VÍNCULOS
ENERO - JUNIO DE 2010
VOLUMEN 7 ■ NÚMERO 1

Arquitectura multicapa mediante AJAX y ORM

Multilayer architecture driven AJAX and ORM

Héctor Arturo Flórez Fernández*

Fecha de recepción: abril 30 de 2010
Fecha de aceptación: mayo 19 de 2010

Resumen

Este artículo pretende mostrar una arquitectura multicapa, basada de la arquitectura de tres capas, la cual complementa la capa de presentación con el concepto de AJAX y la capa de persistencia con el concepto de ORM. Basado en estos conceptos, es posible realizar proyectos de software modernos bajo la web, con grandes beneficios que siguen las buenas prácticas en el diseño y desarrollo de proyectos que plantea la ingeniería de software.

Abstract

This article aims to show a multi-layer architecture, based on the architecture of three layers, complementing the presentation layer with the AJAX concept and the persistence layer with the ORM concept. Based on these concepts, it is possible to make modern software projects on the web, with great benefits that follow best practices in the design and development of projects arising from software engineering.

Palabras Clave: AJAX, Arquitectura de tres capas, GWT, Hibernate, ORM, Ingeniería de Software
Keywords: AJAX, Three layer architecture, GWT, Hibernate, ORM, Software Engineering

* Ingeniero electrónico e ingeniero de sistemas de la Universidad El Bosque, magister en Ciencias de la Información y las Comunicaciones de la Universidad Distrital Francisco José de Caldas, especialista en Alta gerencia y candidato a Magister en Gestión de Organizaciones. Docente investigador Fundación Universitaria Konrad Lorenz, docente Universidad Distrital Francisco José de Caldas. Adscrito al grupo de de investigación PROMENTE de la Fundación Universitaria Konrad Lorenz. Correo electrónico: hectorarturo@yahoo.com.

Introducción

Actualmente, los proyectos de software se desarrollan mediante metodologías y paradigmas que impulsan el uso de arquitecturas, como la arquitectura de tres capas. Sin embargo, los proyectos de software bajo la web requieren de forma inminente el uso de técnicas sofisticadas para la interfaz de usuario que permita una mayor efectividad y funcionalidad en los procesos que allí se involucran. Así mismo, debido a que los modelos de bases de datos son relacionales, se requieren modelos o técnicas que permitan realizar la persistencia de una forma robusta.

Por estos motivos, se encuentra que AJAX es un concepto que permite el desarrollo de interfaces de usuario asíncronas, lo cual genera grandes beneficios en la funcionalidad de los proyectos web. Así mismo, ORM permite realizar un mapeo entre el modelo relacional y el modelo de negocio, al proporcionar herramientas para realizar persistencia de los datos de forma simple y confiable.

Arquitectura de tres capas

La arquitectura de tres capas es una técnica en el desarrollo de aplicaciones de software que tiene como objetivo separar la lógica del negocio de la presentación y de la persistencia.

Una de las principales ventajas se obtiene con el bajo acoplamiento de las aplicaciones debido a que ésta característica permite fácilmente realizar cambios en los servicios, sin tener que revisar todos los componentes de la aplicación. Además, esta técnica permite distribuir el trabajo de los desarrolladores por niveles, en donde cada equipo de desarrollo puede hacer uso de los componentes desarrollados por otro equipo sin necesidad de conocer el desarrollo, solo conociendo los resultados de los servicios.

La división en componentes reduce la complejidad, permite la reutilización y acelera el proceso de ensamblaje de software. Los creadores de componentes pueden especializarse creando objetos cada vez más complejos y de mayor calidad (Cordero; 2006).

Por otro lado, esta técnica permite hacer correcto uso de las metodologías de desarrollo de software. Particularmente, la metodología RUP (Rational Unified Proccess) se adapta cómodamente a la arquitectura y permite la facilidad en los procesos de desarrollo de la aplicación.

La arquitectura de tres capas se basa en las capas de Presentación, Lógica y Persistencia.

La capa de Presentación presenta el sistema al usuario, le muestra la información y captura la información del usuario. Esta capa se comunica con la capa de negocio por medio de objetos que se denominan "Object value".

La capa de Lógica de negocios donde se desarrollan los algoritmos propios de la aplicación. En esta capa se implementa la lógica obtenida por el análisis de requerimientos del proyecto. Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados y con la capa de persistencia, para solicitar la información pertinente al motor de base de datos.

La capa de Persistencia es donde se almacenan los datos los datos y se realiza las operaciones para acceder a los mismos. Esta capa recibe solicitudes de almacenamiento o recuperación de información desde la capa de la lógica del negocio.

La arquitectura de tres capas tiene una característica adicional, que es la facilidad de aplicación de patrones de desarrollo de software. Además, genera grandes beneficios para el proyecto ya que permite realizar escalabilidad, portabilidad usabilidad entre otros.

AJAX (*Asynchronous Javascript And Xml*)

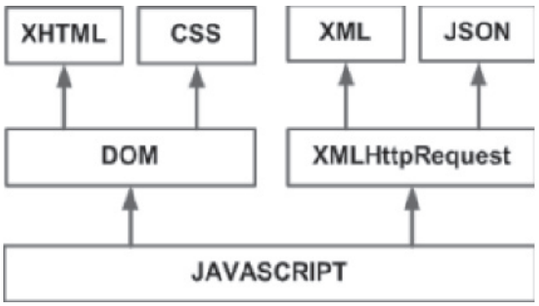
AJAX es la técnica que permite generar aplicaciones web de forma interactiva al manejar una estructura que permite al servidor web comunicarse con el navegador del usuario por componentes individuales. AJAX intenta proveer aplicaciones web con las características de las aplicaciones del escritorio y permite que los datos que se cargan a la aplicación no requieran una recarga del sitio web; de esta forma, permite más velocidad y robustez en la aplicación web.

AJAX es un concepto basado en muchas tecnologías como XHTML y CSS, DOM, XML, XSLT y JSON, XML, Http, Request y Javascript. AJAX reúne las anteriores tecnologías con el fin de poder obtener acceso al servidor de aplicaciones, sin requerir un nuevo llamado mediante protocolo HTTP.

El término AJAX se presentó por primera vez en el artículo "Ajax: A New Approach to Web Applications", publicado por Jesse James Garrett el 18 de Febrero de 2005. Hasta ese momento, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación web que estaba apareciendo. (Eguiluz; 2008).

AJAX es un acrónimo de Asynchronous JavaScript and XML. El siguiente modelo representa el concepto de AJAX con base en las tecnologías mencionadas anteriormente.

Figura 1. Tecnologías usadas en AJAX (Eguiluz; 2008)

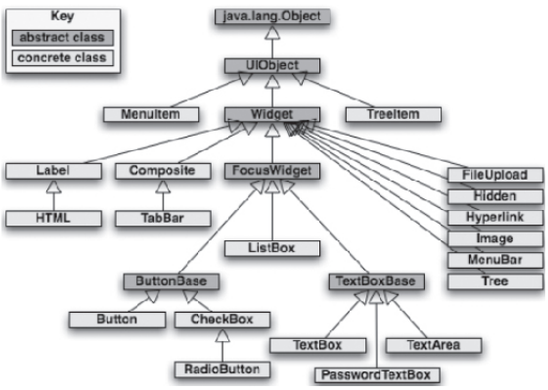


GWT T2

GWT es un *framework* para la construcción de aplicaciones en JAVA que permite escribir AJAX de forma simple. (Shing; 2007).GWT es una aplicación creada por Google, la cual ofrece una capa de presentación utilizando los conceptos de AJAX. GWT compila el código generando los elementos JavaScript necesarios para la utilización de AJAX.

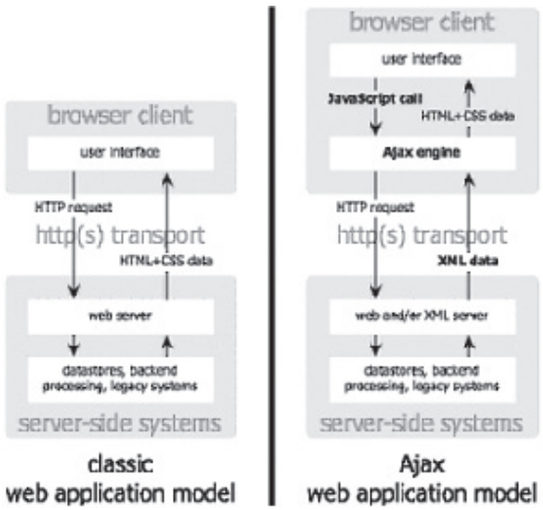
GWT utiliza Widgets, los cuales son componentes o controles visuales que permiten la interacción entre el usuario y la aplicación. GWT obtiene el siguiente modelo de clases de Widgets como soporte para el desarrollo gráfico.

Figura 2. Diagrama de Clases de Widgets de GWT (Volkman; 2006)



La siguiente figura muestra una comparación de los modelos de desarrollo de aplicaciones web cotidianas y aplicaciones web con AJAX:

Figura 3. Modelo de desarrollo clásico y usando AJAX (Stimpson;2006)



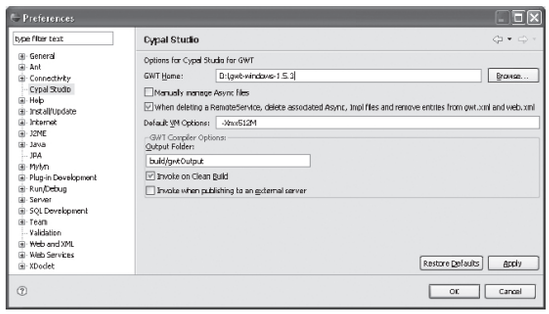
Una buena opción para desarrollar aplicaciones utilizando GWT es con la herramienta de desarrollo Eclipse, en su versión 3.3.2, ya que permite la integración de los componentes de forma sencilla.

Configuración de GWT

Para iniciar, se debe descargar GWT (http://code.google.com/intl/es-ES/webtoolkit/download.html) y un *plugin* para Eclipse denominado CYPAL (http://code.google.com/p/cypal-studio/downloads/list). El GWT se debe descomprimir en cualquier lugar de la máquina y el CYPAL se debe descomprimir en la carpeta *plugins* de Eclipse.

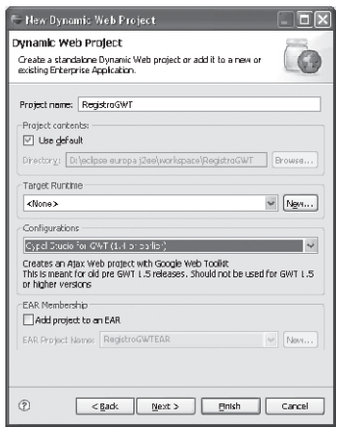
Al ejecutar Eclipse hay que configurar el GWT y, para ello, se ingresa al menú Windows - Preferences de Eclipse, se selecciona Cypal Studio y se indica la ruta donde se descomprimió GWT.

Figura 4. Configuración GWT



Para crear una aplicación en GWT se debe crear un proyecto dinámico en Eclipse por medio de File- new-Project-web-dynamic Web Project. En configuraciones, se debe colocar Cypal Studio for GWT.

Figura 5. Configuración proyecto GWT



Después de haber creado el proyecto, se agrega un nuevo módulo al proyecto creado: new -other - Cypal Studio - gwt Module. Es estrictamente necesario que el nombre del paquete este totalmente en minúscula.

Figura 6. Creación de un módulo GWT

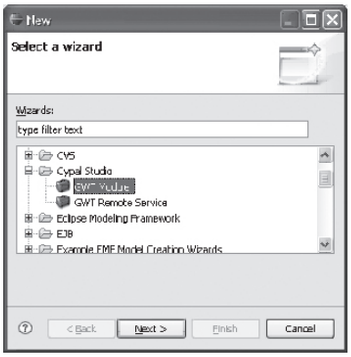
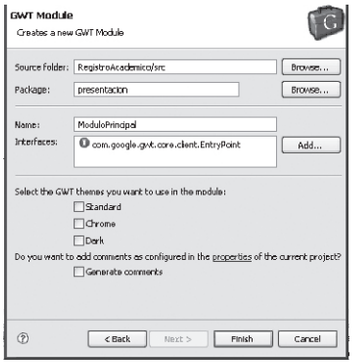


Figura 7. Creación de un módulo GWT



GWT EXT

GWT contiene una extensión bastante útil que se denomina gwt-ext, que es un wrapper, es decir, convierte una interfaz en una simple clase, permite adherirse como una biblioteca y acceder fácilmente a la creación de ventanas, menús y otros. Se debe descargar el gwt-ext 2.0.5 de la dirección http://gwt-ext.com/download/ . Éste contiene el archivo gwtext.jax, el cual se debe ubicar en una carpeta denominada "lib" en la raíz del proyecto. Posteriormente, se agrega en el BuildPath este archivo como una librería. Después se debe descargar el archivo ext 2.0.2. En el paquete creado en el módulo, se crea una carpeta llamada "public". En ella se debe crear la carpeta js/ext y en ésta se descomprime los siguientes elementos del gwt 2.0.2:ext-all.js, ext-all-debug.js, - ext-core.js, ext-core-debug.js, \ adapter, \resources. Al crear el módulo, se genera también un archivo xml llamado en este caso "ModuloPrincipal.gwt.xml" el cual debe quedar con las siguientes líneas de código que permiten incluir el gwtext:

```
<module>
<!-- Inherit the core Web Toolkit
stuff.-->
<inherits name='com.google.gwt.
user.User'/>
<!-- Inherit the GWTExt Toolkit
library configuration.   -->
<inherits name='com.gwtext.GwtExt'
/>
<!-- Specify the app entry point
class. -->
<entry-point class='presentacion.
client.ModuloPrincipal'/>
<stylesheet src="js/ext/resources/
css/ext-all.css" />
<script src="js/ext/adapter/ext/
ext-base.js" />
<script src="js/ext/ext-all.js" />
</module>
```

Desarrollo con GWT Y GWT-EXT

Para el desarrollo de proyectos con GWT-EXT es necesario revisar la documentación de este API. Sin embargo, el elemento fundamental para la implementación es el módulo creado y en el que se debe iniciar el desarrollo de la presentación, de acuerdo a las necesidades del proyecto. Este módulo contiene un objeto heredado de Entry Point llamado Root Panel, en el cual se deben agregar todos los componentes que se desean visualizar. El siguiente código muestra un formulario construido con GWT-EXT.

```
public class ModuloPrincipal implements EntryPoint {

      public void onModuleLoad() {
            Panel panel = new Panel();
panel.setBorder(false);
panel.setPaddings(15);
final FormPanel formPanel = new FormPanel();
formPanel.setFrame(true);
formPanel.setTitle("Formato Agregar estudiante");
formPanel.setWidth(250);
formPanel.setLabelWidth(75);
TextField TCodigo = new TextField("Codigo");
TCodigo.setAllowBlank(false);
formPanel.add(TCodigo);
TextField TNombre = new TextField("Nombre");
TNombre.setAllowBlank(false);
formPanel.add(TNombre);
TextField TApellido = new TextField("Apellido");
TNombre.setAllowBlank(false);
formPanel.add(TApellido);
Button BGuardar = new Button("Guardar");
formPanel.addButton(BGuardar);
panel.add(formPanel);
RootPanel.get().add(panel);
      }
}
```

El resultado es el siguiente:

Figura 8. Formulario GWT-EXT

ORM (object-relational mapping)

El mapeo del dominio de objetos dentro de un modelo relacional es importante para el proceso de desarrollo de software moderno. Los lenguajes de programación, orientados objetos como Java, C# y C++ son los más comúnmente aplicados para el desarrollo de nuevos sistemas de software. Además, las bases de datos relacionales siguen el enfoque preferido para el almacenamiento de información persistente y es probable que siga siéndolo para un futuro cercano.

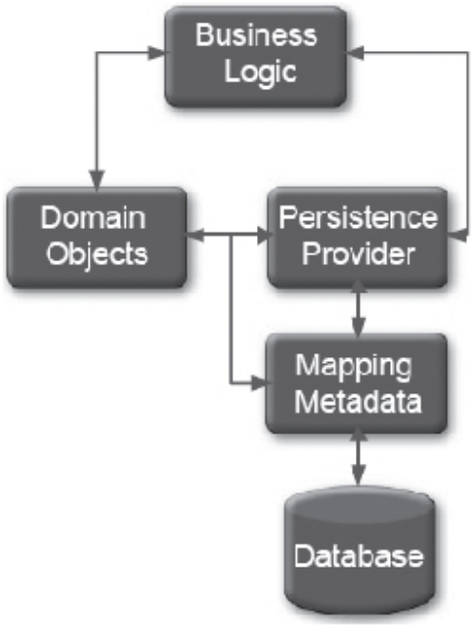
El mapeo objeto-relacional permite formarun mapeo de una base de datos relacional, generando en la aplicación orientada a objetos clases que son directamente la estructura de la base de datos que se posee. Es decir,

en la aplicación se puede tener una base de datos orientada a objetos virtuales sobre la base de datos relacional. Esta característica permite aplicar conceptos de orientación a objetos como herencia y polimorfismo, a los datos almacenados de forma relacional (Doroshenki y Romanenko; 2005).

La implementación de un mapeo objeto-relacional no es trivial, debido a los diferentes paradigmas en la aplicación de modelos de dominio. El objeto de paradigma se basa en principios de ingeniería de software como acoplamiento, cohesión y encapsulado, mientras que el paradigma relacional se basa en principios matemáticos, en particular los de relación y la teoría de conjuntos. Los dos fundamentos teóricos llevan a diferentes puntos fuertes y debilidades. Además, el paradigma de objeto se centra en aplicaciones de construcción de objetos que tienen datos y comportamiento, mientras que el paradigma relacional se centra en el almacenamiento de datos. Esta diferencia fundamental se traduce en una menor a la combinación ideal de los dos paradigmas (Doroshenki y Romanenko; 2005).

La solución de problemas en la aplicación de mapeo objeto-relacional ha modificado la forma de desarrollar aplicaciones, porque los lenguajes orientados a objetos y bases de datos relacionales se han generalizado desde hace mucho tiempo y con frecuencia se usan juntos con el desarrollo de patrones de diseño de metodología. La siguiente figura muestra el modelo de persistencia mediante ORM:

Figura 9. Modelo ORM (Russell, Meswani, White; 2007)



Hibernate

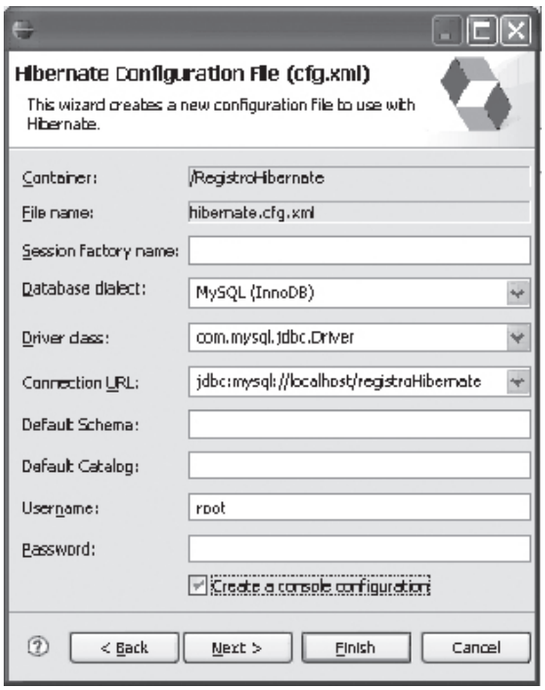
Hibernate es un componente de software libre para JAVA que contiene la funcionalidad necesaria para implementar un proyecto mediante ORM. Hibernate típicamente es implementado usando Eclipse. Para la instalación y uso de hibérnate, se debe realizar los siguientes procesos.

Se debe descargar el *plugin* Hibernate Tools y descomprimirlo en la carpeta Eclipse. Posteriormente, se inicia Eclipse y se crea un proyecto Java. Es necesario a este proyecto colocarle la opción "Enable Hibernate" en Properties – Hibernate Settings.

Se debe crear una carpeta "lib" en la raíz del proyecto e incluir el archivo hibernate3.jar y el conector de la base de datos en donde se está haciendo la persistencia. En este caso, la base de datos es MySQL y el conector se denomina mysql-connector-java-5.0.0.

A su vez, se debe crear el archivo de configuración hibernate.cfg.xml en el proyecto mediante File – new – Other – Hibernate – Hibernate configuration file (cfg.xml)(CCIA; 2006) y se deben hacer la configuración de la figura siguiente:

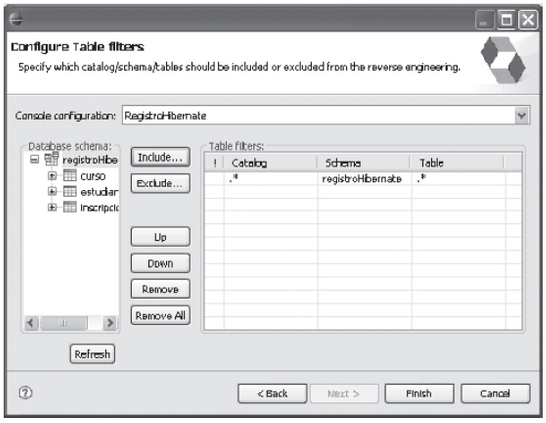
Figura 10. Configuración del archivo hibernate.cfg.xml



En el campo Driver Class se debe colocar el controlador de conexión de la base de datos, que en este caso es de MySQL. Connection URL contiene el servidor de la base de datos y el nombre de la misma.

Igualmente, se debe crear el archivo de configuración de ingeniería inversa reveng.cfg.xml en el proyecto mediante File - new - Other - Hibernate - Hibernate reverse engineering file (reveng.xml), y se deben hacer la configuración de la figura:

Figura 11. Configuración del archivo reveng.xml



En el campo Console Configuration se debe seleccionar el proyecto Hibernate y al dar *click* en *refresh*. A continuación, se actualiza la lista izquierda con la base de datos configurada en el archivo hibérnate.cfg.xml. Posteriormente se agrega a la lista derecha la base de datos a la que se desea hacer el proceso ORM.

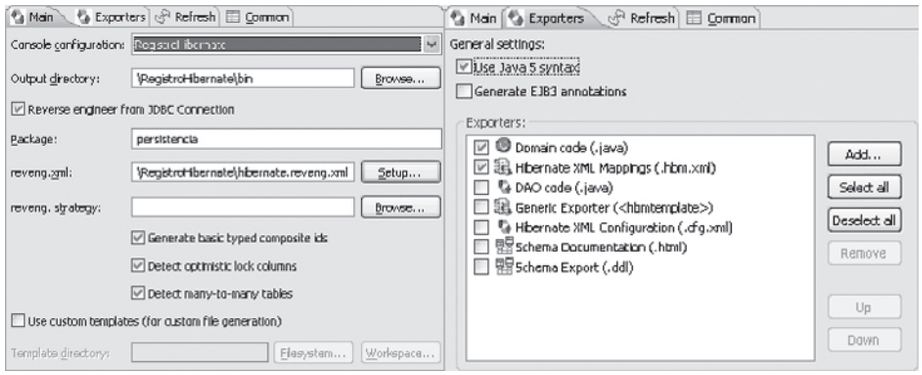
Finalmente, se debe ejecutar Hibernate para realizar el mapeo ORM. Al realizar la instalación del *plugin* de Hibernate, este ha generado en la barra de herramientas un icono para la ejecución del mapeo ORM.

Figura 12. Icono de ejecución de mapeo ORM de Hibernate

Al dar*click*sobre este icono podemos realizar la configuración para el generador de Hibernate. En la pestaña *main*, es necesario colocar en el campo Console configuration el proyecto Hibernate. En output *directori* la carpeta *bin*, package en el paquete en el que se desea colocar las clases del mapeo en reveng.xml, el archivo creado anteriormente.

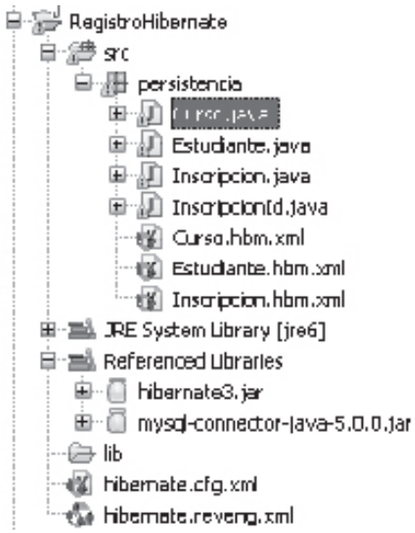
Adicionalmente, en la pestana *exporter* se debe seleccionar los archivos que se desean generar. Para este caso se selecciona Domain code, que son las clases del mapeo y Hibernate XML Mappings, que son archivos xml que especifican las relaciones existentes en la base de datos.

Figura 13. Configuración Hibernate



Al ejecutar la configuración se obtiene los siguientes resultados:

Figura 14. Resultados ORM con Hibernate

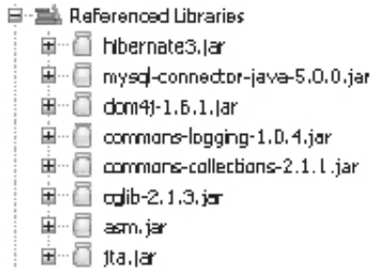


Los archivos .java contienen el código con los atributos y métodos modificadores de cada una de las tablas de la base de datos. Los archivos .hbm.xml contienen la especificación de los atributos, llaves primarias y foráneas de la base de datos.

Desarrollo con Hibernate

Para comenzar a desarrollar ORM con Hibernate en Eclipse, es necesario incluir librerías que se encuentran como archivos .jar. Estas librerías deben quedar en el proyecto Hibernate creado como se muestra en la siguiente figura.

Figura 15. Librerías para desarrollo con Hibernate.



Basado en estas librerías y en la arquitectura multinivel planteada, en el proyecto Hibernate se crea un paquete "dao", donde tenemos las clases que utilizan la persistencia Hibernate.

Además, es necesario modificar el archivo de configuración cfg.xml, agregando las clases que se desean mapear de la base de datos. Igualmente, es importante incluir la línea "property name = "current_session_context_class"> thread </property>" debido a que por medio de esta línea Hibernate puede crear una sesión. Entonces, el archivo de configuración contiene lo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
            "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
            "http://hibernate.sourceforge.net/
hibernate-configuration-3.0.dtd">
<hibernate-configuration>
<session-factory>
<property name="hibernate.connection.driver_class">com.mysql.jdbc.
Driver</property>
<property name="hibernate.connection.url">jdbc:mysql://localhost/
registroHibernate</property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.connection.password"></property>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</
property>

            <property name="use_outer_join">true</property>
            <property name="show_sql">true</property>
            <property name="current_session_context_class">thread</
property>

            <mapping resource="persistencia/Estudiante.hbm.xml" />
            <mapping resource="persistencia/Curso.hbm.xml" />
            <mapping resource="persistencia/Inscripcion.hbm.xml" />
</session-factory>
</hibernate-configuration>
```

Inserción y actualización con Hibernate

El código para insertar o actualizar con Hibernate es el siguiente:

```
public class EstudianteDAO {
      private persistencia.Estudiante estudianteHibernate;

      public EstudianteDAO(persistencia.Estudiante estudianteHibernate)
{
            this.estudianteHibernate=estudianteHibernate;
      }

      public void insertar(){
            try {
                  SessionFactory sessionFactory = new Configuration()
                  .configure(new File("hibernate.cfg.xml")).
buildSessionFactory();
                  Session session = sessionFactory.getCurrentSession();
                  Transaction tx = session.beginTransaction();
                  session.saveOrUpdate(estudianteHibernate);
                  tx.commit();
            } catch (HibernateException e) {
                  e.printStackTrace();
            }
      }
}
```

La clase Session Factory, permite abrir el archivo cfg.xml, en donde se encuentra la información de mapeo ORM. Posteriormente se debe utilizar una transacción y con la sesión creada se utiliza el método saveOrUpdate donde se envía como parámetro un objeto que sea instancia de una clase creada para el mapeo ORM con hibernate. En este método, si el registro que no existe lo crea pero si existe lo actualiza. Hibernate arroja los siguientes resultados si el elemento se inserta.

```
Hibernate: select estudiante_.codigo, estudiante_.nombre as nombre0_,
estudiante_.apellido as apellido0_ from registroHibernate.estudiante
estudiante_ where estudiante_.codigo=?
Hibernate: insert into registroHibernate.estudiante (nombre, apellido,
codigo) values (?, ?, ?)
```

O los siguientes resultados si el elemento se actualiza:

```
Hibernate: select estudiante_.codigo, estudiante_.nombre as nombre0_,
estudiante_.apellido as apellido0_ from registroHibernate.estudiante
estudiante_ where estudiante_.codigo=?
Hibernate: update registroHibernate.estudiante set nombre=?, apellido=?
where codigo=?
```

Consulta con Hibernate

Existen varias formas de realizar consultas en Hibernate. Una forma es por medio de un objeto criteria. El código para consultar mediante el objeto *criteria* con Hibernate es el siguiente:

```
      public List consultar(){
            List resultados=null;
            try {
                  SessionFactory sessionFactory = new Configuration()
.configure(new File("hibernate.cfg.xml")).buildSessionFactory();
                  Session session = sessionFactory.getCurrentSession();
                  Transaction tx = session.beginTransaction();
                  Criteria criteria=session
.createCriteria(persistencia.Estudiante.class);
                  resultados=criteria.list();
                  tx.commit();
            } catch (HibernateException e) {
                  e.printStackTrace();
            } catch (Exception e){
                  e.printStackTrace();
            }
            return resultados;
      }
```

Al objeto *criteria* se le asigna la clase que se desea consultar. El método list devuelve en un java.util.List los resultados de la consulta.

Arquitectura multicapa

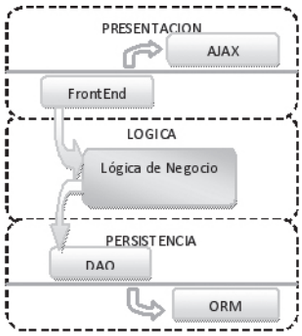
Sustentado en los conceptos de AJAX, que permite un desarrollo sencillo y robusto en la capa de presentación en la arquitectura de tres capas, se puede considerar una ampliación de esta capa utilizando una capa FrontEnd, la cual contiene la lógica de presentación, y otra capa donde se incluya el componente GWT.

Igualmente, basado en los conceptos de ORM, que permite el desarrollo sencillo y robusto en la capa de persistencia en la arquitectura de tres capas, se puede considerar una ampliación de

esta capa utilizando una capa DAO que contiene la lógica de la persistencia y otra capa donde se incluya el componente Hibernate.

De esta forma, el concepto de arquitectura multicapa se plantea en la siguiente figura:

Figura 16. Arquitectura multicapa usando AJAX y ORM



En este modelo se aprecia la independencia de los elementos y robustez en el desarrollo de aplicaciones de software, debido a que el diseño se convierte en un proceso transparente y el desarrollo de los módulos, requeridos para la presentación del proyecto web; a su vez, se plantea una solución de persistencia que permita la integración del modelo del negocio con el modelo relacional del proyecto.

Estas características generan mayor potencial en la aplicación, ya que permiten escalabilidad, alta cohesión y bajo acoplamiento.

Conclusiones

Una arquitectura de tres capas es una solución apropiada para cualquier proyecto de software, ya que permite implementar buenas prácticas de la ingeniería de software. Sin embargo, dados los avances para las aplicaciones, se hace necesario elevar apropiadamente el número de capas utilizando nuevos componentes que resuelven situaciones particulares dentro de los proyectos de software.

AJAX es una solución excelente para el desarrollo de aplicaciones web dinámicas. GWTpermite obtener grandes características en el proyecto basado en Java, debido a que contiene dentro de su desarrollo componentes que implementan el concepto de AJAX.

ORM plantea un mapeo de la base de datos relacional hacia el proyecto desarrollado al usar el paradigma de orientación a objetos. Hibernate implementa para Java y .NET el concepto de ORM al permitir a la aplicación usar la persistencia de forma transparente al modelo relacional. Hibernate se basa de XML para definir las características de las tablas, atributos y relaciones de la base de datos utilizada.

Por medio de la ingeniería inversa de Hibernate se puede lograr hacer el mapeo de la base de datos de forma automática, al definir el motor de la base de datos y la información de configuración de la misma. Este servicio es un aporte fundamental, ya que ahorra bastante esfuerzo en el proceso de desarrollo de las aplicaciones.

Una arquitectura multicapa que considere en la capa de presentación dos subcapas como FrontEnd, la cual tiene la lógica de la presentación y AJAX, contiene el API para el manejo de la presentación y considera en la persistencia dos subcapas como DAO, la cual tiene los servicios para el repositorio de la información, y ORM, la cual contiene el API para la persistencia mediante el mapeo objeto-relacional, provee un modelo que facilita el desarrollo de aplicaciones de software, debido a que el proceso de desarrollo se basaría en la lógica del negocio. De esta forma, el equipo de desarrollo deberá concentrarse solamente en los requerimientos del cliente, ya

que el manejo de la presentación y persistencia se convierte en un proceso simple y transparente.

Referencias bibliográficas

Cordero, R(2006). *Introducción al diseño y a la programación orientada a objetos*. Disponible en: http://www.nielsoft.com/seminario/3capas/introduccion.ppt

Eguiluz, J. (2008).*Introducción a AJAX*. Disponible en: www.librosweb.es

Shing, Sg. (2007).*Google Web Toolkit (GWT), Java Technology Architect & Evangelist Sun Microsystems, Inc.*

Volkman, M. (2006).*Google Web Toolkit (GWT)*.

Stimpson , B (2006).*Google Web Toolkit (GWT)*.

Dali, R. (2007).*Object-Relational Mapping Tool*.

Doroschenlo, A., Romanenko, V.,(2005). *Object-Relational Mapping Techniques for .Net Framework*.

Russell, C. , Meswani, M., White, L. (2007) *Architecture of Popular Object-Relational Mapping Providers*.

CCIA. (2006). *Ejercicios de Persistencia. Introduccion a Hibernate*.

Fuentes electrónicas

Sitio web RUP. http://www-01.ibm.com/software/awdtools/rup/

Sitio web GWT. http://code.google.com/intl/es-ES/webtoolkit/

Sitio web Hibernate. https://www.hibernate.org/

Sitio web JAVA. http://www.java.com/