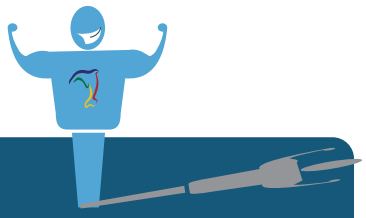
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

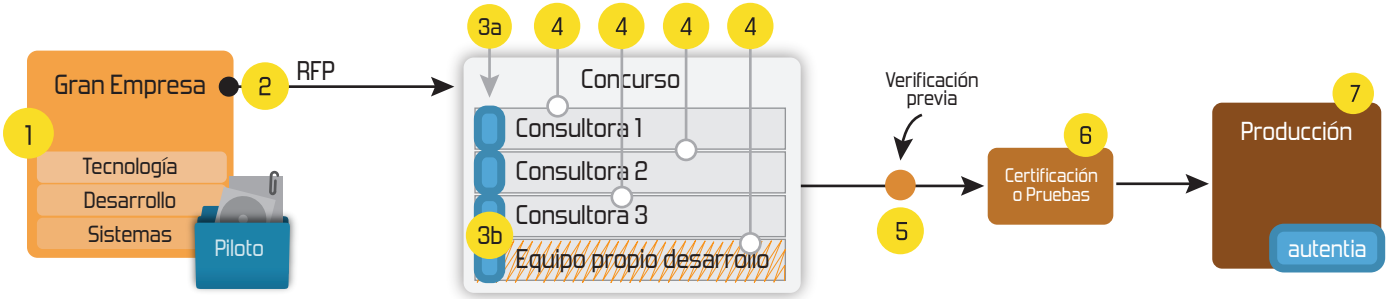
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Lanzado TNTConcept versión 0.6 (12/07/2007)

Desde Autentia ponemos a vuestra disposición el software que hemos construido (100% gratuito y sin restricciones funcionales) para nuestra gestión interna, llamado TNTConcept (au**T**e**NT**ia).

Construida con las últimas tecnologías de desarrollo Java/J2EE (Spring, JSF, Acegi, Hibernate, Maven, Subversion, etc.) y disponible en licencia GPL, seguro que a muchos profesionales independientes y PYMES os ayudará a organizar mejor vuestra operativa.

Las cosas grandes empiezan siendo algo pequeño ..... Saber más en: http://tntconcept.sourceforge.net/

Autor del tutorial: *Cristhian Kirs Herrera Basurto*

- ***Lugar de residencia*:** Quito - Ecuador

Cuento con experiencia en el área de desarrollo de software y en la docencia académica. Dentro de la construcción de software he manejado las etapas de: análisis, diseño, personalización e implementación de aplicaciones bajo ambientes Cliente / Servidor e Internet.
Cristhian.Herrera@gmail.com / cherrera@kruger.com.ec

Descargar este documento en formato PDF hibernateVSEJB3.pdf

Fecha de creación del tutorial: 2007-08-16

Artículos sobre JEE

Comparativa entre Hibernate y EJB3 en la Capa de Persistencia

El presente documento pretende dar algunas luces a la comparativa entre la opción de usar Hibernate y/ó EJB3 para la capa de persistencia[1] en los desarrollos de aplicaciones empresariales bajo un entorno JEE. Las dos primeras secciones son un recordatorio de que es Hibernate y que es EJB respectivamente, luego se orienta la situación actual y finalmente se presenta una tabla que resume en forma de comparación las características de Hibernate versus las características de EJB3 para la capa de persistencia.

Hibernate[2]

Hibernate es una herramienta ORM completa que ha conseguido en un tiempo record una excelente reputación en la comunidad de desarrollo posicionándose claramente como el producto OpenSource[3] líder en este campo gracias a sus prestaciones, buena documentación y estabilidad. Es valorado por muchos incluso como solución superior a productos comerciales dentro de su enfoque, siendo una muestra clara de su reputación y soporte la reciente integración dentro del grupo JBOSS[4] que seguramente generará iniciativas muy interesantes para el uso de Hibernate dentro de este servidor de aplicaciones.

Hibernate parte de una filosofía de mapear objetos Java "normales", también conocidos en la comunidad como "POJO's[5]" (Plain Old Java Objects). No contempla la posibilidad de automatizar directamente la persistencia de Entity Beans tipo BMP (es decir, generar automáticamente este tipo de objetos), aunque aún así es posible combinar Hibernate con este tipo de beans utilizando los patrones para la delegación de persistencia en POJO's.

Una característica de la filosofía de diseño de Hibernate ha de ser destacada especialmente, dada su gran importancia: puede utilizar los objetos Java definidos por el usuario tal cual, es decir, no utiliza técnicas como generación de código a partir de descriptores del modelo de datos o manipulación de bytecodes en tiempo de compilación (técnica conocida por su amplio uso en JDO), ni obliga a implementar interfaces determinados, ni heredar de una superclase. Utiliza en vez de ello el mecanismo de reflexión de Java. Hibernate trata con la reflexión de Java y el aumento de clases en tiempo de ejecución utilizando una librería de generación de código Java muy poderosa y de alto rendimiento llamada CGLIB. CGLIB se utiliza para extender clases Java e implementar interfaces Java en tiempo de ejecución.

EJB[6]

EJB (Enterprise Java Beans) es un marco de trabajo para el desarrollo de aplicaciones empresariales en Java, el modelo de EJB es propuesto a través de un JSR (Java Specification Requerimient) es decir es una especificación que debe ser implementada por cualquier proveedor de servidor de aplicaciones que desee ser compatible con la misma.

La especificación de EJB define una arquitectura para el desarrollo y despliegue de aplicaciones basadas en objetos distribuidos transaccionales, software de componentes del lado del servidor. Las organizaciones pueden construir sus propios componentes o comprarlos a vendedores de terceras partes. Estos componentes del lado del servidor, llamados Beans Enterprise, son objetos distribuidos que están localizados en contenedores de JavaBean Enterprise y proporcionan servicios remotos para clientes distribuidos a lo largo de la red.

La especificación de Enterprise Java Beans define una arquitectura para un sistema transaccional de objetos distribuidos basado en componentes. La especificación manda un modelo de programación; es decir, convenciones o protocolos y un conjunto de clases e interfaces que crean el API EJB. Este modelo de programación proporciona a los desarrolladores de Beans y a los vendedores de servidores EJB un conjunto de contratos que definen una plataforma de desarrollo común. El objetivo de estos contratos es asegurar la portabilidad a través de los vendedores y el soporte de un rico conjunto de funcionalidades.

Los EJB representan la base del modelo de negocio, definido en términos de un conjunto de componentes colaborativos. Los EJB se utilizan para forzar las reglas de negocio, para encapsular la lógica de negocio y para acoplar el modelo de negocio con la aplicación empresarial.

Para crear un componente EJB del lado del servidor, un desarrollador de bean enterprise proporciona dos interfaces que definen los métodos de negocio del bean, además de la implementación real de la clase bean. Entonces el cliente usa un interfaz público del bean para crear, manipular, y eliminar beans del servidor EJB. La clase de implementación, será llamada clase del bean, es ejemplarizada en tiempo de ejecución y se convierte en un objeto distribuido.

Para crear un EJB[7], necesita escribir tres elementos de código java:

- Una interfaz Home, utilizada para gestionar el ciclo de vida de los EJB (contiene los métodos create, locate y remove)
- Una interfaz Remote, utilizada para definir los métodos de negocio.
- Implementar Enterprise Beans.

Además hay que escribir un documento XML llamado descriptor[8] de la distribución que especifica los detalles de cómo deben ser gestionados los EJB; este fichero contiene información sobre la configuración, administración y gestión de recursos.

Los beans enterprise viven en un contenedor EJB y son accedidos por aplicaciones clientes a través de la red usando sus interfaces remoto y local (home). Estos interfaces exponen las capacidades del bean y proporcionan todos los métodos necesarios para crear, actualizar, borrar e interactuar con el bean.

EJB3[9]

EJB 3.0 (la versión actual de EJB) incluye

- El nuevo "API de Persistencia de Java"
- Un modelo más sencillo para la implementación de fachadas
- Por compatibilidad, incluye las APIs del modelo anterior (EJB 2.1)

Nuevo API de Persistencia Java

- Inspirado en Hibernate, TopLink y JDO

- Mapeador objeto/relacional al estilo Hibernate/TopLink
    - En consecuencia, no oculta que el tipo de BD sea relacional
    - Actualmente Hibernate y TopLink implementan el API de Persistencia de Java
    - Se puede usar en entornos J2SE (fuera de un contenedor J2EE) y sin el resto de componentes de EJB
- Consta de
    - Anotaciones
    - El API propiamente dicha
    - EJB-QL
- Anotaciones
    - El desarrollador puede anotar las clases persistentes (llamadas "entidades") para que la implementación del API de Persistencia sepa cómo mapear las instancias a la BD (e.g. indica el nombre de la tabla, los nombres de las columnas a las que mapear los atributos, etc.)
- El API propiamente dicha
    - javax.persistence
    - Objetos principales
        - EntityManager: permite crear, encontrar por clave primera, y eliminar objetos persistentes, y crea objetos Query
        - Query: permite lanzar consultas en EJB-QL
        - Internamente las implementaciones usan el API de JDBC (transparentemente al programador)
- EJB-QL (EJB Query Language)
    - Lenguaje de consultas de búsqueda y borrados/actualizaciones en masa
    - La implementación traduce al SQL de la BD que se esté usando

Implementación de Fachadas

- Pueden ser locales o remotas
- Ocultan las APIs de transacciones y seguridad al desarrollador
- Las versiones anteriores de EJB también proporcionaban lo anterior, pero con un modelo de programación más complejo
- Tipos de fachadas
    - Session Beans
    - Message-Driven Beans
- Session Beans
- Fachadas "normales" (operaciones síncronas)
- Hay que definir una interfaz y una implementación
- Variantes
    - Stateless Session Beans (SLSB)
    - Stateful Session Beans (SFSB)

Situación actual

El modelo de programación propuesto por la versión 2.1 (y anteriores) de EJB conllevaba una serie de inconvenientes que limitaron mucho el uso y aceptación de esta especificación y motivó la aparición de muchas soluciones Open Source que suplían las carencias y dificultades que presentaba EJB 2.1. En este ámbito, las soluciones Open Source que más han marcado el desarrollo empresarial dentro de la plataforma Java han sido Hibernate, y Spring Framework, y en la nueva versión de Java Enterprise Edition se han incorporado muchas de las características de estos frameworks para procurar a los desarrolladores una plataforma de desarrollo bastante más sencilla que su predecesora versión 1.4.

Hibernate es, como ya se mencionó, un framework[10] para el manejo de persistencia. Hasta hace poco era un estándar de facto de la industria pues nació fuera de las especificaciones de SUN pero su facilidad de uso y su gran desempeño hicieron que tuviera una acogida realmente excepcional llegando a ser el framework de persistencia de mayor uso en el mundo java. La principal particularidad de Hibernate es que permite la persistencia de objetos java simples conocidos como POJO's

La especificación de EJB3 reconoce las ventajas de Hibernate y hace eco de mantener un mapeo objeto/relacional transparente. En esta especificación se estandarizaron los APIs de persistencia JPA y JDO (Java Persistence API y Java Data Objects), de esta forma la nueva especificación se convierte en una especie de clon de las mejores características de Hibernate (no en vano el creador de Hibernate Gavin King ha participado también en la especificación de JPA), a las que se sumaron lo mejor y más rescatable de JPA y JDO teniendo como resultado que la implementación de la capa de persistencia de EJB3 es drásticamente diferente a su par en la especificación EJB 2.1. Y desde el lado del mundo Open Source el reconocimiento fue mutuo porque Hibernate 3.2 en su Entity Manager implementa el ciclo de vida y las reglas definidas por la especificación de EJB3 convirtiéndose en una implementación más de JPA. Además aparecen en Hibernate el soporte a las anotaciones de JPA de tal forma que en la parte de persistencia puede pasarse sin mayores problemas de Hibernate a EJB3, ahora el cambio puede ser prácticamente transparente.

Como dato adicional cabe resaltar que la versión actual de Hibernate 3.2 ha sido certificada mediante el Sun Technology Compatibility Kit (TCK) como compatible con la especificación Java Persistence API (JPA).

Tabla de comparación de Hibernate y EJB3 en la capa de persistencia

	Hibernate	**EJB3**
Estándar	Hibernate 3.2 es una implementación del estándar.	Estándar de SUN Cada proveedor debe proporcionar una implementación.
Portabilidad	Se depende de que JBOSS es el único proveedor y se necesita siempre llevar los jars[11] de Hibernate en los proyectos ó en el classpath[12] para que nuestra aplicación funcione cuando se trabaja en otro servidor de aplicaciones.	Cualquier desarrollo con EJB 3 debe ejecutarse en un servidor de aplicaciones certificado para EJB3 sin realizar cambios en la misma.
Curva de aprendizaje	Muy corta, fácil de aprender. Aunque realmente son pocos los profesionales que realmente usan Hibernate en todo su potencial.	En EJB3 es similar a Hibernate Se debe resaltar que EJB3 tiene un ámbito mucho más amplio que sólo la capa de persistencia. En versiones anteriores de EJB la comparación favorecía a Hibernate.
Dependencia de otros proyectos	Hibernate se encarga únicamente de la capa de persistencia, debe soportarse con otros proyectos Open Source como Spring para cubrir otros aspectos del desarrollo de una aplicación empresarial JEE.	Una implementación de la especificación EJB3 cubre todos los aspectos del desarrollo de una aplicación empresarial JEE. EJB3 = Hibernate + Spring + ...
Aceptación de la industria	Completamente aceptado por la industria. Existe un gran número de desarrolladores JAVA en el mundo entero que apoyan a Hibernate.	Existe un fuerte movimiento de los grandes competidores de la industria para adaptar EJB3. Se pueden mencionar a SUN, JBOSS, Oracle, IBM como los más fuertes impulsadores del uso de EJB3. Cada uno de ellos está proveyendo en sus servidores de aplicaciones la respectiva implementación de soporte para EJB3. Al momento de escribir estas líneas, únicamente conozco que SUN App Server, JBOSS, Oracle Aplication Server y JONAS están ya oficialmente certificados para EJB3. El número de desarrolladores que están empleando EJB3 se está incrementando con mucha rapidez, pues agrupa a una buena parte de quienes usaban JPA y JDO y también a un gran número de jóvenes programadores que empezaron en java con la versión JEE 5.0 Un punto adicional es que JBOSS incentiva el uso de EJB3 aún siendo JBOSS el propietario de Hibernate.
Comparativas de rendimiento	No necesariamente el mejor frente a otros competidores como JPOX (JDO), IBatis, Castor, TopLink o JDBC. Sin embargo los resultados con	Las versiones predecesoras no gozaban de buenas referencias dejando a EJB en mala posición frente a las demás alternativas. Aún no existen suficientes comparativas y

	Hibernate siempre estaban entre los mejores en cualquier comparativa. Frente a versiones anteriores de EJB, era indudablemente mejor usar Hibernate	resultados frente a las otras opciones, sin embargo debe reconocerse que EJB3 nace con JEE 5.0 el mismo que viene optimizado y es un 60% más rápido y eficiente que JDK 1.4 Al hablar de Hibernate 3.2 y EJB 3 en la parte de persistencia no debería haber razón de una comparativa pues el primero es una implementación del segundo.
Generación de código	Mediante plug-ins para Eclipse y otros IDES. Generación con XDoclet	JDeveloper proporciona soporte para generación de código de EJB3. Mediante plug-ins para Eclipse y otros IDES. Generación con XDoclet
Archivos de Mapeo (XML)	Opcional en la versión actual Obligatorio en versiones anteriores	Opcional en EJB3 Obligatorio en versiones anteriores
Simplicidad	Sencillamente siempre fue simple.	EJB3 reduce el número de clases y número de interfaces que los programadores deben programar ó implementar. Reduce el número de artefactos (archivos de configuración, descriptores de despliegue) que se requieren para que la aplicación funcione. Evita los antipatrones.
Mapeo Objeto/Relacional	Soporta el mapeo de relaciones complejas de objetos Soporte de claves simples y compuestas Soporte de claves simples y compuestas Mapeo de superclases y relaciones de herencia de clases Uso de Lazy / Eager para carga liviana y pesada de objetos relacionados Actualizaciones y eliminación de datos en cascada	Soporta el mapeo de relaciones complejas de objetos Soporte de claves simples y compuestas Mapeo directo de propiedades y columnas Mapeo de superclases y relaciones de herencia de clases Uso de Lazy / Eager para carga liviana y pesada de objetos relacionados Actualizaciones y eliminación de datos en cascada
Configuración de acceso a base de datos	Se tiene un Hibernate Dialect para cada motor de base de datos. Debe especificar el dialecto de la base de datos (Oracle, DB2, etc) Puede usar su propio archivo de configuración o hacer uso de la configuración de otro framework como Spring.	Requiere de la configuración de un datasource con un nombre JNDI (Java Naming and Directory Interface) para la conexión a la base de datos. En el datasource se debe especificar el driver de base de datos que se empleará para la conexión así como otros datos como usuarios y claves y el uso o no de un pool de conexiones.

	La configuración de acceso a datos también puede hacerse mediante JNDI para nombrar un datasource.	
Trabajar fuera del contenedor de aplicaciones (stand alone)	Hibernate puede trabajar sin un servidor de aplicaciones, tanto en desarrollos de aplicaciones livianas, como en desarrollos de aplicaciones complejas.	JPA y la persistencia en EJB3 también se puede emplear sin un servidor de aplicaciones. Toda la especificación de EJB3 puede trabajar sin un servidor de aplicaciones gracias a desarrollos como Embedded JBoss
POJOS	Soportados	Soportados en capa de persistencia (Entity Beans) Soportados en capa de negocio (Session Beans y Message Driver Beans) En resumen se tiene un amplio soporte a los POJO's en todas las capas del desarrollo de una aplicación JEE. Los POJOS también pueden ser testeados fuera del contenedor de aplicaciones.
Uso de las Anotaciones	Soportada anotaciones Anotaciones JEE 5.0 y superiores Anotaciones JPA Anotaciones Hibernate	Soportadas anotaciones Anotaciones JEE 5.0 y superiores Anotaciones JPA Anotaciones EJB
HQL	Soportado Consultas simples y complejas Joins (inners joins y outer joins) Característica de "Criteria" y composición de objetos Proyecciones y Agrupaciones	
EJBQL		Soportado Consultas simples y complejas Joins (inners joins y outer joins) Subconsultas en las cláusulas (WHERE y HAVING) Funciones en consultas (de cadenas, aritméticas, agregadas) Característica de "Criteria" y composición de objetos

		Proyecciones y Agrupaciones
Querys nativos	Soportados	Soportados
Objetos distribuidos		Soportados
Reflexión (Reflection)	Soportado	Soportado
Inyección de dependencias	Soportados mediante extensiones de Hibernate y mediante el uso de otros frameworks como Spring	Soportados nativamente como parte de la especificación.
Manejo de Transacciones	Soportadas mediante extensiones de Hibernate y mediante el uso de otros frameworks como Spring	Soportadas nativamente como parte de la especificación. Por defecto los métodos de un EJB de sesión son requeridos, es decir son transaccionados
Manejo de seguridad	Soportada mediante extensiones de Hibernate y mediante el uso de otros frameworks como Spring.	Soportada nativamente como parte de la especificación.

Conclusiones

- EJB3 no es Hibernate, el primero es un marco de trabajo para el desarrollo de aplicaciones empresariales en Java, mientras que el segundo es un framework únicamente para la capa de persistencia de datos.
- En este momento una comparación de EJB3 y Hibernate únicamente en la parte de persistencia resulta prácticamente un empate, quizás Hibernate resulta un poco ganador por ser un producto más maduro y con mucho más uso en la actualidad.
- Cuando Hibernate implementa EJB3, en Hibernate 3.2 la comparación pierde peso pues Hibernate 3.2 se convierte en una implementación más de JPA (EJB3).
- Si hablamos de curva de aprendizaje y rapidez de desarrollo, personalmente veo a EJB3 como un claro ganador, la curva es corta, las ventajas evidentes y los tiempos de desarrollo, líneas de código y archivos que se necesitan para que la aplicación funcione se reducen considerablemente con EJB3.
- Se necesitan más comparativas entre EJB3 y otros frameworks como por ejemplo una comparativa entre Spring + Hibernate para cubrir todo el espectro de las aplicaciones JEE, sobre todo en lo referente al rendimiento de los EJB de sesión ya sean con estado (stateful) o sin estado (stateless) pues al menos en teoría la instanciación de los EJB's en sesión va a seguir siendo más pesada que una aplicación liviana desarrollada con Spring, sin embargo debe pensarse en otros aspectos como la seguridad y el manejo de la transaccionabilidad para poder finalmente indicar que es mejor para un desarrollo de una aplicación profesional.
- Como acotación a la conclusión anterior el modelo de POJO's en el que se basa EJB3 hace que sea posible tener aplicaciones EJB3 livianas (ligthweigth applications) que pueden competir en rendimiento (tiempo de respuesta, carga, optimización del ciclo del garbage collector) con las aplicaciones que se desarrollan con frameworks como Spring, aunque personalmente aún no conozco ni he encontrado comparativas que demuestren que esta afirmación sea completamente acertada.
- Una ventaja adicional de EJB3 es que provee todo el entorno de trabajo para el desarrollo de aplicaciones empresariales, evitando la compleja tarea de integrar diferentes frameworks para cada capa distinta de la aplicación.
- Me atrevo a hacer un pronóstico, no pasará mucho tiempo antes de que Spring y otros frameworks similares se reconviertan internamente para cumplir con la especificación.

Bibliografía

1. Persistencia de los Objetos Java utilizando Base de Datos Relacionales, Galeano Cristian, Goette Emmanuel
2. BIPunto Weblog, Hibernate JPA
3. Sizing Up Open Source Java Persistence, Jim White, http://www.devx.com/
4. Comentarios de Alejandro Pérez, Autentia
5. Apuntes de Integración de Sistemas, Fernando Bellas
6. JBOSS Books, What is a Lightweight Web Application, http://docs.jboss.com/books/lightweight/pt01.html

1 En el modelo de desarrollo de aplicaciones en capas, la capa de persistencia hace referencia a la parte de la lógica que se ocupa de interactuar con un motor de base de datos y de garantizar que la información pueda almacenarse y recuperarse de forma transparente y óptima para el usuario final.

2 Sección tomada de primera referencia bibliográfica.

3 Open Source (Código Abierto) corriente que avala la libre compartición y uso del código fuente de las aplicaciones, estableciendo un modelo de negocio de enorme éxito, pues el rédito no está en vender el software sino en darle soporte, en compartir la experiencia de los desarrolladores y en la mejora constante de los desarrollos.

4 Grupo Open Source que popularizó un servidor de aplicaciones con su mismo nombre JBOSS, la empresa acaba de ser adsorbida por Red Hat y el grupo se ha hecho cargo de Hibernate adaptándolo como su producto ORM para la persistencia de datos.

5 Un POJO es una clase JAVA, en su forma más pura, y básicamente consta de un conjunto de atributos y métodos de obtención y establecimiento (métodos getters y setters) para cada uno de los atributos definidos para la clase.

6 Sección tomada de primera referencia bibliográfica

7 En la especificación EJB3 las interfaces local y remota son opcionales, puede declararse la una o la otra ó ambas ya no se obliga a tener las dos, y se reduce y simplifica el código con las anotaciones.

8 En la especificación EJB 3 el descriptor de la aplicación se puede reemplazar con anotaciones

9 Sección tomada de la quinta referencia de la bibliografía.

10 Framework ó marco de trabajo, es básicamente la agrupación de un conjunto de utilidades y mejores prácticas para emplearlos como acelerador para otros desarrollos. Los frameworks se escriben en base a la aplicación explícita de ciertos patrones de diseño, como por ejemplo el modelo MVC y pretenden resolver ciertos problemas comunes y a la par brindarle al desarrollo un conjunto fuerte de utilidades que le permitan hacer su trabajo en forma más rápida y eficiente.

11 JAR es el acrónimo para Java Application Resource, un jar es archivo en el cual se distribuyen librerías (clases Java) para que puedan ser usadas por otras aplicaciones.

12 El classpath de una aplicación java es la ruta o directorio donde puede encontrar a las clases que necesita para poder funcionar.

Ing. Cristhian Kirs Herrera Basurto ***11***

Nombre Corto	Descripción
Hibernate 3.1, Colecciones, Fetch y Lazy	En este tutorial vamos a ver cómo se comportan ciertas relaciones, y cómo podemos optimizar las consultas a la base de datos con Hibernate
Hibernate 3 y los tipos de datos para cadenas largas	En este tutorial se contará la experiencia que hemos tenido a la hora de manejar los diferentes tipos de datos existentes para grandes cadenas de texto, tales como el tipo Clob de Oracle, o los tipos TEXT de MySQL y SQLServer, utilizando la última versión
Anotaciones en EJB 3.0	Este tuotiral nos va a enseñar algunas características del API de EJB 3.0 y las mejoras introducidas en la nueva version 3.0
Interceptando un EJB en JBoss	En este tutorial os vamos a enseñar la arquitectura de EJBs en JBoss y a como modificarla, insertando un interceptor propio dentro de la cadena de interceptores del Proxy Cliente.
Hibernate y Joins con la clase Criteria	En este tutorial vamos a ver como hacer "joins" entre entidades relacionadas, y las implicaciones que esto puede tener, usando Hibernate
Hibernate y el mapeo de la herencia	En este tercer tutorial de la saga vamos a ver en este tutorial como implementar las relaciones de herencia con las anotaciones de JPA
EJB 3.0, un ejemplo práctico con Maven y JBoss	Este tutorial presenta un ejemplo sencillo donde se verá como desarrollar EJBs de sesión y de entidad, inyección de dependencias, llamar a los EJBs desde una aplicación Web, definición de un DataSource, y como configurarlo y hacerlo funcionar en JBoss, y
Proyecto con JSF Java Server Faces Myfaces, Maven y Eclipse: Hibernate (segunda parte)	En este artículo se va a continuar con el desarrollo de la aplicación Myfaces JSF con Maven multimódulo que comenzamos en un tutorial anterior. Además también se tratará de la integración de Hibernate con las aplicaciones.
Hibernate Tools y la generación de código	En este tutorial vamos a ver como usar estas herramientas para hacer el esqueleto de una pequeña aplicación, de manera muy sencilla, generando código a partir de las tablas creadas en la base de datos.
Hibernate y las anotaciones de EJB 3.0	En este tutorial Alejandro Pérez nos muestra las ventajas que nos aporta Hibernate y las anotaciones de EJB 3.0