Ambientes de desarrollo modernos

Análisis comparativo de herramientas

Autor: Miguel López

Director: Javier Blanqué

Universidad Nacional de Luján
Int. Ruta 5 y 7
6700 Luján, Buenos Aires
República Argentina
Año 2006

Ambientes de desarrollo modernos
Análisis comparativo de herramientas

Miguel Ángel López
Universidad Nacional de Luján
Int. Ruta 5 y 7
6700 Luján, Buenos Aires
República Argentina
mlopez@omtw.com.ar

Resumen

El presente trabajo presenta un análisis comparativo entre 3 entornos de desarrollo modernos. El objetivo de este análisis es brindar nueva información acerca del uso de estas herramientas actuales y presentar un punto de partida para el análisis de nuevas herramientas o de mejoras que se incorporen en las existentes en el futuro. A su vez, se pretende brindar una proto-metodología para la toma de decisión respecto de qué entornos utilizar de acuerdo al proyecto de desarrollo elegido, equipo de trabajo o metodología.

Índice

MOTIVACIÓN Y CONTEXTO

El moderno mundo de la información y de las comunicaciones, con mercados globalizados y cada día más exigentes, obliga a las empresas a buscar nuevas ventajas competitivas. Por otro lado, los procesos que éstas manejan se han informatizando a través de los años y han ido aumentando su dependencia de los sistemas de información y de la tecnología. Actualmente, la velocidad de la innovación tecnológica es tan grande que supera, muchas veces, la capacidad de adaptación de las personas y de las Organizaciones. Pero las Organizaciones que sobresalen son aquellas que pueden adaptar sus procesos y negocios más velozmente de manera de responder a las demandas constantes del mercado.

En este contexto, las necesidades respecto de los desarrollos de software se han ido transformando para requerir cada vez más integración y mayor adaptabilidad, en menos tiempo. La integración, como parte importante en la globalización mundial de la información, exige nuevos recursos tecnológicos y la adaptabilidad no solo se traduce en mejores capacidades y funcionalidades, sino en mayor velocidad de respuesta frente a los cambios. También es importante, en estos desarrollos, la disponibilidad de los mismos a bajo costo, integración con las diversas herramientas del mercado, amplitud de lenguajes y plataformas, etc.

Así es que como se han desarrollado nuevas metodologías o conceptos en la ingeniería de software, en respuesta a otras que obstaculizaban la rápida producción de elementos tangibles por el usuario. Entre estos podemos mencionar a los siguientes conceptos: Agile Developments, eXtreme Programming (XP) y RAD (Rapid Application Development), entre otros. Donde no solo prevalece la velocidad del desarrollo sino la calidad, el trabajo en equipo y demás problemáticas inherentes a estas nuevas ópticas.

Particularmente las Organizaciones de desarrollo de software han tenido que adaptarse a estos cambios y no ha sido fácil el desafío. En nuestro país se ha dado en los últimos años un aumento en la tasa de empleos en el sector de IT. Tanto desde el ámbito privado como desde el gobierno han gestado diversas estrategias para captar estudiantes

de carreras relacionadas con Sistemas de información y tecnología Informática y de Comunicaciones o TIC y desarrollar el sector. Además ha habido un crecimiento en empresas de tipo Software Factory (fábricas de software), que han encontrado un crecimiento destacado a la vez que nuevos problemas a resolver, como la capacitación del personal, estandarización de procesos, nuevos requerimientos tecnológicos o rotación permanente de recursos humanos.

En este contexto es que tomamos el análisis de los nuevos ambientes de desarrollo, considerando que se hace necesario, para la construcción del software, un entorno que cumpla con las pautas señaladas permitiendo la estandarización de procesos, circuitos de calidad, integración de herramientas, facilidad de absorción por los programadores, control de versiones, trabajo colaborativo y demás características que acompañen el ciclo de vida de los sistemas de información desarrollados.

1 INTRODUCCIÓN

Este trabajo presenta un análisis de los distintos aspectos de las herramientas (IDEs) que posibilitan la construcción de aplicaciones en entornos modernos.

A partir de un prototipo de aplicación que cumpla con ciertos requisitos de desarrollo moderno, se integrará en las distintas herramientas para lograr realizar una tipificación de éstas y abordar los diversos tópicos cubiertos por las mismas, desde el enfoque de la necesidad de generar aplicaciones orientadas a resolver problemas de gestión de la información. Este análisis se basará principalmente en 3 IDEs considerados como más usados.

El propósito principal de este trabajo, además de brindar nueva información acerca del uso de estas modernas herramientas, es presentar un punto de partida para el análisis de nuevas o de mejoras que se incorporen en las existentes en el futuro. A su vez, con este trabajo se pretende brindar una proto-metodología para la toma de decisión respecto de qué entornos utilizar de acuerdo al proyecto de desarrollo elegido, equipo de trabajo o metodología.

El deseo u objetivo ideal es que este material pueda ser utilizado y principalmente actualizado por futuros estudiantes o profesionales, de acuerdo a los cambios de tendencias en cuanto al uso de herramientas de este tipo, avances en componentes tecnológicos y no tecnológicos que faciliten la gestión del desarrollo, o cambios importantes en el contexto en el cual los proyectos se desarrollan.

Descripción de la estructura del trabajo y sus capítulos

En el Capítulo 2 se dará una introducción a lo que llamamos para este trabajo “Entornos de Desarrollo Modernos”. Se pretenderá brindar una breve introducción a las tecnologías involucradas, las cuales serán tomadas en cuenta para la evaluación de las diferentes herramientas.

Por último, ... conclusiones

2 ENTORNOS DE DESARROLLO MODERNOS

Las herramientas estudiadas y analizadas fueron seleccionadas en base a su popularidad en el contexto de las organizaciones de desarrollo de software, que podrían denominarse “el estado del Arte” en el presente, que tengan alcance en ambientes modernos, principalmente en J2EE y lenguaje Java. Su estudio fue enfocado principalmente hacia el desarrollo de aplicaciones de gestión.

En los capítulos posteriores se realizará una descripción detallada de cada una de ellas y su integración con administradores de bases de datos, sistemas de control de versiones, descripción de sus facilidades, especialmente en lo que respecta a capacidades y ayudas a la modificación de los fuentes, localización de métodos en clases y debuggers a nivel fuente.

Se hace necesario definir qué alcance tiene un entorno de desarrollo moderno. Para ello estableceremos algunas condiciones que deberán cumplir las herramientas seleccionadas.

2.1 DESARROLLOS ÁGILES

Ver si es necesario y relevante introducir el concepto de desarrollo ágil.

Extreme programming

Scrum

Crystal family

Feature Driven Development

Adaptive Software Development

2.2 INTRODUCCIÓN A J2EE

J2EE son las siglas de Java 2 Enterprise Edition, que es la edición empresarial del paquete Java creado y distribuido por Sun Microsystems. Comprenden un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales.

La plataforma Java 2, Enterprise Edition (J2EE) es fruto de la colaboración de SUN con los líderes del sector del software empresarial (IBM, Apple, Bea Systems, Oracle, Inprise, Hewlett-Packard, Novell, etc.) para definir una plataforma robusta y flexible orientada a cubrir las necesidades empresariales en e-business y business-to-business **[Armstrong05]**.

J2EE define una arquitectura para el desarrollo complejo y distribuido de aplicaciones empresariales en lenguaje Java. Aunque fue anunciada en los mediados de los '90s, fue oficialmente liberada en 1999. Por lo tanto es una tecnología relativamente nueva, con lo cual su evolución ha generado y genera permanentes cambios, especialmente en el área de Enterprise JavaBeans (EJB) **[Ahmed01]**.

J2EE consiste en lo siguiente:

- Guías de diseño para el desarrollo de aplicaciones empresariales usando J2EE y Java.
- Una referencia de implementación
- Un conjunto de herramientas para el uso de terceras partes y para verificar productos.
- Interfaces de aplicación (APIs) para proveer un acceso genérico a los recursos empresariales e infraestructura.
- Tecnologías para simplificar el desarrollo empresarial en Java

Las organizaciones necesitan constantemente extender su alcance, reducir sus costos, y bajar sus tiempos de respuesta para proporcionar un fácil acceso a sus clientes, empleados y proveedores. Generalmente, las aplicaciones que proporcionan estos servicios deben combinar los sistemas de información de la organización existentes o heredados, con nuevas funciones que entregan servicios a un gran espectro de usuarios.

Las aplicaciones empresariales, y con ellos las basadas en arquitecturas J2EE, debieran cumplir los siguientes objetivos o cualidades **[Johnson02]**:

- **Robustez**, desarrollada con código de calidad, confiable y libre de errores. Por eso la plataforma debe ayudar a facilitar estas tareas.
- **Performante y escalable.** La performance debe cubrir las espectativas de los usuarios, mientras que la escalabilidad se hace particularmente relevante en ambientes de Internet donde es dificil predecir el número de usuarios y su naturaleza.
- **Tomar ventajas de los principios de la orientación a objetos.** El uso de patrones de diseños probados y funcionalidad como reutilización de código deben ser aprovechados.
- **Evitar la complejidad innecesaria.** Permitir el desarrollo de practicas de desarrollo ágiles con XP o Scrum **(REFERENCIAR BIBLIO).**
- **Ser mantenible y extensible.** El mantenimiento es la fase más costosa del ciclo de vida de un sistema. Por lo tanto debe minimizarse su esfuerzo. A su vez la extensibilidad de un sistema debe permitir el reacomodamiento a partir de las nuevas necesidades del negocio. **(BUSCAR GRAFICOS DE ENCUESTAS DEL CICLO DE VIDA).**
- **Poder ser entregada en tiempo.** Para esto es necesario tener buena productividad.
- **Facil de testear.**
- **Promover su reuso.**

Adicionalmente y dependiendo de los requerimientos del negocio es necesario contar con los objetivos de:

- **Soportar múltiples tipos de clientes.** Las aplicaciones pueden requerir el soporte en varios tipos de clientes como aplicaciones web, GUI usando swing, Applets y otro tipo de sistemas de ventanas.

- **Portabilidad.** Dependiendo del negocio es posible que se requiera que las aplicaciones se ejecuten independientemente de la bases de datos, como así en distintos Application Servers.

Algunas de las funcionalidades más importantes que brinda esta plataforma son:

- Acceso a base de datos (JDBC)
- Utilización de directorios distribuidos (JNDI)
- Acceso a métodos remotos (RMI/CORBA)
- Funciones de correo electrónico (JavaMail)
- Aplicaciones Web (JSP y Servlet)
- Uso de Beans, etc.

2.2.1 Arquitectura de J2EE

J2EE esta basado en una arquitectura del lado del servidor (Served-based). Este tipo de arquitectura concentra la mayoría de los procesos de la aplicación en el servidor o en una parte de este. Tiene dos ventajas críticas en comparación con los otros tipos, estos son:

- **Múltiples Clientes**: Una arquitectura basada en el servidor requiere una clara separación entre la capa cliente (interfaz) y la capa servidor, en la cual se realizan los procesos de la aplicación. Esto permite que una simple aplicación soporte simultáneamente clientes con distintos tipos de interfaces, incluyendo poderosas interfaces (gráficas) para equipos corporativos, interfaces multimedia interactivas para usuarios con conexiones de alta velocidad, interfaces eficientes basadas en texto para usuarios con conexiones de baja velocidad, etc.
- **Operaciones robustas**: Una arquitectura basada en el servidor soporta escalabilidad, confiabilidad, disponibilidad y recuperabilidad. Aplicaciones basadas en el servidor pueden ser divididas y distribuidas en múltiples

procesadores. Los componentes de la aplicación pueden ser replicados para dar soporte a caídas instantáneamente.

La plataforma J2EE consiste en un conjunto de servicios, programas interfaces de aplicación (APIs) de Java y protocolos necesarios para el desarrollo de aplicaciones multi-capas y basadas en la Web **[Gould00]**. La plataforma completa puede ser implementada en un solo sistema, o la plataforma de servicios puede ser distribuida a través de varios sistemas, pero todas las APIs especificadas deben ser incluidas en alguna parte del sistema completo. En la figura 3.2 se presenta una ilustración de la arquitectura.

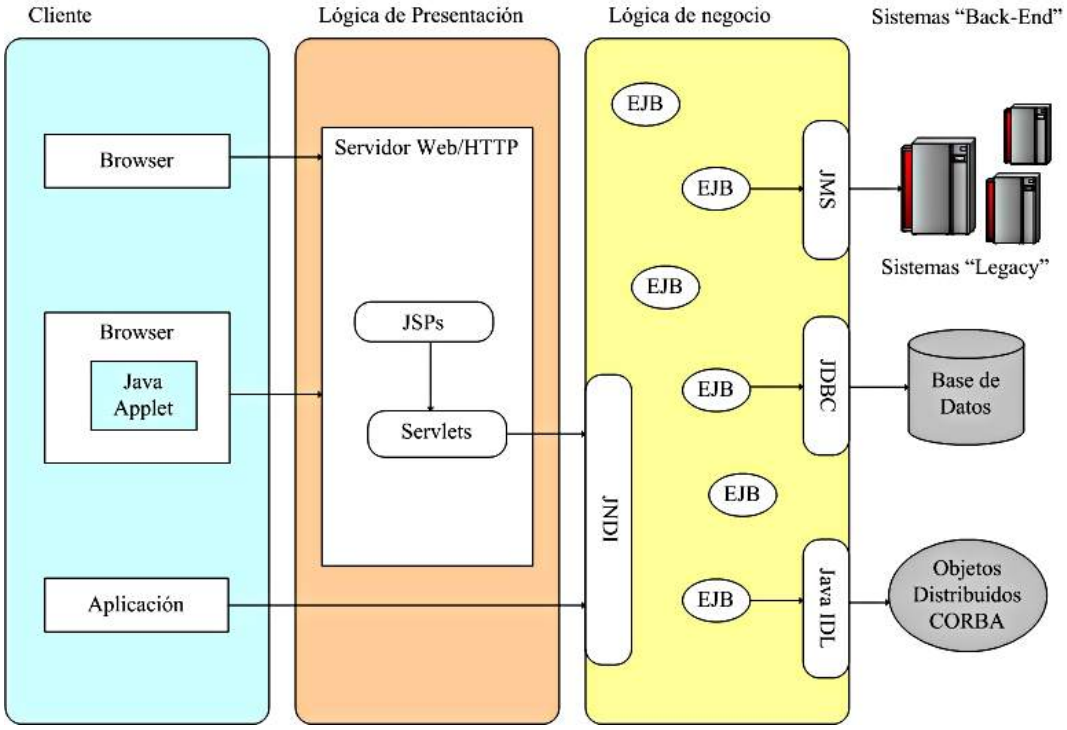


Figura 3.2 – Arquitectura multicapa J2EE

El ambiente de runtime de J2EE consta de las siguientes partes:

2.2.2 Componentes de la aplicación.

El modelo de programación de J2EE define cuatro tipos de componentes de la aplicación que un producto J2EE debe soportar:

1. **Clientes de la Aplicación**, son programas generalmente del tipo GUI , que se ejecutan sobre una computadora de escritorio. La aplicación cliente ofrece a un usuario una vista similar a la de las aplicaciones nativas, y tiene acceso a todos los medios de la J2EE middle-tier.

2. **Applet´s**, es un componente GUI que generalmente procesa un programa en un web-browser, pero pueden procesar una variedad de otras aplicaciones o dispositivos que soportan el modelo de programación del applet. Las Applets pueden ser usadas para proporcionar una poderosa interfaz de usuario para las aplicaciones J2EE.

3. **Páginas Servlets y JSP**, generalmente procesan un programa en un servidor Web y responden a las peticiones HTTP de los clientes Web. Las páginas Servlets y JSP pueden ser utilizadas para que generen páginas HTML (que son aplicaciones de interfaz de usuario más limitadas que los Applets). Pueden también ser usadas para generar XML u otro formato de datos que es consumido por otros componentes de la aplicación. Servlets, y páginas creadas con la tecnología Java Server Pages, se refieren conjuntamente a menudo en ésta especificación como "Componentes Web". Las aplicaciones Web están compuestas de Componentes Web y otros datos tal como las páginas HTML, JavaScript, Flash, etc.

4. **Componentes Enterprise JavaBeans (EJB)**, procesan en un ambiente controlado las transacciones soportadas. Los Enterprise beans generalmente contienen la lógica del negocio por una aplicación J2EE.

Estos componentes de la aplicación se pueden dividir en tres categorías:

1. Componentes que se despliegan, manejan, y se ejecutan sobre un servidor J2EE. Estos componentes incluyen JavaServer Pages, Servlets, y Enterprise JavaBeans.

2. Componentes que se despliegan y manejan en un servidor J2EE, pero está cargado y se ejecuta en una máquina cliente. Estos componentes incluyen páginas HTML y applets incluidas en las páginas HTML.

3. Componentes cuyo despliegue y manejo no se definió completamente en esta especificación. Las aplicaciones de los clientes caen en esta categoría. Futuras versiones de esta especificación pueden definir completamente el manejo de las aplicaciones del cliente.

2.2.3 APIs y otras especificaciones

Hay varias APIs y especificaciones dentro de J2EE. Aquí se presentan las más populares o usadas.

JDBC, es una API enfocada en facilitar la conexión con las bases de datos. Simplifica el acceso a bases de datos relacionales y consiste de una interface genérica, que luego es implementada por cada proveedor de la herramienta (Oracle, MySQL, etc.).

JNDI (Java Naming and Directory Interface) trabaja en conjunto con otras tecnologías para organizar y localizar componentes en un ambiente de computación distribuido **[SunJNDI]**. Maneja sistemas de nombrado, tales como los sistemas de archivos y otros servicios, en una manera de acceso uniforme y genérica.

JMS (Java Messaging Service) es la solución presentada por Sun para los sistemas de mensaje. JMS se encarga de coordinar la sincronía de los mensajes y comunicación entre dos partes, que alternativamente puedan encontrarse en lugares remotos.

"JMS se sitúa como middleware en medio de la comunicación de dos aplicaciones. En entornos cliente servidor, cuando la aplicación A quiere comunicarse con la Aplicación B, necesita saber donde esta B (su IP por ejemplo) y que B esté escuchando en ese momento. Cuando se usa JMS (o cualquier otro sistema de mensajes), la aplicación A envía un mensaje, el sistema de mensajes lo recibe y se lo envía a B cuando se conecte al servicio. De esta manera se consigue una comunicación

asíncrona entre A y B, es decir no hace falta que B esté presente en el momento del envío del mensaje, y no por ello va a dejar de recibirlo" **[Galdo06]**.

JMS puede ser usado directamente en una aplicación corporativa o vía un tipo de EJB conocido como Message-driven bean. Los Message-driven beans aparecen a partir de la especificación 1.3 de J2EE **[Amhed01]**.

RMI (Remote Method Invocation), permite el acceso a los componentes en entornos distribuidos, invocando objetos remotos con métodos Java **[SunRMI].**

Java IDL (Java Interface Definition Language), provee soporte para la interoperabilidad para los estándares de la industria CORBA (Common Object Request Broker Architecture) e IIOP (Internet Inter-Orb Protocol). **IIOP (Internet Inter-ORB Protocol)** es un protocolo desarrollado por el OMG (Object Managment Group) para implementar soluciones CORBA sobre Internet. IIOP permite a los browers servers intercambiar objetos a través de la red, a diferencia de HTTP que solo soporta la transmisión de texto **[SunRMI]**.

JTA (Java Transaction API), es usado para el manejo de las transacciones e incluye una serie de operaciones que aseguran la integridad y consistencia de las mismas. Provee una API de alto nivel para el manejo de las transacciones.

JTS (Java Transaction Service) es un servicio manejador de transacciones que soporta JTA y hace uso de IIOP para comunicarse entre instancias remotas del servicio.

JavaMail provee una API para facilitar la interacción con los sistemas de mensajes en una forma independiente de los proveedores de las herramientas. Consiste en un conjunto de clases abstractas que modelan un sistema de e-mail basado en Java.

JACC (Java Authorization Contract for Containers) es una especificación que define una nueva clase para el modelo de autorización de la plataforma Java 2. Define aspectos de acceso y permisos en los containers J2EE, como los relacionados con la definición de roles, permisos principales relacionados a los roles, etc.

2.2.4 Containers

Un container es un software que se ejecuta en el servidor y proporciona el soporte para los componentes de la aplicación, proporcionándole una vista de a las subyacentes APIs y un entorno de ejecución para los mismos. Interponer un container entre el componente de la aplicación y el servicio J2EE permite a los containers inyectar transparentemente servicios definidos por los componentes, tal como el manejo de la transacción, chequeos de seguridad, logging y otros.

Existen varios tipos de containers y cada componente se ejecuta en uno determinado. Por ejemplo los EJBs se ejecutan en el EJB Container y los servlets en el Web Container. Depending on the container type, it may also provide access to some or all of the J2EE APIs.

Puede clasificarse de la siguiente manera:

• Application container: Hostea aplicaciones Java stand-alone

• Applet container: Provee un entorno de ejecución para los applets.

• Web container: Hostea componentes Web, como los servlets y las páginas JSP.

• Enterprise container: Hostea componentes EJB

2.2.5 Despliegue de una aplicación en el Servidor

Explicar el despliegue de una aplicación y los correspondientes archivos (JAR, WAR y EAR).

JAR is the "normal" Java Application archive, but in this context it usually contains EJBs instead of code libraries or runnable (e.g. from outside an application container) applications.

WAR is an Web Aapplication archive and contains JSPs, "normal" HTTP served files (HTML, images, etc.), servlets, tag libraries, and such.

EAR is an Enterprise Application archive and may contain ejb JAR files, WAR files, and RAR (connector) files. They may also contain third-party libraries - but you have to know how to manipulate the Java extension facilities (e.g. MANIFEST.MF Class-Path directive) to make that work well.

2.3 JAVA

Java es un lenguaje de programación orientado a objetos desarrollado por James Gosling y sus compañeros de Sun Microsystems al inicio de la década de 1990. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, Java es compilado en un bytecode que es ejecutado (usando normalmente un compilador JIT) por una máquina virtual Java **[WikiJAVA]**.

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el Lenguaje de programación Java y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución, y un conjunto de librerías estándar que ofrecen funcionalidad común **[J2EEOverview].**

La plataforma es así llamada "Plataforma Java" e incluye **[Ahmed01]**:

- Plataforma Java, Edición Estándar (Java Platform, Standard Edition), o Java SE (antes J2SE)
- Plataforma Java, Edición Empresa (Java Platform, Enterprise Edition), o Java EE (antes J2EE)
- Plataforma Java, Edición Micro (Java Platform, Micro Edition), o Java ME (antes J2ME)

Un ejemplo de relación entre las distintas plataformas y las tecnologías, estructuras típicas involucradas puede ser observado en la figura 3.3.

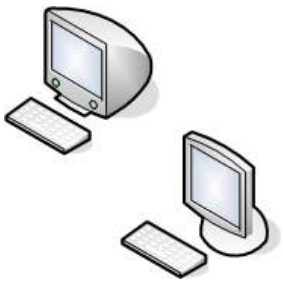
			
Tipo de Aplicaciones	• Misión crítica • Applicaciones corporativas • Conectividad Web	• Aplicaciones centradas en redes locales	• Aplicaciones con recursos o memoria restringida
Tecnología	• Enterprise Java Beans • Servlets • JSPs	• Applets • JavaBeans	• KDM • CDC • CLDC
Estructura típica	• Componentes • App Servers • VM • SO (Sistema operativo) • Hardware	• Aplicación Java • VM • SO • Hardware	• Aplicación Java • VM Especializada • SO • Dispositivo de Hardware
Nombre de la Plataforma	Java 2, Enterprise Edition	Java 2, Standard Edition	Java 2, Micro Edition

Figura 3.3 – Cuadro de tipos de aplicaciones por Plataforma

El modelo de desarrollo de J2EE Enterprise tambien considera una clara división entre el desarrolo, el despligue y la ejecución de los sistemas. Es por eso, que es posible separar las tareas de desarrollo, del desarrollo de componentes para la base de datos actual o configuraciones específicas del servidor. Y así asignar distintos roles de trabajo.

J2EE soporta independencia de hardware y sistemas operativos a través de servicios que puedan accederse a través de APIs de Java. Esto hace que los sistemas basados en arquitecturas J2EE puedan ser fácilmente portador hacia diferentes plataformas.

Uno de los grandes beneficios de J2EE es la posibilidad de desarrollar software basado en componentes, el cual brinda las siguientes ventajas frente a métodos tradicionales:

• **Alta productividad**: Pocos desarrolladores pueden obtener más utilizando componentes pretesteados que modificar una aplicación anterior.

• **Rápido desarrollo**: Tomando componentes existentes se puede crear rapidamente nuevas aplicaciones.

• **Fácil mantenimiento**: El mantenimiento de componentes individuales es más fácil y requiere menor costo.

J2EE facilita el desarrollo con componentes de varias maneras:

• La utilización de Java hace que el desarrollo pueda ser utilizado en varios sistemas operativos y con diferente hardware.

• La separación de aspectos de desarrollo de los aspectos de armado y ensamble de la aplicación para su despliegue.

• La posibilidad de integración y acceso a otros productos a través de APIs en una forma uniforme.

• La posibilidad de optimizar o desarrollar componentes especializados.

2.4 BASES DE DATOS

Una aplicación J2EE debe incluir una base de datos para el almacenamiento persistente de los datos de la empresa. Las aplicaciones y sistemas acceden a la base de datos utilizando el API JDBC. Otros tipos de bases de datos persistentes son permitidos, siempre que soporten el modelo relacional.

Las bases de datos que se han elegido para este trabajo son:

- Oracle Database

  Oracle Corporation. Oracle es una empresa que se dedicada a comercializar tecnología de bases de datos relacionales, aunque también está impulsando otros productos como el uso de Linux y Grid Computing, la Arquitectura Orientada a Servicios (SOA), Servicios Web, XML, etc.

- Microsoft SQL Server 2005

  Microsoft SQL Server 2005 es una plataforma global de base de datos que ofrece administración de datos empresariales con herramientas integradas de inteligencia empresarial (BI).

- MySQL

  MySQL es un sistema de gestión de base de datos, multihilo y multiusuario. La edición comunitaria de MySQL 5.0 es una versión de descarga gratis y de código abierto (Open Source). Es apoyada por una comunidad activa de desarrolladores.

2.5 MODELO MVC

MVC o Modelo Vista Controlador es una arquitectura o patrón de diseño que provee una forma de dividir la funcionalidad envuelta en el mantenimiento y la presentación de los datos (ver Gráfico). Fue desarrollado originalmente para mapear las tradicionales tareas de entrada, procesamiento y salida del modelo de interacción con el usuario. Sin embargo, es útil para mapear estos conceptos en el dominio de las de las aplicaciones empresariales basadas en interfaces web y multi-capas **[Cade02]**.

Ver gráfico. El modelo MVC provee un desarrollo distribuido para aplicar con J2EE

En la arquitectura MVC, el ***modelo*** representa los datos de la aplicación y las reglas de negocio que coordinan el acceso y modificación de los mismos. El modelo mantiene el estado de persistencia del negocio, mientras que el ***controlador*** provee la habilidad de acceder a las funcionalidades de la aplicación encapsuladas por el modelo.

El componente ***vista*** representa el contenido de una parte particular del modelo. Accede a los datos del modelo y especifica como los mismos son presentados. Cuando el modelo cambia, es responsabilidad de la vista mantener la consistencia en su presentación. La vista transmite las acciones del usuario al ***controlador***.

El ***controlador*** define el comportamiento de la aplicación, interpreta las acciones del usuario y los mapea a los procesos que debe ejecutar el modelo. En una aplicación con un cliente web, esas acciones serán clicks en botones y selecciones de menú. Esas acciones ejecutadas por el modelo incluirán la activación de procesos de negocios y cambios de estado del modelo. Luego, de acuerdo a la salida del procesamiento del modelo, el controlador seleccionará una vista para representar la salida como parte de la respuesta al requerimiento del usuario.

2.6 APPLICATION SERVERS

El término Application Server se refiere muchas veces a un Servidor de Aplicaciones J2EE. Estos servidores sirven o dan servicio principalmente a Aplicaciones Webs que ejecuten código Java, como suelen ser las páginas JSP o los Servlets. Las páginas JSP o programas Java que se ejecutan en el servidor (Ej. Servlets) en un container de Java equivalente a Scritps CGI.

Los Application Servers generalmente incluyen productos de middleware (capa de arquitectura media) que le permite a las aplicaciones intercomunicarse con varias calidades de servicios (fiabilidad, seguridad, etc.), proveen APIs a los programadores para facilitarles tareas como logging, manejo de transacciones, persistencia, clustering, caching, etc., y simplifican la interacción con los distintos sistemas operativos y bases de datos.

Existen varios Application Servers en el mercado. Entre los que tienen licencia comercial podemos nombrar a: WebSphere (IBM), Sun Java System Application Server and WebLogic (BEA), JBoss (Red Hat) y Oracle Application Server 10g (Oracle Corporation). También existen algunos open-source como: GlassFish (Sun), JOnAS (ObjectWeb) o Tomcat (Apache). Todos estos tienen certificaciones de compatibilidad con J2EE y el lenguaje usado es Java.

Los Application Servers que se han elegido para este trabajo, para pruebas de integración con los IDEs seleccionados, son:

- Oracle Application Server (IAS 10.1.0) / OC4J Standalone

  XXXXXXXXX

- Tomcat 5.0

  XXXXXXXXX

- JBoss 4.0.4

  XXXXXXXXX

2.7 SISTEMAS OPERATIVOS E INFRAESTRUCTURA TECNOLÓGICA

Las herramientas fueron instaladas en un ambiente Windows, por considerar que esta plataforma es apta para el funcionamiento de los principales IDEs. El servidor tendrá un sistema operativo Windows Server 2003 y se usará Windows XP como cliente.

El Servidor usado para la realización de pruebas cuenta con una CPU Pentium IV con 1Gb de Memoria RAM y un disco SATA (Interface: Ultra ATA-100, Buffer: 2 MB, Speed: 7200 RPM, Capacity: 60 GB, Data transfer rate: 100 MBps, Average seek time: 8.8 ms). En el mismo se instalaron las bases de datos y los Application Servers, al igual que los programas que debieron ejecutarse.

3 ANÁLISIS DE HERRAMIENTAS

3.1 INTRODUCCIÓN. HERRAMIENTAS PRODUCTIVAS.

Las organizaciones más productivas han encontrado maneras de minimizar los riesgos y maximizar las ganancias. Su estrategia depende del reconocimiento de tres realidades críticas **[McConnell96]**:

• Las herramientas productivas rara vez generan un ahorro de tiempo respecto de la promesa de su vendedor (“vendor en inglés”).

• El aprendizaje de cualquier herramienta nueva inicialmente baja la productividad.

• Las herramientas que han sido desacreditadas algunas veces producen un significativo ahorro de tiempo de todas maneras, solo que no tan significante como lo originalmente prometido.

McConnell entiende como herramienta productiva aquellas que tienen el potencial para cambiar significativamente la forma de trabajar (como herramientas 4GLs, lenguajes de programación visual, generadoras de código, código o librerías de clases) y reduzcan los tiempos de desarrollo.

3.2 ROL DE LAS HERRAMIENTAS PRODUCTIVAS EN EL DESARROLLO ÁGIL

La habilidad de desarrollar y desplegar aplicaciones es la clave del éxito en la economía de la información **[Cade02]**. Las aplicaciones deben ir rápidamente del prototipo a producción y continuar su desarrollo después de su despliegue. La productividad se hace vital. Es por esto que los autores creen que J2EE provee las herramientas necesarias para contribuir a brindar las respuestas y flexibilidad necesarias.

Otro factor de complejidad en el tiempo de desarrollo de las aplicaciones es el tipo de cliente. Aunque muchas aplicaciones pueden ser distribuidas a través de web browsers en forma estática o dinámicamente generando HTML, otras necesitan soportar clientes específicos o varios simultáneamente (por ejemplo, WAP). Por lo tanto, el modelo de programación debe soportar una variedad de configuraciones con mínimas consecuencias en la aquitectura básica de la aplicación o la lógica de negocio de la misma.

Las computadoras son buenas en automatizar tareas repetitivas. Muchas de las cosas que el software de desarrollo ha automatizado son estas tareas repetitivas (compilación, creación de descripciones de bases de datos, manejo de procesamiento de ventanas que se ejecutan una y otra vez, etc.). McConnell **[McConnell96]** menciona que sin embargo existen varios aspectos en el desarrollo de software, no repetitivo, que no encuentra las herramientas de hardware o software adecuados. El software se vuelve cada vez más preciso, más detallado y más complejo, no menos. Alguien tiene que pensar la esencia conceptual de cada nuevo programa o cualquier cambio en los viejos. Esa persona debe entender cada detalle de las funcionalidades de los programas y entender cada detalle relacionado al mismo. Tal entendimiento es dificultoso, propenso a errores y consume tiempo. Es por eso que el tiempo requerido para alcanzar ese entendimiento es uno de los principales contribuyentes al tiempo requerido para completar un proyecto de desarrollo de software.

Por otro lado, las herramientas de desarrollo cumplen un papel complementario e importante en los procesos de ingeniería de software, proveyendo un ambiente tecnológico en el desarrollo de aplicaciones (“Application Development Technology Environment” –ADTE) que acompañe las políticas de calidad, procedimientos y estándares de la organización (“Application Development Process Environment” - ADPE) **[Donaldson00]**.

3.3 CRITERIOS DE SELECCIÓN

Las herramientas estudiadas y analizadas fueron seleccionadas en base a su popularidad en el contexto de las organizaciones de desarrollo de software, que podrían denominarse “el estado del Arte” en el presente, que tengan alcance en ambientes modernos, principalmente en J2EE y lenguaje Java. Su estudio fue enfocado principalmente hacia el desarrollo de aplicaciones de gestión.

Otro de los criterios de elección de estas herramientas fue que estuvieran enfocadas principalmente en el trabajo de entornos J2EE y lenguaje Java. Aunque algunas de las mismas tengan plugins o puedan trabajar en otros lenguajes.

El hecho que las herramientas fueran multiplataforma fue considerado como criterio selector, ya que fue expuesta su importancia en los puntos (Referenciar importancia de multiplataforma en el texto).

Si bien puede considerarse como criterio arbitrario, también fue considerado el hecho de la disponibilidad de la herramienta, o sea al alcance de cualquier estudiante o desarrollador, contando con su libre “download” y actualización. De manera que existieran licencias no-comerciales para el uso en desarrollo y testing de las aplicaciones.

Justificar la selección de herramientas. Por ejemplo,

Licencias no-comerciales de uso (¿?? JDeveloper tiene licencia no-comercial?)

Lenguaje Java principalmente

Multiplataforma

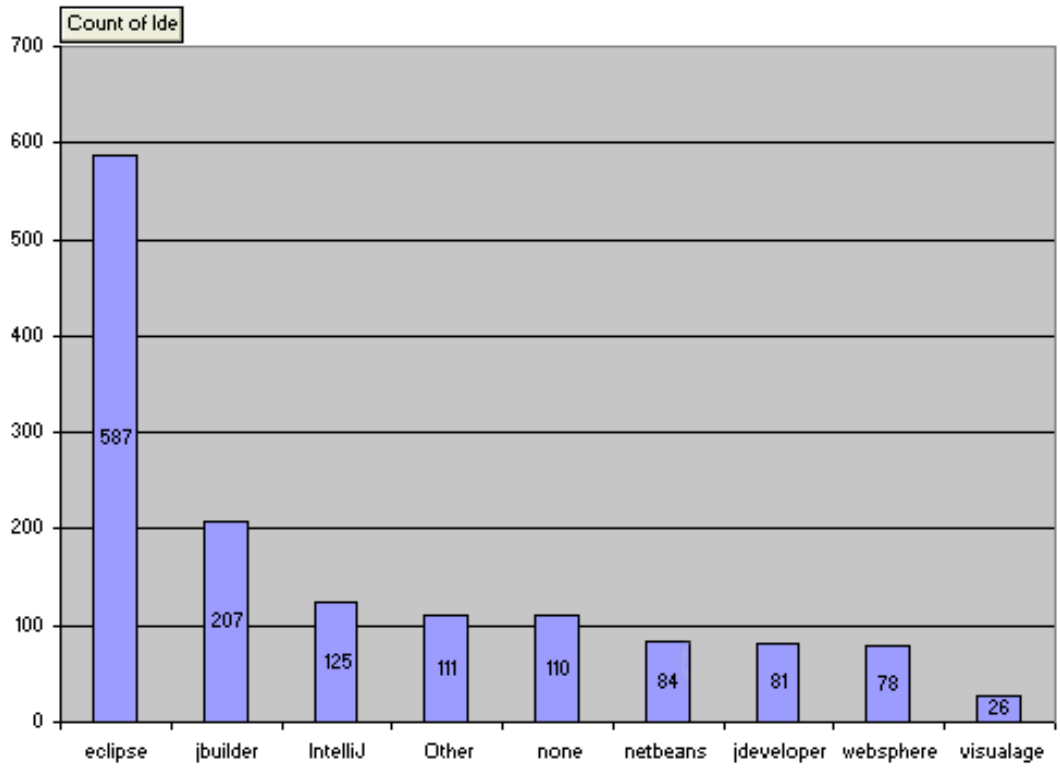
Fácil Instalación

3.3.1 Algunos otros criterios de evaluación

Es muy dificil encontrar, en forma gratuita y disponible para la comunidad, estudios relevantes y con suficientes criterios de cientificidad que puedan medir el estudio de herramientas o que comparen su uso en calidad o cantidad.

Por otro lado, las encuestas realizadas no son confiables respecto a la muestra que se tomó, fecha, población, etc. Sin embargo, a manera informativa ... (ver si vale la pena, o si se puede hablar de tendencia –entiendo que no)

1. **Encuesta de QA-Systems [QAS03][Zef03]**. Esta encuesta es presentada por esta empresa y según ellos, fue realizada en Agosto del 2003 en una encuenta que fue realizada a través de Internet. La población elegida fue la que realizó “downloads” del producto que presenta la misma (QStudio for Java Pro).



2. **El sitio Buiderau.com.ar [Neat05]** presenta un post sobre ciertas pruebas de performance que realizaron en el año 2005. El mismo presenta algunos criterios de comparación que podrían considerarse en algún estudio similar:

   - **Facilidad de uso.** Se evaluaron las distintas herramientas en la facilidad de aprendizaje de la misma y en el diseño del entorno de desarrollo, lugar de los items más comunmente usados e impresión general de la misma.

     Considero que un estudio de este tipo es muy subjetivo y para realizar un análisis de este tipo debe diseñarse una encuesta quizás compleja o determinar el conjunto poblacional a encuestar.

   - **Características y funcionalidades.** Qué características particulares o especiales tiene la herramienta, incluyendo soporte para profile, debbuger, autocompletar el código, ingeniería reversa, entre otros.

   - **Performance.** Testeo de movimientos dentro del IDE (¿?? Como testear eso?) y tiempos de compilación.

   - **Soporte para plugins.** Se consideran como ayuda para el desarrollo de componentes personalizados del entorno o hacer uso de desarrollo de terceras partes.

   - **Licencias y costos de soporte.** Esta evaluación toma mayor importancia en proyectos con muchos desarrolladores.

   - **Soporte de estándares.** Análisis del IDE en términos de incorporar estándares de Java. Esto toma real importancia a partir del crecimiento del desarrollo de nuevos productos y la necesidad de asegurar que todos los estándares y especificaciones se cumplen dentro del IDE.

3. **Encuesta en Tezaa [Tezaa06]**. Esta encuesta (actualmente on-line) presenta la posibilidad de votar por un IDE favorito dentro de los propuestos.

   Resultados al 08/10/2006

4. **Encuesta en usuario de O'Reilly [Adamson05]**. Esta encuesta, del 2005, ha sido realizada por O'Reilly con los usuarios registrados en ONJava. Según el editor de la nota, Eclipse se encabezaba las preferencias entre los lectores de ONJava con el 71% de los votos. Siguiendole otros como NetBeans con el 19% e IntelliJ con el 16% de los votos.

5. **Encuesta en el sitio www.manageability.org [Manageability].** Actualmente on-line y sin tener ningún rigor científico se publica el resultado de los votos realizados por los visitantes del sitio.

Resultados al 08/10/2006

CodeGuide	0% (13)
Eclipse	24% (449)
IDEA	10% (188)
JBuilder	2% (46)
JDeveloper	55% (1000)
NetBeans	3% (59)
Weblogic WorkShop	0% (8)
Other (i.e. vi, Emacs, JEdit etc.)	2% (38)
Total number of votes: 1801	

3.4 AMBIENTE DE PRUEBAS

Explicar bajo qué contexto se realizaron las pruebas, objetivos y formas.

3.5 ECLIPSE

Eclipse es una comunidad de código abierto que se focaliza en la construcción de una plataforma extensible de desarrollo, runtimes y frameworks de aplicaciones para la construcción, despliegue y manejo de software a través de todo su ciclo de vida **[EclipseWS]**. Aunque el nombre Eclipse se ha hecho popular como IDE de Java, su alcance real es más amplio que el desarrollo en ese lenguaje y plataforma. Su arquitectura y sistema de plug-ins hace que pueda adaptarse como un "camaleón" **[Daum04]** y pueda encontrar un habitat en diferentes entornos. Así es que Eclipse Java IDE es solo uno de los plug-ins, como también existen otros para lenguajes como C++ o PHP.

Eclipse también provee una alternativa a las librerías Java de Sun mediante SWT y JFace, de esta manera deja de ser puramente un entorno de desarrollo. Sumandose un completo framework para implementar Aplicaciones Java. Detrás de estas librería GUI, SWT y JFace, existen una cantidad de componentes como editores, "viewers", recursos de administración, tareas, sistemas de ayuda, como así asistentes y "wizards".

3.5.1 Descripción general

Organización que la desarrolla

Arquitectura / Características sobresalientes

Lenguajes soportados

Hacia donde va en la actualidad

Soporte sobre la herramienta

Releases y actualizaciones

3.5.2 Instalación

La instalación de Eclipse requiere, luego de realizar el **download** de http://www.eclipse.org/downloads, desempaquetar el archivo ZIP en un dispositivo o disco duro con suficiente espacio. Adicionalmente se requiere **[Daum04]**: **(Ver referencia y versión)**

- Contar con alguna de las plataformas permitidas. Eclipse se puede ejecutar en una amplia variedad de plataformas: Windows, Linux, Solaris, QNX, AIX, HP-UX, and Mac OS X.
- Espacio suficiente en disco. 300 MB son suficientes en una primera instancia (luego depende de los plugins que se instalen).
- Suficiente RAM. 256 MB es suficiente, aunque es recomendable el suave 500 MB o 1GB como minimo para un desenvolvimiento normal.
- Java SDK 1.4 o superior.
- Eclipse SDK (3.x) para la plataforma elegida.

El proceso de instalación consiste en, una vez instalado el el entorno Java (SDK, JRE) debe descomprimirse el archivo .ZIP en un directorio temporario. Luego debe copiarse la carpeta descomprimida en la que asignemos para la herramienta (por ejemplo C:\Eclipse). No es necesario ejecutar ningun programa instalador o registrar variables en la registry **[Burnette05]** .

A partir de esto podemos decir que la aplicación queda instalada. Esta sencilla instalación permite a su vez, luego de personalizar la herramienta y configurar el entorno de trabajo adecuado, que pueda ser transportada o replicada con la simple tarea de copiar la carpeta que hemos asignado. En esta carpeta se encontrarán las configuraciones que realicemos y los plugins que se instalen.

3.5.3 Organización de la herramienta

Espacio de trabajo (Workspace)

Ingresando por primera vez a Eclipse, y luego de sortear la página de Bienvenida, se despliega una ventana de diálogo que permite seleccionar el entorno de trabajo o filesystem donde se guardarán los proyectos que se desarrollen. Este espacio de trabajo luego podrá ser cambiado, desde el menú en la opción File > Switch Workspace, como también marcar una carpeta o directorio como opción por defecto.

Figura 4.5.1 – Página de Bienvenida

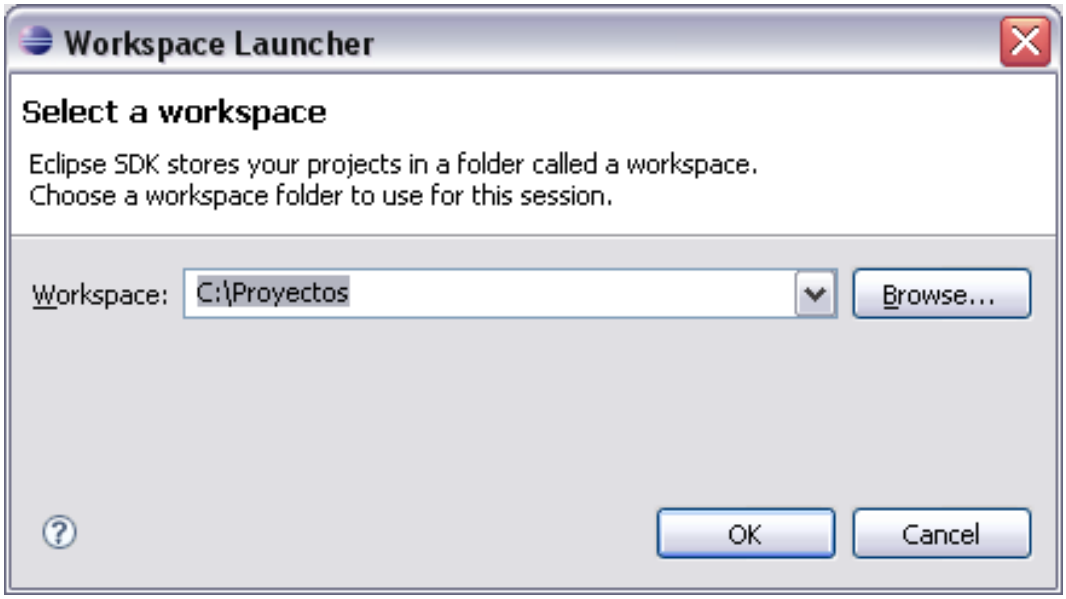


Figura 4.5.2 – Cambiar espacio de trabajo

Arquitectura de la herramienta

Descripción de la arquitectura de la herramienta.

De **[Gallardo03]**

Cuando se realiza el "download" del Eclipse SDK, lo que se obtiene es el Java Development Toolkit (JDT) para escribir y realizar debugging con programas Java. Luego el Plug-in Development Environment (PDE) sirve para extender las funcionalidades de Eclipse. El Java IDE queda incluído dentro de la funcionalidad del JDT, ni siquiera es necesario el PDE. Por lo tanto cuando se habla de Eclipse, se está hablando de la Plataforma Eclipse, y no solo del IDE. De esta manera la plataforma tiene como objetivo la integración de las herramientas, a través de extensiones tales como el JDT o CDT (Plug-in para C/C++).

Como agregado a la plataforma tenemos el workbench o interfaz de usuario, el workspace, help, team (componentes de equipo o trabajo colaborativo) y agregado de otras herramientas de terceros.

Plataforma de ejecución

El trabajo primario de la plataforma de ejecución es saber que plug-ins están disponibles en Eclipse. Cada plug-in tiene un manifiesto en XML, en la carpeta de plug-ins, que indica las conexiones que son requeridas para éste. Los mismos no son leídos o "activados" por la plataforma completamente hasta que no son requeridos, para optimizar el tiempo de start-up y los recursos.

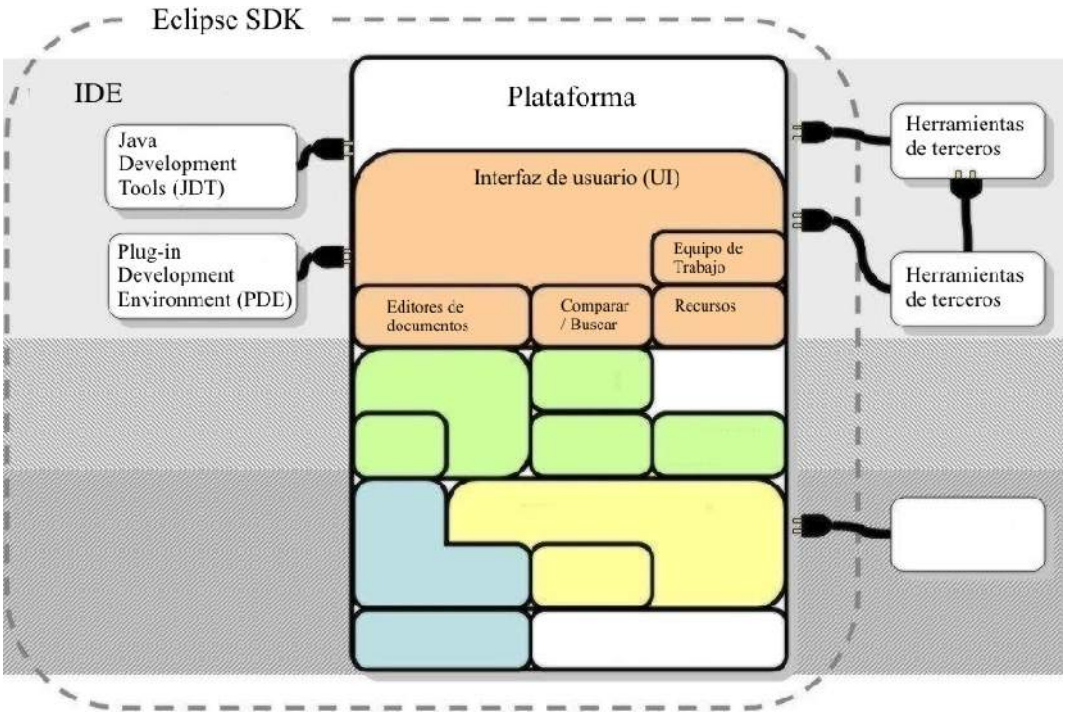


Figura 4.5.3 – Arquitectura de la herramienta FALTA TERMINAR DIBUJO

Soporte para Equipo de Trabajo (Team)

El plug-in de Team facilita el uso de la herramienta en trabajo colaborativo, permitiendo el control de versiones y el manejo de los recursos compartidos. Eclipse incluye un cliente para CVS (Concurrent Versions System) (Ver Integración con herramientas de control de versiones)

Ayuda

Al igual que los demás componentes de la plataforma el módulo de ayuda (Help) es un sistema de documentación extensible. Es posible sumar documentación en HTML y, a través de XML, definir la estructura de navegación.

Disposición del entorno de trabajo

El término inglés **"workbench"** refiere al panel o escritorio de desarrollo. El workbench apunta a alcanzar la integración de herramientas, controlar su apertura y proveer un paradigna común para la creación, manejo y navegación entre los distintos recursos del espacio de trabajo **[EclipseDOC]**.

Cada ventana del workbench contiene una o más perspectivas. Las perspectivas contienen vistas y editores y controlan que la apariencia en determinados menúes y barras de herramientas. Pueden existir varias ventanas del workbench al mismo tiempo.

Una **perspectiva** es un grupo de vistas y editores dentro de una ventana del workbench. Una o más perspectivas pueden coexistir dentro de la misma ventana, y a su vez cada una puede contener una o varias vistas y editores.

Una **vista** es un componente visual dentro del workbench. Se usa tipicamente para navegar dentro una jerarquía de información (como la vista de recusros (Resources)), abrir un editor o mostrar las propiedades del editor activo. Las modificaciones en una vista son salvadas inmediatamente. Normalmente solo una instancia de un tipo particular de vista puede existir dentro de una ventana del workbench. Las referencias 4, 6 y 7 en el gráfico 4.5.4, son ejemplos de vistas.

Un **editor** es también un componente visual dentro del workbench. Es usado generalmente para editar u hojear un recurso. Por ejemplo, un archivo con extensión .java (fuente de código Java). Las modificaciones dentro de un editor siguen el cilo de abrir-grabar-salir como otros editores (por ejemplo de textos).Multiples instancias de un tipo de editor pueden coexistir dentro del workbench. La referencias 3 en el gráfico 4.5.4, es un ejemplo de editor.

Los editores también pueden ser externos. Esto quiere decir que es posible usar una aplicación externa para editar algún tipo de archivo. Puede ocurrir en casos en los que el workbench no tenga un editor para ese tipo de archivo o el usuario desee usar uno diferente del provisto por la herramienta. La herramienta luego provee la forma de

poder abrir el recurso seleccionado con las diferentes herramientas asociadas al tipo de archivo, llamando al menú contextual con el botón derecho del mouse, en la opción "Open with".

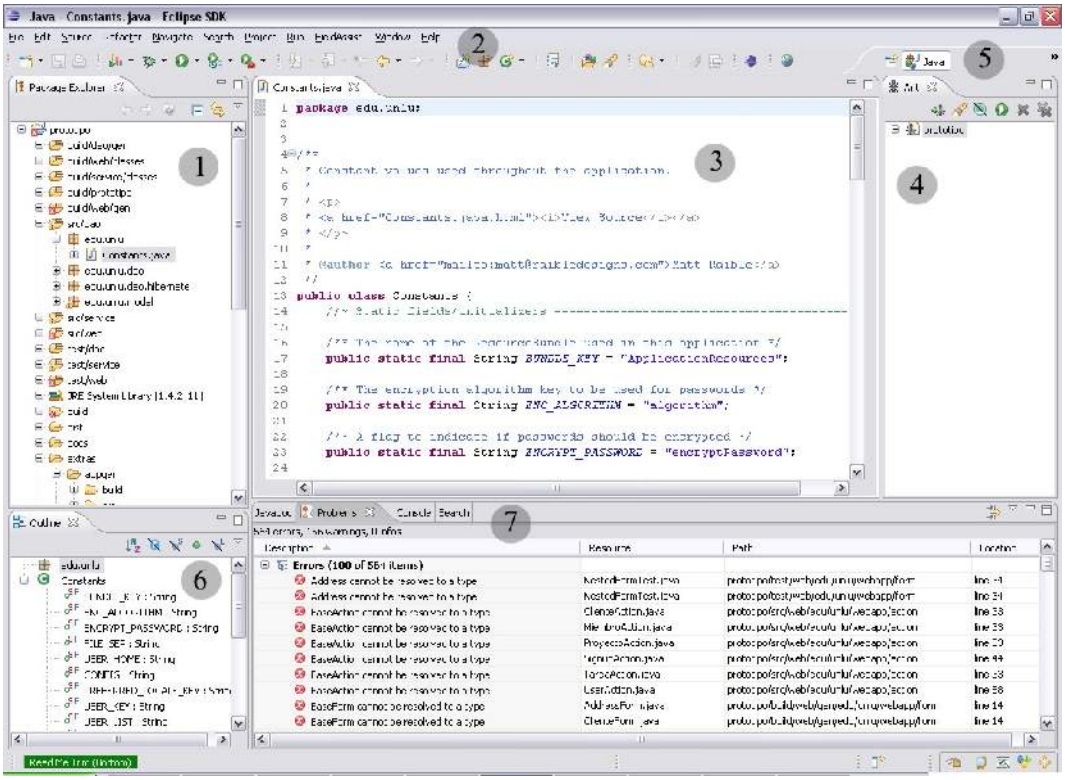


Figura 4.5.4 – Disposición del entorno de trabajo

Referencias: 1 – Package Explorer, 2 – Menú principal, 3 – Editor, 4 – Vista de Ant, 5 – Tab de perspectivas, 6 – Vista de Outline, 7 – Vista de Problemas (Activa).

Menúes y toolbars

Dentro de la interfaz de usuario (Workbench user interface – UI) también se encuentran el menú principal, el menú de herramientas, y las herramientas de atajo (shortcut toolbar). Al igual que las vistas y los editores, estos elementos pueden cambiar dependiendo de las tareas y funcionalidades (features) disponibles en la perspectiva actual.

Opcionalmente se pueden agregar otro tipo de atajos: botones de vistas rápidas (Fast Views). Estos proveen una forma de localizar y abrir una vista dentro de una perspectiva desde un ícono en una forma similar al minimizar una ventana en varias aplicaciones. Esto crea un menú de íconos en la parte inferior de la herramienta para cada perspectiva y de esa manera personalizar el entorno de trabajo. Por ejemplo, es posible que la vista "Ant", dentro de la perspectiva "Java", no la usemos frecuentemente. Por lo cual podemos indicarle, desde el menú contextual de la vista, la opción "Fast View". Esto esconderá la vista e incorporará el ícono en la barra de herramientas de acceso rápido. (Mostrar gráfico).

Lenguajes soportados

Aunque Eclipse es más popular por ser un IDE para Java, existen plug-ins para otros lenguajes de programación. De manera que pueden adicionarse lenguajes como C/C++, Cobol, Fortran y PHP. Estos son proyectos de desarrollos separados con evoluciones distintas. Por ejemplo el proyecto para PHP se llama PHP IDE (Antes era PHPEclipse)

A su vez existen packs para los lenguajes de la herramienta. Estos son traducciones para los distintos componentes de la arquitectura (SDK, Plataform runtime, JDT, PDE, etc.). Los mismos han sido donados por IBM. **[EclipseLANG].**

Portabilidad

Si bien fue escrito en Java, que su principio es permitir la ejecución en cualquier plataforma, Eclipse no es estrictamente neutral en cuanto a las plataformas de ejecución. Esto está dado así, ya que usa gráficos nativos del sistema operativo. Eclipse entonces está disponible para las plataformas para las que se hayan portado las herramientas de SWT. Sin embargo, al ser un proyecto open-source, es posible portar Eclipse a otras plataformas. En algunos casos estas versiones son retornadas a la

comunidad de desarrollo y de esta manera pueden quedar incorporadas al proyecto y a las versiones oficiales del producto.

(VER ESTA PARTE Y COMPLETAR con versiones disponibles y plataformas)

3.5.4 Codificación

Manejo de proyectos

De **[Gallardo03]**

Crear un proyecto Java

Antes de escribir cualquier tipo de código, en Eclipse es necesario crear un proyecto. Si bien soporta potencialmente, a través de plug-ins, una variedad importante de tipos de proyectos (EJB, Web, etc.), soporta básicamente 3 tipos:

Plug-in Development. El cual provee el entorno para desarrollar los propios plug-ins para Eclipse.

Simple. Provee un entorno genérico, por ejemplo para crear documentación.

Java (u otro lenguaje). Esta opción configura el entorno de desarrollo específico para el tópico elegido disponiendo de funcionalidades y configuraciones específicas para el lenguaje.

Al crear un proyecto se crea una carpeta en el directorio seleccionado como workspace, donde serán guardados los archivos generados dentro del proyecto. Si se desea eliminar el proyecto, Eclipse dará la opción de mantener esta carpeta, eliminando solo lógicamente el proyecto, o eliminándola junto con el mismo.

Importar un proyecto externo

Hay varias maneras de importar un proyecto externo, por ejemplo realizado en otro entorno o IDE. Una forma es copiar el código fuente en una carpeta dentro del workspace en un proyecto existente o creando una nueva carpeta.

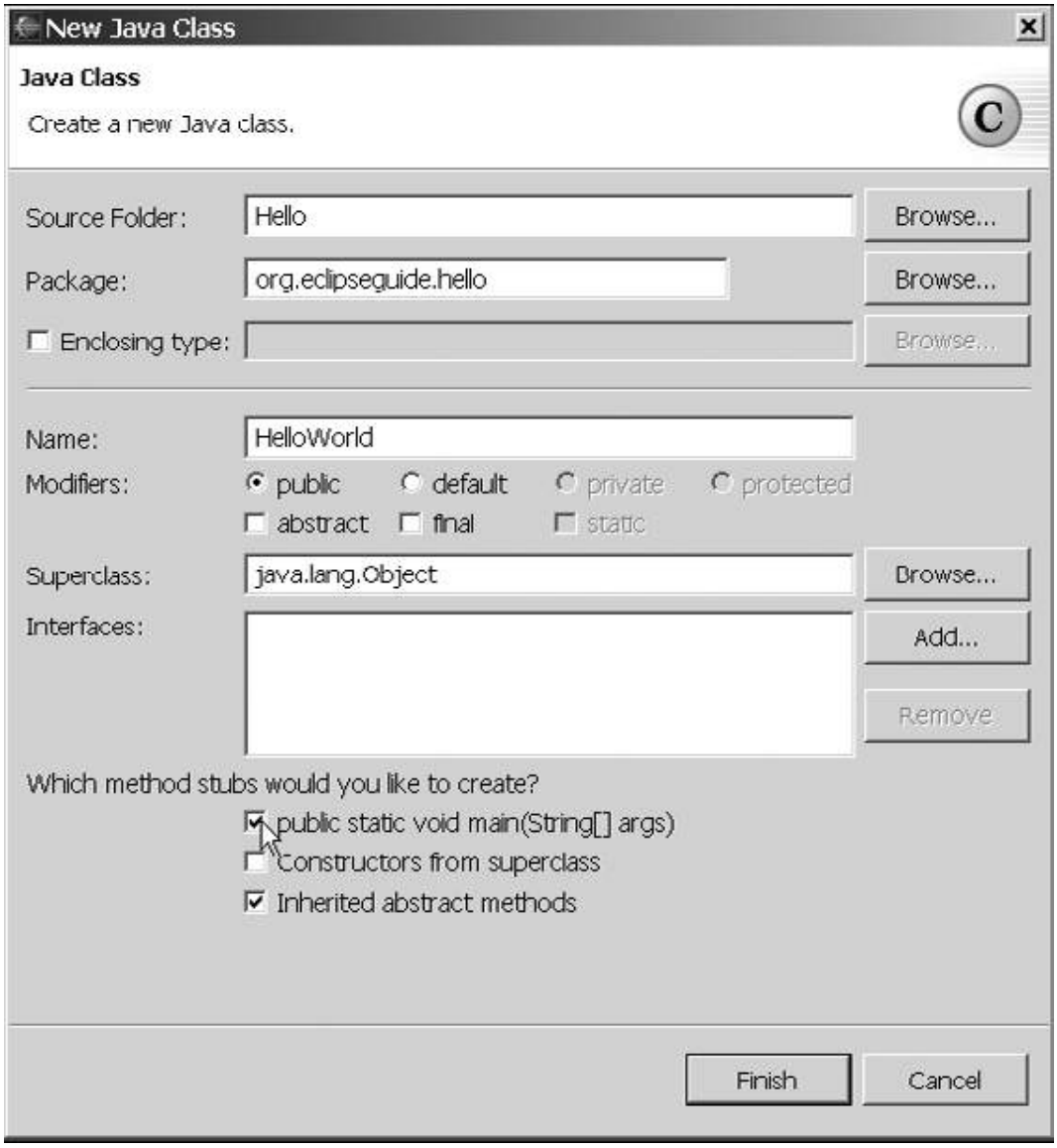
Otra forma es creando una carpeta en el sistema de archivos y luego crear un proyecto indicando el nombre de la carpeta. Eclipse notará que la carpeta existe y dará la opción para crear el proyecto en la misma. Luego buscará los archivos fuentes y las librerías existentes y los tomará por defecto. Adicionalmente se le pueden indicar otras carpetas fuentes.

(AMPLIAR ESTOS CONCEPTOS – Poner como ejemplo la importación de un proyecto proveniente de JDeveloper o Netbeans)

Creando una clase Java

Una vez creado el proyecto, es posible crear una clase Java. (VER CONTENIDO)

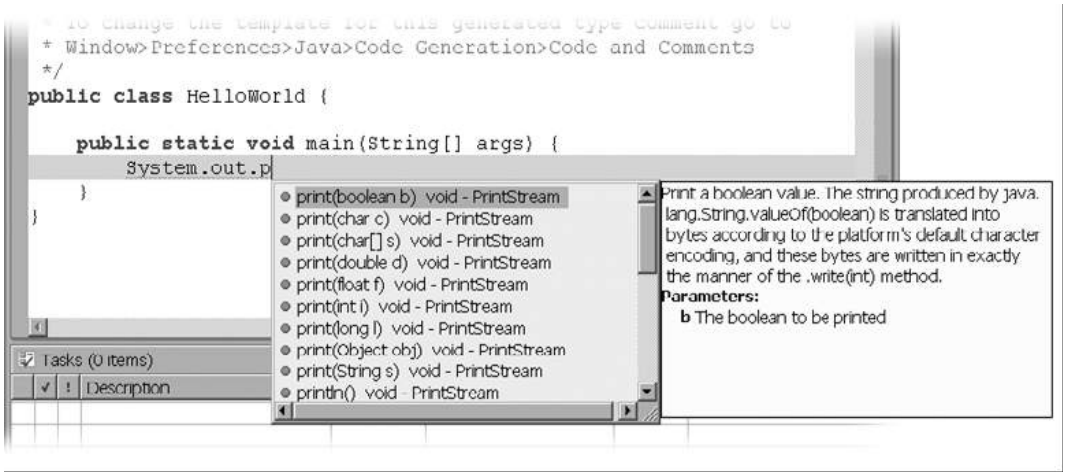
(PONER GRAFICO)



Ayuda en línea y creación de código automáticamente

De **[Gallardo03] y otros**

Uno de las primeras habilidades para completar el código automáticamente se encuentra en el hecho que si tipea un paréntesis de apertura o las comillas dobles, Eclipse automáticamente pone su par. Esta funcionalidad puede desactivarse entrando a la opción Preferentes → Java / Editor / Typing / Automatically close.



Otra de las funcionalidades para completar código es la de asistente (code assist). Por ejemplo, si se está escribiendo System.out.println (dentro de una clase main()), a medida que se va escribiendo, Eclipse presenta una serie de propuestas que incluyen los métodos de la clase y sus atributos, con los comentarios de Javadoc. Estos pueden ser navegados y seleccionados, o bien con las flechas o tipeando la primer letra de la palabra buscada. Esta opción también puede ser activada con Ctrl. + Tecla espaciadora. Esto último puede ser útil si se está escribiendo el nombre de una clase larga. Por ejemplo al escribir Sys y luego presionar Ctrl+Space se mostrarán las opciones System, SysexMessage, SystemColor, SystemException, etc.

Herramientas para análisis de código

La herramienta TPTP (Test and Performance Tools Platform) provee la capacidad de integrar una variedad de herramientas de análisis estático. Es posible definir reglas o categorías de reglas. El análisis genera un reporte, en una nueva vista, con los resultados que no conforman las reglas definidas. Las perspectivas de Java y Debug proveen las capacidades de análisis por defecto, pero éstas pueden ser incluidas en otras.

VER SI SE PUEDE PONER UN EJEMPLO

Páginas Scrapbook

Una funcionalidad de ayuda cuando se está programando son las páginas scrapbook. Estas sirven para probar expresiones de Java en forma aislada. Al crear páginas de este tipo Eclipse provee un editor de texto donde pueden volcarse expresiones como System.out.print(3 + 4) , java.util.Date() o for(int i = 1; i < 10; i++) { HolaMundo.mostrar(Integer.toString(i)); }. Las mismas pueden ser ejecutadas, inspeccionadas o mostrarse en el editor **[EclipseHLP]**.

VER SCRAPBOOK para SQL - Interesante

Plantillas de generación de código

Como anteriormente se mencionó, presionando las teclas Ctrl+Space Eclipse propone alternativas para completar el código automáticamente o propone estructuras pre-definidas. Por ejemplo, si se escribe la sentencia "for" y luego se presiona Ctrl+Space, Eclipse propondrá una lista alternativa de estructuras posibles para esa palabra, como puede ser "for (int ${index} = 0; ${index} < ${array}.length; ${index} ++) {${line_selection}${cursor}}". Estas plantillas o "templates" pueden ser cambiadas e incluso agregar nuevas.

A su vez es posible agregar o modificar las plantillas de documentación en las acciones de creación de clases o métodos del tipo get o set, entre otros **[EclipseHLP]**.

Refactoring

La recomendación de los métodos ágiles (VER REFERENCIAS) de construir aplicaciones de manera incremental, hace que se haga necesario el re-diseño o la revisisión de aspectos que fueron considerados de una forma en principio y luego deben ser cambiados. Esta reconsideración, que puede afectar los nuevos requerimientos, hace necesaria una readaptación y reestructuración del código, sin afectar las estructuras pre-existentes. Podemos decir que la implementación de estos cambios es lo que llamamos "refactoring".

Renombrar un componente

Este cambio es relativamente simple y consiste en la revisión del código para cambiar todas las referencias al componente determinado. Este puede ser una clase, un

método o un campo. Por otro lado un reemplazo de la forma de un procesador de textos puede cometer errores ya que debe tenerse en cuenta el contexto y la semántica de la sentencia.

Completar con un ejemplo y con las pruebas realizadas

3.5.5 Desarrollo Web (Web Development, Deploy, etc.)

De **[Gallardo03]**

Eclipse provee varias herramientas que dan versatibilidad y permiten la adaptación de distintos estilos de programación. (XP en contraste con Waterfall). Por ejemplo, en XP (eXtreme Programming) y demás metodologías ágiles en contraste con otras metodologías más tradicionales (Waterfall, por ejemplo) se basan en procesos iterativos más que en fases. COMPLETAR IDEA

(Explicar como se adapta al concepto de arquitectura MVC)

Uno de los sub-proyectos de desarrollo de Eclipse es el de Web Tools Platform (WTP). El mismo consiste en el desarrollo de herramientas para la construcción de aplicaciones J2EE. Este proyecto incluye, entre otros, tópicos como: editores para HTML, Javascript, CSS, JSP, SQL, XML, DTD, XSD, y WSDL (Web Services); constructores de modelos para J2EE, wizards y exploradores para proyectos J2EE y Web Services.

El plug-in de WTP viene incluido en la versión 3.2 de Eclipse y provee un entorno integrado con capadidades como:

- Crear un proyecto de tipo Dynamic Web, usando una jerarquía definida para J2EE.
- Creación y edición del archivo descriptor para una aplicación Web (web.xml)
- Creación, edición y debbuging para JSP y HTML.
- Asistente de código dinámico para los tags e HTML, JSP y JavaScript. Incluyendo una vista extensible para organizar código reusable.

- Edición de CSS
- Importación HTTP/FTP y exportación FTP a un servidor.
- Manejo de archivos WAR
- Wizards para creación de servlets

Creando y testeando JSP

Abrir tipo de proyecto WEB y poner Dynamic Web Project, e indicarle como Target Runtime Apache Tomcat v5.0
Probar con JSP e integración con Tomcat

Crear un proyecto JBoss (ver en integración con Application Servers)

Creando y testeando un servlet

Armar un servlet y ejecutarlo

Tecnologías involucradas para la realización de las distintas etapas de desarrollo.

Uso de frameworks

3.5.6 Testing

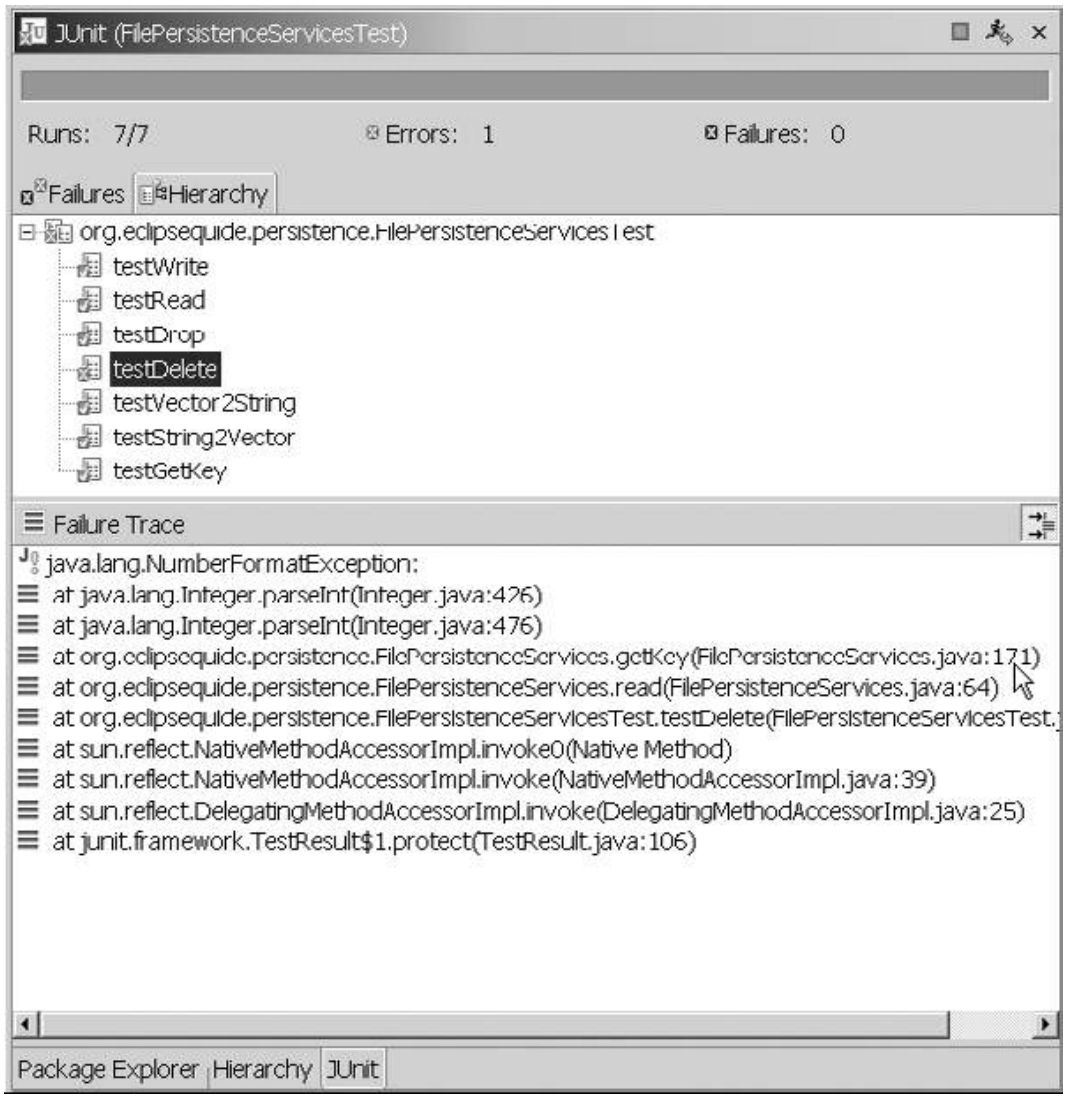
JUnit es un framework open-source escrito por Kent Beck, principal divulgador de XP, y Erich Gamma, uno de los lideres de desarrollo de Eclipse JDT. Esta herramienta es muy útil para hacer las cosas más fáciles en lo que a testing se refiere, y en Eclipse se integra a través de un plug-in, como los demás componentes.

El primer paso es sumar el archivo JAR JUnit al classpath. Esto se puede hacer o bien sumando explícitamente el archivo o definir una variable de classpath en el proyecto o configuración general. (MOSTRAR como sería).

Mostrar una prueba con JUnit.

3.5.7 Debugging

Una dificultad común es la de encontrar donde ocurre un problema dentro del código.



Configurando breakpoints

Es posible configurar un breakpoint o punto de corte en un lugar específico de nuestro código. Esto se puede hacer … (ARMAR UN EJEMPLO)

Otras veces no se sabe cuantas veces se pasa por el breakpoint antes que ocurra el error, pero se hace necesario mirar condiciones específicas. (ARMAR UN EJEMPLO).

3.5.8 Logging

Una alternativa a las técnicas de testing y debbuging es la de usar comandos de impresión. Por ejemplo utilizar el método println() dentro de una clase que es instanciada y quiere imprimirse el resultado de tal instanciación o del valor de alguna variable. Es posible crear unidades de testeo poniendo un código como:

```
public class ClasePrueba {
	public static void main(String[] args)
	{
		FileIO.drop("TablaDePrueba");
		Vector v = new Vector();
		v.addElement("Uno");
		v.addElement("Dos");
		v.addElement("Tres");
		boolean b = FileIO.write("TablaDePrueba", 1, v);
		Vector w = FileIO.read("TablaDePrueba", 1);
		System.out.print("Cantidad: " + w.size());
		v = new Vector();
		v.addElement("A");
		v.addElement("B");
		v.addElement("C");
		v.addElement("D");
		b = FileIO.write("TablaDePrueba", 2, v);
		w = FileIO.read("TablaDePrueba", 2);
		System.out.print("Cantidad: " + w.size());
//		etc.
	}
//	etc.
```

REEMPLAZAR POR EJEMPLO DEL PROTOTIPO

Sin embargo existen herramientas específicas que pueden facilitar este tipo de trabajo. Por ejemplo, **log4j**. Esta herramienta tiene mayor flexibilidad que la de imprimir mensajes en un archivo o en la consola. Existen 3 acciones que pueden

suceder dinámicamente (normalmente configurado en un archivo), antes que un mensaje sea enviado:

- Al mensaje se le asigna una prioridad y se filtra de acuerdo a la misma;
- El o los destinos del mensaje son determinados dinámicamente;
- El mensaje es formateado apropiadamente para cada destino.

loggers, *appenders,* and *pattern layouts*.

Loggers

Un logger es usado en una aplicación como el System.out. Es usado para imprimir sentencias y es un objeto que se usa para enviar mensajes.

Los loggers tienen jerarquía. El logger root es anónimo y existe automáticamente. Para obtener este logger se usa el método estático Logger.getRootLogger(). Sin embargo es preferible instanciar uno propio que derive del root, llamándolo con el método Logger.getLogger().

Asumiendo que está todo correctamente configurado, se puede obtener un logger, llamado myLogger, de la siguiente forma:

logger = Logger.getLogger("myLogger");

Los Loggers no tienen métodos simples como print() o println(); en cambio tienen métodos que indican la prioridad del mensaje. Los 5 métodos, en orden ascendente de prioridad, son los siguientes:

- ü debug()
- ü info()
- ü warn()
- ü error()
- ü fatal()

Como ejemplo, se puede enviar información útil para propósitos de debbuging con el método debug(): logger.debug("Ejecutando método"). Por otro lado, es posible

que se quiera realizar logs de excepciones (como IOExceptions) con alta prioridad. Por lo tanto es posible tener un código como el siguiente:

```
catch (IOException e)
{
logger.error("Caught:" + e);
}
```

Luego la prioridad determina si el mensaje es enviado a su destino. Esto hace posible que se puedan proveer diferentes niveles de información dependiendo de la circunstancia.

(Ver el uso de log4j en Eclipse)

3.5.9 Integración con RDBMS

El proyecto WTP, mencionado anteriormente, también incluye herramientas de **modelado y acceso a la base de datos** a través de consultas SQL. Estas herramientas facilitan las tareas para el tipo de desarrollo en capas, donde es necesario separar la presentación de la lógica de negocios y acceso a datos.

En Eclipse es posible agregar un acceso a la base de datos a través de JDBC, en una forma relativamente sencilla. Luego los datos podrán ser consultados a través de SQL, usando la herramienta **Data** del WTP.

3.5.10 JDBC Soporte para otras Bases de Datos

La figura 4.5.8 muestra una configuración para la base de datos MySQL. Para configurar una conexión es necesario especificarle el driver, generalmente provisto por el "Vendor" o Proveedor de la base, y la URL de la conexión para la base de datos seleccionada. De esta manera, a través de JDBC, es posible configurar otras bases de datos. A continuación se listan las conexiones, con las clases de driver, las URLs de conexión y los ficheros .jar de los drivers, para las bases de datos más utilizadas:

- **MySQL**
  - *Clase Driver* : com.mysql.jdbc.Driver
  - *URL de Conexión*: jdbc:mysql://<host>/<database>
  - *Fichero .jar/.zip*: mysql-connector-java-3.1.14-bin.jar
- **Oracle**
  - *Clase Driver* : oracle.jdbc.driver.OracleDriver
  - *URL de Conexión*: jdbc:oracle:thin:@ <host>:<port>:<sid>
  - *Fichero .jar/.zip*: classes12.zip
- **SQLServer**
  - *Clase Driver* : com.microsoft.jdbc.sqlserver.SQLServerDriver
  - *URL de Conexión*: jdbc:microsoft:sqlserver://localhost:1433
  - *Fichero .jar/.zip*: mssqlserver.jar, msbase.jar, msutil.jar
- **PostgreSQL**
  - *Clase Driver*: org.postgresql.Driver
  - *URL de Conexión*: jdbc:postgresql://<server>:<port>/<database>
  - *Fichero .jar/.zip*: postgresql.jar
- **DB2**
  - *Clase Driver* : COM.ibm.db2.jdbc.app.DB2Driver
  - *URL de Conexión:*: jdbc:db2:<database>
  - *Fichero .jar/.zip*: db2java.zip
- **Sybase**
  - *Clase Driver* : com.sybase.jdbc2.jdbc.SybDriver
  - *URL de Conexión*: jdbc:sybase:Tds:<host>:<port>/<database>
  - *Fichero .jar/.zip*: jconn2.jar

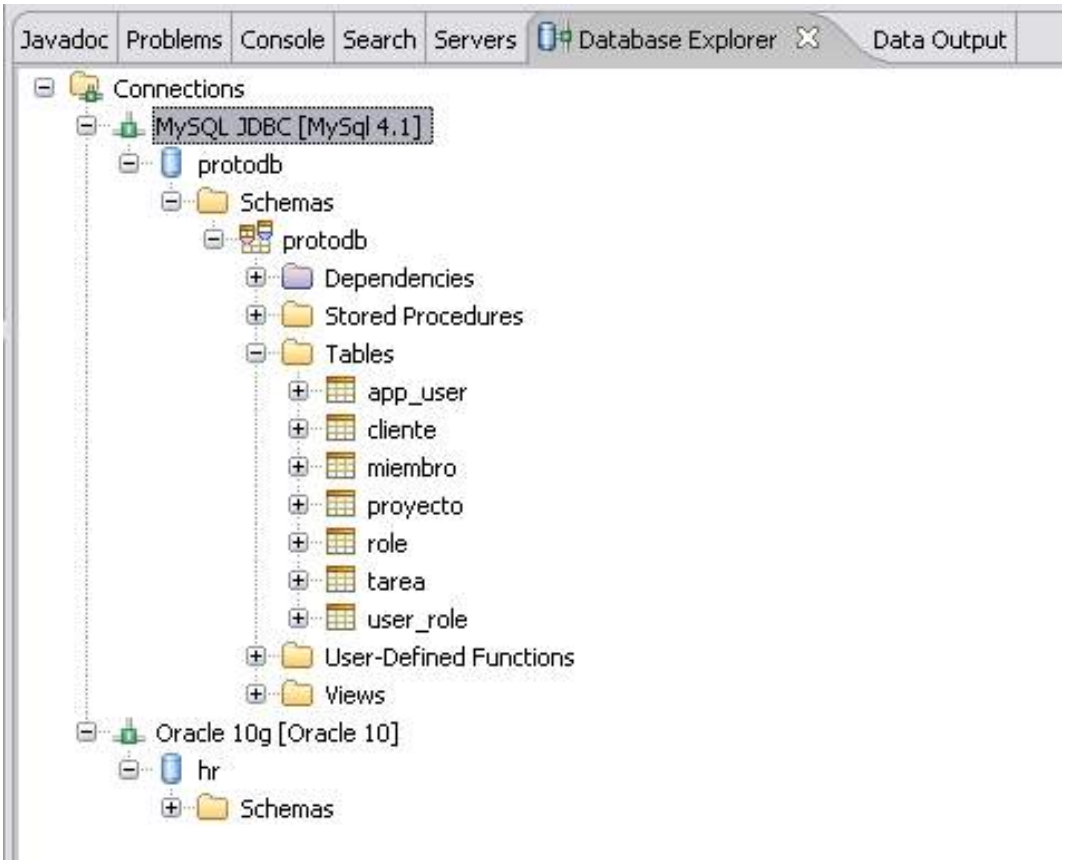


Figura 4.5.8 – Vista de Database Explorer

Otra forma de integración es a través de plug-ins de terceros. Uno de esos plug-ins es Eclipse SQL Explorer **[EclipseSQLExp].** En este caso el plug-in se copia en la carpeta de plugins del directorio de Eclipse y queda instalado al reiniciar la aplicación. Al igual que el caso de las herramientas de WTP, esta herramienta suministra una serie de vistas como Connections, para editar las conexiones; Database Detail, para acceder a la descripción y detalles de las tablas; Database Structure, para observar la estructura de la base con sus objetos; y SQL Results y History, para mostrar las consultas realizadas y un historial de las últimas realizadas.

Es importante resaltar que muchos de los plugins pueden convivir con las herramientas WTP. Esto hace que la herramienta sea flexible y pueda ser configurada por el usuario con la herramientas más cómoda.

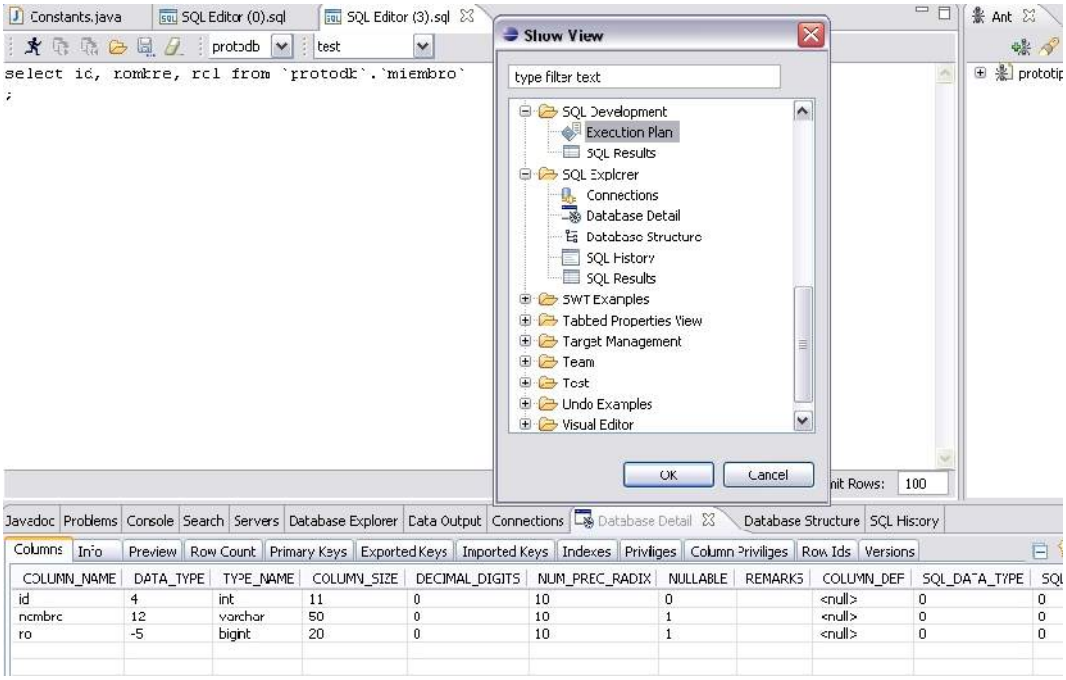


Figura 4.5.8 – Vistas de Eclipse SQL Explorer

Referencia de Downloads de los JDBC Drivers

SQL Server 2005 - http://www.microsoft.com/downloads/details.aspx?FamilyId=6D483869-816A-44CB-9787-A866235EFC7C&displaylang=en

MySQL - http://dev.mysql.com/downloads/connector/j/3.1.html

Oracle - http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html

Third-party Database Configuration

http://www.caucho.com/resin-3.0/db/thirdparty.xtp#Oracle

3.5.11 Integración con herramientas de UML

Generación de diagramas de UML con la herramienta e interacción con el código. Ingeniería reversa e integración.

3.5.12 Integración con Application Servers

Así como es posible integrar a través de las herramientas WTP las diferentes base de datos, tambien es posible agregar Application Servers para poder ejecutarlos dentros del entorno de Eclipse y de esa manera poder tener un ambiente integrado para Testing y Desarrollo.

Para empezar a integrar Application Servers con las herramientas provistas por WTP es necesario agregar la vista de Servers. Esto se realiza desde el menú Window / Show view / Others. Esta acción agregará la vista de Servidores que estén configurados y dará la opción a configurar nuevos (ver figura 4.5.5).

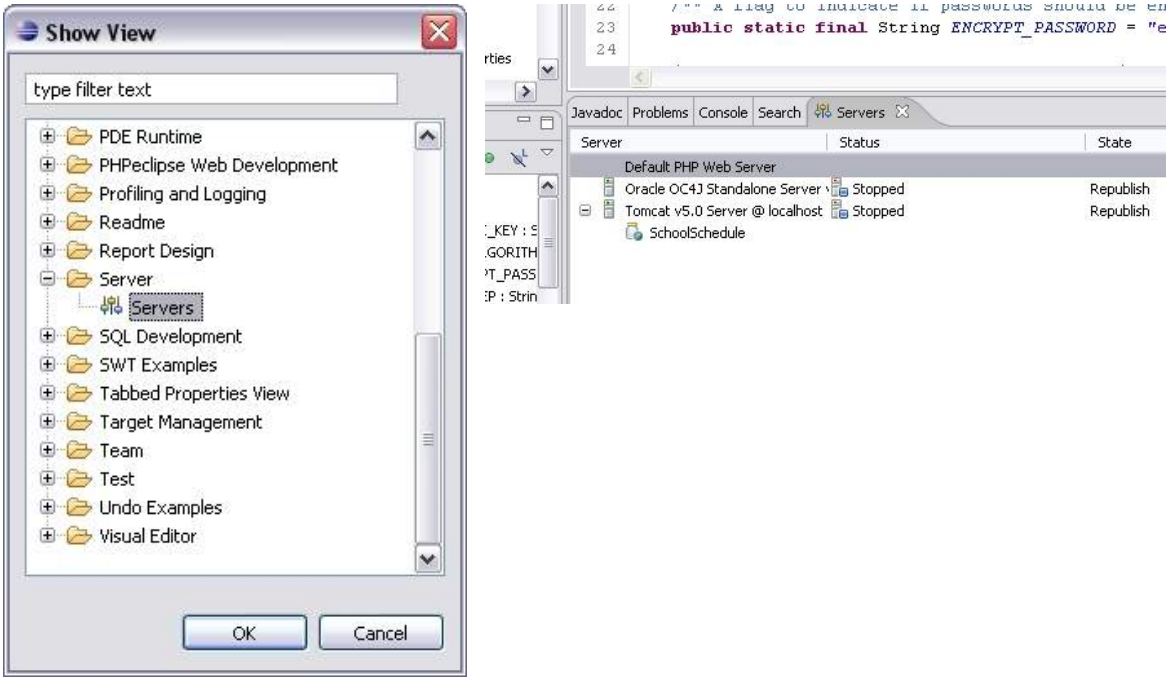


Figura 4.5.5 – Vista de Servers

Para definir un nuevo servidor Eclipse muestra un wizard que pide los datos del servidor y los datos de configuracion del mismo. Estos pueden variar en los distintos servidores. La lista soportada por defecto es la siguiente: Tomcat de Apache (en las versiones 3.2 a 5.5), Generic BEA Weblogic Server (versiones 8.1 y 9.0), IBM WebSphere (v6.0), JBoss (v3.2.x y 4.0), JonAS v4 de ObjectWeb y Oracle OC4J

Standalone Server 10.1.3. Adicionalmente se pueden bajar e installar nuevos servidores que van publicando los desarrolladores del proyecto.

Una vez configurados los servidores pueden ser ejecutados dentro del entorno de Eclipse. Esto significa que solo se ejecutan dentro de Eclipse y no son accesibles desde fuera de la herramienta. Por ejemplo, se configuró el servidor Tomcat 5.0 para ejecutarlo en el puerto 8081. El mismo puede ser arrancado desde el sistema operativo por medio del archivo batch provisto por la aplicación (Directorio de Tomcat\bin\startup.bat), o arrancando el servicio si previamente fue instalado. De esta forma es posible ejecutar las aplicaciones que son copiadas en el Application Server o la aplicación de Manager que provee Apache. Estas tareas no pueden ser realizadas si el servidor es ejecutado dentro del entorno Eclipse. La imposibilidad de realizar esta tarea posiblemente se de porque los parámetros de ejecución de la VM (Virtual Machine) no son los mismos. Eclipse permite modificarlos en la opción Open launch configuration / Arguments. Sin embargo se notó que si bien pueden ser cambiados los argumentos y aplicados, al volver a ingresar se ven los parámetros anteriormente existentes, configurados inicialmente por la herramienta. Esto supone que puede existir un bug en este desarrollo **[BitácoraS18]**.

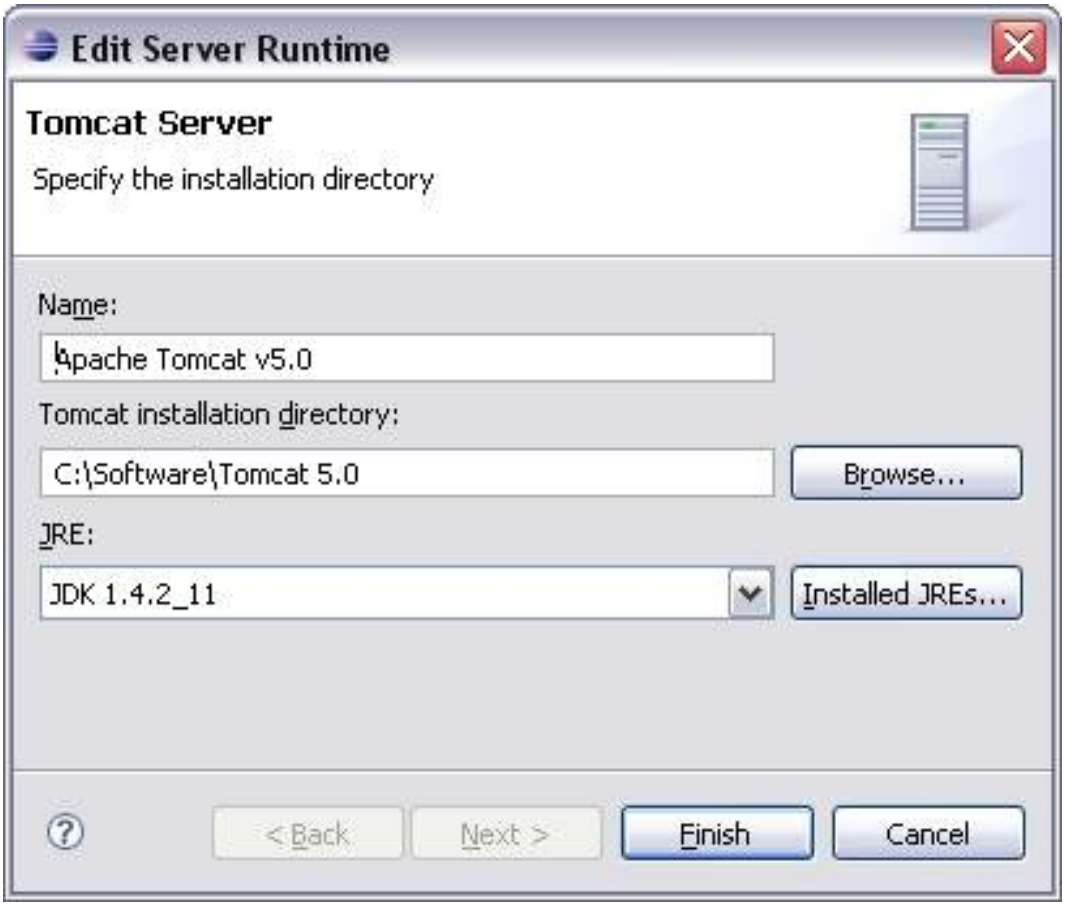


Figura 4.5.6 – Configuración de Servers (1)

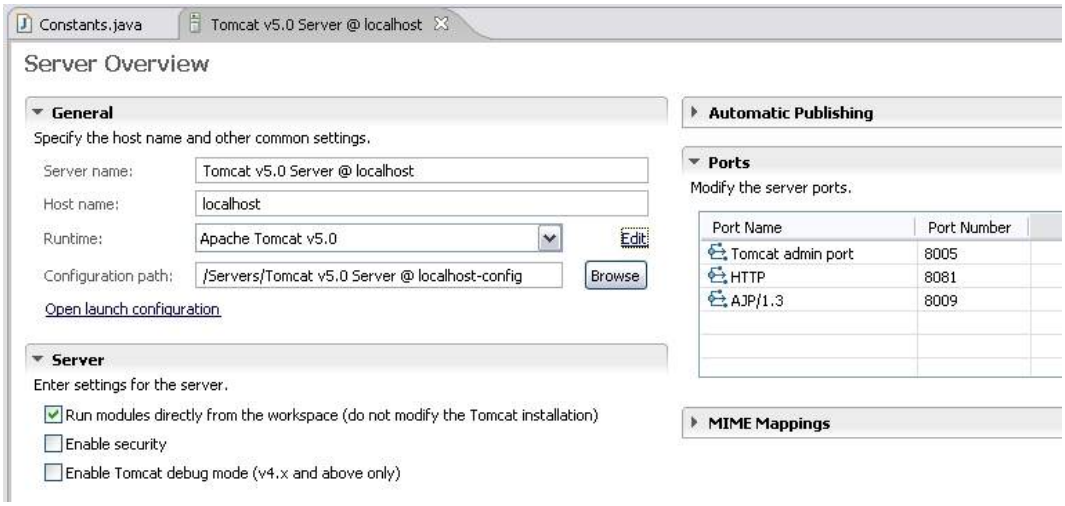


Figura 4.5.6 – Configuración de Servers (2)

Existen otras maneras de integración con Application Servers que no son a través de WTP. Tal es el caso de herramientas de terceros como **ObjectWeb Lomboz** o **JBoss Eclipse IDE**.

Lomboz es un entorno de desarrollo J2EE open source y libre que brinda herramientas para desarrollar, testear, hacer profiling y construir aplicaciones Web. Lomboz está construido sobre la paltaforma Eclipse y las herramientas de WTP y es desarrollado por una comunidad de desarrolladores denominada ObjectWeb.

JBoss

JBoss Eclipse IDE son una serie de plug-ins integrados de Eclipse para la construcción de aplicaciones basadas en JEMS (JBoss Enterprise Middleware System). Estos plug-ins son desarrollados por JBoss, ahora perteneciente a la empresa RED HAT. Según JBoss **[JBossPRDS]**, estas herramientas incluyen, entre otras cosas, lo siguiente:

- Un entorno de desarrollo integrado para Eclipse 3
- Un wizard para proyectos EJB 3.0
- Herramientas para Hibernate
- Herramientas para Programación Orientada a Aspectos (AOP)
- Wizards para facilitar el desarrollo J2EE
- Un diseñador gráfico de procesos JBoss
- Integración de debugging, monitoreo, y cliclo de vida de los servidores JBoss
- Editores para JSP, HTML, y XML

La integración del IDE con el Application Server debiera brindarnos algunas ventajas respecto del funcionamiento de cada componente en forma separada. Las mismas pueden estar basadas en distintos aspectos. Nos basaremos en 3 principales que son: facilidades en el despliegue de la aplicación en el Servidor de Aplicaciones,

mejoras y habilidades para facilitar la codificación en temas inherentes al Servidor de Aplicaciones (tales como persistencia en EJB) y por ultimo tareas de debugging o profiling. Un aspecto a tener en cuenta es que los distintos IDEs no se integran de la misma forma con los diferentes Application Servers. Esto nos lleva a decir que existen ciertas dependencias o preferencias entre las herramientas de desarrollo elegidas y el Servidor de Aplicaciones, ya que en algunos casos se brindarán mejores funcionalidades de acuerdo a la organización que provea la herramienta. Es así que es posible que JBoss se integre de mejor forma con Eclipse, proveyendo una herramienta como JBoss IDE for Eclipse **[JBossIDE]** o con NetBeans **[JBossNBIDE]**, mientras que Oracle apoyará sus esfuerzos de integración con el Application de su organización, dando ventajas a JDeveloper para integrar con el Oracle Application Server.

Para el caso de Eclipse veremos como se integra a JBoss a través de JBoss Eclipse. Describiremos algunas de las funcionalidades anteriormente mencionadas sobre esta herramienta:

- Asistentes para proyectos EJB 3.0 y facilidades para el desarrollo J2EE.

  Estos asistentes ayudan en el proceso de desarrollo de una aplicación a crear los distintos componentes que deben intervenir para cumplir con los requisitos de J2EE. En primer lugar es posible dar de alta un nuevo proyecto indicándole que es de tipo, por ejemplo, J2EE 1.4. Si lo hacemos a través del menú que provee JBoss-IDE, se incorporan automáticamente las librerías necesarias para esto. Estas son: Librerias J2EE (provistas por JBoss-IDE), librerías de Web Services y las del JRE configurado. Luego a medida que se desarrolla el proyecto los distintos componentes pueden ser dados de alta, tales como los descriptores de despliegue, los diferentes beans (entity, message driven, session) y componentes web como el HTTP Servlet.

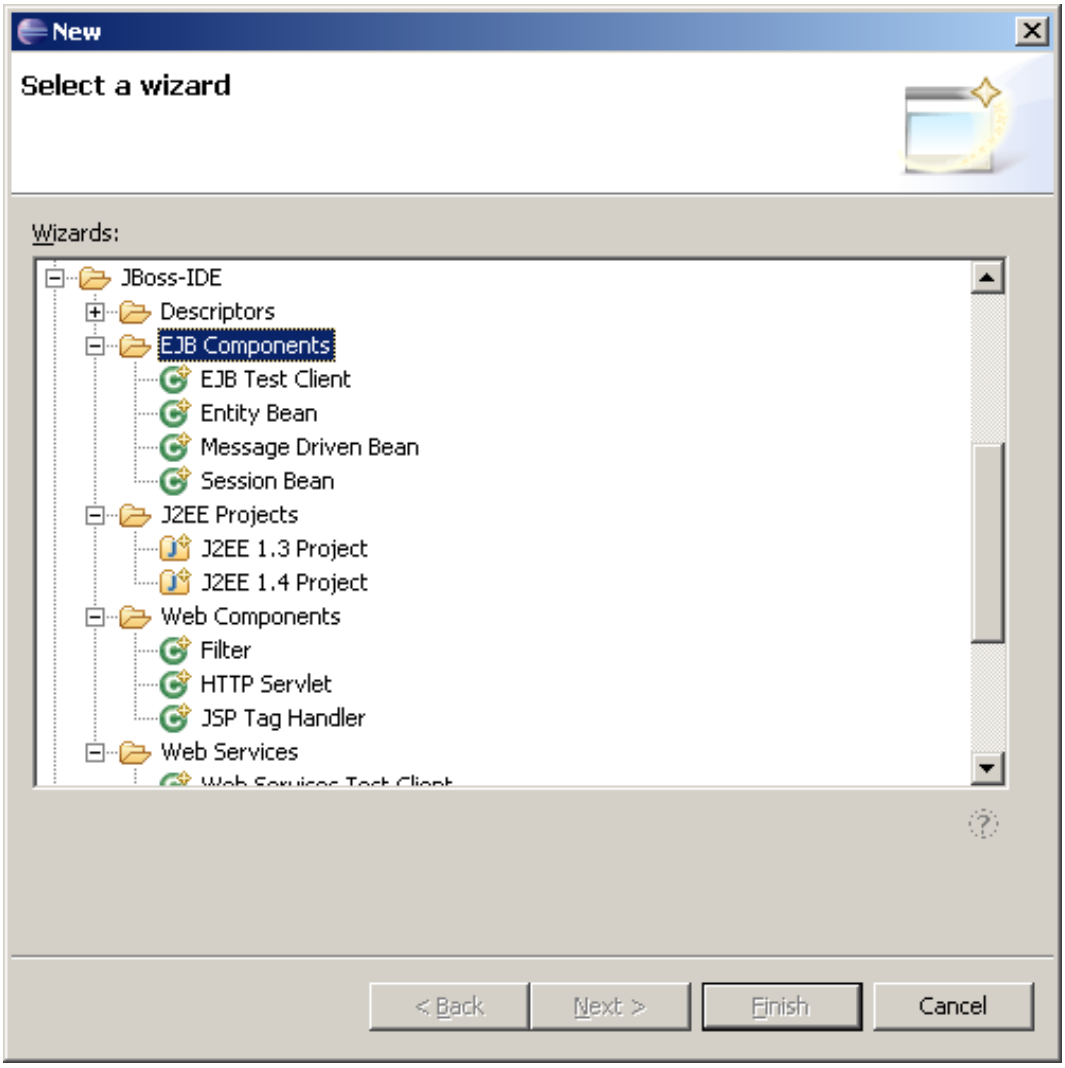


Figura 4.5.7 – Selección de Wizards de JBoss-IDE

Las herramientas de generación de código automático en JBoss-IDE están basadas en **XDoclet**. XDoclet es un motor extendido de Javadoc. Es una herramienta genérica que permite crear tags de Javadoc y a partir de estos generar código fuente o nuevos archivos (como descriptores XML) usando templates que provee el motor **[XDoclet]**. Es por eso que se hace necesario configurar esta herramienta para poder indicarle los diferentes tags que generarán automáticamente el código. Para esto deben editarse en las propiedades del proyecto e indicarle en "XDoclet Configurations", las diferentes características que tendrán los descriptores y tags que generemos. Cabe destacar

que esto debe hacerse por cada proyecto. Esto significa que no es posible establecerlo para todos lo proyectos que creemos. Aunque la herramienta provee estándares de configuración para Beans, EJB, Hibernate, JDO, JMX y Web, los valores deben ser revisados en el caso que no coincidan con los requeridos. **En esta versión de la herramienta (1.6) no es posible editar estos estándares**. Pongamos el caso que debemos generar los tags para los EJBs. Debemos crear cada atributo y luego asignarle un valor. Por ejemplo, dentro de la configuración generada para EJB, debemos dar de alta el Docle ejbdoclet y luego el atributo deploymentdescriptor con los diferentes valores, como destDir (destination directory), donde pondremos por ejemplo src/META-INF. Este trabajo puede resultar tedioso si debe realizarse por cada proyecto.

Por otro lado es necesario agregar los diferentes tags en los archivos que creemos y necesitemos que se generen descriptores. Por ejemplo si creamos un Session Bean. En este caso tendremos tags del estilo:

```
/**
 * @ejb.bean name="BeanEjemplo"
 * display-name="Nombre del Bean "
 * description="Descripción del Bean"
 * jndi-name="ejb/Bean"
 * type="Stateless"
 * view-type="remote"
 */
```

Estos tags indican que al ejecutar XDoclet se generará el correspondiente archivo ejb-jar.xml con estos atributos. Estos tags pueden ser configurados en las preferencias de Eclipse en la opción JBoss-IDE / XDoclet / Code Assist / Templates. Aquí de acuerdo al tipo de archivo que se genere (EJB, Servlet, etc.) se pueden configurar los tags que se incluirán en la creación de los archivos.

Una vez completados estos pasos, desde el proyecto, es posible indicarle el comando Run XDoclet y automáticamente se generar los archivos necesarios.

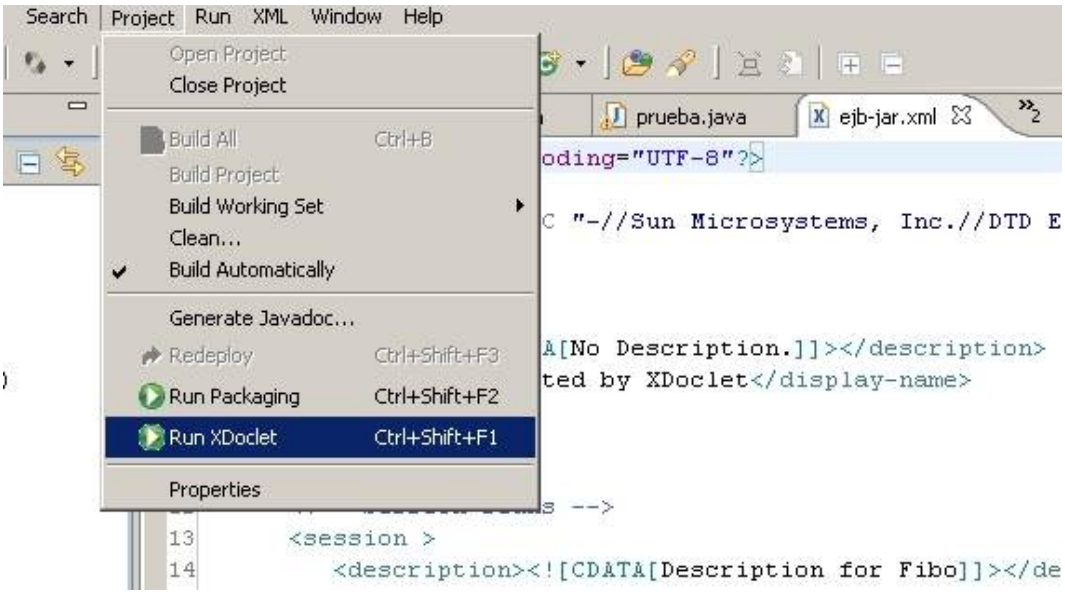


Figura 4.5.8 – Ejecución de XDoclet

Una vez generado los archivos de configuración necesarios para el despliegue es necesario empaquetarlos para publicarlos en el servidor. Este packaging se puede configurar en Eclipse …

- Herramientas de debugging, monitoreo, y control del servidor Jboss

Desde Eclipse es posible configurar los diferentes Application Servers para poder ser controlados desde el IDE y poder realizar debugs sobre la aplicación. JBoss-IDE provee las configuraciones correspondientes a las diferentes versiones de JBoss Application Server y de esta manera es posible inicializarlo. Para ello debe configurarse desde el menú de Debug. Solo debe suministrarse el directorio raíz donde el Servidor ha sido instalado e indicarle donde están los archivos fuentes en la solapa de Source.

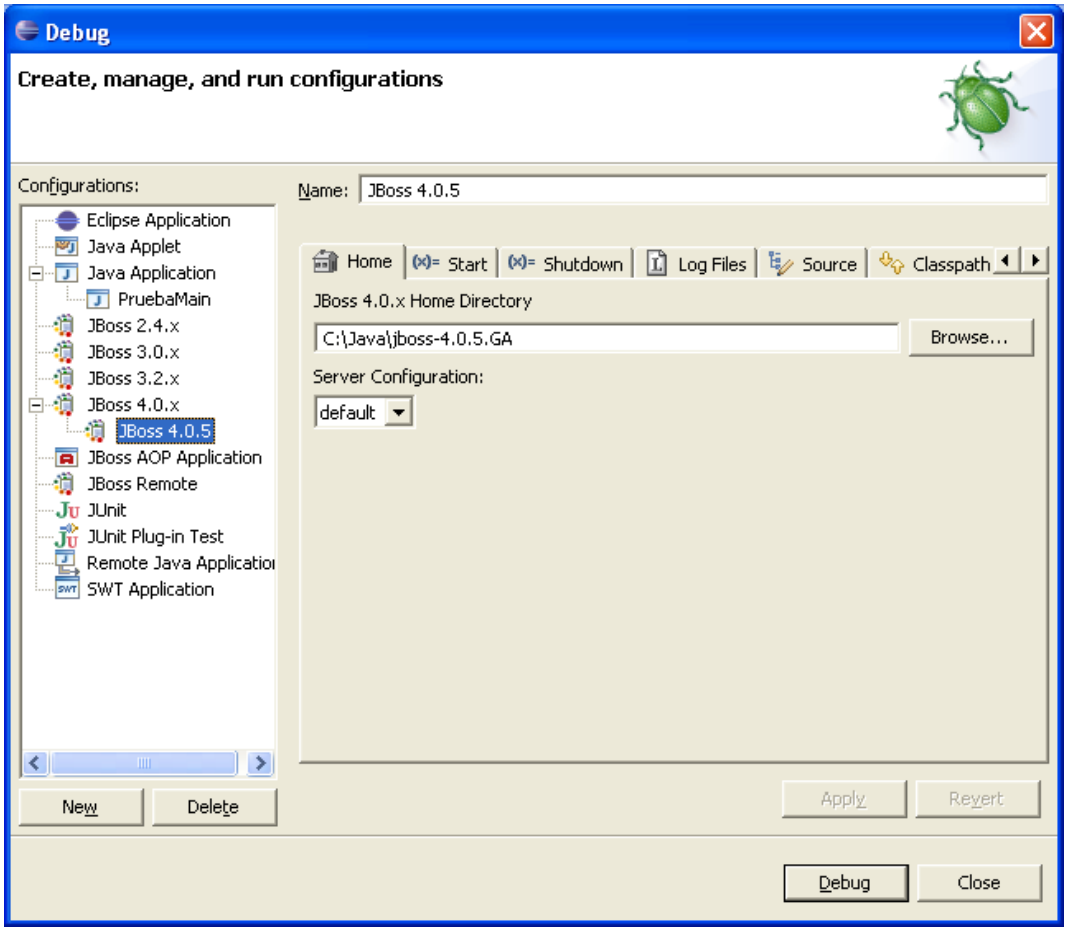


Figura 4.5.8 – Configuración del Application Server para Debug

Una vez configurado el servidor se puede inicializar presionando el botón de Debug. Esto ejecutará las acciones para levantar el servicio de JBoss. Luego podrá visualizarse la consola de JBoss desde un browser en la URL correspondiente (http://<nombre del servidor>:<puerto>).

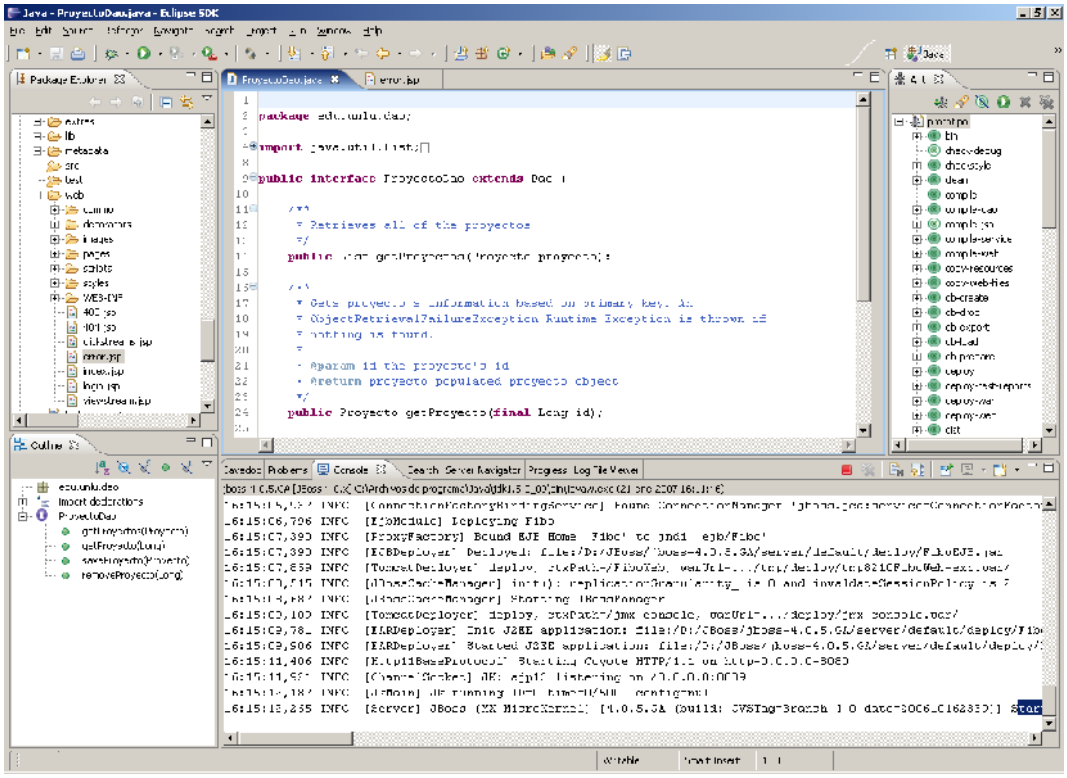


Figura 4.5.8 – Vista de la consola de Eclipse al ejecutar el Debug para JBoss

- Editores JSP, HTML, XML, CSS, y Javascript.
- Herramientas para Hibernate (ver Integración con herramientas de persistencia).

Oracle Application Server

En el caso de Oracle Application Server, solo podemos integrarlo a Eclipse a través de las herramientas WTP, provistas por Eclipse (mencionado en 3.5.12). Esto nos permite poder ejecutar el contenedor OC4J y de esta manera ejecutar Aplicaciones Web (básicamente páginas JSP) directamente del IDE. Existen algunos tutorials respecto de esta funcionalidad provista por Eclipse.org **[EclipseWTPOC4J]**.

Sin embargo, estas herramientas no proveen funcionalidades para poder desarrollar EJBs en forma asistida ni generar el deployment de la aplicación en el servidor, como en el caso de JBoss IDE.

VER Puesta en producción del prototipo.

3.5.13 Integración con herramientas de versionado

De [Gallardo03]

The need for source control

Source control systems (also called *version control systems*)

- *Locking Pesimista*
- *Locking Optimista*

Using CVS with Eclipse

Sharing a project with CVS

Creating a repository location

Sharing the project

Adding and committing files

Checking a project out of CVS

Working with CVS

Checking in to CVS

Resolving conflicts in an updated file

Synchronizing with the repository

Synchronization modes: incoming, outgoing, incoming/outgoing

Creating and applying a patch

Versions and branches

Adding a version label

Retrieving a version

Creating and using a branch

3.5.14 Extensibilidad y actualización

Descripción de incorporación plugins, módulos, adicionales, etc.

Updates de la herramienta

La arquitectura de Eclipse permite agregar funcionalidades adicionales a la plataforma.

Plug-ins and extension points

Anatomy of a plug-in

The plug-in lifecycle

3.5.15 Fucionalidades especiales

Funcionalidades adicionales que diferencia a la herramienta de otras.

3.5.16 Conclusiones

Conclusiones parciales de la herramienta.

3.6 JDEVELOPER

Oracle JDeveloper es un entorno de desarrollo integrado para la construcción de aplicaciones orientadas a servicios, usando tecnología Java y demás estándares de la industria como XML, Web Services e integración con SQL.

Uno de los puntos más importantes en JDeveloper es la idea de cubrir todo el ciclo de vida en el desarrollo de las aplicaciones. Desde el modelado de datos hasta la codificación, pruebas y puesta en marcha. Este punto se apoya fuertemente en un framework propietario de Oracle llamado Application Development Framework, o nombrado con su acrónimo ADF. A través de este framework integra tareas de manera "amigable" a través de wizards intentando simplificar el proceso de desarrollo y las tareas de codificación manual, o sea focalizado en un desarrollo visual.

3.6.1 Descripción general

Al igual que Eclipse, JDeveloper es una herramienta desarrollada en lenguaje Java, por lo tanto, potencialmente puede ser ejecutado en cualquier plataforma que soporte, en este caso, JDK 5.0. Como también soporta sistemas operativos que soporten Sun J2SE 1.5.0_05. Sin embargo no es de código abierto (open-source), como el caso de Eclipse, por lo tanto no es posible compilarlo para una plataforma no soportada. Los sistemas operativos soportados son: Windows 2000-Service Pack 4, Windows NT-Service Pack 6a, Windows XP-Service Pack 2, Red Hat Enterprise Linux 3.0 y Apple Mac OS X Version 10.4.x. **[JDevInstGuide]**.

Respecto de la arquitectura J2EE, JDeveloper se basa en el patrón de diseño MVC. De esta manera trata de dar solución al desarrollo de aplicaciones que combinan lógica distribuida y desarrollo web con interfaces de usuario más complejas. Esta no es una característica exclusiva de JDeveloper, ya que otras herramientas dan posibilidades para este tipo de desarrollo, pero es importante señalar que las estructuras funcionales que propone están fuertemente basadas en este concepto. Eso le suma cierta potencia y productividad, pero a la vez puede quitarle flexibilidad al momento de decidirse por algún otro paradigma. Las tecnologías que la herramienta incorpora como sus pilares

son tecnologías ya aceptadas en el mercado y probadas (por ejemplo Struts). Sin embargo existe una lista limitada de las mismas y el desarrollo debe realizarse dentro de estos paradigmas o frameworks propuestos.

JDeveloper pone su esfuerzo más importante en minimizar el esfuerzo de integración, eligiendo ciertos paradigmas establecidos y acompañar el ciclo de vida con herramientas que permitan escribir código automáticamente a través de diferentes las fases del desarrollo y de las distintas capas de las aplicaciones.

JDeveloper incluye, en su versión completa, un framework integrado con J2EE llamado Oracle ADF u Oracle Application Development Framework. Este framework provee un modelo para adaptar los distintos servicios a través de los cuales pueden ser creadas las distintas vistas independientes de su fuente, controlar transacciones e interactuar con la base de datos. De esta manera soporta el modelo MVC y genera a través los wizards una interacción más amigable con el desarrollador.

Organización que la desarrolla

Arquitectura / Características sobresalientes

Lenguajes soportados

Hacia donde va en la actualidad

Soporte sobre la herramienta

Releases y actualizaciones

3.6.2 Instalación

JDeveloper no requiere instalador. De manera similar a Eclipse la herramienta se ejecuta una vez que se ha descomprimido el archivo con extensión ZIP, provisto para la instalación. Se debe evitar copiar el producto en una carpeta perteneciente a una instalación previa de otra herramienta de Oracle. Esto no se recomienda ya que luego es posible que no se puedan desintalar los productos a través del instalador universal de Oracle o puedan sufrir incompatibilidades en su funcionamiento. Pero no hay problemas si se instala en una nueva carpeta, por ejemplo C:\JDeveloper.

Existen varias versiones para bajar desde Oracle.com. Existe una primera versión básica llamada **Java Edition** que inlcuye las herramientas necesarias para programar en dicho lenguaje y funciones para trabajar con XML, pero sin incluir funcionalidades de UML o de J2EE. Estas son provistas en una segunda versión llamada **J2EE Edition**. La versión completa, que incluye todas las funcionalidades provistas por JDeveloper, es la **Studio Edition**. Incluye herramientas para desarrollo Java y facilidades para J2EE, integración UML y SOA. Esta versión también contiene el framework ADF. La versión completa comprende además todas las herramientas necesarias para la compilación y la ejecución Java sin que sea necesaria la configuración desde el sistema operativo. Las otras instalaciones requieren de l configuración de los paths para el JDK y demás herramientas que se requieran (por ejemplo, herramientas de profiling).

3.6.3 Organización de la herramienta

Al entrar en la herramienta se presentan por defectos las vistas de Aplicaciones y Conexiones arriba a la derecha y debajo la de Estructura donde se muestra generalmente un detalle de estructura o de componentes de los seleccionados en la vista de Aplicaciones. En la parte central aparece el panel en el que los componentes son abiertos como clases, páginas JSP o mapas de configuración de Struts u otros tipos XML. Tanto en este panel como en los demás se pueden presentar varias pestañas cuando existe más de un componente abierto. En este particularmente, según el tipo de archivo que se presente se muestran solapas inferiores donde muestran, por ejemplo la vista de Diseño si corresponde, el código y la historia de cambios del archivo. De esta manera es posible trabajar por ejemplo en el diseño de una página JSP cambiando de lugar una etiqueta y ver reflejado el cambio automáticamente en el código de la misma. Sobre la derecha se presentan la vista de Componentes y la de Propiedades. Estos componentes pueden ser arrastrados hacia el panel central e incorporados al código de la forma “drag & drop” (arrastrar y soltar). Luego pueden ser inspeccionadas las propiedades de los mismos haciendo click sobre el mismo componente o a través de la vista de propiedades (Property Inspector) seleccionándolo.

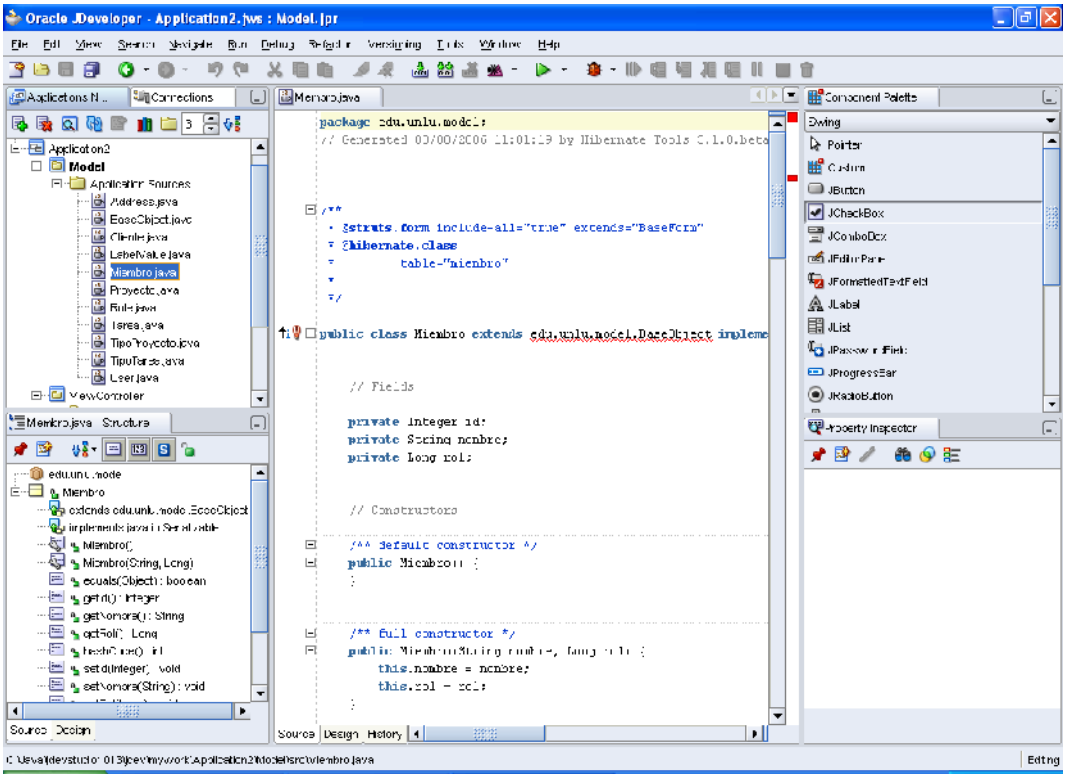


Figura 4.5.7 – Disposición de ventanas en JDeveloper

Arquitectura de la herramienta

Disposición de ventanas y vistas o perspectivas.

3.6.4 Codificación

Manejo de proyectos

Crear un proyecto Java

Al igual que en Eclipse es necesario crear un proyecto para empezar a trabajar. Pero estos son generados dentro del concepto de Aplicación. Por lo tanto es necesario primero crear una aplicación y luego un proyecto dentro de ésta. Así es posible tener varios proyectos por aplicación. La aplicación puede ser preconfigurada con algún template (por ejemplo Web Application [JSF, EJB] o Java Application [Java, Swing]).

Esto dará una estructura determinada que agrupará por ejemplo una carpeta para View-Controller y otra para Data Model. De esta manera se podrá decir qué tecnologías quedan comprendidas en cada carpeta. Por ejemplo, se puede decir que en una aplicación del tipo [JSF, EJB] existen 2 carpetas. Una llamada View and Controller, que contendrá los archivos de tipo HTML, Java, JSF y Servlets, y otra llamada Data Model que contendrá los EJB y código Java. Si bien es posible crear nuevos templates las carpetas solo pueden contener componentes de las tecnologías disponibles para JDeveloper. Estas son: ADF Business Components, ADF Swing, Ant, Database, EJB, HTML, Integration (para BPEL y SOA), JavaBeans, JSF, JSP, Mobile, Oracle JSP tag Libraries, Struts, TopLink, UML, Web Services, XML y Documentos XSQL.

Una vez creada la aplicación se debe crear el proyecto sobre el cual trabajar. Existen tipos de proyectos Generales (Aplicaciones, Conexiones, Proyectos [Vacío, BPEL Process, Business components, Java Application, TopLink Project y Web Project]. Luego también es posible importar desde archivos fuentes y de archivos WAR.

Si bien soporta potencialmente, a través de plug-ins, una variedad importante de tipos de proyectos (EJB, Web, etc.), soporta básicamente 3 tipos:

Plug-in Development. El cual provee el entorno para desarrollar los propios plug-ins para Eclipse.

Simple. Provee un entorno genérico, por ejemplo para crear documentación.

Java (u otro lenguaje). Esta opción configura el entorno de desarrollo específico para el tópico elegido disponiendo de funcionalidades y configuraciones específicas para el lenguaje.

Al crear un proyecto se crea una carpeta en el directorio seleccionado como workspace, donde serán guardados los archivos generados dentro del proyecto. Si se desea eliminar el proyecto, Eclipse dará la opción de mantener esta carpeta, eliminando solo lógicamente el proyecto, o eliminándola junto con el mismo.

Importar un proyecto externo

Hay varias maneras de importar un proyecto externo, por ejemplo realizado en otro entorno o IDE. Una forma es copiar el código fuente en una carpeta dentro del workspace en un proyecto existente o creando una nueva carpeta.

Otra forma es creando una carpeta en el sistema de archivos y luego crear un proyecto indicando el nombre de la carpeta. Eclipse notará que la carpeta existe y dará la opción para crear el proyecto en la misma. Luego buscará los archivos fuentes y las librerías existentes y los tomará por defecto. Adicionalmente se le pueden indicar otras carpetas fuentes.

(AMPLIAR ESTOS CONCEPTOS – Poner como ejemplo la importación de un proyecto proveniente de JDeveloper o Netbeans)

Manejo de Proyectos.

Ayuda en línea.

Creación de código automáticamente.

Refactoring.

3.6.5 Facilidades de uso (Web Development, Deploy, etc.)

JDeveloper integra un conjunto de tecnologías y frameworks para la creación de los distintos proyectos. Para cada etapa del desarrollo o capa del mismo ofrece diversas opciones. Las mismas son:

Para la capa de Modelo:

- Oracle ADF Business Components
- TopLink (ORM)
- JavaBeans
- Enterprise JavaBeans
- Web Services

Para la capa de Controlador:

- JSF (Java Server Faces) controller
- Struts servlet controller
- Swing toolkit (para aplicaciones Java)

Para la capa de Vista:

- Applet
- HTML
- JSF
- JSP
- Oracle ADF Faces (componentes basados en JSF)
- Oracle ADF Swing (para aplicaciones Java)

Tecnologías involucradas para la realización de las distintas etapas de desarrollo.

Uso de frameworks

3.6.6 Debugging

Descripción de las herramientas de debugging

3.6.7 Profiling

JDeveloper cuenta con una máquina virtual de Java especializada para el uso de profilers, asistentes de código y deburggers. Esta es llamada OJVM (Oracle Java Virtual Machine) y requiere ser instalada, si no se instaló la versión completa de JDeveloper.

OVJM incrementa está optimizada para incrementar la velocidad en el debugger y proveer funcionalidades de detección de “deadlocks” o debugging en memoria.

3.6.8 Logging

Descripción de las herramientas de debugging

3.6.9 Extensibilidad y actualización

Descripción de incorporación plugins, módulos, adicionales, etc.

Updates de la herramienta

3.6.10 Integración con RDBMS

Para agregar un conexión a la base de datos, es necesario ir a la solapa Connections y luego a la carpeta Databases. Luego se debe elegir del menú contextual (se activa con el botón derecho del mouse) la opción New Database Connection. A partir de este punto es posible agregar conecciones para Oracle a través de JDBC (la herramienta ya provee los drivers para las versiones 10g y 9i (driver thin) y anteriores como 8i (oci)), conexiones ODBC, Oracle Lite (versión liviana de Oracle) o drivers JDBC de terceras partes.

Una vez realizada la conexión se puede testear y consultar sus objetos. En el menú contextual aparece la opción de SQL Worksheet. De esta manera se despliegan varias herramientas para consultar la estructura y datos de la base. En el caso que la base sea Oracle, también se habilita la posibilidad de ejecutar la aplicación externa SQL*Plus (herramienta de Oracle para ejecución de código SQL).

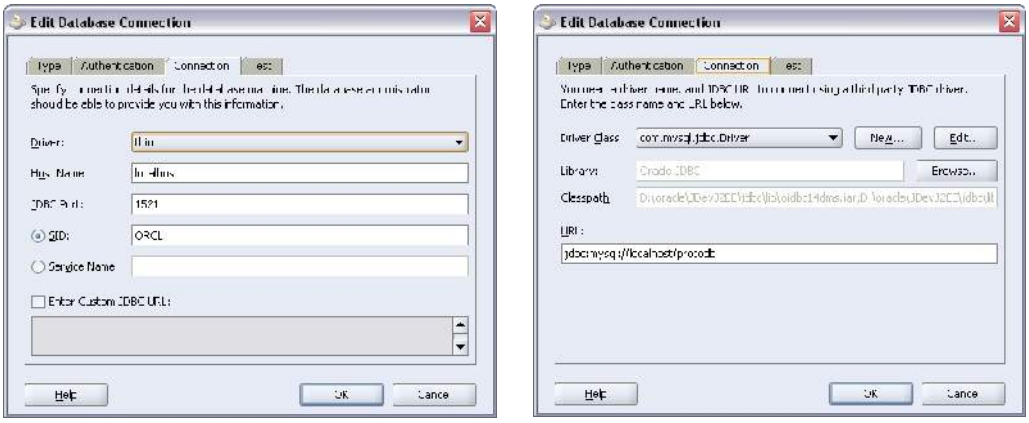


Figura 4.6.1 – Configuración de conexiones

Izquierda – Conexión Oracle, Derecha – Conexión JDBC terceras partes

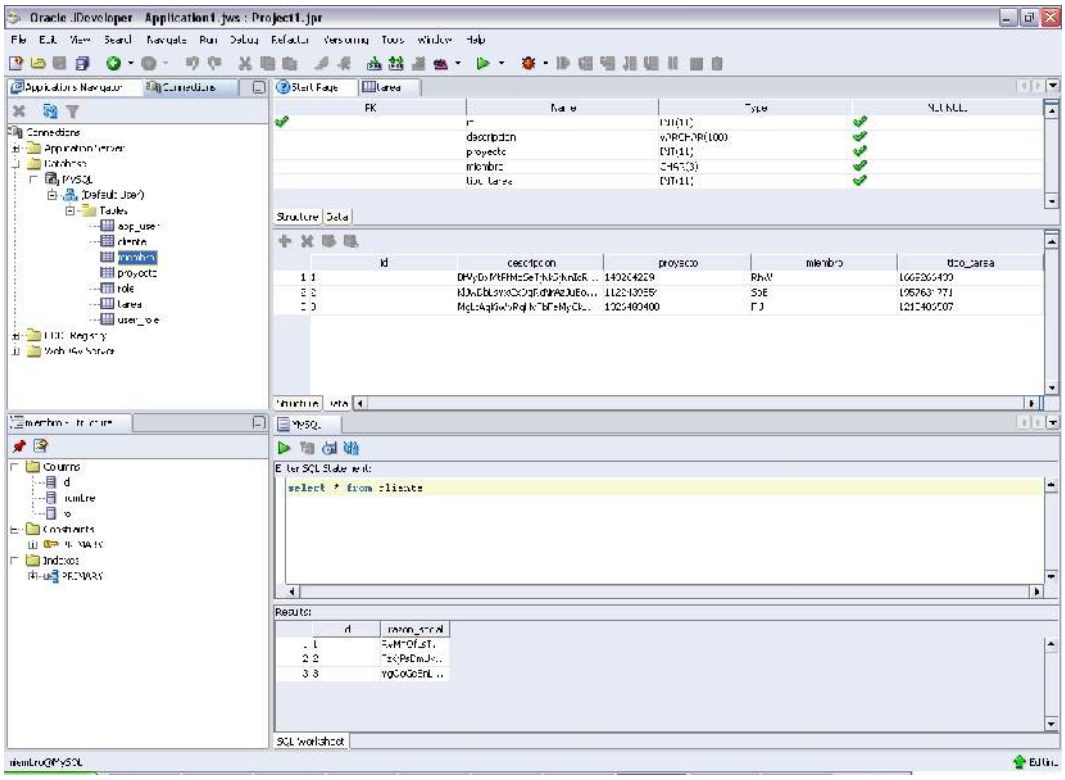


Figura 4.6.2 – Vistas de conexiones y SQL Worksheet desplegado

Integración con las bases de datos elegisdas: Oracle, MySQL y SQLServer.

3.6.11 Integración con herramientas de UML

Generación de diagramas de UML con la herramienta e interacción con el código. Ingeniería reversa e integración.

3.6.12 Integración con Application Servers

Es posible integrar dentro de JDeveloper varios Application Servers. El release 10.1.3.1 soporta los siguientes: Oracle Application Server, Standalone OC4J, Tomcat, JBoss, WebLogic y WebSphere **[JDevAppSrv]**.

Para integrar un Application Server a la herramienta son necesarios los siguientes pasos:

Dentro de la solapa "Connections", ir a la carpeta Application Server. Desde aquí, con el botón derecho del mouse, se puede desplegar el menú contextual y luego indicarle "New Application Server Connection". Esto nos habilitará la lista de opciones con los Application Servers soportados.

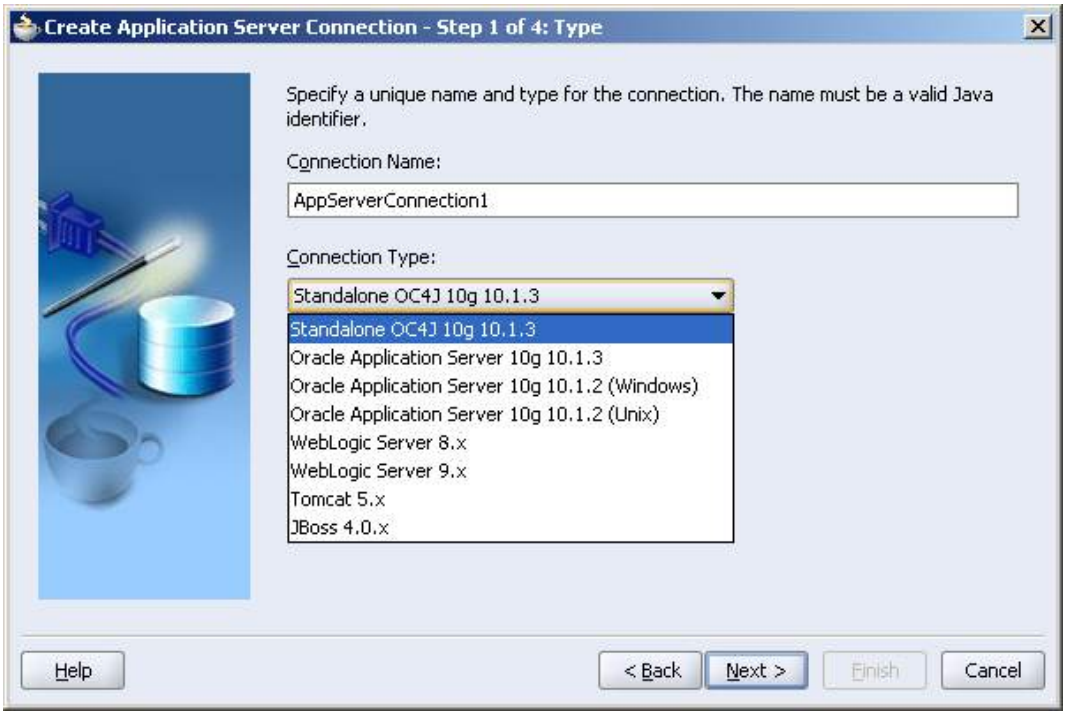


Figura 4.6.2 – REVISAR

En el caso que el Application elegido fuera OC4J u Oracle Application Server, el asistente pedirá el usuario administrador y su password, luego el nombre o dirección del servidor y el puerto RMI. Esta información es para componer la URL correspondiente para crear una conexión JMX con el servidor. Luego se dará la opción de testear la conexión.

Una vez configurada la conexión al Servidor de aplicaciones, necesitaremos realizar el deployment de la aplicación. Para esto se hace necesario armar los paquetes correspondientes para poder realizar el deployment en el Servidor de Aplicaciones. Para esto, JDeveloper permite la creación de perfiles de deployment (Deployment Profiles).

Los perfiles de deployment son componentes del proyecto para manejar el despliegue de una aplicación en el servidor. Estos componentes incluirán el código de la aplicación, los descriptores que sean necesarios y otros archivos auxiliares que deben incluirse en el paquete de deployment.

Existen 3 partes en el paquete de deployment en la aplicación prototipo, según la configuración que propone JDeveloper. Por un lado el proyecto del modelo (que se incluirá en un archivo con extensión .jar), el proyecto de vista-controlador (incluido en un archivo .war) y la aplicación completa (integrada en un archivo .ear).

Para crear el primer “deployment profile” (JAR), debemos ubicarnos con el mouse en la carpeta Model. Desde allí, con el botón derecho, seleccionar la opción New. En la nueva galería de opciones deberemos seleccionar General > Deployment Profiles en la lista de Categorías, y luego seleccionar la opción “JAR File” en la lista de ítems. Luego, deberán introducirse los valores para el nombre del archivo y algunas opciones adicionales que propone el asistente. Esto generará en la carpeta de Model > Resources un archivo de deploy que luego la herramienta utilizará para generar el despliegue en el Application Server.

De igual manera a la generación del archivo .JAR se puede construir el archivo .WAR, desde el proyecto ViewController. Luego podremos generar el

archivo .EAR para completar el proceso de empaquetamiento de la aplicación. Para este último paso crearemos un nuevo proyecto dentro de la aplicación (que por ejemplo llamaremos Deployment) que contendrá los recursos para generar el archivo .EAR. Esto es simplemente para que quede ordenado. Luego, posicionado en este proyecto, daremos el comando New en el menú contextual (al igual que en los casos anteriores) y podremos a través del asistente configurar el archivo de despliegue. Esto nos permitirá luego poder realizar el deployment en el Servidor de Aplicaciones seleccionando el nuevo archivo de recursos generado (Ver gráfico).

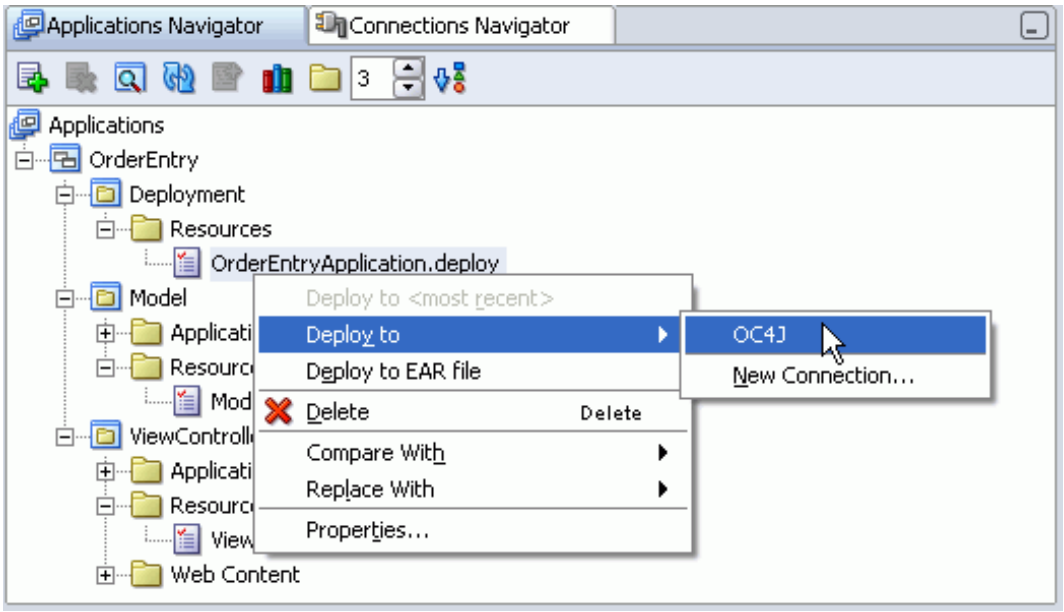


Figura 4.6.2 – REVISAR

Integración framework Maven.

3.6.13 Integración con herramientas de versionado

Integración con las herramientas CVS, Subversion.

3.6.14 Funcionalidades especiales

Funcionalidades adicionales que diferencia a la herramienta de otras.

3.6.15 Conclusiones

Conclusiones parciales de la herramienta.

3.7 NETBEANS

Descricpción de la herramienta en forma general.

3.7.1 Descripción general

Organización que la desarrolla

Arquitectura / Características sobresalientes

Lenguajes soportados

Hacia donde va en la actualidad

Soporte sobre la herramienta

Releases y actualizaciones

3.7.2 Instalación

Como se instala?

3.7.3 Organización de la herramienta

Arquitectura de la herramienta

Disposición de ventanas y vistas o perspectivas.

3.7.4 Codificación

Manejo de Proyectos.

Ayuda en línea.

Creación de código automáticamente.

Refactoring.

3.7.5 Facilidades de uso (Web Development, Deploy, etc.)

Tecnologías involucradas para la realización de las distintas etapas de desarrollo.

Uso de frameworks

3.7.6 Debugging

Descripción de las herramientas de debugging

3.7.7 Logging

Descripción de las herramientas de debugging

3.7.8 Extensibilidad y actualización

Descripción de incorporación plugins, módulos, adicionales, etc.

Updates de la herramienta

3.7.9 Integración con RDBMS

Integración con las bases de datos elegisdas: Oracle, MySQL y SQLServer.

3.7.10 Integración con herramientas de UML

Generación de diagramas de UML con la herramienta e interacción con el código. Ingeniería reversa e integración.

3.7.11 Integración con Application Servers

Descripción de procesos de integración con los Application Servers elegidos. Puesta en producción del prototipo.

3.7.12 Integración con herramientas de versionado

Integración con las herramientas CVS, Subversion.

3.7.13 Fucionalidades especiales

Funcionalidades adicionales que diferencia a la herramienta de otras.

3.7.14 Conclusiones

Conclusiones parciales de la herramienta.

4 ANÁLISIS COMPARATIVO

4.1.1 Introducción

Para poder realizar un análisis comparativo de los diferentes ambientes se propondrá un prototipo de aplicación que cumpla con los requisitos necesarios para que su implementación se deba realizar bajo una arquitectura J2EE en un entorno Web. Se presentará una empresa ficticia (**Consulting Corporation**) con un caso concreto a resolver.

4.1.2 Desarrollo del prototipo de aplicación

Consulting Corporation es una empresa que brinda servicios de consultoría en sistemas y realiza los diversos trabajos en clientes. Estos trabajos son facturados por cantidad de horas insumidas al finalizar el mes. Las horas trabajadas son informadas por cada consultor indicando el cliente, la fecha y la cantidad de horas incurridas. Luego los líderes de cada proyecto listarán los diferentes informes requeridos para facturar e informar al cliente.

El proceso de carga de horas puede resumirse en el siguiente caso:

1. El consultor informa las horas en una planilla
2. Las planillas son remitidas por correo hacia la empresa
3. La persona que recibe las planillas lo carga en una nueva planilla de informe por cliente y consultor.
4. El líder de proyecto saca informes de las horas para realizar la facturación mensual. Le remite al cliente el detalle de las horas trabajadas.

4.1.3 Problema a resolver

La carga manual en planillas de cálculo y su posterior envío y carga en el sistema de la empresa hacen que se realicen tareas manuales que pueden generar errores. Por otro lado, la información no se obtiene en tiempo real y no se puedan realizar previsiones de facturación antes de fin de mes. Estos procesos aumentan el costo operativo de la compañía y retrasan el proceso de facturación, sumado a los errores que se generan en las diferentes cargas manuales.

Por otro lado la compañía quiere brindarles un nuevo servicio a sus clientes para que puedan consultar el avance de sus proyectos y cuenten con la información on-line de las horas incurridas. Esta información le será de utilidad también a los consultores y líderes de proyecto involucrados en el trabajo.

4.1.4 Objetivos

La compañía se plantea los siguientes objetivos:

- Realizar una interfase para que los consultores puedan cargar, actualizar y chequear el status de sus horas.
- Realizar una interfase para que la compañía pueda crear, actualizar y gestionar las horas cargadas, actualizando el estado de las mismas cuando sean facturadas.
- Realizar una interfase de visualización para que los clientes puedan ver sus horas.
- Generar diferentes reportes de horas.

4.1.5 Solución de negocio

Consulting Corporation decide usar JDeveloper, Eclipse o NetBeans para crear una aplicación Java EE que permita la gestión de horas por proyectos.

Los aspectos técnicos de la aplicación incluyen:

- Las interfases de usuarios serán basadas en un browser para permitir la carga tanto de los consultores externos como internos en la empresa.
- La tecnología debe ser compatible con Java EE.
- El Application Server será Oracle Application Server 10g. ó JBoss 4.0.x
- Los componentes de datos serán objetos EJB 3.0. Enterprise JavaBeans (EJB) 3.0 provee la capa de servicios de negocios de la aplicación. Un Enterprise JavaBean es un componente Java EE reusable y portable. La especificación EJB 3.0 hace más fácil el desarrollo de Enterprise JavaBeans y los IDEs elegidos permiten el uso de asistentes o facilidades para la creación de EJBs.
- Se usará Java ServerFaces (JSF) para la interfase de usuario (UI) de la aplicación.

4.1.6 Funciones

La siguiente funcionalidad es la requerida por la aplicación:

- Log-in y validación de consultores / clientes
- Carga de horas
- Actualización de horas
- Mantenimiento histórico de la carga de horas (con la habilidad de ocultar notas de los consultores a los clientes)

… Agregar

4.1.7 Páginas

La aplicación tendrá las siguientes páginas:

- Página de Login: Cada usuario es un consultor, un cliente o un líder de proyecto.
- Insertar horas: Tanto consultores como líderes pueden cargar horas.
- Listado de horas (vista para consultores): Esta página es la que usan los consultores para ver las horas cargadas. Mostrará las horas con su estado (abiertas, facturadas, etc.). Cuando se haga click en una carga determinada debe direccionarse a la página de edición de horas cargadas (si están abiertas), dando la posibilidad de dar de alta otras (página de Carga de Horas).
- Edición de horas (vista para consultores). En esta página los consultores pueden crear o actualizar la carga de horas. La historia de carga puede contener información sensible para la compañía por lo cual se requiere que solo lo vean los líderes de proyecto.
- Edición de horas (vista para consultores). En esta página los consultores pueden crear o actualizar la carga de horas.

... Agregar

4.1.8 Flujo entre páginas

Será la siguiente:

1. Los usuarios invocan a la aplicación y visualizan una página de Bienvenida, en la cual tendrán la opción de log-in.
2. Los usuarios pueden ser consultores o clientes. Si los datos de log-in son correctos navegarán en las páginas según su perfil.
3. Si el usuario es un consultor, la página de Bienvenida lo llevará a las opciones para ver las horas cargadas o cargar nuevas.
4. Si el usuario es un cliente, la página de Bienvenida lo llevará a las ver las horas cargadas a ese cliente únicamente.

5. Si el usuario es un lider de proyecto, además de ver las horas cargadas, podrá actualizar el estado de las mismas, poniéndolas en facturadas o anuladas, filtrando por consultor, cliente o fecha.
6. Si el usuario elige crear horas, navegará a la página de carga de detalle de horas.
7. Si el usuario elige ver las horas cargadas, navegará a la página que lista las horas cargadas. Esta lista contiene links para permitir a visualización o edición de los detalles.
8. Si el usuario elige actualizar, navegará a una página que permita modificar los detalles de las horas cargadas.
9. Si el usuario elige consultar, navegará a una página donde se pueda ingresar un criterio de búsqueda y retorne la lista de registros correspondiente

4.1.9 Tablas

Cliente: En esta tabla se cargarán los clientes de la empresa.

Proyecto: En esta tabla se identificarán los diferentes proyectos de cada cliente. Pueden existir varios proyectos para un mismo cliente.

Tipo_proyecto: Codificación del tipo de proyectos que pueden cargarse.

Tarea: En esta tabla se cargarán las tareas realizadas por cada proyecto y consultor (miembro). Cada tarea podrá pertenecer solo a un miembro. Pueden existir varias tareas en un mismo día, para distintos proyectos o de distintos tipos de tareas.
Por ejemplo: Análisis 6 horas, Programación 2 hs.

Tipo_tarea: Codificación del tipo de tareas que puede realizar un consultor.

Miembro: En esta tabla se cargarán tanto consultores como líderes de proyecto.

Usuario: En esta tabla se

Rol: En esta tabla se

Rol_x_usuario:

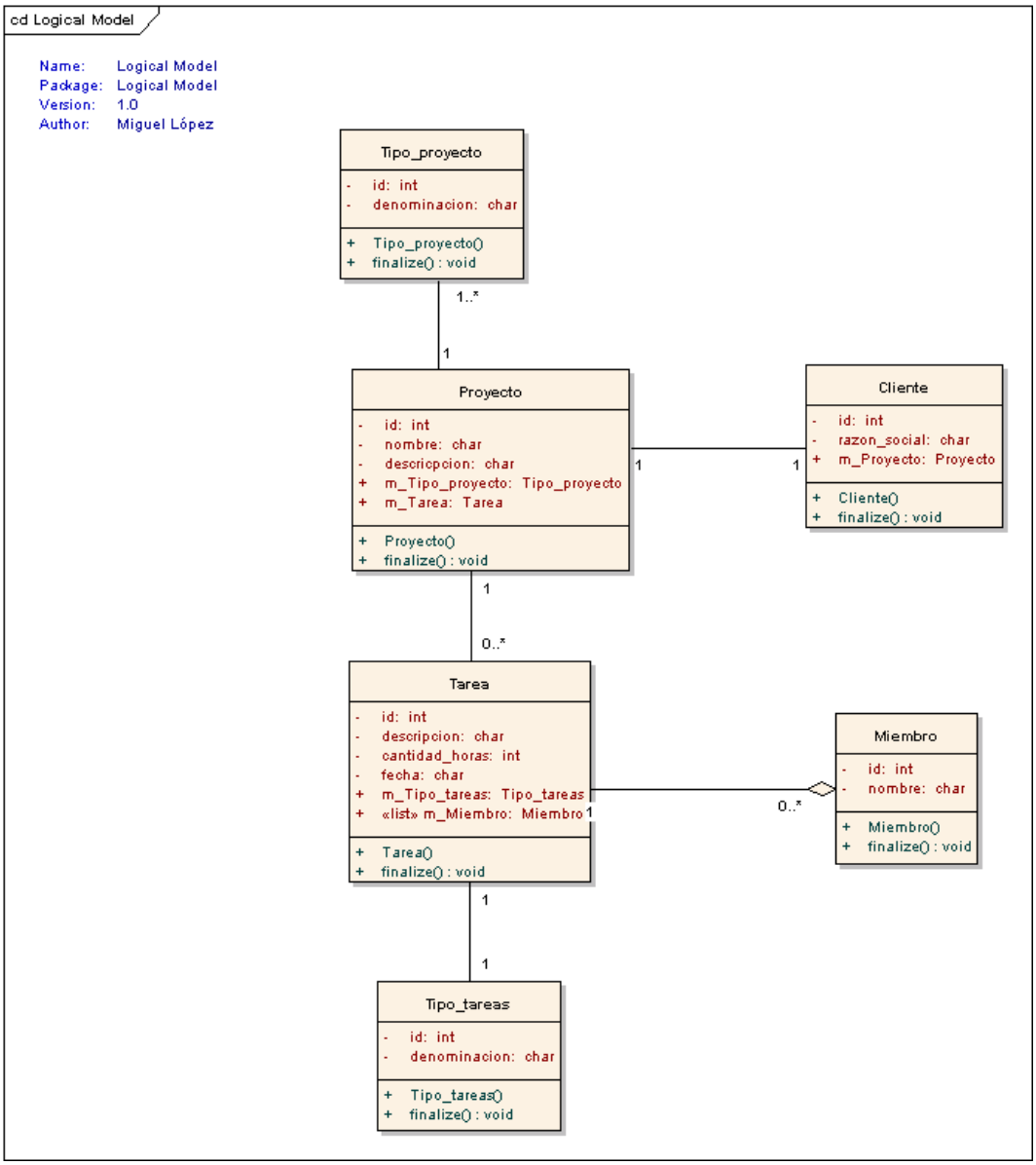


Figura 4.1 – Modelo de datos del Prototipo (1ra. Versión – revisar)

4.1.10 Adaptaciones para las distintas herramientas

Completar. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

4.2 Setup inicial y ejecución

Completar. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

4.3 Facilidades de revisión y modificación (Web Development, Deploy, etc.)

Completar. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

4.4 Mediciones

Completar. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

4.4.1 Medición 1

Completar. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

4.4.2 Medición 2

Completar. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

4.4.3 Conclusiones

Completar. XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXX

5 CONCLUSIONES

{Acá especificar las distintas conclusiones en cuanto a ventajas y desventajas asociadas a la utilización de una u otra herramienta en el desarrollo de Software para sistemas de gestión.

También intentar generar una visión sobre 'futuras' tendencias y/o herramientas}

6 BIBLIOGRAFÍA

Durante el desarrollo de este trabajo, se consultó el siguiente material:

[Adamson05] Chris Adamson, Results from the Second 2004 ONJava Reader Survey, OnJava.com, posteado el 1 de Mayo del 2005. http://www.onjava.com/pub/a/onjava/2005/01/05/2004-survey-2-results.html

[Ahmed01] Khawar Zaman Ahmed y Cary E. Umrysh, Developing Enterprise Java Applications with J2EE and UML. Addison Wesley Professional, 17 de Octubre del 2001. ISBN-10: 0-201-73829-5. 368 páginas.

[Armstrong05] Eric Armstrong; et al. The J2EE 1.4 Tutorial For Sun Java System Application Server Platform Edition 8.2, 5 de diciembre de 2005, http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

[BitácoraS18] Miguel López, Bitácora de sesiones. Sesión 18. http://ambientes-desarrollo-modernos.blogspot.com/2006/08/sesin-18-15-17082006.html

[Burnette05] Ed Burnette. Eclipse IDE Pocket Guide. O'Reilly, Agosto 2005. ISBN: 0-596-10065-5, 127 páginas.

[Cade02] Mark Cade y Simon Roberts, Sun Certified Enterprise Architect for J2EE Technology Study Guide. Sun Microsystem Press, marzo 2002.

[CoderSource] Model View Controller Pattern, http://www.codersource.net/aspnet_model_view_controller_sample.html, online; recuperado el 16 de octubre de 2006.

[Dai06] Naci Dai, Lawrence Mandel y Arthur Ryman. Java Web Application Development with Eclipse, 1ra. edición. Addison Wesley Professional, 20 de octubre de 2006. ISBN: 0-321-39685-5, 400 páginas.

[Daum04] Berthold Daum. Professional Eclipse 3 for Java Developers. Wiley, 2004. Traducción al inglés por John Wiley & Sons Ltd., Inglaterra. ISBN: 0-470-02005-9

[Deacon05] Model-View-Controller (MVC) Architecture, http://www.jdl.co.uk/briefings/index.html - mvc

Online; recuperado el 16 de octubre de 2006.

[Donaldson00] Scott E. Donaldson y Stanley G. Siegel, Successful Software Development 2nd Edition. Prentice Hall PTR, Segunda edición 27 de diciembre del 2000. ISBN: 0-13-086826-4, 784 páginas.

[EclipseDOC] Documentación de Eclipse, http://www.eclipse.org/documentation/

[EclipseHLP] Help de Eclipse SDK.

[EclipseLANG] Language Packs: 3.2, publicado el 12 de Julio de 2006, recuperado el 21 de octubre del 2006. http://download.eclipse.org/eclipse/downloads/drops/L-3.2_Language_Packs-200607121700/index.php

[EclipseSQLExp] Eclipse SQL Explorer Web Site, http://www.sqlexplorer.org/

[EclipseWS] Eclipse Web Site, http://www.eclipse.org/

[EclipseWTPOC4J] WTP Tutorials - Deploy Web Applications to the Oracle Application Server, publicado el 27 de Octubre del 2005, recuperado el 28 de Enero del 2007. http://www.eclipse.org/webtools/community/tutorials/OracleServerAdapter/OracleServerAdapter.html

[CATALOGAR] Eclipse overview - http://www.eclipse.org/whitepapers/eclipse-overview.pdf

[Galdo06] Alberto Rodríguez Galdo, Java en Castellano Web Site, Introducción a JMS (Java Message Service). Recuperado el 29 de octubre del 2006. http://www.programacion.com/java/articulo/jms/

[Gallardo03] David Gallardo, Ed Brunette y Robert McGovern, Eclipse in Action: A guide for Java Developers, Manning Publications Co., 2003. ISBN 1-930110-96-0.

[Gould00] Steven Gould, "Develop N-tier Applications Using J2EE. An introduction to the Java 2 Platform, Enterprise Edition specification by way of BEA's WebLogic Server ".

JavaWorld.com, 1 de diciembre del 2000. is a primer on J2EE technologies . http://www.javaworld.com/javaworld/jw-12-2000/jw-1201-weblogic.html

[J2EEOverview] Sun Developer Network Web Site, Java 2 Platform, Enterprise Edition (J2EE) Overview. http://java.sun.com/j2ee/overview.html

[J2EETutorial] http://java.sun.com/j2ee/1.4/docs/tutorial/doc/J2EETutorial.pdf

[JBossOrg] JBoss Web Site, http://www.jboss.org

[JBossIDE] JBoss IDE Web Site, http://www.jboss.org/products/jbosside

[JBossNBIDE] JBoss NetBeans IDE Web Site, http://www.jboss.org/products/jbossnetbeans

[JDevInstGuide] Oracle JDeveloper 10g Release 3 (10.1.3) Installation Guide, Studio Version 10.1.3, January 2006. Recuperado el 03 de noviembre de 2006 http://www.oracle.com/technology/documentation/jdev/1013install/install.html

[JDevAppSrv] Application Servers Supported by JDeveloper, Octubre de 2006. Recuperado el 04 de noviembre de 2006. http://www.oracle.com/technology/products/jdev/collateral/papers/10g/as_supportmatrix.html

[Johnson02] Rod Johnson, Expert one-on-one J2EE Design and Development. Wrox Press Ltd., Primera Edición Octubre 2002. ISBN 1-86100-784-1.

[JohnsonJW01] Mark Johnson. “A walking tour of J2EE: What makes the J2EE platform?”. JavaWorld.com, 27 de julio del 2001 - http://www.javaworld.com/javaworld/jw-07-2001/jw-0727-enterprisejava.html?page=1

[Manageability] Manageability.org, What is your favorite Java IDE?. Recuperado el 16 de octubre del 2006. http://www.manageability.org/polls/what-is-your-favorite-java-ide

[McConnell96] Steve McConnell, Rapid Development: Taming Wild Software Schedules. Microsoft Press, 1996. ISBN 1-55615-900-5

[MSDNMVC] Model-View-Controller, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpatterns/html/DesMVC.asp Online; recuperado el 16 de octubre de 2006.

[Neat05] Adam Neat, Java jams: five IDEs tested. Posteado el 19 de enero del 2005. http://www.builderau.com.au/program/java/soa/Java_jams_five_IDEs_tested/0,339024620,339174040,00.htm

[SunJNDI] Sun Developer Network Web Site, Java Naming and Directory Interface (JNDI). http://java.sun.com/products/jndi/.

[SunRMI] Sun Developer Network Web Site, Java RMI over IIOP. http://java.sun.com/products/rmi-iiop/

[Tezaa06] Tezaa.com, Your favourite IDE. Recuperado el 16 de octubre del 2006. http://www.tezaa.com/view/Your_favourite_IDE

[QAS03] QA-Systems.com, Java IDE Survey. http://www.qa-systems.com/products/qstudioforjava/ide_marketshare.html

[WikiJAVA] Wikipedia, "Java (programming language)", recuperado el 16 de octubre de 2006. http://en.wikipedia.org/wiki/Java_%28programming_language%29_

[WikiMVC] Model-View-Controller - Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/Model-view-controller, Online; recuperado el 16 de octubre de 2006.

[XDoclet] XDoclet Project Web Site, http://xdoclet.sourceforge.net/xdoclet/index.html

[Zef03] ZefHemel.com, Java IDE Market Share Survey. Posteado el 18 de octubre de 2003.

http://www.zefhemel.com/archives/2003/10/18/java-ide-market-share-survey

7 APÉNDICES

7.1 GLOSARIO

Término	Significado o descripción
Aplicación Web	Una aplicación desarrollada para Internet, incluyendo las que usan tecnología Java como JavaServer Pages y Servlets, como las que no usan Java como CGI y Perl.
Applet	Un componente Java que se ejecuta en una aplicación o dispositivo, usualmente un web browser, que soporta el modelo de programación applet (???).
Contenedor (Container)	Una entidad que provee manejo del ciclo de vida, seguridad, despliegue, ejecución y servicios específicos a los componentes.
Contenedor Web	Un contenedor provisto por un Application o Web Server que implementa la especificación J2EE. Este define el entorno de ejecución y los servicios para los componentes web, incluyendo concurrencia, despliegue, cilo de vida, seguridad, transacciones y otros servicios.
Despliegue (Deployment)	El proceso de instalar módulos y aplicaciones en el entorno operativo.
HTML	HyperText Markup Language. Un formato de archivo para crear documentos de hipertexto en la Web.
J2EE Application Server	Provee EJB y/o Contenedor Web para soportar el entorno de ejecución de un producto J2EE.
JSP	JavaServer Page.Un JSP es una combinación de sintaxis HTML y Java, que es ejecutado en tiempo real para crear contenido para clientes web dinamicamente. JSPs más avanzadas pueden usar templates (plantillas) y tags personalizados para poder reusarlas.
Lógica de negocio	El código que implementa la funcionalidad requerida de una aplicación.
Persistencia	Protocolo para cambiar el estado de una entidad (entity bean) entre sus variables de instancia y de almacenamiento (ej., una base de datos).
Servlet	Un programa Java que genera contenido dinámico e intereactúa con los clientes web usando un modelo de requerimiento y respuesta.
SQL	Structured Query Language. El lenguaje estandarizado de las bases de datos relacionales para definir y mantener los objetos de la base y manipular los datos dentro de los mismos.
WAP	Wireless Application Protocol
Viewers	
Wizards	Una interfase gráfica que guía al usuarioa a través de pasos con ventanas de diálogo.
