Universidad de las Ciencias Informáticas
Facultad 4

Título: “ArBaWeb Integrator Tools: Plugin de Eclipse para desarrollar proyectos web integrados con ArBaWeb”.

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autores: Liyanis Velázquez Galá.

Gueorgui Obregón Obregón.

Tutor: Ing. Iósev Pérez Rivero.

Consultante: Ing. Madelín Haro Pérez.

Mayo, 2008.

Declaración de Autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Liyanis Velázquez Galá

_____

Firma del Autor

Gueorgui Obregón Obregón

_____

Firma del Autor

Ing. Iósev Pérez Rivero

_____

Firma del Tutor

Datos de Contacto

Ing. Iósev Pérez Rivero graduado en el año 2007 en la Universidad de las Ciencias Informáticas de Ingeniería en Ciencias Informáticas. Obtuvo una certificación el 18 de marzo del 2008 por el Cuba-India Knowledge Center en Enterprise Application Development (J2EE Technologies) Level 1. Su categoría docente actual es Adiestrado. Es miembro del Grupo de Arquitectura de Software y Tecnologías de la Dirección Técnica de la Infraestructura Productiva.

Ing. Madelín Haro Pérez graduada en el 2002 de la carrera Ingeniería Informática en el ISPJAE. Ha trabajado en la Empresa de Servicios Informáticos (ESI) de Cienfuegos y a partir de enero de 2003 en la UCI. Ha sido tutora de tesis durante los 5 cursos de la UCI y cotutor o consultor. Los temas trabajados han estado relacionados con el ciclo de vida del software (análisis, base de datos, arquitectura, prototipos, requisitos,…), Gestión de Proyecto, Seguridad de sistemas informáticos y Linux. Ha participado en eventos como Fórum de Ciencia y Técnica, UCIENCIA y Universidad 2008. Opta actualmente por la categoría docente de asistente y es integrante en la maestría de Gestión de Proyecto. Se ha vinculado a los proyectos productivos en el rol de Jefe de Proyecto, Analista y Jefe de Capacitación Ha impartido cursos de postgrado a funcionarios de empresas, profesores y trabajadores de la UCI y a los Directores de los IPI. Ha impartido cursos de capacitación en dos proyectos de producción en el extranjero. Ha recibido curso de postgrado en el instituto TATA en la India recibiendo el certificado internacional de habilidades. Es asesora del Departamento Docente Central de Práctica Profesional.

AGRADECIMIENTOS

Les agradecemos a todas aquellas personas que hicieron posible la realización de este trabajo, que nos apoyaron, nos guiaron y orientaron; especialmente a nuestro tutor Iosev, a Madelín, a Luis Alberto y a Jorge Ernesto.

DEDICATORIA

A mi Dios, porque siempre me ha cuidado y me ha dado la sabiduría, la inteligencia y el amor necesario que me han permitido ser quien soy.

A mi mamá, mi papá y mi hermana porque siempre han estado a mi lado y me han guiado por el buen camino.

A Obre, por el amor y la dedicación que siempre ha tenido conmigo, porque ha estado presente siempre dándome fuerzas para seguir adelante en todo momento.

A mis tías especialmente a Paula, pero todas han aportado su granito de arena.

A mis primos son muchos pero especialmente Reni, Omarito, Cary y Jose.

A mis amigos especialmente los que encontré en esta escuela que son mis hermanos.

A todas aquellas personas que me han ayudado en esta ardua tarea y que siempre me han brindado su amor y su apoyo.

A la Revolución que llevó a cabo esta escuela en la que pude formarme como una profesional.

Liyanis.

A mis padres que siempre lo dieron todo para que sus hijos fueran hombres de bien.

A mi hermano porque ya tenemos médico en la familia.

A mi nena que siempre ha estado ahí en los momentos buenos y en los malos. Y que sin ella la vida no sería igual.

A todos mis amigos que de una forma u otra me han ayudado en mi desempeño como persona y como profesional.

A la Revolución por haberme dado la oportunidad de estudiar y de cumplir mi sueño de ser informático.

Gueorgui.

RESUMEN

Actualmente en la Universidad de las Ciencias Informáticas se están desarrollando un gran número de proyectos. Muchos de ellos poseen características comunes: basados en soluciones Web[1] sobre Java Enterprise Edition (JEE)[2], hacen uso de tecnologías libres como Spring e Hibernate y están sostenidos por una arquitectura de software de dominio específico denominada (DSSA) ArBaWeb.

Para la realización de estos proyectos, se utilizan herramientas libres y de código abierto, como el ambiente de desarrollo integrado (IDE) Eclipse y un conjunto de plugins[3] que se integran al mismo, entre los que se encuentran el Spring IDE y el Hibernate Tools[4]. Hasta el momento no se cuenta con herramientas que faciliten la integración y configuración de ArBaWeb en proyectos Web sobre JEE en Eclipse, lo que lleva a los desarrolladores a tener que realizar manualmente todas las configuraciones establecidas por ArBaWeb.

El objetivo del presente trabajo de diploma es realizar la construcción de un plugin de Eclipse para automatizar las funcionalidades de integración y configuración de ArBaWeb en un proyecto Web sobre JEE, para dar solución a la problemática anterior. Como parte del mismo se construirán componentes que permitan crear proyectos, generar todas las unidades organizacionales definidas por ArBaWeb, generar todos los elementos definidos para las diferentes capas arquitectónicas, además de elaborar un manual de usuario.

Palabras Claves: Plugin, Ambiente de desarrollo integrado, Generador de Código, Arquitectura.

[1] Web: abreviatura de World Wide Web (WWW); interfaz de comunicación en la Internet, que hace uso de enlaces de hipertexto en el interior de una misma página, o entre distintas páginas.

[2] JEE: La versión empresarial de Java después de J2EE 1.4 es llamada Java EE 5.0; destacando así los cambios significantes de los framework de peso ligero traídos en los estándares empresariales de Java.

[3] plugin :(o plug-in -en inglés "enchufar"-, también conocido como addin, add-in, addon o add-on) es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

[4] Hibernate Tools: plugin de Eclipse diseñado para trabajar en proyectos que hagan uso del framework de persistencia Hibernate.

Índice

Introducción

La Universidad de las Ciencias Informáticas (UCI) es el sueño hecho realidad de nuestro Comandante en Jefe Fidel Castro Ruz, centro de excelencia, donde se forman especialistas de alto nivel en ciencias informáticas, basados en un modelo de educación donde se vincula al docente directamente con la producción de software.

La Universidad, con el acaecer de los años, ha incrementado poco a poco su prestigio a nivel internacional, lo que le ha permitido introducirse cada vez más en el mundo del negocio de la producción de software. Actualmente en el centro docente-productivo se están desarrollando un gran número de proyectos. Muchos de ellos tienen características comunes: están basados en soluciones Web sobre Java Enterprise Edition (JEE) y cada uno de ellos debe estar sostenido por una arquitectura que ofrezca una solución tecnológica en un marco de tiempo factible. En estos momentos se cuenta con una arquitectura de software de dominio específico (por sus siglas en inglés, DSSA, véase sección 1.3) denominada ArBaWeb, que da soporte a los requerimientos antes mencionados.

Para llevar a cabo la ejecución de estos proyectos, se hace uso de herramientas libres y de código abierto como el ambiente de desarrollo integrado (por sus siglas en inglés, IDE[5]) multiplataforma Eclipse[6] y un conjunto de plugins como son el Web Tools Platform (WTP)[7] y el Hibernate Tools[8]. En la actualidad no se cuenta con herramientas que faciliten la integración y configuración de ArBaWeb en proyectos Web sobre JEE utilizando Eclipse, lo que lleva a los desarrolladores a tener que realizar de forma manual, las extensas configuraciones requeridas por ArBaWeb y provoque el aumento del tiempo de desarrollo del software.

De la situación antes planteada surge el siguiente problema:

¿Cómo automatizar las funcionalidades de integración y configuración de ArBaWeb en un proyecto Web sobre JEE?

[5] IDE: ambiente de desarrollo integrado en la sección 1.4 se desarrolla con mayor amplitud este término.

[6] Eclipse: plataforma de software de código. En el contexto que se utiliza, se ubica como IDE ya que la herramienta se utiliza con el plugin "Java Development Tooling (JDT)" integrado, que lo convierte en un IDE de java, más adelante en la sección 1.5 se aborda con mayor profundidad el termino.

[7] WTP: plugin de Eclipse utilizado para desarrollar aplicaciones sobre la Web y JEE, véase sección 1.6.

[8] Hibernate Tools: plugin de Eclipse diseñado para trabajar en proyectos que hagan uso del framework de persistencia Hibernate.

El objeto de estudio lo constituyen los plugins que permiten la integración y configuración de frameworks y la arquitectura que imponen para el desarrollo. El campo de acción queda enmarcado en un plugin de integración y configuración de ArBaWeb.

Con el propósito de dar solución al problema anteriormente planteado, se ha trazado como objetivo general de esta investigación, la implementación de un plugin para Eclipse que facilite la integración y configuración de ArBaWeb en un proyecto Web.

Para guiar el proceso de investigación se plantea a continuación la siguiente pregunta científica:

- ¿Qué procesos o áreas de ArBaWeb pueden ser automatizados para lograr una mejor integración y configuración del mismo con aplicaciones web sobre JEE?

Para cumplir con el objetivo propuesto y dar solución a la problemática planteada en esta investigación, se proponen las siguientes tareas:

- Analizar la arquitectura orientada a plugin de Eclipse y redactar un marco teórico base.

- Investigar cuáles son las mejores prácticas o patrones de diseño para el desarrollo de plugins de la plataforma Eclipse.

- Realizar un diseño del plugin.

- Implementación de un plugin para Eclipse, que sirva como base para la integración y configuración de ArBaWeb en los proyectos Web sobre JEE que permita:

    - Agregar acciones necesarias para integrar ArBaWeb con un proyecto Web en el proceso de creación del mismo.

    - Crear asistentes que permitan generar elementos de un proyecto Web con naturaleza ArBaWeb como módulos, subsistemas y clases que cumplan con los patrones especificados.

    - Manual de usuario para la utilización del plugin en Eclipse IDE.

El presente trabajo está estructurado en 3 capítulos que abarcan toda la investigación realizada:

En el capítulo I, Fundamentación Teórica, se hace alusión a los conceptos y definiciones necesarias para llevar a cabo el desarrollo de un plugin de Eclipse; también se mencionan los plugin más utilizados que permiten el proceso de integración y configuración de frameworks, así como la tecnología a utilizar para desarrollar un plugin.

En el capítulo II, Características del Sistema, se describen las características definidas por ArBaWeb que se van a automatizar; del plugin que se desarrollará, se identifican las funcionalidades que debe tener, su diseño, las principales clases que posee y su estructura.

En el capítulo III*,* Manual de Usuario, ArBaWeb Integrator Tools brinda un manual de usuario integrado al Eclipse, donde se describen todos los pasos a seguir para llevar a cabo cada una de las funcionalidades que ofrece el plugin.

Capítulo I: Fundamentación Teórica

Introducción

En el presente capítulo se dan a conocer conceptos y definiciones necesarias para llevar a cabo el desarrollo de un plugin de Eclipse. Se hace un análisis sobre las características y funcionalidades de los plugins más utilizados, tanto en la universidad como en el mundo, que facilitan el proceso de configuración e integración de frameworks. Se definen además, las tecnologías a utilizar en la construcción de un plugin.

1.1 Ambiente de desarrollo integrado

Un *ambiente de desarrollo integrado o IDE*, es un programa compuesto por un conjunto de herramientas que proveen facilidades a los programadores para agilizar el proceso de desarrollo de software.

Un *IDE* es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI), algunas veces contiene también un sistema para llevar a cabo el control de versiones. Muchos de los IDEs modernos llevan integrados además un navegador de clases, un inspector de objetos y diagramas de herencia de clases para ser usados en el desarrollo orientado a objetos.

Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación, tales como, C++, C#, Java Object Pascal, Visual Basic, entre otros.

En muchos casos se puede dedicar a un sólo lenguaje de programación, por ejemplo, Borland C++ o puede utilizarse para varios como es el caso de Eclipse.

1.2 Eclipse

En noviembre de 1998, una de las transnacionales más grandes del mundo IBM[9], otorgó la tarea a uno de sus grupos de desarrollo de software denominado Object Technology International (OTI), de crear una plataforma de desarrollo común para confeccionar sus productos, basados en el lenguaje de programación Java, uno de los populares de ese momento. Fue así como surgió el denominado proyecto Eclipse.

En sus comienzos, Eclipse no era muy conocido en la comunidad internacional, por lo que los socios de IBM se mostraron inicialmente renuentes a invertir en la plataforma. Esto llevó a que en noviembre del

[9] IBM: empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática.

año 2001, IBM decidiera adoptar una licencia de código abierto, para de esta forma incrementar la influencia de Eclipse en el mundo y acelerar su adopción. En aquel entonces, IBM junto con otras 8 empresas líderes del mercado, establecieron un consorcio para velar por el buen desarrollo del mismo. Este consorcio inicialmente lo integraron empresas como Borland, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft y Webgain. Más tarde el número de miembros se incrementó, llegando en el año 2003 a conformarlo 80 integrantes.

El 2 de febrero del 2004, el consorcio anunció la reorganización de Eclipse. Lo cual condujo a que la organización original se transformara en una corporación sin ánimo de lucro, independientemente de su fundadora original IBM, denominada Fundación Eclipse. A partir de ese momento toda la tecnología y el código fuente desarrollado por la comunidad de Eclipse, se encuentran disponibles libremente a través de la licencia pública de Eclipse o en inglés Eclipse Public License.

Hasta el momento se ha dado una introducción de cómo surgió Eclipse, pero ahora tal vez se pregunte: ¿Qué es Eclipse?

En la web oficial de ***Eclipse***, el mismo se define como "una plataforma (IDE), abierta para todo y para nada en particular". Eclipse es un *plataforma* porque no es una aplicación acabada de por sí, pero está diseñada para ser extendida indefinidamente con herramientas sofisticadas. Es un *ambiente de desarrollo integrado (IDE)* porque provee herramientas para administrar áreas de trabajo (o workspaces en inglés), para construir, lanzar y depurar aplicaciones, para compartir artefactos con un equipo y versionar el código fuente. Eclipse es *abierto* porque su diseño le permite ser extendido fácilmente por terceras partes. Es apropiado para *todo*, ya que ha sido utilizado exitosamente para construir ambientes para un amplio rango de temas, como el desarrollo en lenguaje de programación Java, Servicios Web, certámenes de programación de juegos. Eclipse no es *para nada en particular*, pues no se enfoca en ningún dominio específico. (JOHN ARTHORNE 2004)

Como se muestra en este concepto, Eclipse es un producto singular por la versatilidad que ofrece. Pero, ¿qué es lo que le permite a Eclipse tener estas características? Todo esto es posible debido a la poderosa arquitectura basada en plugins con el cual fue diseñado.

1.2.1 Arquitectura de Eclipse

Cuando la plataforma Eclipse fue donada a la comunidad open source por IBM en el año 2001, la noticia no fue notable en el mundo por la enorme suma de dinero que IBM declaró que se había gastado en el desarrollo de la misma (40 millones de dólares); sino debido al resultado obtenido con esta millonaria inversión: un producto inigualable por su arquitectura madura, bien diseñada y extensible basada en plugins. (GALLARDO 2002)

Con la excepción de un pequeño núcleo o kernel denominado Platform Runtime, todos los demás componentes que conforman la plataforma Eclipse son plugins. Entre ellos encontramos el Workbench, el Workspace, el Help y el Team como se muestra en la figura 1-1.

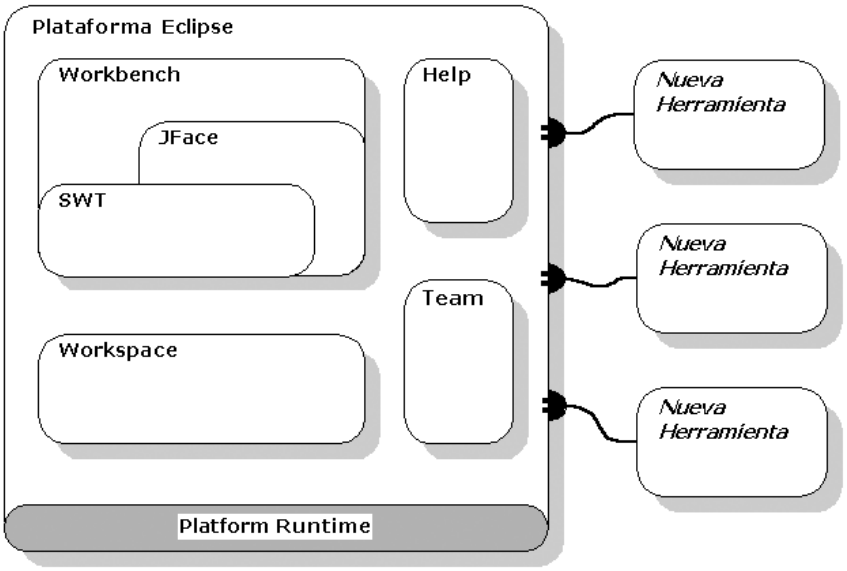

Figura 1-1 Arquitectura basada en plugins

El *Workbench* es el componente de la plataforma responsable de la presentación y la coordinación de la interfaz gráfica de usuario (GUI), además de que ofrece la estructura básica de la misma para Eclipse y los *puntos de extensión* necesarios para extender las funcionalidades de la GUI por otros plugins. Es el encargado, en fin, de mostrar todos los menús, editores de textos, vistas y perspectivas de Eclipse.

El *Workspace* se caracteriza por ser el responsable de manejar los recursos de los usuarios, los cuales se organizan en uno o más proyectos a un nivel superior. Cada proyecto corresponde a un subdirectorio del directorio de trabajo de Eclipse (en inglés *workspace*) y en él se pueden encontrar carpetas y ficheros. También puede proporcionar el código para configurar los recursos del proyecto según sea necesario y como en el caso del *Workbench,* contiene además los puntos de extensión para que otros plugins interactúen con los recursos del los usuarios.

Los plugins que anteriormente mencionamos son los componentes esenciales de la plataforma. Pero además de ellos, la misma contiene integrado otros como el plugin *Team* y el *Help*. El plugin *Team* facilita el uso del control de versiones (o gestión de configuración) de los sistemas para administrar los recursos en los proyectos, además de que define el flujo de trabajo necesario para salvar y recuperar

datos de un repositorio[10]. El componente *Help*, no es más que un sistema de documentación, extensible como la misma plataforma Eclipse, el cual mediante el uso de herramientas provistas por la plataforma, se le puede añadir documentación en formato HTML y usando XML puede definir una estructura de navegación.

Además de los plugins ya referidos podemos encontrar otros que aunque no son parte integral de la plataforma, se encuentran integrados en el Eclipse SDK[11], ellos son el Java Develop Tooling (JDT) y el Ambiente de Desarrollo de Plugin o en inglés Plug-in Development Environment (PDE), que como describe su nombre, es un plugin que facilita el desarrollo de plugins para de esta forma extender las funcionalidades de la misma plataforma. En la sección 1.5 se abordan con más profundidad estos componentes.

Hasta el momento se ha hablado de plugins y puntos de extensión, pero no se han definido concretamente los conceptos anteriores. A continuación se explicarán los mismos.

1.2.2 Plugins y puntos de extensión

Imagine un gigantesco rompecabezas, al principio algunas de las piezas cuando se comienza a armar el mismo, ya han sido conectadas (esto conformaría el núcleo en torno al cual el resto del rompecabezas se construye). Si Ud. se percata, los bordes de las piezas están especialmente conformados para que cada una pueda conectarse a otra perfectamente. Si Eclipse fuera el rompecabezas, los plugins serían las piezas que lo conforman.

Un *plugin* es la unidad más atómica y extensible de Eclipse. Está escrito puramente en el lenguaje de programación Java y típicamente consiste en código Java empaquetado en un archivo de Java (JAR), con algunos archivos de solo lectura, y otros recursos como imágenes, catálogo de mensajes, etc.

[10] El **repositorio** es el lugar en el que se almacenan los datos actualizados e históricos, es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. A veces se le denomina depósito o depot.

[11] Eclipse SDK: se denomina también proyecto Eclipse. Es la forma entregable de la plataforma Eclipse, y que al descargarse contiene también además de la Plataforma Eclipse el plugin JDT y el PDE.

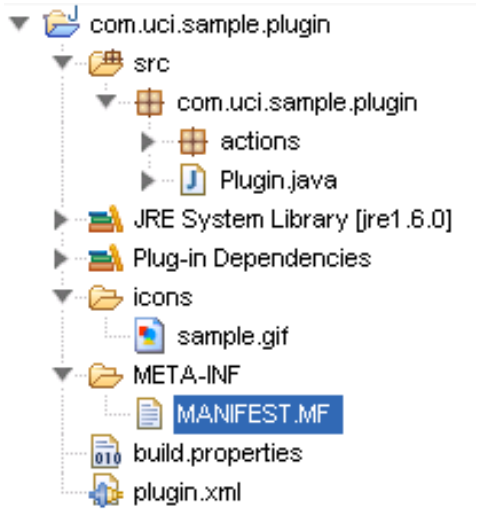

Figura 1-2 Estructura simple de un plugin de Eclipse

Remitiéndonos a la ilustración inicial del rompecabezas, cada pieza tenía bordes especiales que permitían a otras piezas conectarse perfectamente entre sí, estos bordes o conexiones en el mundo de Eclipse se denominan puntos de extensión. Un *punto de extensión* es el mecanismo por el cual un plugin puede añadir funcionalidades específicas a otro.

Anatomía de un plugin

Cada plugin posee un *manifiesto del plugin* (en inglés, *plugin manifest*) en el cual se describe su interconexión con otros plugin, o sea, se declara un número determinado de puntos de extensión y a su vez se exponen las extensiones que se han realizado de otros puntos de extensión definidos por otros plugins.

El manifiesto del plugin está representado por un par de ficheros (Ver en la figura 1-2). El MANIFEST.MF y el plugin.xml. A continuación se hace una breve descripción de los mismos:

- *MANIFEST.MF:* este fichero es generado dentro del directorio META-INF cuando es creado un proyecto plugin. Es un manifiesto del paquete OSGi[12] en donde se describen las dependencias que posee el plugin con otros plugins en tiempo de ejecución y sus atributos como son: el nombre que posee el plugin, el proveedor, la versión del mismo entre otros.

[12] OSGi: se refiere a **O**pen **S**ervices **G**ateway **I**nitiative (Iniciativa de enlace de servicios abiertos) , es una corporación independiente, sin ánimo de lucro que trabaja para definir y promover especificaciones abiertas de software que permitan diseñar plataformas compatibles que puedan proporcionar múltiples servicios.

- *plugin.xml: se* encuentra en la misma raíz del proyecto y consiste en un archivo de formato XML que describe todas las extensiones realizadas por el plugin y declara además los puntos de extensión que el propio plugin ha definido.

Además de estos elementos en el directorio de un plugin se encuentran otros archivos y carpetas como se muestra a continuación:

- *icons*: directorio en el cual se encuentran las imágenes o iconos que va a utilizar el plugin. Usualmente son archivos en formato GIF.
- *plugin.properties*: contiene cadenas de caracteres externalizadas que son referenciadas en el plugin.xml.
- *about.html*: archivo en formato HTML que es utilizado para mostrar informaciones relacionadas con licencias.

1.3 ArBaWeb.

Hasta el momento se han visto todos los aspectos relacionados con la plataforma Eclipse: su arquitectura, plugins que lo componen, entre otros aspectos; pero no se han definido los conceptos que comprenden a ArBaWeb. Es objetivo de este epígrafe tratar brevemente cada uno de estos.

1.3.1 Arquitectura de software de dominio específico

Una *arquitectura de software de dominio específico* (*DSSA*, por sus siglas en ingles) es un proceso e infraestructura que soporta el desarrollo de un modelo del dominio, requerimientos de referencia y una arquitectura de referencia para una familia de aplicaciones dentro de un dominio del problema.

1.3.2 DSSA ArBaWeb

ArBaWeb es una DSSA que permite estructurar y organizar el desarrollo de las aplicaciones que la usen, haciendo uso de buenas prácticas y patrones de diseños. Realiza una serie de modificaciones al concepto original de DSSA. Estas modificaciones consisten en reemplazar los términos: “arquitectura de referencia” por “arquitectura base”, “componentes” por “frameworks y/o componentes”, “herramientas de desarrollo” por “ambiente de desarrollo”. Además adiciona al concepto original desarrollo”, además del elemento flujo de trabajo. (GONZÁLEZ and RIVERO 2007).

El siguiente esquema representa los componentes que establece ArBaWeb. (figura 1-3)

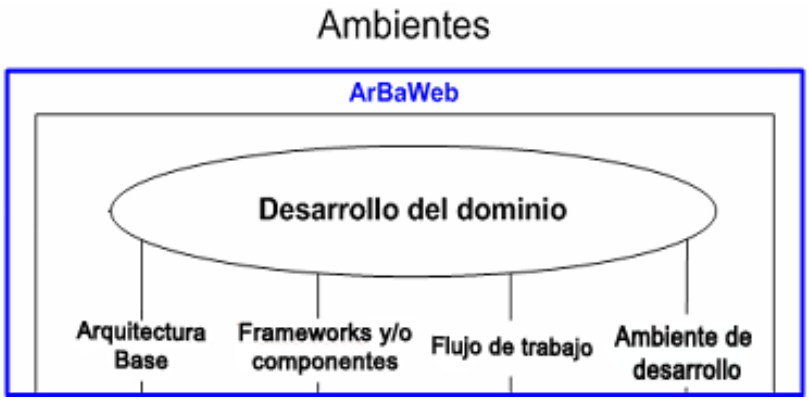

Figura 1-3 Componentes de ArBaWeb

Dominio de ArBaWeb.

ArBaWeb se enmarca en todas las aplicaciones web desarrolladas sobre JEE, específicamente las que utilicen el Spring Framework e Hibernate para la persistencia. Presenta un framework y un flujo de trabajo asociado que brindan facilidades en el desarrollo de software. (GONZÁLEZ and RIVERO 2007).

Arquitectura base

ArBaWeb presenta una arquitectura base que orienta el diseño de las capas lógicas a través de una propuesta de diseño base, organiza la forma de codificar según las propuestas de convenciones o estándares de códigos y recursos, brinda una estructura física para soportar el código, creando así un esqueleto base. (GONZÁLEZ and RIVERO 2007).

Framework

ArBaWeb ofrece dentro de sus componentes un *framework de dominio* nombrado ArBaWeb Framework, basado en Spring e Hibernate, lo que le permite hacer uso de todas las funcionalidades y características de cada uno de estos frameworks. Además de brindar facilidades en el uso de los mismos, incluye también funcionalidades adicionales a otros frameworks. (GONZÁLEZ and RIVERO 2007)

1.4 Plugins existentes que permiten la integración y configuración de frameworks.

A través de los años, desde la aparición del primer framework, siempre ha sido una tarea primordial de los programadores, crear herramientas que faciliten el trabajo de configuración e integración de los mismos. Mucho se ha hecho al respecto y actualmente plataformas de desarrollo como Eclipse y NetBeans, poseen cientos de estas herramientas o plugins que facilitan el trabajo con los más diversos tipos de frameworks.

Es objetivo de este epígrafe, realizar un estudio de un grupo de los plugins de Eclipse más utilizados, tanto en la universidad como internacionalmente, que brindan soporte a frameworks de amplia difusión. Se analizarán de ellos las diferentes características y funcionalidades que brindan para dar soporte.

1.4.1 Spring IDE

Spring Framework (también conocido simplemente como *Spring*) es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java; su diseño facilita la integración con los estándares JEE y herramientas comerciales existentes. Es el más popular y el más ambicioso de todos los framework de peso ligero.

Spring IDE es un plugin de Eclipse que sirve como interfaz gráfica del usuario para configurar los archivos usados por Spring Framework.

Spring IDE provee las siguientes funcionalidades:

- *Naturaleza Spring*[13]*:* adiciona naturaleza Spring a un proyecto para hacerlo un proyecto Spring. Con este tipo de naturaleza se adiciona a la lista de constructores de proyectos de Eclipse, un constructor incremental[14] genérico denominado: "Spring Project Builder". Además soporta una lista de archivos de configuración de beans[15] de Spring o Spring beans.

- *Constructor Incremental:* permite validar todo lo modificado en los archivos de configuración Spring beans definidos en un proyecto Spring.

- *Vista:* muestra un árbol con todos los proyectos Spring y sus archivos de configuración.

- *Editor Gráfico:* muestra gráficamente todos los beans definidos y sus relaciones en uno o varios archivos de configuración.

- *Editor de XML:* configura archivos de Spring beans. Permite el completamiento de etiquetas, valores de atributos y elementos en estos archivos de configuración. Soporta la creación de plantillas personalizadas para generar todas las etiquetas de un beans.

- *Extensión de Eclipse:* permite hacer búsquedas de beans definidos en los archivos de configuración.

- *Asistentes:* dispone de asistentes para crear nuevos proyectos integrados con Spring.

[13] Naturaleza o *nature* en inglés: el término es utilizado para asociar un proyecto con una funcionalidad, como una herramienta o un proceso. Por ejemplo la naturaleza Java es lo que hace que un proyecto sea un proyecto Java, distinguiéndolo de otros tipos de proyectos.

[14] Constructor incremental o incremental builder en inglés, es un término que se utiliza en Eclipse para una herramienta que manipula los recursos de un proyecto de alguna forma. Son siempre usados para aplicar una transformación a un recurso para producir otro diferente o un artefacto de otra índole.

[15] Beans: Son simples clases de Java con sus métodos de acceso.

1.4.2 Hibernate Tools

Hibernate es un framework de Java que provee mecanismos de mapeo de objetos relacionales para definir cómo los objetos de Java son almacenados, actualizados, eliminados y recuperados. Ofrece servicios de búsqueda y recuperación que pueden optimizar el esfuerzo de desarrollo en los ambientes de SQL y JDBC.

Hibernate Tools es un conjunto de herramientas enteramente para trabajar con el framework Hibernate implementado como una suite integrada de plugins para Eclipse. Proporciona facilidades para trabajar con los archivos de mapeos de Hibernate y realizar diferentes tipos de consultas mediante el lenguaje de consultas de Hibernate (HQL) y otras operaciones que soporta este plugin, de una forma más cómoda para los desarrolladores.

Hibernate Tools presenta las siguientes funcionalidades:

- *Editor de mapeos:* es un editor para los archivos de mapeos XML de Hibernate. Soporta auto completamiento semántico para nombres de clases, propiedades, tablas y columnas, haciéndolo más versátil que un editor de XML normal.
- *Consola de Hibernate:* la consola es una nueva perspectiva en Eclipse. Provee la facilidad de visualizar las clases persistentes y sus relaciones. La consola permite ejecutar consultas HQL[16] a una base de datos y buscar los resultados directamente en Eclipse.
- *Ingeniería inversa:* Es su característica más importante, ya que permite generar las clases del modelo de dominio y los archivos de mapeos de Hibernate, los EJB 3 entity beans anotados y documentación en formato HTML, a partir de un esquema de base de datos.
- *Asistentes:* provee un conjunto de asistentes que pueden generar rápidamente entre otros artefactos, archivos de configuración de Hibernate (cfg.xml).
- *Integración con Eclipse JDT:* Hibernate Tools se integra dentro del completamiento de código Java de Eclipse. Esto permite completar código de HQL dentro de código Java y puede incluso señalar los errores dentro de la consulta, si la misma no es válida, en la consola de configuración asociada al proyecto.

1.4.3 Exadel Studio Pro

Exadel Studio Pro es un plugin propietario de la compañía Exadel, que provee un ambiente de desarrollo avanzado de aplicaciones Web para potencializar al máximo tecnologías de código abierto, JEE y AJAX,

[16] HQL: lenguaje de consultas definido por Hibenate.

dentro de Eclipse. Provee soporte para múltiples tecnologías que incluyen Java Server Face (JSF)[17], Apache Struts[18], Hibernate, Spring entre otros. Exadel Studio permite trabajar fácilmente con estos frameworks, todo dentro de un mismo ambiente.

Exadel Studio Pro contiene una extensa cantidad de asistentes, editores y vistas especializadas para brindar soporte a diversos frameworks. A continuación se hace mención de un conjunto de las principales características y funcionalidades que ofrece para el desarrollo de algunos de los mismos.

Apache Struts

- Contiene asistentes para crear nuevos proyectos Struts.
- Define nuevas plantillas de proyectos que son flexibles y fáciles de adaptar.
- Importa proyectos Struts existentes.
- Agrega capacidad Struts a cualquier proyecto Eclipse existente.
- Posee asistentes de completamiento de código dinámico para todos los ficheros de un proyecto Struts.
- Automatiza la generación de código para los diferentes tipos de clases que define el framework Struts.

Hibernate.

- Genera automáticamente ficheros de mapeo de Hibernate a través de asistentes.
- Genera automáticamente clases que cumplen con el patrón Data Access Object (DAO) para las clases persistentes.
- Valida los ficheros de mapeo de Hibernate.
- Agiliza la configuración de Hibernate utilizando asistentes que soportan configuraciones tanto de ficheros .properties como los basados en XML.

[17] Java Server Faces: es un framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones JEE.

[18] Apache Struts o simplemente: Struts: es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma JEE.

Java Server Faces (JSF)

- Crea nuevos proyectos JSF utilizando asistentes o wizards.
- Edita visualmente archivos de configuración de JSF.
- Crea nuevos JSP basados en plantillas flexibles y adaptables.
- Posee asistentes de completamiento de código para los artefactos de JSF (basados en los datos del proyecto).

1.4.4 MyEclipse

MyEclipse es un IDE disponible comercialmente para la plataforma JEE, creado y mantenido por la compañía Genuitec, miembro fundador de la Fundación Eclipse. MyEclipse está construido sobre la plataforma Eclipse e integra soluciones tanto propietarias como de código abierto en un mismo ambiente, proveyendo soporte para el ciclo de desarrollo completo: codificación, despliegue, prueba y depuración. Compite con el WTP, aunque extiende del mismo, es un proyecto separado y ofrece un conjunto de características y funcionalidades diferentes.

My Eclipse tiene dos versiones principales: una profesional y otra estándar. La edición estándar brinda herramientas para intercambiar con sistemas gestores de base de datos y diseñadores web. Ofrece herramientas que brindan soporte a diferentes frameworks como Spring, Struts, JSF y el API[19] de Persistencia de Java (JPA)[20], que incorpora proyectos como Hibernate y TopLinks. A continuación se muestran un grupo de esas funcionalidades y herramientas para los diferentes frameworks:

Spring

- Asistentes para agregar capacidad Spring.
- Importa todas las principales librerías de Spring y sus dependencias a un nuevo proyecto en el proceso de agregar capacidad Spring al mismo.
- Extiende y mejora el Spring IDE adicionando herramientas que permiten integrar un proyecto Spring con Hibernate.

[19] API: es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

[20] JPA: es la API de persistencia desarrollada para la plataforma JEE.

JPA

- Provee soporte para configurar proyectos JPA.
- Genera las entidades persistentes desde la base de datos junto con sus clases DAO, pudiendo generar clases DAOs integradas con el framework Spring.
- Define una perspectiva denominada “MyEclipse Java Persistence” y editores avanzados para trabajar con los diferentes elementos que define el API JPA.

Struts

- Contiene editores multipáginas para los ficheros de configuración de Struts.
- Permite crear plantillas para generar páginas JSP pre configuradas para utilizar los elementos de Struts.
- Agrega capacidad Struts a cualquier proyecto Web de MyEclipse. El proceso agrega automáticamente un proyecto web con las librerías y los XML de configuración de Struts.

1.5 Tecnologías utilizadas

Como se ha podido ver, Eclipse no solo es una plataforma que permite desarrollar todo tipo de lenguajes (Java, C++, PHP, Python), sino que viene equipada con toda la maquinaria pesada para poder construir sus propias herramientas y extender sus funcionalidades.

En este punto del Capítulo I, se abordarán las principales tecnologías utilizadas para lograr la construcción de un plugin de Eclipse, de forma eficiente y con calidad. Es objetivo de esta sección además, definir herramientas que faciliten la administración de los recursos de un proyecto Plugin.

1.5.1 Plugins

1.5.1.1 JDT

El Java Develop Tooling (*JDT*) es uno de los principales subproyectos de Eclipse y es el que hace que la plataforma Eclipse se transforme en un potente IDE de Java, uno de los más reconocidos del mundo. Es el conjunto de plugin más avanzado y elaborado que posee Eclipse. El JDT asiste a los usuarios en los procesos de escribir, compilar, probar, depurar y editar programas escritos en el lenguaje de programación Java, incluyendo plugins de Eclipse.

La plataforma Eclipse combinada con el plugin JDT, provee tantas propiedades y funcionalidades que se podría escribir un libro sobre el tema. A continuación solo se mencionan algunas de ellas.

- *Editores:* ofrece editores para todos los archivos de un proyecto Java, brindando funcionalidades como asistentes de completamiento de código, corrección de problemas a través de su funcionalidad denominada "Quick Fix" y formateo de código fuente, entre otros.

- *Vistas:* provee un conjunto de vistas para navegar y administrar proyectos Java. Entre ellas se tiene el Explorador de Paquetes (Package Explorer), que es una de las piedras angulares de la perspectiva Java; así como la perspectiva Buscador Java o Java Browsing, que está especializada en asistir a los desarrolladores en la comprensión y navegación de grandes aplicaciones con varios proyectos.

- *Configuración de Proyectos:* incluye un amplio soporte para configurar el classpath[21], las dependencias, librerías, opciones del compilador y muchas otras características de un proyecto Java.

- *Búsquedas:* permite realizar búsquedas complejas de métodos, variables, ficheros e incluso realizar búsquedas de plugins.

- *Asistentes:* contiene una gran cantidad de asistentes o wizards que permiten crear diversos elementos Java como proyectos Java, paquetes, clases e interfaces.

- *Depuradores:* provee un ambiente enriquecido para depurar errores, en donde se puede insertar puntos de detención o breakpoints, tracear la ejecución de un programa, inspeccionar y modificar el valor de una variable e incluso cambiar el código de un método durante la depuración del programa.

El plugin JDT dota a la plataforma Eclipse de un IDE de Java completo, como se ha visto hasta el momento; pero además de esto, JDT provee también un API y varios puntos de extensión. Llegados hasta este punto, después de haber visto un conjunto de las numerosas características que aporta JDT a Eclipse, cabría preguntar ¿para qué se necesita utilizar el API de JDT? Si se desea construir un plugin que entre sus funciones se encuentre la de interactuar con aplicaciones Java o con recursos, entonces el API de JDT es indispensable. Posee entre muchas otras capacidades la posibilidad de realizar las siguientes tareas:

- Manipular programáticamente recursos Java posibilitando realizar acciones como: crear proyectos, generar clases, interfaces o incluso detectar errores en el código.

[21] classpath: es un argumento que le muestra a la Maquina Virtual de Java donde buscar las clases y paquetes definidos por el usuario en un programa Java.

- Levantar programáticamente una aplicación Java desde la plataforma.
- Agregar nuevas funcionalidades y extensiones al IDE de Java.

El JDT está estructurado en tres componentes principales que contienen los paquetes del API:

- *JDT Core:* define el núcleo de los elementos Java y provee clases útiles para manipular archivos de extensión .class y el modelo de elementos Java.
- *JDT UI:* contiene la interfaz de usuario que provee el IDE.
- *JDT Debug:* brinda un conjunto de clases que permiten correr y depurar código Java.

1.5.1.2 PDE

El Plugin Development Enviroment o *PDE*, no es más que el Ambiente de Desarrollo para Plugin que provee Eclipse y uno de los principales subproyectos del mismo, junto con JDT. Está compuesto por un conjunto de herramientas que cubren todo el ciclo de vida completo del desarrollo de un plugin. Facilita todo el proceso de crear, desarrollar, probar, depurar, construir y desplegar plugins de Eclipse. Algunas de estas herramientas de PDE incluyen:

- *Editores del Manifiesto*: ofrece editores multipáginas basados en formularios, que administran centralmente todos los ficheros del manifiesto del plugin, o sea, el MANIFEST.MF y/o el plugin.xml
- *Asistentes para la creación de nuevos proyectos:* facilitan todo el proceso de creación de un proyecto plugin.
- *Asistentes para importar:* provee un conjunto de asistentes que simplifican el proceso de importar plugins del sistema de archivos o file system.
- *Asistentes para exportar:* provee un conjunto de asistentes que construyen, empaquetan y exportan plugins con un simple clic a un formato de despliegue.
- *Launchers (Probadores):* permiten probar y depurar aplicaciones de Eclipse, así como paquetes OSGi.
- *Vistas:* PDE provee cuatro vistas y una perspectiva de desarrollo, denominada Plug-in Development, que ayudan a los desarrolladores de plugins a inspeccionar los diferentes aspectos de su ambiente de desarrollo.

- *Herramientas Combinadas:* posee asistentes para externalizar y purificar los ficheros del manifiesto del plugin.
- *Herramientas de Conversión:* consiste en asistentes para facilitar el proceso de convertir un proyecto simple de Java o un archivo JARs simple, en un proyecto plugin.
- *Integración con el JDT*: los ficheros del manifiesto del plugin participan en búsquedas Java y en la reestructuración del código.

1.5.1.3 Web Tools Platform (WTP)

La *plataforma de herramientas web* o *Web Tools Platform (WTP),* al igual que JDT y PDE, es uno de los principales subproyectos de la Fundación Eclipse, cuyo propósito es el de extender la plataforma Eclipse hacia el dominio del desarrollo de aplicaciones sobre la Web y JEE. WTP provee una gran cantidad de herramientas para facilitar el trabajo de los desarrolladores, estas incluyen editores gráficos de código fuente para una variedad de lenguajes, asistentes y aplicaciones incorporadas para simplificar el desarrollo de servicios web, además de herramientas y APIs para soportar el despliegue, ejecución y prueba de aplicaciones.

Como hemos visto, WTP ofrece un sinnúmero de herramientas que de cierto modo satisfacen como punto de partida a muchos desarrolladores, pero que en muchos casos quedan con la solución incompleta, ya que el alcance del WTP se encuentra restringido solamente a los estándares de las aplicaciones sobre la Web y JEE y no ofrece soporte a otras tecnologías populares como Struts, Spring e Hibernate; no obstante, WTP como proyecto de Eclipse, ofrece la posibilidad de expandir la plataforma, agregándole los requerimientos específicos necesitados a través de los puntos de extensión definidos dentro del WTP; ejemplo de ello son el org.eclipse.wst.common.project.facet.core.facets que permite agregar acciones necesarias mientras se está creando un proyecto Web, o el org.eclipse.wst.validation.validator que ofrece la posibilidad de crear validadores para nuevos tipos de ficheros.

1.5.1.4 Subclipse

Subclipse es un plugin que agrega soporte Subversion (SVN) al proyecto Eclipse y que permite realizar entre otras tareas las de sincronización y actualización de los elementos de un proyecto. Provee una perspectiva denominada “SVN Repository Exploring” para trabajar con varios repositorios SVN al mismo tiempo. Dispone de varias vistas, que permiten mostrar al usuario los diferentes repositorios con que se está trabajando, o el historial de los cambios realizados en un recurso determinado.

1.5.1.5 Mylyn

Mylyn es uno de los plugins "de primera" que vienen con Eclipse, permite integrar esta plataforma con gestores de tareas como Bugzilla o JIRA, enfocando el desarrollo a la resolución de tareas. Ofrece la posibilidad de almacenar tareas localmente en un espacio de trabajo o en uno o más repositorios de tarea.

Mylyn usa el contexto de las tareas para enfocar la Interfaz Grafica de Usuario (UI) en la información de interés, oculta lo que es poco interesante y encuentra lo que está automáticamente relacionado, es decir, considerando la información que se necesita para trabajar, logra una mejora en la productividad, al reducir el tiempo que se dedica en la búsqueda, desplazamiento y navegación de la misma. Haciendo el contexto de la tarea explícito, Mylyn también ayuda con la ejecución de tareas múltiples, planificando, rehusando esfuerzos anteriores y compartiendo técnicas.(KERSTEN 2007)

1.5.2 Control de versiones

Se denomina *Control de Versiones,* a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas.

Un sistema de control de versiones debe proporcionar:

- Mecanismos de almacenaje para los elementos que debe gestionar (ej. archivos de texto, imágenes, documentación).
- Posibilidad de realizar cambios sobre los elementos almacenados (ej. modificaciones parciales, añadir, borrar, renombrar o mover elementos).
- Registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente pudiendo volver o extraer un estado anterior del producto).

Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados.

Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial. Este proceso suele conocerse como check out o desproteger. Para modificar la copia local existen dos semánticas básicas:

- *Exclusivos:* para poder realizar un cambio es necesario marcar en el repositorio el elemento que se desea modificar y el sistema se encargará de impedir que otro usuario pueda modificar dicho elemento.

- *Colaborativos*: en el que cada usuario se descarga la copia y la modifica, el sistema automáticamente mezcla las diversas modificaciones. El principal problema es la posible aparición de conflictos que deban ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas. Además, esta semántica no es apropiada para ficheros binarios.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión y un ejemplo de ello podría ser el Subversion.

1.5.2.1 Subversion

Subversion (SVN) es un sistema de control de versiones libre y de código fuente abierto. Uno de los reemplazos más populares de Concurrent Versions System (CVS), ofreciendo mejoras en muchas de las dificultades que presenta este último. Subversion maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central, esto permite que se recuperen versiones antiguas de los datos o examinar el historial de cambios de los mismos. En este aspecto, mucha gente piensa en los sistemas de versiones como en una especie de “máquina del tiempo”. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros ya sea código fuente o de otro tipo.

Funcionalidades de Subversion:

- *Versionado de directorios:* implementa un sistema de ficheros versionado “virtual” que sigue los cambios sobre árboles de directorios completos a través del tiempo.

- *Verdadero historial de versiones:* permite añadir, borrar, copiar y renombrar ficheros y directorios. Los ficheros nuevos que son añadidos comienzan con un historial nuevo, limpio y completamente suyo.

- *Envíos atómicos:* permite a los desarrolladores construir y enviar los cambios como fragmentos lógicos e impide que ocurran problemas cuando sólo una parte de los cambios enviados lo hace con éxito.

- *Versionado de metadatos*: cada fichero y directorio tiene un conjunto de propiedades —claves y sus valores —asociados a él. Se puede crear y almacenar cualquier par arbitrario de clave/valor que desee. Las propiedades son versionadas a través del tiempo, al igual que el contenido de los ficheros.

- *Manipulación consistente de datos:* Subversion expresa las diferencias del fichero usando un algoritmo de diferenciación binario, que funciona idénticamente con ficheros de texto (legibles para humanos) y ficheros binarios (ilegibles para humanos).

- *Ramificación y etiquetado eficientes:* se pueden crear ramas y etiquetas simplemente copiando el proyecto, usando un mecanismo similar al enlace duro[22]. De este modo estas operaciones toman solamente una cantidad de tiempo pequeña y constante.

- *Hackability:* Subversion no tiene un equipaje histórico, está implementado como una colección de bibliotecas compartidas en C con APIs bien definidas. Esto hace a Subversion extremadamente fácil de mantener y reutilizable por otras aplicaciones y lenguajes.

Conclusiones del capítulo

En este capítulo se hizo un análisis detallado de los conceptos necesarios para tener una mayor comprensión del objeto de estudio. Además se mencionaron las características fundamentales de las principales herramientas que hoy en día se utilizan para configurar frameworks, así como, las tecnologías necesarias para implementar el sistema propuesto.

[22] Enlace duro: es una referencia, o indicador, a los datos físicos sobre un sistema de archivos. En la mayoría de los sistemas de archivos todos los ficheros (archivos) son enlaces duros. El nombre asociado al archivo es simplemente una etiqueta que apunta en el sistema operativo a los datos reales.

Capítulo II: Características del Sistema

Introducción

En el presente capítulo se exponen algunas de las características automatizables que define ArBaWeb para un proyecto Web sobre JEE. Se identifican las funcionalidades que debe tener un plugin para facilitar la integración y configuración de cada una de estas características. Se concibe el sistema que se desarrollará.

2.1 Características que establece ArBaWeb para un proyecto Web

Las características principales propuestas por los autores de ArBaWeb como arquitectura base para aplicaciones Web son:

1. *Estructuras de código.*
2. *Diseño de capas lógicas.*
3. *Convenciones o estándares de códigos y recursos.*
4. *Requerimiento de clases de ArBaWeb.*

A continuación se describen a partir de las propias ideas de los creadores Luis A. Pimentel y Iosev Pérez; al final de cada característica se adicionan elementos opcionales que según la opinión de los autores, completaría cada una de ellas.

2.1.1 Estructuras de código

Toda aplicación que utilice ArBaWeb como arquitectura base, se regirá por las especificaciones estándares definidas en JEE para las aplicaciones web. Esta arquitectura define una estructura que permite organizar cada tipo de clase o componente necesario en el desarrollo de software.

Unidades organizacionales

La arquitectura base que establece ArBaWeb define dos unidades organizacionales fundamentales: los subsistemas y los módulos. Los subsistemas engloban un conjunto de funcionalidades del sistema que tienen características muy propias y están a su vez subdivididas en módulos. Los módulos encapsulan un conjunto de funciones que debe realizar el sistema. El uso que se haga de estas unidades organizacionales en la aplicación final debe estar regido atendiendo a la complejidad que posea la misma de acuerdo a las clasificaciones establecidas por ArBaWeb. (GONZÁLEZ and RIVERO 2007)

Clasificaciones de complejidad

Según la complejidad que podría presentar el proyecto, ArBaWeb define tres clasificaciones: baja, mediana y alta. Para cada una crea una estructura de paquetes correspondiente atendiendo a las posibles necesidades que pueda presentar.

• ***Baja:*** se define para soportar las necesidades de las aplicaciones que sean de tamaño pequeño debido a que presentan un solo módulo. Esta contiene el paquete raíz, el nombre del módulo y a continuación toda la estructura referente al mismo (figura 2-1). (GONZÁLEZ and RIVERO 2007)

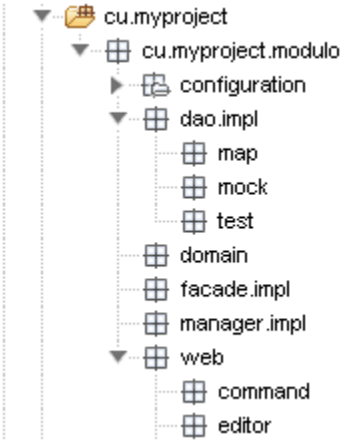

Figura 2-1 Representación de la clasificación de la complejidad baja

• ***Media:*** responde a sistemas no muy grandes, que contengan solo un conjunto de módulos y no sea necesario definir subsistemas. Esta estructura está compuesta por el paquete raíz, seguido en el árbol por los paquetes de los módulos definidos (figura 2-2). (GONZÁLEZ and RIVERO 2007)

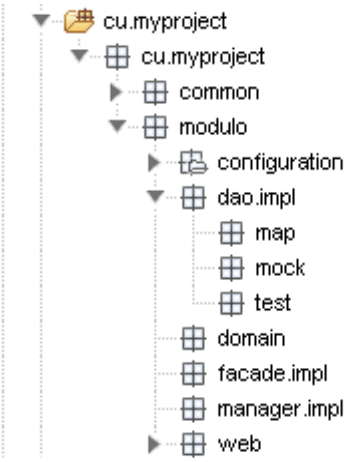

Figura 2-2 Representación de la clasificación de la complejidad media

• ***Alta:*** responde a sistemas complejos que tengan subsistemas y módulos, permitiendo además que se adicionen otros niveles organizacionales definidos por el equipo de arquitectura. Esta estructura está compuesta por el paquete raíz, seguido por los paquetes referentes a los subsistemas. Cada subsistema tendrá como hijos en el árbol de paquetes a los módulos que responden al mismo (figura 2-3). (GONZÁLEZ and RIVERO 2007)

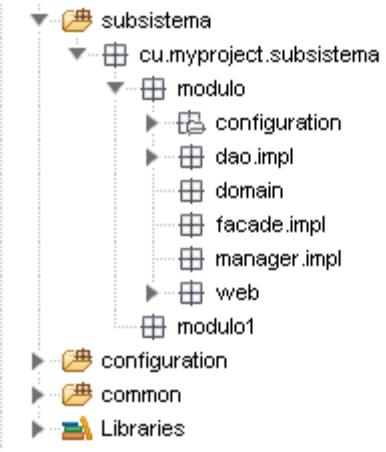

Figura 2-3 Representación de la clasificación de la complejidad alta

Organización de paquetes de los módulos

En un módulo generalmente existen todas las capas lógicas que se definen en la arquitectura base definida por ArBaWeb. A continuación se expone esta estructura.(GONZÁLEZ and RIVERO 2007)

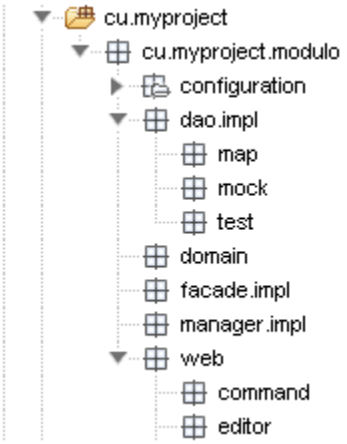

Figura 2-4 Representación de la estructura de paquetes en ArBaWeb

- ***configuration:*** se localizan todos los archivos que tienen que ver con la configuración del módulo, los "*Application-Context"* de Spring Framework, los archivos utilizados para la internacionalización, ficheros de propiedades *".properties"* y cualquier otro destinado a estos fines.

- ***dao:*** se encuentran las interfaces DAOs.
- ***dao.impl:*** se encuentran las implementaciones de las interfaces DAOs.
- ***dao.map:*** se encuentran los ficheros de mapeo utilizados por Hibernate.
- ***dao.test:*** se encuentran las clases de prueba utilizadas para realizar las pruebas de unidad a las implementaciones de los DAO.
- ***domain:*** se encuentran todas las entidades persistentes pertenecientes al módulo.
- ***facade:*** se encuentran las interfaces de las fachadas de negocio.
- ***facade.impl:*** se encuentran las implementaciones de las fachadas de negocio.
- ***service:*** se encuentran las interfaces que representan las operaciones de negocio que se van a exponer o consumir como servicio.
- ***web:*** se encuentran los controladores de la capa de presentación.
- ***web.command:*** se encuentran las clases utilizadas para guardar datos procedentes de las peticiones web (Command).
- ***web.editor:*** se encuentran los *PropertyEditors*[23].
- ***web.validator:*** se encuentran las clases utilizadas para validar datos (Validadores).

Se agregan paquetes opcionales a lo que se estableció como base para ArBaWeb. Estos paquetes son:

- ***manager:*** contiene las interfaces de las clases que encapsulan la lógica de negocio.
- ***manager.impl :*** tiene las implementaciones de las interfaces de los manager

2.1.2 Diseño de capas lógicas

ArBaWeb establece para un proyecto Web un conjunto de capas lógicas para desacoplar sus capas arquitectónicas. Seguidamente se muestran las características que tienen estas capas y los elementos que deben contener.(GONZÁLEZ and RIVERO 2007)

Capa de acceso a datos

[23] Clases que heredan de *java.beans.PropertyEditorSupport*..

En la capa de acceso a datos es donde se encuentran los Objetos de Acceso a Datos o en inglés, Data Access Object (DAO); estos encapsulan la persistencia de los objetos de dominio. En la siguiente figura se muestra un ejemplo simple de los elementos que debe presentar un diagrama de clase de esta capa. (GONZÁLEZ and RIVERO 2007)

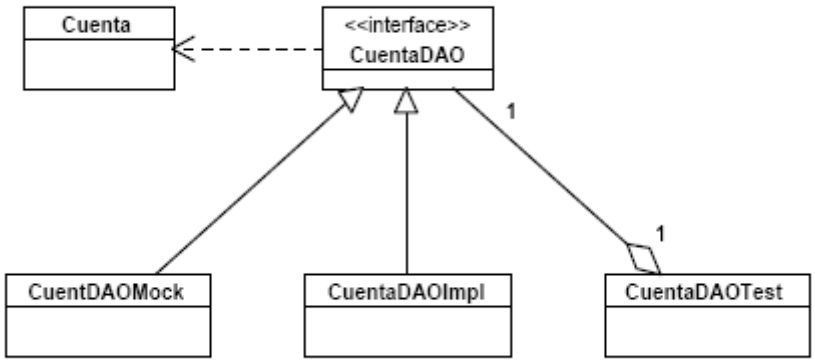

Figura 2-5 Diagrama de clases de la capa de acceso a datos

Los elementos del diagrama de clases según la figura 2-5.

• ***Cuenta****:* es la entidad de dominio a ser persistida por los métodos expuesto en la interfaz CuentaDAO.

• ***CuentaDAO****:* es la interfaz que expone a la capa de servicios de negocio los métodos del DAO para persistir la entidad de dominio Cuenta.

• ***CuentaDAOMock****:* esta clase implementa los métodos expuestos en la interfaz CuentaDAO pero generando datos falsos estáticamente o usando objetos moqueados o falsos, sin conectarse a ninguna base de datos u otra fuente de almacenamiento.

• ***CuentaDAOImpl****:* esta clase es la implementación de la interfaz que realmente se conecta a la base de datos usando Hibernate (puede usarse otro ORM o JDBC directamente) dando, de esta forma, la funcionalidad real a los métodos de CuentaDAO.

• ***CuentaDAOTest****:* es la clase que prueba todos los métodos de la clase CuentaDAOImpl, utilizando valores que testeen casos extremos.

Capa de servicios de negocio

En la capa de servicios de negocio radican los objetos de negocio o Business Objects. Los objetos de negocio separan los datos y la lógica de negocio, usando un modelo de objetos. (GONZÁLEZ and RIVERO 2007)

En la figura 2-6 se muestra en más detalles el diseño básico de clases en esta capa.

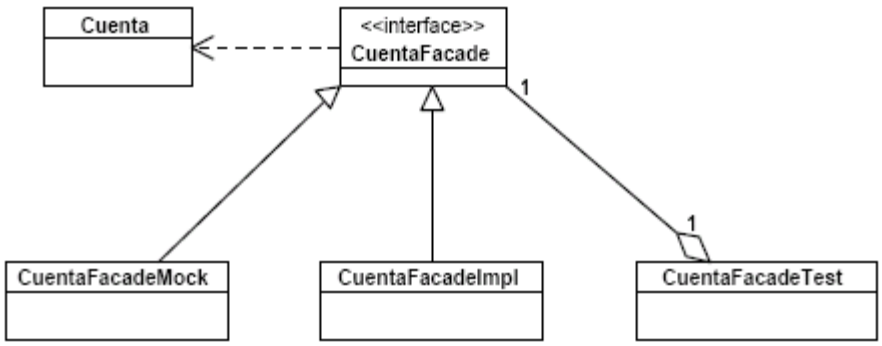

Figura 2-6 Ejemplo de elementos que conforman la capa de negocio

• ***Cuenta****:* es la entidad del dominio que utilizará los métodos expuestos en la interfaz del negocio CuentaFacade.

• ***CuentaFacade****:* es la interfaz que expone a la capa de presentación los métodos para las operaciones de negocio sobre la entidad del dominio Cuenta.

• ***CuentaFacadeMock:*** esta clase implementa los métodos expuestos en la interfaz CuentaFacade, pero generando datos falsos estáticamente o usando objetos moqueados o falsos.

• ***CuentaFacadeImpl****:* esta clase es la implementación de la interfaz de negocio desarrollando los métodos que contendrán la lógica de negocio.

• ***CuentaFacadeTest****:* es la clase que prueba todos los métodos de la clase CuentaFacadeImpl, utilizando valores que prueben los casos extremos. Esta clase es la responsable de asegurar que la lógica de negocio implementada en el objeto CuentaFacadeImpl responda a los requerimientos que debe cumplir.

Elementos opcionales

Aunque no se encuentra definido en ArBaWeb, es de uso general en los desarrolladores que lo utilizan, agregar en la capa de negocios clases denominadas Manager, además de clases especificadas anteriormente, o sea las clases *Facade*, especificadas anteriormente.

En la figura 2-7 se muestra con más detalles el diseño básico de estas clases opcionales para la capa de negocios.

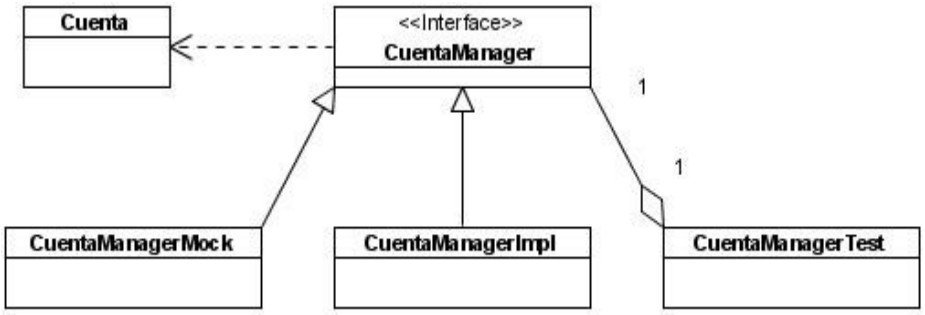

Figura 2-7 Ejemplo de elementos opcionales de la capa de negocio

•***Cuenta:*** es la entidad del dominio que utilizará los métodos expuestos en la interfaz CuentaManager.

• ***CuentaManager****:* es la interfaz que expone a las clases *Facade* los métodos para las operaciones sobre la entidad del dominio Cuenta.

• ***CuentaManagerMock:*** esta clase implementa los métodos expuestos en la interfaz CuentaManager, pero generando datos falsos estáticamente o usando objetos falsos.

• ***CuentaManagerImpl****:* esta clase es la implementación de la interfaz CuentaManager desarrollando los métodos que contendrán la lógica de negocio.

• ***CuentaManagerTest****:* es la clase que prueba todos los métodos de la clase CuentaManagerImpl, utilizando valores que prueben los casos extremos.

2.1.3 Convenciones o estándares de códigos y recursos

ArBaWeb especifica convenciones de nombres y estándares de código para los distintos recursos de un proyecto Web con el objetivo de lograr un lenguaje común y uniforme. Para las clases Java se hacen uso de las convenciones y estándares presentados en la Especificación del Lenguaje Java de la compañía Sun Microsystem.(GONZÁLEZ and RIVERO 2007)

ArBaWeb también define convenciones de nombre para los archivos que tienen que ver con la configuración de la aplicación, los *"Application-Context" (AppContext)* de Spring Framework, archivos utilizados para la internacionalización, ficheros *".properties"* o cualquier otro archivo de configuración. A continuación se especifica para cada elemento de cada capa, las convenciones de nombre y ejemplos para cada una de ellas.(GONZÁLEZ and RIVERO 2007)

Clases de la capa de Acceso a Datos:

- Las interfaces que representan las operaciones sobre los objetos de acceso a datos, correspondientes al patrón de diseño Data Access Object terminan con la palabra “DAO”. Ejemplo: EstudianteDAO.
- Las implementaciones reales de las interfaces DAO comienzan con el nombre de la interfaz correspondiente y terminan con la palabra “Impl”. Ejemplo: EstudianteDAOImpl.
- Las implementaciones falsas de las interfaces DAO comienzan con el nombre de la interfaz correspondiente y terminan con la palabra “Mock”. Ejemplo: EstudianteDAOMock.
- Las clases utilizadas para realizar pruebas a las interfaces DAO comienzan con el nombre de la interfaz correspondiente y terminan con la palabra “Test”. Ejemplo: EstudianteDAOTest.

Clases de la capa de Negocio:

- Las interfaces que representan las operaciones del negocio terminarán con la palabra “Facade”. Ejemplo: EstudianteFacade.
- Las implementaciones de las interfaces de negocio comenzarán con el nombre de la interfaz correspondiente y terminaran con la palabra “Impl”. Ejemplo: EstudianteFacadeImpl.
- Las implementaciones falsas de las interfaces de negocio comienzan con el nombre de la interfaz correspondiente y terminan en la palabra “Mock”. Ejemplo: EstudianteFacadeMock.
- Las clases utilizadas para probar las funcionalidades del negocio terminan con la palabra “Test”. Ejemplo: EstudianteFacadeTest.

Además de las convenciones establecidas por ArBaWeb, se adicionan las siguientes para las clases opcionales “Manager”.

- Las interfaces que expone a las clases *Facade* los métodos para las operaciones sobre la entidad del dominio terminarán con la palabra “Manager”. Ejemplo: EstudianteManager.
- Las implementaciones de la interfaces Manager comenzarán con el nombre de la interfaz correspondiente y concluirán con el sufijo “Impl”. Ejemplo: EstudianteManagerImpl.
- Las implementaciones falsas de las interfaces Manager comienzan con el nombre de la interfaz correspondiente y terminan en la palabra “Mock”. Ejemplo: EstudianteManagerMock.
- Las clases utilizadas para probar las funcionalidades de las clases Manager finalizan con el sufijo “Test”. Ejemplo: EstudianteManagerTest.

Recursos

Los AppContext tendrán la siguiente estructura:

[nombre del proyecto] -[subsistema]-[módulo]-[capa lógica]-context-[tipo].xml

- [nombre del proyecto]: representa el nombre del proyecto en abreviatura.
- [subsistema], [módulo]: indican las unidades organizacionales utilizadas en el proyecto de forma de árbol, pueden ser tantas como la complejidad del mismo lo requiera, según las estructuras definidas para cada nivel de complejidad.
- [capa lógica]: representa la capa lógica que maneja del *Application-Context* de acuerdo al tipo de *beans* que se manejan en él. Se establecen las siguientes clasificaciones que se explicarán a continuación:
    - [dataaccess]: contiene los beans con las funcionales referentes a la capa de acceso a datos.
    - [servlet]: encapsula todos los beans de la capa de presentación.
    - [remoting]: contiene los beans que interactuarán con aplicaciones externas y residirá toda la lógica necesaria para que se pueda tanto exponer servicios a través de diferentes protocolos, como consumirlos de sistemas externos.
    - Cuando no se especifica el tipo de AppContext se toma por defecto que se refiere a las operaciones de negocio de la aplicación, en este XML se definen las fachadas y otros objetos que representen la capa de servicios.
- [tipo]: se refiere al tipo de función que cumplirá este AppContext en la aplicación si es un AppContext utilizado para definir pruebas a la capa correspondiente el valor será “test”, cuando se utiliza para configurar los beans falsos será “mock” y en caso de ser un AppContext de uso final en la aplicación no se especificará el uso.

2.1.4 Requerimiento de clases de ArBaWeb

ArBaWeb Framework define tres clases principales que deben ser declaradas dentro del fichero web.xml de la aplicación y que se encargan de levantar todos los módulos de la aplicación, los filtros, entre otros elementos. Según la configuración definida, ellos son:

- *BaseContextLoaderListener*: es un listener que se declara en el web.xml. Este listener es el encargado de levantar la configuración definida en la aplicación. Implica cargar los módulos definidos, si se activa o no la seguridad y auditoría. Para que pueda cargar la configuración se necesita definir un parámetro en el web.xml (appConfigLocation)

- *UniqueServlet*: es un servlet que se declara en el web.xml. Es el responsable de cargar los Web Application Context de los módulos web de la aplicación, según los definidos en la configuración.
- *UniqueFilter*: es un filtro web que se define en el web.xml. Tiene la responsabilidad de cargar los filtros definidos en ArBaWeb Framework según la configuración de la seguridad y auditoría, incluyendo a los filtros específicos de la aplicación. Para cargar estos últimos es necesario declararlos como beans en los application context y hacer referencias a ellos en el archivo de configuración.

2.2 Funcionalidades que debe tener el plugin

Los problemas que implica el desempeño manual de las actividades de integrar y configurar cada una de las características establecidas por ArBaWeb, crean la necesidad de desarrollar un plugin de Eclipse para automatizar las funcionalidades mencionadas.

Dicho plugin debe contener las siguientes características:

- Debe permitir que la estructura de código de un proyecto Web que haga uso de ArBaWeb sea de acuerdo a la complejidad que posea el mismo, es decir, a la hora de ser creado, debe ofrecer asistentes o wizards que posibiliten al usuario seleccionar qué complejidad posee el proyecto y según la selección generar toda la estructura de código correspondiente. Además el proyecto debe contar con todos los ficheros de configuración y clases necesarias.
- Debe garantizar que se cree toda la estructura de paquetes definidas por ArBaWeb para un módulo. Para ello el plugin a través de asistentes, permitirá que al crearse el módulo contenga la estructura establecida por dicha arquitectura.
- Debe poseer asistentes que ayuden a los usuarios a generar todos los elementos definidos para las diferentes capas lógicas, dígase interfaces y clases de implementación de patrones específicos, clases falsas, clases de testeo según los estándares y convenciones de códigos establecidos.
- Debe permitir que toda la generación de elementos sea lo más configurable posible debido a la versatilidad de ArBaWeb y a que actualmente se ha adaptado para ser usado con diferentes frameworks de persistencia entre otros aspectos.
- Debe contener un manual de usuario integrado al Eclipse que permita consultar los pasos necesarios para trabajar con cada uno de los asistentes creados y agilizar el aprendizaje y uso de ArBaWeb Integrator Tools.

2.3 Diseño del plugin

El diseño que se propone se fundamenta en las características expuestas anteriormente en este capítulo. El plugin se denomina ArBaWeb Integrator Tools (Aitools) o herramientas de integración de ArBaWeb.

En la figura 2-8 se muestra la estructura básica que posee Aitools. El mismo contiene tres subproyectos: Facets, Wizards, Help. Los dos primeros ofrecen las funcionalidades necesarias para integrar un proyecto Web con las características que define ArBaWeb y el subproyecto Help guía al usuario en la utilización del plugin.

Figura 2-8 Estructura básica de Aitools

El componente *Facets* es un plugin que permite que, en el proceso de creación de un nuevo proyecto Web, se le agreguen todos los recursos y especificaciones necesarias para que la aplicación haga uso de la arquitectura base definida por ArBaWeb; así como las configuraciones básicas exigidas por ArBaWeb Framework. Para ello contiene un conjunto de facetas o pantallas que permiten a los desarrolladores crear la estructura de un proyecto Web según la complejidad que posee.

El subproyecto *Wizards* es el encargado de realizar todas las funcionalidades correspondientes para facilitar la creación de cada uno de los elementos definidos por la arquitectura de ArBaWeb. Contiene dentro de sí dos grupos de generadores diferentes: generadores de estructuras y generadores de clases de los patrones utilizados en ArBaWeb.

El plugin *Help* es el elemento que se responsabiliza de ofrecer la documentación necesaria para desarrollar con Aitools, muestra mediante un manual de usuario integrado a la ayuda del Eclipse todos los pasos e instrucciones a seguir.

Todos los plugins fueron construidos en el propio Eclipse haciendo uso del PDE. Están diseñados para ser usados con la versión 3.3 o superior. Los plugins que contiene Aitools son multiplataforma debido a que fueron programados con el lenguaje de programación Java versión 1.5 siempre y cuando se utilice la máquina virtual JVM en la misma versión que el lenguaje o superior.

Para la construcción de Aitools se utilizaron funcionalidades y puntos de extensión definidos principalmente por los plugin JDT y WTP. El JDT brinda un conjunto de plugins para agregar asistentes o wizards al entorno de desarrollo, además de facilitar un conjunto de APIs para el manejo de todos los elementos de un proyecto Java. El WTP contiene puntos de extensión para agregar pantallas para adicionar configuraciones a la hora de crear proyectos Web basados en la plataforma JEE. La presencia de JDT y WTP en las versiones 3.3 y 2.0.1 respectivamente o superior es un requerimiento imprescindible a la hora de instalar Aitools.

2.3.1 Estándares de codificación

Para el desarrollo uniforme de cada unos de los subproyectos creados, se definieron los siguientes estándares de codificación:

- El paquete raíz para todos los subproyectos de Aitools se denomina *org.uci.aitools.*
- Los paquetes que contengan clases que brinden métodos útiles a otras terminarán con la palabra *utils.*
- Se definió el uso de la palabra *internals* para los paquetes que contengan clases o interfaces que no están diseñadas para ser extendidas o utilizadas públicamente por otras clases.
- Los paquetes que posean la palabra *preferences* contienen pantallas o clases para permitir la configuración de la generación tanto de archivos como de clases.
- Las clases que sean asistentes o wizards deben terminar con el sufijo *Wizard.*
- Las clases que representen pantallas deben terminar con el sufijo *WizardPage* para el caso del subproyecto *Wizards* y para el caso del plugin *Facets* debe terminar con *FacetInstallPage.*

2.3.2 Facets

Como se muestra en la figura 2-9, el plugin *Facets* va a estar estructurado por dos paquetes principales que se encuentran dentro del paquete raíz definido por Aitools.

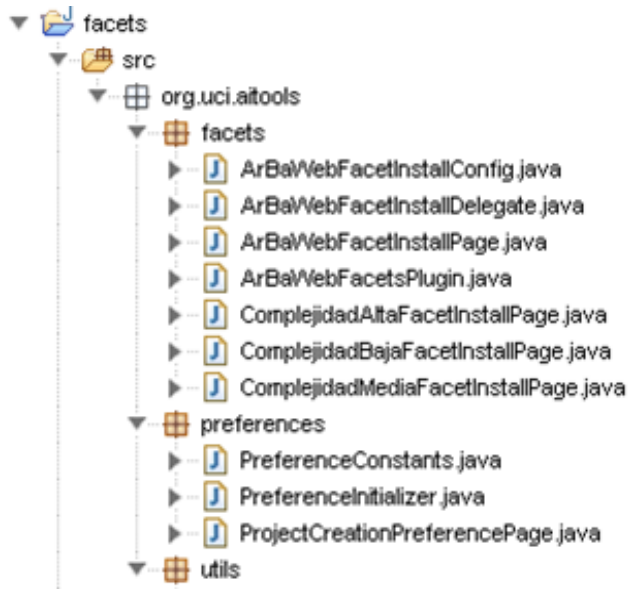

Figura 2-9 Estructura de código del proyecto *Facets*

En el paquete *facets* se encuentran todas las clases que tienen las responsabilidades de extender las funcionalidades del *Faceted Project Framework* del plugin WTP de Eclipse. Adiciona al asistente de crear proyectos Web, las pantallas necesarias para agregar los recursos y las configuraciones de ArBaWeb.

El paquete *preferences* permite al usuario modificar los elementos que pueden ser creados de ArBaWeb, dígase las clases definidas que deben aparecer en el *web.xml* de la aplicación Web, así como los elementos que deben ser creados de la arquitectura base de ArBaWeb.

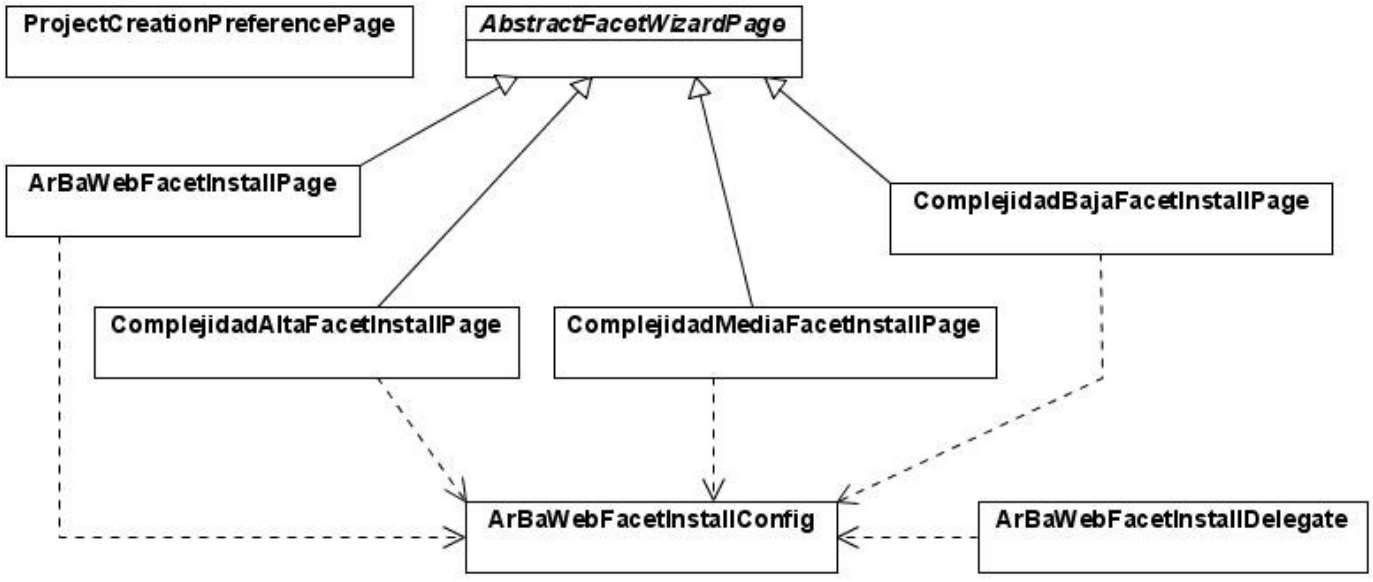

Figura 2-10 Diagrama de clases de *Facets*

2.3.2.1 Descripción de las principales clases del diseño

***ArBaWebFacetInstallPage*:** faceta o pantalla donde el usuario especifica el paquete raíz y la complejidad que poseerá la aplicación Web una vez creada.

***ComplejidadAltaFacetInstallPage**:* faceta o pantalla para proyectos Web que posean complejidad alta. Posibilita definir el nombre de un subsistema y un módulo y si se desea generar el subsistema y el módulo común o *common*.

***ComplejidadMediaFacetInstallPage**:* faceta o pantalla para proyectos Web que posean complejidad media. Permite definir el nombre del módulo inicial que poseerá el proyecto luego de ser creado y generar el módulo común o *common* si es necesario.

***ComplejidadBajaFacetInstallPage**:* faceta o pantalla para aplicaciones Web que posean complejidad baja. Define el nombre del módulo con el cual se creará inicialmente el proyecto.

ArBaWebFacetInstallConfig: clase que se encarga de almacenar todas las configuraciones o datos de las facetas utilizadas.

***ArBaWebFacetInstallDelegate**:* clase que se encarga de ejecutar todas las acciones correspondientes a la creación de un proyecto con todos los elementos definidos por ArBaWeb. Por ejemplo: copiar todas las librerías, registrar las clases necesarias, generar la estructura de paquetes según la complejidad que se haya establecido.

***ProjectCreationPreferencePage**:* clase que permite configurar la generación de los elementos que va a contener el nuevo proyecto Web, dígase el registro de las clases en el *web.xml*, qué clases se van a registrar, entre otros.

2.3.3 Wizards

El componente Wizards al igual que el Facets contiene un conjunto de paquetes ubicados dentro del paquete raíz de Aitools, como se muestra en la figura 2-11.

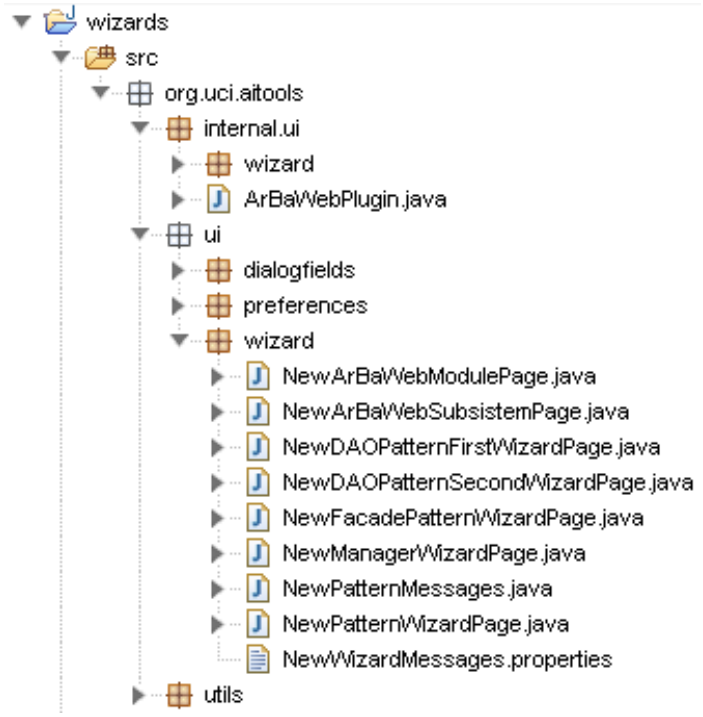

Figura 2-11 Estructura del código del plugin *Wizards*

Por convenciones y estándares de código definidos por Eclipse se especificó el paquete *internal.ui.wizard,* el cual contiene clases que no están diseñadas para ser extendidas, además de todas las clases wizards o asistentes de Aitools.

Dentro del paquete *ui.wizard* se encuentran todas las pantallas de los asistentes creados en el paquete *internal.ui.wizard*.

El paquete *ui.preferences* posee las clases que permiten configurar los diferentes elementos que se muestran en cada una de las pantallas, por ejemplo, clases bases a ser heredadas que pueden ser definidas por los desarrolladores.

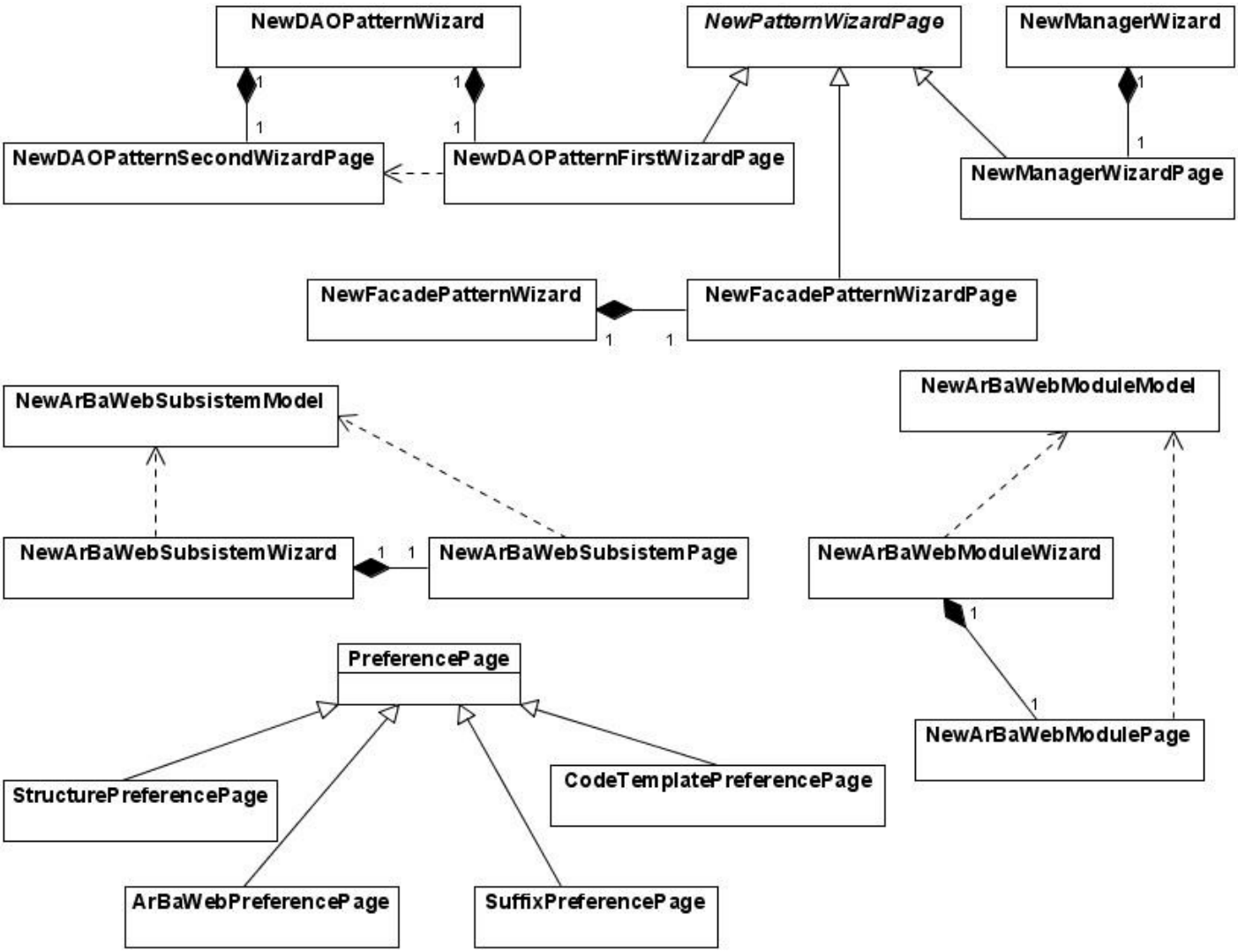

Figura 2-12 Diagrama de clases de *Wizards*

2.3.3.1 Descripción de las principales clases del diseño

NewArBaWebModuleWizard*:* asistente que permite generar nuevos módulos con la estructura definida por ArBaWeb. Contiene la clase *NewArBaWebModulePage* que posee todos los componentes gráficos que se muestran a los usuarios, además de la clase *NewArBaWebModuleModel* que almacena las configuraciones realizadas en la clase *NewArBaWebModulePage*.

NewPatternWizardPage: clase abstracta que contiene todos los controles gráficos comunes de los asistentes que generan los distintos elementos establecidos por ArBaWeb para las capas lógicas. Define métodos básicos que son reimplementados en las clases *NewDAOPatternFirstWizardPage, NewFacadePatternWizardPage, NewManagerWizardPage*, ejemplo: métodos para crear la interfaz del patrón utilizado, crear clases de implementación, registrar las clases en los application context de Spring.

***NewDAOPatternWizard**:* asistente que permite generar todos los elementos definidos por ArBaWeb para la capa de acceso a dato, haciendo uso del patrón DAO. Contiene dos pantallas: *NewDAOPatternFirstWizardPage* y *NewDAOPatternSecondWizardPage* que poseen los controles gráficos para configurar los nombres de las clases que se van a generar, las superclases de las cuales que se extenderán entre otros elementos.

ArBaWebPreferencePage: pantalla que permite modificar las clases de las cuales van a heredar los elementos definidos para las capas lógicas de de negocio y acceso a datos.

CodeTemplatePreferencePage: clase que brinda los controles necesarios para editar las plantillas que contienen cómo se va a generar el código de las clases.

***StructurePreferencePage**:* pantalla que facilita los controles necesarios para editar los nombres de los paquetes de un módulo, según la estructura definida por ArBaWeb.

SuffixPreferencePage: clase que permite configurar los nombres de los sufijos de las clases e interfaces de los diferentes patrones automatizados. Ejemplo: los sufijos DAO, Facade.

2.3.4 Help

El componente *Help* es el encargado de brindar toda la documentación concerniente al desarrollo con el plugin. Extiende del plugin *Help* de la plataforma Eclipse (ver sección 1.2.1). A través de archivos en formato XML y HTML, el plugin ofrece al desarrollador la posibilidad de consultar los pasos necesarios para crear los elementos definidos por ArBaWeb a través de los asistentes ofrecidos.

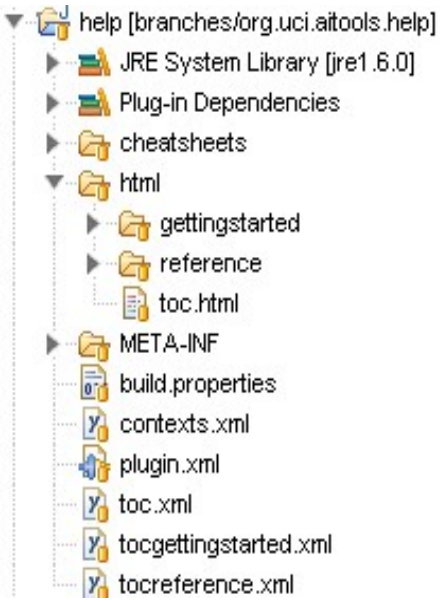

Figura 2-12 Estructura del plugin *Help*

Conclusiones del capítulo

En este capítulo se enmarcaron las principales áreas de ArBaWeb que se automatizaron. Se definieron qué funcionalidades debe tener la propuesta de solución. Se realizó el diseño de los componentes que debe contener el plugin, explicando cómo están estructurados, las funcionalidades que poseen y los estándares de codificación definidos y utilizados para cada uno de los mismos.

Capítulo III: Manual de Usuario

Introducción

ArBaWeb Integrator Tools brinda un manual de usuario integrado directamente al Eclipse, extendiendo del plugin *Help* (*ver sección 2.3.4*), en donde se explican detalladamente todos los pasos necesarios a seguir para su uso. A continuación se muestra cómo trabajar con los asistentes creados y todas las configuraciones que se pueden realizar para que sea más flexible y adaptable la generación de los elementos definidos por ArBaWeb.

3.1 Crear un proyecto web integrado con ArBaWeb

Para crear un nuevo proyecto web integrado con *ArBaWeb*, se deben seguir los siguientes pasos:

1- En la barra Menú seleccionar el menú ítem File->New->Other->Web->Dynamic Web Project y completar los campos de la pantalla que se mostrarán.
2- Haga clic en el botón Next hasta que aparezca la pantalla que se muestra a continuación (figura 3-1).
3- Seleccione el checkbox denominado ArBaWeb para poder pasar a las pantallas que le brindan la posibilidad de configurar los elementos a generar.
4- Haga clic en el botón Next para continuar.

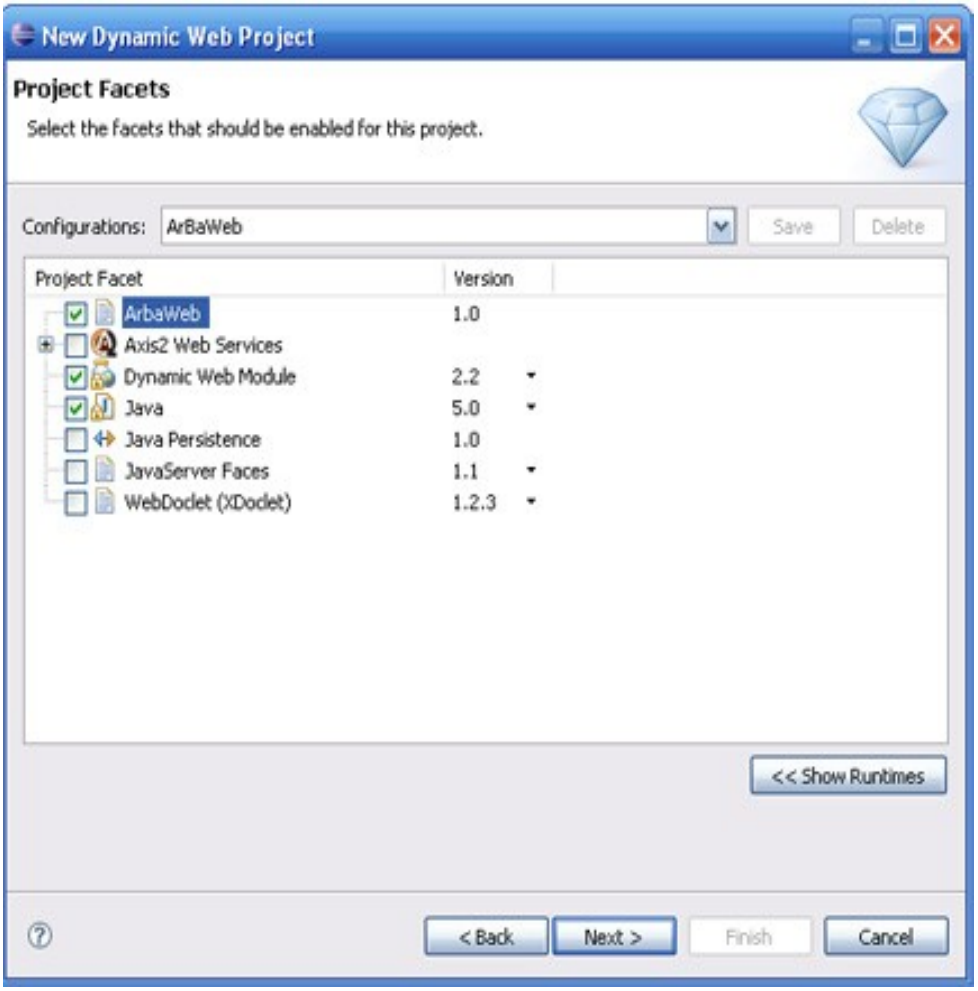

Figura 3-1 Pantalla para seleccionar la integración con ArBaWeb

5- En esta nueva pantalla (figura 3-2), debe editar los campos que aparecen.
6- Haga clic en el botón Next.

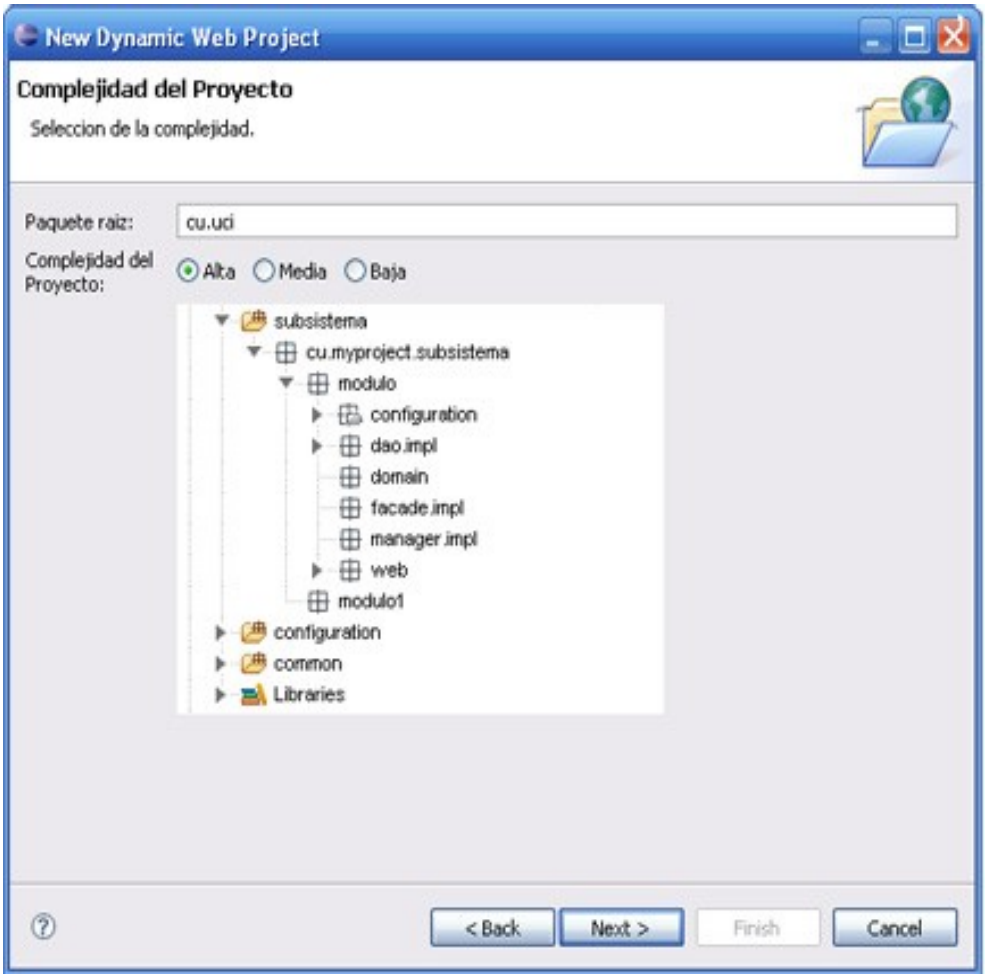

Figura 3-2 Pantalla para seleccionar la complejidad del proyecto

7- La próxima pantalla que se mostrará (figura 3-3), será en dependencia de la elección de la complejidad del proyecto y debe llenar los campos que aparecen.

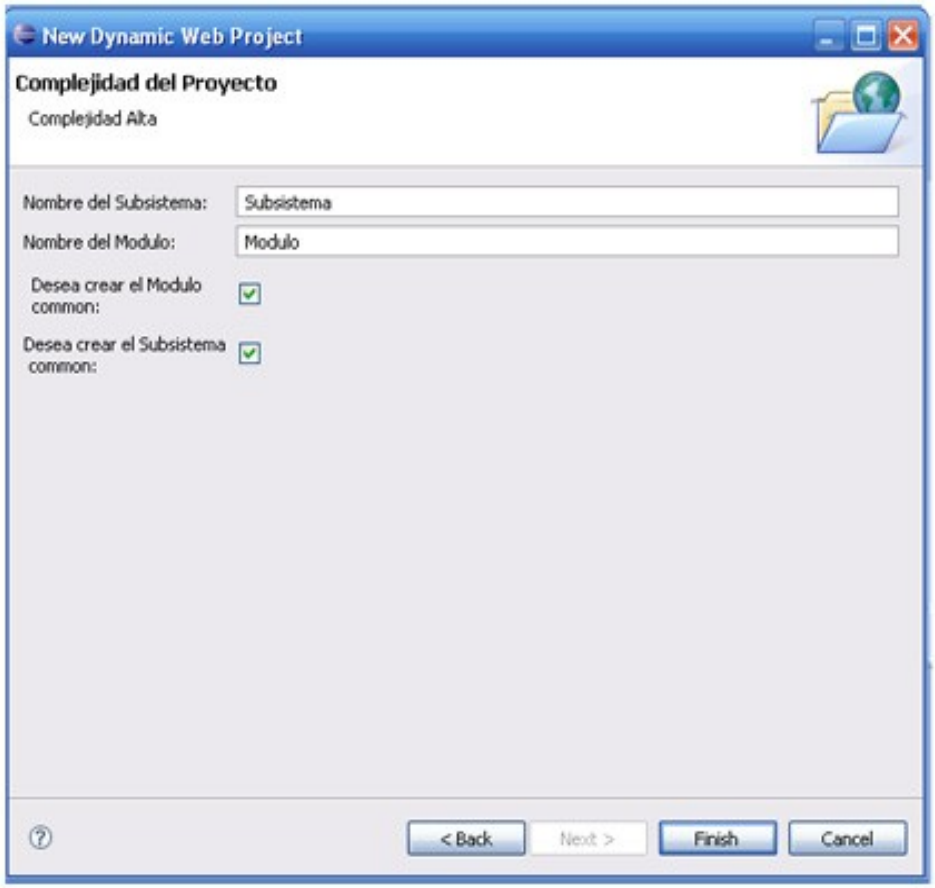

Figura 3-3 Pantalla para definir el nombre de los subsistemas y módulos

8- Haga clic en el botón Finish para finalizar el proceso de creación.

3.2 Crear nuevo Módulo

Una vez creado ya el proyecto, si desea adicionarle nuevos módulos debe seguir el procedimiento que se explica a continuación:

1- En el proyecto o subsistema que se desee crear el nuevo módulo, debe hacer clic derecho y seleccionar New->Other o Ctrl + N y se mostrará la siguiente pantalla (figura 3-4).

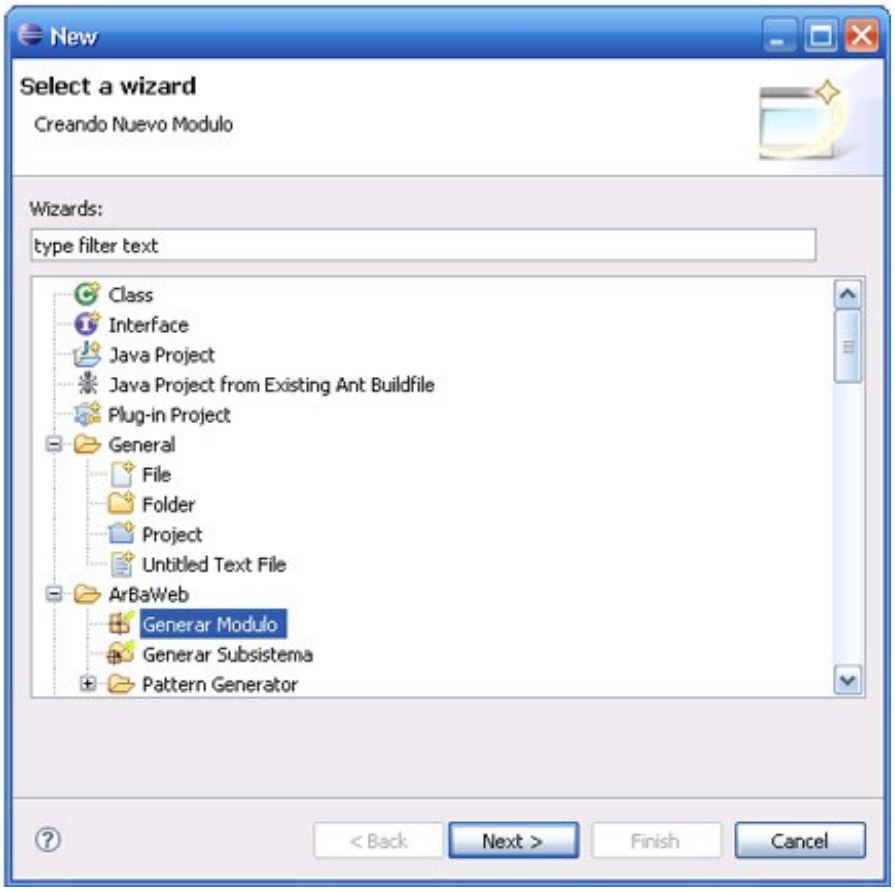

Figura 3-4 Escoger el asistente para crear Módulos

2- Seleccione en la categoría ArBaWeb el elemento Generar Módulo.
3- Haga clic en el botón Next.
4- En la próxima pantalla (figura 3-5), debe llenar el campo que aparece vacío y si desea modificar el nombre del subsistema haga clic en el botón Browse para editarlo.

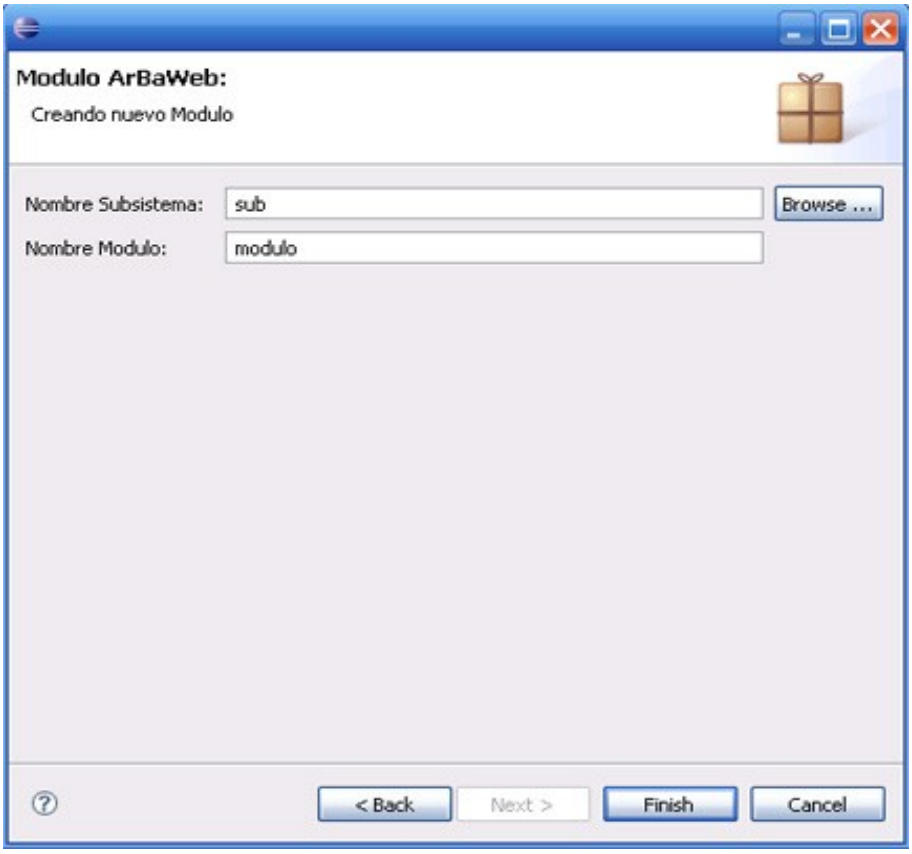

Figura 3-5 Editar el nombre del Módulo

5- Para finalizar, haga clic en el botón Finish.

3.3 Crear nuevo Subsistema

Si ya el proyecto está creado y desea adicionarle un nuevo subsistema, teniendo en cuenta que la complejidad que posee el mismo es alta, debe seguir los siguientes pasos:

1- En el proyecto haga clic derecho y seleccione New->Other o Ctrl + N, y se mostrará la siguiente pantalla (figura 3-6).
2- Seleccione en la categoría ArBaWeb el elemento Generar Subsistema.
3- Haga clic en el botón Next.

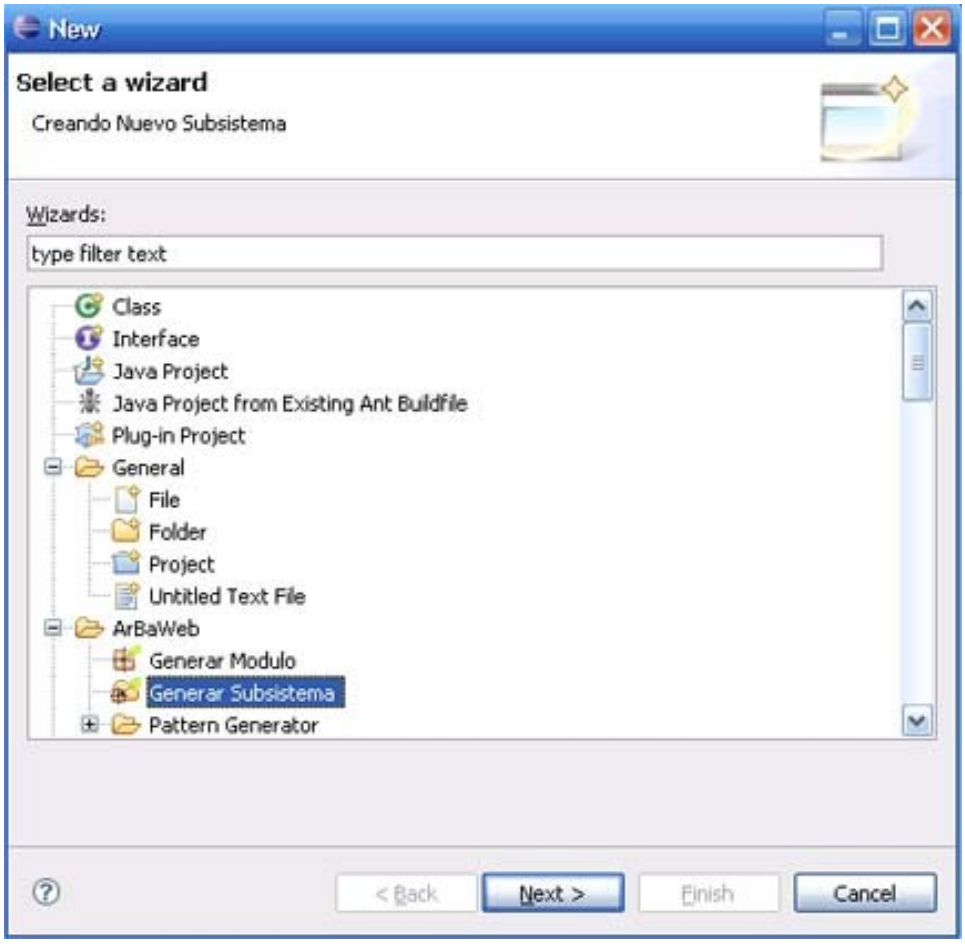

Figura 3-6 Seleccionar el asistente para crear un Subsistema

4- En la próxima pantalla (figura 3-7), debe llenar los campos que aparecen y si desea modificar el nombre del proyecto haga clic en el botón Browse para editarlo.
5- Para finalizar, haga clic en el botón Finish.

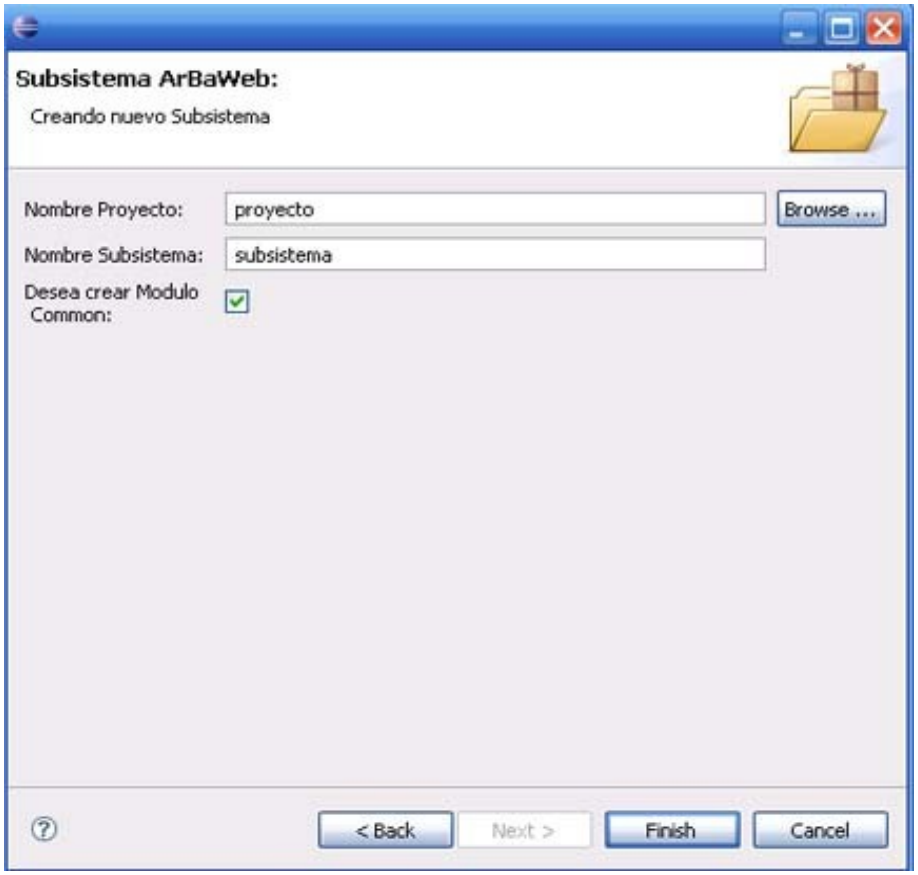

Figura 3-7 Editar el nombre del subsistema

3.4 Crear clases del patrón DAO

Para crear los objetos de la capa de acceso a datos para persistir las entidades del modelo de dominio debe seguir el procedimiento que se muestra a continuación.

1- En el proyecto haga clic derecho y seleccione New->Other o Ctrl + N y se mostrará la siguiente pantalla (figura 3-8).
2- Seleccione en la categoría ArBaWeb/Pattern Generator.
3- Pulse el botón Next para continuar.

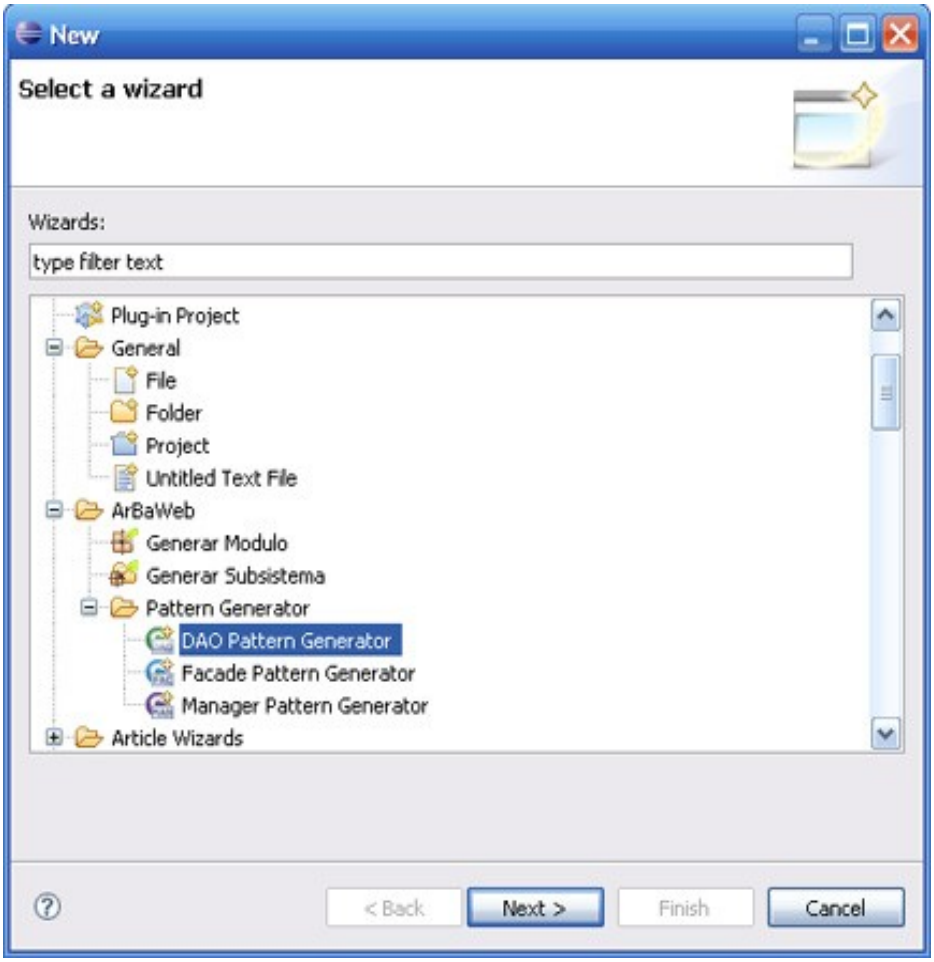

Figura 3-8 Seleccionar asistente del patrón DAO

4- En la próxima pantalla (figura 3-9), debe llenar los campos que aparecen y si desea modificar alguno pulse el botón Browse para editarlos.

Figura 3-9 Editar los elementos del patrón DAO

5- Para crear los elementos adicionales definidos por ArBaWeb, el Test y el Mock, pulse el botón Next (figura 3-10), sino, haga clic en Finish para terminar.
6- En esta nueva pantalla edite los campos que aparecen (figura 3-10), si desea cambiar alguno pulse en el botón Browse.

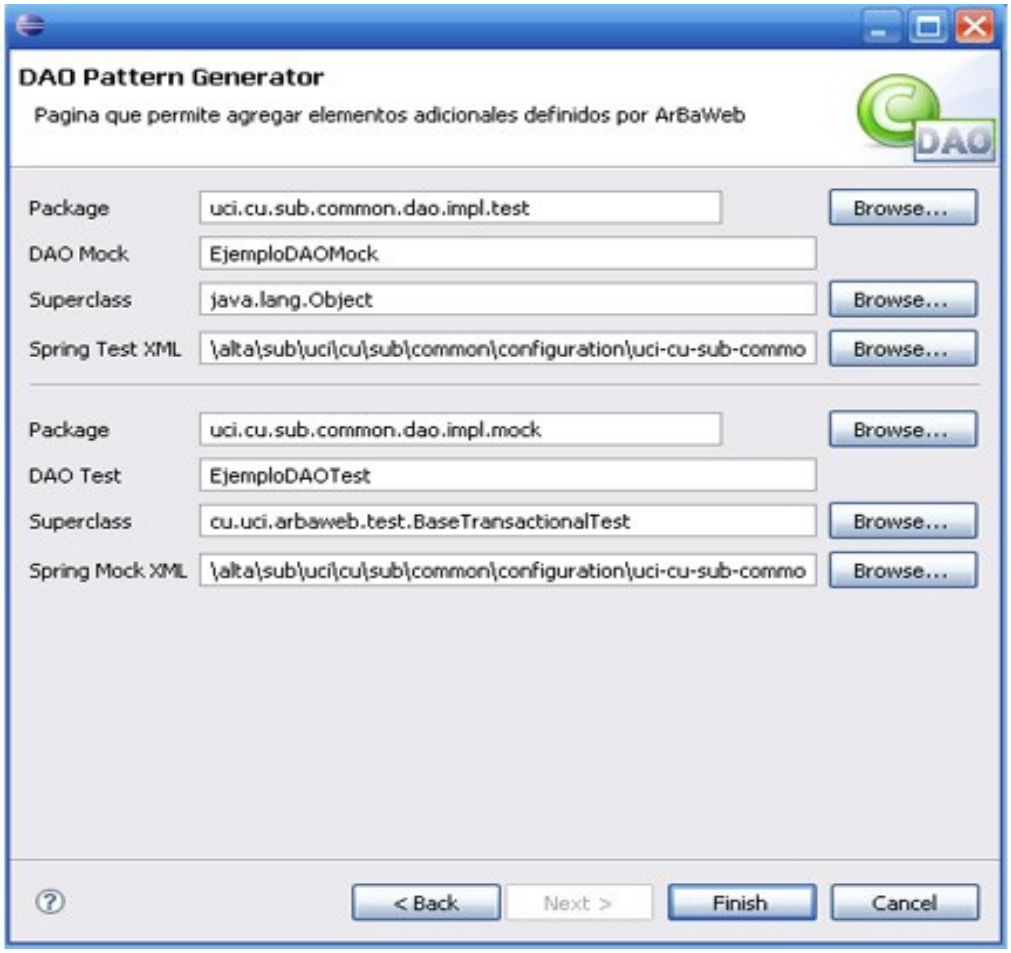

Figura 3-10 Elementos opcionales del patrón DAO

7- Para finalizar, haga clic en el botón Finish.

3.5 Crear clases del patrón Facade

En esta sección se van a crear las clases de fachada de la capa de negocio.

1- En el proyecto haga clic derecho y seleccione New->Other o Ctrl + N y se mostrará la siguiente pantalla (figura 3-11).
2- Seleccione en la categoría ArBaWeb/ Pattern Generators el elemento Facade Pattern Generator.
3- Pulse el botón Next para continuar.

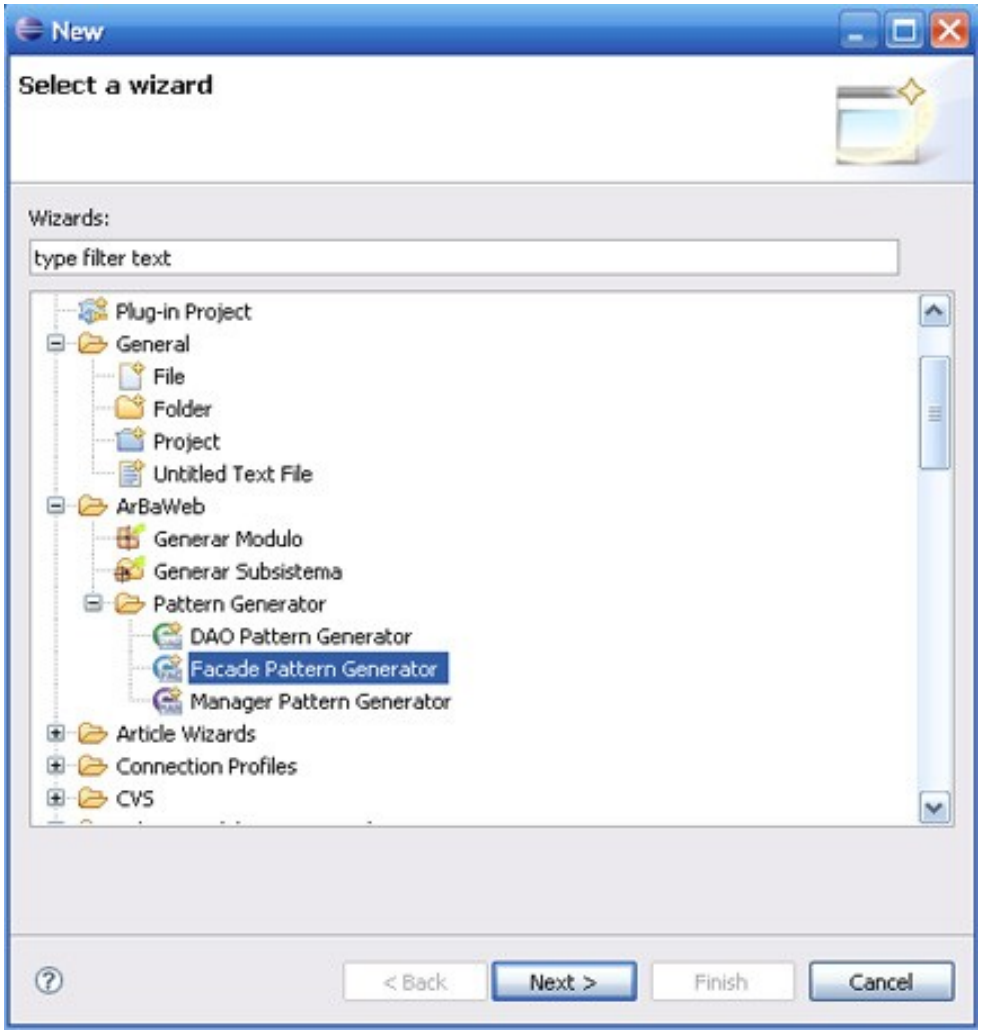

Figura 3-11 Seleccionar el asistente para generar las clases Facade

4- En la pantalla que se muestra en la figura 3-12, debe llenar los campos que aparecen y si desea modificar alguno pulse el botón Browse para editarlos.
5- Para finalizar, haga clic en el botón Finish.

Figura 3-12 Editar los elementos del patrón Facade

3.6 Crear clases del patrón Manager

En esta sección se van a crear las clases que van a operar en la capa de servicio de negocio.

1- En el proyecto haga clic derecho y seleccione New->Other o Ctrl + N, a continuación se mostrará la siguiente pantalla (figura 3-13).
2- Seleccione en la categoría ArBaWeb/Pattern Generator el elemento Manager Pattern Generator.
3- Pulse el botón Next.

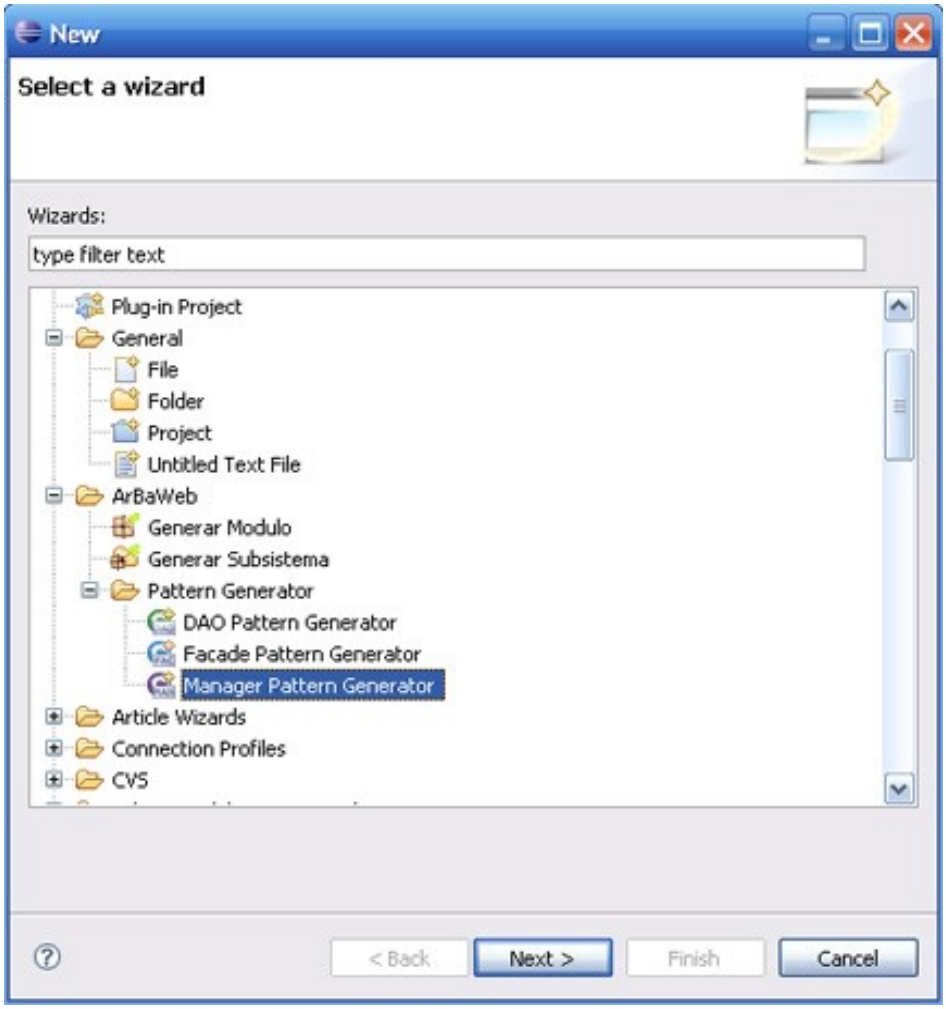

Figura 3-13 Seleccionar el asistente del patrón Manager

4- En la pantalla que se muestra a continuación (figura 3-14), debe llenar los campos que aparecen y si desea modificar alguno pulse el botón Browse para editarlos.
5- Para finalizar, pulse el botón Finish.

Figura 3-14 Editar los elementos del patrón Manager

3.7 Configurar las clases

En esta sección se va a mostrar el procedimiento a seguir para configurar los nombres de las clases.

1- En la barra Menú seleccione el menú ítem Windows->Preferences para abrir la página de preferencias de Eclipse.
2- Seleccione el árbol ArBaWeb y aparecerá la pantalla que se muestra en la figura 3-15.
3- En los campos aparecen los nombres actuales de las superclases, de las cuales van a heredar las clases definidas para los patrones a utilizar. Para editarlos haga clic en ellos y escriba el nuevo nombre.
4- Para finalizar pulse el botón OK.

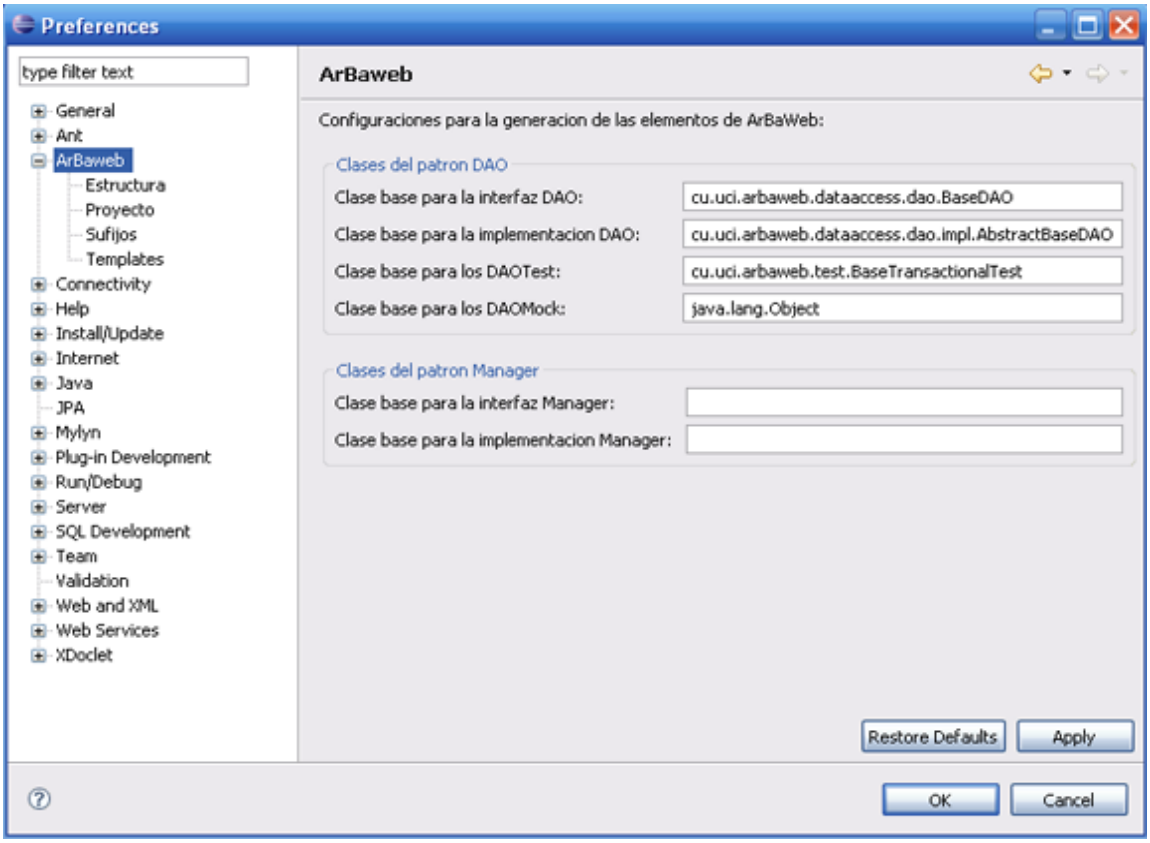

Figura 3-15 Configurar clases bases

3.8 Configurar los Módulos

Para configurar los nombres que deben tener los paquetes de los módulos, debe seguir los siguientes pasos:

1- En la barra Menú seleccione el menú ítem Windows->Preferences para abrir la página de preferencias de Eclipse.
2- Seleccione el elemento del árbol ArBaWeb->Estructura y aparecerá la pantalla que se muestra a continuación (figura 3-16).

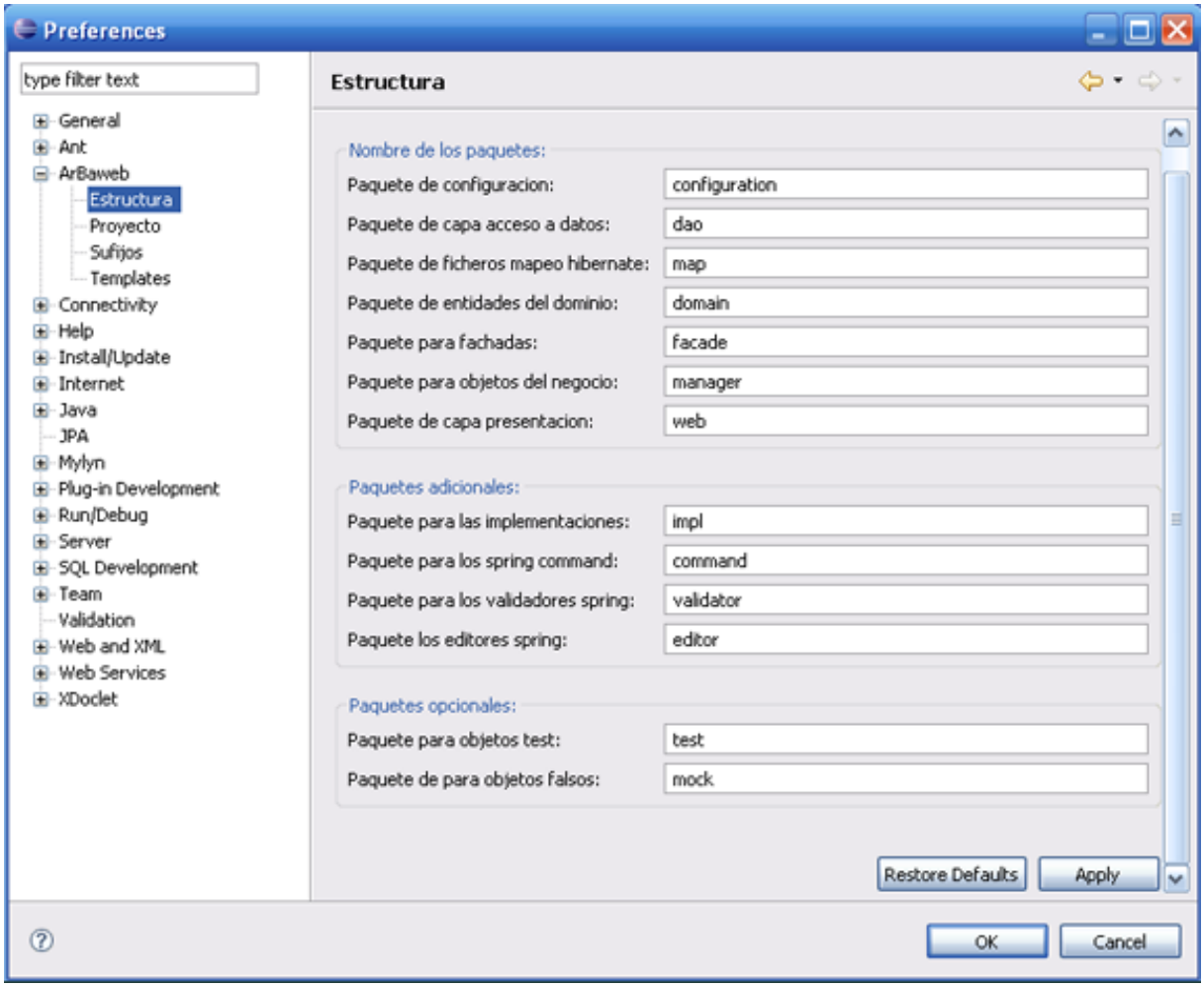

Figura 3-16 Configurar nombre de los paquetes de los módulos

3- En los campos aparecen los nombres actuales con los que se van a generar los paquetes del módulo, para editarlos haga clic en ellos y escriba el o los nuevos nombres con los que se deseen generar.
4- Para terminar pulse el botón OK.

3.9 Configurar la creación de un nuevo proyecto Web

Para configurar la creación de un nuevo proyecto web, debe seguir los siguientes pasos:

1- En la barra Menú seleccione el menú ítem Windows->Preferences para abrir la página de preferencias de Eclipse.

Seleccione el elemento del árbol ArBaWeb->Proyecto y aparecerá la pantalla que se muestra a continuación (figura 3-17).

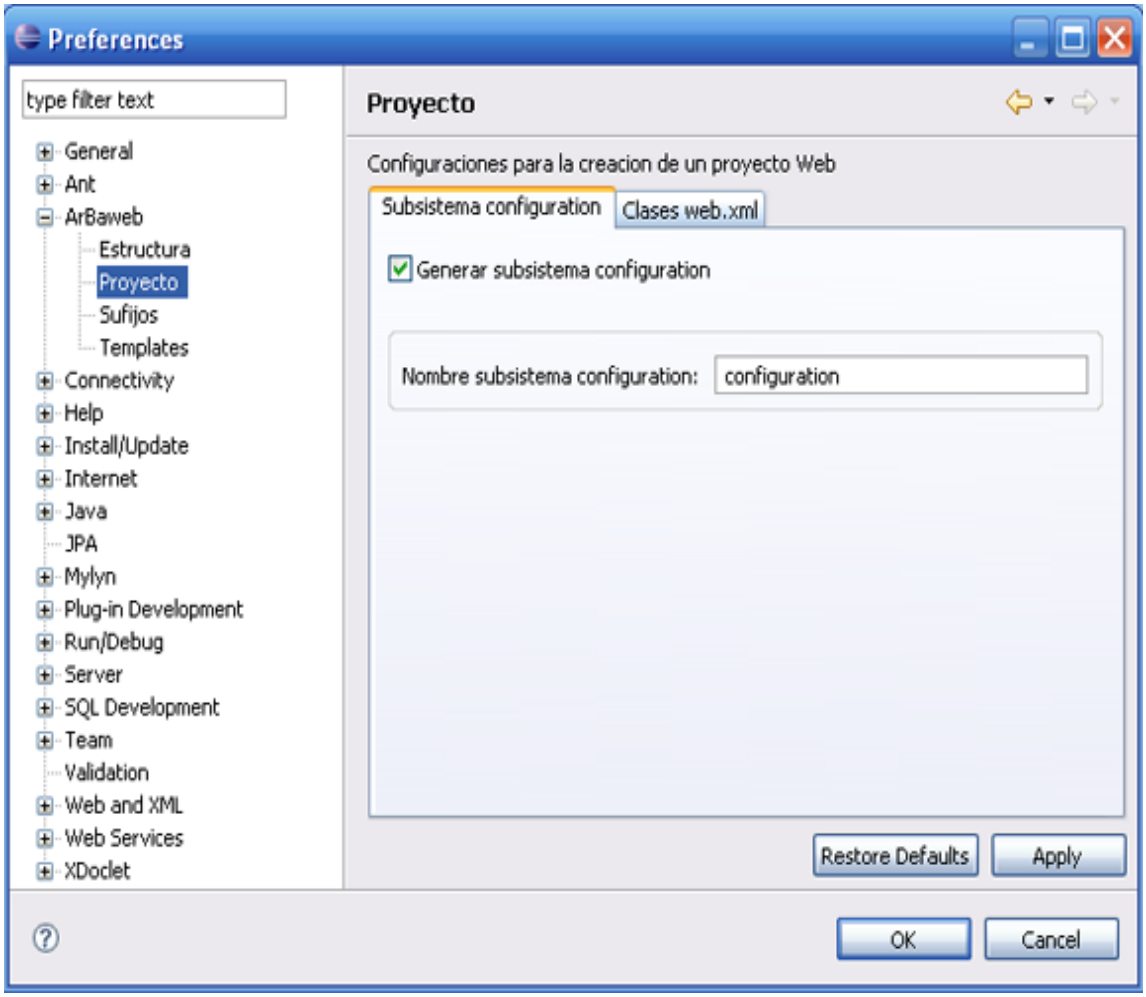

Figura 3-17 Configurar elementos de la creación de un proyecto

2- Si no desea generar el paquete configuration solo desmarque el checkbox "*Generar subsistema configuration"* y haga clic en el botón OK. La próxima vez que se cree un proyecto Web integrado con ArBaWeb haciendo uso de Aitools, no se generará este subsistema ni los ficheros de configuración básicos que contiene.
3- Para cambiar el nombre del paquete *configuration,* solo edite el campo "*Nombre del subsistema configuration*" y haga clic en OK.
4- En la figura 3-18 se muestra la pestaña correspondiente a la configuración de la generación de las clases a ser registradas en el *web.xml* de la aplicación Web. Si no desea que se registren las mismas a la hora de crear un proyecto, solo desmarque el checkbox *"Registrar clases en web.xml".*
5- Para cambiar el nombre de las clases que se van a registrar en el *web.xml* sólo edite los campos y selecciones el botón OK.

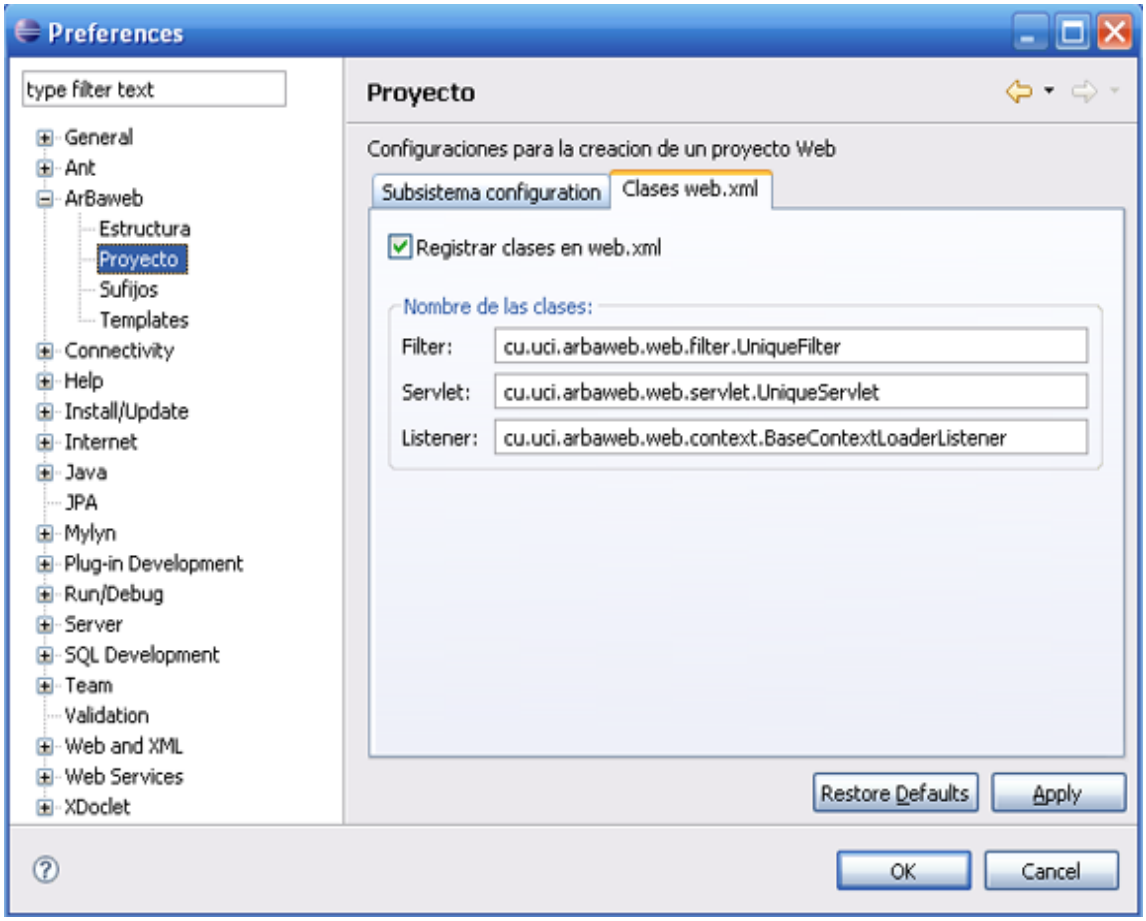

Figura 3-18 Configurar clases para registrar en el Web Application Context

3.10 Configuración de los sufijos

En esta sección se mostrarán los pasos a seguir para configurar los sufijos de las clases y ficheros XML que se van a generar.

1- En la barra Menú seleccione el menú ítem Windows->Preferences para abrir la página de preferencias de Eclipse.

Seleccione el elemento del árbol ArBaWeb->Sufijos y aparecerá la pantalla que se muestra a continuación (figura 3-19).

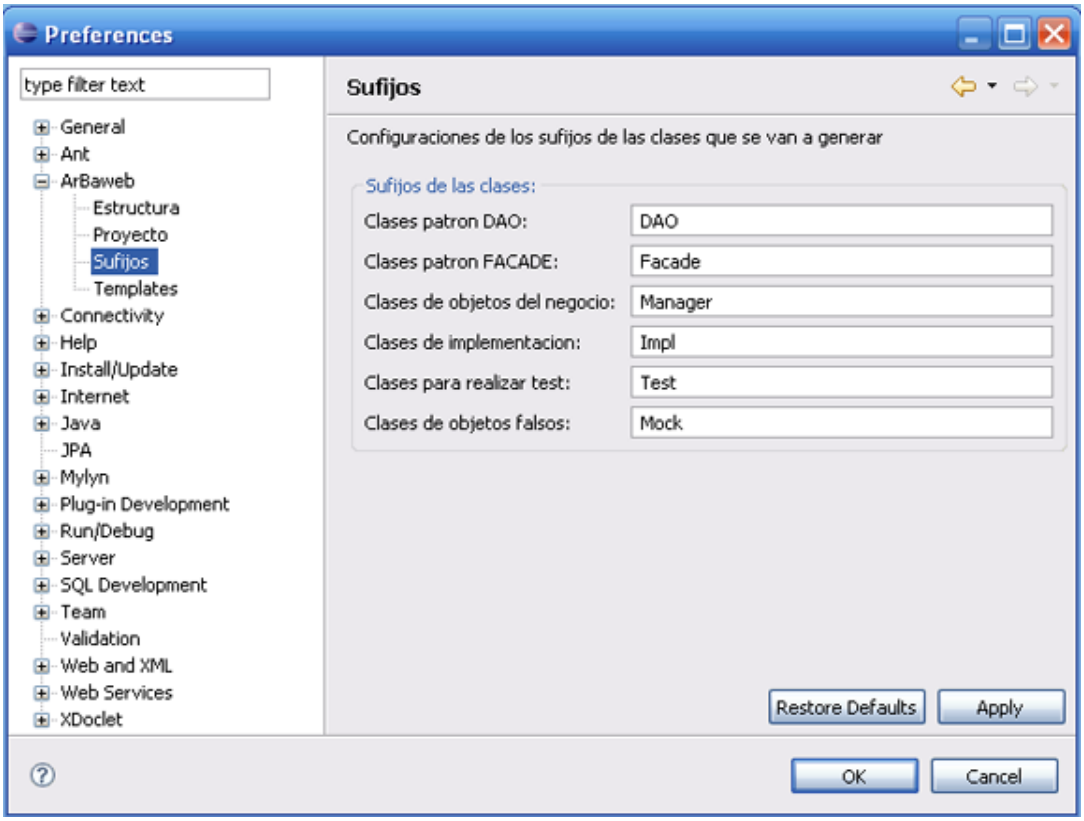

Figura 3-19 Configurar los sufijos de las clases

2- Para hacer algún cambio, edite el/los campos deseados y luego seleccione el botón OK.

3.11 Plantillas para generar elementos de ArBaWeb

Para editar y modificar las plantillas definidas para generar los diferentes elementos de ArBaWeb, debe seguir los siguientes pasos:

1- En la barra Menú seleccione el menú ítem Windows->Preferences para abrir la página de preferencias de Eclipse.

Seleccione el elemento del árbol ArBaWeb->Templates y aparecerá la pantalla que se muestra a continuación (figura 3-20).

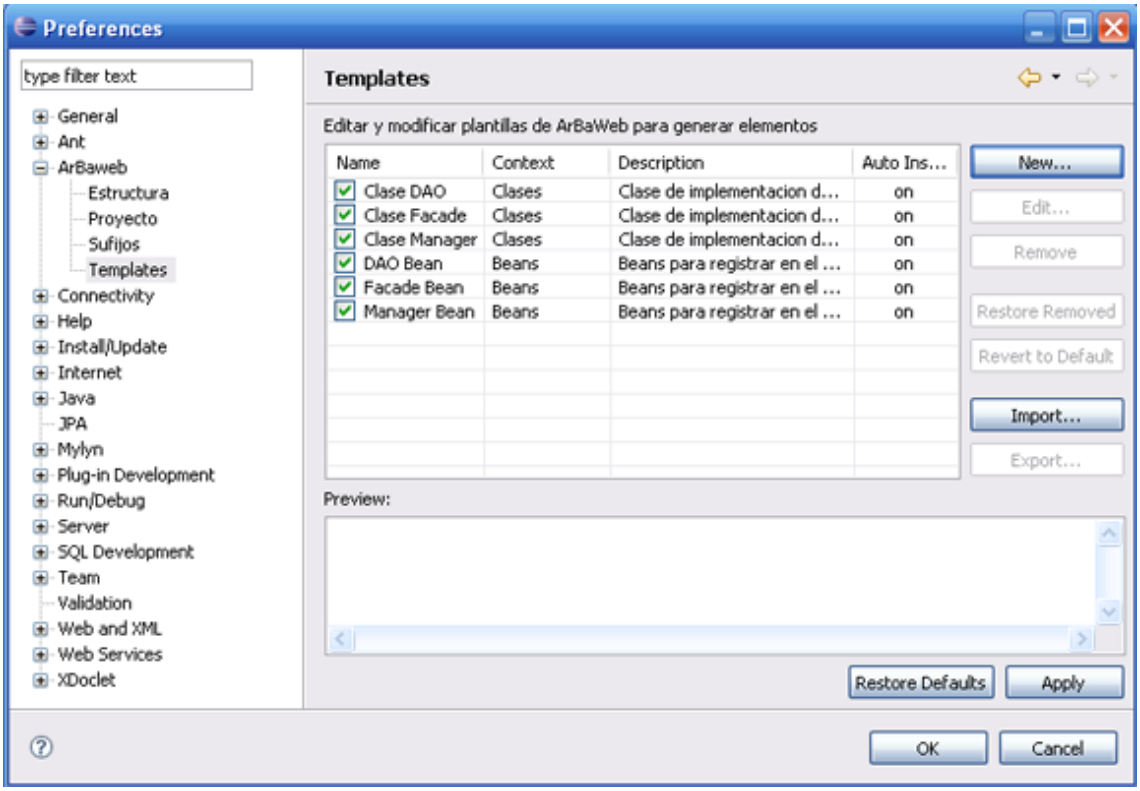

Figura 3-20 Editar y modificar plantillas

2- Para cambiar la forma en que se genere un elemento, seleccione la plantilla que se desea transformar, luego haga clic en el botón Edit. Aparecerá la siguiente ventana (ver figura 3-21) que le permite editar la plantilla.

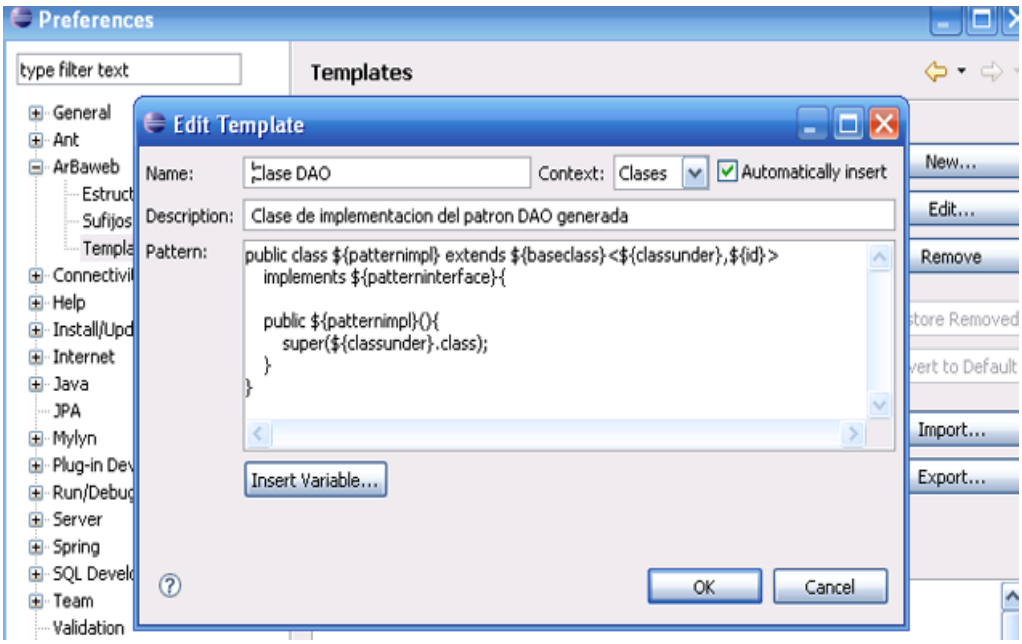

Figura 3-21 Editar la plantilla

3- Para concluir presione el botón OK y la plantilla quedará modificada para que la próxima vez que se genere el elemento, el mismo aparezca con los cambios introducidos en la plantilla.

3.12 Configuración del paquete raíz

Para modificar el nombre del paquete raíz de un proyecto Web integrado con ArBaWeb, debe seguir los siguientes pasos:

1- Seleccione el proyecto al cual le desea cambiar el nombre del paquete raíz.
2- Haga clic derecho en el mismo y en el Pop-up menú seleccione el ítem Properties o simplemente presione las teclas Alt + Enter.
3- Del árbol seleccione el elemento ArBaWeb y aparecerá la pantalla que se muestra a en la figura 3-22.

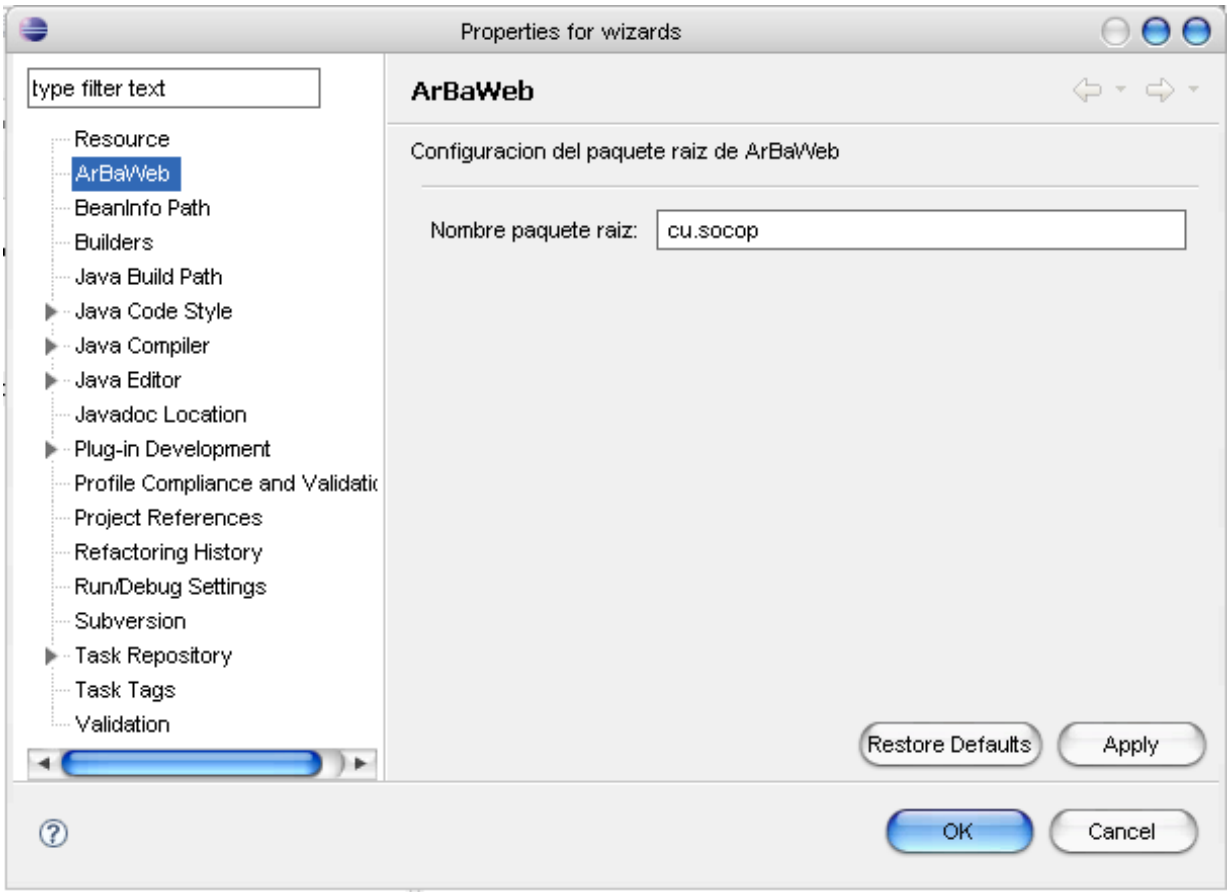

Figura 3-22 Cambiar el paquete raíz

4- Para modificar el paquete raíz solo edite el campo *"Nombre del paquete raíz"* y seleccione el botón OK. Cuando se genere un nuevo módulo el mismo se creará con el nuevo paquete de raíz.

Conclusiones del capítulo

En este capítulo se mostraron todos los pasos a seguir para hacer uso del plugin ArBaWeb Integrator Tools. Se explicaron cada una de las funcionalidades que ofrece: cómo se utilizan los asistentes creados y cómo configurar las generaciones de código.

Conclusiones

Como resultado del trabajo realizado se creó ArBaWeb Integrator Tools, plugin de Eclipse para desarrollar proyectos web integrados con ArBaWeb.

Con la creación de este nuevo plugin se registran los siguientes beneficios:

- La creación de un proyecto web que esté integrado con ArBaWeb, se hace mediante wizards o asistentes que brindan la posibilidad al usuario de seleccionar qué complejidad posee el mismo y genera toda la estructura de código correspondiente. Además el proyecto contiene todos los ficheros de configuración y clases necesarias.
- Garantiza que se cree toda la estructura de paquetes definida por ArBaWeb para un módulo y que contenga la estructura establecida por dicha arquitectura.
- Posee asistentes que ayudan a los usuarios a generar todos los elementos definidos para las diferentes capas.
- Permite que toda la generación de elementos sea lo más configurable posible, debido a la versatilidad de ArBaWeb y a que actualmente el mismo se ha adaptado para ser usado con diferentes frameworks de persistencia.
- Posee un manual de usuario integrado a Eclipse.

Por lo que se puede arribar a la conclusión de que el objetivo de este trabajo se ha cumplido con la realización de las tareas investigativas planteadas.

Recomendaciones

Se recomienda:

- Continuar el desarrollo del plugin con la creación de editores multipáginas para los ficheros de configuración de ArBaWeb, para cuando los mismos se lleguen a estandarizar.
- Promover la creación de plugins que permitan facilitar el desarrollo de proyectos productivos que hagan uso del Eclipse IDE, según la arquitectura, estándares de código y lenguajes de programación utilizados en los mismos.

Bibliografía

BEATON, W. *Eclipse Platform Technical Overview*, 2006. [Disponible en: http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html

CERNOSEK, G. *A brief history of Eclipse*, 2005. [Disponible en: http://www.ibm.com/developerworks/rational/library/nov05/cernosek/index.html

COLLINS-SUSSMAN, B.; B. W. FITZPATRICK*, et al. Control de versiones con Subversion*, 2002. [Disponible en: http://svnbook.red-bean.com/nightly/es/index.html

Control de versiones. Disponible en: http://es.wikipedia.org/wiki/Control_de_versiones

DAVID GALLARDO, E. B. A. R. M. *Eclipse in Action*. Manning Publications, 2003. 220-221 p.

Entorno integrado de desarrollo. 2007. [Disponible en: http://es.wikipedia.org/wiki/Entorno_integrado_de_desarrollo

ERIC CLAYBERG, D. R. *Eclipse: Building Commercial-Quality Plug-ins*. 2. Addison Wesley Professional, 2006. p. *The Eclipse Series.*

GALLARDO, D. *Developing Eclipse plug-ins*, 2002. [Disponible en: http://www.ibm.com/developerworks/java/library/os-ecplug/

GONZÁLEZ, L. A. P. and I. P. RIVERO. *ArBaWeb: ARQUITECTURA BASE SOBRE LA WEB*. Habana, Universidad de Ciencias Informáticas, 2007. 35-37,40-43, 45,47-51. p.

GUCLU, K. A First Look at Eclipse Plug-In Programming, 2004.

H. CERVANTES, R. S. H. *OSGi in a nutshell*, 2004. [Disponible en: http://gravity.sourceforge.net/servicebinder/osginutshell.html

HANSON, J. *Simplify Java Object Persistence with Hibernate*, 2004 [2008]. Disponible en: http://www.devx.com/Java/Article/22652

HEMRAJANI, A. *Agile Java Development with Spring, Hibernate and Eclipse* Sams, 2006. p.

Integrated development environment. 2007. [Disponible en: http://en.wikipedia.org/wiki/Integrated_development_environment

JDT Programmer's Guide. 2008]. Disponible en: http://help.eclipse.org/help33/index.jsp?topic=/org.eclipse.jdt.doc.isv/guide/jdt_int.htm

JOHN ARTHORNE, C. L. *Official Eclipse 3.0 Faqs*. Addison-Wesley Professional, 2004. 384 p. *Eclipse Series.*

KERSTEN, M. *Mylyn 2.0, Part 1: Integrated task management*, 2007. [2008]. Disponible en: Mylyn 2.0, Part 1: Integrated task management

NACI DAI, L. M., ARTHUR RYMAN. *Eclipse Web tools platform : developing Java Web applications*. Addison-Wesley, 2007. p. *The Eclipse Series.*

Plug-in Development Environment Overview. 2008]. Disponible en: ttp://help.eclipse.org/help33/topic/org.eclipse.pde.doc.user/guide/intro/pde_overview.htm

RETAMAR, A. *Mi primera hora con Eclipse*, 2004. 1.

Spring Ide Features. 2008]. Disponible en: http://springide.org/project/wiki/SpringideFeatures

TIM WAGNER, T. B., PAUL MEIJER, PIETER HUMPHREY. *Overview of the Eclipse Web Tools Platform*, 2005. [Disponible en: http://dev2dev.bea.com/pub/a/2005/09/eclipse_web_tools_platform.html

UI Components. 2007]. Disponible en: http://www.eclipse.org/eclipse/platform-ui/

VENU. *History of Eclipse*, 2006. [Disponible en: http://www.venukb.com/blog/2006/07/21/history-of-eclipse/