DISEÑO E IMPLEMENTACION DE UN PROTOTIPO DE SOFTWARE PARA LA CONSULTA DE UNA BASE DE DATOS DESDE UN DISPOSITIVO MÓVIL, MEDIANTE WAP GENERADO DINÁMICAMENTE CON JAVA EN EL SERVIDOR

DISEÑO E IMPLEMENTACION DE UN PROTOTIPO DE SOFTWARE PARA LA CONSULTA DE UNA BASE DE DATOS DESDE UN DISPOSITIVO MÓVIL, MEDIANTE WAP GENERADO DINÁMICAMENTE CON JAVA EN EL SERVIDOR

Johanna Patricia Aranguren Rodríguez
Luis Roberto Ballestas Vergara

AGRADECIMIENTOS

Queremos darles las gracias primero a toda la tecnología, por estar en constante avance para nuestro propio beneficio y sabiduría, además a todas las personas que son profesores, quienes nos guiaron y nos mostraron las puertas hacia el futuro tecnológico, para que podamos emprender nuestro camino dentro de este mundo.

También agradecemos a nuestra familia por todo el apoyo brindado y la confianza depositada hasta el día de hoy; y sobre todas las cosas damos gracias por estar juntos en este proyecto y en esta etapa de la vida.

TABLA DE CONTENIDO

INTRODUCCION

La temática dentro de la cual se enmarca esta monografía es el desarrollo de interfaces en WAP para lograr la interconexión entre un dispositivo móvil y una base de datos que contiene información comercial de productos u otros relacionados a través de Internet.

La cobertura de este tipo de investigación es a nivel internacional ya que cada día con la evolución de los proveedores de servicios y creadores de celulares están reduciendo sus precios de venta haciendo que la conexión de Internet y otros servicios agregados sean asequibles para cualquier tipo de personas con un ingreso medio e inclusive bajo.

El campo de investigación para poder desarrollar esta idea es muy amplio, va desde una simple persona que desea conocer información de alguna promoción o servicio en particular hasta infraestructuras seguras en empresas internacionales para que sus delegaciones y demás personas interesadas fuera de la sede central tengan acceso a la información necesaria con tal de realizar ventas rápidas y eficaces.

Desde el punto de vista de usuarios cotidianos esta idea sale a partir de que no existen muchos sitios en Internet diseñados exclusivamente para dispositivos móviles y la navegación en sitios diseñados para PC en móviles se hace un imposible.

Desde el punto de vista del comercio y a nivel empresarial seria una gran opción además de rápida e instantánea que las personas encargadas de comercializar algún tipo de productos posean inmediatamente y sin necesidad de un computador, solo con su celular tener acceso a listas de información de producción tales como inventarios, programaciones de producción, listado de productos disponibles, entre otros indicadores que le ayuden a tomar mejores decisiones a nivel de ventas.

JUSTIFICACION

Este tema fue elegido para apoyar el desarrollo tecnológico que están experimentando los dispositivos móviles en nuestro país, ya que poseemos la documentación y estamos en la capacidad de desarrollar nuevas tecnologías que contribuyan a este avance.

Esta investigación es de dos tipos. El primero es de tipo descriptiva ya que se explicaran punto a punto la tecnología que se utiliza para que un celular navegue en Internet y como un lenguaje de programación hace que una pagina se acomode y atienda a los eventos en la pantalla de un dispositivo móvil. El segundo es de tipo experimental ya que se montará una aplicación en un servidor que esta en Internet demostrando el alcance de nuestra idea por medio de un dispositivo móvil.

Dentro de los recursos con los que se cuentan para llevar a cabo este trabajo se encuentran:

- Celulares para pruebas.
- Servidor en Internet.
- Servidor Web Java: de tipo Open Source.
- Base de datos licenciada.
- Documentación en Internet para WAP.
- Entre otros.

Que hacen que este trabajo alcance la totalidad de los objetivos planteados.

RESUMEN

WAP, como sus siglas lo indican, es un protocolo para las aplicaciones wireless, cuyo objetivo es acceder a contenidos de Internet desde dispositivos móviles.

También define un protocolo para el diseño de aplicaciones Web que permitan acceder a contenidos y servicios desde estos mismos dispositivos móviles.

Está diseñado para “micro-browsers” con las limitaciones:

- ✓ Presentación de información en la pantalla.
- ✓ Limites de memoria y capacidad de computación de los dispositivos “clientes”.
- ✓ Ancho de banda.

Las características principales que cumple la arquitectura Wap es la siguiente:

- ✓ Las capas son escalables.
- ✓ Las capas posean una integración lo máximo posible.
- ✓ Compatibilidad con todo tipo de redes inalámbricas.

El proceso de comunicación entre el dispositivo WAP y el servidor Web que ofrece los contenidos, se puede resumir de la siguiente manera:

- ✓ El dispositivo móvil establece una conexión con la estación base, e inicia la conexión a una pasarela WAP preestablecida en la configuración del teléfono.

- ✓ El micro-browser solicita una URL a la pasarela WAP, mediante una petición codificada y comprimida en un formato binario WTP.
- ✓ La pasarela WAP recibe la petición, y la traduce a una petición HTTP, que se envía a través de Internet al servidor adecuado.
- ✓ El servidor Web recibe la petición y como resultado envía una página con datos WML a la pasarela WAP mediante HTTP.
- ✓ La pasarela WAP recibe la página WML, la comprime a un formato binario optimizado, que la reenvía al dispositivo móvil mediante el protocolo WTP.
- ✓ El micro-browser decodifica la señal recibida, y muestra la página.

El lenguaje WML es un lenguaje de marcas basado en XML, posee una ventaja a poder diseñarse rápidamente, pero no todos los navegadores Wap soportan tablas, imágenes o tipos de texto; una página WML es muy similar a un documento HTML. WAP Forum lo diseño y también se encarga de su mantenimiento.

En lugar de tener un contenido largo y extenso, un archivo de WML representa una baraja (deck) de cartas (cards). Solamente una carta es mostrada a la vez, pero se puede pasar de una carta a la otra y guardarlas en una baraja en el mismo archivo. Las cartas en al baraja son descargadas al mismo tiempo, así que el usuario debe esperar una sola vez, y accede a las cartas casi instantáneamente.

WMLScript es el lenguaje de script que se ejecuta en el dispositivo del cliente. Es una versión simplificada de JavaScript la cual sirve para validar datos introducidos por el usuario, cálculos, etc.

El código es “compilado” por el WAP Gateway y enviado al cliente. El código WMLScript no se guarda en los archivos wml, sino en archivos wmls.

WMLScript es un lenguaje con poca estructura, posee cinco tipos de datos: cadena (string), entero (integer), punto flotante (flota-point), booleano (bolean) y el tipo especial invalido (invalid). Proveen estructuras de control como loops (while, for, break, return) y condicionales (if, else), también como la habilidad de devolver un valor de una función.

Los Servlets son la respuesta de la tecnología Java a la programación CGI. Son programas que se ejecutan en un servidor Web y construyen páginas Web. Su trabajo es: tener listo todos los datos enviados por el usuario, generar los resultados, enviar el documento de vuelta al cliente.

Los servlets surgen debido a la limitación existente que existían en los applets desarrollados para el Web en dos vertientes distintas: la imposibilidad de acceder a otro servidor que no sea el mismo en el que el applet se este ejecutando y la limitación en el acceso a los servicios de esa máquina.

Java Server Pages (JSP) es una tecnología que permite mezclar HTML estático con HTML generado dinámicamente. El JSP permite separar la parte dinámica de las páginas Web del HTML estático. Simplemente se escribe el HTML regular de la forma normal, usando cualquier herramienta de construcción de páginas Web que se utilice normalmente. Se encierra el código de las partes dinámicas en unas etiquetas especiales, la mayoría de las cuales empiezan con "<%" y terminan con "%>. Normalmente se dará a los archivos una extensión .jsp, y se instalaran en el mismo sitio que una página Web normal.

Tomcat, algunas veces conocido como Apache Tomcat, otras veces como Jakarta Tomcat. Tomcat es un contenedor de Servlets con un entorno JSP. Un contenedor de Servlets es un shell de ejecución que maneja e invoca

servlets por cuenta del usuario. Se puede dividir los contenedores de Servlets en: Contenedores de Servlets independientes (Stand-alone), Contenedores de Servlets dentro-de-Proceso, Contenedores de Servlets fuera-de-proceso.

Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor Web existente (actualmente se soportan los servidores Apache, IIS y Netscape). Esto significa que siempre que se despliegue Tomcat se tendrá que decidir cómo usarlo.

Los fundamentos de seguridad fueron elaborados con base al modelo OSI pero un poco más liviano, en donde se tiene en cuenta que su desarrollo para el envío y recepción de la información.

OBJETIVOS GENERALES

Diseñar e implementar un prototipo de software para la consulta de una base de datos desde un dispositivo móvil, mediante WAP generado dinámicamente con java en el servidor.

OBJETIVOS ESPECIFICOS

El capítulo de WAP tiene como objetivo estudiar los fundamentos de WAP. Cuales fueron las necesidades del mundo moderno las que llevo a que revolucionara la forma de comunicación a través de Internet.

El capítulo de WML y WMLScript hace una breve introducción al lector al desarrollo de la programación con ayuda de estas herramientas que posee WML y WMLScript.

El objetivo del capítulo sobre los fundamentos de los Servlets tiene como fin describir su estructura básica, como también estos Servlets son muy útiles cuando se va a crear una aplicación utilizando WML y que papel juegan dentro de la integración con el servidor y con JSP.

El capítulo de los fundamentos de JSP tiene como objetivo principal dar a conocer los elementos de escritura necesarios para poder crear expresiones JSP dentro del lenguaje WML.

El capítulo de introducción al servidor de aplicaciones Tomcat de Apache explica el funcionamiento del mismo, como se debe instalar y configurar este servidor Web, también explica como ha ido evolucionando desde su creación.

El objetivo del capítulo de la integración de WML, Servlets y Apache Tomcat es poder explicar la unión del servidor Web con los Servlets y el lenguaje WML.

El último capítulo se refiere a los fundamentos de la seguridad en redes y el principal objetivo es poder explicar que tipo de seguridad requiere el servidor y las redes por donde va a viajar la información para evitar posibles intrusos no deseados.

Al realizar el laboratorio, los objetivos de la práctica son:

- Analizar y diseñar cada una de las capas del prototipo de software con UML.
- Investigar las formas en que se establece la conexión de un dispositivo móvil a Internet.
- Describir algunos de los servidores Web que soporte tecnología Java (JSP).
- Describir el diseño, conexión y estructura de la base de datos.
- Describir el sistema de seguridad (Firewall) de este prototipo.
- Implementar el prototipo de software para demostrar la teoría descrita.

WIRELESS APLICATION PROTOCOL (WAP)

SERVICIOS DE DATOS MOVILES

Las afirmaciones más extravagantes sobre tecnología sin cable se han realizado en torno a los datos móviles. Si se hubiera creído a las compañías vendedoras, pronto se podría ver televisión de alta definición en directo en una pantalla del tamaño de un bolsillo, así como navegar por la red y participar en una videoconferencia.

Este Internet móvil se encuentra todavía muy lejos del ciberespacio que, quienes navegan, conocen y adoran. [Las velocidades de datos son todavía penosamente lentas y para reducir los rasgos de las páginas más que para incrementar las velocidades de datos. En medio del entusiasmo, merece la pena recordar que la mayor parte del tráfico de datos móviles todavía se centra en los servicios de mensajería (transmisiones de texto, las cuales parecen horriblemente primitivas para cualquiera que haya utilizado el correo electrónico)].[1]

MENSAJERIA

Todos los sistemas de telefonía móvil digital ya incorporan algún tipo de mensajería. Esto permite a los suscriptores recibir y a veces enviar mensajes

[1] WAP guía práctica para usuarios, Autor Andy Dornan, Ediciones ANAYA MULTIMEDIA (GRUPO ANAYA S.A.), 2001, página 141.

de texto corto (en esencia, lo mismo que el paging, pero con los datos apareciendo en un teléfono móvil en lugar de un buscapersonas independiente).

En teoría, los servicios de mensajes deberían permitir que la gente recibiera correo electrónico a través de su teléfono móvil y que, del mismo modo, gestionara el paging. Pero en la práctica, ninguno de estos objetos se ha alcanzado. Las razones son en parte técnicas, debido a los límites de longitud impuestos por la muy baja capacidad de datos, pero principalmente comerciales.

La mayoría de operadores cobran por SMS utilizando el mismo modelo que por las propias llamadas (quien envía el mensaje, paga por ello). Esto permite al cliente controlar sus propios gastos; no importa cuantos mensajes se envíen, no serán cobrados a no ser que se responda. Algunos operadores han empezado a ofrecer correo electrónico a pasarelas SMS, donde un cliente puede enviar un mensaje a una dirección del tipo número_de_telefono@proveedor.com, pero no hay manera de cobrar por esto. También abre una puerta al envío masivo a una red de clientes, por eso muchos operadores establecen un límite en el número de mensajes que puede enviar un usuario a desde una dirección de correo electrónico concreta.

De modo parecido, los operadores de telefonía móvil no están dispuestos a responsabilizarse del tipo de servicios que ofrecen las compañías de paging. La mayoría de los mensajes enviados a un buscapersonas tiene su origen en una red normal de teléfonos (para buscar a alguien, el usuario llama a un operador humano, que transcribe el mensaje y lo transmite). Establecer centros que den empleo a gente es muy caro y muchos operadores, además, tienen intereses en negocios de sistemas de buscapersonas, que no están dispuestos a desmontar.

Los operadores han hecho, ligeramente, más progresos en interconectar sus propias redes de mensajes, para que los clientes de un operador puedan mandar mensajes a los de otro.

Los teléfonos celulares digitales usan tres tipos de servicios de mensajería, aunque sólo dos por ahora, son soportados por GSM.

SMS

El único estándar que ha alcanzado amplia aceptación es el Servicio de Mensajes Cortos (SMS). Comenzó como una gran parte de las características del GSM, pero desde entonces se extendió al resto de sistemas digitales sin excepción.

La gran limitación del SMS está dada en su propio nombre: los mensajes deben ser cortos. El GSM impone un límite de sólo 160 bytes o caracteres. El límite en la longitud está propiciado por el medio en que SMS es transmitido. Normalmente discurre a través de los canales de control, las mismas frecuencias o slots de tiempo utilizados para la información de establecimiento de llamada. Esto significa que los usuarios pueden enviar o recibir mensajes SMS mientras que están haciendo una llamada, aunque necesiten un equipo de manos libres para leer en la pantalla o teclear.

SMS es conocido como un servicio de store-and-forward, porque los mensajes pueden ser almacenados durante unos segundos antes de ser transmitidos. En este aspecto, es como un correo electrónico o como el servio normal de correo: el que lo envía no tiene garantía que el mensaje no será recibido en un determinado límite de tiempo y no se tiene el conocimiento automático de cuando esto sucede. Esto hace al SMS inadecuado para servicios interactivos.

Algunas mejoras del SMS permiten que los mensajes sean enviados desde la red al teléfono móvil y son conocidos como Terminate Only. Otras permiten mensajería en dos direcciones, llamas Terminate and Originate. El

soporte SMS necesita estar constituido en el teléfono. Todos los teléfonos GSM pueden enviar SMS y la mayoría de los fabricados después de 1996 pueden también originarlos, pero para otros estándares, muchos teléfonos de finalización pobre solamente pueden enviar o carece de SMS por completo.

CBS

Si la misma información necesita ser enviada a muchos usuarios diferentes, la difusión es más eficaz que transmitir por separado cada uno de ellos. Detrás de esta teoría esta el Servicio de Difusión Celular (CBS), es una variante del SMS utilizado sólo en GSM. Cada mensaje es conocido como una página y sólo puede tener 93 bytes de largo, pero se pueden concatenar un total de 15 páginas para un mensaje de una longitud final de 1395 bytes, suficiente para varios párrafos de texto o un programa corto.

A pesar de estas claras ventajas técnicas, el CBS no ha sido ampliamente utilizado. La principal razón es que no ofrece un modo para cobrar los servicios por parte de los operadores; por definición, un mensaje difundido puede ser recibido por cualquiera que se encuentre en una celda. [En teoría, debería de haber un modo de superar esto utilizando las facilidades de encriptación del GSM, pero ningún operador lo ha conseguido aún. Un mensaje tiene que ser pensado para un solo usuario o para todos los que estén en una celda].[2]

USSD

Las redes GSM tienen acceso a una tercera tecnología de mensajería, Datos de Servicios Suplementarios ni Estructurados (USSD). Como el SMS, utiliza el canal de control y puede operar cuando se esté usando el teléfono.

---

[2] WAP guía práctica para usuarios, Autor Andy Dornan, Ediciones ANAYA MULTIMEDIA (GRUPO ANAYA S.A.), 2001, página 145.

Los mensajes son ligeramente más largos, con un máximo de 182 bytes comparados con los 160 del SMS bajo GSM.

La ventaja principal que tiene el USSD es que es de conexión orientada. Esto quiere decir que la red establece una conexión con el teléfono entes de enviar los datos, del mismo modo que una llamada. El que lo envía sabe cuando se recibe el mensaje y si es necesario, se puede enviar una respuesta rápida a través de la misma conexión. Esto lo hace ideal para aplicaciones interactivas como navegar en la red.

USSD está diseñado para que accedan a él programas manejados por el teléfono, a través de una interfaz de usuario dirigida por menú.

APLICACIONES DE LA MENSAJERIA

Aunque la mensajería no es un autentico correo electrónico ni servicio de buscapersonas, está empezando a encontrar aplicaciones propias. Las más populares son:

- Notificación de buzón de voz — Casi todos los operadores ofrecen a sus clientes un servicio de buzón de voz, que se utiliza cando el teléfono está apagado o fuera de cobertura. Una vez que los clientes establecen una conexión con la red, reciben un SMS diciéndoles que tienen mensajes en el buzón de voz. La mayoría de los teléfonos interceptan estos mensajes e iluminan un icono para que el cliente no tenga que leer y borrar “Tiene coreo” varias veces al día.
- Reprogramación — La mayoría de los teléfonos pueden tener una parte de su software actualizado a distancia, usando mensajes SMS o CBS. Los clientes también pueden descargar nuevos tonos.
- Servicios de difusión informativa — Muchos operadores están empezando a ofrecer servicios sencillos de información utilizando CBS. Son gratuitos

para el cliente y aportan titulares de noticias cuando ocurren. Desafortunadamente, no hay manera de elegir entre tipos de titulares distintos, todo se difunde a todo el mundo. Si los clientes quieren noticias, tienen que quedarse con el paquete entero de finanzas, deportes e información meteorológica.

- Servicios de información especializada — SMS puede utilizarse para informar a la gente sobre eventos determinados, como cuando las acciones llegan a un proceso determinado o los vuelos aéreos llegan retrasados. Los operadores normalmente cobran un cargo extra por este tipo de servicio, pero a los clientes no están interesados en la tarifa.

EL KIT DE HERRAMIENTAS SIM

En el corazón de cualquier teléfono GSM está un SIM (Módulo de Identidad del Suscriptor), una minúscula tarjeta inteligente que almacena la identidad de cada cliente y otras informaciones, como su agenda de teléfonos. [La teoría era que las tarjetas podían intercambiarse entre teléfonos que operaban en distintas frecuencias y que los propios teléfonos se podrían actualizar reemplazando el SIM en vez de cambiar el teléfono entero. La primera parte funcionó, pero se hizo innecesario con la llegada de teléfonos de banda dual y de triple banda. La segunda parte no funcionó, debido a las mejoras que se realizan a la batería y en la tecnología electrónica normalmente hacen que el resto del teléfono se convierta en algo obsoleto mucho antes que la tarjeta.][3]

Algunas personas les gustan guardar dos o tres tarjetas SIM con el fin de poder utilizar sus teléfonos para acceder a redes diferentes sin pagar cargos

[3] WAP guía práctica para usuarios, Autor Andy Dornan, Ediciones ANAYA MULTIMEDIA (GRUPO ANAYA S.A.), 2001, página 148.

de roaming. Algunos teléfonos han sido diseñados para admitir dos o más tarjetas para este propósito, almacenando la información como el directorio personal en el propio teléfono.

El kit de herramientas de aplicaciones SIM es un estándar que da al SIM acceso al conjunto entero de características del teléfono, incluyendo servicios de mensajería. El operador puede enviar mensajes al SIM para reprogramarlo, añadiendo características específicas o incluso aplicaciones. Esto varía desde un programa sencillo que permite cambiar entre líneas de negocios y personales, hasta uno mas complicado que ofrece banca móvil. Cuando el programa está en ejecución, puede entonces variar en mensajes mientras el cliente introduce o solicita datos. Una aplicación de banca móvil puede solicitar el balance del usuario utilizando USSD, para luego enviar una orden de transferencia de dinero vía SMS.

La gran desventaja del kit de herramientas SIM es que muchos teléfonos y operadores no lo soportan. A pesar de que fue introducido por primera vez en 1998, ha sufrido una aceptación mucho más lenta que la mayor parte de la tecnología sin cable. Se percibe como algo demasiado complejo y ha sido eclipsado por otras tecnologías que prometen (pero normalmente no proporcionan) navegación móvil por Internet.

LA WEB SIN CABLE

De todas las nuevas aplicaciones de datos móviles de actualidad, ninguna despierta mayor entusiasmo que la navegación de la Web sin cable. La red ha demostrado ser una aplicación “salvaje” para las redes de datos fijas, convenciendo a millones de personas para comprar su primer computado. La industria sin cable ha ideado varios esquemas para presentar el contenido de la red de teléfonos móviles o aparatos PDA.

Transferir la Web a teléfonos móviles presenta varios desafíos, incluyendo un retraso variable y un diseño de aparatos de entrada. Pero los dos que más han preocupado a la industria móvil son:

- **Pantallas pequeñas —** Muchos teléfonos móviles pueden mostrar solamente una líneas de texto, limitando de esta manera profundamente la experiencia "multimedia" de la red. Los terminales PDA cuentan con pantallas más grandes, pero incluso estas tienen sus límites. La mayor parte de las páginas se encuentran diseñadas para tener una resolución mínima de 640x480 píxeles, y algunas requieren una resolución mejor.
- **Baja capacidad —** La mayoría de los sistemas existentes de teléfonos celulares permiten velocidades de datos muy lentos. Incluso las tecnologías avanzadas, como los GPRS, podrán elevarla solo hasta los módems actuales. Por comparación, quienes navegan por la red corporativa o una línea domestica DLS están acostumbrados a velocidades limitadas simplemente por la propia congestión de la Internet.

WAP

[En junio de 1997, el operador estadounidense Omnipoint decidió desarrollar un servicio Web móvil. No tenia idea de cómo abordarlo, ni de que tecnología emplear, por eso abrió un proceso de ofertas competitivas. Cualquiera que pudiera presentar una propuesta de datos sin cable, podía enviarla a Omnipoint y explicar cómo daría servicio a los clientes de la compañía y aumentaría los beneficios.

Cuatro compañías aceptaron el desafío. Las compañías de teléfonos móviles Nokia, Ericsson y Motorola presentaron sus propias variantes de mensajería, mientras que Unwired Planet envió HDML. Todas tenían sus ventajas, pero la

gran desventaja de estar en propiedad: los clientes de Omnipoint tendrían que comprar teléfonos y software a un proveedor en particular, del mismo modo que la mayoría de los usuarios de PC están atados a Microsoft.

Omnipoint dijo a los postores que no aceptaría una solución con sistema de propiedad (ellos tendrían que unirse y diseñar un estándar). El resultado fue la formación del Foro WAP, que en su origen compondría de estas cuatro compañías, aunque no a la propia Omnipoint. Un año después, abrieron el Foro a nuevos miembros y anunciaron la primera versión de WAP.][4]

Desde la aparición de la primera versión de WAP en abril de 1998, el Foro WAP ha desarrollado tres versiones más de este protocolo. El Foro WAP se encuentra en estos momentos la última versión que se conoce por el nombre de «WAP June 2000», que es la versión que cumple con las especificaciones de WAP de junio de 2000.

Los principales objetivos para el futuro de WAP se centran en incrementar la seguridad, reducir las interferencias de las tarjetas, mejorar la tecnología y consolidar a WAP en el mercado.

Un área de gran interés para Foro WAP ha sido la evolución para soportar los servicios multimedia. WAP v1.1 y v1.2 su protocolos abiertos que permiten el transporte de muchas maneras de contenidos multimedia. Sin embargo, algunos servicios multimedia requieren notables avances en WAP.

WAP ha sido diseñado para ser casi totalmente independiente de la tecnología que lo soporte, en el futuro podrá lograr gran compatibilidad con la tecnología inalámbrica de tercera generación.

---

[4] WAP guía práctica para usuarios, Autor Andy Dornan, Ediciones ANAYA MULTIMEDIA (GRUPO ANAYA S.A.), 2001, página 158-159.

PROTOCOLOS

Hablando de redes, un protocolo es un conjunto de reglas para la comunicación entre aparatos similares. En general, regula situaciones como de quien es el turno de transmisión, como se detectan o se fijan errores y como distinguir datos de otras señales que los acompañan.

La arquitectura WAP resulta claramente de una gran similitud a la tradicional de Internet (TCP/IP). Además, se han modificado ciertas capas para adaptarlas a los requerimientos especiales para los que esta tecnología ha sido creada, como son ancho de banda limitado, transmisión por medio aéreo, estabilidad de la conexión, entre otras, la figura 4.1.3.1.1 muestra esta arquitectura

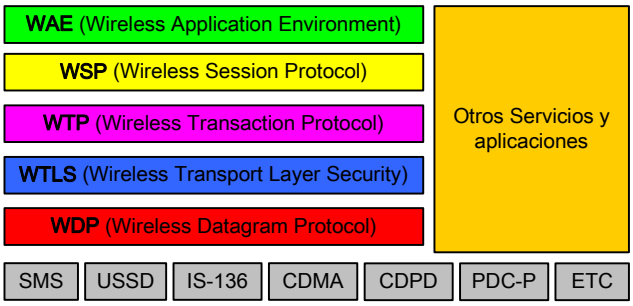


Figura 1 Arquitectura WAP

Las características principales que cumple la arquitectura WAP son:

- Integración en lo máximo posible con los estándares actuales.
- Arquitectura por capas escalables.
- Compatibilidad con todo tipo de redes inalámbricas.
- Optimizada para portadoras de banda estrecha con alto índice de latencia.
- Optimizada para un uso de dispositivos con limitaciones.

- Seguridad a nivel de aplicación y transmisión de información.
- Ofrecer un modelo de desarrollo de servicios para telefonía de forma integrada.
- Estándar que integre a todo tipo de fabricantes, modelos de dispositivos, etc.

1. **Capa de aplicación (WAE) —** Es un entorno de aplicación de propósito general, basado en la combinación del World Wide Web y tecnologías de Comunicaciones Móviles. Soporta aplicaciones diferentes:
   - WML (Wireless Markup Language), la variante que está basada en XML de HDML.
   - WMLScript (o WMLS), un lenguaje de escritura basado en JavaScript.
   - WTA (Wireless Telephony Application), una interfaz que permite a Wap acceder a características del teléfono como la agenda de direcciones, e incluso hacer llamadas. Al igual que WTLS, no se añadió al estándar hasta la versión 1.2, por eso no se desarrolló adecuadamente hasta la primera remesa de teléfonos WAP, que apareció a principios del año 2000.
   - WTAI (Wireless Telephony Application Interface), es una interfaz utilizada en los terminales móviles para operaciones locales de control de llamadas (recepción, iniciación y terminación) y acceso a listines telefónicos.
2. **Capa de sesión (WSP) —** Este protocolo proporciona a la Capa de Aplicación (WAE) interfaz con dos servicios de sesión :
   - Un servicio orientado a conexión que funciona por encima de la Capa WTP (Wireless Transaction Protocol), está diseñado para asegurar que los datos que han sido enviados correctamente; comprueba mensajes de error del receptor y reenvía los datos cuando es necesario.

- Un servicio no orientado a conexión de la Capa (WTP) y que funciona por encima es esta es WSP (Wireless Sesision Protocol), a veces pronunciado whisper. Por si mismo no hace nada, pero permite que el tráfico circunvale WTP o WTLS si no se necesitan corrección de error o encriptación. Proporciona servicio de datagramas seguro o no seguro.

Esta capa proporciona las siguientes funcionalidades:

- Establecimiento y liberación de conexiones entre cliente y servidor.
- Intercambio de información entre cliente y servidor.
- Negociación de las características del protocolo.
- Suspensión y reanudación de la sesión.

3. Capa de transacciones (WTP) — Este protocolo funciona por encima de un servicio de datagramas ya sean seguros como no seguros, y proporciona las siguientes funcionalidades:
   - Proporciona los servicios necesarios para soportar las transacciones, estos servicios pueden ser de tres clases: Peticiones inseguras de un solo camino.
   - Peticiones seguras de un solo camino.
   - Transacciones seguras de dos caminos.

   También proporciona seguridad en las transacciones.
4. Capa de seguridad (WTLS) — La capa inalámbrica es un protocolo basado en el estándar SSL (Secure Sockets Layer), utilizado en el entorno Web, y se usa cuando se conecta a páginas Web que piden información comprometida, como los detalles de la tarjeta de crédito, esta capa proporciona a las capas de nivel superior de WAP una interfaz de servicio de transporte seguro, que lo resguarde de una interfaz de transporte inferior.

   Las funcionalidades de esta capa son las siguientes:

- Integridad de los datos: se asegura que la información intercambiada entre el terminal y el servidor de aplicaciones, no haya sido modificada.
- Privacidad de los datos: se asegura que la información intercambiada entre el terminal y el servidor de aplicaciones, no pueda ser captada ni entendida por elementos externos a la comunicación.
- Autentificación: se ofrecen servicios para determinar la autenticidad del terminal y del servidor de aplicaciones.
- También puede ser utilizado para el establecimiento de una comunicación segura entre terminales.

5. Capa de transporte (WDP) — El Protocolo Inalámbrico de Datagramas (WDP) proporciona las siguientes funcionalidades:

   Proporciona un servicio fiable a los protocolos de las capas superiores de WAP.

   Permite la comunicación de forma transparente sobre los protocolos portadores CDMA, SMS, GSM... Incluye el control de errores y la priorización del tráfico.

PORTADORES

WAP está diseñado para ser independiente del portador, lo que significa que puede funcionar con cualquier tecnología sin cable. Realmente está diseñado para sistemas digitales celulares, pero también funciona con radios de alcances más cortos. Por ejemplo, [la compañía de telecomunicaciones finlandesa Sonera ha demostrado que WAP y la radio de un solo chip Bluetooth pueden operar conjuntamente, saltándose la red de teléfonos celular cuando dos aparatos están juntos muy cerca uno del otro. Está probando máquinas de Coca-Cola equipadas con un servidor WAP y un transmisor-receptor Bluetooth, posibilitando a los usuarios

próximos acceder a un menú de bebidas. Si también tienen un chip Bluetooth, el menú aparece en la pantalla del micro navegador WAP y el costo de las bebidas que compre se añade a la factura de teléfono.][5]

La mayoría de los primeros auténticos servicios WAP funcionaban con GSM, normalmente necesitando que el usuario marcara un número de teléfono. Esto no es eficaz por dos razones. Significa que la línea del usuario está ocupada y que se le cobra por cada segundo que permanece conectado. Durante la mayor parte del tiempo, la conexión se malgasta.

El tráfico intermitente es ideal para la comunicación de paquetes, es que no requiere que se mantenga abierta la línea de teléfono. Algunos operadores han experimentado ejecutar WAP como SMS o USSD, permitiendo que el usuario haga llamadas de teléfono y navegue por la res simultáneamente, pero los pequeños límites de los servicios de mensajería lo hacen demasiado lento. [Muchos analistas piensan que WAP despegará realmente cuando integre con GPRS, la mejora de GSM que añade comunicación de datos y velocidades de datos más altas. También funciona bien, aunque no tan rápido, con servicios de conmutación.][6]

La figura 4.6.1 muestra el conjunto completo de protocolo WAP, además de algunos portadores más lentos por los que puede viajar.

ARQUITECTURA

Aunque Wap Utiliza sus propios protocolos, como lo muestra la figura 4.6.1, está diseñado para ser compatible con Internet. Las páginas escritas

---

[5] WAP guía práctica para usuarios, Autor Andy Dornan, Ediciones ANAYA MULTIMEDIA (GRUPO ANAYA S.A.), 2001, página 162.

[6] WAP guía práctica para usuarios, Autor Andy Dornan, Ediciones ANAYA MULTIMEDIA (GRUPO ANAYA S.A.), 2001, página 163.

con WML pueden viajar por Internet utilizando HTTP normal sobre TCP/IP y luego convertirse a WAP en la pasarela entre Internet y la rede sin cable.

La conversión del protocolo conlleva una debilidad en la seguridad, porque los dos conjuntos de protocolos utilizan diferentes sistemas de encriptación. Los datos se encriptan en Internet utilizando SSL; también se encriptan en la conexión sin cable utilizando WTLS.

Pero entre ambos, es vulnerable. Los operadores de páginas donde la seguridad es importante deben actuar como ISP o fiarse completamente de aquellos a través de los cuales sus clientes marcan. Normalmente es un operador móvil, aunque podría ser cualquiera.

WAP a menudo se describe que la mayoría de la inteligencia está contenida en la red, en lugar de en los teléfonos. Esto puede causar problemas de compatibilidad, particularmente en lo que concierne a WMLScript. Mientras que los navegadores normales de Internet pueden contener un intérprete para su lenguaje de escritura, los teléfonos móviles no pueden. En su lugar, la estructura ha de ser compilada por la pasarela, un proceso que depende del tipo de chip de procesado que esté dentro del teléfono celular. Para asegurar que WAP funciona realmente, las pasarelas tienen que ser probadas para cada procesador y luego actualizadas cada vez que surge un teléfono basado en un nuevo procesador.

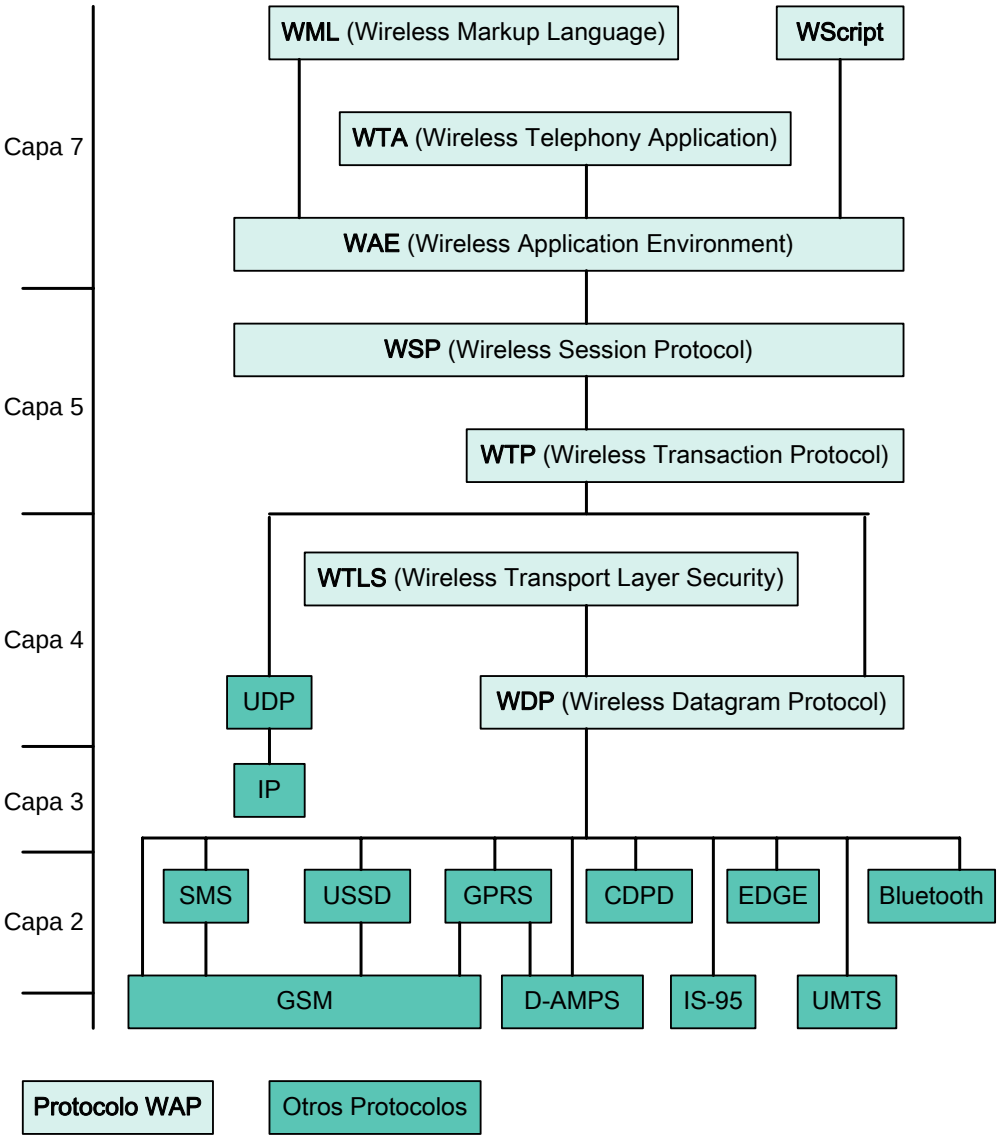


Figura 2 Conjunto de protocolos WAP

WWW:MMM

Los teléfonos WAP a menudo no tienen un aspecto diferente al de sus homólogos no capacitados para acceder a Internet. Pueden tener una pantalla ligeramente mayor o una rueda para navegar por ella, pero ni

siquiera eso supone una garantía. Muchos teléfonos han incluido pantallas simplemente para juegos o mostrar mensajes SMS.

Para hacer que los aparatos WAP sean mas fáciles de identificar, la mayoría tienen la marca WWW:MMM que viene de World Wide Web: Mobile Medio Mode.

DISEÑO DE PÁGINAS

Si quiere que la página sea accesible desde internet sin cable, le elección la encuentra en WAP. Pero primero se necesita entender la manera en que afecta al contenido de los métodos de acceso de los usuarios. Si la página está llena de artículos largos, se debe olvidar le elección de WAP –nadie va a leer un libro electrónico a través de un aparato que solamente puede mostrar unas cuantas palabras al mismo tiempo y además tiene que cargar cada párrafo por separado.

Lo mismo es aplicable también para los gráficos; del mismo modo, los negocios que dependen de la publicidad a través de banners perderán dinero incluso en el caso de que se conviertan al sin cable, porque tampoco hay lugar para ellos.

PAGINAS WAP

Una página WAP es más difícil de crear que una basada en C-HTML. Requiere un lenguaje nuevo, nuevos protocolos y posiblemente nuevos contenidos.

Considerando que se necesita desarrollar una página completamente independiente, se ha de doblar la cantidad de tiempo (para una sola persona) o el número de personas (parar una compañía grande) que se dedica a la

presencia en la Web, una alternativa es echar un vistazo a uno de los múltiples programas traductores disponibles. Muchas compañías dicen que tienen programas que convierten HTML en WML pero no hay que creer en sus afirmaciones, los lenguajes son tan diferentes que, para muchas páginas, los resultados no serían dignos de ser situados en al red.

INTRODUCCION A WML

El corazón de WAP desde el punto de vista del desarrollador, es el lenguaje llamado WML (Wireless Markup Language). Si ha trabajado con HTML (Hypertext Markup Language) parte del lenguaje WML puede resultar parecido, pero existen muchas diferencias que se pueden pasar por alto, muchas de esas diferencias vienen de la simplicidad de WML comparado con HTML.

Un ejemplo muy común, HTML brinda el control total del texto en un documento: se puede cambiar el tamaño de la letra, el tipo de letra, añadirle subrayado o negrita y tener el control sobre el tono del color del texto.

El único control que WML brinda sobre el texto son unos cuantos estilos (emphasis, strongemphasis, boldface, Italic, underline) y la habilidad de decidir si es mal grande o pequeño el texto de lo normal.

Perdiendo el control sobre la apariencia que provee HTML se fuerza a pensar en una forma diferente de escribir las páginas, enfocándose más en el contenido que en la apariencia. No todos los navegadores WAP soportan imágenes, tablas o tipos de texto.

Pero no todo es malo. Mientras que WML elimina muchas de las características, y como fue diseñado desde el comienzo para aplicaciones interactivas, muchos de los dolores de cabeza del mundo Web simplemente

desaparece. También tiene la ventaja de ser diseñado rápidamente, más bien que las características sean agregadas por diferentes personas sin ninguna coordinación apropiada.

CARDS (CARTAS)

La razón de la etiqueta <card> es porque la estructura de WML es un poco diferente a HTML. En lugar de tener un contenido largo y extenso, un archivo de WML representa una baraja (deck) de cartas (cards). Solamente una carta es mostrada a la vez, pero se puede pasar de una carta a la otra y guardarlas en una baraja en el mismo archivo. Cada carta se comporta muy parecida a una página HTML; así que se puede pensar que una baraja de WML es similar a un grupo de páginas HTML agrupadas. Las cartas en al baraja son descargadas al mismo tiempo, así que el usuario debe esperar una sola vez, y accede a las cartas casi instantáneamente.

[No es recomendable sobrecargar la baraja, si se tiene una baraja con 30 cartas, el usuario se aburrida esperando que se descargue toda la baraja. Es difícil poner un límite al número de cartas que debe poseer una baraja, pero si se utilizan mas de cinco o seis cartas en una misma baraja, se debe pensar mejor en como están organizadas las páginas.}[7]

ELEMENTOS

Los elementos utilizados en WML son muy parecidos a los elementos de HTML.

[7] Learning WML y WMLScript, Autor Martin Frost, O'REILLY, Octubre 2000, página 7.

El elemento <wml>

El elemento <wml> sirve de propósito como el elemento <html> en páginas HTML; encierra la entrada de la baraja. Efectivamente, una baraja de WML consiste en un solo elemento <wml>. No toma ningún atributo.

El elemento <head>

El elemento <head> en WML es similar al electo <head> en HTML; este elemento demarca un lugar para la meta-información acerca del documento que será archivado. La meta-información acerca del documento mismo, más bien que su contenido. Si se presenta, este elemento debe ser la primera cosa dentro del elemento <wml>. No toma ningún atributo.

El elemento <acces>

El elemento <acces> provee una forma sencilla de control de acceso a la baraja. Esto permite a una baraja especificar que solamente ciertas barajas pueden acceder a esta (estas barajas son conocida como refering URLs). Desde que este control es realizado en el navegador, no en el servidor, no existe seguridad en este mecanismo, y por lo tanto su uso es limitado.

Puede no haber más elementos <acces> en la baraja, y debe ser el primer elemento dentro del elemento <head>.

El elemento <meta>

El elemento <meta> coloca un ítem arbitrariamente de la meta-información en la baraja WML. Este ítem está estructurado un nombre y su valor. Se puede colocar varios elementos <meta> en el elemento <head>.

El elemento <card>

El elemento <card> permite indicar el titulo de la carta, posee un identificador que diferencia una carta de la otra.

El elemento <template>

El elemento <template> provee unas características interesantes que no se encuentran en HTML. Cuando se escribe una baraja, esta contiene varias cartas, en estas se pueden encontrar en su estructura partes comunes entre todas. Por ejemplo, es muy común que todas las cartas tengan un control para “atrás” y otro control para “inicio”. Esos controles normalmente son implementados como elementos <do>. Sería conveniente especificar esos elementos comunes una vez por baraja mejor que una vez por carta.

ATRIBUTOS

Los atributos afectan el comportamiento de todos los elementos. Los efectos de los atributos varían de acuerdo al elemento: el atributo title en el elemento <card> coloca un titulo opcional que será mostrado con al carta, mientras el atributo, src en el elemento <img> da el URL en donde la imagen puede ser encontrada. También se puede añadir el atributo align=”center” al elemento <p>, que centra el párrafo.

Una diferencia entre WML y HTML son las comillas de los valores de los atributos. En HTML, los valores de los atributos pueden aparecer en comillas sencillas (attr='value') en comillas doble (attr=”value”) y muchos navegadores también permiten sin comillas (attr=values), si embargo esto no es estrictamente válido. WML no permite los valores de atributos sin comillas: todos los valores debe aparecer con comillas dobles o sencillas.

ENTIDADES

El pedazo final de la sintaxis de WML para comenzar en el rango de los diferentes elementos es la entidad. El propósito de las entidades es representar símbolos que no pueden ser fácilmente escritos o que tienen un significado especial en WML. En al tabla 5.1. se muestra tres formas de entidades en WML.

Nombre	Decimal	Hexadecimal	Carácter
&quot;	&#34;	&#x22;	Comillas dobles (")
&amp;	&#38;	&#x26;	Ampersan (&)
&apos;	&#39;	&#x27;	Apostrofe (')
&lt;	&#60;	&#x3C;	Menor que (<)
&gt;	&#62;	&#x3E;	Mayor que (>)

Todas la entidades comienzan con ampersan (&) y terminan con punto y coma (;). Este punto y coma es muy importante: algunas de las páginas Web olvidan esto y causan problemas para lo navegadores que quieren corregir el HTML. Los navegadores WAP son parecidos a cerca de errores como este.

COMENTARIOS

Un comentario comienza con cuatro caracteres <!-- y termina con tres caracteres -->. Todo lo que aparezca dentro de estos dos marcadores, incluyendo entidades, texto es ignorado. Por ejemplo

<!--Esto es un comentario-->

```
<!--Este es un comentario
    que se extiende
    varias líneas-->
```

TAREAS Y EVENTOS

Las tareas (tasks) y los eventos (events) no tienen equivalencia real en HTML. En algunos casos se puede utilizar JavaScript para alcanzar efectos similares.

Tareas

Una tarea en WML es un elemento que especifica una acción para ser realizada por el navegador. Por ejemplo, la acción de cambiar a una nueva carta es representada por una tarea <go>, y la acción de regresar a la anterior carta visitada es representada por la tarea <prev>. Las tareas encapsulan toda la información requerida para realizar la acción.

Para observar como las tareas son utilizadas en el contexto se considera el elemento <do>, que representa algún tipo de control que el usuario puede activar, como un menú ítem en el teléfono celular. Un elemento <do> no es por si solo una tarea. Más bien, contiene un subelemento que especifica la acción a realizar cuando el usuario activa el control.

Un elemento <do> que cuando se activa, simplemente activa el valor value a la variable test, puede ser escrito así:

```
<do type="acept">
    <refresh>
        <setvar name="test" value="value"/>
    </refresh>
</do>
```

Eventos

Un evento en WML es simplemente algo que puede suceder para algún elemento en algún momento. Por ejemplo, incorporando un elemento <card> acciona un evento en el elemento <card>, y seleccionando <option> de la lista de selección acciona un evento en el elemento <option>.

Por ejemplo, el elemento <option> declara un ítem en la lista de selección. Cuando este ítem es seleccionado, acciona un elemento onpick en el elemento <option>. Suponiendo que el elemento fuera declarado sin ningún manejador de evento, de esta forma:

```
<option>
    blue
</option>
```

En este caso, el evento onpick es ignorado, desde que no halla un manejador. Si la opción es declarada como:

```
<option>
    <onevent type=”onpick”>
        <go href=”#blue”/>
    </onevent>
    blue
</option>
```

El evento onpick es manejado por la ejecución de la tarea <go>, enviando al navegador a una nueva carta.

INTERACCIONES

Este capitulo cubre todas las características que WML provee al recibir las entradas del usuario, y muchas de estas son mas poderosas que sus equivalentes en HTML.

La razón principal para este poder extra es que WML posee variables. En HTML, se puede tener control como los menús desplegables, entradas en textfield, pero estos solo se pueden utilizar en algunos casos.

El elemento <input>

Este elemento es utilizado cuando el usuario necesita ingresar un string u otro tipo de texto. Usualmente, esto debe permanecer lo mas corto posible.

Este elemento también puede ser utilizado para ingresar contraseñas u otro tipo de información sensible. En estos casos, el elemento puede ser configurado para que en estos casos no se despliegue el texto de igual manera a como se ingresa. También es utilizado para ingresar números. En casos como estos, cuando el rango de caracteres es restringido, es posible crear un formato, que puede acelerar el ingreso de los mismos.

El elemento <select>

El otro control que WML provee es una selección desde una lista de ítem. Este reemplaza muchos tipos de control, como checkboxes, menús desplegables, en su forma mas simple, el elemento <select> provee un atributo iname dándole un nombre a la variable. Dentro de <select> está la lista de elementos <option>. Seleccionando una opción igual a la variable al ítem escogido, de acuerdo al número del ítem, comenzando desde uno. Por ejemplo:

```
<select iname=”animal”>
    <option>Spider</option>
    <option>Lion</option>
    <option>Fish</option>
</select>
```

Seleccionando Spider la variable animal en 1, a Lion la variable animal en 2, Fish la variable animal en 3.

Cuando un ítem tiene el atributo multiple=”true” permite seleccionar mas de un ítem al mismo tiempo de la lista.

El elemento <do>

Los elementos <input> y <select> proveen un alto nivel de control para los usuarios, pero algunas veces solo se quiere un simple menú ítem o botón. En estos casos, el elemento <do> es exactamente lo que se necesita.

Un elemento <do> contiene únicamente la tarea que será realizada cuando el elemento es activado.

TEXTO Y SU FORMATO

Es el momento de hablar del texto simple y ver que se puede hacer con él para volverlo mas interesante. Esta es un área en donde WML es muy parecida en comparación con HTML que provee muchas características para cambiar el tamaño, el tipo de letra, etc.

El elemento <p>

EL elemento <p> marca un párrafo de texto en una carta en WML. Todo el cuerpo del texto, controles de usuario (como los elementos <input> y <select>) e imágenes deben aparecer dentro de un elemento <p>. Muchos navegadores ignoran cualquier cosa que esté por fuera de <p>, pero algunos rechazan la baraja y se rehúsan a mostrarla. La excepción a esta regla es el elemento <do> que puede aparecer afuera o dentro de <p>. (Ahora es mejor ubicarlo fuera de <p>)

El elemento

El elemento <br> es uno de los más simples en WML. No lleva atributos y siempre es especificado como un elemento vacío, <br/> hace un cambio de línea en un párrafo de texto: cuando un navegador encuentra un <br/> comienza una nueva línea. Como por ejemplo.

```
<P>Texto<br/>mas texto</p>
```

Utilizando otro párrafo

```
<p>Texto</p><p>mas texto</p>
```

Formato del caracter

El soporte del formato de carácter en WML es limitado comparado con HTML. No hay soporte para especificar el color, tipo de texto, además el tamaño del texto esta limitado a “bigger” o “smaller”.

Tablas

Las tablas son unas de las menos características soportada por WML. La razón de esto es que mostrando las tablas apropiadamente requiere gran espacio en la pantalla.

WML incluye un número de filas, cada una contiene un número de celdas. Las celdas pueden contener varias líneas de texto y todas son consideradas parte de la misma fila.

INTRODUCCIÓN A WMLScript

WMLScript provee variables, tareas y eventos para hacer la interacción mucho más fácil, pero hay tareas que no pueden realizarse solamente con WML. Si la información debe ser verificada, por ejemplo que un número de teléfono es realmente un número o que un número de tarjeta de cerdito tenga la cantidad de dígitos necesarios, estos chequeos deben hacerse fuera de WML.

El camino mas obvio para hacer esta verificación es la forma en que se ha hecho esto en al Web desde hace mucho tiempo: se envía la información al servidor y deja que el servidor haga la verificación. Este acercamiento tiene un gran inconveniente, el tiempo que toma en ir y volver la información del servidor. Piense en los sitios Web donde se ingresa la dirección, después oprime clic en OK, y espera un buen tiempo antes de que le indique que no ha escrito el numero de teléfono.

Una mejor vista es hacer la validación, en el navegador, antes de mandar algo al servidor. Esto es exactamente para lo que fue diseñado WMLScript.

QUE ES WMLScript

WMLScript está basado en JavaScript pero es mucho más simple en orden de reducir los requerimientos en el celular que corre el navegador.

Si alguna vez ha hecho un programa en Java, C, C++, JavaScript o cualquier otro lenguaje de programación, en WMLScript la sintaxis es muy parecida.

En WMLScript hay dos tipos de comentarios: como en C comenzando con /* y terminando con */, y como en C++ comenzando con // hasta el final de la línea. Por ejemplo:

```
/* Un comentario en C */
//Un comentario en C++

/*Este tipo de comentarios puede
tener varias líneas */
//Estos comentarios deben tener
//uno nuevo en cada línea de comentario
```

En WMLScript los nombres, las variables, los valores y todo lo demás son sensibles a las mayúsculas y minúsculas. Si embargo el espacio es un poco flexible, por ejemplo

```
if (x)
    a = b;
else if (y)
    c = c + a;
```

En donde el único espacio requerido es entre else e if, porque se debe separar las palabras claves.

VARIABLES Y TIPOS DE DATOS

WMLScript es un lenguaje con poca estructura. Esto quiere decir que nunca se especifica el tipo de datos o la función de retorno. Todas las expresiones tiene un tipo de dato internamente, pero WMLScript convierte los valores entrantes y salientes en los diferentes tipos que son requeridos, sin necesidad de hacerlo manualmente.

Por ejemplo, el valor 123 es un entero pero si se pasa por la función string.length ( ) que espera un string implícitamente lo convierte en una cadena “123” con una longitud de 3.

Tipos de datos

WMLScript tiene cinco tipos de datos: cadena (string), entero (integer), punto flotante (flota-point), booleano (bolean) y el tipo especial invalido (invalid). Todo valor en WMLScript pertenece a alguno de estos tipos, auque pueden ser convertidos a otros.

- String — Puede ser una secuencia de caracteres. Pueden ser encerradas en comillas dobles o sencillas.
- Integer — Los valores enteros están guardados como un entero de 32 bits. Este puede ser especificado en octal, decimal o hexadecimal, pero no afecta el número en si. Los enteros siempre son mostrados como decimales cuando son convertidos a cadenas. Las letras en hexadecimal pueden ser en mayúscula o minúscula.
- Bolean — Representan ya sea falso o verdadero. Estos deben ser escritos en minúsculas.

- Flota-point — Los valores de punto flotante son almacenados en formatos de la IEEE single-precision, que es un formato de 32 bits. Esto permite números con una aproximación de $3.4e^{+38}$ para ser almacenado.
- Invalid — El propósito es representar un valor que no es ninguno de los otros. Es usualmente una condición de error. Por ejemplo, la expresión (1/2) da un punto flotante 0.5. La expresión (1/0) no puede ser evaluada, así el resultado es invalido.

Variables

WMLScript provee solamente variables locales y acceso a las variables WML por el navegador. No hay variables locales y no se puede tener valores persistentes entre llamada y llamada. Cuando una función devuelve un valor, este valor se pierde.

DECLARACIONES

Las declaraciones proveen estructuras de control como loops y condicionales, también como la habilidad de devolver un valor de una función.

Casi todas las características son similares a las encontradas en los lenguajes como C y Java, excepto que WMLScript no provee la misma variedad de selección.

Esto no debería representar ningún inconveniente, sin embargo WMLScript ha sido diseñado para operaciones simples, como validar la información entes de enviarlas al servidor.

Condicionales

Las declaraciones condicionales en WMLScript tienen un comportamiento similar a C. Por ejemplo:

```
if (condición)
    Declaración-cierta
```

Un else puede ser agregado.

```
if (condición)
    Declaración-cierta
else
Declaración-falsa
```

Loops

Hay dos tipos de estructuras de loops: for y while.

- **While —** Se ejecuta su cuerpo mientras la condiciones verdadera. Si la condición es falsa al comienzo, el cuerpo nunca es ejecutado. Ejemplo:

  ```
  while (condición)
      cuerpo
  ```

- **For —** Es muchos mas flexible y poderoso. Ejemplo:

  ```
  for(inicializado, condición, incremento)
      cuerpo
  ```

- **Break —** La declaración break puede aparecer en el cuerpo de cualquier loop, se declara así:

  ```
  break;
  ```

  Si es ejecutado, se salta fuera del loop a la línea de código siguiente. Normalmente es utilizado cuando una condición extra es detectada en la mitad del loop, lo que indica que el loop debe detenerse.

- Continue — La declaración continue también controla el comportamiento del loop, pero además de detener el loop, simplemente se salta el resto de la iteración actual y continua con la siguiente. Se declara de esta forma:

  continue;

  En la declaración, se salta hasta el final del cuerpo y este vuelve a evaluar la condición. En al condición for ejecuta la parte del incremento.
- Return — La declaración return termina la ejecución de la función actual. Tiene dos formas:

  return;

  Y

  return expresión;

FUNDAMENTOS DE LOS SERVLETS

[Los Servlets son la respuesta de la tecnología Java a la programación CGI. Son programas que se ejecutan en un servidor Web y construyen páginas Web.

Hay programas que corren en el servidor Web, actuando como una capa intermedia entre el pedido entrante de navegador Web u otro cliente http y base de datos o aplicaciones en el servidor HTTP.][8] Su trabajo es:

1. **Tener listo todos los datos enviados por el usuario —** esta información es usualmente ingresada en forma de una pagina Web, pero también puede provenir de un applet de Java o un cliente HTTP.
2. **Generar los resultados —** este proceso puede requerir hablar con la base de datos, ejecutar una llamada RMI o COBRA, invocando una aplicación, o computando el resultado directamente.
3. **Enviar el documento de vuelta al cliente —** este documento puede ser enviado en la formato de texto (HTML), binario (imágenes GIF), o incluso en un formato comprimido como gzip.

Muchos pedidos de clientes pueden ser respondidos devolviendo documentos pre-compilados, y esa respuesta es manejada por el servidor sin tener que invocar el servlet. En muchos casos, si embargo, un resultado estático no es suficiente, y las paginas necesitan ser generadas por cada

[8] http://www.apl.jhu.edu/%7Ehall/java/Servlet-Tutorial/Servlet-Tutorial-Overview.html

pedido. Por eso hay muchas razones por las cuales una página se debe compilar en el tiempo porque:

- La página Web está basada en datos enviados por el usuario — Por ejemplo, las páginas de resultados de los motores de búsqueda se generan de esta forma, y los programas que procesan pedidos desde los sitios de comercio electrónico también.
- Los datos cambian frecuentemente — Por ejemplo, un informe sobre el tiempo o páginas de cabeceras de noticias podrían construir la página dinámicamente, quizás devolviendo una página previamente construida y luego actualizándola.
- Las páginas Web que usan información desde bases de datos corporativas u otras fuentes — Por ejemplo, se usa esto para hacer una página Web en una tienda on-line que liste los precios actuales y el número de artículos en stock.

ESTRUCTURA BASICA DE LOS SERVLETS

Un servlet maneja peticiones GET; para aquellos que no son familiares a HTML, es un tipo usual de navegador para páginas Web. Un navegador genera este pedido cuando un usuario escribe una dirección URL, desde una pagina Web. Los servlets pueden manejar fácilmente los pedidos POST, que es generado cuando alguien somete una forma HTML que especifica METHOD=”POST“. Para ser un servlet, una clase debe extender de HttpServlet y sobrescribir el método doGet o doPost, dependiendo si los datos han sido enviados por GET o POST. Si se desea que el mismo servidor maneje tanto GET como POST y que tome las mismas acciones

para los dos, simplemente se debe tener doGet que llame doPost, o viceversa.

Estrictamente hablando, HttpServlets no es el punto de inicio de los servlets, desde que los servlets pueden, en principio, extender correos, FTP, u otros tipos de servicios. Los servidores para estos ambientes podrán extender una clase derivada de GenericServlet, la clase padre de HttpServlet. En la práctica, los servlets son utilizados casi exclusivamente para servidores que se comunican vía HTTP.

UN SERVLET QUE GENERA HTML

Muchos servidores generan HTML. Para compilar HTML, se necesitan los siguientes pasos:

1. Decirle al navegador, que se esta enviando de vuelta HTML.
2. Modificar la declaración println para compilar una pagina Web legal.

[El logro del primer paso para enviar el http conten-type en el header. En general, el header es puesto por el método setHeader de httpServletResponse, pero colocando el conten-type es una tarea muy común que también es un método de setContenType solo por este propósito. El camino para diseñar HTML, es con un tipo de texto/html, el código lucirá así:

Reponse.setContenType (“texto/html”);][9]

[9] Core Servlets and Java Server Pages, Autor Marty Hall, A Sun Microsystems Press/Prentice Hall PTR Book, Capitulo 2, página 26

CREACIÓN DE APLICACIONES UTILIZANDO WML Y SERVLETS

En este aparte hablaremos de cómo la tecnología Java y se integra con WML, nos referimos a los servlets, esos pequeños programas que actúan como pasarela entre nuestras aplicaciones WML y los recursos del servidor y los servidores de aplicaciones. En este capitulo vamos a ver las posibilidades que los servlets nos brindan para hacer nuestras aplicaciones mucho más interactivas con la ventaja añadida que el propio lenguaje Java proporciona que es la multiplataforma, de este modo vamos completando las posibilidades que existen actualmente para hacer páginas WML interactivas y dinámicas. No obstante, este capitulo presenta el inconveniente que para poder hacer un uso correcto de los servlets y sacarles el máximo provecho implica conocer por un lado el lenguaje Java y por otro conceptos relacionados con la invocación de métodos remotos o RMI así como tener unas nociones básicas de orientación a objetos, esto es, saber que es una clase, que es un objeto, que se entiende por invocación de métodos, en definitiva toda la jerga que forma parte de este paradigma y que debemos conocer sino queremos perdernos antes de empezar, sin embargo vamos a intentar que todos los conceptos nuevos que puedan aparecer y que puedan dar lugar a dudas queden explicados para que todos podamos sacar partido de las posibilidades que nos brindan los servlets para que al finalizar del capitulo se pueda decir que implementar un servlet es muy sencillo.

CICLO DE VIDA DE UN SERVLET

Los servlets surgen debido a la limitación existente que existían en los applets desarrollados para el Web en dos vertientes distintas: la imposibilidad de acceder a otro servidor que no sea el mismo en el que el applet se este

ejecutando y la limitación en el acceso a los servicios de esa máquina. Por otro lado un applet es un programa que se descarga desde el Web y se ejecuta en el navegador y puede resultar relativamente sencillo llegar al código fuente del mismo el cual puede verse comprometido seriamente dejando al descubierto información que puede que no queramos que se vea. Un servlet no es más que una aplicación que se ejecuta en un servidor Web quedando a la espera para resolver peticiones efectuadas por los clientes. Mediante los servlets se puede tener acceso a otros servidores y acceder a la información que en ellos haya contenida, por ejemplo una base de datos. Según todo lo expuesto anteriormente los servlets podemos verlos como la evolución lógica de los CGI, ya que estos últimos no son más que aplicaciones cuyo destino es resolver peticiones hechas por los clientes que se generan de un número de inconvenientes considerables como era el hecho del lenguaje de programación utilizado (generalmente lenguajes interpretados), un rendimiento muy bajo, baja portabilidad, dificultad de comunicación entre CGIs, etc. que se han visto superados por la aparición de los servlets.

Los servlets se pueden invocar desde un formulario, un applet, otro servlet y lo que a nosotros nos interesa desde una aplicación WML, el único requisito imprescindible que necesitamos tener es que nuestro servidor Web sea capaz de ejecutar servlets y aquí entramos en una problemática de que motor utilizar, por un lado si estamos usando Internet Information Server, Apache o Netscape podemos hacer uso de JRun o de ServletExec que se integran a la perfección con estos servidores Web. Otra posibilidad es hacer uso del Java Web Server de SUN, el cual incorpora un motor para ejecución de servlets dentro del propio servidor Web.

CARACTERÍSTICAS DE LOS SERVLETS

- Son independientes de la plataforma en donde se ejecutan.
- Son muy rápidos (en comparación con los CGIs)
- Se pueden comunicar distintos servlets entre si.
- Son muy seguros (usa el Security Manager de Java)
- Son más eficientes dado que múltiples llamadas sobre el mismo servlet no origina la ejecución de distintos procesos sino que crea threads de ejecución una vez que ya existe el proceso del servlet ejecutándose en la máquina

La plataforma más utilizada para la ejecución de servlets, JSP, EJB entre otros relacionados es JRun que junto con Internet Information Server 4.0 o Apache hacen un conjunto que permite obtener muy buenos resultados con una configuración mínima.

Visto todo lo anterior podemos decir que el ciclo de vida de un servlet es un proceso que consta de cuatro pasos básicos que se repiten siempre:

1. El cliente solicita una petición a un servidor mediante una URL.
2. El servidor recibe la petición y la procesa, en este caso el servidor detecta que es un servlet o JSP e inicia la ejecución del mismo sino se encuentra ya en ejecución alguna instancia del mismo.
3. El servlet procesa la petición y la retorna al cliente.
4. El servlet creado solo se destruye cuando el motor de servlets se para o bien cuando hay un fallo en el sistema.

EMPAQUETANDO LOS SERVLETS

En un ambiente de producción, múltiples programadores pueden estar desarrollando servlets para el mismo servidor. Colocando todos los servlets en el top del directorio de los servlets, resulta muy difícil de manejar el directorio y se pueden presentar conflictos en los nombres cuando accidentalmente dos programadores escogen el mismo nombre para el servlet. Los paquetes es la solución natural a este problema. Utilizando los paquetes cambia la forma en que son creados los servlets, el modo en que son compilados, y la manera en que son invocados.

Creando servlets en paquetes

Se necesitan solamente dos pasos para empaquetar los servlets.

1. Mover los archivos a un subdirectorio que concuerda con el nombre del paquete previsto.
2. Insertar la declaración del paquete en la clase.

Compilando los servlets en paquetes

Hay dos formas de compilar las clases que están empaquetadas.

1. Se ubica el subdirectorio del paquete en el directorio cuando el servidor Web espera que los servlets se envíen. Después se coloca el CLASSPATH que apunte al directorio del cual realmente contienen los servlets, así, el subdirectorio del menú del servidor es utilizado por el servidor Web. También se puede compilar normalmente desde el subdirectorio del paquete específico. Por ejemplo, si el directorio del servidor es C:\JavWebServer2.0\servlets y nombre del paquete (y el nombre de su subdirectorio) es nameservlets y se esta en Windows, se haría de la siguiente manera:

DOS> set CLASSPATH= C:\JavWebServer2.0\servlets;

DOS> cd C:\JavWebServer2.0\servlets\nameservlets

DOS> javac HolaMundo.java

2. Mantener el código fuente en un lugar diferente a las clases. Primero se coloca los directorios de los paquetes en un lugar que se crea conveniente. El CLASSPATH se refiere a esta ubicación. Segundo se utiliza la opción –d de javac para instalar la clase en el directorio que el servidor Web espera.

DOS> cd C:\MyServlets\nameservlets

DOS> set CLASSPATH=C:\MyServlets;

DOS> javac –d C:\tomcat\WebPages\WEB-INF\clases HolaMundo2.java

Invocando los servlets en paquetes

Para invocar un servlet que esta empaquetado se utiliza el siguiente URL.

http://host/servlet/packageName.ServletName

en vez de

http://host/servlet/ServletName

VENTAJAS DE LOS SERVLETS SOBRE EL CGI "TRADICIONAL"

Los Servlets Java son más eficientes, fáciles de usar, más poderosos, más portables, y más baratos que el CGI tradicional y otras muchas tecnologías del tipo CGI. Y lo que es más importante, los desarrolladores de servlets cobran más que los programadores de Perl.

- [Eficiencia — Con CGI tradicional, se arranca un nuevo proceso para cada solicitud HTTP. Si el programa CGI hace una operación

relativamente rápida, la sobrecarga del proceso de arrancada puede dominar el tiempo de ejecución. Con los Servlets, la máquina Virtual Java permanece arrancada, y cada petición es manejada por un hilo (thread) Java de peso ligero, no un pesado proceso del sistema operativo. De forma similar, en CGI tradicional, si hay N peticiones simultáneas para el mismo programa CGI, el código de este problema se cargará N veces en memoria. Sin embargo, con los Servlets, hay N hilos pero sólo una copia de la clase Servlet. Los Servlet también tienen más alternativas que los programas normales CGI para optimizaciones como los cachés de cálculos previos, mantener abiertas las conexiones de bases de datos, etc.

- Conveniencia —¿Por qué aprender Perl? Junto con la conveniencia de poder utilizar un lenguaje familiar, los Servlets tienen una gran infraestructura para análisis automático y decodificación de datos de formularios HTML, leer y seleccionar cabeceras HTTP, manejar cookies, seguimiento de sesiones, y muchas otras utilidades.
- Potencia — Los Servlets Java permiten fácilmente hacer muchas cosas que son difíciles o imposibles con CGI normal. Por algo, los servlets pueden hablar directamente con el servidor Web. Esto simplifica las operaciones que se necesitan para buscar imágenes y otros datos almacenados en situaciones estándar. Los Servlets también pueden compartir los datos entre ellos, haciendo las cosas útiles como almacenes de conexiones a bases de datos fáciles de implementar. También pueden mantener información de solicitud en solicitud, simplificando cosas como seguimiento de sesión y el caché de cálculos anteriores.
- Portable — Los Servlets están escritos en Java y siguen un API bien estandarizado. Consecuentemente, los servlets escritos se pueden ejecutar sin modificarse en Apache, Microsoft IIS, o WebStar. Los Servlets

están soportados directamente o mediante plug-in en la mayoría de los servidores Web.

- **Económico** — Hay un número de servidores Web gratuitos o muy económicos que son buenos para el uso "personal" o el uso en los sitios Web de bajo nivel. Sin embargo, con la excepción de Apache, que es gratuito, la mayoría de los servidores Web comerciales son relativamente caros. Una vez se tenga un servidor Web, no importa el costo del servidor, añadirle soporte para Servlets (si no viene preconfigurado para soportarlos) es gratuito o muy económico.][10]

[10] http://www.apl.jhu.edu/%7Ehall/java/Servlet-Tutorial/

FUNDAMENTOS DE JAVA SERVER PAGES

Java Server Pages (JSP) es una tecnología que permite mezclar HTML estático con HTML generado dinámicamente. El JSP permite separar la parte dinámica de las páginas Web del HTML estático. Simplemente se escribe el HTML regular de la forma normal, usando cualquier herramienta de construcción de páginas Web que se utilice normalmente. Se encierra el código de las partes dinámicas en unas etiquetas especiales, la mayoría de las cuales empiezan con "<%" y terminan con "%>". Por ejemplo, aquí está una sección de una página JSP que resulta en algo así como "Gracias por ordenar Comidas Rápidas" para una URL como http://server_ip/OrderConfirmation.jsp?title=Comidas+rapidas

```
Gracias por ordenar
<I><%= request.getParameter ("title") %></I>
```

Normalmente se dará a los archivos una extensión .jsp, y se instalaran en el mismo sitio que una página Web normal. Aunque lo que se escriba frecuentemente se parezca a un fichero HTML normal en vez de un servelt, detrás de la escena, la página JSP se convierte en un servlet normal, donde el HTML estático simplemente se imprime en el stream de salida estándar asociado con el método service del servlet. Esto normalmente sólo se hace la primera vez que se solicita la página, y los desarrolladores pueden solicitar la página ellos mismos cuando la instalan si quieren estar seguros de que el primer usuario real no tenga un retardo momentáneo cuando la página JSP sea traducida a un servlet y el servlet sea compilado y cargado. Observa

también, que muchos servidores Web permiten definir alias para que una URL que parece apuntar a un fichero HTML realmente apunte a un servelt o a una página JSP.

Además del HTML normal, hay tres tipos de construcciones JSP que embeberemos en una página: elementos de script, directivas y acciones. Los elementos de script permiten especificar código Java que se convertirá en parte del servlet resultante, las directivas que permiten controlar la estructura general del servlet, y las acciones permiten especificar componentes que deberían ser usados, y de otro modo controlar el comportamiento del motor JSP. Para simplificar los elementos de script, se tiene acceso a un número de variables predefinidas como request del fragmento de código anterior.

[Muchas páginas Web que están construidas con programas CGI son casi estáticas, con la parte dinámica limitada a muy pocas localizaciones. Pero muchas variaciones CGI, incluyendo los servlets, hacen que se genere la página completa mediante el programa, incluso aunque la mayoría de ella sea siempre lo mismo. JSP permite crear dos partes de forma separada.][11] Aquí se ve un ejemplo:

```
[<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD><TITLE>Welcome to Our Store</TITLE></HEAD>
<BODY>
<H1>Welcome to Our Store</H1>
<SMALL>Welcome,
<!-- User name is "New User" for first-time visitors -->
<% out.println(Utils.getUserNameFromCookie(request)); %>
```

[11] http://www.apl.jhu.edu/%7Ehall/java/Servlet-Tutorial/

To access your account settings, click
<A HREF="Account-Settings.html">here.</A></SMALL>
<P>
Regular HTML for all the rest of the on-line store's Web page.
</BODY></HTML>][12]

VENTAJAS DE JSP

- [**Contra Active Server Pages (ASP)** — ASP es una tecnología similar de Microsoft. Las ventajas de JSP están duplicadas. Primero, la parte dinámica está escrita en Java, no en Visual Basic, por eso es mucho más poderosa y fácil de usar. Segundo, es portable a otros sistemas operativos y servidores Web.
- **Contra los Servlets** — JSP no da nada que no se pudiera en principio hacer con un servlet. Pero es mucho más conveniente escribir (y modificar) HTML normal que tener que hacer muchas sentencias println que generen HTML. Además, separando el formato del contenido se puede poner diferentes personas en diferentes tareas: los expertos en diseño de páginas Web pueden construir el HTML, dejando espacio para que los programadores de servlets inserten el contenido dinámico.
- **Contra Server-Side Includes (SSI)** — SSI es una tecnología ampliamente soportada que incluye piezas definidas externamente dentro de una página Web estática. JSP es mejor porque permite usar servlets en vez de un programa separado para generar las partes dinámicas. Además, SSI, realmente está diseñado para inclusiones sencillas, no para programas

[12] Core Servlets and Java Server Pages, Autor Marty Hall, A Sun Microsystems Press/Prentice Hall PTR Book, Capitulo 1, Página 10

"reales" que usen formularios de datos, hagan conexiones a bases de datos, etc.

- Contra JavaScript — JavaScript puede general HTML dinámicamente en el cliente. Este una capacidad útil, pero sólo maneja situaciones donde la información dinámica está basada en el entorno del cliente. Con la excepción de las cookies, el HTTP y el envió de formularios no están disponibles con JavaScript. Y, como se ejecuta en el cliente, JavaScript no puede acceder a los recursos en el lado del servidor, como bases de datos, catálogos, información de precios, etc.][13]

ELEMENTOS DE ESCRITURA

Los elementos de escritura de JSP deja insertar código en el servlet que va a ser generado desde la pagina JSP. Hay tres formas:

1. Expresiones — <% expresión %> que es evaluada e insertada en el output de los servlets.
2. Scriptlets — <% código %> que son insertados en el método del servidor _jspService (llamado por service).
3. Declaraciones — <%! código %> que es insertado dentro del cuerpo de la clase del servlet, fuera de cualquier método existente.

Plantilla de texto

En muchos casos, un largo porcentaje de las paginas JSP solo consisten de HTML estático, conocido como plantilla de texto. En todo lo que respecta, este HTML luce igual como el normal HTML, siguiendo todas las reglas de sintaxis, creado para manejar la pagina, es simplemente enviado al cliente

[13] http://www.apl.jhu.edu/%7Ehall/java/Servlet-Tutorial/

por el servlet. No solamente hace que el HTML luzca normal, puede ser creado por cualquier herramienta que se utilice para crear páginas Web.

Existen dos excepciones de esta regla. Primero, si se quiere pasar <% en el output, se necesita colocar <\% en la plantilla de texto. Segundo, si se quiere que un comentario aparezca en la página JSP no en el documento restante, se debe utilizar:

<%-- JSP comentario --%>

Comentarios en HTML son de la forma

<!-- HTML comentario -->

son enviados normalmente al resultado en HTML.

EXPRESIONES JSP

Una expresión JSP es utilizada para insertar valores directamente en el output. Tiene la forma:

<%= Expresión Java %>

La expresión es evaluada, convertida a un string e insertada a la página. Esta evaluación es evaluada el mismo tiempo que corre (cuando la paina es requerida) y así tiene acceso a toda la información acerca de la petición. El siguiente ejemplo muestra la fecha y hora en que la página fue solicitada:

Current time: <%= new java.util.Date() %>

Variables predefinidas

Para simplificar estas expresiones, se puede utilizar un número predefinido de variables. Las más importantes son:

- Request — el HttpServletRequest asociada con el pedido; da acceso a los parámetros del pedido, el tipo de pedido (ejemplo: GET o POST) y los headers del HTML (ejemplo: cookies).
- Response — el HttpServletResponse asociada a la respuesta para el cliente.
- Session — el HttpSession objeto asociado con el pedido. Llama a esas sesiones creadas automáticamente, así que esta variable es el límite aun si no hay referencias entrantes en la sesión.
- Out — el PrintWriter utilizada para el output hacia el cliente utilizado para enviar el output al cliente. Si embargo, para hacer el útil el objeto responsable, esta es una versión protegida de PrintWriter llamada JspWriter.
- Application — Esta variable es el ServletContext y es obtenida vía getServletConfig().getContext(). Servlets y paginas JSP pueden guardar datos persistentes en el objeto ServletConext mas bien que variables instanciadas.
- PageContext — JSP introdujo una nueva clase llamada PageContext para dar un solo punto de acceso a muchos de los atributos de las páginas y para proveer un lugar conveniente para guardar información compartida.
- Page — Esta variable es un sinónimo para THIS y no es muy útil en el lenguaje de programación Java. Fue creada como un lugar contenedor para el tiempo cuando el lenguaje script puede ser otro que Java.

LOS COMPONENTES QUE MANTIENEN LA LIBRERÍA DE ETIQUETAS

En orden de utilizar las etiquetas de JSP, se necesitan definir tres componentes diferentes.

La etiqueta Handler Class

Cuando se define una etiqueta nueva, la primera tarea es definirla en la clase Java que le dice al sistema que hacer cuando vea la etiqueta. Esta etiqueta debe implementar la interfaz javax.servlet.jsp.tagext.Tag. Esto es usualmente esta acompañado de la extensión TagSupport o la clase BodyTagSupport .

La etiqueta Library Descriptor File

Cuando se haya definido la etiqueta Handler, la siguiente tarea es identificar la clase hacia el servidor y asociarla con el nombre de una etiqueta en particular de WML.

El archivo JSP

Una vez que se tenga una implementación de la etiqueta Handler y una descripción de la etiqueta Library, se esta listo para escribir un archivo JSP que hace uso del las etiquetas.

INTRODUCCIÓN AL SERVIDOR DE APLICACIONES TOMCAT DE APACHE

Tomcat, algunas veces conocido como Apache Tomcat, otras veces como Jakarta Tomcat.

Tomcat es un contenedor de Servlets con un entorno JSP. Un contenedor de Servlets es un shell de ejecución que maneja e invoca servlets por cuenta del usuario. Se puede dividir los contenedores de Servlets en:

Contenedores de Servlets independientes (Stand-alone) — Estos son una parte integral del servidor Web. Este es el caso cuando usando un servidor Web basado en Java, por ejemplo, el contenedor de servlets es parte de JavaWebServer. Este el modo por defecto usado por Tomcat. Sin embargo, la mayoría de los servidores, no están basados en Java, los que lleva los dos siguientes tipos de contenedores:

Contenedores de Servlets dentro-de-Proceso — El contenedor Servlet es una combinación de un plugin para el servidor Web y una implementación de contenedor Java. El plugin del servidor Web abre una JVM (Máquina Virtual Java) dentro del espacio de direcciones del servidor Web y permite que el contenedor Java se ejecute en él. Si una cierta petición debería ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java. Un contenedor de este tipo es adecuado para servidores multihilo de un sólo proceso y proporciona un buen rendimiento pero está limitado en escalabilidad.

Contenedores de Servlets fuera-de-proceso — El contenedor Servlet es una combinación de un plugin para el servidor Web y una implementación de contenedor Java que se ejecuta en una JVM fuera del servidor Web. El plugin del servidor Web y el JVM del contenedor Java se comunican usando algún mecanismo IPC (normalmente sockets). Si una cierta petición debería ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java. El tiempo de respuesta en este tipo de contenedores no es tan bueno como el anterior, pero obtiene mejores rendimientos en otras cosas (escalabilidad, estabilidad, etc.)

Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor Web existente (actualmente se soportan los servidores Apache, IIS y Netscape). Esto significa que siempre que se despliegue Tomcat se tendrá que decidir cómo usarlo.

VERSIONES DE TOMCAT

[Para la calidad impaciente, actual de la producción de Apache Tomcat se lista las versiones vrs. especificaciones de Servlet/JSP.

Especificaciones de Servlet/JSP	Versión Apache Tomcat
2.4/2.0	5.5.17
2.3/1.2	4.1.31
2.2/1.1	3.3.2

Aunque se puede descargar las versiones anteriores con su respectiva documentación, es aconsejable que los usuarios utilicen la última actualización de Apache Tomcat. No es una tarea difícil actualizar Apache Tomcat y se podrá encontrar el apoyo en su pagina principal http://tomcat.apache.org/; si embargo, gracias al apoyo que han brindado la comunidad, entre mas viejo sea la versión menos personas van a estar interesadas en brindar apoyo técnico.][14]

COMO INSTALAR LA VERSION BINARIA DE TOMCAT

Muy sencillo. Siguiendo los siguientes pasos:

1. Se descargar el fichero zip/tar.gz/ desde la página http://jakarta.apache.org/downloads/binindex.html.
2. Se desempaqueta el fichero en algún directorio (por ejemplo foo). Esto debería crear un subdirectorio llamado jakarta-tomcat-3.2.1. Si no es el lugar que se quiere, se mueve este directorio a la localización deseada.
3. Se cambia al directorio jakarta-tomcat-3.2.1 y se configura una nueva variable de entorno (TOMCAT_HOME) que apunte a la raíz del directorio Tomcat.
   - En Win32 se escribe: set TOMCAT_HOME=foo\jakarta-tomcat-3.2.1
   - Sobre UNIX se escribe:

   para **bash/sh**: TOMCAT_HOME=foo/jakarta-tomcat-3.2.1;
   export TOMCAT_HOME

   para **tcsh**: setenv TOMCAT_HOME foo/jakarta-tomcat-3.2.1

[14] http://tomcat.apache.org/

4. Se configura la variable de entorno JAVA_HOME para que apunte al directorio raíz de la instalación del JDK, luego se añade el intérprete Java a la variable de entorno PATH.

¡Esto es todo!, ahora se pude ejecutar Tomcat y se ejecutará como un contenedor de servlets independiente (dentro-de-proceso).

INSTALACIÓN Y CONFIGURACIÓN DEL SERVIDOR WEB APACHE TOMCAT

En este aparte mostramos únicamente una instalación sencilla de la instalación de Apache Tomcat en Linux. Si desean personalizar la instalación según sus necesidades podrían remitirse a la guía completa de Apache Tomcat publicada por O-Reilly o el sitio Web. Ingresando a http://mundogeek.net/archivos/2006/04/04/apache-y-tomcat-en-windows/ podrán encontrar la breve descripción de esta instalación para un servidor Windows.

Tomcat es un contenedor de servlets (es lo que necesitamos para ejecutar JSP y Servlets) creado por la fundación Apache dentro del proyecto Jakarta. Aunque se puede utilizar como servidor Web no está tan optimizado como el servidor Web de la misma fundación, Apache. A lo largo del capitulo se mostrara como se compilarán los paquetes a partir del código fuente, en lugar de utilizar precompilados, por lo que debería poder seguirse fácilmente utilizando cualquier distribución de Linux.

Instalación del JDK

Evidentemente lo primero que necesitamos para desarrollar en Java es instalar el kit de desarrollo (JDK) que podemos descargar desde la Web de

Sun. Utilizaremos la versión auto extraíble (Linux self-extracting file) en lugar del paquete RPM.

Introducir contraseña de root

```
cp -p jdk-1_5_0_06-linux-i586.bin /usr/local
cd /usr/local
chmod +x jdk-1_5_0_06-linux-i586.bin
./jdk-1_5_0_06-linux-i586.bin
rm jdk-1_5_0_06-linux-i586.bin
```

Esto extraerá el contenido del archivo en una nueva carpeta jdk1.5.0_06 en /usr/local. Ahora basta crear la variable de entorno para indicar dónde está instalado el JDK y añadir a la variable PATH el directorio en el que se encuentran los binarios para poder ejecutarlos desde cualquier sitio. Para ello abrimos el archivo /etc/profile con nuestro editor favorito (como root) y añadimos las siguientes líneas al final:

```
JAVA_HOME=/usr/local/jdk1.5.0_06
PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME
```

Actualizamos las variables de entorno:

```
source /etc/profile
```

Si todo ha salido bien al escribir javac -versión deberíamos obtener el número de versión del compilador de Java. En el caso de que algo halla salido mal el sistema nos informará de que no encontró ningún ejecutable con ese nombre.

COMPILAR E INSTALAR TOMCAT

Descargamos la aplicación desde su Web. En este caso tenemos que descomprimir el archivo en el directorio que queramos, ya que se trata de una aplicación Java.

cp -p apache-tomcat-5.5.16.tar.gz /usr/local/

cd /usr/local

tar xvzf apache-tomcat-5.5.16.tar.gz

rm apache-tomcat-5.5.16.tar.gz

De nuevo vamos a editar /etc/profile para añadir la variable de entorno CATALINA_HOME:

CATALINA_HOME=/usr/local/apache-tomcat-5.5.16

export CATALINA_HOME

Y actualizamos:

source /etc/profile

Por último ejecutamos el script de iniciación de Tomcat:

/usr/local/apache-tomcat-5.5.16/bin/startup.sh

Deberías ver la página de bienvenida de Tomcat introduciendo la URL http://localhost:8080 en un navegador en el mismo servidor.

ARRANCAR Y DETENER TOMCAT

Se arranca y se para Tomcat usando los scripts que hay en el directorio bin.

Para arrancar Tomcat se ejecuta:

Sobre UNIX	Sobre Win32
bin/startup.sh	bin\startup

Para parar Tomcat se ejecuta:

Sobre UNIX	Sobre Win32
bin/shutdown.sh	bin\shutdown

LA ESTRUCTURA DE LOS DIRECTORIOS DE TOMCAT

Asumiendo que se ha descomprimido la distribución binaria de Tomcat se debería tener la siguiente estructura de directorios:

Directorio	Descripción
Bin	Contiene los scripts de arrancar/parar.
Conf	Contiene varios ficheros de configuración incluyendo server.xml (el fichero de configuración principal de Tomcat) y web.xml que configura los valores por defecto para las distintas aplicaciones desplegadas en Tomcat.
Doc	Contiene varia documentación sobre Tomcat.
lib	Contiene varios ficheros jar que son utilizados por Tomcat. Sobre UNIX, cualquier fichero de este directorio se añade al classpath de Tomcat.
Logs	Aquí es donde Tomcat sitúa los ficheros de diario.
Src	Los ficheros fuentes del API Servlet. Estos son sólo los interfaces vacíos y las clases abstractas que debería implementar cualquier contenedor de servlets.
webapps	Contiene aplicaciones Web de Ejemplo.

Adicionalmente se puede, o Tomcat creará, los siguientes directorios:

Directorio	Descripción
Work	Generado automáticamente por Tomcat, este es el sitio donde Tomcat sitúa los ficheros intermedios (como las páginas JSP compiladas) durante su trabajo. Si se borra este directorio mientras se está ejecutando Tomcat no podremos ejecutar páginas JSP.
Classes	Se puede crear este directorio para añadir clases adicionales al classpath. Cualquier clase que se añada a este directorio encontrará un lugar en el classpath de Tomcat.

LOS SCRIPTS DE TOMCAT

Tomcat es un programa Java, y por lo tanto es posible ejecutarlo desde la línea de comandos, después de configurar varias variables de entorno. Sin embargo, configurar cada variable de entorno y seguir los parámetros de la línea de comandos usados por Tomcat es tedioso y propenso a errores. En su lugar, el equipo de desarrollo de Tomcat proporciona unos pocos scripts para arrancar y parar Tomcat fácilmente.

Qué son los Scripts

La siguiente tabla presenta los scripts más importantes para el usuario común:

Script	Descripción
Tomcat	El script principal. Configura el entorno apropiado, incluyendo CLASSPATH, TOMCAT_HOME y JAVA_HOME, y arranca Tomcat con los parámetros de la línea de comando apropiados.
Startup	Arrancar tomcat en segundo plano. Acceso directo para Tomcat start.
Shutdown	Para tomcat (lo apaga). Acceso directo para tomcat stop.

El script más importante para los usuarios es tomcat (tomcat.sh/tomcat.bat). Los otros scripts relacionados con tomcat sirven como un punto de entrada simplificado a una sola tarea (configuran diferentes parámetros de la línea de comandos, etc).

Una mirada más cercana a tomcat.sh/tomcat.bat nos muestra que realiza las siguientes acciones:

Unix

- Averigua donde está TOMCAT_HOME, si no se especifica.
- Averigua donde está JAVA_HOME, si no se especifica.
- Configura un CLASSPATH que contiene:
  1. el directorio ${TOMCAT_HOME}/classes (si existe).
  2. Todo el contenido de ${TOMCAT_HOME}/lib.
  3. ${JAVA_HOME}/lib/tools.jar; este fichero jar contiene la herramienta javac, que se necesita para compilar los ficheros JSP.
- Ejecuta java con los parámetros de la línea de comandos que ha configurado un entorno de sistema Java, llamado tomcat.home, con org.apache.tomcat.startup. Tomcat como la clase de arranque. También

procesa los parámetros de la línea de comandos para org.apache.tomcat.startup. Tomcat, como:

1. La operación a ejecutar start/stop/run/etc.
2. Un path al fichero server.xml usado por este proceso Tomcat.

- Por ejemplo, si server.xml está localizado en /etc/server_1.xml y el usuario quiere arrancar Tomcat en segundo plano, debería introducir la siguiente línea de comandos: bin/tomcat.sh start -f /etc/server_1.xml.

Win32

- Graba las configuraciones actuales para TOMCAT_HOME y CLASSPATH.
- Prueba JAVA_HOME para asegurarse de que está configurado.
- Prueba si TOMCAT_HOME está configurado y los valores por defecto a "." no lo están. Entonces se usa TOMCAT_HOME para probar la existencia de servlet.jar para asegurarse de que TOMCAT_HOME es válido.
- Configura la variable CLASSPATH que contiene:
  1. %TOMCAT_HOME%\classes, incluso si no existe.
  2. Los ficheros Jar de %TOMCAT_HOME%\lib. Si es posible, todos los ficheros jar en %TOMCAT_HOME%\lib son incluidos dinámicamente. Si no es posible, se incluyen estáticamente los siguientes ficheros jar: ant.jar, jasper.jar, jaxp.jar, parser.jar, servlet.jar, y webserver.jar
  3. %JAVA_HOME%\lib\tools.jar, si existe (este fichero jar contiene la herramienta javac necesaria para compilar los ficheros JSP).
- Ejecuta %JAVA_HOME%\bin\java, con los parámetros de la línea de comandos que configuran el entorno de sistema Java, llamado tomcat.home, con org.apache.tomcat.startup. Tomcat como la clase de

arranque. También le pasa los parámetros de la línea de comandos a org.apache.tomcat.startup. Tomcat, como:

1. La operación a realizar: start/stop/run/etc.
2. Un path al fichero server.xml usado por este proceso Tomcat.

Por ejemplo, si server.xml está localizado en conf\server_1.xml y el usuario quiere arrancar Tomcat en una nueva ventana, debería proporcionar la siguiente línea de comando: bin\tomcat.bat start -f conf\server_1.xml

- Restaura las configuraciones de TOMCAT_HOME y CLASSPATH grabadas previamente.

Como se puede ver, la versión Win32 de tomcat.bat no es muy robusta como la de UNIX. Especialmente, no se averigua los valores de JAVA_HOME y sólo intenta "." como averiguación de TOMCAT_HOME. Puede construir el CLASSPATH dinámicamente, pero no en todos los casos. No puede construir el CLASSPATH dinámincamente si TOMCAT_HOME contiene espacios, o sobre Win9x, si TOMCAT_HOME contiene nombres de directorios que no son 8.3 caracteres.

UTILIZAR SECURITY MANAGER DE JAVA CON TOMCAT

El SecurityManager de Java es el que permite a un navegador ejecutar un applet en su propia caja para evitar que código no firmado acceda a ficheros del sistema local, conectar con un host distinto de donde se cargó el applet, etc.

De la misma forma que el SecurityManager protege de que se ejecute un applet no firmado en el navegador, el uso de un SecurityManager mientras se

ejecuta Tomcat puede proteger el servidor de servelts, JSP's, beans JSP, y librerías de etiquetas troyanos. O incluso de errores inadvertidos.
Imagine que alguien que está autorizado a publicar un JSP en nuestro site inadvertidamente incluye esto en su JSP:

```
<% System.exit(1); %>
```

Cada vez que el JSP sea ejecutado por Tomcat, Tomcat se cerrará.

Usar el SecurityManager de Java es sólo una línea más de defensa que un administrador de sistemas puede usar para mantener el servidor seguro y fiable.

Precauciones

La implementación de un SecurityManager en Tomcat no ha sido completamente probada para asegurar la seguridad de Tomcat. No se han creado Permissions especiales para evitar accesos a clases internas de Tomcat por parte de JSPs, aplicaciones Web, beans o librerías de etiquetas. Se debe estar seguro de que se esta satisfecho con la configuración de SecurityManager antes de permitir que los usuarios no creíbles publiquen aplicaciones Web, JSPs, servlets, beans o librerías de etiquetas en el sitio.

Aún así, ejecutarlo con un SecurityManager definitivamente es mejor que hacerlo sin ninguno.

Tipos de permisos

Las clases Permission se usan para definir que clases de permisos tendrán las clases cargadas por Tomcat. Hay varias clases de Permission como parte del JDK e incluso se pude crear las propias para usarlas en las aplicaciones Web.

Este es sólo un pequeño sumario de las clases de System SecurityManager Permission aplicables a Tomcat. Se puedes encontrar más documentación sobre el uso de las clases siguientes en la documentación del JDK.

- java.util.PropertyPermission — controla los accesos de lectura / escritura a las propiedades de JVM como java.home.
- java.lang.RuntimePermission — controla el uso de algunas funciones de sistema / ejecución como exit() y exec().
- java.io.FilePermission — controla los acceso de lectura / escritura / ejecución a ficheros y directorios.
- java.net.SocketPermission — controla el uso de sockets de red.
- java.net.NetPermission — controla el uso de conexiones de red multicast.
- java.lang.reflect.ReflectPermission — controla el uso de reflection para hacer introspection de clase.
- java.security.SecurityPermission — controla el acceso a los métodos de Security.
- java.security.AllPermission — permite acceder a todos los permisos, como si se estuviera ejecutando Tomcat sin un SecurityManager.

LOS WORKERS TOMCAT

Un worker Tomcat es un ejemplar Tomcat que está esperando para ejecutar servlets por cuenta de algún servidor Web. Por ejemplo, se puede tener un servidor Web como Apache reenviando peticiones servlets a un proceso Tomcat (el worker que se ejecuta detrás de él).

El escenario descrito arriba es muy simple; de hecho se puede configurar múltiples workers para servir servlets por cuenta de un cierto servidor Web. Las razones para dicha configuración pueden ser:

- Diferentes contextos sean servidos por diferentes workers Tomcat para proporcionar un entorno de desarrollo donde todos los desarrolladores compartan el mismo servidor pero con su propio worker Tomcat.
- Diferentes host virtuales servidos por diferentes procesos Tomcat proporcionen una clara separación entre los sitios pertenecientes a distintas compañías.
- Proporcionar un balance de carga, lo que significa ejecutar múltiples workers Tomcat cada uno en su propia máquina y distribuir las peticiones entre ellos.

Los workers están definidos en un fichero de propiedades llamado workers.properties y acá se explica como trabajar con él.

Definir workers

La definición de workers para el plugin Tomcat del servidor Web puede hacerse usando un fichero de propiedades (un fichero de ejemplo llamado workers.properties está disponible en el directorio conf/); el fichero contiene entradas con la siguiente forma:
worker.list=<una lista separada por comas de nombres de workers >

Por ejemplo:
worker.list= ajp12, ajp13
Y
worker.<nombre de worker>.<property>=<valor de propiedad>

Por ejemplo:
worker.local.port=8007

Cuando se arranca, el plugin del servidor Web ejemplarizará los workers cuyos nombres aparezcan en la propiedad worker.list, estos también son los workers a los que se puede mapear peticiones.

Cada worker nombrado debería tener una pocas entradas para proporcionar información adicional sobre sí mismo; esta información incluye el tipo de worker y otra información relacionada. Actualmente existen estos tipos de workers en (Tomcat 3.2-dev):

Tipo	Descripción
Ajp12	Este worker sabe cómo reenviar peticiones a workers Tomcat fuera-de-proceso usando el protocolo **ajpv12**.
Ajp13	Este worker sabe cómo reenviar peticiones a workers Tomcat fuera-de-proceso usando el protocolo **ajpv13**.
Jni	Este worker sabe cómo reenviar peticiones a workers Tomcat fuera-de-proceso usando **jni**.
Lb	Este es un worker de balance de carga, que sabe como proporcionar un balance de carga basado en redondeo con un cierto nivel de tolerancia.

Definir workers de un cierto tipo debería hacerse siguiendo este formato de propiedad:

worker.<worker name>.type=<worker type>

Donde worker name es el nombre asignado al worker y worker type es uno de los cuatro tipos definidos en la tabla. Un nombre de worker podría no contener espacios (una buena convención de nombres sería utilizar las reglas de nombrado para las variables Java). Por ejemplo:

Definición de Worker	Significado
Worker.local.type=ajp12	Define un worker llamado "local" que usa el protocolo ajpv12 para reenviar peticiones a un proceso Tomcat.
Worker.remote.type=ajp13	Define un worker llamado "remote" que usa el protocolo ajpv13 para reenviar peticiones a un proceso Tomcat.
Worker.fast.type=jni	Define un worker llamado "fast" que usa JNI para reenviar peticiones a un proceso Tomcat.
Worker.loadbalancer.type=lb	Define un worker llamado "loadbalancer" que hace balance de carga de varios procesos Tomcat de forma transparente.

Configurar las propiedades del worker

Después de definir los workers se puede especificar las propiedades para ellos. Las propiedades se pueden especificar de la siguiente manera:
worker.<worker name>.<property>=<property value>

Cada worker tiene un conjunto de propiedades que se puede configurar según se especifica en las siguientes subsecciones:

- **Propiedades de un Worker ajp12 —** Los workers del tipo ajp12 reenvían peticiones a workers Tomcat fuera-de-proceso usando el protocolo ajpv12 sobre los sockets.
- **Propiedades de un Wroker ajp13 —** Los workers del tipo ajp13 reenvían peticiones a workers Tomcat fuera-de-proceso usando el protocolo ajpv13 sobre los sockets. Las principales diferencias entre ajpv12 y ajpv13 son que:

- ajpv13 es un protocolo más binario e intenta comprimir algunos de los datos solicitados codificando los strings más frecuentemente usados en enteros pequeños.
- ajpv13 rehúsa sockets abiertos y los deja abiertos para futuras peticiones.
- ajpv13 tiene un tratamiento especial para información SSL por eso el contenedor puede implementar métodos relacionados cono SSL como isSecure().

- **Propiedades de un Worker lb —** El worker de balanceo de cargas realmente no se comunica con otros workers Tomcat, en su lugar es el responsable de varios workers "reales". Este control incluye:
  - Ejemplarizar los workers en el servidor Web.
  - Usar el factor de balanceo de carga, realizando un balanceo de carga al redondeo de peso donde el lbfactor más alto significa una máquina más fuerte (es la que manejará más peticiones).
  - Seguimiento de las peticiones que pertenecen a la misma sesión y que se ejecutan en el mismo worker Tomcat.
  - Identificar los workers Tomcat fallidos, suspender las peticiones a estos workers y hacer que otros workers las manejen.

  El resultado general es que los workers manejados por el mismo worker lb tienen balance de cargas (basándose en su lbfactor y la sesión de usuario actual) y también tienen anti-caída por lo que si un proceso Tomcat muere, no "matará" todo el sitio.
- **Propiedades de un Worker jni —** El worker jni abre una JVM dentro del proceso del servidor Web y ejecuta Tomcat dentro de ella (es decir en-proceso). Después de esto, los mensajes pasados hacia y desde la JVM son pasados usando llamadas a métodos JNI, esto hace al worker jni más

rápido que los workers fuera-de-proceso que necesitan comunicarse con los workers Tomcat escribiendo mensajes AJP sobre sockets TCP/IP.

Propiedades de los ficheros en macros

Desde Tomcat3.2 se puede definir macros en los ficheros de propiedades. Estas macros permiten definir propiedades y posteriormente usarlas cuando se construya otras propiedades. Por ejemplo, el siguiente fragmento:

```
workers.tomcat_home=c:\jakarta-tomcat
workers.java_home=c:\jdk1.2.2
ps=\
worker.inprocess.class_path=$(workers.tomcat_home)$(ps)classes
worker.inprocess.class_path=$(workers.java_home)$(ps)lib$(ps)tools.jar
```

Terminará con los siguientes valores para las propiedades:

```
worker.inprocess.class_path:
worker.inprocess.class_path= c:\jakarta-tomcat\classes
worker.inprocess.class_path=c:\jdk1.2.2\lib\tools.jar
```

INTEGRACIÓN WML, SERVLETS Y APACHE TOMCAT

INSTALACIÓN DE LA MAQUINA VIRTUAL DE JAVA (JVM)

Para Microsoft Windows

En este procedimiento, se instalara el ejecutable para desempacar e instalar el software JDK. Como parte del JDK, esta instalación incluye el Plug-in de Java y Java Web Start, como una opción incluye Java 2 Runtime Environment.

1. **Observe el tamaño del archivo a descargar —** si ha descargado el ejecutable al disco sin correrlo desde la página Web, observe que el tamaño sea el mismo al mostrado en la página Web. Una ve que se halla terminado la descarga, observe que se ha descargado la versión completa.
2. **Si esta instalado el 5.0 Beta 1, Beta 2 o Rc, desinstálelo —** utilice el agregar y quitar programas de Microsoft Windows, desde el panel de control (inico –> configuración –> panel de control)
3. **Corra el instalador JDK —** el archivo jdk-1_5_0_<versión>-windows-i586-i.exe es el instalador JDK. Si lo ha descargado completamente desde la página y no lo ha corrido desde la página, doble clic en el icono del instalador. Después continúe las instrucciones dadas por el instalador. El instalador le preguntara que si quiere reiniciar su equipo. Cuando termine

con esta instalación, se podrá borrar el archivo descargado para recuperar espacio en el disco.

4. **Comience a utilizar JDK —** su sistema ahora deberá estar listo para utilizar JDK. En este paso, simplemente correrá algunos comandos y se asegurara de que de que funcione correctamente.
5. **Desinstalar JDK —** si algunas querrá desinstalar JDK, utilice la utilidad de agregar y quitar programas en el control panel de Microsoft Windows. Como un método alternativo, si tiene el programa de la instalación original que utilizo para instalar JDK, puede dar doble clic para abrirlo y desinstalarlo.

Para Linux

Instalando JDK automáticamente se instalara el Plug-in de Java y Java Web Start. El Java Plug-in necesitar estar registrado con el navegador.

Siga las instrucciones si quiere instalar manualmente el archivo binario de JDK.

1. **Observe el tamaño del archivo a descargar —** observe que el tamaño sea el mismo al mostrado en la pagina Web. Una ve que se halla terminado la descarga, observe que se ha descargado la versión completa.
2. **Asegúrese de los permisos—** para ejecutarse el archivo binario. Corra este comando: chmod +x jdk-1_5_0<version>-linux-i586.bin
3. **Cambie de directorio —** cambie la ubicación del directorio en donde quiera que los archivos sean instalados.

   EL próximo paso instalara JDK en el directorio actual.
4. **Corra el archivo binario —** Ejecute el archivo descargado. Si el archivo esta en el directorio actual, PREPEND con "./" (es necesario "." Si no esta en el PATH): ./jdk-1_5_0_<version>-linux-i586.bin. Los archivos JDK con

instalados en el directorio llamado jdk1.5.0_<versión> en el directorio actual.

EVOLUCIÓN A TOMCAT

(Java Servlet Developer Kit) proporciona el conjunto de herramientas necesarias para el desarrollo de servlets, la última versión disponible es la 2.2 que se encuentra disponible en el Web de Sun. La instalación del JSDK es muy sencilla. El JSDK consta básicamente de 3 partes:

1. El API del JSDK, que se encuentra diseñada como una extensión del JDK y consta de dos packages javax.servlet y javax.servlet.http
2. La documentación del API y el código fuente de las clases (similar a la de los JDK 1.1 y 1.2).
3. La aplicación servletrunner, que es una utilidad que permite probar los servlets creados sin necesidad de hacer instalaciones de servidores HTTP. Si solo queremos probar nuestros servlets para que servletrunner funcione el único requisito es que tenga acceso a los packages del Servlet API.

[Antes hemos dicho que la última versión del JSDK es la 2.2 si bien esto es cierto hay que precisar que actualmente toda la plataforma servlets desarrollada por Sun ha sido cedida junto con la especificación de JSP para ser usada en un proyecto denominado Jakarta liderado por Apache, del proyecto Jakarta el producto estrella es Tomcat (hay otros dos proyectos bajo Jakarta que son Watchdog y Taglibs que también son muy conocidos) que no es más que la particular adaptación que se ha hecho de la tecnología JSP y de los Servlets de manera que el código del mismo es abierto y cualquier persona puede formar parte del proyecto y aportar sus ideas y experiencias,

además Tomcat por su propia naturaleza también sirve como motor para la ejecución de servlets.][15]

CLASES BÁSICAS PARA LA MANIPULACIÓN DE SERVLETS

El JSDK contiene dos packages distintos: javax.servlet y javax.servlet.http. Todas las clases e interfaces que hay que utilizar en la programación de servlets están en estos dos packages.

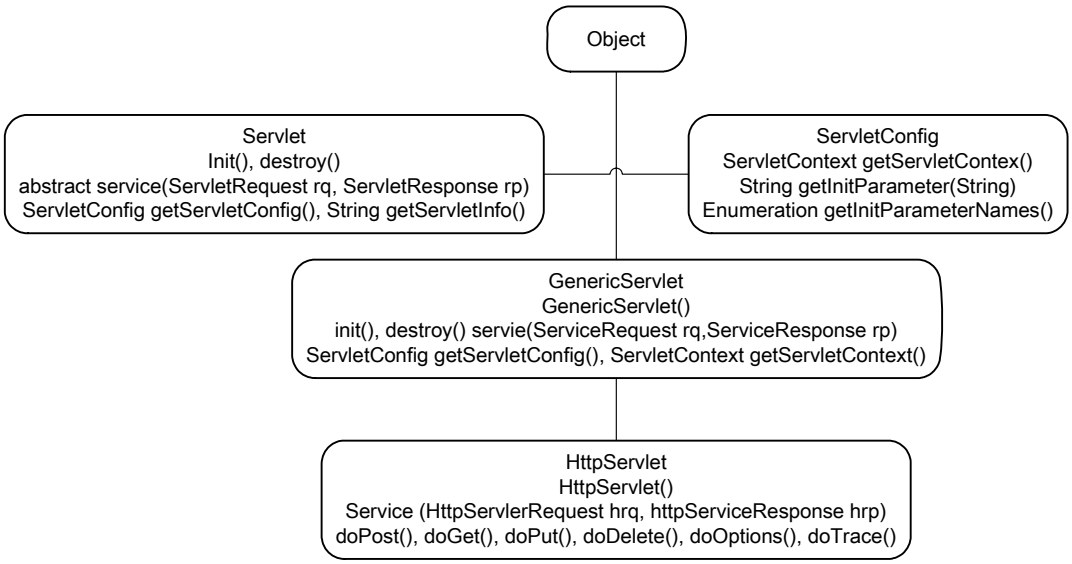


Figura 3 Clases básicas del JSDK

Si nos fijamos en la figura veremos que la clase GenericServlet es una clase abstracta que implementa dos interfaces, Servlet y ServletConfig, siendo fundamental la interface Servlet ya que en ella se declaran los métodos más importantes de cara a la vida de un servlet (recordar el ciclo de vida que veíamos antes) como son:

[15] La practica desarrollada en esta monografía no esta enfocada a IIS o Apache, esta dirigida hacia Jakarta Tomcat.

1. init(): que se ejecuta sólo al arrancar el servlet.
2. destroy() que se ejecuta cuando va a ser destruido.
3. service() que se ejecutará cada vez que el servlet deba atender una solicitud de servicio.

Siempre que derivemos una nueva clase de GenericServlet se deberá definir al menos el método service() aunque en la práctica todos los servlets deberán construirse a partir de la clase HttpServlet que es una subclase de GenericServlet y que ya no es abstract y dispone de una implementación del método service().

En esta implementación se detecta el tipo de servicio HTTP que ha sido solicitado desde el terminal y llama al método adecuado de la misma clase. En este apartado y dado que la mayor parte de los servlets que crearemos serán del tipo HttpServlet vamos a hacer un estudio más detallado de esta clase.

1. ServletContext: permite a un servlet acceder a información sobre el entorno en que se están ejecutando.
2. ServletConfig: contiene métodos que permiten pasar al servlet información sobre sus parámetros de inicialización.
3. ServletRequest: permite al método service() de GenericServlet obtener información sobre una petición de servicio recibida de un cliente.
4. ServletResponse: sirve al método service() de GenericServlet enviar la respuesta al cliente que ha solicitado el servicio
5. HttpServletRequest: deriva de ServletRequest. Sirve a los métodos de la clase HttpServlet recibir una petición de servicio HTTP.
6. HttpServletResponse: extiende ServletResponse. A través de esta interface los métodos de HttpServlet envían información a los clientes que les han pedido algún servicio.

7. Interfaces del JSDK

La clase HttpServlet

Los servlets que utilizan el protocolo HTTP son los más comunes como ya hemos dicho y sabemos que en HTTP los terminales (o browsers) y los servidores puedan comunicarse entre sí, mediante la utilización de una serie de métodos como son GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT y OPTIONS. Estos métodos son los que van dentro de la clase HttpServlet van a interesarnos implementar como ya veremos.

La clase abstracta javax.servlet.http.HttpServlet implementa la interfase javax.servlet.Servlet e incluye un número importante de funciones adicionales. La forma más sencilla de escribir un servlet HTTP es heredando de HttpServlet que es también una clase abstract, de modo que es necesario definir una clase que derive de ella y redefinir en la clase derivada al menos uno de sus métodos, tales como doGet(), doPost(), etc.

Como ya se ha comentado, la clase HttpServlet proporciona una implementación del método service () en la que distingue qué método se ha utilizado en la petición (GET, POST, etc.), llamando seguidamente al método adecuado (doGet(), doHead(), doDelete(), doOptions(), doPost() y doTrace()). Estos métodos e corresponden con los métodos HTTP anteriormente citados. Así pues, la clase HttpServlet no define el método service() como abstract, sino como protected, al igual que los métodos init(), destroy(), doGet(), doPost(), etc., de forma que ya no es necesario escribir una implementación de service() en un servlet que herede de dicha clase.

La clase HttpServlet es bastante hábil ya que es también capaz de saber qué métodos han sido redefinidos en una sub-clase, de forma que puede comunicar al cliente qué tipos de métodos soporta el servlet en cuestión. Así, si en la clase MiServlet sólo se ha redefinido el método doPost() y el cliente

realiza una petición de tipo HTTP GET el servidor lanzará automáticamente un mensaje de error similar al siguiente:

501 Method GET Not Supported

Donde el número que aparece antes del mensaje es un código empleado por los servidores HTTP para indicar su estado actual.

Para finalizar con todo esta explicación lo mejor es ver un servlet muy básico para que nos muestre su estructura, como pueden ver el esqueleto de este tipo de programas es algo tan simple como esto:

```
import javax.servlet.*;
import javax.servlet.http.*;
public class MiServlet extends HttpServlet {
public void init(ServletConfig config) throws ServletException {
super.init (config);
Resto del código de init
} // fin del método init()

public void destroy() {
Código del destroy
} // fin del método destroy()

public void doPost (HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
Código del doPost generalmente:
```

- obtención de variables
- Tratamiento de datos
- Devolver Pagina WML } // fin del método doPost()

```
public String getServletInfo() {
Método adicional de información
```

```
} // fin del método getServletInfo()
}
```

Notamos que no es necesario hacer una redefinición del método service ya que estamos heredando de la clase HttpServlet por lo que dicho método se hereda de esta y solamente hacemos el trabajo para aquellos métodos que son realmente necesarios como son init, destroy y suponiendo que las variables se pasen desde el terminal mediante el método Post la definición de la función doPost, aunque podríamos haber definido un doGet o similares en caso de necesitarlas. También es interesante ver que en la definición del método init se ha hecho una llamada al método init de su súper clase lo cual es siempre necesario a fin de garantizar la correcta inicialización del mismo, a continuación de esta instrucción podemos incluir el resto de inicializaciones que consideremos necesarias como puede ser la apertura de ficheros o de conexiones contra base de datos o cualquier otra estructura que pueda sernos de interés.

También se puede observar en que la especificación del servlet anterior hemos añadido una función denominada getServletInfo() la cual no es más que una función que podemos declarar o no y que sirve para retornar información sobre el servlet como puede ser el autor, la fecha de creación o cualquier dato que consideremos interesante. También podríamos haber incluido otra función adicional denominada getServletConfig a través de la cual podríamos pasar al servlet parámetros de configuración básicos. Estas dos últimas funciones junto con init, destroy y service son los métodos principales que forman parte de la interface javax.servlet.Servlet y son como ya hemos visto el mecanismo de comunicación entre el servidor web y nuestro servlet.

Con respecto a destroy hay que indicar que no es necesario su definición salvo en el caso que debamos de concluir aquellas tareas que hayamos levantado y que estén pendientes de finalización como puede ser el cierre de ficheros.

Por otro lado todos los métodos de clase HttpServlet que puede redefinir el programador reciben como argumentos un objeto HttpServletRequest y otro HttpServletResponse. La interface HttpServletRequest proporciona métodos para obtener información acerca de la petición del cliente, por otro lado, el objeto de la interface HttpServletResponse permite enviar desde el servlet al cliente información acerca del estado del servidor así como establecer los valores del header del mensaje saliente, en las siguientes tablas mostramos los métodos más útiles de estas dos clases, también se añaden los métodos de la clase ServletConfig del método init.

Clase ServletConfig	
GetInitParameter(String)	Devuelve un string que contiene el valor de los parámetros de inicialización del servlet o null si el parámetro no existe
GetInitParameterName()	Devuelve los nombres de los parámetros de inicialización del servlet como una enumeración de strings
GetServletContext()	Devuelve el contexto de un servlet

Clase HttpSerlvetRequest	
GetCookies()	Devuelve un array de cookies encontradas en la petición

GetDateHeader(String)	Devuelve la fecha de la petición
GetHeader(String)	Devuelve el contenido del header HTTP de la petición
GetMethod()	Devuelve el método de la petición Get, Post, Put, etc
GetRemoteUser()	Devuelve el nombre del usuario que está haciendo la petición
GetRequestedSessionId()	Devuelve el ID de sesión de la petición
GetHeaderNames()	Devuelve el nombre del header HTTP

Clase HttpServletResponse	
AddCookie(Cookie)	Sirve para añadir una nueva cookie a la respuesta
ContainsHeader(String)	Verifica si el header HTTP del mensaje de respuesta contiene un campo con el nombre especificado
SendRedirect(String)	Redirige al cliente a la URL especificada
SendError(int)	Envía un error de respuesta al cliente usando el code indicado y un mensaje por defecto
SendError(int,String)	Envía un error de respuesta al cliente usando el code indicado y un mensaje por defecto

Excepciones

Si es impórtate en un programa que el código escrito sea estable lo debemos fundamentalmente a que dentro de dicho código se haga una adecuada gestión de errores, en el caso de los servlets los errores en la ejecución de los mismos se manipulan del mismo modo que en cualquier otro programa Java a través del uso de las excepciones. Dentro del JSDK vamos a encontrar que hay dos tipo de excepciones diferentes: Por un lado tenemos la excepción javax.servlet.ServletException que se utiliza cuando se produce un fallo general en el servlet durante su ejecución. Por otro lado tenemos la excepción javax.servletUnavailableException que indica que el servlet no está disponible, la excepción UnavailableException hereda de ServletException por lo que además de los métodos que ella implementa dispone además de algunos adicionales. Igual que hicimos antes en las siguientes tablas mostramos los métodos más útiles:

Clase ServletException	
ServletException()	Constructor para crear una nueva excepción
ServletException(String)	Constructor para crear una nueva excepción con un mensaje descriptivo

Clase UnavailableException	
GetServlet()	Devuelve el servlet que no está disponible
IsPermanet()	Devuelve true si el servlet va estar permanentemente fuera de servicio, generalmente cuando se produce esta situación se requiere la intervención del administrador de

	la máquina para poder solucionar el problema
GetUnavailableSeconds()	Devuelve el tiempo que se espera que el servidor este temporalmente fuera de servicio

INTEGRACIÓN DE SERVLETS Y WML

Con lo dicho hasta ahora solamente nos queda ver un ejemplo completo en el cual toda la tecnología que acabamos de ver se integran en algo concreto, vamos a desarrollar un pequeño servlet que genera una página WML con una serie de datos que recibe como parámetros en el método init, estos datos se suponen son pasados por una página WML la cual puede ser nosotros queramos, no vamos a detenernos en esos detalles de implementación ya que lo realmente importante es ver como recoger los parámetros y generar la posterior respuesta al terminal del usuario, según esto el código podría ser algo similar a lo siguiente:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletEjemplo extends HttpServlet {
private String Param1=null;
private String Param2=null;
private String Param3=null;
private String Param4=null;
public void init(ServletConfig config) throws ServletException {
        super.init(config);
} // fin del método init()
public void destroy(){
```

```
/* No hay nada que hacer, incluimos la función a modo de ejemplo */;
} fin del método destroy()
public void doPost (HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    Param1=req.getParameter("Param1");
    Param2=req.getParameter("Param2");
    Param3=req.getParameter("Param3");
    Param4=req.getParameter("Param4");
    GenerarPaginaWML(resp);
} // fin del método doPost()
public void doGet (HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
     doPost(req,resp);
} // fin del método doPost()
public void GenerarPaginaWML (HttpServletResponse resp) {
    resp.setContentType("text/vnd.wap.wml");
    PrintWriter out = null;
    try {
    out=resp.getWriter();
    }
    catch (IOException io) {
    ;
}//Se genera el contenido de la página WML
out.println("<% Response.ContentType = "text/vnd.wap.wml" %>");
out.println(" <?xml version="1.0"?> ");
out.println("<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml"> ");
out.println("<WML>");
```

```
out.println("<card id="Ejemplo" title="Ejemplo">");
out.println("<p><strong>Valores devueltos:</strong></p>");
out.println("<p>Param1: "+Param1+"</p>");
out.println("<br><p>Param2: " + Param2 + "</p>");
out.println("<br><p>Param3: " + Param3 + "</p>");
out.println("<br><p>Param4: " + Param4 + "</p>");
out.println("</card>");
out.println("</wml>");
out.flush();
out.close();
} // fin de GenerarPaginaWML ()
public String getServletInfo() {   return "Un servlet completo para WML";
} // fin del método getServletInfo()
}
```

Este ejemplo aunque muy sencillo incluye algunas de las posibilidades más interesantes que podemos hacer mediante los servlets. Quizás lo más interesante sea fijarse por un lado en la manera de inicialización, con la llamada al método init de la superclase tal y como ya hemos comentado y la manera de generación de la página de respuesta que nuestro terminal recibirá (en la figura mostramos un ejemplo del aspecto que presentaría suponiendo que los valores de los parámetros son Uno, Dos, Tres y Cuatro respectivamente para cada parámetro), en esta página generada podemos ver que abrimos un flujo de salida a través del parámetro resp de la clase HttpServletResponse, a través de este flujo únicamente devolvemos los resultados de las variables junto con la generación de la página WML que queremos visualizar en el terminal, muy importante establecer al principio del

envío del resultado el tipo de contenido que se está generando ya que de no hacerlo así obtendríamos un mensaje de error.

Figura 4 Aspecto de la salida generada

El método en que llamamos al servlet sería similar a la llamada a cualquier otra página WML salvo por la particularidad de que ahora la llamada será atendida por un servlet y como podemos ver en la función doPost para recoger los parámetros hacemos uso del método getParameter perteneciente a la clase HttpServletRequest. La función doGet la hemos implementado como una llamada a doPost de manera que las llamadas al servlet puedan ser tanto de un tipo como de otro.

Existe un aspecto que no hemos tratado hasta ahora y que muchos de nosotros seguro que nos estaremos preguntado por ellas, ¿donde están las cookies?, ¿se pueden hacer uso de ellas desde un servlet?, la respuesta como puedes imaginar es afirmativa ya que el JSDK implementa la posibilidad de hacer uso de ellas, veamos como podemos hacerlo.

- **Crear un objeto Cookie -** La clase javax.servlet.http.Cookie tiene un constructor que presenta como argumentos un String con el nombre de la cookie y otro String con su valor, dado que la información almacenada en una cookies lo es en forma de String será preciso convertir cualquier valor a String antes de añadirlo a una cookie. El código para crear una cookie dentro de un servlet es tan simple como escribir las líneas siguientes:

  String ValorNuevaCookie = new String("IDUser0001"); if(ValorNuevaCookie!=null) Cookie miCookie=new Cookie("Valor", ValorNuevaCookie);
- **Establecer el valor de una Cookie -** La clase Cookie proporciona varios métodos para establecer los valores de una cookie y sus atributos:

  Cookie miCookie=new Cookie("Nombre", "ValorInicial"); miCookie.setValue("ValorFinal");

  Para cambiar los atributos que la cookie presenta se pueden hacer uso de las funciones siguientes:
  - public void setComment(String): Añade un comentario a la cookie, podemos verlo por ejemplo desde el emulador de Phone.
  - public void setDomain(String): Establece el patrón de dominio a quien permitir el acceso a la información contenida en la cookie. Por ejemplo .cocotero.com permite el acceso a la cookie al servidor www.cocotero.com pero no a a.b.cocotero.com
  - public void setMaxAge(int): Establece el tiempo de caducidad de la cookie siendo la unidad de tiempo los segundos. Un valor de -1 indica al terminal que borre la cookie cuando se apague mientras que un valor 0 borra la cookie de inmediato.
  - public void setPath(String): Establece la ruta de acceso del directorio de los servlets que tienen acceso a la cookie. Por defecto es aquel que originó la cookie.

- public void setVersion(int): Establece la versión de la cookie.

- **Enviar una Cookie -** Las cookies son enviadas como parte del header de la respuesta al cliente por lo que deben de incluirse dentro del objeto de la clase HttpServletResponse de cualquiera de los parámetros de las funciones que se definen dentro de la clase, por ejemplo suponiendo una llamada a doGet podríamos hacerlo del modo siguiente:

  ```
  public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
      Cookie miCookie=new Cookie("Nombre","Valor");
      resp.addCookie(miCookie);
      PrintWriter out=resp.getWriter();
  ```

  Fijaros en la llamada al método addCookie para incluirlo dentro de la salida resp y la posterior llamada a getWriter que realiza el envió propiamente dicho.

- **Recibir una Cookie** - Del mismo modo que la cookie se envía en el header de respuesta del servidor los clientes devuelven las cookies siguiendo el mismo mecanismo. Por este motivo, las cookies enviadas deberán recogerse del objeto HttpServletRequest mediante el método getCookies(), que devuelve un array de objetos Cookie tal y como se ve en el siguiente ejemplo:

  ```
  Cookie miCookie = null;
  Cookie[] arrayCookies = req.getCookies();
  miCookie = arrayCookies[0];
  ```

  El anterior ejemplo recoge la primera cookie del array de cookies si quisiéramos recoger todas las cookies recibidas habría que implementar el código anterior dentro de un bucle *for*.

- **Obtener el valor de la cookie** - Para obtener el valor de una cookie se utiliza el método getValue() de la clase Cookie.

JAWAP 1.3.1b1 de Ericsson

Bien, hasta ahora hemos visto que si queremos hacer uso de los servlets tenemos que las únicas herramientas disponibles son las que proporciona Sun, aunque en este punto no vamos a detenernos en demasiados detalles de implementación de código, sí que conviene saber que otras alternativas tenemos. Como notamos actualmente no solo contamos con las herramientas de Sun, a parte de las herramientas de otras empresas como Borland, o IBM entre otras muchas para el desarrollo Java y que contienen utilidades para el desarrollo de servlets, existe un kit de desarrollado por Ericsson en el cual se han desarrollado un conjunto de clases y utilidades que bajo el nombre de Jawap (Java Application Framework) están orientadas y pensadas para facilitar el desarrollo de programas basados en tecnología Java para plataformas móviles, la versión actual de este kit es la 1.3.1 y pueden descargarlo desde la siguiente dirección de Internet y recuerdo que para poder acceder a dicha Web deben estar registrados para poder bajaros el programa: http://www.ericsson.com/developerszone

El Jawap se basa en el uso de servlets para funcionar y si estudiamos su documentación veremos que está pensado para ser usado en un entorno donde el servidor web es Apache y las conexiones que se efectúan entre el servidor web y el servidor de aplicaciones se basan en la invocación de métodos remotos de Java (RMI o Remote Method Invocation) que comentaremos brevemente más adelante.

Una vez que bajemos el programa obtendremos un archivo ZIP jawap131b1.zip que al descomprimirlo generara en el disco duro una serie de carpetas y subcarpetas conteniendo los distintos paquetes que forman parte del kit de desarrollo. El Jawap esta divido en cinco paquetes diferentes cada uno de los cuales presenta funcionalidades distintas, son los siguientes:

1. com.ericsson.wasalab.KWWS_UPL
   (contenido en el fichero:wlrmi.jar)
2. com.ericsson.wasalab.DSSOLFDWLRQVHUYHU_UPL
   (contenido en el fichero: wlapsrmi.jar)
3. com.ericsson.wasalab.DSSOLFDWLRQVHUYHU_MDZDS
   (contenido en el fichero: wlapsrv.jar)
4. com.ericsson.wasalab.DSSOLFDWLRQVHUYHU_ZPO
   (contenido en el fichero: wlapswml.jar)
5. com.ericsson.wasalab.DSSOLFDWLRQVHUYHU_GHEXJ
   (contenido en el fichero: wlapsdbg.jar)

El primer paquete contiene la clase Wapplet la cual proporciona los mecanismos necesarios para que el programador pueda crear una clase que trabaje como una aplicación cliente. Este paquete debe instalarse en el servidor Web y debería ser visible para el motor de los servlets. El segundo paquete contiene las clases WapplicationServer, WapplicationServer_Stub (subclase de WapplicationServer) y la WapplicationServer_Skel (subclase de WapplicationServer), este paquete debe residir tanto en el lado del servidor de aplicaciones como en el lado del servidor Web.

El tercer paquete contiene la clase Wapplication que hace uso de las clases contenidas en el paquete cuatro, de manera que ambos representan un conjunto de utilidades que pueden ser accedidas por nuestros propios programas. Finalmente el paquete quinto es un paquete de debug que sirve para depurar los programas que elaboremos haciendo uso de Jawap.

Funcionamiento RMI

RMI o Remote Method Invocation (Invocación de Métodos Remotos) surge con la idea de poder comunicar aplicaciones de forma remota y de esta

manera hacer que las aplicaciones trabajen cooperando unas con otras con la finalidad de tener un sistema distribuido potente, escalable y seguro. No obstante, no hay que pensar que RMI es la única forma de comunicar aplicaciones dentro de una red, existen otros métodos de hacerlo como es CORBA o Java IDL (o incluso DCOM si nos gusta más el mundo Microsoft) sin embargo RMI forma parte del JDK y no tiene asociado ningún costo por lo que para nosotros resulta muy interesante, además como ocurre con todo (o casi) de lo que forma parte del mundo Java el que sea gratuito no significa que sea menos potente.

La idea que habita debajo de RMI es sencilla, lo que se pretende es llamar o ejecutar (realmente invocar) métodos de objetos remotos como si estuvieran ejecutándose en local de modo que resultan totalmente transparente las llamadas que se efectúan a los objetos, en RMI la principal característica que tenemos es que la invocación al objeto de remoto se realiza sin saber a priori en que servidor reside, para ello se hace uso de una técnica que se conoce como resolución de nombre (o naming) que posibilita invocar al objeto por su nombre y no por la máquina en la que se encuentre, es importante indicar que gracias a esto no es necesario que los objetos que van a ser invocados estén disponibles en tiempo de compilación bastándonos con que existan en tiempo de ejecución sino queremos tener un error.

La arquitectura de RMI es una arquitectura en capas cada una de las cuales presenta una serie de particularidades que vemos a continuación:

1. Capa de Enlace: Sirve de interfaz entre clientes y servidores, la capa de enlace del cliente se le conoce como Stub y a la del servidor como Skeleton. Las capas de enlace sirven para asegurar las llamadas remotas, en el Jawap si os fijáis en el apartado anterior veréis que teníamos las

clases WapplicationServer_Stub y the WapplicationServer_Skel que se derivaban de WapplicationServer y que nos proporcionan esta capa de enlace tanto a nivel de cliente (WapplicationServer_Stub) como a nivel de servidor (WapplicationServer_Skel).

2. Capa de Referencia: Se encarga de manipular las llamadas que sobre la capa de enlace se vayan efectuando buscando los objetos y ejecutando las llamadas que se efectúen sobre el mismo.
3. Capa de Transporte: Se encarga de mantener la conexión entre el cliente y el servidor y por tanto entre las máquinas virtuales que cada uno utilice. La tabla con los diferentes objetos remotos disponibles esta situada en esta capa y siempre que se crea un nuevo objeto remoto se añade una entrada en la misma para que puedan crearse objetos de esta clase desde la capa de referencia.

En la figura 5 se muestra un pequeño gráfico que representa la arquitectura RMI.

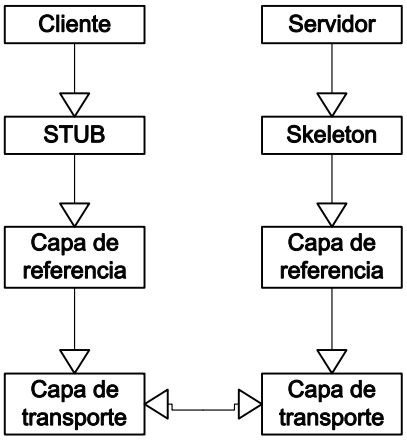


Figura5 Arquitectura RMI

Concluyendo con el Jawap podemos decir que fundamentalmente está pensado para la generación de páginas WML de manera dinámica usando la

plataforma Java y haciendo que la comunicación se produzca a través de RMI donde un servlet espera para la elaboración de los contenidos.

Generalizando teoría

Bien, comprobando todo lo que hemos contado en este capitulo las posibilidades que los servlets nos ofrecen para la programación de aplicaciones inalámbricas son muchas, lo mejor que presenta este tipo de tecnología como hemos visto es su independencia de la plataforma hardware en la cual se ejecutan por lo que la portabilidad a otros entornos diferentes del que habitualmente estemos acostumbrados a trabajar es muy sencilla, ya que al estar basados en la tecnología Java el único requisito que vamos a necesitar es tener configurada la máquina virtual correspondiente a nuestra plataforma para poder trabajar sin ningún tipo de problemas, además la posibilidad de hacer uso de objetos remotos y la existencia del kit Jawap orientado completamente al desarrollo WML hacen que nuestras aplicaciones tengan una nueva perspectiva.

WIRELESS APPLICATION PROTOCOL (WAP)

Como ya se ha comentado este protocolo es usado por los dispositivos móviles para tener acceso a Internet u otras redes asociadas. Como fue diseñado para dispositivos con poca capacidad y consumo de ancho de banda menor no se usa para manejar o mostrar grandes volúmenes de información. Cabe recordar que además de ser utilizado por celulares y PDAs, WAP también es aplicado a la navegación de Internet a través de televisores y otros visores asociados. Posee también una analogía con TCP/IP, IP y HTML en las conexiones cableadas (ejemplo: Ethernet), y su actual gama de protocolos desde la capa 3 hasta la capa 7 del modelo OSI. Dadas sus limitaciones de procesamiento de información (CPU) y memoria, WAP necesita para su funcionamiento menos sobrecarga que la utilizada por TCP/IP.

Como otros protocolos WAP han tenido su evolución a través de distintas versiones, la última de estas es la 2.0. Incluye soporte para la transmisión y recepción de sonido e imágenes en movimiento sobre teléfonos y otros dispositivos, así mismo provee una herramienta para diseño y desarrollo de nuevos servicios, esta es la llamada XHTML (Extensible Hypertext Markup Language).

La arquitectura de WAP posee las siguientes capas:

- **Capa de Aplicación —** Contiene el WAE (Wireless Application Environment) y es la interfaz directa con el usuario final. Además esta capa incluye lo siguiente:
  1. WML que es el Wireless Markup Language.
  2. Las especificaciones del micro navegador para acceso a Internet
  3. WMLScript que es el homologo de JavaScript para ambientes que utilizan HTML.

Existe otra alternativa además de WML y es el HDML o Handheld Device Markup Language. HDML contiene funciones mínimas de seguridad. Otra alternativa es C-HTML o HTML compacto. Utilizada inicialmente en Japón a través del servicio NTT DoCoMo's i-mode, C-HTML es esencialmente una versión reprimida de HTML. Gracias a esta reducción desde HTML, C-HTML puede ser desplegado o interpretado en cualquier navegador estándar tal como Internet Explorer, Mozilla Firefox, Opera, etc.

- **Capa de Sesión —** Esta capa posee el WSP o protocolo inalámbrico de sesión, con este se facilita la transferencia de contenido entre los clientes y WAP. Esta capa provee una interfaz hacia el WAE explicado en la capa de aplicación, utilizando las siguientes actividades:
  1. Conexión, creación, y finalización entre el cliente y el servidor
  2. Intercambio de datos entre el cliente y el servidor
  3. Suspensión de sesiones y eliminación entre el cliente y el servidor.

- **Capa de transacciones —** Esta capa provee una funcionalidad similar a la de TCP/IP mediante el protocolo WTP o Wireless Transactional Protocol. WTP provee servicios de transacción para WAP, incluyendo el

conocimiento de las transmisiones, retransmisiones, y eliminación de transacciones duplicadas.

- **Capa de Seguridad —** Acá era donde se quería llegar en este capitulo, esta capa de la arquitectura de WAP contiene la capa de seguridad para el transporte inalámbrico o WTLS y son siglas en ingles de Wireless Transport Layer Security. WTSL esta basado en TLS y puede ser invocado similarmente a través del navegador con la palabra HTTPS. WTLS soporta privacidad, integridad de datos, protección contra negación de servicios o Denial of Services (DoS) y autenticación. Provee tres tipos de autenticación:
  1. Clase 1 (autenticación anónima): el cliente ingresa en el servidor, pero en este modo, ni el cliente ni el servidor puede corroborar la identidad de uno o de otro.
  2. Clase 2 (autenticación de servidor): el servidor es autenticado en el cliente, pero el cliente no se autentica contra el servidor
  3. Clase 3 (dos vías, autenticación del cliente y del servidor): El servidor es autenticado en el cliente y el cliente es autenticado en el servidor.

Autenticación y autorización pueden ser configurados en el dispositivo móvil utilizando tarjetas inteligentes para ejecutar transacciones de llaves públicas. Un punto específico de seguridad asociado con WAP es el WAP GAP. WAP GAP resulta del requerimiento de cambiar protocolos de seguridad en el Gateway WAP hacia otras redes desde WTLS a protocolos protegidos con la capa de socket seguro o SSL utilizado comúnmente en redes cableadas. Hasta el Gateway o frente de la red WAP la transmisión es protegida por WTLS, acá es descifrada y luego re-encriptada para la transmisión utilizando SSL. Como se ve la información es temporalmente

convertida a texto plano en el enrutador y puede verse comprometida si el Gateway no esta adecuadamente protegido. Apuntando a este punto el WAP Forum, iniciadores de este modo de transmisión, colocaron o impusieron algunas especificaciones que podrían reducir esta vulnerabilidad y responder a las críticas en cuanto se refiere a aplicaciones de comercio electrónico. Dentro de esas especificaciones se incluye el soporte de WMLScript Crypto Library y WIM o Wireless Identity Module. WMLScript Crypto Library provee de punto a punto en la transmisión cliente servidor funciones criptográficas para iniciar en el cliente WAP una transmisión segura hacia el servidor, ósea que cuando el paquete llega al Gateway y este desencripta WTLS no quedara el texto plano sino el criptograma generado por alguna función de la Crypto Library de WMLScript. Estas funciones incluyen firmas digitales originadas desde el cliente y ciframiento y desciframiento de datos. El WIM o Wireless Identity Module es un dispositivo de fuerza-resistencia tal como una tarjeta inteligente, que coopera con WTLS y provee operaciones criptográficas durante la fase de flujo de información. Una tercera alternativa es usar un servidor Proxy para clientes que comunica la información de autorización y autenticación al servidor de red Wireless.

- Capa de Transporte — Esta capa soporta el WDP (Wireless Datagram Protocol), que provee una interfase con las redes inalámbricas. Soporta protocolos de red tales como GSM, CDMA, y TDMA. Además se encarga de la corrección de errores.

La infraestructura de llave pública PKI para aplicaciones móviles se utiliza para el cifrado de las comunicaciones y la mutua autenticación del usuario y el proveedor de aplicaciones. Algo asociado con PKI se refiere al posible tiempo antes de la expiración del certificado de llave pública y la creación de

un nuevo certificado y su asociación con la llave publica. Este tiempo muerto puede ser critico en inconvenientes de transmisión o situaciones que necesitan tiempos relativamente cortos, una solución a teste problema es generar en tiempo de ejecución llaves para usarse en ese tipo de transacciones especiales.

Cabe recordar que todas estas especificaciones de seguridad están registradas en el estándar de la IEEE 802.11 partiendo de la corrección y desarrollos de las deficiencias presentadas en WEP.

SEGURIDAD DE TOMCAT

Existen dos configuraciones posibles: la más sencilla consiste en usar la herramienta keytool, distribuida con el kit de desarrollo de Java, para generar un certificado autofirmado del servidor. Es necesario cambiar la configuración del fichero server.xml para activar un conector que permitirá acceder a Tomcat a través de https. Esta configuración tiene una limitación importante: no es posible aprovechar el mecanismo de autoridades de certificación X509.

La segunda alternativa es un poco más compleja, porque requiere crear una autoridad de certificación (CA) con la cual se pueden generar certificados para Tomcat y para los clientes que acceden a Tomcat. Como la herramienta keytool no permite realizar las operaciones necesarias, se requiere otra herramienta para crear la autoridad de certificación y firmar solicitudes de certificado.

Configuración

Tanto si se emplea un certificado autofirmado como si se crea una autoridad de certificación, los cambios en el fichero server.xml de Tomcat son

los mismos. Hay que buscar el siguiente elemento Connector que aparece comentado:

```
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
                port="8443" minProcessors="5" maxProcessors="75"
                enableLookups="true"
                acceptCount="10" debug="0" scheme="https" secure="true"
                useURIValidationHack="false">
<Factory
   className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
      clientAuth="false" protocol="TLS" />
</Connector>
```

Para activar el uso de SSL, hay que dejar sin comentarios el elemento Connector. Los aspectos más destacables de la configuración que se activa son:

- Se habilita el acceso al servidor de forma segura a través del puerto 8443. Por tanto, las URL que se tienen que emplear son de la forma: https://servidor:8443/aplicacionweb/recurso.
- No se solicita certificado al cliente durante el establecimiento de la conexión SSL. Por tanto, no es necesario cambiar nada en el navegador web que se use.

Generación del certificado autofirmado para el servidor

Para completar la instalación sencilla de Tomcat en modo seguro es necesario generar un certificado autofirmado y almacenarlo en un repositorio de claves (keystore) al alcance de Tomcat. Los certificados se almacenan en los repositorios asociándoles un alias. En el caso de Tomcat, la configuración

por defecto busca el certificado con alias tomcat. Los keystores se protegen con una clave. Por defecto, dicha clave es changeit.

El uso de la herramienta keytool para generar un keystore con un certificado autofirmado es:

%JAVA_HOME%\bin\keytool –genkey –alias tomcat –keyalg RSA

Al introducir los datos relativos al certificado, es importante emplear el nombre del servidor como CN (Common Name). Otros datos solicitados son: unidad organizativa, organización, provincia, ciudad y país.

La ejecución del comando anterior genera un fichero llamado .keystore. En los sistemas Windows, el fichero va a parar a un directorio diferente, de acuerdo con la versión:

C:\Documents and Settings\user en sistemas Windows XP

C:\Winnt\Profiles\user en sistemas multi-usuario Windows NT

C:\Windows\Profiles\user en sistemas multi-usuario Windows 95/98

C:\Windows en sistemas de un solo usuario Windows 95/98

En adelante, se considera que tanto la herramienta keystore como el propio Tomcat emplean el fichero .keystore correspondiente al usuario que está ejecutando dichos programas.

Generación de una autoridad de certificación

Se considera que no se han realizado las operaciones referidas en el apartado anterior.

El primer paso para establecer la segunda configuración de Tomcat en modo seguro es la generación de un certificado autofirmado de una autoridad de certificación.

En la distribución openSSL existe un directorio apps que contiene una herramienta escrita en Perl, llamada CA.pl. Esta herramienta funciona basándose en la configuración del fichero openssl.cnf que se encuentra en el mismo directorio. Para su correcto funcionamiento, debe estar definida la variable de entorno OPENSSL_CONF cuyo valor debe ser la localización del fichero openssl.cnf. Únicamente puede resultar conveniente modificar una línea del fichero de configuración de OpenSSL, que es la siguiente:

```
# and for everything including object signing:
nsCertType = client, email, objsign
```

nsCertType aparece, por defecto, comentado con un signo #. Al descomentarlo se permite que los certificados firmados por la CA puedan ser empleados para clientes SSL, para correo SMIME y para firma de código.

El primer paso para crear la autoridad de certificación es ejecutar la herramienta con el siguiente parámetro:

```
perl CA.pl –newca
```

El comando anterior genera un subdirectorio dentro de apps, llamado demoCA. Los ficheros y carpetas más importantes de demoCA son:

- cacert.pem: contiene el certificado autofirmado de la CA
- index.txt: contiene un resumen de los certificados que se han firmado con esta CA
- serial: contiene el número de serie que se asignará al siguiente certificado firmado con esta CA
- carpeta private: contiene el fichero cakey.pem, que es la clave privada de la CA, protegida por contraseña

- carpeta newcerts: contiene ficheros XX.pem con los certificados firmados por la CA (XX = número de serie)

Una vez generado el certificado de la CA, es necesario añadirlo a un almacén de certificados de autoridades de certificación del JRE que use Tomcat, cuya localización es: %JAVA_HOME%\jre\lib\security\cacerts. Para ello se emplea keytool de la siguiente forma:

%JAVA_HOME%\bin\keytool –import -trustcacerts –alias nombreCA– file cacert.pem
–keystore %JAVA_HOME%\jre\lib\security\cacerts

Generación de un certificado para Tomcat

Para generar un certificado firmado por la CA para Tomcat es preciso dar los siguientes pasos:

- Generar el par de claves para Tomcat, usando keytool. Es importante recordar que el CN de Tomcat debe ser el nombre DNS del servidor en el que se ejecute.

  %JAVA_HOME%\bin\keytool –genkey –alias tomcat –keyalg RSA
- Generar una solicitud de certificado partiendo del par de claves generado en el paso previo:

  %JAVA_HOME%\bin\keytool –certreq –alias tomcat –file newreq.pem
- Copiar el fichero newreq.pem al directorio apps de OpenSSL. Firmar la solicitud de certificado con la CA:

  perl CA.pl –sign

El comando anterior genera el fichero newcert.pem en el directorio apps de OpenSSL. Es necesario editar dicho fichero y eliminar las líneas iniciales hasta la línea que contiene BEGÍN CERTIFICATE.

Certificate:

Data:

Version: 3 (0x2)

Serial Number: 4 (0x4)

[...]

Eliminar hasta aquí

-----BEGIN CERTIFICATE-----

MIIDwzCCAyygAwIBAgIBBDANBgkqhkiG9w0BAQQFADCBjTELMAkGA1UE
BhMCRVMx
DzANBgNVBAgTBk11cmNpYTEPMA0GA1UEBxMGTXVyY2lhMRQwEgYDV
QQKEwtVbml2
4zTJKZXoVQu4HYMtylrJLtVv5f2N3bP5sv63tIIYiSqUiIHDPQENbOSc2C7grp
0O rBJfiRuJMw==

-----END CERTIFICATE-----

El bloque desde BEGIN hasta END se puede guardar en un nuevo fichero llamado tomcat.pem. El último paso de este bloque de operaciones es instalar el certificado firmado por la CA en el keystore:

%JAVA_HOME%\bin\keytool –import –alias tomcat –trustcacerts –file tomcat.pem

Con los pasos dados hasta ahora se puede probar el funcionamiento de Tomcat en modo seguro. El resultado debe ser que un navegador como Internet Explorer nos avisa de que el servidor Tomcat presenta un certificado firmado por una CA desconocida. La mayoría de los navegadores permiten, aún así, continuar la conexión y ver las páginas solicitadas. Para que el navegador reconozca la CA y, por tanto, pueda verificar la validez del certificado de Tomcat, es necesario instalar el certificado de la CA en el

propio navegador. En el caso de Internet Explorer, esto se puede conseguir fácilmente accediendo a las Opciones de Internet en el menú Herramientas, y seleccionando la solapa Contenido.

En la parte central del cuadro de diálogo aparece un botón “Certificados”, con el cual se puede ver los certificados instalados en el navegador. Para instalar la CA se debe seleccionar el grupo de certificados denominado “Entidades emisoras raíz de confianza”, y pulsar el botón Importar. Se selecciona el fichero cacert.pem.

Certificado de cliente

Para activar la solicitud de certificado al navegador, es necesario realizar un pequeño cambio en el fichero server.xml. El atributo clientAuth del elemento Factory que aparece dentro del Connector activado en el apartado “Configuración de Tomcat” debe tomar el valor “true”:

```
<Connector>
...
<Factory
   className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
   clientAuth="true" protocol="TLS" />
</Connector>
```

Este cambio obliga a detener y reiniciar Tomcat. Por otra parte, es necesario generar un certificado para el cliente, que esté firmado por la misma CA que firmó el certificado del Tomcat. Para ello, se deben seguir los siguientes pasos:

- Generar una solicitud de certificado (la configuración por defecto está en el fichero openssl.cnf):

perl CA.pl –newreq

- Firmar la solicitud de certificado:

  perl CA.pl –sign

- Exportar el fichero con el certificado y la clave privada a un fichero con formato PKCS12 que puede ser importado por un navegador, como Internet Explorer. El último argumento permite especificar el nombre con el que se visualizará el certificado en la lista de certificados instalados en el del navegador.

  perl CA.pl –pkcs12 “Nombre para el certificado”

- Al generar el fichero pkcs12 se solicita un password para proteger la clave privada que contiene. Ese password será solicitado por el navegador al importar el certificado, siguiendo el procedimiento de importación descrito para el certificado de la CA, con la salvedad de que, en este caso, se selecciona el grupo de certificados llamado “Personal”.

Los pasos anteriores permiten que el navegador pueda conectarse con Tomcat a través de la conexión segura. Sin un certificado de cliente instalado en el navegador, sería imposible establecer la conexión.

Consulta de certificados de cliente desde servlets

Es posible acceder a la información del certificado del cliente en un servlet instalado en un servidor Tomcat que esté configurado para solicitarlo.

El atributo javax.servlet.request.X509Certificate permite recuperar un array de certificados presentados por el cliente. Habitualmente sólo será uno.

GLOSARIO

802.11

Comité del IEEE responsable de la implantación de estándares LAN sin cable.

Capacidad

La velocidad bruta de datos en un enlace de comunicaciones, medida en bits por segundo.

CMDA (Code Division Multiple Access)

Método de compartir frecuencia entre muchos usuarios encriptando la señal de cada usuario utilizando un código diferente.

DLS (Digital subscriber Line)

Familia de tecnología la cual transmite datos en frecuencias que no son utilizadas por las líneas telefónicas de cobre.

Ethernet

Sistema Lan de estándar industrial que opera a 10, 100, 1000 y 100000 Mbps.

GPRS (General Packet Radio Service)

Técnica de modulación de fase que usa un bit por símbolo de onda, utilizada en GSM.

GSM (Global System for Mobile Communications)

Estándar TDMA de banda ancha originalmente desarrollado en Europa, pero utilizado por todo el mundo.

HTML (Hypertext Markup Language)

Código de programación utilizado para describir páginas Web, interpretado por un navegador.

IEEE (Institute of Electrical Engineers)

Sociedad profesional responsable de muchos estándares de redes, principalmente Ethernet y su derivada sin cable 802.11.

IP (Internet Protocol)

Protocolo que dirige el modo en que los paquetes de datos se conducen a través de Internet.

Java

Lenguaje de programación desarrollado por Sun Microsystems y normalmente interpretado por un navegador Web.

MMM (Mobile Media Mode)

Marca comercial que aparece en hardware y servicios compatibles con WAP.

PDA (Personal Digital Assistant)

Computador de tamaño de bolsillo que se utiliza para almacenamiento de información, pero incorpora cada vez más aplicaciones de comunicaciones sin cable.

Protocolo

Conjunto de reglas que controlan el formato y la transmisión de datos.

Roaming

Movimiento de un terminal móvil fuera de su celda de origen, especialmente a otra red en otro país.

SIM (Subscriber Identity Module)

Tarjeta inteligente integrada en los teléfonos digitales o aparatos sin cable, que contiene toda la información particular del suscriptor móvil, incluyendo el número de teléfono, operador de red, y la agenda de teléfonos del usuario.

SMS (Short Message Device)

Función disponible en algunos teléfonos móviles que permite que los usuarios envíen mensajes alfanuméricos cortos.

SSL (Secure Sockets Layer)

Protocolo de encriptación diseñado para transacciones seguras por Internet.

TCP/IP (Transport Control Protocol)

Un protocolo de conexión orientada utilizando un Internet fijo con el fin de verificar que los datos enviados han sido recibidos.

TDMA (Time Division Multiple Access)

Método de compartir una frecuencia entre varios usuarios dividiéndola en fracciones de tiempo separadas; a menudo utilizado APRA referirse al sistema D-AMPS.

USSD (Unstructured Suplementary Services Data)

Protocolo APRA mensajes alfanuméricos bidireccionales a través de una red móvil digital posibilitando así que los usuarios interactúen en tiempo real con un servidor remoto como una pasarela WAP.

WAP (Wireless Aplication Protocol)

Conjunto de protocolos que están diseñados para enviar páginas Web reducidas a aparatos sin cable. Reemplaza a los protocolos con el suyo propio, y requiere que las páginas sean escritas en WML.

WML (Wireless Markup Language)

Lenguaje de programación utilizado para especificar páginas Web optimizadas para equipos sin cable pequeños, como los teléfonos móviles.

WML Card

Bloque de navegación wml, dentro de una baraja WML. Dentro de una baraja WML debe existir una o más cartas. Cuando el usuario accede a una baraja WML, se le presentará en la pantalla únicamente la primera de sus cartas.

WML deck

Conjunto de cartas WML. Cuando se requiere un URL se carga automáticamente todo el conjunto de cartas, la carta que se especifique en el URL será a la que se acceda. Si no se especifica ninguna en particular, se accederá a la primera.

WMLScript

Lenguaje de programación con base en JavaScript, aunque menos potente, es dirigida a los dispositivos WAP.

WSP (Wireless Session Protocol)

Hace referencia a la aplicación de más alto nivel que ofrece WAP a través de un interfase para servicios de dos sesiones. La primera consistiría en un servicio con conexión que operaría sobre el protocolo del nivel de transacción y el segundo sería sin conexión y operaría sobre el servicio de transporte de información.

WTA (Wireless Telephony Applications)

Un entorno para aplicaciones de telefonía que permite a los operadores la integración de funciones de telefonía del propio dispositivo móvil con el micronavegador incorporado.

WTAI (Wireless Telephony Applications Interface)

Es una interfaz utilizada en los terminales móviles para operaciones locales de control de llamadas (recepción, iniciación y terminación) y de acceso a listines telefónicos.

RECOMENDACIONES

Durante el desarrollo del prototipo se presentaron una serie de inconvenientes en cada una de las capas de la aplicación, a continuación nombraremos cada una con la solución respectiva en su momento.

Una vez terminada la práctica notamos que de acuerdo al proveedor de Internet GPRS varía la estructura del contenido de las páginas jsp y wml, es decir hay que programar de manera diferente las páginas dependiendo de la tecnología del ISP.

Como herramienta de desarrollo se utilizaron los emuladores para tener un a idea de cómo se podría ver en los dispositivos móviles; pero al momento en que se realizaron las pruebas en los dispositivos hubo una falla al mostrar la páginas, este error se presentó porque no todos los dispositivos poseen la misma tecnología y por tal razón no soportan el mismo contenido de las páginas y hace que todas lo muestren de manera diferente. La recomendación para evitar este inconveniente es probar las páginas diseñadas en diferentes dispositivos y estos a su vez que posean la última tecnología del momento.

CONCLUSIONES

Si se esta buscando una forma muy conveniente de crear aplicaciones Web que se conecte al servidor a través de los componentes de Java y Java Server Pages (JSP). Debido a la posibilidad inherente de Java y JSP estén listo a acceder a las tecnologías como RMI, JDBC y JavaBeans, la separación de la presentación del código HTML y la presentación hace que JSP sea muy fácil para organizaciones que trabajan con ello.

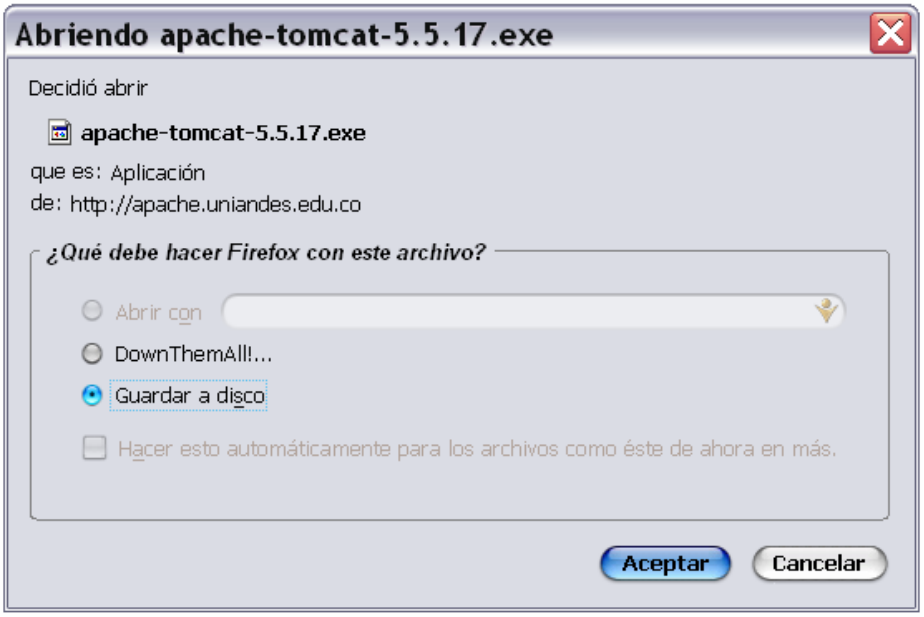
Mediante esta guía nos damos cuenta que podemos aprender de la mejor manera el diseño e implementación de un prototipo de software para la consulta de una base de datos desde un dispositivo móvil, mediante WAP generado dinámicamente con java en el servidor.

Esta guía muestra en su procedimiento e ilustración que la aplicación puede ser desarrollada y mostrada en un solo computador, es decir en el localhost. Esto no quiere decir que la aplicación no fue montada y mostrada a través de Internet, solamente es necesario que el dispositivo móvil tenga acceso a Internet y que el servidor Web sea visible (dirección IP válida).

ANEXO 1: INSTALCION DE VERSIÓN 5.0 DE TOMCAT

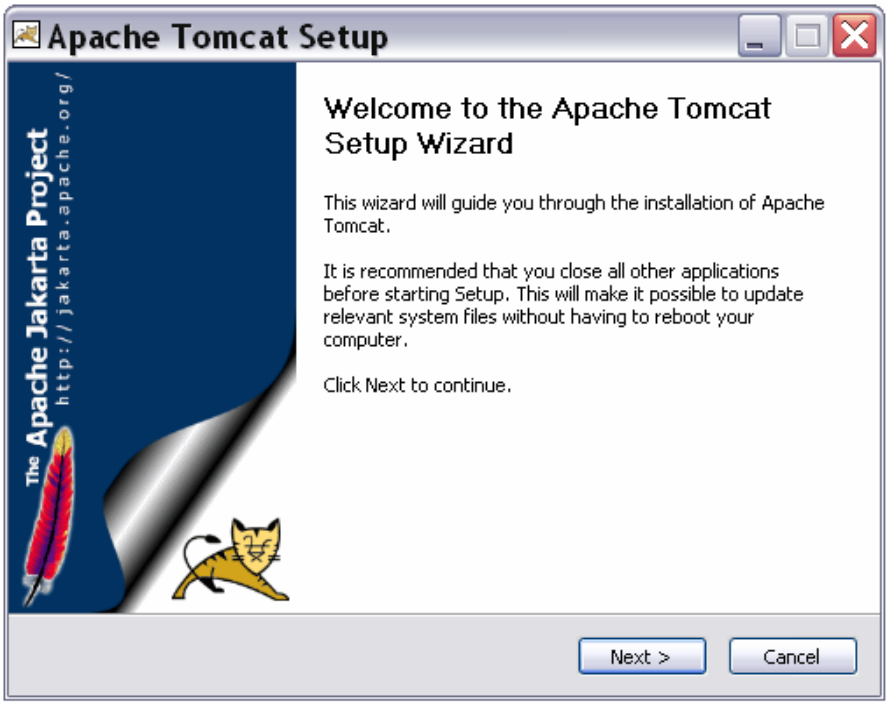
Los servidores de aplicación están en constante evolución, por esta razón la versión que se utilizó para realizar esta práctica fue 5.5.17. para descargar esta versión se puede hacer desde la página oficial de Tomcat: http://tomcat.apache.org/download-55.cgi

1. Se descarga el archivo ejecutable.

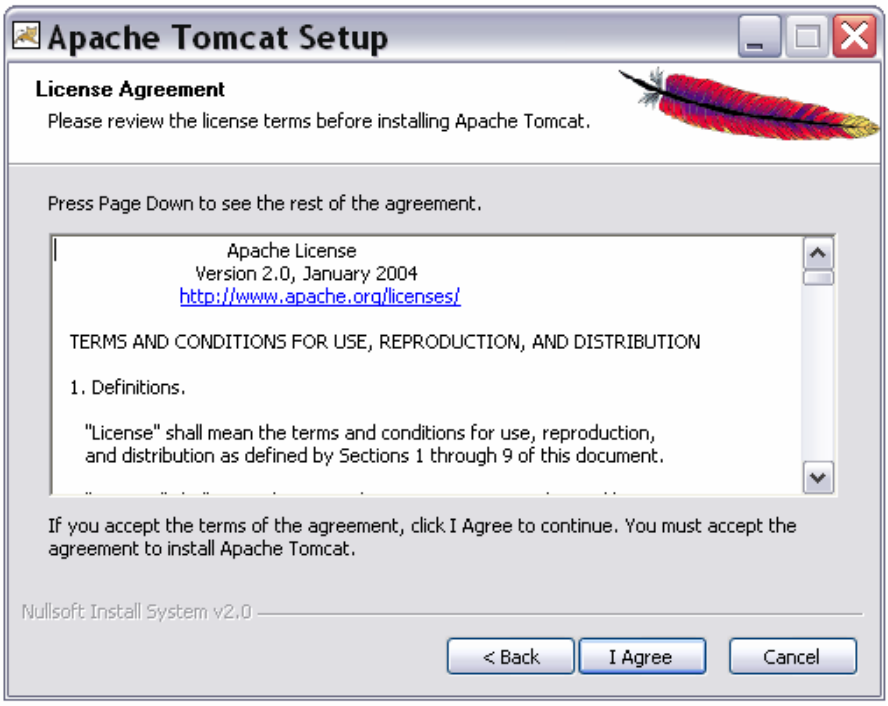


2. Después de que se halla descargado, se procede a ejecutar el archivo.

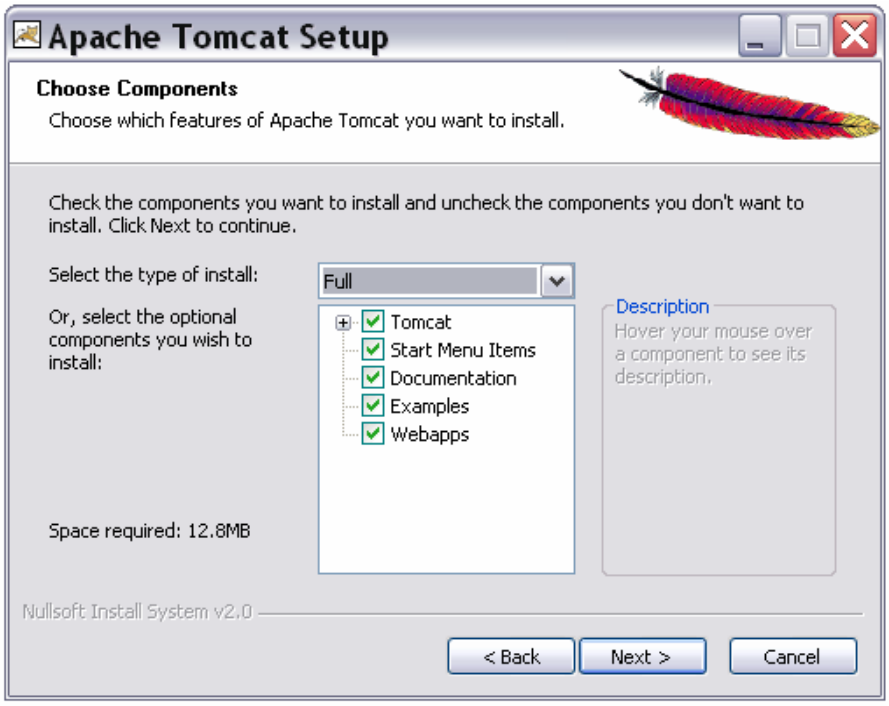
3. Comienza la configuración de Tomcat 5.



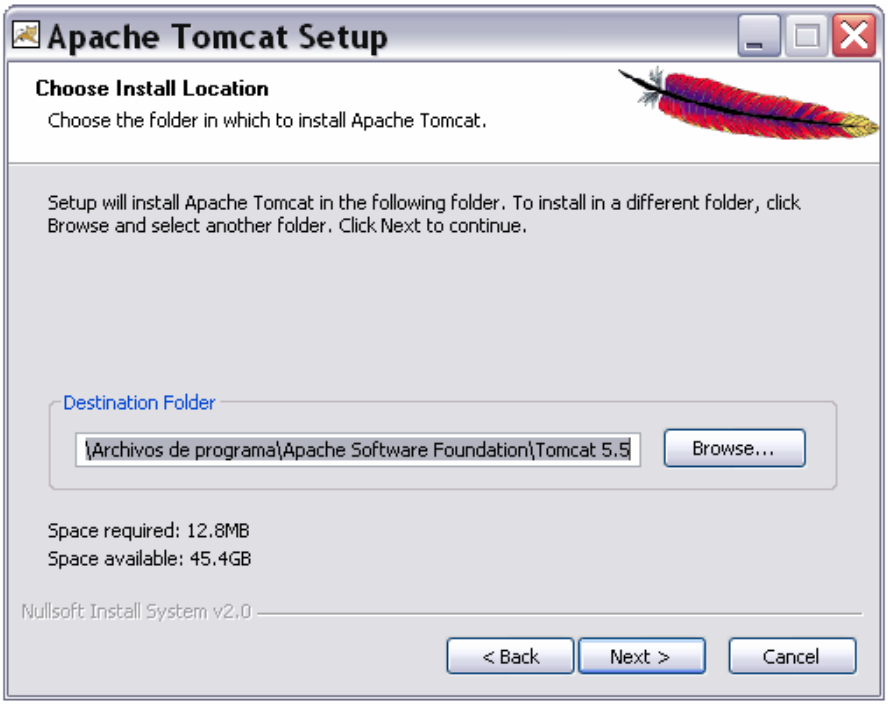
4. Se acepta los términos y las condiciones de uso.



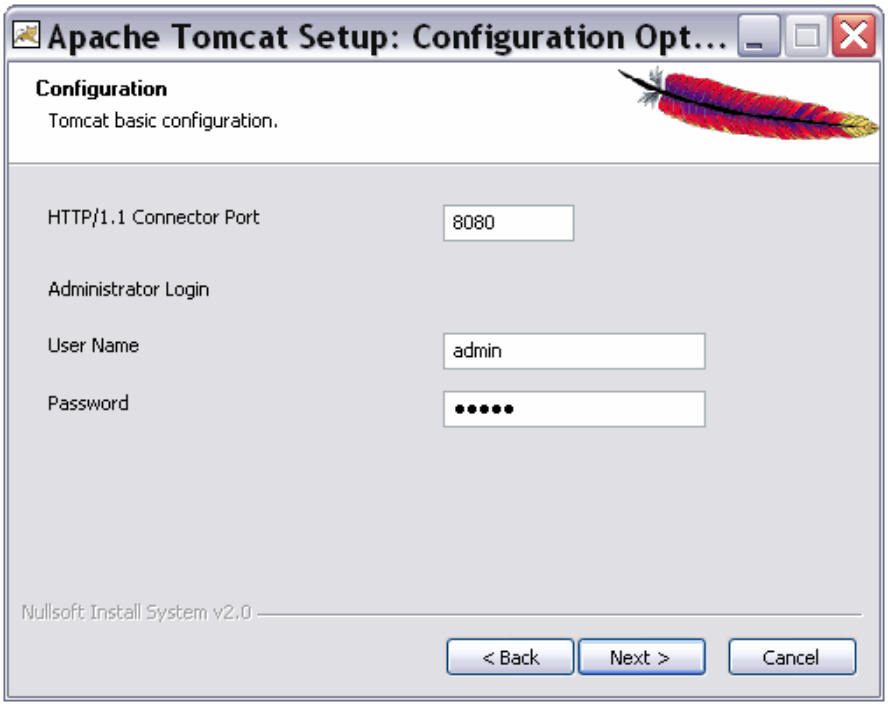
5. Se seleccionan los componentes que se requieran instalar.



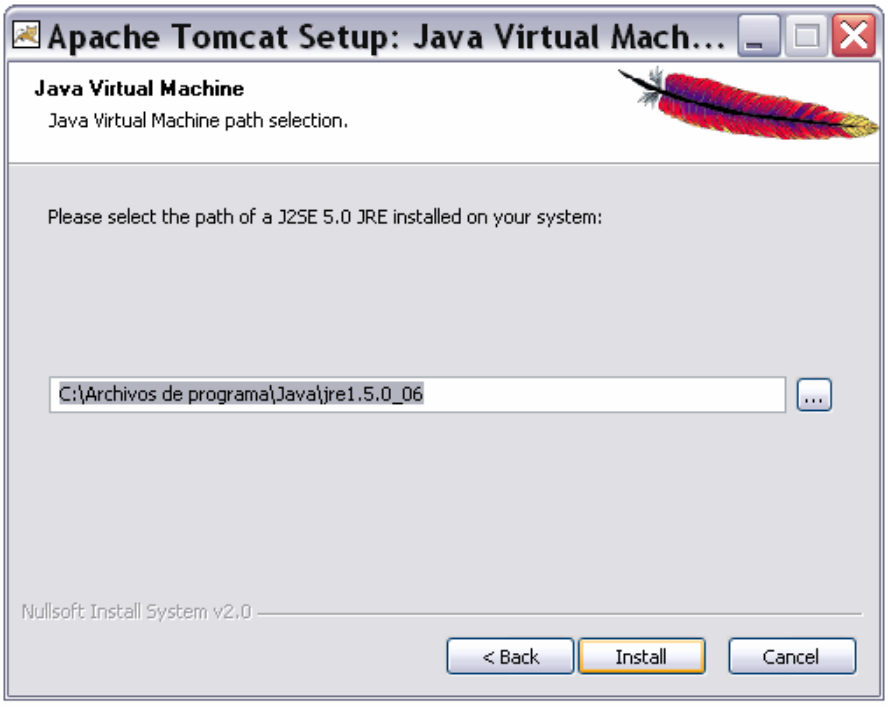
6. Se selecciona el directorio en donde se desea instalar Tomcat.



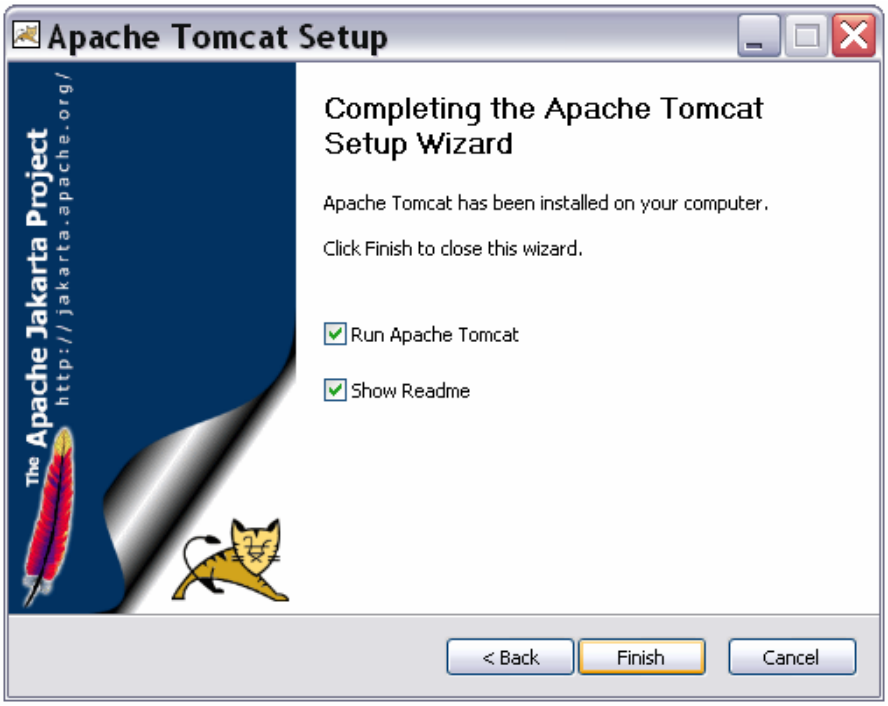
7. Se selecciona el usuario y la contraseña del administrador.



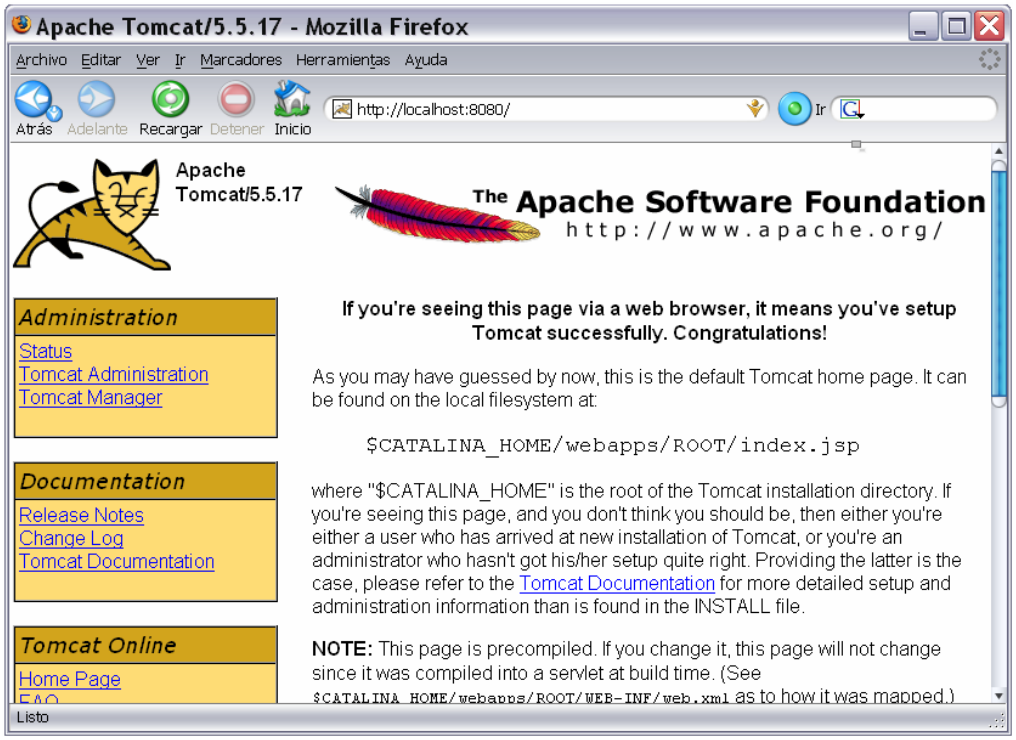
8. Se selecciona el directorio en donde está instalada la Máquina Virtual de Java (JVM).



9. Se finaliza la instalación y se inicia el servicio.



10. Se comprueba que se ha instalado correctamente.



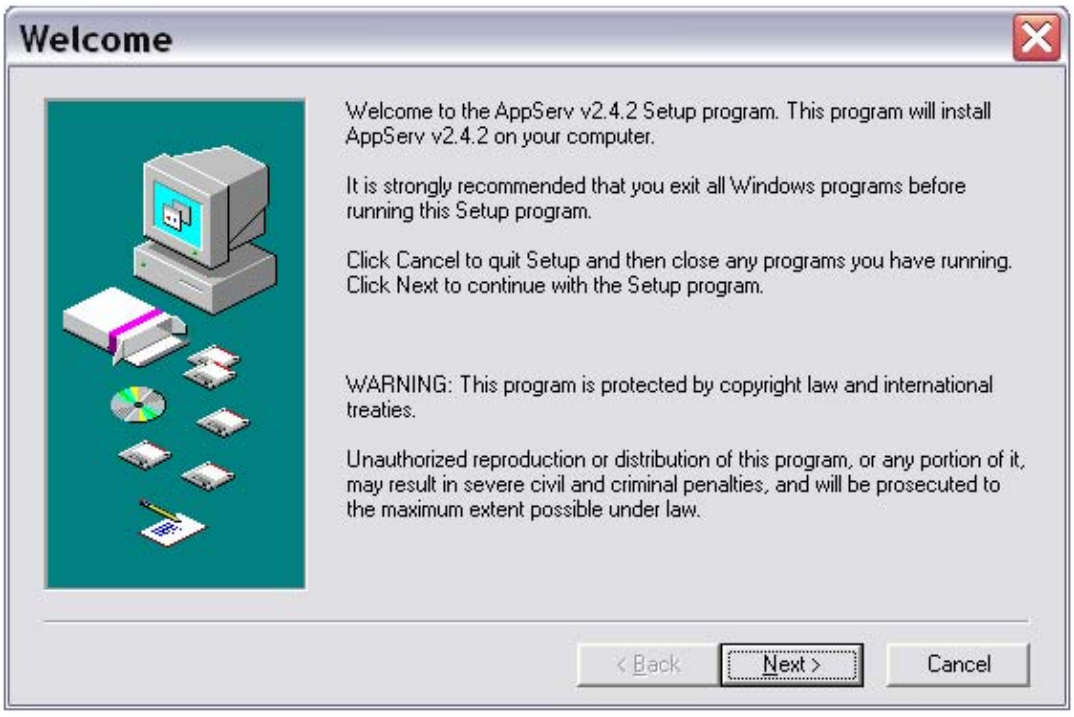
La carpeta del root está ubicada en la siguiente dirección: C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps\ROOT; y es en esta dirección en donde van a estar ubicadas las páginas con las que se va a ver las páginas wml, jsp e incluso HTML.

ANEXO 2: INSTALACIÓN DE LA BASE DE DATOS MYSQL

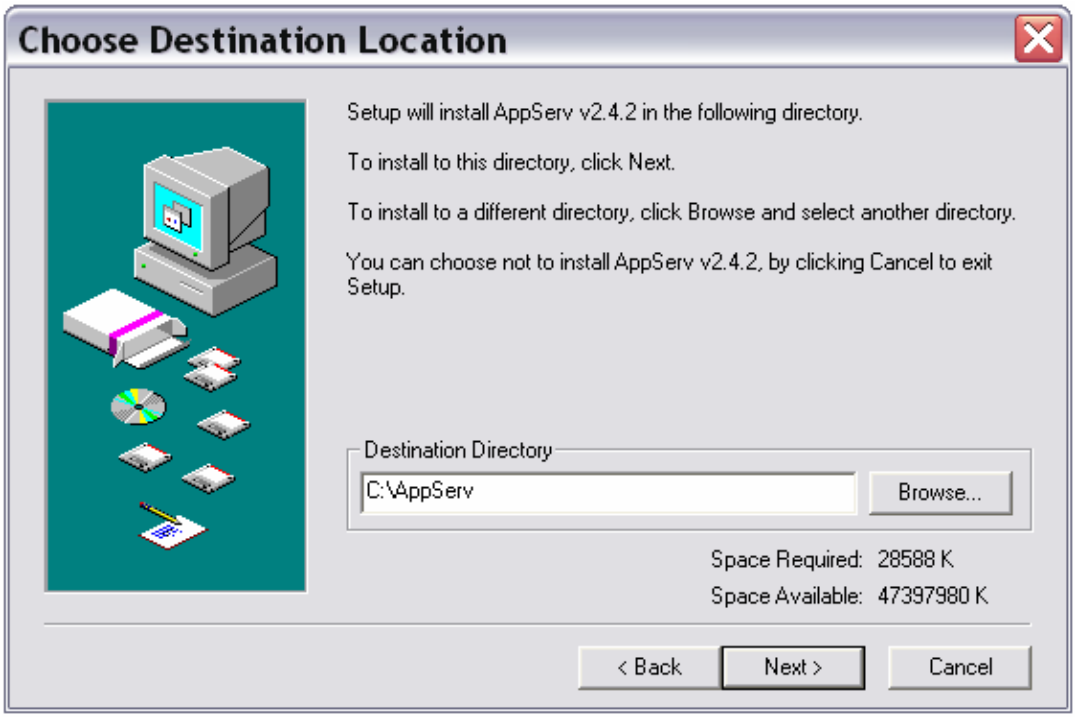
La versión que se utilizó para realizar esta práctica fue AppServ 2.4.2.

1. Se ejecuta el archivo.

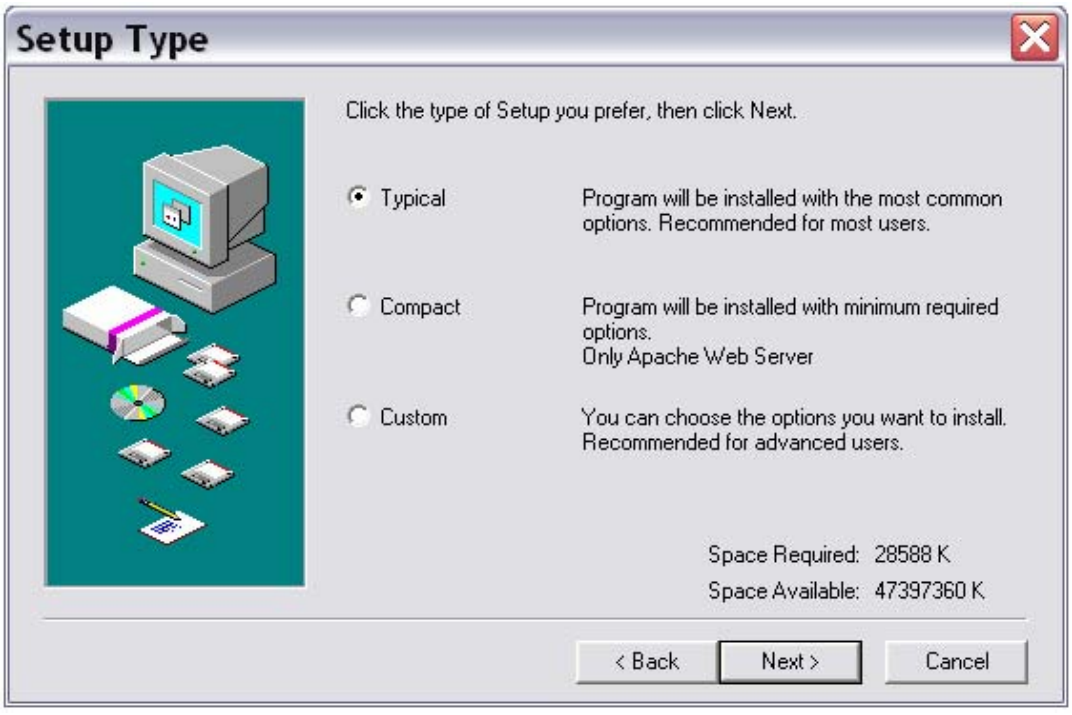
2. Comienza la instalación de la base de datos.



3. Se escoge el directorio en donde se desea instalarlo.



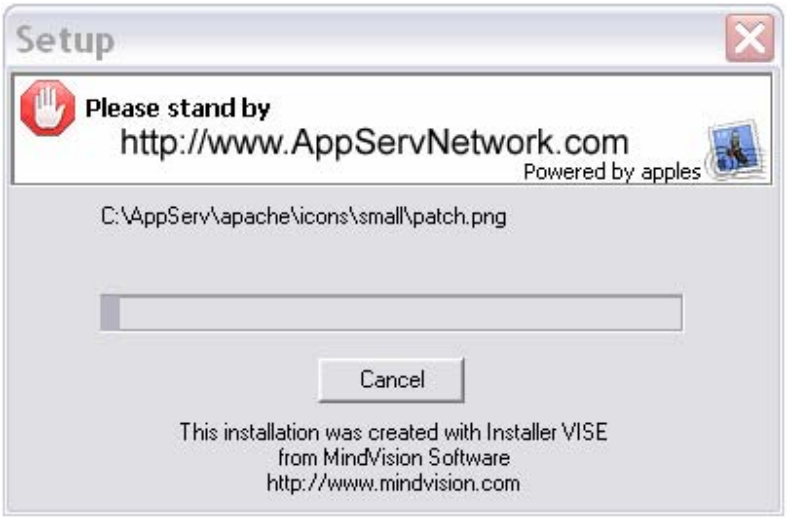
4. Se escoge como se desea que se realice la instalación.



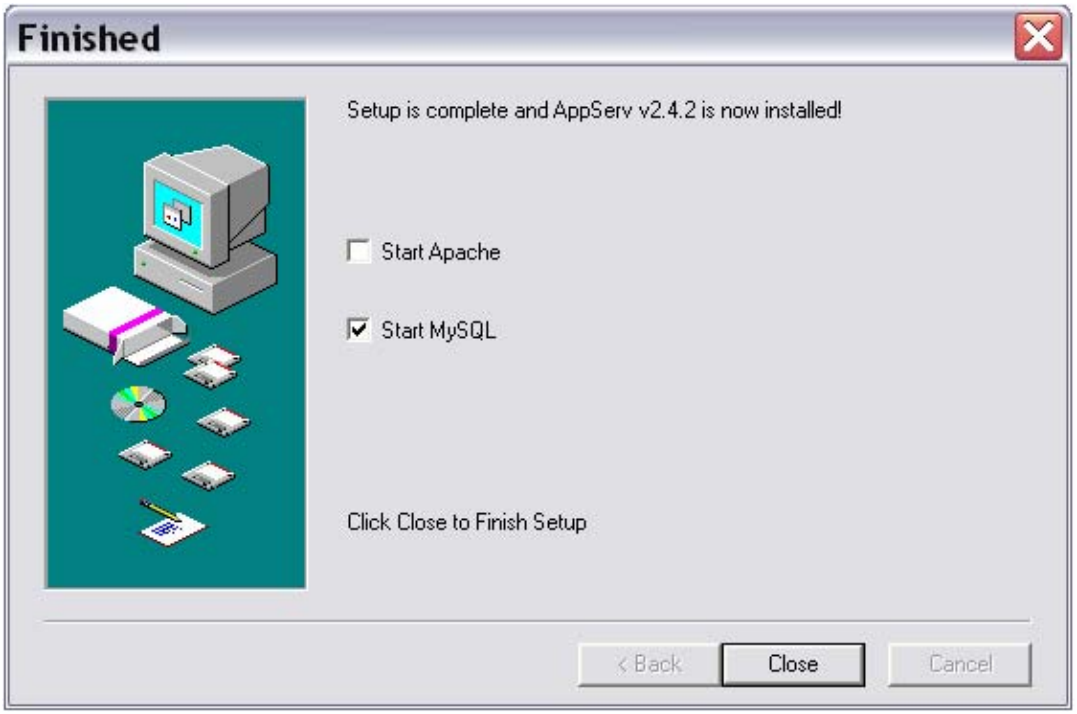
5. Se ingresa la información del servidor.

6. Se ingresa la información de la base de datos.

7. A continuación se procede a realizar la instalación.



8. Se finaliza la instalación.



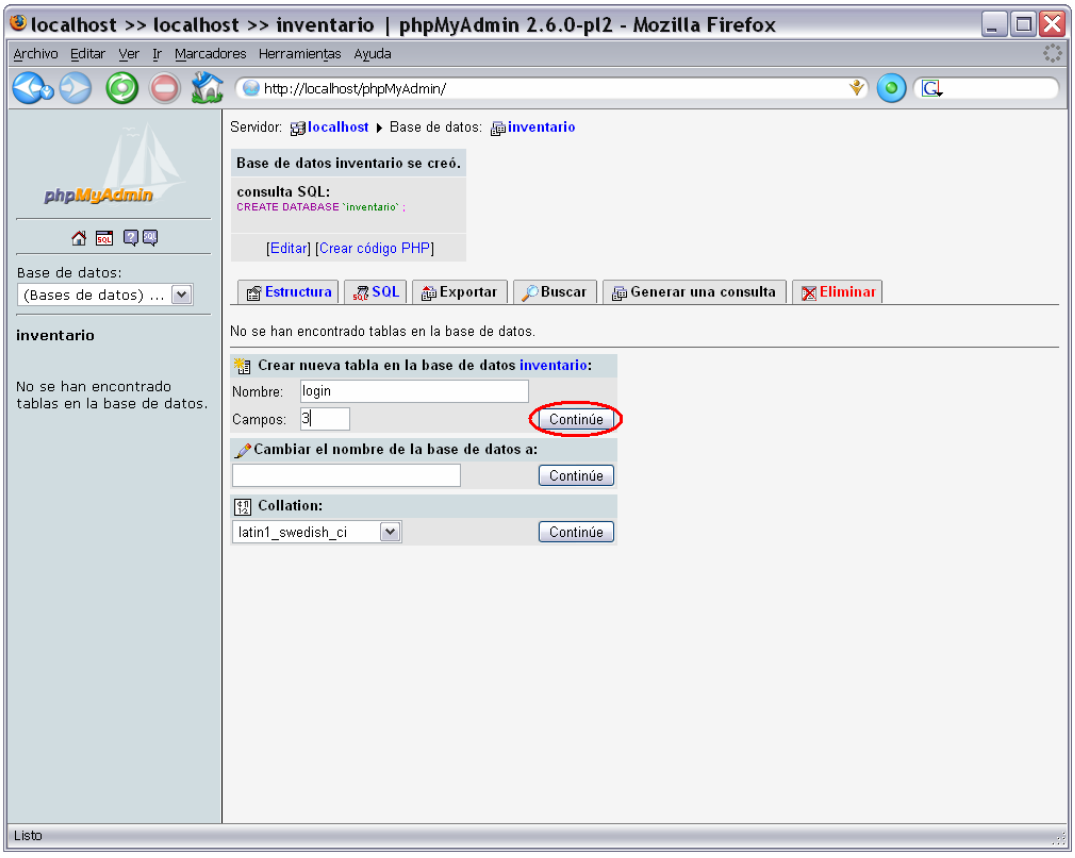
Para agregar el conector de la base de datos, se debe copiar el conector en la siguiente dirección: C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\common\lib\mysql-connector-java-3.1.10-bin

Crear una base de datos

Para poder crear una base de datos se debe iniciar el servicio de MySQL y a continuación abrir el navegador y escribir en la barra de navegación: http://localhost/phpMyAdmin/, en esta ventan se muestran todas las bases de datos que han sido creadas y las que MySQL muestra como ejemplo.

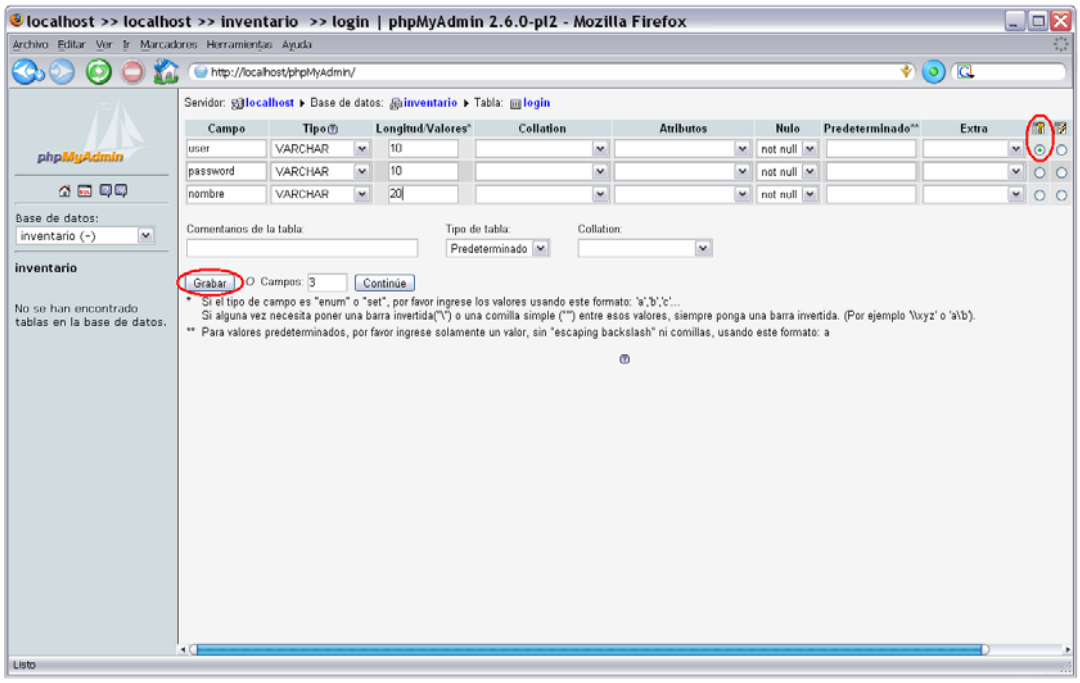
1. Para comenzar a crear la base de datos para la aplicación primero se le da el nombre a la base de datos, en este ejemplo se creo con el nombre de inventario.

2. A continuación aparece una nueva ventana en donde indica que no se han creado ninguna tabla, entonces se procede a crear las dos tablas que son necesarias para el desarrollo de la aplicación, login y producción; la primera servirá para que las personas se puedan autenticar con el servidor antes de ver cualquier información de la base de datos que estará ubicada en la tabla producción; también se deben especificar el número de campos que tendrá la tabla. Como se muestra en la figura:



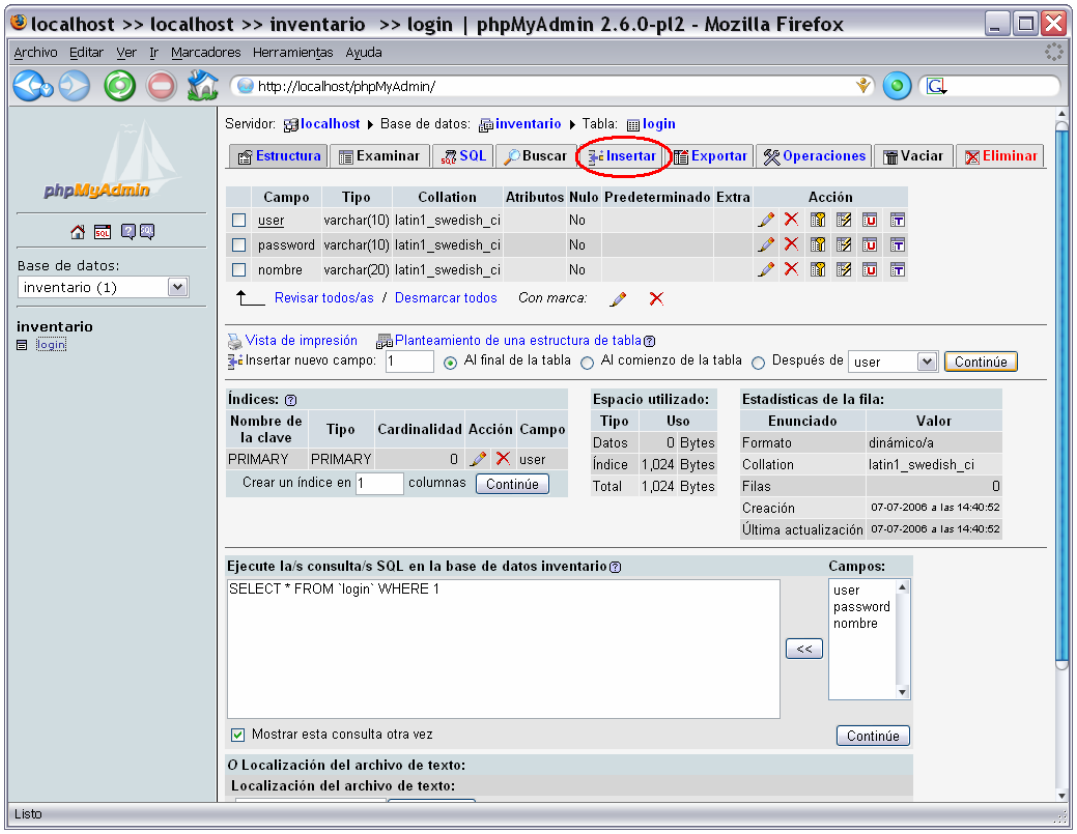
3. En la siguiente ventana se procede a llenar los campos de la tabla. El primer campo corresponderá a la llave primaria y se podrá distinguir con la opción demarcada en la parte izquierda de la fila del campo; una vez

completado todos los campos, se oprime el botón grabar y esta tabla quedará creada. Como lo muestra la figura:



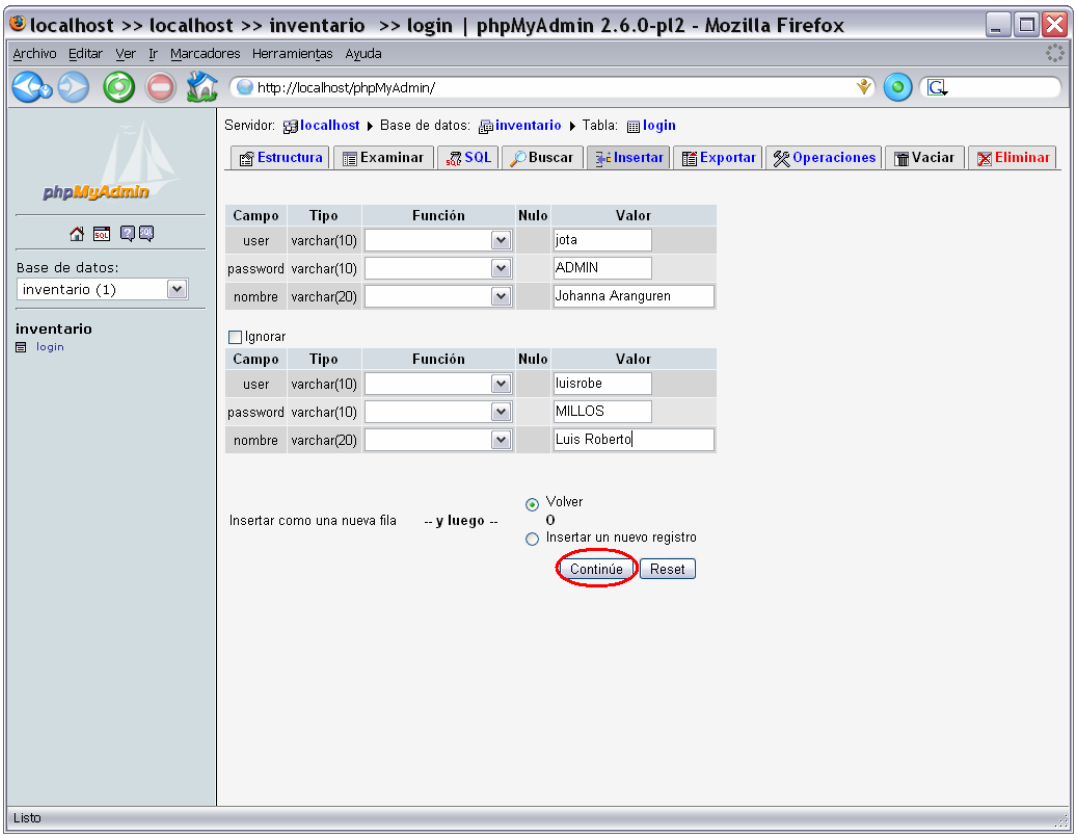
4. Una vez creada la tabla, esta se podrá apreciar en la parte izquierda de la ventana. El siguiente paso es poder llenar la tabla con la información requerida. En la parte superior de la ventana se oprime en link Insertar, el cual sirve para poder ingresar los datos en la tabla; Se realiza como lo muestra la figura:

   Si se desea agregar un nuevo campo se puede hacer desde esta misma ventana.



5. A continuación aparecerá una nueva ventan con los campos anteriormente creados, en los cuales se podrá ingresar los datos para así guardarlos en la tabla; como se indica en la figura:

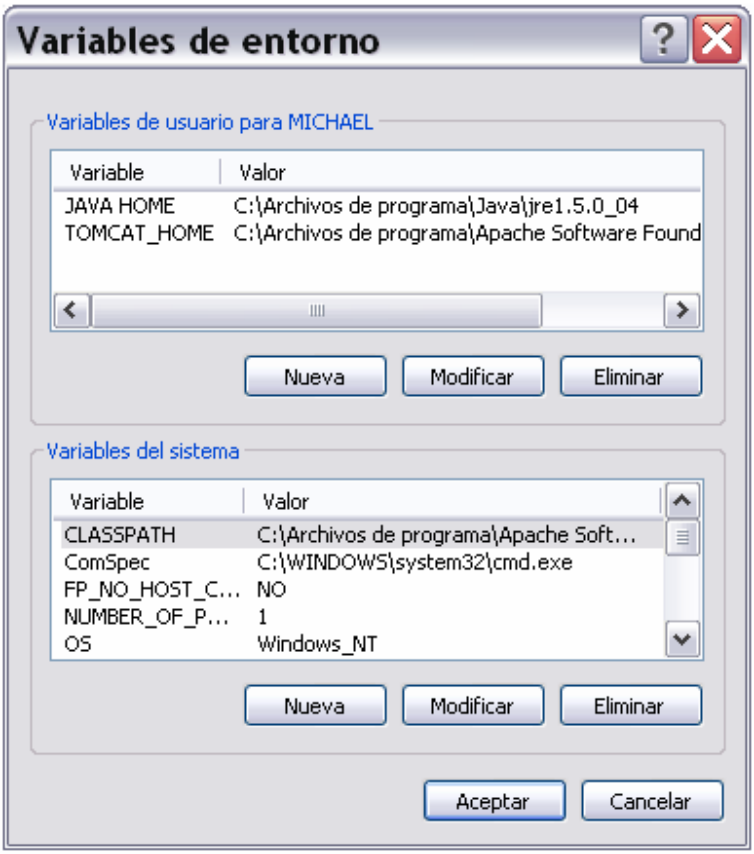
   Hay dos opciones de guardar los datos; la primera es que al guardarlos la venta que se muestra a continuación es la principal de la tabla, y la segunda es que se puede ingresar datos y al guardarlos se carga nuevamente la misma página para seguir guardando más datos. Y para finalizar se oprime en el botón Continúe para guardar los datos escritos.



Si se desea agregar una nueva tabla a la base de datos se debe realizar los pasos anteriores.

ANEXO 3: AGREGAR LAS VARIABLES DE ENTORNO

Una vez instalada la base de datos, el servidor Tomcat y la Máquina de Java J2RE, se oprime inicio y en Mi PC se oprime clic derecho en propiedades; en la ventana que aparece se oprime en la pestaña que dice Opciones Avanzadas y por último se oprime en el botón Variables de Entorno y enseguida aparece una ventana en donde se debe agregar las variables que se muestran en la siguiente figura:



JAVA_HOME = C:\Archivos de programa\Java\jre1.5.0_04; se escribe la dirección en donde está instalado la Máquina Virtual de Java.

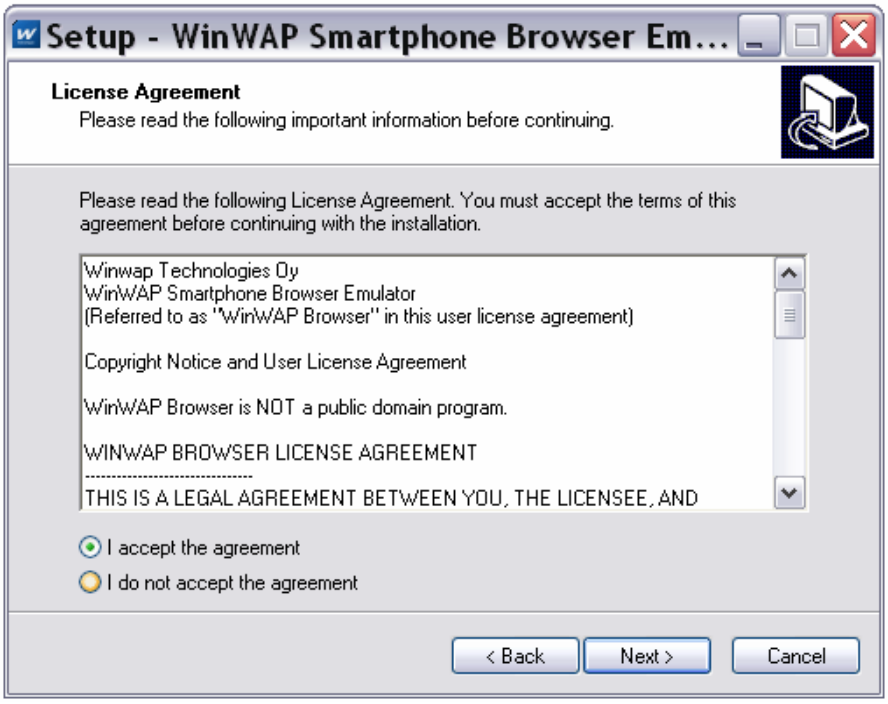
TOMCAT_HOME = C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5; se escribe la dirección en donde está instalado el servidor Tomcat.

ANEXO 4: INSTALACIÓN DEL EMULADOR WINWAP

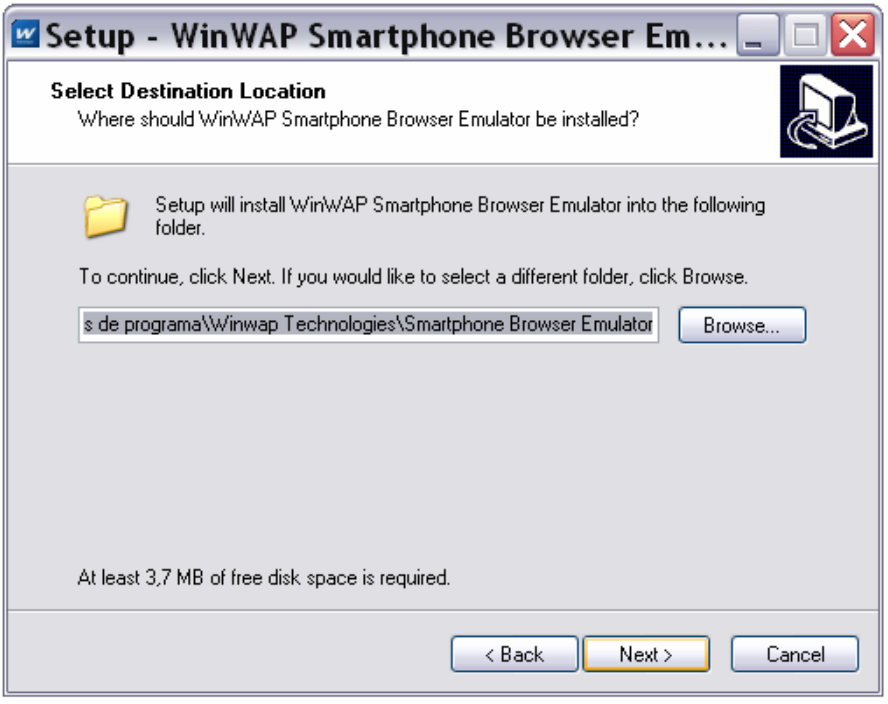
Para instalar el emulador en el cual podrá dar una idea de cómo se va a ver la aplicación desde un celular o cualquier otro dispositivo móvil, se ejecuta el archivo.

1. En la primera ventana se inicia los pasos para la configuración de la instalación del emulador.

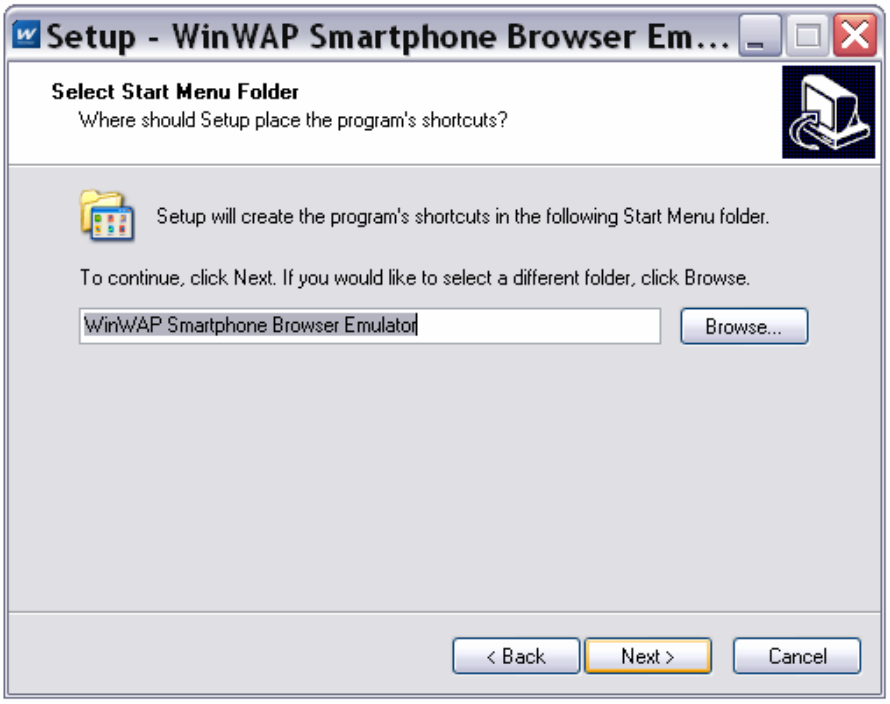
2. Para poder continuar con la instalación se debe aceptar los términos y condiciones de uso.



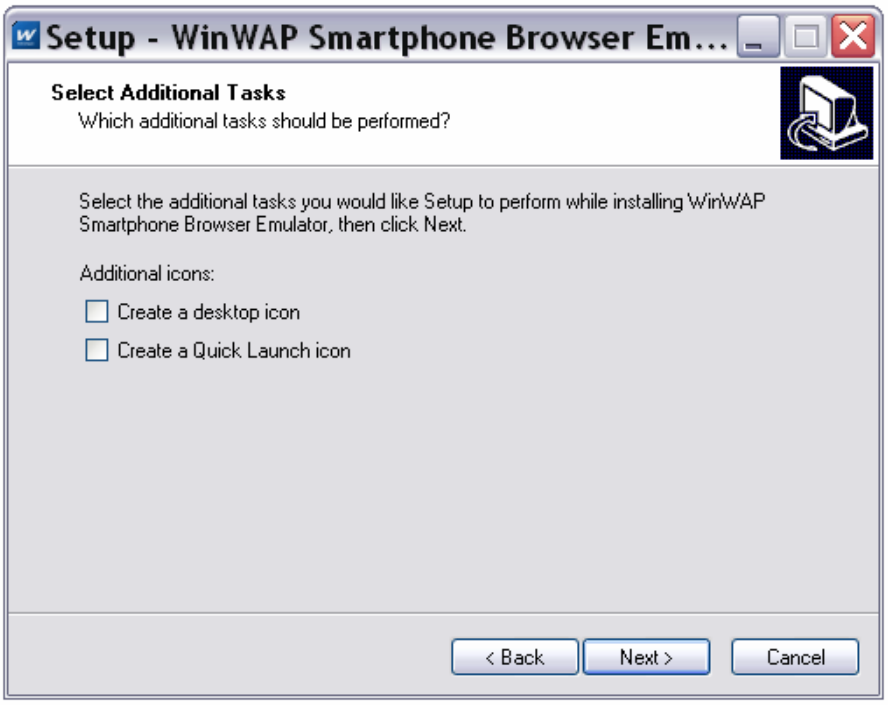
3. En esta ventana se escoge el directorio en donde va a instalarse.



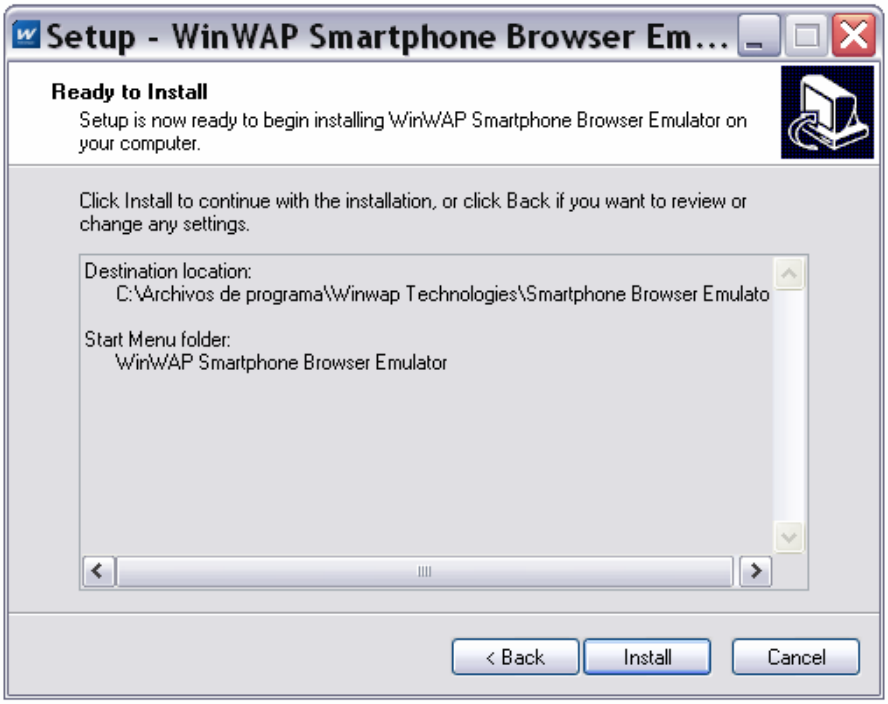
4. En la esta ventana por defecto se instalara en archivos de programa.



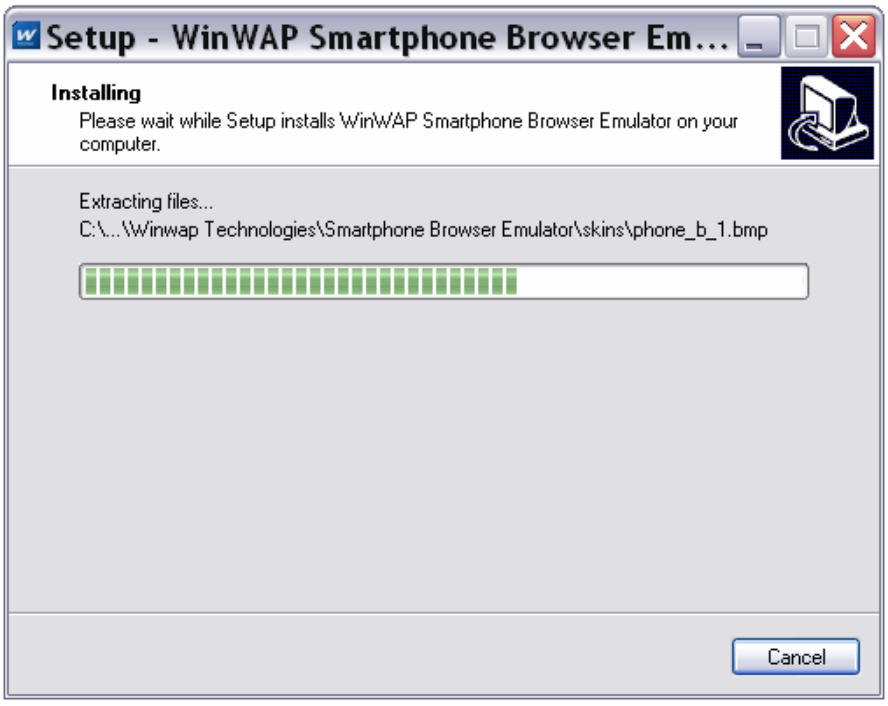
5. Se escogen algunas opciones adicionales.



6. Antes de comenzar la instalción se muestra la configuración escogida.



7. Se observa la instalación de la aplicación.



8. En esta última ventana se da por terminado la aplicación y se lanza el mutador para su posterior uso.

ANEXO 5: VISTA DE LA APLICACIÓN A TRAVES DE UN SIMULADOR

Los requisitos necesarios para poder ver la aplicación en un simulador se deben iniciar los servicios de Tomcat y MySQL. Las páginas que se necesitan para crear la aplicación son creadas en la carpeta ROOT de Tomcat, dentro de esta carpeta se ha creado una carpeta llamada inv.

Esta aplicación consiste en que el usuario final va revisar un inventario de la producción de prendas de vestir de hombres y mujeres, en donde el interés final es conocer la totalidad de prendas confeccionadas por cada prenda o en

su totalidad. Para poder acceder a la base de datos, el usuario debe autentificarse para poder tener acceso a la misma.

1. Para poder visualizar la aplicación, se inicia el emulador, en este debe aparecer la página principal, se oprime el botón “select” y se escoge la opción “Enter Addres...”

2. A continuación se debe ingresar la primera página, esta se muestra a través de la dirección URL, si se está trabajando localmente, http://localhost:8080/inv/login.wml y se oprime el botón “Go!”.

3. Esta es la ventana principal de la aplicación, para poder continuar se debe ingresar el usuario y la contraseña y oprimir en el botón “Select” y escoger la opción “Aceptar” el cual ha sido creado para esta baraja.

   Cada emulador y cada dispositivo móvil, se muestran las páginas wml de manera diferente, esto sucede porque la tecnología varía.

   Esa forma en como son mostradas las páginas de esta aplicación pueden variar de acuerdo al dispositivo móvil, con ayuda de este emulador se puede comprobar el buen funcionamiento de las páginas.

4. Esta es la segunda ventana en donde se verifican los datos anteriormente ingresados, para poder continuar se debe oprimir en el link “aquí”.

5. Esta es la tercera ventana en la cual se puede escoger la opción que se desea consultar; para poder continuar se debe escoger cualquiera de los tres links: “Hombres”, “Mujeres” o “Consulta general”.

6. Si se ha escogido las opciones “hombre” o “Mujer”, estas son las ventanas que se muestran, en donde para poder ver la producción de una prenda específica se debe escoger la opción que allí se muestra. Para poder escoger otra opción se debe oprimir en “Select” y escoger la opción “Back”.

7. En la opción “Consulta general”, se muestra la producción total. Para poder volver se debe oprimir en “Select” y escoger la opción “Back”.

ANEXO 6: CODIGO FUENTE DE LA APLICACIÓN

Las páginas mostradas a continuación son mostradas en orden para que la aplicación pueda llevar la lógica mostrada en el emulador.

login.wml

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <!--Se ingresa el User  y Password para la utenticación del mismo-->
    <card title="Inicio" id="inicio">
         <!--Lee el usuario y el password -->
        <p>Usuario: <input name="userinput" format="*x"/></p>
        <p>Password: <input name="passwordinput" type="password"/></p>
        <!--Se oprime el boton Aceptar y se envia la información al servidor -->
        <do type="accept" label="Aceptar">
            <go href="validator.jsp?userinput=$(userinput:e)&amp;
                    passwordinput=$(passwordinput:e)" method="post"/>
        </do>
        <!--Se oprime al boton Borrar para limpiar el usuario y el password -->
        <do type="reset" label="Borrar">
            <refresh>
                <setvar name="userinput" value=""/>
                <setvar name="passwordinput" value=""/>
            </refresh>
        </do>
    </card>
</wml>
```

validator.jsp

```
<%@ page import="java.sql.*" %>
<%
    String connectionURL = "jdbc:mysql://localhost:3306/inventario";
    Connection connection = null;
    Statement statement = null;
    ResultSet rs = null;
    String usu = request.getParameter("userinput");
    String pas = request.getParameter("passwordinput");
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    connection = DriverManager.getConnection(connectionURL, "", "");
    statement = connection.createStatement();
    rs = statement.executeQuery("SELECT * FROM login
                                        WHERE user='"+usu+"' and password='"+pas+"'");
    out.println("<?xml version='1.0'?>");
    out.println("<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'
                    'http://www.wapforum.org/DTD/wml_1.1.xml'>");
    out.println("<wml>");
        out.println("<card title='Ingreso' id='ingreso'>");

    if(rs.next()){
            out.println("<p>Bienvenido "+rs.getString("nombre")+".</p>");
            out.println("<p>Para continuar oprima <anchor>aquí
                            <go href='entrada.wml'></go></anchor>.</p>");
    }else{
            out.println("<p>Usuario invalido.</p>");
            out.println("<p>Para continuar oprima <anchor>aqui
            <go href='login.wml'></go></anchor>.</p>");
    }
        out.println("</card>");
    out.println("</wml>");
    rs.close();
%>
```

entrada.wml

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <!--Se muestra la opcion para mostrar la produccion de la ropa de hombre o mujer -->
    <card title="Principal" id="principal" >
        <!-- Al escojer la opcion se para a un nuevo card -->
        <p>Revisar la produccion de:</p>
        <p>
            <anchor>Hombres
                <go href="hombre.wml"/>
            </anchor>
        </p>
        <p>
            <anchor>Mujeres
                <go href="mujer.wml"/>
            </anchor>
        </p>
        <p>
            <anchor>Consulta general
                <go href="general.jsp"/>
            </anchor>
        </p>
    </card>

</wml>
```

hombre.wml

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <!--Se muestra la produccion de la ropa del hombre -->
    <card title="Hombre" id="hombre">
        <p>Escoja la prenda masculina que desea revisar.</p>
        <p>
            <fieldset title="Masculino">
                <select name="name">
                    <option value="busos" onpick="resultado.jsp?
                        genero=masculino&amp;value=busos" method="post">Busos
                    </option>
                    <option value="camisas" onpick="resultado.jsp?
                        genero=masculino&amp;value=camisas" method="post">Camisas
                    </option>
                    <option value="camisetas" onpick="resultado.jsp?
                        genero=masculino&amp;value=camisetas" method="post">Camisetas
                    </option>
                    <option value="chaquetas" onpick="resultado.jsp?
                        genero=masculino&amp;value=chaquetas" method="post">Chaquetas
                    </option>
                    <option value="jeans" onpick="resultado.jsp?
                        genero=masculino&amp;value=jeans" method="post">Jeans
                    </option>
                </select>
            </fieldset>
        </p>
    </card>
</wml>
```

mujer.wml

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
    "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
    <card title="Mujer" id="mujer"><!--Se muestra la producción de la ropa de la mujer -->
        <p>Escoja la prenda femenina que desea revisar.</p>
        <p>
            <fieldset title="Femenino">
                <select name="onpick">
                    <option value="blusas" onpick="resultado.jsp?
                        genero=femenino&amp;value=blusas" method="post">Blusas
                    </option>
                    <option value="busos" onpick="resultado.jsp?
                        genero=femenino&amp;value=busos" method="post">Busos
                    </option>
                    <option value="camisetas" onpick="resultado.jsp?
                        genero=femenino&amp;value=camisetas" method="post">Camisetas
                    </option>
                    <option value="chaquetas" onpick="resultado.jsp?
                        genero=femenino&amp;value=chaquetas" method="post">Chaquetas
                    </option>
                    <option value="jeans" onpick="resultado.jsp?
                        genero=femenino&amp;value=jeans" method="post">Jeans
                    </option>
                    <option value="shorts" onpick="resultado.jsp?
                        genero=femenino&amp;value=shorts" method="post">Shorts
                    </option>
                </select>
            </fieldset>
        </p>
    </card>
</wml>
```

general.jsp

```
<%@ page import="java.sql.*" %>
<%
    String connectionURL = "jdbc:mysql://localhost:3306/inventario";
    Connection connection = null;
    Statement statement = null;
    ResultSet rs = null;

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    connection = DriverManager.getConnection(connectionURL, "", "");
    statement = connection.createStatement();
    rs = statement.executeQuery("SELECT * FROM produccion");

    out.println("<?xml version='1.0'?>");
    out.println("<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'
                    'http://www.wapforum.org/DTD/wml_1.1.xml'>");
    out.println("<wml>");
        out.println("<card title='Bodega' id='bodega'>");
            out.println("<p>Estas son las existencias general de todas la prendas.</p>");

            while (rs.next()){
                out.println("<p><b>Diseño: </b>"+rs.getString("diseno")+"</p>");
                out.println("<p><b>Talla: </b>"+rs.getString("talla")+"</p>");
                out.println("<p><b>Cantidad: </b>"+rs.getString("cantidad")+"</p>");
                out.println("<p><b>Color: </b>"+rs.getString("color")+"</p>");
            }

        out.println("</card>");
    out.println("</wml>");

    rs.close();
%>
```

resultado.jsp

```
<%@ page import="java.sql.*" %>
<%
    String connectionURL = "jdbc:mysql://localhost:3306/inventario";
    Connection connection = null;
    Statement statement = null;
    ResultSet rs = null;

    String opc = request.getParameter("value");
    String gen = request.getParameter("genero");

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    connection = DriverManager.getConnection(connectionURL, "", "");
    statement = connection.createStatement();
    rs = statement.executeQuery("SELECT * FROM produccion
                                        WHERE diseno='"+opc+"' and persona='"+gen+"'");
    out.println("<?xml version='1.0'?>");
    out.println("<!DOCTYPE wml PUBLIC '-//WAPFORUM//DTD WML 1.1//EN'
                    'http://www.wapforum.org/DTD/wml_1.1.xml'>");
    out.println("<wml>");
        out.println("<card title='Bodega' id='bodega'>");
            out.println("<p>Estas son las existencias del modelo "+gen+".</p>");
            while (rs.next()){
                out.println("<p><b>Diseño: </b>"+rs.getString("diseno")+"</p>");
                out.println("<p><b>Talla: </b>"+rs.getString("talla")+"</p>");
                out.println("<p><b>Cantidad: </b>"+rs.getString("cantidad")+"</p>");
                out.println("<p><b>Color: </b>"+rs.getString("color")+"</p>");
            }
        out.println("</card>");
    out.println("</wml>");
    rs.close();
%>
```

BIBLIOGRAFIA

- WAP guía practica para usuarios, Autor Andy Dornan, Ediciones ANAYA MULTIMEDIA (GRUPO ANAYA, S.A.), 2001.

- Learning WML y WMLScript, Autor Martin Frost, O'REILLY, Octubre 2000.

- http://www.programacion.com/tutorial/tomcatintro/

- http://tomcat.apache.org/

- http://www.apl.jhu.edu/%7Ehall/java/Servlet-Tutorial/

- http://www.apl.jhu.edu/%7Ehall/java/Servlet-Tutorial/Servlet-Tutorial-Overview.html

- Core Servlets and Java Server Pages, Autor Marty Hall, A Sun Microsystem Press/Prentice Hall PTR Book.
  Para mayor información: http://pdf.coreservlets.com/

- Emulador WinWap, descargar la versión gratis en la dirección http://www.winwap.com/downloads.php

- Descargar la versión 5 de Apache Tomcat gratis en al dirección http://tomcat.apache.org/download-55.cgi