Desarrollo Web con Grails Framework

Sistemas de Información

García Granados Alejandro, Cornejo Velázquez Eduardo

sat_vai_mal_1261@hotmail.com, ecornejo@uaeh.edu.mx

Universidad Autónoma del Estado de Hidalgo, Centro de Investigación en Tecnologías de la Información y Sistemas

RESUMEN

Una de las tendencias actuales del desarrollo Web es la necesidad de tener un incremento en la productividad de los equipo de trabajo, mediante la reutilización de código para buscar reducir los tiempos de desarrollo y de puesta en línea, así como el disminuir los riesgos asociados con la construcción de las aplicaciones. El Framework Grails es una herramienta que agiliza y simplifica el desarrollo de aplicaciones Web facilitando al programador algunas de las tareas más tediosas a la hora de escribir códigos. Grails Framework es una herramienta que permite el desarrollo ágil de aplicaciones y reduce el tiempo de entrega.

PALABRAS CLAVE

Desarrollo Web, Grails Framework, desarrollo ágil.

Introducción.

La demanda de desarrollo de aplicaciones para Internet ha crecido de forma desmesurada en los últimos 10 años (Smith, 2009). Además, los tiempos de desarrollo y puesta en línea se han acortado (Pressman, 2005). Para atender a esta demanda la filosofía Ágil de Desarrollo de Software (Agile Alliance, 2001) ha contribuido con métodos, técnicas y herramientas. Dichos métodos del desarrollo ágil de aplicaciones se basan en el principio de "*reutilización de código para aumentar la productividad y disminuir los riesgos de desarrollo*" (Rocher, 2006; Gomis, 2010).

En la mayoría de los desarrollos Web reutilizamos código de alguna forma (funciones o clases de otros proyectos propios, framework o librería incluidas, etc.) ya que ahorra tiempo y esfuerzo (Pressman, 2005; Gomis, 2010).

En este contexto, Grails Framework es una herramienta que agiliza y simplifica el desarrollo de aplicaciones Web (Rocher, 2006; Brito, 2009), permitiendo al desarrollador fortalecer la construcción de proyecto robustos, aprovechando por un lado la reutilización de código, pero sobre todo acortando los tiempo de codificación, pruebas y puesta en línea.

1. Lenguaje Groovy.

Java ha mostrado ser un lenguaje de programación poderoso y de alto nivel. Sin embargo, posee sutilezas que en algunas ocasiones vuelven tediosa y redundante la codificación. Además, a diferencia de algunos lenguajes orientados a objetos, Java no poseen características dinámicas.

El lenguaje de programación Groovy (Koening, 2007) surgió en el año 2003 gracias a la iniciativa de James Strachan y Bob McWhirter. La intención del lenguaje no es ser sucesor de Java, ya que toda su funcionalidad se encuentra disponible en Groovy.

Groovy reúne características de Java, Ruby, Python y SmallTalk en un solo lenguaje, lo que lo convierte en un lenguaje poderoso y fácil de usar. La curva de aprendizaje de Groovy para los desarrolladores Java es relativamente pequeña, lo cual les permite crear aplicaciones en Groovy de forma inmediata. Los desarrolladores de Grails eligieron a Groovy como su lenguaje base debido a su poder y dinamismo.

2. Spring Framework.

El desarrollo de aplicaciones web con Java Enterprise Edition (JEE) tiene detalles que lo hacen tedioso y repetitivo. Teniendo ello en cuenta Rod Johnson, un desarrollador férreo en JEE, publicó en su libro Expert One-on-One J2EE Design and Development (2003) una serie de herramientas que facilitan, aceleran y simplifican el desarrollo de aplicaciones Web en JEE. Este conjunto de herramientas son la base del Spring Framework.

Spring Framework es un conjunto de herramientas que promueven la separación de los módulos de aplicación mediante la *Inyección de Dependencias* (Fowler, 2004), que propone el bajo acoplamiento de componentes, así como su reutilización dentro del mismo programa. Por otra parte, Spring Framework abstrae mucha de la funcionalidad de JEE mediante la automatización de ésta o proporcionando versiones más sencillas.

En los últimos años, Spring Framework ha ganado popularidad entre los desarrolladores de software y es cada vez más utilizado en la creación de aplicaciones Web.

3. Hibernate Framework.

En JEE el uso de conexiones a bases de datos se hace a través de la API JDBC. Su funcionalidad básica consiste en realizar una conexión mediante un controlador y hacer peticiones al DBMS mediante SQL.

Existen varios inconvenientes al utilizar JDBC. En primer lugar, la gran variedad de DBMS provoca que cada uno de ellos contenga versiones diferentes de SQL, lo cual aminora la portabilidad de una aplicación y dificulta su mantenimiento. Por otro lado, la manipulación de los datos es un proceso tedioso, redundante y dependiente del gestor utilizado.

Hibernate es un framework enfocado al uso de bases de datos desarrollado por JBoss que elimina los problemas antes mencionados (Bauer, 2004). Utiliza el paradigma Object Relational Mapping (ORM) el cual permite a un programa manipular una base de datos relacional con el paradigma de programación orientada a objetos (Prokhorenko, 2004). Con ello, a nivel de la aplicación, las tablas se vuelven clases, los campos de las tablas atributos de la clase, los registros se convierten en objetos, las llaves foráneas se transforman en asociaciones entre objetos y las consultas se traducen en llamadas a métodos.

Hibernate libera al desarrollador de la escritura del 95% de sentencias SQL, lo cual aumenta la velocidad de desarrollo y portabilidad entre gestores de bases de datos. Hibernate proporciona su propio lenguaje de consultas denominado Hibernate Query Languaje (HQL), el cual es muy parecido a SQL pero con la diferencia de que es completamente orientado a objetos.

4. Grails Framework.

Grails Framework es creado por Graeme Rocher en el año 2006 como una respuesta a la necesidad de agilizar, automatizar y simplificar el desarrollo de aplicaciones Web. Se basa en el principio "*mejoremos la rueda, no la reinventemos*".

Grails está basado Hibernate y Spring (Smith, 2009), que llevan mucho tiempo funcionando, han sido probadas y se ha demostrado su correcto funcionamiento. Además, se utiliza el lenguaje Groovy que permite realizar una gran cantidad de acciones en pocas líneas de código, como se muestra en la Figura 1. En general, Grails toma las mejores prácticas de cada uno de los frameworks para formar un marco de trabajo estable, robusto, sencillo de usar y de fácil mantenimiento.

Del libro Grails in Action (Smith, 2009) retomamos las "7 grandes ideas" que hacen único a Grails Framework y lo distingue de otras herramientas:

a. *Convención sobre configuración*. Grails posee una estructura especial para la ubicación de cada uno de sus elementos, clases y archivos de configuración. Esto evita que el desarrollador tenga que hacer la configuración mediante XML.

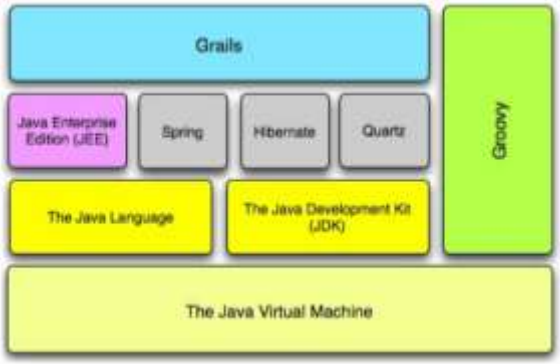

Fig 1. Arquitectura del Grails Framework (Smith, 2009).

b. *Filosofía ágil*. Grails permite la reutilización de código para aumentar la productividad; además, es posible hacer cambios y visualizarlos en tiempo real sin necesidad de reiniciar el servidor de aplicaciones.
c. *Fundamentos sólidos.* Grails se basa en las mejores herramientas de desarrollo existentes, lo que brinda un soporte sólido para el desarrollo de aplicaciones.
d. *Plantillas y Scaffolding*. Grails maneja el concepto de Scaffolding para la generación automática de código para controladores y vistas basado en una serie de plantillas o templates.
e. *Integración con Java*. Módulos desarrollados en Java pueden ser integrados a Grails de forma natural. Lo anterior, debido a que Groovy es completamente compatible con Java, permitiendo utilizar cualquier clase, API y biblioteca de Java en Grails.
f. *Wetware*. Grails ha ganado muchos seguidores, lo que ha incrementado el número de recursos dedicados al soporte y mantenimiento. Sitios Web, foros de discusión, libros, podcast y plugins son algunas de las herramientas disponibles.
g. *Productividad*. Una de las consecuencias más evidentes del uso de Grails es el nivel de productividad que se logra. Una aplicación que podría llevar semanas o incluso meses queda lista en cuestión de horas o días, disminuyendo el tiempo de entrega.

Por otro lado, una aplicación desarrollada con Grails Framework sigue la arquitectura Modelo/Vista/Controlador (MVC) (Koening, 2007), Figura 2. Donde el *Modelo* es el objeto que representa los datos del programa, la *Vista* es el objeto que maneja la presentación visual de los datos representados por el Modelo y el *Controlador* es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo.

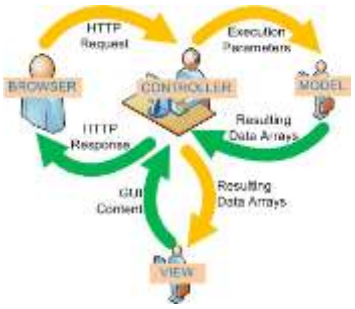

Figura 2. Arquitectura MVC (Koening, 2007).

Desarrollo.

En esta sección se presenta el proceso general para para construir una nueva aplicación Web utilizando el Grails Framework. Para ellos se requiere descargarlo desde el portal www.grails.org y seguir las instrucciones de instalación disponibles en el paquete descargado.

A continuación se describe el proceso para construir la aplicación *blogClase*, basada en el diagrama de clases mostrado en la Figura 3, utilizando los comandos básicos de Grails para construir los elementos de la aplicación y probar su ejecución.

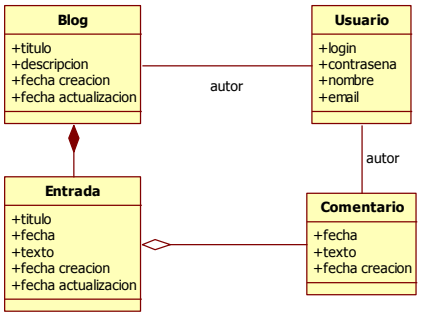

Figura 3. Diagrama de clases de la aplicación blogClase.

a) Para crear una nueva aplicación se utiliza la siguiente línea de comando:

```
> grails create-app blogClase
```

Como resultado de lo anterior, se genera un directorio con el nombre blogClase y dentro de este se ubican los archivos y directorios necesario para la aplicación siguiendo el principio de Convención sobre Configuración.

b) Creación de las clases de dominio, o modelos de la arquitectura MVC, para nuestro ejemplo tenemos 4 clases: usuario, entrada, comentario y blog. Se crea la clase de dominio para usuario con el siguiente comando:

```
> grails create-domain-class Usuario
```

El resultado es la generación del archivo *usuario.groovy*, cuyo contenido es el siguiente:

```
1.  package blogClase.usuario
2.  class Usuario {
3.      static constraints = {
4.      }
5.  }
```

Para modelar la clase Usuario de acuerdo con el diagrama de clase, se modifica el archivo *usuario.groovy* agregando las siguientes líneas después de la línea 2.

```
String login
String contrasena
String nombre
String email
```

Además, se incorporar las siguientes reglas de validación para los atributos de la clase, después de la línea 3, dentro de las llaves {}.

```
login size:1..13,nullable: false, blank: false
contrasena size: 1:13,nullable: false, blank: false
nombre size: 1:100,nullable: false, blank: false
email size: 1:60,email: true
```

Con lo anterior, se han agregado los atributos y restricciones de la clase de dominio: tamaños (`size`), no permitir nullos (`nullable`), no aceptar cadenas vacías (`blank`) y que sea una dirección de correo electrónico válida (`email`).

c) Para la generación del controlador de la clase Usuario para administrar las peticiones del cliente Web, se utiliza el siguiente comando:

```
> grails generate-controller Usuario
```

El resultado es la generación del archivo *UsuarioController.groovy* que contiene el código necesario para atender las peticiones CRUD (Create/Retrieve/Update/Delete) del usuario.

d) Con lo anterior se tiene construido el Modelo y el controlador de la Aplicación, resta construir las vistas para la clase Usuario, para ello se utiliza el siguiente comando:

```
> grails generate-views Usuario
```

El resultado es la generación de los archivos `create.gsp`, `show.gsp`, `edit.gsp` y `list.gsp` para las vistas CRUD de la clase Usuario.

e) Grails permite la creación de catálogos CRUD de forma rápida. Por defecto, Grails utiliza la tecnología HSQLDB (http://hsqldb.org/) para generar una base de datos in memory (en memoria), lo que significa que mientras el servidor de aplicaciones no se apague o reinicie, los datos almacenados en memoria se conservan. El uso de esta tecnología permite al desarrollador hacer pruebas de su aplicación sin necesidad de realizar la conexión con una base de datos real.

Aprovechando esta funcionalidad. Podemos probar la aplicación creada con el siguiente comando:

```
> grails run-app
```

Si todo marcha bien, al final de su ejecución, Grails muestra la siguiente salida:

```
Server running. Browse to http://localhost:8080/blogClase
```

f). Para visualizar la aplicación, se debe abrir un navegador Web y dirigirse a la dirección

```
http://localhost:8080/blogClase
```

Grails genera y despliega de forma automática toda la infraestructura necesaria para una aplicación Web ¡con tan solo cinco comandos!. No hay necesidad de instalar un servidor de aplicaciones, no hay necesidad de escribir archivos XML como ocurriría con cualquier otro framework de desarrollo Web y no hay necesidad de escribir HTML para la página inicial. Grails ahorra todos esos pasos tediosos y repetitivos.

Para completar la aplicación es necesario seguir el procedimiento descrito para las otras tres clases del modelo de clases.

Conclusiones

Se han mostrado las capacidades básicas de Grails Framework, para el desarrollo de aplicaciones Web. El desarrollo de aplicaciones Web con Grails se convierte en una experiencia ágil e interesante; ya que con escribir algunos comandos y algunas líneas de código, se puede implementar una aplicación Web completamente funcional, de fácil mantenimiento y sencilla de usar.

El Grails Framework es una herramienta que presenta una curva de aprendizaje corta para los programadores de JEE y proporciona grandes beneficios en las etapas de desarrollo y permite acortar los tiempos, posibilitando la disminución de los tiempo de entrega.

Referencias

- Agile Alliance Home Page. (2001). Disponible en: www.agilealliance.org
- Bauer, C., King, G. (2004). Hibernate in Action. Manning Publications Co.
- Brito, N. (2009). Manual de desarrollo Web con Grails, JavaEE como siempre debió haber sido. ImaginaWorks.
- Fowler, M. (2004). The Dependency Injection pattern. Obtenido en febrero de 2011 de http://martinfowler.com/articles/injection.html
- Gomis, R. (2010). Grails: un paso hacia el desarrollo Web ágil. Disponible en: www.eltallerdigital.com/informacion.jsp?idArticulo=119
- Koening, D., Glover, A. (2007). Groovy in Action. Second Edition. Manning Publications Co.
- Pressman, R. (2005). Ingeniería de Software, Un enfoque práctico. Sexta Edición. Editorial Mc Graw Hill.
- Prokhorenko, A. (2004). An Introduction to Object-Relational Mapping with Hibernate. Disponible en: http://www.developer.com/java/other/article.php/10936_3391131_1
- Rocher, G. (2006). The Definitive Guide to Grails. USA. Editorial Apress.
- Smith, G., Ledbrook, P. (2009). Grails in Action. Manning Publications Co.