Arquitectura empresarial y software libre, J2EE

Fecha de creación: 12.08.2002

Revisión 2.0 (01.11.2002)

Alberto Molpeceres Touris (al@javahispano.org)
Martín Pérez Mariñán (martin@javahispano.org)

http://www.javaHispano.org

Arquitectura empresarial y software libre, J2EE

El diseño de aplicaciones empresariales ha sido desde siempre uno de los campos más prolíficos para el software propietario. Las empresas siempre han sido reticentes a la implantación de software libre para tratar ese preciado tesoro que es su información.

Una de las causas de esta situación es sobre todo el desconocimiento del mercado, tanto por parte de las empresas proveedoras de servicios como por parte de quien contrata dichos servicios. Muchas consultorías utilizan soluciones propietarias simplemente porque no conocen otras, incluso cuando los márgenes de beneficios que puedan obtener sean menores. Por su parte es muy común que los clientes conozcan tan sólo las soluciones de las empresas más importantes y sigan el camino que éstas marcan. La consecuencia de todo esto es normalmente la adquisición de productos tan caros como innecesarios, a la vez que se generan unos altos costes de formación, consultoría y subcontratación.

Afortunadamente, el panorama está cambiando en los últimos años. En la actualidad el número de soluciones empresariales en forma de software libre está creciendo a un ritmo que parece imparable. Ya no son únicamente los clásicos servidores web o bases de datos los protagonistas, sino que un gran número de productos como servidores de aplicaciones, soluciones de mensajería, sistemas de servicios web, etc., están surgiendo con fuerza. Cada vez es más díficil ignorar la existencia de tales soluciones, y los responsables en la toma de decisiones están considerando, más a menudo, al mundo del software libre como una solución válida.

Sin ir más lejos, la prestigiosa consultora Meta Group ha afirmado para ZDNet que en la actualidad el 14% de las empresas que desarrollan con Java están utilizando soluciones basadas en software Open Source y software libre, es más, la misma compañía asegura que para el 2006 será el 80% el porcentaje de empresas que hagan uso de soluciones libres.

Todo esto hace que en la actualidad sea factible crear una arquitectura empresarial con todos los requisitos de escalabilidad, fiabilidad, mantenibilidad, flexibilidad, seguridad que necesitan estos sistemas y utilizando única y exclusivamente software libre.

En este documento queremos mostrar como las soluciones libres, pueden, y de hecho lo están haciendo, crearse un hueco dentro del mundo empresarial y al mismo tiempo ofrecer todas las características que estos entornos necesitan.

Arquitectura empresarial y el panorama actual

Una posible definición abstracta de arquitectura empresarial sería : "El estudio de sistemas empresariales complejos desde el punto de vista de su estructura". Un arquitecto empresarial ha de ser capaz de estudiar un problema en concreto y de escoger una serie de componentes con los que modelar la arquitectura más adecuada para el problema en cuestión. Dichos componentes pueden ser servidores de aplicaciones, contenedores web, servidores de mensajería, etc. El arquitecto, asimismo, ha de ser capaz de establecer el

modo de trabajo de dichos componentes, las herramientas utilizadas y las relaciones existentes entre los mismos.

La labor más complicada y con mayor responsabilidad para un arquitecto empresarial es con diferencia la elección de la plataforma empresarial sobre la que se cimentará la arquitectura de una empresa. Una elección errónea puede tener resultados catastróficos tanto para la empresa como para el responsable de dicha elección.

Una plataforma de desarrollo empresarial ha de ofrecer una serie de servicios a los arquitectos y desarrolladores encaminados a facilitar el desarrollo de aplicaciones empresariales, al tiempo que ofrece la mayor cantidad posible de funcionalidades a los usuarios. Normalmente una plataforma de desarrollo empresarial tiene los siguientes requisitos:

- Escalabilidad : Ha de ofrecer una buena escalabilidad tanto horizontal como vertical de modo que si aumenta la carga del sistema podamos añadir servidores o ampliar los existentes sin que sea necesario realizar modificaciones.
- Mantenibilidad : Ha de permitir añadir modificar los componentes existentes sin que se modifique el comportamiento del sistema.
- Fiabilidad
- Disponibilidad : Hemos de tener el soporte de arquitecturas tolerantes a fallos, sistemas de redundancia, etc., que nos aseguren que nuestros sistema estará siempre disponible.
- Extensibilidad : Ha de ser posible añadir nuevos componentes y capacidades al sistema sin que se vean afectados el resto de componentes.
- Manejabilidad : Nuestros sistemas han de ser fácilmente manejables y configurables.
- Seguridad : Hemos de tener buenos sistemas de seguridad tanto a nivel de autenticación, como de autorización y como de transporte.
- Rendimiento : Se ha de ofrecer automáticamente soporte de clustering, balanceo de carga, pools de objetos, pools de conexiones, cachés, y en general mecanismos que permitan aumentar el rendimiento de manera transparente al usuario.

La importancia de una plataforma empresarial es que todos estos componentes se nos ofrecen de manera automática de modo que los desarrolladores son mucho más productivos. La diferencia entre utilizar una plataforma de desarrollo empresarial y no utilizarla radica en que en el segundo caso nuestros desarrolladores perderán mucho tiempo realizando sistemas de bajo nivel, no pudiéndose centrar en el desarrollo de aplicaciones y por consiguiente disminuyendo considerablemente la productividad de los mismos.

Desarrollar una arquitectura empresarial utilizando única y exclusivamente productos basados en software libre es realmente complicado. Probablemente, al proponer la utilización de cualquier producto basado en software libre en nuestra empresa nos

encontraremos con las típicas preguntas: "¿Quién va a dar soporte técnico?", "¿ A qué teléfono llamo si se produce algún error?", "¿Me estás diciendo que este producto gratuito es mejor que este otro producto que me cuesta 30.000 euros por CPU? ¡ Me tomas por tonto !"

Por suerte parece que algo está cambiando en el panorama empresarial. Muchos arquitectos e ingenieros están comenzando a ver un nuevo requisito indispensable para cualquier plataforma de desarrollo empresarial: **La disponibilidad de soluciones libres**. El que una plataforma disponga de soluciones libres aporta unos beneficios extra muy importantes a los arquitectos ya que obtienen un gran abanico de soluciones de bajo coste que pueden evaluar, y que además suelen tener detrás una amplia comunidad de desarrolladores para poder resolver los problemas que vayan apareciendo.

Volviendo al tema de las soluciones cerradas, muchas de éstas, como pueden ser los diferentes ERPs presentes en el mercado, a menudo no disponen de alguna (o de varias) de estas características por lo que hay tener mucho cuidado cuando se contratan estos productos (a menudo muy caros) y asegurarse de que lo que nos ofrecen realmente vale lo que estamos pagando.

Por si fuera poco, las soluciones cerradas, a menudo pueden representar problemas para las empresas ya que éstas se verán ligadas a terceras entidades de las que dependerán para la evolución de sus estructuras de información. Es por ello que en la actualidad las plataformas más importantes son aquellas que disponen del apoyo de gran cantidad de empresas, entidades o asociaciones y que disponen de grupos de estandarización que aseguren el futuro de las mismas. En la actualidad las plataformas de desarrollo empresarial más importantes son tres:

- CORBA
- .NET
- J2EE

CORBA

Sin duda, el modelo a seguir por cualquier plataforma empresarial. Las ventajas que ofrece CORBA son muy importantes:

- Soporte de múltiples sistemas operativos
- Soporte de múltiples lenguajes
- Gran cantidad de servicios : mensajería, eventos, transacciones, persistencia, etc.
- Controlada por un organismo serio, OMG

Las ventajas son realmente muy importantes, aún así, CORBA arrastra unos problemas que suponen un verdadero lastre:

- Complejidad: CORBA es una plataforma de desarrollo muy compleja, aunque

existen capas de abstracción que facilitan el desarrollo de aplicaciones, lo cierto es que desarrollar un simple programa de "Hola Mundo" no es una labor trivial.

» Burocracia: La evolución de las especificaciones de CORBA está sujeta a demasiados pasos de burocracia, lo que origina en que para ver novedades en la plataforma sea necesario esperar grandes cantidades de tiempo

» Pocas soluciones libres: Como consecuencia de su complejidad y de la lentitud de su evolución se deriva que existen pocas soluciones libres. OpenORB es un ejemplo y existen algunos otros pero la cantidad no es demasiado elevada.

.NET

La plataforma de desarrollo empresarial de Microsoft desembocó en el panorama empresarial hace un año y ofrece a los desarrolladores algunas ventajas interesantes.

» Soporte de múltiples sistemas lenguajes: Aunque no soporte todas sus características, lo cierto es que con .NET es posible desarrollar aplicaciones utilizando simultáneamente varios lenguajes de programación.

» Ideal para entornos Microsoft: Si en nuestra empresa disponemos de gran cantidad de software y hardware dependiente de Microsoft, probablemente la mejor opción para continuar desarrollando sea esta plataforma, ya que su integración con los productos de la empresa de Redmond es perfecta.

» Visual Studio .NET: La plataforma .NET dispone de esta gran herramienta que además de su potencia ofrece a un entorno homogéneo de desarrollo.

» Un gran departamento de marketing: .NET será una plataforma que se utilizará ampliamente a nivel empresarial gracias al deparamento de marketing de Microsoft que destaca por su tremenda eficacia.

» Requiere desarrolladores poco experimentados: Bajo la plataforma de desarrollo de Microsoft es posible utilizar lenguajes como VB .NET que hacen muy sencilla la creación de aplicaciones empresariales. De este modo es posible tener un equipo de desarrolladores poco experimentados y sin embargo que éstos puedan crear fácilmente aplicaciones.

Aún así, si la lista de ventajas es bastante grande, la lista de desventajas no le tiene nada que envidiar:

» No soporta múltiples sistemas operativos: El mundo de .NET gira en torno al sistema operativo Windows y aunque se están intentando trasladar partes importantes de la plataforma , como la CLR o C#, a otros sistemas operativos, lo cierto es que estas partes forman una parte ínfima de la totalidad de la plataforma de Microsoft.

» Un único dueño : La plataforma .NET está dominada única y exlusivamente

por Microsoft. Esto supone un grave problema ya que es una única empresa la que puede añadir y quitar características según crea necesario. Además esto hace que la competencia sea nula y no se estimula la evolución de la plataforma.

» Es una tecnología inmadura: Con sólo un año en el mercado, apenas ha salido algún proyecto importante desarrollado con esta tecnología. Su inmadurez hace que probablemente deba pasar algún tiempo hasta que sea realmente productiva.

» Pocas soluciones libres: No existe una correspondencia exacta entre las partes de la plataforma .NET y soluciones libres. Existen proyectos como Mono o dotGNU que están portando algunas de sus partes, aún así, no se puede crear una arquitectura completa utilizando sólamente productos basados en software libre.

J2EE

J2EE, la plataforma creada por SUN en el año 1997 es según nuestra opinión la que ofrece mejores perspectivas de desarrollo para empresas que quieran basar su arquitectura en productos basados en software libre. J2EE, nos ofrece entre otras las siguientes ventajas:

» Soporte de múltiples sistemas operativos: Al ser una plataforma basada en el lenguaje Java, es posible desarrollar arquitecturas basadas en J2EE utilizando cualquier sistema operativo donde se pueda ejecutar una máquina virtual Java.

» Organismo de control: La plataforma J2EE está contolada por el JCP[1], un organismo formado por más de 500 empresas. Entre las empresas que lo forman estan todas las más importantes del mundo informático (SUN, IBM, Oracle, SAP, HP, AOL, etc.) lo que garantiza la evolución de la misma.

» Competitividad: Muchas empresas crean soluciones basadas en J2EE y que ofrecen características como rendimiento, precio, etc., muy diferentes. De este modo el cliente tiene una gran cantidad de opciones a elegir.

» Madurez: Creada en el año 1997 como respuesta a la tecnología MTS de Microsoft, J2EE tiene ya cinco años de vida y una gran cantidad de proyectos importantes a sus espaldas.

» Soluciones libres: En la plataforma J2EE es posible crear arquitecturas completas basadas única y exclusivamente en productos de software libre. No sólo eso, sino que los arquitctos normalmente disponen de varias soluciones libres para cada una de las partes de su arquitectura.

Aún así, la plataforma de J2EE también tiene desventajas, algunas importantes:

» Depende de un único lenguaje: La plataforma J2EE depende exclusivamente del lenguaje Java. Sólo se puede utilizar este lenguaje para desarrollar aplicaciones lo que puede suponer un gran problema si nuestro equipo no

dispone de los conocimientos suficientes o tiene otras preferencias.

» Complejidad: Aunque no es una plataforma tan compleja como CORBA, no existe un VB .NET en Java. La creación de aplicaciones bajo J2EE requiere normalmente desarrolladores más experimentados que los necesarios para desarrollar bajo .NET

» Heterogeneidad: Existe una gran heterogeneidad en las soluciones de desarrollo. No existe en J2EE un simil a Visual Studio .NET. La gran cantidad de herramientas disponibles causa confusión dentro de los desarrolladores y puede crear dependencias dentro de las empresas.

Introducción a J2EE

Existe mucha confusión, sobre todo entre la gente alejada del mundo de Java, sobre lo que es en realidad J2EE. La confusión más habitual es pensar que J2EE es un producto concreto que distrubuye SUN Microsystems, como hace con su JDK, y que te puedes descargar desde su página web. Nada más lejos de la realidad. No existe un J2EE concreto, no puedes ir a la página web de SUN Microsystems y descargarte "el J2EE".

JSR

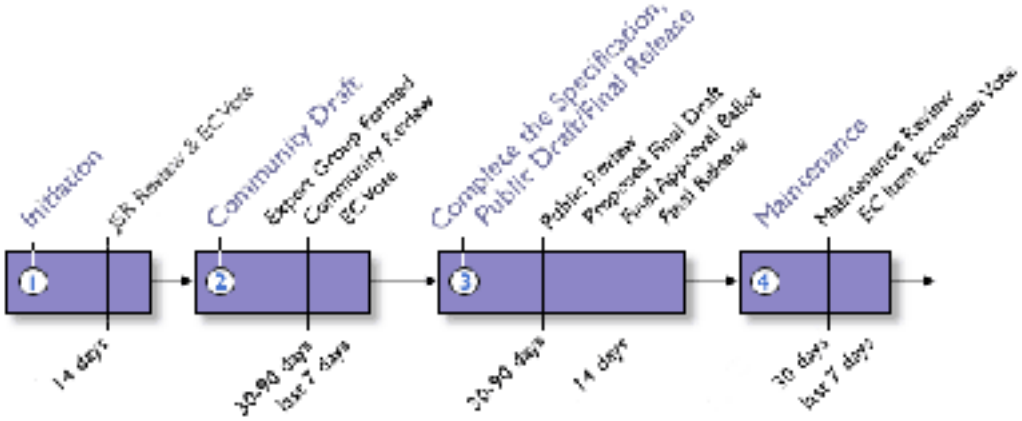
JSR es el acrónimo de *Java Specification Request*. Cuando una persona o entidad cree que es necesario la presencia de una determinada tecnología dentro de las plataformas basadas en Java, lo que hace es crear un JSR y presentarlo para su aprobación. Dentro de este documento se relata por que es necesaria dicha tecnología, por que no se pueden abordar los problemas que soluciona con las tecnologías existentes, etc.

Si dicha petición se aprueba entonces se crea una especificación, un documento en el cual se describe dicha tecnología, sus partes, las relaciones entre las mismas y los roles de las personas que usarán dicha tecnología. Además de la realización de este documento, el equipo encargado del desarrollo de la especificación ha de proporcionar un test de compatibilidad y una implememtación de referencia de los que hablaremos posteriormente.

JCP

Java, siempre fue criticado por ser única y exclusivamente de SUN. A raiz de todas esas críticas, SUN, el 8 de diciembre de 1998, decidió dar la posibilidad a todo el mundo de participar en la evolución de Java y de todas las plataformas que se basan en Java. Con esa intención se creó el JCP, que como hemos dicho es un organimo formado por alrededor de 500 empresas, asociaciones y particulares cuyo objetivo es asegurar la evolución de las plataformas basadas en Java.

Para entrar a formar parte del JCP las empresas e individuales han de pagar una cuota anual a SUN. Cualquiera puede participar en el desarrollo de cualquier parte de la plataforma, previo pago de esa cuota, y por supuesto, si el comité de la especificación en la que quiere participar considera que esa persona o entidad tiene los conocimientos necesarios para aportar algún beneficio.



1. Esquema del JCP

Una de las labores del JCP es la de controlar la evolución de las diferentes especificaciones que forman las plataformas basadas en Java. Este organismo es el encargado de decidir que especificaciones se aprueban y de controlar las fases por las que pasan.

Aunque la labor del JCP parece adecuada, lo cierto es que si se analiza desde el punto de vista del desarrollo del software libre nos encontramos con que expone una serie de limitaciones demasiado grandes:

» Cuotas: Para participar dentro de este organismo es necesario pagar una cuota de participación destinada a paliar los gastos administrativos de dicho organismo. Además, si alguien implementa alguna de las especificaciones es necesario que pase un test de compatibilidad. El problema radica en que para pasar este test hay que ser licenciatario de SUN lo que deriva en más dinero a pagar. Obviamente los más perjudicados por esto son los grupos de desarrollo de software libre y en general las asociaciones sin ánimo de lucro.

» El derecho de SUN: SUN tiene el derecho y privilegio de poder anular cualquier especificación si no es de su agrado.

» Patentes: Si alguna especificación tiene patentes dentro de ella, en caso de que se implemente dicha especificación será necesario que se respeten dichas patentes.

» Licencias: Por si fuera poco, las implementaciones no pueden cambiar la licencia de ninguna implementación. Por lo tanto, no es posible realizar implementaciones libres de especificaciones que tengan una licencia propietaria.

Esta claro que todas estas restricciones no hacen más que poner serias trabas al desarrollo de software libre. La fundación Apache, consciente de esto, exigió a SUN Microsystems la rectificación de la licencia que rige el funcionamiento del JCP y con motivo de esto llegaron a un acuerdo en Marzo del año 2002. A partir de Noviembre del año 2002 ha entrado a funcionar la versión 2.5 del JCP que presenta las siguientes novedades:

» Cuotas: Un comité se encargará de decidir si una asociación sin ánimo de

lucro, un grupo de software libre, universidad, etc., es apto para participar en el JCP o puede pasar los tests de compatibilidad. El propósito de esta limitación es intentar evitar en la medida de lo posible la diversificación de las implementaciones.

» El derecho de SUN: SUN Microsystems ha renunciado al derecho de veto que disponía sobre las especificaciones.

» Patentes: A partir de ahora, si una especificación tiene alguna patente, las implementaciones no están obligadas a respetar dicha patente.

» Licencias: La novedad más importante, sin duda alguna, es que ahora es posible implementar cualquier especificación con cualquier licencia, lo que permitirá la creación de implementaciones libres de muchas especificaciones que hasta ahora no permitían eso.

Aunque la especificación del lenguaje Java no se encuentra sujeta al proceso del JCP, lo cierto es que todos estos cambios han vuelto a hacer aparecer la luz dentro del mundo de desarrollo de software libre bajo la plataforma Java. Según palabras del vicepresidente de la fundación Apache, Jason Hunter, es posible que en el próximo año comiencen a aparecer JDKs compatibles con la especificación del lenguaje y totalmente libres.

J2EE

Después de ver los conceptos de JSR y JCP estamos preparados para definir lo que es J2EE. J2EE es una especificación, un JSR (concretamente el JSR-151[2]), que define una plataforma de desarrollo empresarial, a la que llamaremos la plataforma J2EE. La plataforma J2EE[3] está formada varios componentes:

» Un conjunto de especificaciones.

» Un test de compatibilidad, el J2EE Compatibility Test Suite (CTS).

» La implementación de referencia de J2EE.

» Un conjunto de guías de desarrollo y de prácticas aconsejadas denominadas J2EE BluePrints.

En las siguientes secciones veremos más a fondo cada una de estas partes.

J2EE, una especificación de especificaciones

Como hemos dicho, la plataforma J2EE está definida por una especificación en formato electrónico. Esta especificación se encuentra bajo el JSR-151, y en ella se definen, de manera muy general, las pautas, reglas y servicios que han de seguir y ofrecer los diferentes servidores de aplicaciones que quieran implementar la plataforma J2EE. Además de eso, define también las normas generales que han de cumplir los desarrolladores que quieran crear aplicaciones empresariales compatibles con J2EE y los diferentes roles que éstos pueden tomar.

Los servicios que han de ofrecer los servidores de aplicaciones que implementen la

plataforma J2EE están a su vez definidos por diferentes especificaciones. Estas especificaciones definen con mucho más detalle los diferentes componentes de los servidores de aplicaciones, como puedan ser un contenedor web, un servidor de mensajería, el sistema de seguridad, etc. De algún modo, se podría decir que la especificación J2EE engloba a un gran conjunto de especificaciones.

Por poner un ejemplo. A día de hoy, en la especificación de J2EE 1.4 se definen las siguientes especificaciones:

- JSR-109, (Servicios Web)
- JSR-101, (JAX-RPC, Java API for XML-based RPC)
- JSR-67, (JAXM, Java API for XML Messaging)
- JSR-93, (JAXR, Java API for XML Registries)
- JSR-77, (Configuración y control)
- JSR-88, (API de despliegue)
- JSR-115, (Interfaz de servicios de autorización)
- JSR-56, (JNLP, Ejecución remota de aplicaciones)
- JSR-112, (JCA 2.0, Arquitectura de conectores)
- JSR-152, (JSP 1.3, Java Server Pages)
- JSR-152, (Servlets 2.4)
- JSR-153, (EJB 2.1, Enterprise Java Beans)
- JSR-9XX, (JAXP 1.2, Soporte de esquemas XML)
- JSR-9XX, (JMS 1.1, API de mensajería)

Como se puede ver, todas estas especificaciones tienen asociado un JSR, regido por un comité de empresas, asociaciones o individuos que se aseguran de crear dichas especificaciones y de que vayan evolucionando. Como hemos dicho antes, cualquiera puede participar en la creación de estas especificaciones y por supuesto cualquiera puede descargarlas y leerlas cómodamente en su casa.

La gran importancia de toda esta enorme lista de especificacioens radica en que cuando utilizamos un servidor de aplicaciones que implementa la plataforma J2EE, los desarrolladores, obtenemos de manera automática todos estos servicios. Es decir, se pone a nuestra disposición una gran caja de herramientas que podemos aprovechar para realizar aplicaciones de una manera mucho más eficaz.

Por último, hay que señalar que cada una de estas especificaciones puede utilizarse perfectamente por separado, es más, seguro que muchos lo habréis hecho alguna vez. Por otra parte, cada una puede tener diferentes implementaciones que se pueden ofrecer como productos independientes, como por ejemplo Apache Tomcat (Servlets/JSP) o JORAM (JMS) y que por supuesto pueden tener la licencia que deseen, libre o

propietaria.

El test de compatibilidad de J2EE

Como vimos, cualquier especificación regida por un JSR tenía que proporcionar un test de compatibilidad. Estos test ayudan a que los fabricantes que implementen la especificación puedan probar su compatibilidad con la misma. En el caso de J2EE este test se denomina J2EE CTS (Compatibility Test Suite).

El CTS[4] es otra de las causas de confusión dentro del mundo de J2EE. Este test de compatilidad consta de una serie de pruebas, más de 15000, cuyo objetivo es asegurar que un producto, típicamente un servidor de aplicaciones, está conforme con la especificación de J2EE. La finalidad de este test es clara: asegurar que los productos tienen un mínimo de calidad, de modo que cumplan todas las normas de la especificación y con esto asegurar que no surjan implementaciones que desvirtuen a la plataforma. Pasando el test de compatibilidad los clientes estamos seguros de que un producto cumple con todas las normas que marca la especificación, y a su vez, los fabricantes obtienen una certificación de compatibilidad que pueden utilizar dentro de sus estrategias de marketing.

El problema surge a raiz de que para pasar este test de compatibilidad había que pagarle una cuota a SUN Microsystems. Esto creó, y está aún creando, grandes polémicas dentro del mundo del software libre ya que organizaciones sin ánimo de lucro como la Apache Software Foundation, tienen los medios suficientes para abordar dichas cuotas.

La confusión comienza cuando a raiz de esto algunas personas, que no conocen la plataforma, afirman que no es posible crear implementaciones de J2EE sin pagarle a SUN. Esto no es cierto. Lo que no es posible es crear implementaciones de J2EE certificadas sin haber pagado la cuota del test de certificación a SUN. Por supuesto que es posible crear implementaciones de J2EE, o de algunas de sus partes, que sean libres. Sin embargo, hasta el momento, estas implementaciones no se podían certificar como compatibles con J2EE sin pagar la cuota de certificación. Cabría preguntarse si realmente esto es tan importante.

Por poner un símil, se produce la misma situación que se da con las normas de calidad ISO y los organismos de certificación. Una empresa que fabrique coches puede decidir pasar estos test de calidad o no. Si los quiere pasar, tendrá que pagar una cuota a un organismo de certificación, por ejemplo AENOR, y después de pasarlos podrá presumir de ser compatible con las normas ISO de calidad. Si no los quiere pasar, no quiere decir que no pueda fabricar coches, ni tampoco quiere decir que sus coches sean malos, simplemente quiere decir que no ha podido o no ha querido pasar dichas pruebas de calidad.

Sea como sea, parece que el nuevo JCP ha satisfecho las exigencias de ambas partes y ya es posible pasar dichos test de compatibilidad sin necesidad de pagar las respectivas cuotas.

La implementación de referencia

Además de ofrecer unos tests de compatibilidad, cualquier especificación ha de ofrecer también una implementación de referencia de la tecnología. Esta implementación sirve para que los desarrolladores tengan un producto con el que empezar a desarrollar y que

al mismo tiempo puedan asegurarse de que estos desarrollos sean compatibles con la especificación.

La especificación de J2EE no es ninguna excepción y disponemos de una implementación de referencia, que se puede descargar desde la página de J2EE de SUN[5]. Esta implementación de referencia es la causa de que muchas personas piensen que J2EE es un producto que se puede descargar de la web de SUN y que posee una licencia muy restrictiva. Ahí está la equivocación, lo que estamos descargando es la implementación de referencia, que tendrá la licencia que SUN crea oportuna y sobre la que probablemente no seamos quienes para opinar si está bien o está mal. Por suerte para nosotros, existen otras implementaciones de la especificación que son absolutamente libres, y es de ellas de las que vamos a hablar en este artículo.

Sea como sea, utilizar esta implementación de referencia para el desarrollo de aplicaciones nos garantiza que estamos desarrollando conforme a la especificación sin utilizar librerías específicas de terceros fabricantes. Además, la implementación de referencia nos permite probar las últimas novedades de la especificación mucho antes de que se encuentren presentes en el resto de productos. Aún así, la propia SUN avisa de que esta implementación no es apta para entornos de producción, y sólo se debería utilizar para desarrollo.

J2EE BluePrints

Por último, y para finalizar con este recorrido por las partes de la plataforma J2EE, tenemos los J2EE BluePrints[6]. Los J2EE BluePrints son un conjunto de documentos, que podemos descargar desde la web de SUN Microsystems o encargar copias impresas de los mismos. Estos documentos definen patrones de diseño y prácticas recomendadas en la creación de aplicaciones utilizando J2EE. Dentro de ellos podemos encontrar como crear aplicaciones con JSP y Servlets de n-capas, que patrones de diseño utilizar al desarrollar con Enterprise Java Beans (EJB), etc.

El modelo de desarrollo de J2EE

Para finalizar con esta breve descripción de la plataforma J2EE vamos a ver algunos conceptos básicos sobre el modelo de desarrollo de aplicaciones bajo esta plataforma. Realmente abordar el modelo de desarrollo de J2EE nos llevaría varios libros completos y aún nos dejaríamos partes sin tratar, es por ello que en los siguientes párrafos se va ver por encima como sería una arquitectura típica basada en esta tecnología.

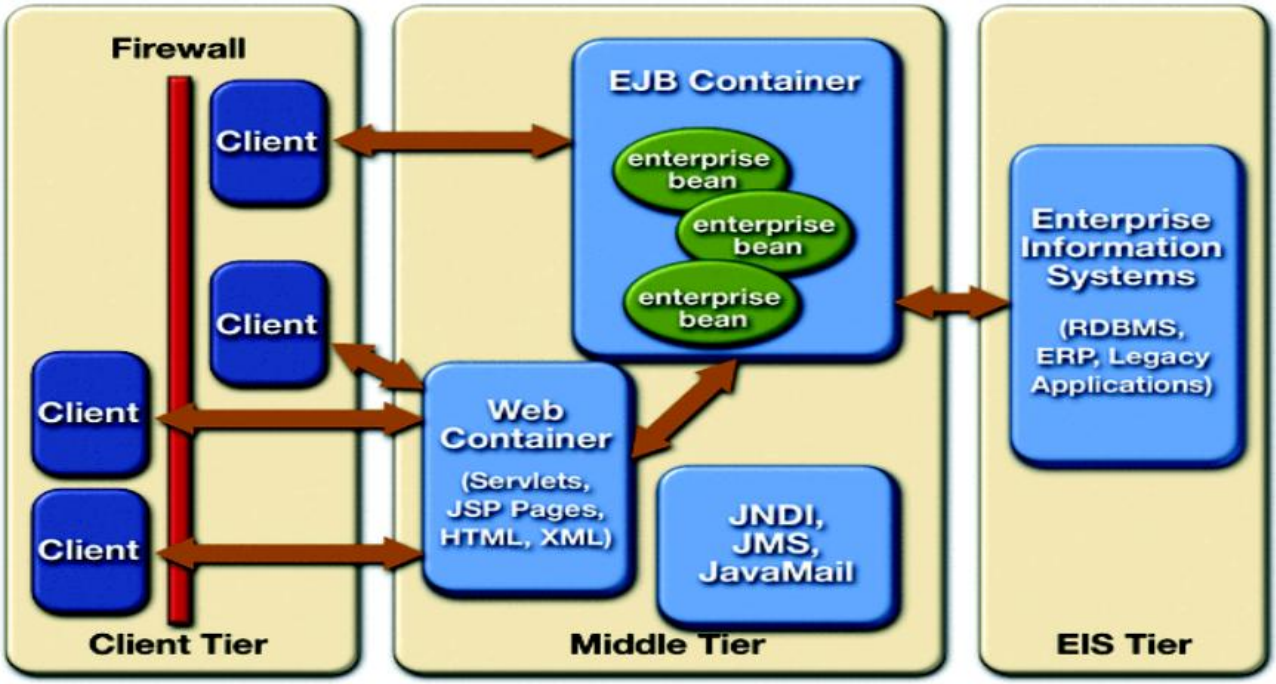
La plataforma de J2EE define un modelo de programación encaminado a la creación de aplicaciones basadas en n-capas. Tipícamente una aplicación puede tener cinco capas diferentes:

- Capa de cliente: Representa el interfaz de usuario que maneja el cliente.
- Capa de presentación: Representa el conjunto de componentes que generan el la información que se representará en el interfaz de usuario del cliente. Típicamente se creará a través de componentes basados en Servlets y JSP.
- Capa de lógica de negocio: Contiene nuestros componentes de negocio reutilizables. Normalmente se forma a partir de componentes EJB.
- Capa de integración: Aquí se encuentran componentes que nos permiten

hacer más transparente el acceso a la capa de sistemas de información. Por ejemplo este es el lugar idóneo para implementar una lógica de objetos de acceso a datos, DAO (Data Access Objects).

» Capa de sistemas de información: Esta capa engloba a nuestros sistemas de información : bases de datos relacionales, bases de datos orientadas a objetos, sistemas legacy, etc.

Las ventajas de un modelo como este son muy importantes. Al tener las capas separadas tenemos que existe poco acoplamiento entre las mismas, de modo que es mucho más fácil hacer modificaciones en ellas sin que interfieran en las demás. Todo esto redunda en la obtención de mejoras en cuanto a mantenibilidad, extensibilidad y reutilización de componentes. Otra de las ventajas que obtenemos es que se promueve la heterogeneidad de los clientes ya que añadir nuevos tipos de cliente (móviles, set-top-boxes, PCs, etc.) se reduce a añadir nuevas capas de interfaz de usuario y presentación, sin tener que modificar todo el resto de capas.



2. Esquema de un modelo mixto de 3 y 4 capas

En la figura 2 se puede ver lo que sería una posible arquitectura J2EE. El modelo que aparece en la figura está dividido en varias capas, con una separación clara entre presentación, lógica de negocio y sistemas de información empresariales. En este caso además, podemos ver como se trata de un modelo mixto. En la parte de arriba tenemos que se recorre un camino por una estructura de tres capas (aplicación - lógica de negocio - sistemas de información), mientras que por el segundo camino recorremos una estructura de cuatro capas (aplicación - lógica de presentación - lógica de negocio - sistemas de información).

Este modelo es sólo un ejemplo, en muchos casos hasta puede que sea demasiado complejo y nos conformemos con utilizar únicamente la capa de Servlets/JSP para acceder a nuestros sistemas de información, en otros casos prescindiremos de la capa web, en otros casos utilizaremos servicios web para acceder a la lógica de negocio, o crearemos capas de persistencia intermedia con patrones de acceso a datos. Las

combinaciones son prácticamente ilimitadas.

Como ya hemos dicho, el modelo de desarrollo con J2EE está basado en componentes reutilizables, con el objetivo de aumentar la reusabilidad de las aplicaciones. Estos componentes, además, gracias a las especificaciones, son intercambiables entre servidores de aplicaciones, por lo que la portabilidad de nuestros desarrollos es máxima. Si hay un tipo de componentes que requieren una atención especial estos son los Enterprise Java Beans. Se trata de objetos distribuidos, que como hemos dicho contienen la lógica de negocio de nuestras aplicaciones y que hacen transparente al programador operaciones como la persistencia, la seguridad, la gestión de transacciones, etc.

En cuanto a la integración con otros sistemas, gran parte del modelo de J2EE (en especial los EJB) está basado en CORBA lo que quiere decir que es posible comunicarse sin ningún tipo de problema a traves de IIOP con otras aplicaciones creadas en otros lenguajes diferentes de Java y viceversa. Las posibilidades de integración todavía son mayores si consideramos que a partir de enero del 2003 (cuando salga J2EE 1.4) cualquier EJB que hayamos creado se podrá exponer como un servicio web. Por último otra tecnología que merece la pena nombrar es la de los conectores, que por explicarlo de una manera clara, son un puente entre J2EE y sistemas legacy (por ejemplo un conjunto de ficheros de datos en COBOL), Los conectores tienen un API estándar que nos permite acceder a estos sistemas legacy de una manera transparente y sin apenas esfuerzo.

Para finalizar, otra de las grandes ventajas de J2EE viene del hecho de que está basado en Java, ya que la variedad de clientes que pueden conectarse a una arquitectura J2EE es inmensa, desde aplicaciones de escritorio, hasta móviles, pasando por televisiones, PDAs, etc., cualquiera de estos productos puede conectarse ya sea a través de RMI-IIOP, HTTP, SOAP, XML-RPC, etc., lo que nos da una gran gama de posibilidades en cuanto a la conectividad de nuestros sistemas.

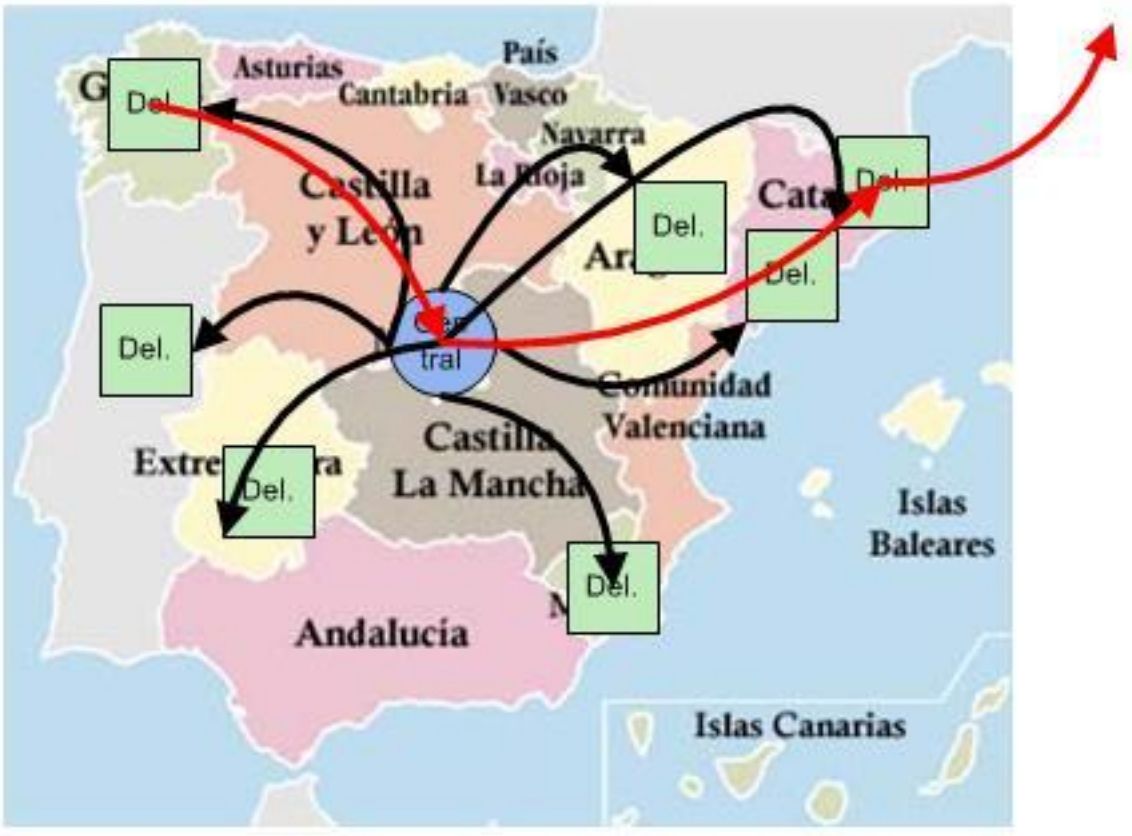
Soluciones libres con J2EE / Caso de uso

Hasta aquí termina el tostón sobre J2EE y arquitectura. Espero que todos todavía sigáis aquí. A partir de ahora vamos a ir analizando las diferentes partes que pueden formar una arquitectura empresarial típica. Bueno, realmente la arquitectura empresarial no va a ser tan típica sino que va a ser muy completa y compleja. Realmente va a ser poco probable que nos encontremos alguna empresa que tenga semejante cantidad de requisitos, y seguramente nos llegaría con sólamente una parte de la arquitectura que veremos, pero el objetivo de este ejemplo es mostrar la calidad de las soluciones que se pueden crear utilizando sólamente software libre bajo la plataforma J2EE.

Para que entendáis mucho mejor la arquitectura que vamos a realizar vamos a utilizar un caso de uso. Vamos a suponer una empresa imaginaria, digamos una empresa transportista que opera en la península. Vamos a suponer también que esta empresa ha presentado a concurso la creación de su arquitectura empresarial y que a esta han optado varias consultoras. La primera consultora ofrece una solución basada totalmente en productos Microsoft, la segunda consultora ofrece una solución basada en J2EE y productos propietarios para componentes como servidores de aplicaciones, entornos de desarrollo, etc. Por último, la tercera consultora va a ofrecer una solución basada íntegramente en productos libres.

Nosotros vamos a tomar el papel de esta última consultora y describiremos detalladamente la oferta que vamos a realizar. Es importante tener en cuenta que estamos ante un problema de bastante magnitud. En la figura siguiente se puede apreciar

un diagrama de la empresa con su central y sus delegaciones.



3. Caso de uso

Como veis la empresa tiene varias delegaciones por toda España y se producirá un flujo constante de información entre todos los puntos. Toda la información pasará por la central que se encuentra en Madrid. Como podréis intuir, tenemos unos altos requerimientos de persistencia (no se pueden perder paquetes), de seguridad (es muy importante que nadie nos robe o modifique información), de transacciones (toda la información ha de permanecer en un estado consistente), de escalabilidad (hoy tenemos 7 delegaciones pero mañana pueden ser 30 y el sistema tiene que mantener la misma calidad de servicio), y muchos otros requerimientos en los que no vamos a parar.

Otro aspecto muy importante es que se requiere el control absoluto del flujo de información (Workflow) del sistema. Por ejemplo, imaginad que se envia un paquete desde A Coruña hasta Barcelona. Este paquete pasará primero por Madrid, después irá a Barcelona, después puede que nos envien una respuesta primero a Madrid y después desde ahí a Barcelona. Todo este flujo de información se realiza automáticamente en base a eventos que van sucediendo en el sistema. Esto es controlar el workflow. Este tipo de control es muy importante ya que nos permite tener controlado en cada momento todo lo que está pasando en nuestro sistema. De este modo, si se produce algún error podremos realizar algún mecanismo de vuelta atrás y recuperar algún estado consistente de la información.

Obviamente, intentar proponer una arquitectura empresarial sin utilizar una plataforma de desarrollo empresarial sería una locura ya que todos los mecanismos de persistencia, transacciones, etc., los tendríamos que crear manualmente. Probablemente cuando finalizásemos de crear sólo esa parte, nuestra competencia ya habría vendido el proyecto

no sólo a esta sino a treinta empresaas más del sector. Nosotros vamos a utilizar la plataforma J2EE y además sólo software libre para que nuestra solución se beneficie de las ventajas asociadas al uso de este software. En las siguientes secciones iremos viendo más en detalle este caso de uso.

Servidores de aplicaciones

Los servidores de aplicaciones son el corazón de la plataforma J2EE. En ellos residirán todos los componentes de nuestra empresa, ya sean objetos distribuidos accesibles remotamente, páginas web, o incluso aplicaciones completas accesibles utilizando Java Web Start. Un servidor de aplicaciones que implemente la plataforma J2EE nos tendrá que ofrecer de manera automática todas las especificaciones que vimos antes. Esto es muy importante, ya que por arte de magia el desarrollador se encuentra con una gran caja de herramientas. De este modo, en un servidor de aplicaciones normal tendremos un contenedor de servlets, un contenedor de EJBs, sistemas de mensajería, y un gran número de herramientas que nos ayudarán a incrementar la productividad de nuestro equipo.

Sin duda, la pieza más importante dentro de un servidor de aplicaciones es el contenedor de EJBs. Los EJBs son objetos reutilizables que contienen la lógica de negocio de nuestro sistema. Los contenedores de EJBs son servidores que se encargan de controlar todos estos componentes, esto es muy importante ya que se automatizan tareas como la gestión del ciclo de vida, la gestión de las transacciones, la gestión de persistencia, etc. Los desarrolladores, de este modo, no tienen que centrarse en crear servicios de bajo nivel y pueden centrarse en la creación de lógica de negocio empresarial.

En la actualidad el mercado de los servidores de aplicaciones es uno de los más activos y todas las empresas están luchando ferozmente por ser los líderes del sector, como por ejemplo en el caso de BEA WebLogic e IBM WebSphere que mantienen una lucha encarnizada por ver quien es el servidor más utilizado. Estos dos servidores son realmente maravillas tecnológicas pero su precio hace que muchas empresas no se puedan permitir el lujo de comprar sus licencias. Por suerte existen una serie de servidores de aplicaciones absolutamente libres que no tienen demasiado que envidiarles.

JBoss

JBoss es el servidor de aplicaciones libre por excelencia. Su licencia es LGPL y está implementado al 100% en Java. Tiene más de 150.000 descargas mensuales lo que le convierte en el servidor de aplicaciones más descargado del mundo. JBoss está abalado por un grupo de desarrollo formado por arquitectos con gran experiencia y repartido por todo el globo.

4. JBoss es el servidor de aplicaciones que está más de moda actualmente

La última versión de JBoss[7], la 3.0, soporta gran parte del estandar J2EE 1.3. Entre sus características destacan su nueva base sobre la especificación JMX, el soporte de EJB 2.0 y la posibilidad de crear clusters. Otra de las cosas más interesantes de este servidor de aplicaciones es que lo podemos descargar en versiones que integran Apache Tomcat o Jetty como contenedores web.

Pero quizás lo más interesante de JBoss es que rotúndamente podemos afirmar que es un gran ejemplo para toda la comunidad de software libre. El grupo JBoss es una entidad con ánimo de lucro que está obteniendo grandes beneficios a partir de la venta de documentación, consultoría, formación y seminarios y servicios en general. Fruto de toda esta actividad, recientemente JBoss ha llegado a un acuerdo para implantar sistemas empresariales dentro del gobierno de Noruega.

Por poner algunos datos más, recientemente la empresa TogetherSoft realizó una encuesta entre sus clientes en la que se encontró que el 43% de los encuestados estaban utilizando JBoss como desarrollo, siendo el siguiente servidor de aplicaciones utilizado BEA WebLogic con un 26%. Aunque estos datos se refieren únicamente a entornos de desarrollo no dejan de ser llamativos. Además, JBoss está siendo alabado en todo el mundo y prueba de ello es la elección por parte de los lectores de las prestigiosas revistas Java Developer's Journal y JavaWorld como mejor servidor de aplicaciones del año 2001.

A principios de Diciembre del 2002, JBoss recibió la luz verde por parte de Sun para comenzar su proceso de certificación como compatible con la plataforma J2EE 1.4. JBoss será el primer servidor de aplicaciones libre que reciba el galardón de certificado y demuestra que el proceso de apertura de Java está comenzando a dar sus frutos.

JOnAS

Aunque quizás no tan conocido como JBoss este servidor de aplicaciones, por sus características, es uno de los proyectos más ambiciosos del mundo del software libre en Java y no sería de extrañar que pronto superase a JBoss en aceptación.

JOnAS[8] es la implementación de la especificación de EJB realizada por el consorcio Object, formado entre otros por Bull, France Telecom e Inria, y disponible bajo licencia LGPL. Más interesante que esto es que JOnAS forma parte de la iniciativa OpenWeb lanzada por Bull en colaboración con empresas como France Télécom e Inria. El objetivo de esta iniciativa es la creación de middleware libre, que facilite la creación de sistemas distribuidos. Esta organización basa todo su trabajo en la creación de implementaciones libres de las especificaciones más notables de J2EE y CORBA. Por todo esto y por el apoyo de grandes empresas, como Bull o France Telecom, el futuro de JOnAS parece realmente muy prometedor.

JOnAS surgió en el 1999 y actualmente se encuentra ya por la versión 2.5. Entre sus características más notables está el soporte de EJB 2.0 (por ahora no CMP 2.0), la integración de productos de software libre como Apache Tomcat como contenedor web o JORAM como sistema de mensajería. Además tiene la ventaja de que incluye herramientas adicionales como una utilidad de administración o un generador automático de EJBs. La documentación de JOnAS está disponible gratuitamente en su página web.

OpenEJB

Estríctametente, OpenEJB[9] no es un servidor de aplicaciones. OpenEJB es un contenedor de EJBs pero lo hemos incluido dentro del apartado de servidores de aplicaciones ya que compaginado con otros productos de la empresa que lo desarrolla, Exolab, puede llegar a formar soluciones realmente muy competitivas. Exolab, es una empresa que se dedica a la creación de diferentes productos de software libre como son OpenJMS u OpenORB. Forma parte del Java Community Process y de la Apache Software Foundation. Todos sus productos se distribuyen bajo una licencia propia equivalente a la licencia de Apache reconocida como Open Source por la Free Software Foundation.

OpenEJB soporta parte de la especificación 2.0 de EJB aunque un aspecto muy interesante de este servidor de aplicaciones es que su desarrollo está liderado por Richard Monson-Haefel, gurú de J2EE y escritor del libro Enterprise JavaBeans, todo un éxito de ventas y del que se planea ya su cuarta edición.

También cabe destacar que al tratarse de un contenedor web, este producto es mucho más ligero que JBoss o JOnAS que son servidores de aplicaciones completos. Con la posibilidad introducida en la última versión de OpenEJB, en la que se permite la integración con Apache Tomcat, se introduce una posibilidad muy interesante ya que esto permite a desarrolladores web integrar dentro de un sistema un contenedor de EJBs con las ventajas que esto supone.

Caso de uso (I)

Vamos a comenzar la arquitectura empresarial de nuestro caso de uso instalando un servidor de aplicaciones. Concretamente vamos a instalar un clúster de servidores JBoss, de modo que podamos aprovechar las características de balanceo de carga que nos ofrecen estos sistemas. Nuestro clúster se localizará en la central de Madrid y desde ahí se servirán las peticiones de los cliente situados en las diferentes delegaciones.

5. Hemos instalado nuestro servidor de aplicaciones

Como se puede apreciar en la figura 5, dentro del servidor de aplicaciones estará nuestro contenedor de EJBs donde residirán los diferentes EJBs de nuestro sistema que contienen la lógica de negocio. Estos componentes reciben de manera automática una serie de servicios que proporciona el contenedor: transacciones, persistencia, seguridad, etc. Además, estos componentes pueden aprovecharse de los conectores que proporciona el servidor de aplicaciones para acceder a sistemas legacy, por ejemplo para acceder a ficheros COBOL, acceder a Tuxedo, etc.

Sistemas de mensajería

Los sistemas de mensajería empresarial proporcionan servicios adicionales que los sistemas tradicionales síncronos no pueden ofrecer. Algunas de las características más interesantes de estos sistemas son :

» Funcionamiento síncrono o asíncrono: En modo asíncrono el emisor del mensaje no tiene por que esperar a que éste llegue a su destino. El emisor emite el mensaje y continua trabajando, el receptor lo recibe y a partir de ahí puede realizar numerosas tareas con el, tales como reprocesarolo, reenviarlo en base a alguna cadena de eventos, etc. Después de realizar este proceso el receptor, si lo considera necesario, puede avisar al emisor de que el mensaje se ha tratado correctamente.

» Fiabilidad: Si así lo desea, el emisor puede tener asegurado que el mensaje siempre llegará a su destino aunque éste no esté disponible. Esto se consigue a través del uso de un intermediario, que tradicionalmente se conoce como MOM (Messaging Oriented Middleware). El MOM es el encargado de recoger los mensajes enviados por el emisor y redireccionarlos al destino o destinos

adecuados. En caso de que un destino no se encuentre disponible el mensaje se persiste en algún medio de almacenamiento secundario y se reenviará posteriormente. De este modo, el emisor tiene garantizado que el mensaje llegará a su destino aunque este no se encuentre disponible.

» Escaso acoplamiento: Las capas de cliente y servidor están completamente aisladas lo que redunda en una mayor mantenibilidad, extensibilidad, reusabilidad, etc.

» Múltiples destinos: El intermediario puede encargarse de enviar los mensajes a múltiples destinos si fuese necesario. Este modo de comunicación se conoce como publicación/subscripción y en él se crean tópicos a los que múltiples subscriptores se subscriben. Cuando llega un mensaje de un emisor a un tópico determinado, el MOM se encarga de redireccionar ese mensaje a todos los destinos subscritos a ese tópico.

El API JMS permite el desarrollo de aplicaciones Java que utilicen sistemas de mensajería de una manera portable. JMS hace transparente al desarrollador el servidor de mensajería utilizado del mismo modo que JDBC hace transparente el acceso a bases de datos.

JORAM

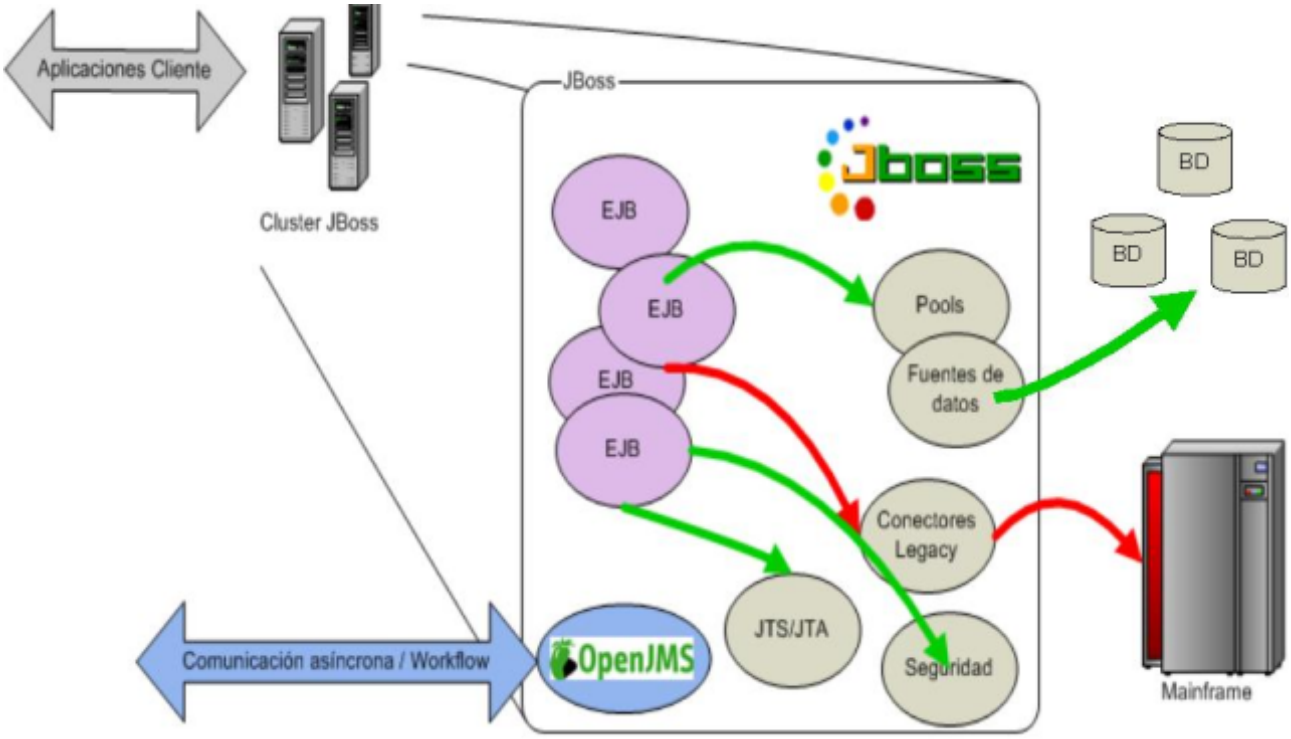
JORAM JORAM[10] es una implementación libre de la especificación JMS 1.1. Entre sus características más importantes están el soporte de los dominios de mensajería punto a punto y publicación/subscripción, selección de mensajes, temporizadores de envío retardado, posibilidad de destinos distribuidos o una implementación de JNDI 1.1.2.

JORAM al igual que JOnAS es un producto desarrollado por el consorcio ObjectWeb y con licencia OPL (Open Public License).JORAM implementa métodos para poder realizar las operaciones JMS utilizando los servicios proporcionados por los servidores de aplicaciones, viene integrado de serie con el servidor de aplicaciones JOnAS y puede integrarse con otros servidores de aplicaciones o funcionar por separado.

OpenJMS

OpenJMS OpenJMS[11] es un servidor de mensajería que está siendo desarrollado por Exolab, la misma empresa que se encarga de OpenEJB. Se integra a la perfección con dicho servidor de aplicaciones aunque también puede funcionar por separado. Tiene una licencia estilo BSD y las características son las típicas de estos productos : varios tipos de mensajería, envio garantizado, herramientas gráficas de administración, filtros de mensajes, persistencia con JDBC, etc.

Caso de uso (II)



6. Ahora nuestro modelo soporta mensajería asíncrona gracias a JMS

En la figura 6 se puede ver como queda nuestra arquitectura empresarial después de añadir el servidor de mensajería OpenJMS. Realmente, podríamos haber utilizado JBoss MQ que es el sistema de mensajería que trae incorporado JBoss, sin embargo, hemos optado por colocar OpenJMS para mostrar la flexibilidad de los servidores de aplicaciones.

Gracias a un API llamado JMX que están adoptando todos los servidores de aplicaciones, es posible quitar partes de dichos servidores para incluir otros sistemas, y todo de una manera totalmente transparente. En nuestro ejemplo, hemos retirado el servidor de mensajería de JBoss y hemos incrustado OpenJMS, que será el subsistema que se encargue de controlar toda la comunicación asíncrona con el exterior.

Queda pro ver aún como vamos a vender este sistema de mensajería, para ello vamos a volver sobre el ejemplo de los pedidos. Imaginemos que se realiza un pedido de A Coruña a Barcelona. El pedido saldrá desde A Coruña y llegará a la central de Madrid, ahí podría ser recogido por nuestro sistema de mensajería, de este modo el sistema que envió el mensaje desde A Coruña podrá seguir trabajando. Una vez en Madrid, el pedido puede modificarse como sea conveniente y reenviarse a Barcelona. En Barcelona puede que sea necesario enviar una respuesta que se procesará del mismo modo. Para finalizar, la central de Madrid podría enviar un mensaje de manera asíncrona a la delegación de A Coruña para indicarles que todo el proceso del envio se ha realizado de manera correcta.

Como veis, todo el proceso se ha realizado de manera asíncrona, sin bloquear a los clientes, y lo que también es muy importante, de manera automática. Todo el flujo de mensajes se ha realizado automáticamente y éstos han pasado una serie de filtros de reelaboración también de manera automática. En todo momento sabemos donde se encuentra un mensaje por lo que si se produce algún error en el sistema podemos lanzar un mensaje de aviso y realizar las acciones correspondientes como corregirlo, reenviarlo, etc. Esto que estamos haciendo se denomina controlar el Workflow de nuestras aplicaciones y como hemos visto los sistemas de mensajería son un modo de realizar

este proceso.

Contenedores web

Los contenedores web son la puerta de las aplicaciones Java a la World Wide Web. Son servidores, escritos normalmente en Java, que son capaces de recibir peticiones HTTP y ejecutar las clases de Java (que a su vez utilizarán recursos tales como páginas HTML e imágenes) indicadas para darles respuesta, formando lo que se llama una aplicación web. Además, aunque casi todos incluyen un pequeño servidor web que les permite funcionar en solitario, normalmente pueden ser integrados con los servidores más conocidos como Apache o IIS, de modo que estos últimos se encarguen de realizar el procesado del contenido estático, labor para la que están optimizados.

Las aplicaciones web realizadas en Java pueden tomar dos formas: Servlets y páginas JSP (Java Server Pages). Las especificaciones de estas tecnologías forman parte de J2EE, y actualmente están en sus versiones 2.3 y 1.2 respectivamente. El principal cometido de estas tecnologías es el de crear páginas web dinámicas, es decir, páginas web que se generan en el servidor en base a nuestra lógica de negocio. Tanto los servlets como las páginas JSP aprovechan todas las posibilidades de Java y por lo tanto, además de poder aprovechar la flexibilidad que ofrece un lenguaje de programación, heredan todas sus ventajas en forma de reutilización, mantenibilidad, extensibilidad, etc. jsp El de los contenedores web es sin duda uno de los apartados donde el software libre empresarial escrito en Java se ha hecho más famoso, en gran medida gracias a la joya de la corona, Apache Tomcat.

Apache Tomcat

Apache Tomcat[12] es sin lugar a dudas el proyecto de software libre más famoso escrito en Java. Creado a partir de código donado por SUN Microsystems a la Apache Software Foundation, ha crecido hasta convertirse en la implementación oficial de referencia de Servlets y JSP sustituyendo a la implementación de SUN original. Esto lo convierte, obviamente, en el contenedor sobre el que todos los demás prueban su adhesión a las especificaciones.

Aunque siempre se le ha acusado de un rendimiento pobre, el caso es que a partir de las nuevas versiones de la serie 4.x, creadas para cumplir las especificaciones 2.3 de Servlets y 1.2 de JSP, Tomcat se ha reescrito entero lo que mejora considerablemente su rendimiento. Quizás siga sin ser la opción más indicada para una aplicación con miles de clientes simultáneos, pero ningún servidor en solitario es adecuado para una carga así. En todos los demás casos su rendimiento es más que aceptable.

Por supuesto puede integrarse sin demasiados problemas con el servidor web Apache, y para aplicaciones empresariales grandes con servidores de aplicaciones como JBoss o JOnAS. Asimismo puede integrarse con muchos otros productos con licencias propietarias. Es más, sin ir más lejos, Tomcat es el contenedor web que más se está utilizando en servidores de aplicaciones comerciales, donde gigantes como IBM con su todopoderoso WebSphere han elegido a Tomcat como contenedor web.

Otra de las muestras de lo estandarizado del uso de Tomcat es su utilización como servidor de Servlets y JSP por parte de los entornos de desarrollo, tanto libres como comerciales. Ejemplos de esto son Borland JBuilder o Eclipse que integran un servidor Tomcat para poder desarrollar nuestras páginas JSP y Servlets.

Jetty

Jetty[13] es el otro gran contenedor web libre del que dispone el mercado. En marcha desde 1995, donde actuaba ya como servidor web tradicional, este contenedor creado originalmente en Australia es la mayor competencia de Tomcat, aunque el menor tamaño de su comunidad y de la publicidad que recibe lo hacen pasar más desapercibido. Al igual que Tomcat, también cumple con las especificaciones más actuales de Servlets y JSP.

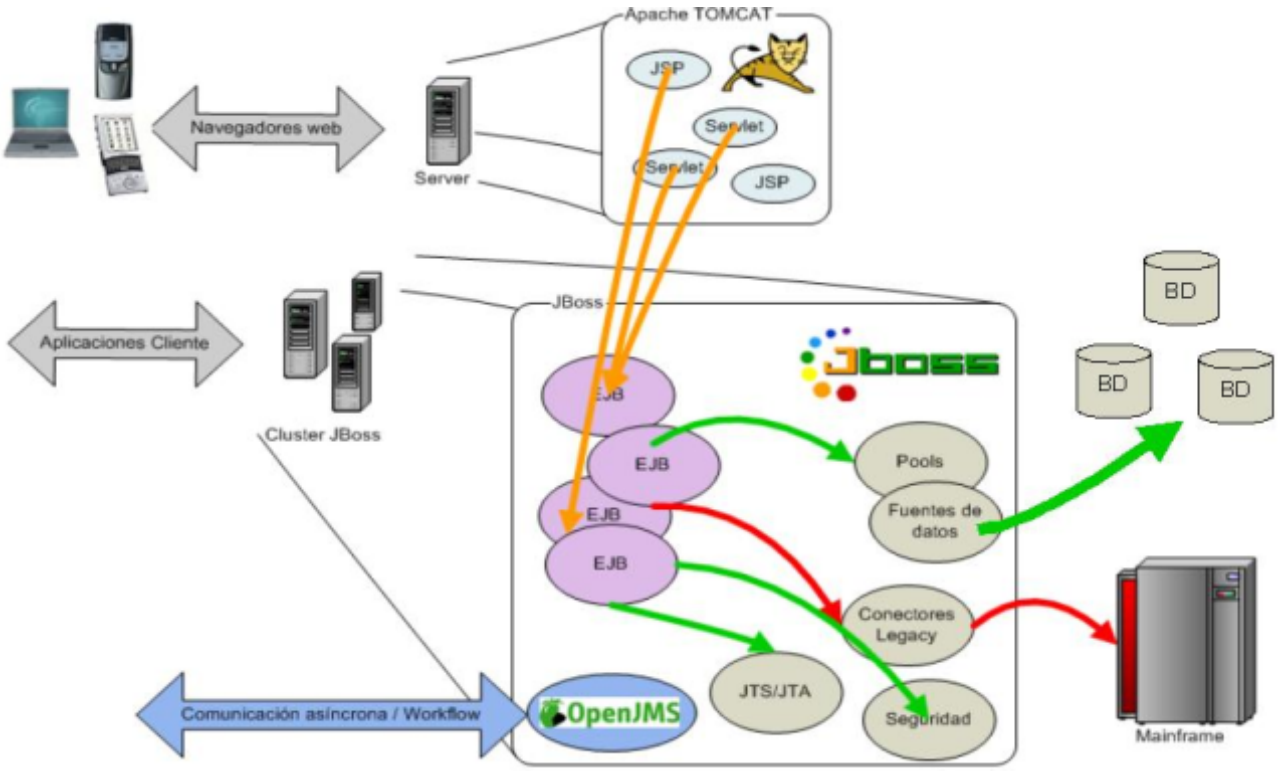
Jetty incluye un servidor HTTP bastante aceptable, y aunque puede integrarse con Apache, su rendimiento está mejor valorado que el de Tomcat. Desde la última versión de JBoss, es el contenedor web que viene por defecto con este popular servidor de aplicaciones por lo que es de esperar que su uso se vaya incrementando. Jetty además se integra con gran cantidad de productos libres y está siendo ampliamente utilizado en proyectos como Tapestry, Avalon o Maven. Su licencia es compatible con la Artistic License.

Enhydra

Enhydra[14] es un proyecto bastante maduro. Creado originalmente por la empresa Lutris, fue abandonado cuando ésta decidió dedicarse al desarrollo de proyectos comerciales. Por fortuna, ha vuelto a la carga de manos del consorcio ObjectWeb que ha recogido el relevo de todos sus desarrollos. Obviamente, Enhydra se integra a la perfección con productos como JOnAS y su licencia es una mezcla de varios tipos (BSD, Apache, LGPL, etc.) En la última versión, Enhydra ha tomado como base el código de Apache Tomcat 4.

Quizás una de las cosas más interesantes de Enhydra es su proyecto Enhydra MicroEdition. Este proyecto pretende ofrecer una serie de productos, que tradicionalmente tendríamos en un servidor, destinados hacia pequeños dispositivos. Así podemos utilizar sistemas como kXML o kSOAP para utilizar tanto SOAP como XML en nuestros teléfonos móviles, o instalar un servidor web como kHTTP o incluso transformar nuestro pequeño dispositivo en un nodo de mensajería utilizando kJMS.

Caso de uso (III)



7. Con tomcat ya tenemos nuestra capa de presentación

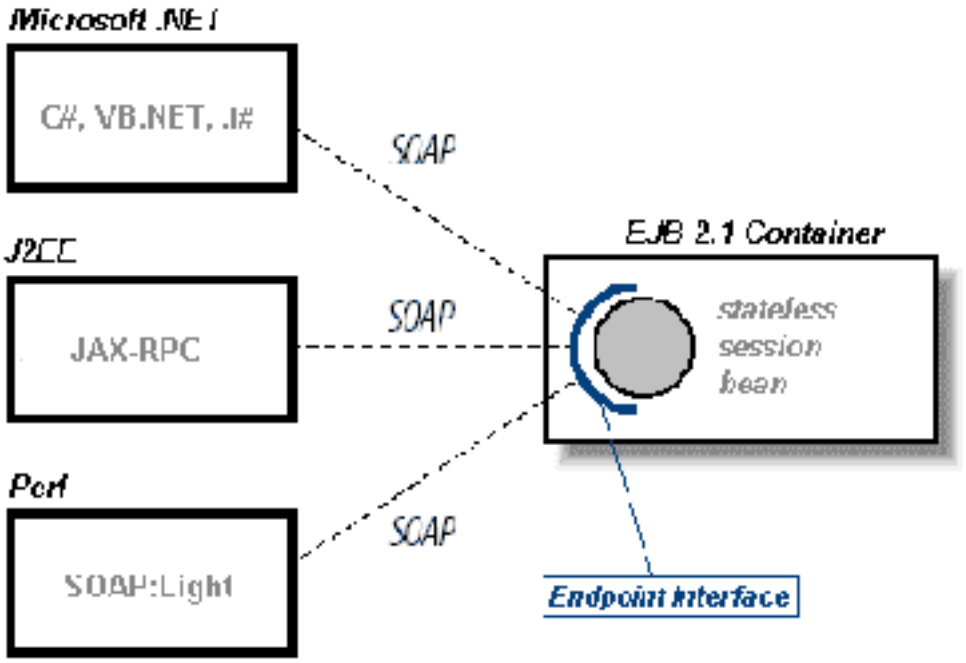
Vamos a añadir a nuestra arquitectura un contenedor web. En este caso utilizaremos Apache Tomcat, por ser el más extendido. Como se puede apreciar en la figura 7, hemos decidido instalar Tomcat en un servidor propio, que separaremos de nuestro servidor de aplicaciones (probablemente instalándolo en la DMZ de nuestra empresa), para tratar de minimizar el efecto de posibles ataques a nuestro sistema.

Dentro de Tomcat se situarán nuestras páginas web dinámicas que accederán a nuestra lógica de negocio y a partir de los datos obtenidos de la lógica, generarán páginas que llegarán a múltiples tipos de clientes, que podrán ver la información en sus navegadores web. Como veis, hemos ampliado nuestro modelo original de tres capas a un modelo ya de cuatro capas, que proporciona una mejor separación entre la lógica y la presentación.

Servicios webApache Axis

Aunque hasta ahora tenemos una arquitectura que colmaría las necesidades de la mayor parte de empresas, en la actualidad no seríamos capaces de venderla utilizásemos servicios web.

Ironías aparte, los servicios web son sin lugar a dudas la tecnología más de moda en la actualidad, aunque la realidad es que ya existían desde hace muchísimos años en las empresas, principalmente en la forma de protococolos de mensajes y comunicación propietarios. Un ejemplo claro es el estándar HL7 utilizado en los sistemas médicos. La verdadera importancia de los servicios web es que se han utilizado tecnologías estándar como HTTP, XML o SOAP para homogeneizar el modo de intercomunicación de las aplicaciones dentro de los sistemas empresariales.



8. Los servicios web nos permiten integrar sistemas heterogéneos

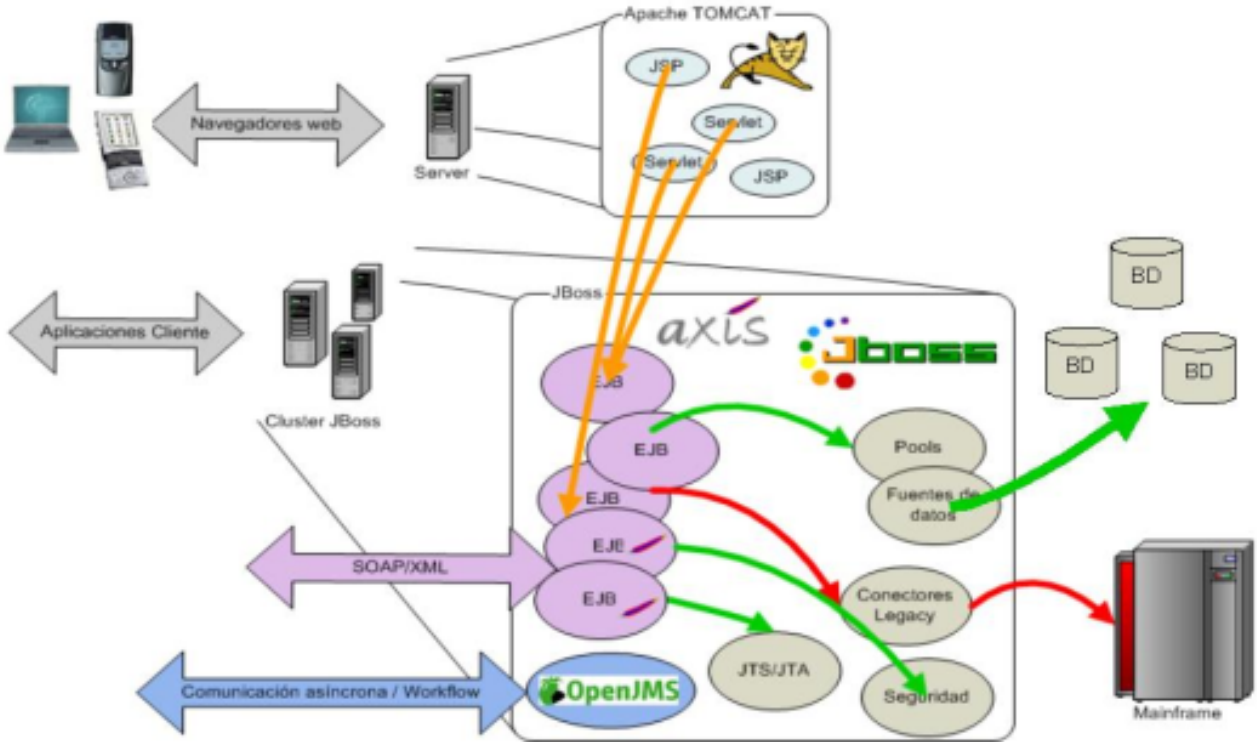
Las ventajas de los servicios web frente a sistemas de comunicación tradicionales son bastantes: independencia de la plataforma, independencia del lenguaje, uso de tecnologías y protocolos estándar al tiempo que promueven el aislamiento de las diferentes capas de los sistemas empresariales lo que redunda en una mayor extensibilidad, flexibilidad, etc. Tradicionalmente los servicios web se basan en SOAP bajo HTTP como protocolo de transporte.

En Java existe una herramienta para el manejo de servicios web que se está utilizando ampliamente, Apache Axis[15]. Este producto puede funcionar de dos modos diferentes. Por una parte, puede actuar como un motor SOAP, es decir, un conjunto de librerías que podemos utilizar desde nuestros programas para acceder a servicios web. Por la otra, Axis puede funcionar como servidor independiente de modo que podamos registrar en él diferentes servicios que serán accesibles desde el exterior.

a versión actual está escrita en Java aunque se planea realizar una implementación en C++ de la parte de cliente. Además de esto Axis incluye un servidor simple, soporte de WSDL, un generador de clases Java a partir del WSDL y un monitor de paquetes TCP/IP. Además, por si no fuera poco, Apache Axis se integra a la perfección con contenedores web como Tomcat y está siendo integrado en gran cantidad de productos comerciales.

Entre las ventajas de Axis se encuentran su velocidad, su flexibilidad, su estabilidad, su orientación a componentes, su independencia del protocolo de transporte y el soporte de WSDL.

Caso de uso (IV)



9. Integramos axis dentro de JBoss

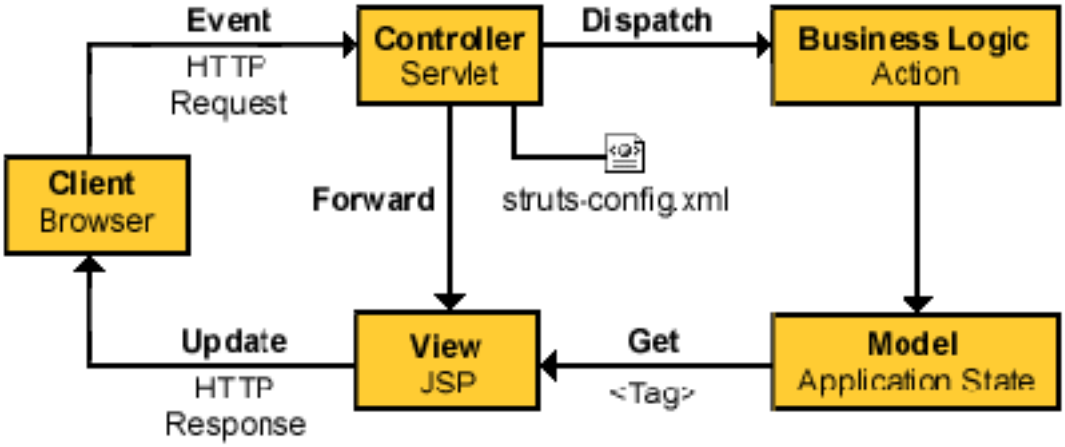
Vamos a aprovechar que Axis se integra con gran cantidad de productos para añadirlo dentro de nuestro servidor JBoss. Si os fijáis en la figura 9, veréis que ahora hay algunos componentes que tienen la flechita de Apache. Los que estéis familiarizados con el mundo de EJBs, rápidamente os habréis dado cuenta que estos componentes son beans de sesión sin estado. Estos beans, están sirviendo algunos de sus métodos al exterior a través de SOAP sobre HTTP, enviando la información en formato XML.

Para nuestros clientes esto puede ser muy importante. Imaginaros el pedido que se hizo desde A Coruña hasta Barcelona y que vimos en las partes anteriores. Ese pedido, pudo ser realizado por un cliente desde su despacho y gracias a que el departamento informático de su empresa integró nuestro servicio web dentro de alguna de las aplicaciones. De este modo, nuestro cliente puede estar viendo en tiempo real nuestro catálogo desde sus aplicaciones y realizar pedidos de manera automática. En resumen, hemos aprovechado el poder de los servicios web para realizar una integración B2B completa, lo que se puede traducir en un mayor volumen de ventas.

Estructuras y sistemas de desarrollo

Hasta ahora nos hemos centrado en los servidores que pueden reemplazar a los productos propietarios habituales en una empresa. Como todos sabemos, esos servidores son sólo una pequeña parte de una arquitectura empresarial. Hace falta escribir las aplicaciones adecuadas a las necesitades de la empresa, y en este punto tenemos gran cantidad de librerías y sistemas que facilitan enormemente la vida de los desarolladores.

Dado que actualmente el mayor ámbito de aplicación de Java es el lado del servidor, especialmente en aplicaciones web, si hay un tipo de sistemas de desarrollo que están teniendo un gran éxito en toda la comunidad son aquellos basados en el patrón de diseño Modelo-Vista-Controlador o MVC, para la web. De entre todos ellos, el más destacado es sin duda Struts[16], proyecto que forma parte de la Apache Software Foundation.



10. Struts implementa un modelo MVC-II

Struts nos ofrece un sistema de desarrollo muy estable que nos permite crear aplicaciones web de forma muy sencilla. Struts, siguiendo la filosofía del patrón MVC, separa completamente la lógica de presentación de la lógica de negocio, consiguiendo de este modo que nuestros proyectos tengan un grado mayor de mantenibilidad, reusabilidad, extensibilidad y flexibilidad.

Struts, puede implementar el modelo lógico utilizando bases de datos relacionales, EJBs, herramientas de mapeo relacional/objetos, etc. Es capaz de sacar la vista a páginas JSP, motores de plantillas como Velocity o a ficheros XML que se formatean mediante plantillas XSL. Por último, una serie de controladores se encargan de manejar todas las interacciones entre la vista y el modelo. Su punto débil es el mismo que el de muchos de los proyectos de software libre y Open Source, la documentación, aunque con un poco de esfuerzo conseguiremos dominarlo rápidamente.

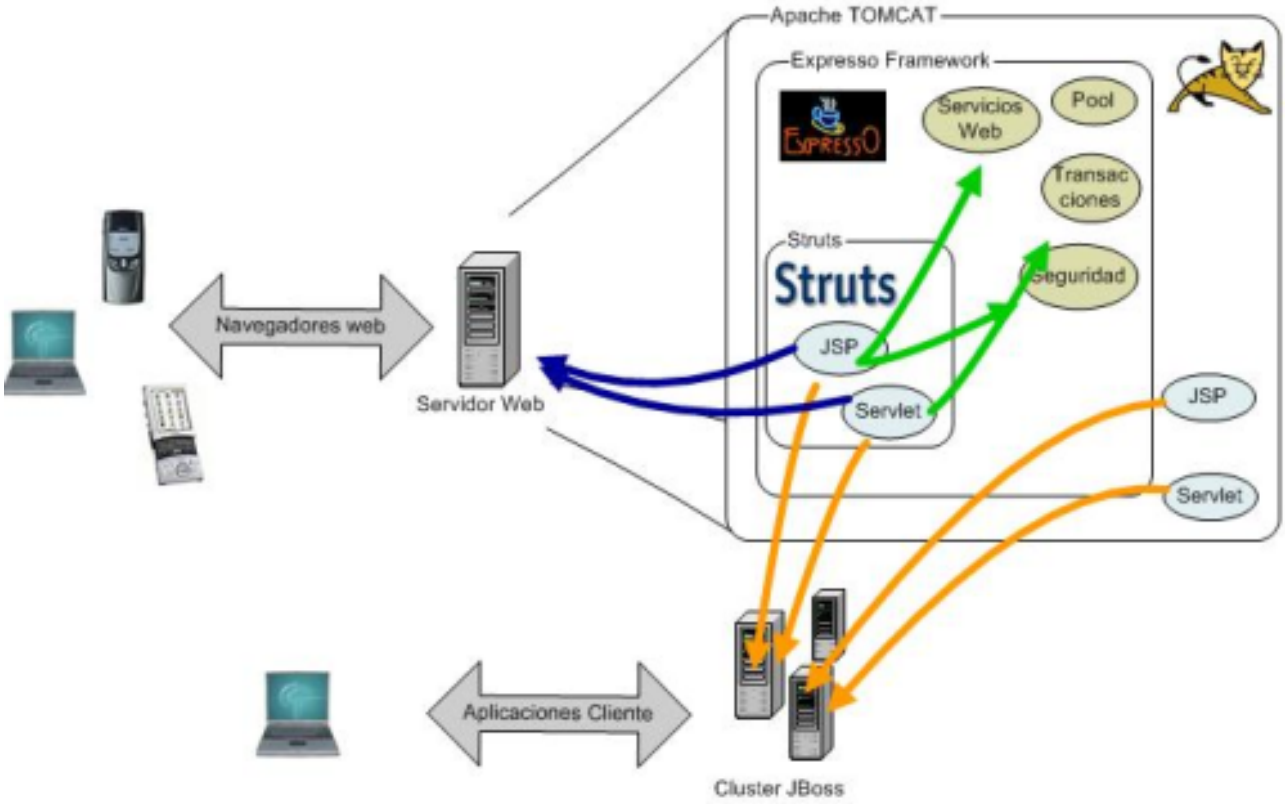
Existen gran cantidad de sistemas de desarrollo como Struts que implementan también el modelo MVC y algunos están siendo muy utilizados: Niggle[17], Webwork[18], el proyecto español WebLEAF[19], Kona[20], Turbine[21], Velocity[22], Tapestry[23], o Canyamo[24], un proyecto que estamos desarrollando en javaHispano y para el que os animamos a participar.

Para terminar con los sistemas de desarrollo web, hemos decidido dedicarle también un espacio especial a otros sistemas mucho más enfocados hacia diseñadores que hacia desarrolladores. Si hay algo en común entre los modelos anteriores, es que además de implementar el modelo MVC, están más o menos enfocados hacia programadores. Nuestros equipo no tiene porque conocer un lenguaje de programación o conocer lo que es un patrón MVC y sin embargo también debería ser productivo.

Esto es lo que intentan solucionar sistemas de desarrollo como Expresso[25]. Expresso es un sistema de mayor nivel, que se sustenta en Struts y que ofrece automáticamente a los desarrolladores web una serie de servicios que no se verán obligados a implementar. Utilizando Expresso, nuestro equipo se podrá olvidar de implementar sistemas de internacionalización, de servicios web, de pools o cachés, de auditoría, etc., simplemente porque ya vienen integrados en el sistema. Además, al igual que otros entornos, Expresso está basado en componentes con lo que los desarrolladores manejan entidades de mayor nivel que añaden una capa de abstracción que permite que diseñadores no acostumbrados con la programación puedan ser más productivos. Por último señalar que Expresso tiene una licencia estilo Apache, con la única limitación de que no permite crear otros sistemas de desarrollo para web a partir de él.

Caso de uso (V)

Como la empresa de transportes no tiene necesariamente porque haber gente que esté acostumbrada al desarrollo de aplicaciones web, o al uso del patrón MVC, y como además nuestro producto no es un sistema de desarrollo para web, hemos decidido incluir en nuestra arquitectura Expresso como sistema de desarrollo.



11. Expresso nos ofrece muchos servicios de manera automática

Como se puede ver en la figura 11, hemos integrado Expresso dentro de Apache Tomcat aunque mantenemos la posibilidad de que se puedan utilizar Servlets y JSP fuera de este sistema de desarrollo. El uso de Expresso, permitirá que las personas del departamento de informática de la empresa de transportes que no estén acostumbradas con el mundo de la programación puedan ser también productivas. Como se ve, internamente Expresso integra Struts por lo que nos aprovechamos implícitamente de las ventajas del modelo MVC, mientras que al arquitectura general no se ve afectada.

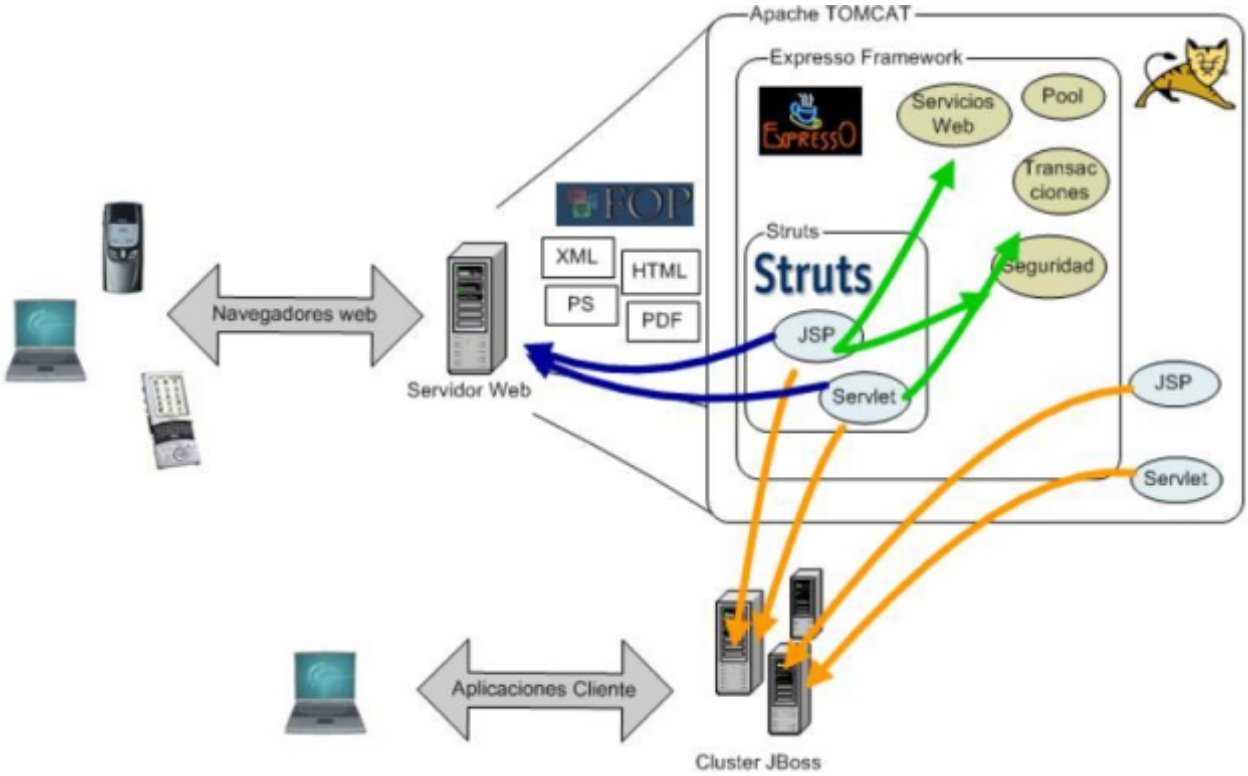
Sistemas de desarrollo con XML

Otro tipo de sistemas de gran importancia dentro de las aplicaciones empresariales en la actualidad son los sistemas que permiten la creación y procesado de ficheros XML. En Java podemos utilizar infinidad de librerías Open Source y de software libre para manejar este tipo de información, sirvan como ejemplo de la variedad existente de proyectos como Xerces[26], Xalan[27], JDOM[28], Cocoon[29], dom4j[30], Crimson[31] o Xindice[32], JXPath[33], etc.

En especial, vamos a hacer especial hincapié en los productos que podremos utilizar

para tratar información y presentarsela al usuario en diferentes formatos. Sistemas como Coocoon nos permiten la generación de portales de información basados completamente en XML, mientras que productos más simples como FOP nos permiten formatear documentos XML utilizando plantillas XML para presentar el mismo documento en diferentes estilos.

Caso de uso (VI)



12. Con FOP transformamos un documento en múltiples formatos

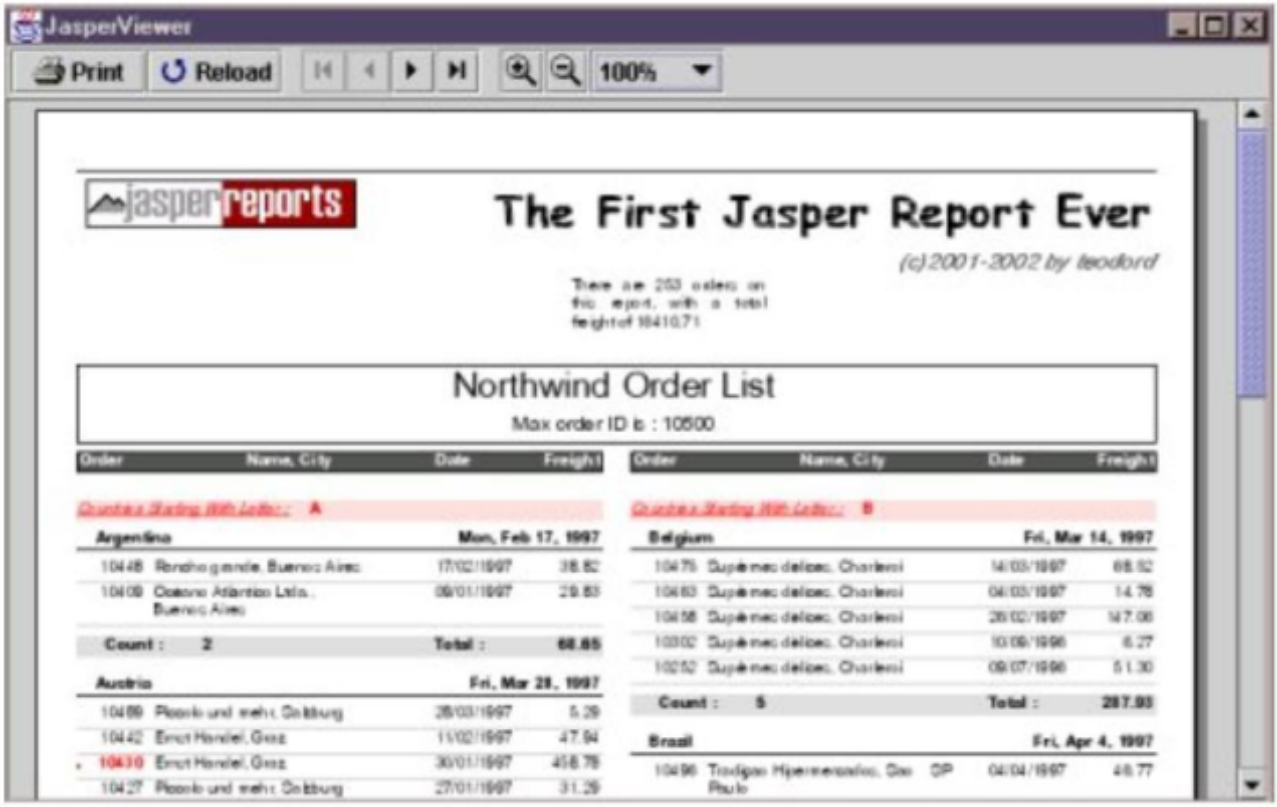
En nuestro caso de uso vamos a integrar FOP[34] como se ve en la figura 12. FOP nos permitirá sacar la salida en diferentes formatos según el dispositivo que se encuentre accediendo a la información. Esto tiene una gran importancia. Volviendo al ejemplo del pedido, imaginémonos que al llegar a Barcelona el responsable de la delegación no está presente. El pedido habrá llegado en un formato XML estándar para nuestra información y será transformado quizás a PDF, quizás a otro formato. Sin embargo, al no estar en la delegación puede que no podamos responder rápidamente y nuestro cliente desestime la compra.

Por suerte nuestra arquitectura es muy potente. Al llegar el mensaje, un sistema podría detectar que el encargado no está presente e inmediatamente enviar una alerta a su teléfono móvil. El encargado, recibiría dicha alerta e inmediatamente accedería a la base de datos de pedidos que no han sido tratados, desde su terminal móvil. En ese momento, el pedido en formato XML sería transformado en un formato de presentación mucho más sencillo, apto para la visualización en un pequeño terminal. A continuación, el encargado podría ver la información y aprobar el pedido. La compra se ha llevado a cabo felizmente.

Generación de informes

La generación de informes es uno de los aspectos más importantes a tener en cuenta a

la hora de realizar una aplicación empresarial. La mayor parte de las aplicaciones de empresa requieren la presentación impresa de gran cantidad de datos para que el equipo de análisis del departamento correspondiente pueda sacar sus conclusiones sobre esos datos.



13. La generación de informes es un factor fundamental en nuestra arquitectura

Cualquier desarrollador habrá tenido que realizar en algún momento de su carrera algún informe y sabrá de la importancia de tener buenas herramientas para su rápida creación, y aún más importante, para su rápida modificación y mantenimiento. Hace años, tener herramientas de este estilo en Java era prácticamente impensable debido a las limitaciones del sistema de impresión que tenía la plataforma. Por suerte, las cosas han cambiado, y con el paso de los años el soporte de impresión ha ido mejorando poco a poco.

En la actualidad existen programas comerciales potentísimos y lamentablemente no existe ninguna alternativa que pueda hacer sombra a productos como Crystal Reports. Sin embargo, lo cierto es que la mayor parte de las características de este tipo de productos no las utilizaremos y casi siempre nos encontraremos con que los productos libres son más que suficientes.

JFreeReport

Con licencia LGPL, JFreeReport[35],permite la previsualización de informes, utiliza directamente los modelos de datos de Swing, define los informes utilizando XML y puede sacar la salida por pantalla, impresora o a fichero PDF.

JasperReports

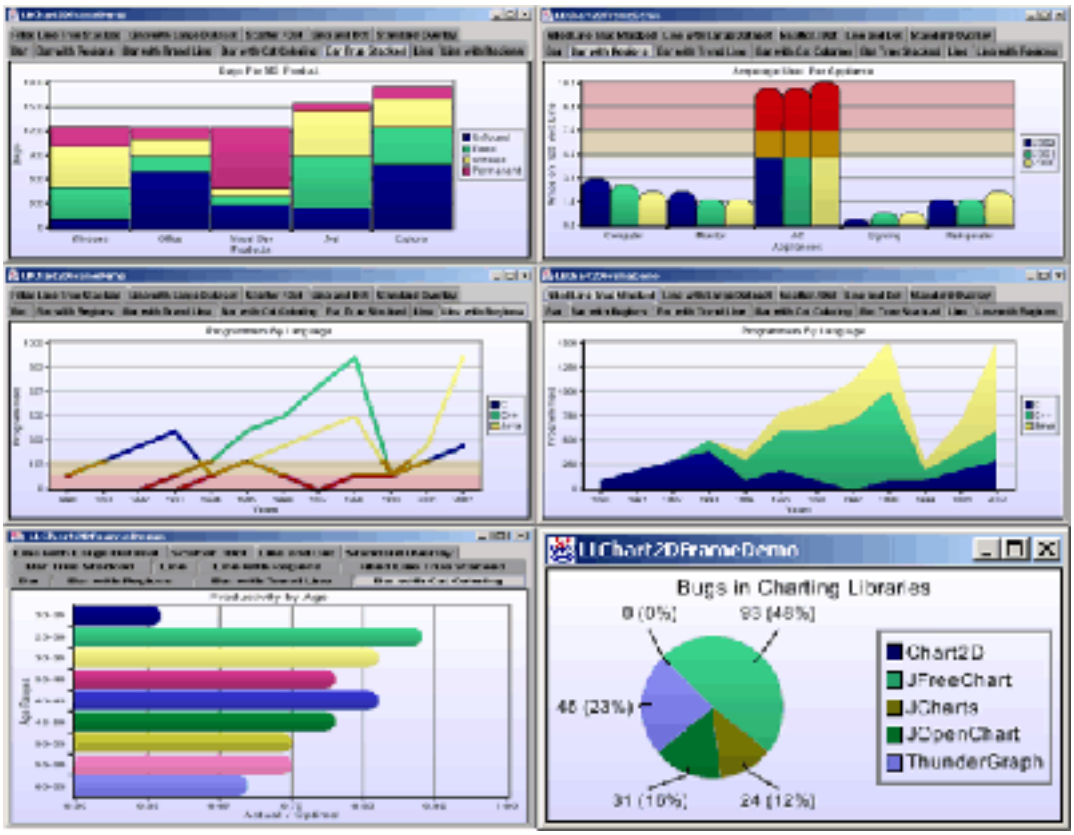
Bajo licencia LGPL, JasperReports[36] tiene una serie de características que lo hacen una alternativa absolutamente válida a productos comerciales: contiene un diseñador gráfico de informes, los informes pueden ser realmente complejos pudiendo incluir varias columnas, imágenes y muchas más cosas, soporta la extracción de datos directamente de bases de datos utilizando JDBC, ofrece posibilidad de previsualización, soporta hipervínculos en los documentos y puede sacar la salida a pantalla, impresora y ficheros XML o PDF.

DataVision

DataVision[37] es un producto que disfruta de muchas de las características avanzadas de JasperReports como soporte de JDBC o la disponibilidad de herramientas gráficas para la generación de informes y que a su vez añade alguna característica especial como la posibilidad de exportar los informes a formato DocBook o LaTeX2e. Su licencia es de estilo Apache.

Sistemas de generación de gráficas

Si generar informes es una tarea fundamental, no lo es menos la generación de gráficas que permitan la explotación de los datos de nuestro sistema. Al igual que en el caso de los informes disponemos de una gran cantidad de productos libres y de calidad que podemos utilizar en nuestros proyectos:



14. Las gráficas proporcionan información adicional muy importante

JFreeChart

Con una licencia LGPL, JFreeChart[38] nos permite generar gráficas de tarta, de barras

2D o 3D, lineales, series de tiempo, diagramas de Gantt, etc., además de dejarnos combinar varios tipos entre sí. Otras características destacadas son: acceso a los datos basado en tablas, se puede utilizar en Servlets, Applets o clientes tradicionales, basado en Java2D para obtener mayor velocidad, posibilidad de hacer zoom a las gráficas, permite imprimir las gráficas y permite también grabar las gráficas en formatos JPG, PNG, SVG o PDF.

Chart2D

También con licencia LGPL, Chart2D[39], soporta los tipos de gráficas tradicionales: tarta, barras, lineal, etc., o sus combinaciones. Las gráficas se pueden tratar como imágenes o como componentes gráficos y tiene características curiosas como la posibilidad de aplicar efectos de luz, redondear los bordes de las barras, colorear por categorías o establecer áreas peligrosas que salen en diferentes colores.

Caso de uso (VII)

15. Generación dinámica de informes y gráficas

En la figura 13 podemos ver el resultado de añadir estos sistemas de generación a nuestra arquitectura. Ahora ya estamos manejando un nivel de abstracción mayor en el que nuestros sistemas, ya sea a través de aplicaciones o a través de navegadores web, ofrecen una serie de información que será de vital importancia para nuestra empresa.

Organismos como gerencia, departamentos de análisis estadístico, dirección de gestión, etc. podrán obtener informes y gráficas a través de medios muy diferentes, en diferentes formatos y que serán fácilmente modificables gracias a la flexibilidad de todas las herramientas que hemos visto.

Arquitectura del software

Por ahora, nos hemos centrado en la parte estructural de nuestra arquitectura pero aún

no hemos pensado en que herramientas vamos a utilizar tanto a la hora de modelar la arquitectura software de nuestro sistema como a la hora de comenzar a picar código.

En Java, como lenguaje orientado a objetos que es, son aplicables las técnicas más modernas de ingenería del software que han estado surgiendo durante los últimos años. Patrones de diseño, antipatrones, programación extrema o refactorizaciones son algunos conceptos cada vez más importantes y fundamentales para el desarrollo de aplicaciones empresariales.

El número de documentación, herramientas y sistemas de desarrollo existentes, que nos permiten tomar ventaja de todos estos instrumentos fundamentales para cualquier arquitecto es realmente grande.

En cuanto al modelado de nuestra arquitectura, lamentablemente, no disponemos de alternativas libres a productos como Together o Rational Rose, y que tengan la misma calidad. Sin embargo, existen una serie de herramientas libres como ArgoUML[40] con licencia BSD, Eclipse UML[41] con licencia LGPL o EMF (Eclipse Modelling Framework)[42] también con licencia LGPL, que nos permiten de una manera bastante básica el modelado de nuestro sistema utilizando UML.

Una vez que tengamos el diagrama de clases (no suelen permitir otros tipos de diagramas), el siguiente paso es utilizar otras herramientas como UML2EJB[43], RadTool[44] o MiddleGen[45], que son capaces de importar modelos UML en formato XMI y a partir de ahí generar automáticamente los EJBs, ficheros de Struts, ficheros de persistencia, etc., que forman nuestro sistema. Estas herramientas unidas a las de modelado nos ofrecen soluciones case muy avanzadas que nos permitirán aumentar considerablemente aumentar la productividad de nuestro equipo de desarrolladores.

Dentro de estas herramientas case, hay dos productos que requieren mención especial: XDoclet[46] y Jenerator[47]. XDoclet, utiliza unas etiquetas especiales en formato javadoc que se colocan en las cabeceras los métodos de nuestros ficheros y a partir de las cuales se genera gran cantidad de información como interfaces locales y remotas, beans, patrones como los value objects, delegados de negocios o fachadas de sesión, servicios web, ficheros para struts, ficheros de persistencia con JDO, etc. Lo más importante de XDoclet es que dispone de etiquetas para múltiples servidores de aplicaciones con lo que es posible generar a partir del mismo código aplicaciones para diferentes servidores, tanto libres como comerciales, con lo que la portabilidad de nuestras aplicaciones crece enormememente.

Jenerator, por su parte, tiene una filosofía ligeramente diferente. La definición de los ficheros se realiza a través de descriptores XML y a partir de dichos descriptores se generan todas las interfaces, los patrones, etc. A diferencia de XDoclet, Jenerator sólo genera aplicaciones estándar, que por supuesto, funcionarán en cualquier servidor compatible con la plataforma J2EE.

Ya olvidándonos del tema de la generación de código, no podemos tampoco podíamos dejar de mencionar productos como jUnit[48] o Cactus[49] que nos permiten realizar todos los test unitarios de nuestras aplicaciones. También son de obligada mención entornos de desarrollo libres, como Eclipse o netBeans (aunque hablaremos después de ellos), que cada vez más, comienzan a incluir utilidades de refactorización de modo que ya no es necesario obtener un programa comercial para realizar todas estas tareas cotidianas. Programas de análisis de la estructura y la métrica de los proyectos como SmallWorlds[50], aunque no completamente libre (si con versión para la comunidad Open Source) también son extremadamente útiles.

Y sólo podíamos terminar este recorrido con otra de las maravillas que nos ha dado la Apache Software Foundation: Ant[51]. Se trata de un sistema de construcción de aplicaciones desarrollado en Java pero que sirve para multitud de lenguajes. Ant elimina todas las limitaciones de herramientas como make, gnumake, etc., a la vez que elimina los engorrosos ficheros makefile. Ant, sin embargo, basa su potencia en la flexibilidad de XML y en su funcionamiento multiplataforma. Tal ha sido el éxito de esta herramienta que casi todos los entornos de desarrollo comerciales, como JBuilder, ya la utilizan.

Entornos integrados de desarrollo

Tan importante como disponer de un buen sistema empresarial es el disponer de unas buenas herramientas de desarrollo. Los entornos integrados son herramientas de díficil análisis. Existen tantas opiniones sobre estas herramientas como desarrolladores hay en el mundo y por lo tanto es imposible encontrar un super-entorno. Afortunadamente, en Java disponemos de varios entornos de desarrollo libres que se adaptan a las diferentes características de los desarrolladores. Desde entornos ligeros y que ofrecen justo lo necesario hasta entornos que ofrecen un gran número de opciones y funcionalidades que pueden satisfacer a los usuarios más exigentes.

netBeans

El entorno de desarrollo netBeans[52] surgió como proyecto en el año 1996, por lo que como es de suponer, es un proyecto estable y maduro. Su licencia es la Sun Public License, compatible con la definición Open Source, y que permite el uso del entorno para proyectos libres y comerciales.

Describir sólo las características nos ocuparía prácticamente lo mismo que este artículo. Destacar: posibilidad de edición de múltiples lenguajes, creación de aplicaciones web, creación de Enterprise Beans, soporte de CVS, gran cantidad de asistentes, coloreado del código, autocompletado del código, navegación de clases, etc. A todas estas características hay que añadir la gran cantidad de plugins disponibles en su comunidad.

Eclipse

Como proyecto Open Source, Eclipse[53], fue creado por IBM, y está liberado bajo la Common Public License del mismo fabricante. Con Eclipse vienen también herramientas de terceras partes que se distribuyen con la licencia propia de esos fabricantes.

En general, las características del entorno de desarrollo Eclipse son las mismas que las que vimos en netBeans. Cabe destacar la inclusión de SWT, un conjunto de liberías de la plataforma para el desarrollo de interfaces gráficos que siguen un modelo muy similar al de Swing pero con rendimiento similar al de las aplicaciones nativas. Al igual que en el caso de netBeans, a todas sus características hay que añadir los plugins disponibles dentro de su comunidad.

Aunque la elección de un entorno de desarrollo, como ya hemos dicho, es muy subjetiva. Lo cierto es que Eclipse se está imponiendo en la comunidad de software libre. Gran parte de la culpa la tiene su arquitectura modular que ofrece gran cantidad de plugins (más de 180), y el extraordinario soporte que ofrece para la refactorización de código. Por si fuera poco a mediados de diciembre del 2002 se han incorporado al proyecto Eclipse 15 nuevas importantes empresas entre las que se encuentran Oracle,

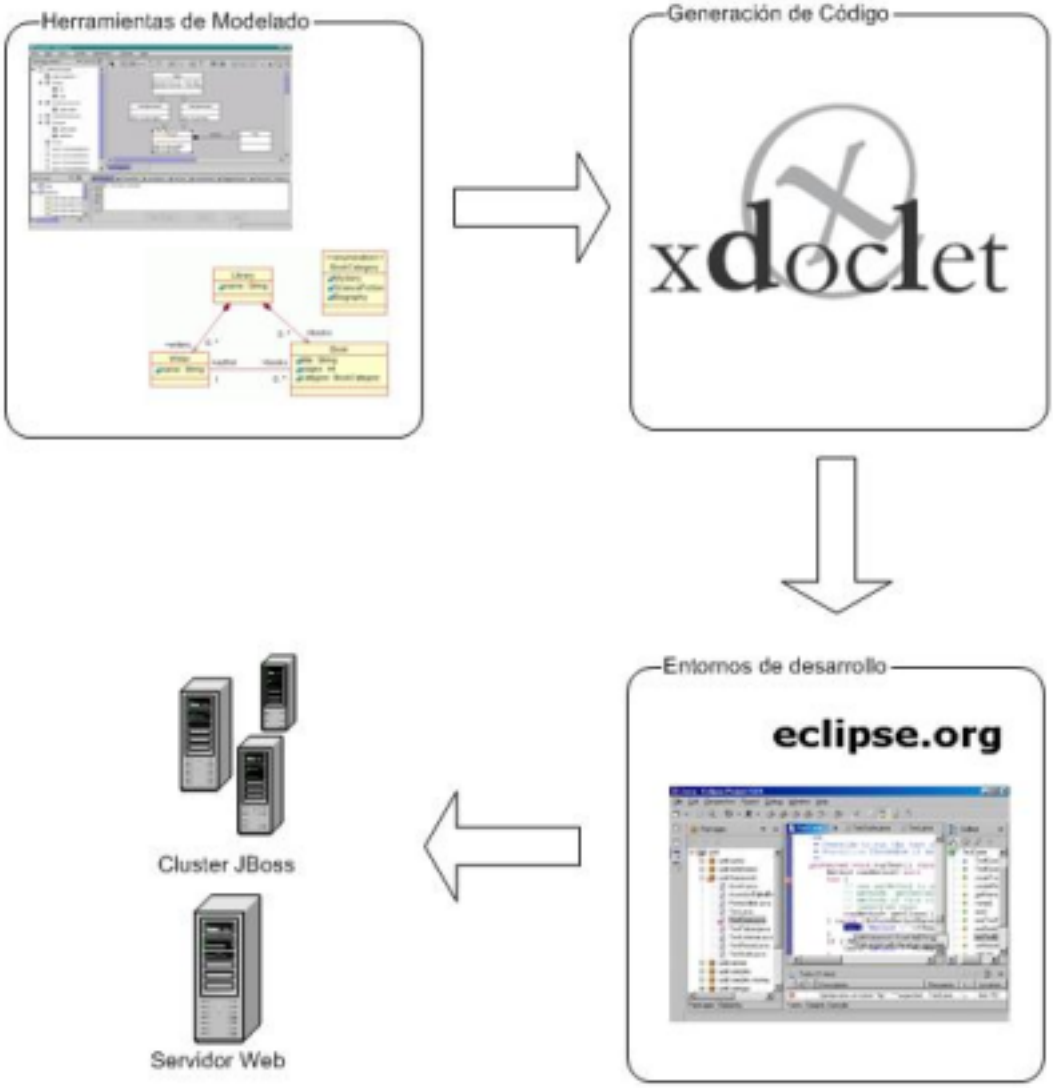
IONA o HP, lo que da una idea del apoyo que está recibiendo este proyecto.

jEdit

jEdit[54], a diferencia de los dos ejemplos anteriores, es un entorno de desarrollo muy ligero. Está liberado bajo una licencia GPL y su ventaja frente a los entornos anteriores es que necesita mucha menos memoria y procesador por lo que puede funcionar en máquinas con menos requisitos.

Entre sus características básicas están el soporte de coloreado para más de 70 lenguajes, lenguaje de macros, autotabulación del código, gran capacidad de configuración y la más importante, su arquitectura modular con más de 50 plugins disponibles y que van desde el manejo de FTP a reescritura de los ficheros según unas normas de estilo definibles, pasando por cosas como gestión de proyectos o tareas.

Caso de uso (VIII)



16. Un posible ciclo de desarrollo

En la figura 16 se puede ver el modelo de desarrollo que proponemos, uno de tantos que serían posibles con la cantidad de herramientas de las que disponemos. Primero

utilizaríamos alguna herramienta de modelado para la creación de nuestro modelo lógico, modelo que a continuación pasaríamos a código utilizando herramientas como XDoclet. Este esqueleto de código lo podríamos editar utilizando Eclipse, o incluso mejor, podríamos realizar ambas tareas desde Eclipse gracias a plugins como Lomboz[55]. Para finalizar, desplegaríamos nuestras aplicaciones a nuestros servidores.

Sistemas de información empresarial

Ya hemos visto que los servidores de aplicaciones nos ofrecen automáticamente conectores para acceder a sistemas legacy basados en COBOL, Tuxedo, etc. Sin embargo, queda por definir un punto importante dentro de nuestra arquitectura, la base de datos.

Aquí tenemos muchos productos libres para elegir, como MySQL[56], Postgres[57] o Hypersonic SQL[58]. Sin embargo nos hemos decidido por utilizar una base de datos como SAP DB[59]. Esta base de datos, con licencia LGPL, y que no tiene nada que envidiar a los productos comerciales, viene abalada por el gigante alemán por lo que nos puede proporcionar una base sólida a la hora de crear la estrategia de marketing de nuestra solución. Como cabría de esperar, esta base de datos se integra a la perfección con sistemas SAP, factor de gran importancia para las empresas.

Y podríamos haber utilizado más...

... porque nos han quedado gran cantidad de productos en el tintero: OpenSymphony[60], MX4J[61], Open Business[62], ION-CMS[63], Avalon[64], Maven[65], Otemba[66], Hibernate[67], Luxor XUL[68], Swing ML[69], OpenJMX[70], FreeMarker[71], OJB[72], WebMacro[73], OFBiz[74], GLUE[75], BeanShell[76], Castor[77], ORO[78], Log4J[79], Tyrex[80], JIRA[81], Barracuda[82], JPublish[83], Joist[84], OpenCMS[85], Xoplon[86], Prowler[87],etc., son sólo algunos de los nombres de sistemas de desarrollo, gestores de contenidos, sistemas de workflow, etc. que podríamos haber utilizado en nuestra arquitectura y que con las que con seguridad obtendríamos nuevas funcionalidades que podríamos ofrecer en nuestra solución.

Caso de uso (y IX)

17. Implantaremos Linux en todas las delegaciones

Un momento, todavía falta la guinda del pastel, no podríamos terminar nuestra arquitectura sin colocar un sistema operativo libre en nuestros clientes. En este caso ha sido Linux, podría haber sido FreeBSD o cualquier otro sistema que disponga de máquina virtual Java. Aunque fijándonos únicamente en la figura 17 podríamos pensar que nuestra arquitectura es muy simple, hemos visto que no. Hemos visto que hemos creado una arquitectura completísima utilizando única y exclusivamente productos basados en software libre.

Conclusión

» **J2EE es una plataforma empresarial ideal para el desarrollo de software libre**. La cantidad y riqueza de las soluciones libres que podemos utilizar para crear arquitecturas empresariales dentro de la plataforma J2EE ha quedado de manifiesto en este artículo. No sólo son muchos productos sino que además son productos de calidad contrastada y que están siendo adoptados por numerosas soluciones comerciales.

» **Hemos sido capaces de crear una infraestructura empresarial muy completa únicamente con productos libres**. Además, esta arquitectura tiene todos los requisitos que se le pedían a cualquier plataforma empresarial, escalabilidad, fiabilidad, disponibilidad, extensibilidad, etc., y lo hemos conseguido utilizando productos libres.

» **Se está abriendo un mercado muy amplio y hay grandes oportunidades**. La solución que hemos visto colmaría las espectativas de cualquier empresa u organización. Nuestra solución no tiene grandes costes de licencias por lo que el márgen de beneficios que podemos ofrecer es muy grande. Queda claramente demostrado que las arquitecturas basadas en software libre son las que ofrecen la mejor relación calidad/precio. Desde luego, para una pequeña consultora es el momento más adecuado para entrar en un nicho de mercado que actualmente no está siendo demasiado explotado.

» En resumén : **J2EE + Software Libre = Garantía de éxito**

Recursos y referencias

[1] Java Community Process, *http://www.jcp.org*
[2] JSR 151 (J2EE 1.4), *http://www.jcp.org/jsr/detail/151.jsp*
[3] Página web de J2EE, *http://www.sun.com/j2ee/*
[4] Test de compatibilidad de J2EE, *http://java.sun.com/j2ee/compatibility*
[5] Implementación de referencia de J2EE, *http://java.sun.com/j2ee/download.html#sdk*
[6] J2EE BluePrints, *http://java.sun.com/blueprints/enterprise/index.html*
[7] JBoss, *http://www.jboss.org*
[8] JOnAS, *http://www.objectweb.org/jonas/*
[9] OpenEJB, *http://openejb.sourceforge.net*
[10] JORAM, *http://www.objectweb.org/joram/*
[11] OpenJMS, *http://openjms.sourceforge.net*
[12] Apache Tomcat, *http://jakarta.apache.org/tomcat/*
[13] Jetty, *http://jetty.mortbay.org/jetty/*
[14] Enhydra, *http://www.enhydra.org/*
[15] Apache Axis, *http://jakarta.apache.org/axis/*
[16] Struts, *http://jakarta.apache.org/struts/*
[17] Niggle, *http://www.niggle.org*
[18] WebWork, *http://www.sourceforge.net/projects/webwork/*
[19] WebLEAF, *http://www.sourceforge.net/projects/webleaf/*
[20] Kona, *http://www.aki.com/kona/*
[21] Turbine, *http://jakarta.apache.org/turbine/*
[22] Velocity, *http://jakarta.apache.org/velocity/*
[23] Tapestry, *http://tapestry.sourceforge.net/*
[24] Canyamo, *http://canyamo.sourceforge.net/*
[25] Expresso, *http://jcorporate.com*
[26] Xerces, *http://xml.apache.org/xerces-j/*
[27] Xalan, *http://xml.apache.org/xalan-j/*
[28] JDOM, *http://www.jdom.org*
[29] Cocoon, *http://xml.apache.org/cocoon/*
[30] dom4j, *http://www.dom4j.org*
[31] Crimson, *http://xml.apache.org/crimson*
[32] XIndice, *http://www.xindice.org*
[33] JXPath, *http://jakarta.apache.org/commons/jxpath/*
[34] FOP, *http://xml.apache.org/fop/*
[35] JFreeReport, *http://www.object-refinery.com/jfreereport/*
[36] JasperReports, *http://jasperreports.sourceforge.net/*
[37] DataVision, *http://datavision.sourceforge.net/*
[38] JFreeChart, *http://www.object-refinery.com/jfreechart/*
[39] Chart2D, *http://chart2d.sourceforge.net*
[40] ArgoUML, *http://argouml.tigris.org*
[41] Eclipse UML, *http://www.eclipseuml.com*
[42] Eclipse Modelling Framework, *http://www.eclipse.org/emf/*
[43] UML2EJB, *http://uml2ejb.sourceforge.net/*
[44] RadTool, *http://radtool.sourceforge.net*

[45] MiddleGen, *http://boss.bekk.no/boss/middlegen/*
[46] XDoclet, *http://xdoclet.sourceforge.net*
[47] Jenerator, *http://www.visioncodified.com/over.html*
[48] jUnit, *http://www.junit.org*
[49] Cactus, *http://jakarta.apache.org/cactus/*
[50] SmallWorlds, *http://www.thesmallworlds.com/*
[51] Ant, *http://jakarta.apache.org/ant/*
[52] netBeans, *http://www.netbeans.org*
[53] Eclipse, *http://www.eclipse.org*
[54] jEdit, *http://www.jedit.org*
[55] Lomboz (plugin para Eclipse), *http://www.mycgiserver.com/~objectlearn/products/lomboz.html*
[56] MySQL, *http://www.mysql.com*
[57] Postgres, *http://www.postgresql.org*
[58] Hypersonic SQL, *http://hsqldb.sourceforge.net*
[59] SapDB, *http://www.sapdb.org*
[60] OpenSymphony, *http://www.opensymphony.com/*
[61] MX4J, *http://mx4j.sourceforge.net/*
[62] Open Business Engine, *http://www.openbusinessengine.org/index.html*
[63] ION-CMS, *http://ion-cms.sourceforge.net/*
[64] Avalon, *http://jakarta.apache.org/avalon/*
[65] Maven, *http://jakarta.apache.org/turbine/maven/*
[66] Otemba, *http://sourceforge.net/projects/otemba*
[67] Hibernate, *http://hibernate.sourceforge.net/*
[68] Luxor-XUL, *http://luxor-xul.sourceforge.net/*
[69] SwingML, *http://swingml.sourceforge.net/*
[70] OpenJMX, *http://openjmx.sourceforge.net/*
[71] FreeMarker, *http://freemarker.sourceforge.net*
[72] OJB, *http://jakarta.apache.org/ojb/*
[73] WebMacro, *http://www.webmacro.org*
[74] OFBiz, *http://www.ofbiz.org*
[75] GLUE, *http://www.gnu.org/software/glue/glue.html*
[76] BeanShell, *http://www.beanshell.org/*
[77] Castor, *http://castor.exolab.org/*
[78] ORO, *http://jakarta.apache.org/oro/*
[79] log4j, *http://jakarta.apache.org/log4j/docs/*
[80] Tyrex, *http://tyrex.exolab.org*
[81] JIRA, *http://www.atlassian.com/software/jira/*
[82] Barracuda, *http://barracuda.enhydra.org/*
[83] JPublish, *http://www.jpublish.org*
[84] Joist, *http://joist.tigris.org*
[85] OpenCMS, *http://www.opencms.com*
[86] Xoplon, *http://www.tonbeller.com/xoplon/*
[87] Prowler, *http://www.infozone-group.org/prowlerDocs/html/proposal.html*
[88] Listado de herramientas Open Source en javaHispano, *http://www.javahispano.org/opensource/opensource.jsp*
[89] Big Step Forward in Apache/SUN Agreement, *http://www.servlets.com/blog/archives/000028.html*
[90] Nueva licencia de Java, *http://www.javahispano.com/articulos/ver_articulo.jsp?id=64*

Acerca de los autores

Alberto es ahora mismo desarrollador de aplicaciones en ámbito cliente/servidor para la empresa T-Systems International GmbH en Dresden (Alemania). Cuando no está trabajando o "metiendo caña" al resto de los integrantes de javaHispano intenta pasear con su novia, buscar la desaparecida lógica del idioma alemán o intenta disfrutar del buen ambiente de la Neustadt de Dresden.

Martín Pérez Mariñán trabaja en España para INTECNO, una empresa del Grupo DINSA, desarrollando aplicaciones empresariales para el Hospital Juan Canalejo de A Coruña. Martín es Ingeniero de Sistemas por la Universidad de La Coruña además de SUN Certified Java Programmer y SUN Certified Java Developer.