DISEÑO Y PROGRAMACIÓN ORIENTADA A OBJETOS

Taller 5

Andres Mateo Chilito Avella

codigo:202214992
a.chilitoa@uniandes.edu.co

Universidad de los Andes
Dr. Héctor Arturo Flórez Fernández

Bogotá - Colombía

17 de mayo de 2023

1 Proyecto TomCat, patrón Structural Front Controller

1.1 Información proyecto

1.1.1. Descripción proyecto

Teniendo conocimiento del modelo OSI, específicamente de la capa de aplicación (En el otro documento del repositorio se puede documentar y explorar un análisis de caso de la capa de aplicación realizado por mi) Tomcat es un servidor web el cual fue desarrollado por Apache Software Foundation. Actúa como un intermediario de la capa siete recibiendo las solicitudes HTTP de los clientes y las aplicaciones para su correcto procesamiento.
Tomcat se especializa en el soporte de aplicaciones web desarrolladas en Java, como Servlets y JavaServer Pages (JSP). Puede ejecutar estas aplicaciones utilizando el contenedor de servlets incorporado. Conjuntamente, como es de costumbre en java, se ofrece gran seguridad y la garantía de manejar múltiples solicitudes, atendiéndolas de la manera más eficiente posible. Tomcat permite realizar múltiples configuraciones y personalizaciones, por lo que se convierte en un gran aliado en el despliegue de las apps.[01]

1.1.2. Objetivos del proyecto

Tomcat al ser una app de gran trascendencia para la web, tiene que estar muy bien estructurada, por lo que algunos de sus objetivos son:
1. Ser confiable y preciso como contenedor de servlets y JSP: Tomcat está diseñado para ser un contenedor robusto y eficiente que puede ejecutar aplicaciones web basadas en servlets y JavaServer Pages (JSP).
2. Ofrecer un entorno seguro y escalable: Implementa medidas de seguridad y permite la configuración de autenticación y autorización. Conjuntamente, Tomcat está diseñado para ser escalable y capaz de manejar una carga considerable de solicitudes web simultáneas.
3. Proporcionar herramientas de administración y monitoreo: Tomcat posee herramientas de administración como de monitoreo que sirven para la gestión de las aplicaciones. Lo anterior incluye: la capacidad de administrar los recursos del servidor, controlar el rendimiento y realizar ajustes según sea necesario. [01]

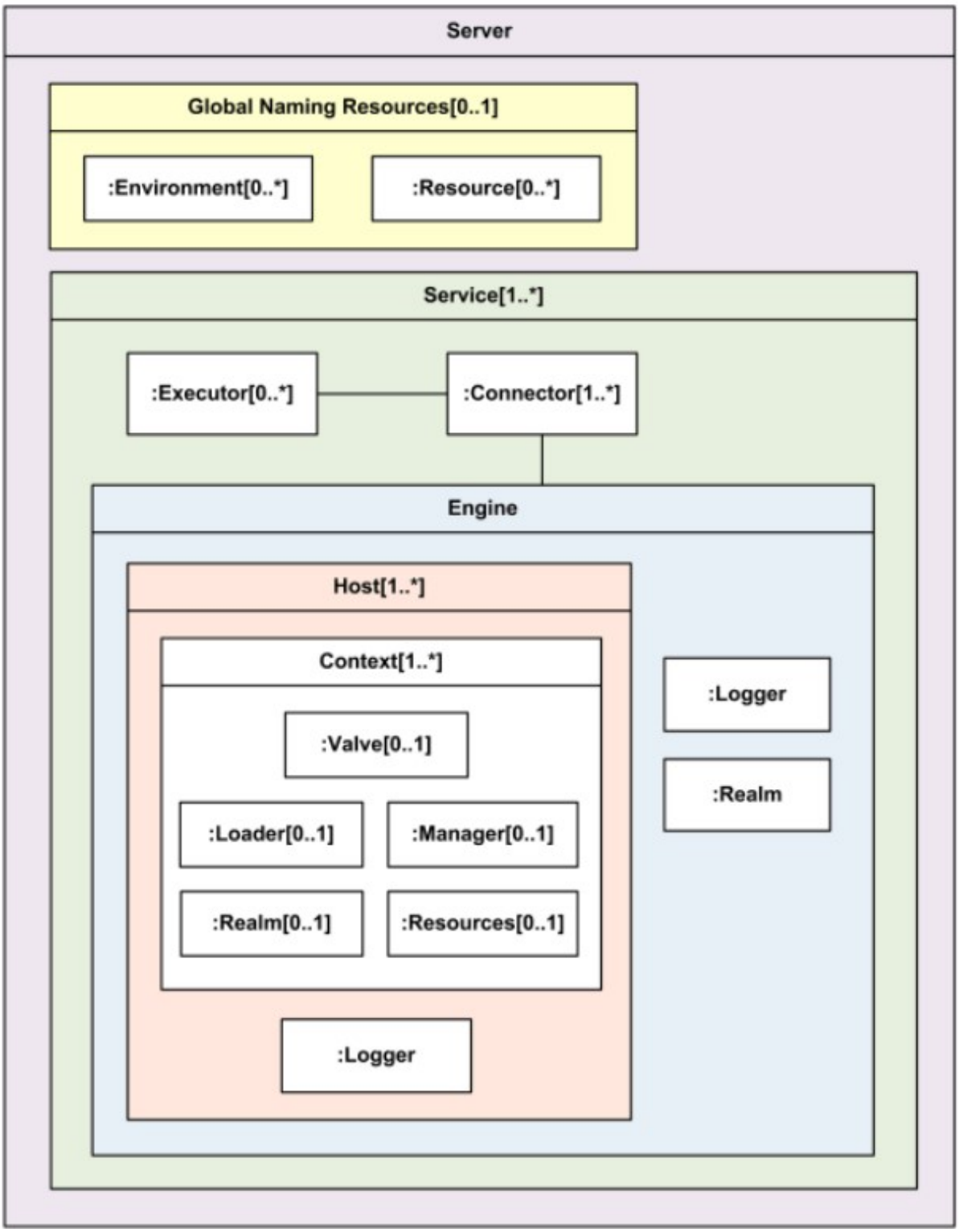
1.1.3. Retos del proyecto

Los retos que ha enfrentado Tomcat a lo largo de los años son:
- Mantener la compatibilidad: Tomcat para perdurar en los años ha tenido que mantener compatibilidad con las especificaciones de Java Servlet y JSP, así como con las versiones anteriores del servidor. Lo anteriormente dicho implica garantizar que las nuevas versiones de Tomcat puedan ejecutar las aplicaciones desarrolladas para versiones anteriores sin problemas y sin requerir cambios significativos.
- Gestión de recursos y escalabilidad: A medida que se crece en cuanto a servicios web y se evoluciona en las versiones de las apps, es necesario proporcionar mecanismos eficientes para administrar recursos, como conexiones de base de datos y conexiones de red. [01]

1.1.4. Estructura general

La arquitectura del servidor Tomcat consta de varios componentes, incluyendo un Servidor, un Servicio, un Motor, un Host, un Conector y un Contexto. Un Servidor representa todo el contenedor, y Tomcat proporciona una implementación por defecto de la interfaz de Servidor. Un Servicio es un componente intermedio que vive dentro de un Servidor y vincula uno o más Conectores a exactamente un Motor. Un Motor representa una cadena de procesamiento de peticiones para un Servicio específico. Un Host es una asociación de un nombre de red al servidor Tomcat. Un Conector maneja las comunicaciones con el cliente. Un Contexto representa una aplicación web.[01]

Esta es la estructura de Tomcat:



1.1.5. Enlace al repositorio

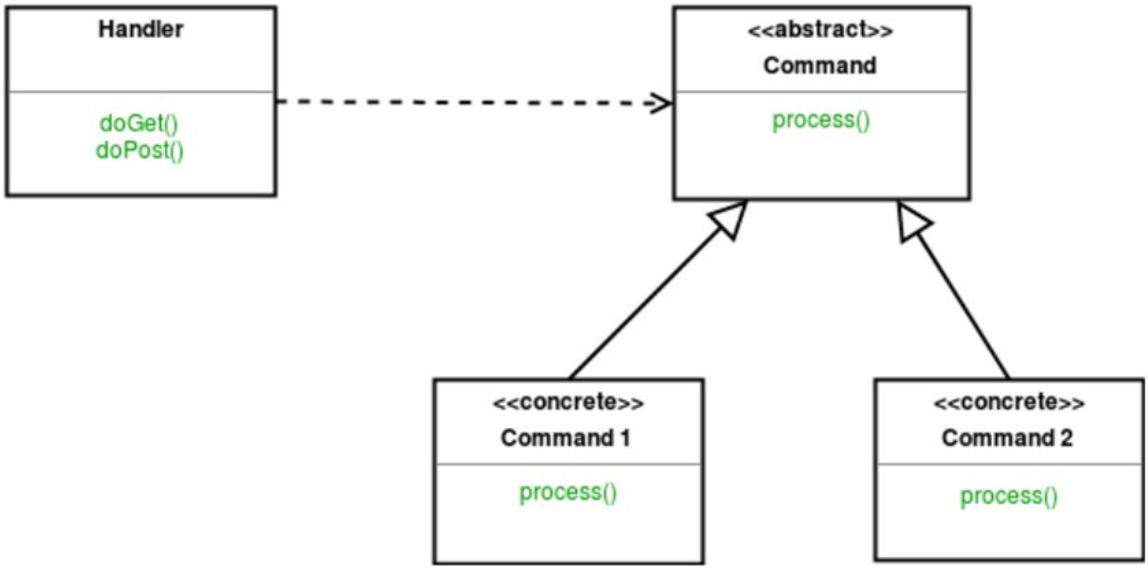
Click Acá

1.2 Información sobre el patrón implementado

1.2.1. Descripción del patrón y partes

El patrón de diseño Front Controller se utiliza para proporcionar un mecanismo centralizado de gestión de solicitudes, de modo que todas las solicitudes de un recurso en una aplicación sean gestionadas por un único gestor.[02] Este gestor puede realizar la autenticación, autorización, registro o seguimiento de las solicitudes y, a continuación, pasar las solicitudes a los gestores adecuados Este manejador puede hacer la autenticación, autorización, registro o seguimiento de las solicitudes y luego pasar las solicitudes a los manejadores apropiados.[03]
El patrón Front Controller forma parte de los Enterprise Patterns tal y como se definen en el libro de Martin Fowler "Patterns of Enterprise Application Architecture". Se sitúa delante de una aplicación web y delega las peticiones a los recursos subsiguientes. También proporciona una interfaz para comportamientos comunes como la seguridad, la internacionalización y la presentación de vistas particulares a determinados usuarios.
El patrón Front Controller se divide principalmente en dos partes: un único controlador despachador y una jerarquía de comandos. El controlador único envía peticiones a los comandos para desencadenar el comportamiento asociado a una petición. [04]

1.2.2. Estructura del patrón



1.3 El patrón en el proyecto

1.3.1. Implementación del patrón en el proyecto

El patrón de Front controller se aplica en Tomcat definiendo un controlador principal mediante la configuración del archivo web.xml. Aquí se especifica el servlet que actuará como Front Controller y se asigna una URL de patrón para interceptar todas las solicitudes que lleguen. Luego sigue un enrutamiento de solicitudes donde el controlador principal recibe todas las solicitudes determina cómo se procesa cada una de ellas.
Posterior a ello se definen unos controladores específicos para cada acción o función en la aplicación. Los controladores específicos se configuran como servlets individuales en el archivo web.xml y se asocian a

diferentes URL específicas. Cabe resaltar que cada controlador se encarga de una parte especifica de la lógica. Cuando ya todas las responsabilidades son asignadas a cada controlador especifico, es decir, se sabe que controlador va a manejar cada solicitud, se dirige la solicitud al controlador correspondiente y este la contesta, casi como un TCP (ver el documento que al inicio adjunte para documentarse sobre el Transmission Control Protocol). Para finalizar la respuesta del controlador especifico se representa por medio de páginas JSP.

1.3.2. Ventajas de utilizar este patrón en el proyecto

Como se vio en clase, la aplicación de los patrones tiene muchas ventajas, en específico para el caso de Tomcat con el Front Controller se evidencian las siguientes ventajas:
- Es evidente la separación de preocupaciones que ofrece este patrón a Tomcat, puesto a que tener un controlador principal encargado del enrutamiento y la toma de decisiones sobre el procesamiento de las solicitudes, logra una mejor separación de preocupaciones en la aplicación. Gracias a lo anterior, los controladores específicos pueden centrarse únicamente en la lógica de negocio de cada función o acción, lo que facilita el mantenimiento y la escalabilidad del código.
- Otra de las ventajas que ofrece este patrón en el despliegue de Tomcat es la flexibilidad y extensibilidad, debido a que, al centralizar el control de las solicitudes, es más fácil agregar nuevas funcionalidades, modificar el flujo de la aplicación o realizar ajustes en el enrutamiento sin afectar en gran medida el resto del código.

1.3.3. Desventajas de utilizar el patrón en el proyecto

Aunque el patrón nos muestra grandes ventajas, también existen diversas desventajas en las que es necesario estar atento para evitar colapsos del sistema:
- Una de las desventajas más importantes es la complejidad debido a la estructura de patrón, donde se requiere un controlador principal que administre el enrutamiento, así como controladores específicos para cada acción o función. Por lo anterior, las personas/desarrolladores que no se encuentren relacionados con la estructura que maneja Front controller, les resultara muy complicado verificar posibles errores.
- Debido al diseño centralizado del patrón, el rendimiento se puede ver afectado gracias al enrutamiento adicional y la ejecución de la lógica con los controladores específicos, si se llegara a tener un tope en el canal de información, el patrón ralentizaría el funcionamiento de Tomcat.

1.3.4. Otras formas de solucionar los desafíos del proyecto sin necesidad del patrón

Si bien el patrón es realmente útil en el flujo de control de Tomcat, podemos utilizar algunas alternativas para enfrentar los desafíos del proyecto y tener unos resultados cercanos a los entregados por Front Controller, por ejemplo, podemos utilizar Frameworks de MVC, facilitándonos la separación de responsabilidades y logrando tener un flujo de control predeterminado para la puesta en acción de Tomcat, otra posible alternativa seria utilizar Frameworks de enrutamiento para maneral el enrutamiento de manera más flexible y personalizada, logrando tener más desglosadas las responsabilidades y por lo tanto mas acciones especificas para solucionar cada petición.

Referencias

[01] Apache Tomcat 9 Architecture (9.0.75) - Architecture Overview. (n.d.). Retrieved from https://tomcat.apache.org/tomcat-9.0-do/architecture/overview.html
[02] GeeksforGeeks. (n.d.). Front Controller Design Pattern. Retrieved from https://www.geeksforgeeks.org/front-controller-design-pattern/
[03] tutorialspoint.com. (n.d.). Design Pattern - Front Controller Pattern. Retrieved from https://www.tutorialspoint.com/desgn$_p attern/front_c ontroller_p attern.htm$
[04]$Baeldung.(n.d.).AGuidetotheFrontControllerPatterninJava.Retrievedfromhttps : //www.baeldng.com/java - front - controller - pattern$