la clara separación de componentes y funciones que se logra con el Modelo 2, éste ha jugado un papel muy importante en el desarrollo de los frameworks orientados a aplicaciones Web en Java, ya que la mayoría de estos aprovechan sus beneficios y lo toman como paradigma.

2.1.4. Tipos de frameworks para aplicaciones Web en Java

Aunque no existe una clasificación estándar y formal que se aplique a los frameworks para aplicaciones Web, existen algunos aspectos que se pueden considerar para diferenciar unos de otros.

De esta manera, teniendo en cuenta el enfoque de los frameworks, se pueden mencionar dos tipos:

- frameworks de aplicación y
- frameworks de interfaz de usuario.

Los frameworks de aplicación son aquellos que están basados en los *request* HTTP, es decir, utilizan el API Servlet. Estos se enfocan en soportar el desarrollo de aplicaciones completas, en la imagen completa. Así mismo, estos frameworks no se preocupan acerca de los detalles de cómo se renderiza la interfaz de usuario y no hacen distinción entre acciones

que sólo afectan a la interfaz de usuario y aquellas que requieren ser procesadas por código de la aplicación [Bergsten, 2004].

Por la otra parte se encuentran los frameworks de interfaz de usuario, basados en componentes. Estos se enfocan en los detalles de la interfaz de usuario y no se interesan en cómo es implementado el resto de la aplicación. De igual forma, definen un API detallado para los componentes que formarán la interfaz de usuario con los objetivos de ligar a estos con la lógica de la aplicación adecuada, determinar qué acciones de los usuarios resultarán eventos en la interfaz y cómo serán manejados estos últimos, entre otras finalidades [Bergsten, 2004].

2.1.5. Algunos frameworks para aplicaciones Web en Java

Al momento de finalizar esta Tesis, existen alrededor de cincuenta frameworks para aplicaciones Web en Java. Lo que comenzó como algo bueno para la comunidad de desarrolladores, debido a la posibilidad de elegir aquel framework que se ajustara más a las necesidades del proyecto, se ha vuelto al final un problema, poniendo al programador a realizar un análisis bastante agobiante para encontrar su mejor opción.

De entre todos estos frameworks han sobresalido algunos por una u otra razón. A continuación se describen brevemente algunos de estos.

- **Nombre:** Struts.
  Open Source: Sí.
  Tipo: Framework de aplicación.
  Aspecto que lo destaca: Primer framework en salir; es el más usado; madurez.

- **Nombre:** Spring.
  Open Source: Sí.
  Tipo: Framework de aplicación.
  Aspecto que lo destaca: Muy popular últimamente.

- **Nombre:** JavaServer Faces (JSF).
  Open Source: No.
  Tipo: Framework de interfaz de usuario.
  Aspecto que lo destaca: Framework estándar; soporte para IDEs (*Integrated Development Environment*).

- **Nombre:** Tapestry.
  Open Source: Sí.
  Tipo: Framework de interfaz de usuario.
  Aspecto que lo destaca: Primer framework de su tipo.

2.2. Competidores de JavaServer Faces

A pesar de que todos los frameworks se consideran competidores entre sí, el más directo de JavaServer Faces es Tapestry, debido a que es del mismo tipo, y por lo tanto, busca realizar casi las mismas tareas; en cuanto a los frameworks de aplicación, sucede algo diferente, ya que pueden utilizarse en conjunto con JSF o con cualquiera orientado a la interfaz de usuario que sea lo suficientemente flexible, cada uno desempeñando su propio papel.

De acuerdo al sitio oficial de Tapestry [Tapestry, 2006], éste se define como un framework *open-source* para crear aplicaciones Web en Java que sean dinámicas, robustas y altamente escalables y cuya filosofía está basada en el uso de componentes para el desarrollo Web; así mismo, está específicamente diseñado para crear nuevos componentes fácilmente.

Otro competidor que enfrenta no sólo JavaServer Faces, sino todos los frameworks para aplicaciones Web, es Ajax (*Asynchronous JavaScript And XML*), que ha venido a proponer una alternativa al modelo clásico que utilizan estas aplicaciones.

Ajax no es un framework, ni tampoco una tecnología, consiste en realidad de varias tecnologías, cada una prosperando por su parte, uniéndose para convertirse en un nuevo y poderoso medio [James, 2005].

Ajax critica el modelo Web clásico, argumentando que tiene mucho sentido para el aspecto técnico, pero que sin embargo no hace nada para mejorar la experiencia del usuario [James, 2005], pues mientras se manda la petición al servidor y se regresa la respuesta, el usuario sólo espera a que se le presente la información. Ante tal situación, el modelo que propone, consiste en un "motor Ajax" entre el usuario y el servidor, a manera de intermediario que se encargará de desplegar la interfaz y de comunicarse con este último de manera asíncrona; así, las acciones del usuario que no requieran información del servidor, serán procesadas por el motor, mientras que en aquellas que sí lo requieran, el motor intervendrá para hacer la petición a aquél (de manera asíncrona), generalmente a través de XML, para no interrumpir la interacción del usuario con la aplicación [James, 2005].

Lo que propone Ajax es básicamente reducir el envío de *request*/*response*, es decir recurrir al servidor lo menos posible, y tratar de realizar la mayor parte de operaciones en la parte del cliente para acelerar y mejorar la interacción usuario-aplicación.

2.3. ¿Por qué JavaServer Faces?

A pesar de que Ajax ha surgido con gran fuerza y ha ganado bastante popularidad en su corta vida, su tesis central es proporcionar una mejor interacción con el usuario mejorando los tiempos de respuesta de la aplicación Web, dejando a terceros la construcción de la interfaz gráfica. Por este motivo, no es de extrañar que JavaServer Faces pueda utilizarse

junto con Ajax, como sucede en la versión 1.2 de JSF, para aportar los beneficios de ambos y obtener mejores resultados con menos esfuerzo.

Por otra parte, como se mencionó anteriormente, Tapestry fue el primer framework orientado a la interfaz de usuario, y por este motivo JSF es muy parecido a él, teniendo cada uno de estos sus respectivos defensores y críticos. Sin embargo, a pesar de que realizan muchas cosas similares, e independientemente de la facilidad con que cada uno de ellos logra éstas, JavaServer Faces tiene algunas características que lo vuelven más atractivo y que influyeron de manera importante para elegir a éste como el framework con el cual trabajar; por ejemplo:

- Es una tecnología estándar.
- Permite a los creadores de herramientas desarrollar IDEs para un framework estándar.
- Promete realizar un desarrollo rápido de aplicaciones (RAD, *Rapid Application Development*).
- Su constante evolución (tiene tres versiones en dos años).
- Flexibilidad para adaptarse o conectarse con otras tecnologías (por ejemplo *Spring* o *Hibernate*).