1. Fundamentos

Groovy y Grails

Groovy es un lenguaje de programación dinámico que se ejecuta en la JVM (Java Virtual Machine). Es un lenguaje de programación orientado a objetos que se utiliza principalmente para escribir scripts y aplicaciones de automatización.

Por otro lado, GSP (Groovy Server Pages) es un lenguaje de plantillas que se utiliza para generar contenido HTML dinámico en aplicaciones web de Grails. Los archivos GSP se utilizan para definir la estructura y el diseño de las páginas web, mientras que los archivos Groovy se utilizan para definir la lógica de la aplicación.

Los archivos .groovy contienen código Groovy que se ejecuta en el servidor, mientras que los archivos .gsp contienen código GSP que se utiliza para generar contenido HTML dinámico en el navegador del usuario.

Groovy

```
String mensaje = "Hola, mundo!"
println  mensaje
```

GSP

```
<h1>${mensaje}</h1>
<h1>Hola, mundo!</h1>
```

Grails es un framework de aplicaciones web que se ejecuta en la plataforma Java. Está diseñado para ser altamente productivo y utiliza el lenguaje de programación Groovy. Grails utiliza el patrón de arquitectura Modelo-Vista-Controlador (MVC).

MVC

MVC es un patrón de arquitectura de software que se utiliza comúnmente en el desarrollo de aplicaciones web. MVC significa Modelo-Vista-Controlador y se divide en tres componentes principales:

Modelo: representa los datos y la lógica de negocio de la aplicación. El modelo es responsable de interactuar con la base de datos y proporcionar los datos necesarios a la vista. Aca vamos a definir clases de dominio, que atributos van a tener, de que tipo y otras configuraciones.

Vista: representa la interfaz de usuario de la aplicación. La vista es responsable de mostrar los datos al usuario y de proporcionar una forma de interactuar con la aplicación. Estos son nuestras archivos .gsp

Controlador: actúa como intermediario entre el modelo y la vista. El controlador es responsable de recibir las solicitudes del usuario, procesarlas y enviar los datos necesarios al modelo y a la vista. Va a ser el encargado de recibir y enviar la informacion a la vista

Servicios: Es donde vamos a realizar la mayor parte de la logica y es la capa que se conecta con la base de datos para manipular la data

POO

La Programación Orientada a Objetos (POO) es un paradigma de programación que se basa en el uso de objetos para representar y manipular datos. Los objetos son instancias de clases, que son plantillas que definen las propiedades y métodos de los objetos.

Los cuatro pilares de la POO son:

Encapsulamiento: es la capacidad de ocultar la complejidad de un objeto y exponer solo los métodos y propiedades necesarios para su uso. Esto permite que los objetos sean más fáciles de usar y mantener, y reduce la posibilidad de errores.

Herencia: es la capacidad de crear nuevas clases a partir de clases existentes, heredando sus propiedades y métodos. Esto permite reutilizar el código existente y crear jerarquías de clases que reflejen la relación entre los objetos.

Polimorfismo: es la capacidad de un objeto de tomar diferentes formas o comportarse de diferentes maneras según el contexto. Esto permite que los objetos se adapten a diferentes situaciones y se comporten de manera más flexible.

Abstracción: es la capacidad de representar conceptos complejos de manera simplificada. Esto permite que los objetos se centren en lo esencial y oculten los detalles innecesarios, lo que hace que el código sea más fácil de entender y mantener.

Git

Git es un sistema de control de versiones que se utiliza para rastrear los cambios en el código fuente y coordinar el trabajo en equipo. Algunos conceptos básicos de Git son:

Repositorio: es el lugar donde se almacena el código fuente y los cambios realizados en él. Un repositorio de Git puede estar alojado en un servidor remoto o en una carpeta local.

Commit: es una instantánea del código fuente en un momento determinado. Cada commit tiene un mensaje que describe los cambios realizados.

Rama (branch): es una línea de desarrollo independiente que se crea a partir de una rama principal (normalmente llamada "master"). Las ramas se utilizan para trabajar en nuevas características o correcciones de errores sin afectar la rama principal.

Fusionar (merge): es el proceso de combinar los cambios realizados en una rama con otra rama. Esto se utiliza para integrar las nuevas características o correcciones de errores en la rama principal.

Repositorio remoto: es un repositorio alojado en un servidor remoto, como GitHub o GitLab. Los repositorios remotos se utilizan para compartir el código fuente y colaborar en proyectos con otros desarrolladores.

Clonar (clone): es el proceso de copiar un repositorio remoto en una carpeta local. Esto se utiliza para obtener una copia del código fuente y trabajar en él de forma local.

Push: es el proceso de enviar los cambios realizados en una rama local a un repositorio remoto.

Pull: es el proceso de obtener los cambios realizados en un repositorio remoto y fusionarlos con una rama local.