palasi.com
IT and management

Programación para el Web con Java

Dr. Vicent-Ramon Palasí Lallana
Universidad Francisco Gavidia
Mayo 2005

¿De qué trata este curso?

- El curso trata sobre la programación de software para el Web con el lenguaje de programación Java.

Presentación del profesor

- Dr. Vicent-Ramon Palasí Lallana.
  - Licenciado en Informática por Universidad Politécnica de Cataluña (Barcelona, España)
  - Doctor en Ingeniería Informática por Universidad Jaume I (Castellón, España).
  - Máster en Nuevas Tecnologías aplicadas a la Educación por la Universidad Carlos III (Madrid, España).
  - Trece años de experiencia profesional en España y El Salvador: desarrollo de software, gerencia y en educación pública y privada.
  - En la actualidad, Gerente General de Aurum Solutions: desarrollo de sistemas para empresas salvadoreñas y extranjeras.

¿Y sobre Java y su capacitación?

- Desarrollo de sistemas en n-capas en Java para empresas españolas y salvadoreñas.
- Capacitación en Java de tres meses para Secretaría Técnica de la Presidencia, con asistentes de varios Ministerios, desde lo más básico a lo más avanzado.
- Curso de Maestría sobre Programación Web en Java el año 2004.

Presentación de los participantes

- Nombre.
- Empresa.
- A qué se dedican.

Programa del curso

- Objetivos y metodología del curso.
- Tema 1. Introducción a la programación Web en Java.
- Tema 2. Fundamentos de la programación orientada a objetos.
- Tema 3. Motores de persistencia.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Objetivos y metodología del curso

Objetivos y metodología del curso

- 1. Motivación de aprender el lenguaje Java.
- 2. Objetivos del curso.
- 3. Enfoque que se usará.
- 4. Metodología.

Objetivos y metodología del curso

- 1. Motivación de aprender el lenguaje Java.
- 2. Objetivos del curso.
- 3. Enfoque que se usará.
- 4. Metodología.

Motivación de aprender el lenguaje Java

- ¿Por qué Java?
- ¿Por qué no otro lenguaje?
- Pregunta: ¿si sólo pudieran aprender un idioma humano a parte del nativo, qué idioma aprenderían? ¿Por qué?

El inglés es un lenguaje estándar

- Es el más usado en las situaciones de negocios. Por ello, su conocimiento ayuda al éxito laboral.
- En aplicaciones para empresas, Java es el lenguaje dominante y estándar.
- ¿Cómo lo sabemos?

Algunos datos

- El 75% de las aplicaciones corporativas están escritas en Java (BZ Search).
- Java es una industria de 100 mil millones de dólares por año.
- Mayor implantación en servicios Web, procedimientos almacenados en BD.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Pero además

- Java es el lenguaje estándar para importantes productos de software:
  - La base de datos Oracle.
  - El ERP SAP.
- El lenguaje único de los servidores de aplicaciones (excepto uno): BEA WebLogic, IBM Websphere, Oracle 9i, Macromedia JRun, JBoss, etc.
- 2.2 mil millones de dólares son invertidos anualmente en servidores de aplicaciones Java y 110 mil millones en tecnologías relacionadas con Java.

Pero además

- Java es la plataforma que tiene más número de herramientas. Una inmensa cantidad de ellos: Motores de persistencia, frameworks MVC, librerías de todos los tipos, herramientas de desarrollo y un larguísimo etcétera.
- Lo que es mejor: la inmensa mayoría de ellos son de código abierto. Por ejemplo, en Sourceforge, hay quince mil proyectos en Java. Pero hay otros sitios: java.net, fundación Apache (aquí veremos algunos: Tomcat, Eclipse, Lomboz, Hibernate).
- Para cualquier área : estadísticas, redes neuronales, ERP, gráficos, celulares, etc. seguro que hay algún software de código abierto en Java.

En los celulares

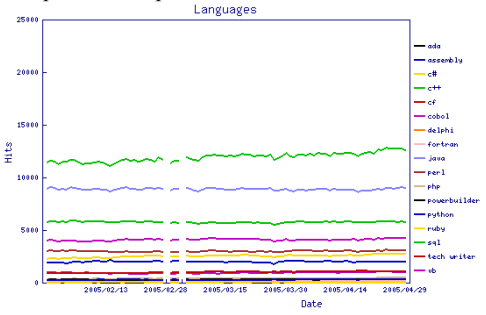
- Hay cerca de 100 implantaciones de Java en celulares y 579 millones de celulares con Java.
- 7 de cada 10 aplicaciones inalámbricas en construcción usarán Java.
- El mercado de juegos Java en celulares es estimado en 3 mil millones de dólares.

En el mercado de trabajo

- Globalmente cerca de 4.5 millones de desarrolladores trabajan con Java.
- Java es el lenguaje de programación más demandado en las búsquedas de empleo (excepto SQL). Por ejemplo, Monster.com.

Por ejemplo, Dice.com

- A parte de SQL (que es diferente), Java encabeza la búsqueda de empleo.



Pero además

- Java es gratuito. Además, uno puede ver los fuentes.
- La mayoría de herramientas de Java son gratuitas.
- El presente del desarrollo se llama Java. Microsoft ha copiado Java (J2EE) y lo llama .NET

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Pero además

- Java es un lenguaje moderno y en plena efervescencia.
- Es muy diferente de dinosaurios como COBOL o, en menor medida, C.
- Tiene cientos de miles de proyectos de código abierto y miles de especificaciones

En El Salvador

- La Administración lo ha escogido como el lenguaje estándar de los Ministerios.*
- Es la herramienta que están implementando las grandes empresas: bancos, SIMAN, TACA, etc.
- Sin embargo, desgraciadamente, el conocimiento de Java por parte de los programadores es bajo. Muchas veces se contrata personal extranjero*.
- Por eso, aprender Java a nivel empresarial* es una excelente inversión para la carrera profesional.

Objetivos y metodología del curso

- 1. Motivación de aprender el lenguaje Java.
- 2. Objetivos del curso.
- 3. Enfoque que se usará.
- 4. Metodología.

Objetivos generales del curso

- Ser capaz de desarrollar programas en Java para el Web que hagan uso de una base de datos.
- Adquirir buenas prácticas de programación desde el principio que hagan los programas flexibles y escalables.

Objetivos específicos del curso

- Conocer la sintaxis y características del lenguaje Java.
- Aprender a pensar y programar en tecnología orientada a objetos.
- Identificar los diferentes componentes de la plataforma Java.
- Aprender a pensar y programar en arquitectura de n-capas.
- Aumentar la calidad y la eficiencia de la programación y reducir su costo y mantenimiento.

Objetivos y metodología del curso

- 1. Motivación de aprender el lenguaje Java.
- 2. Objetivos del curso.
- 3. Enfoque que se usará.
- 4. Metodología.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

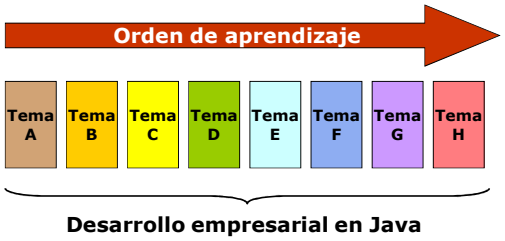
Una ventaja de este curso

- Este curso implementa un enfoque nuevo para enseñar Java (hasta donde se sabe).
- Este enfoque es diferente al tradicional que se encuentra casi universalmente en los libros y en el material de la red.

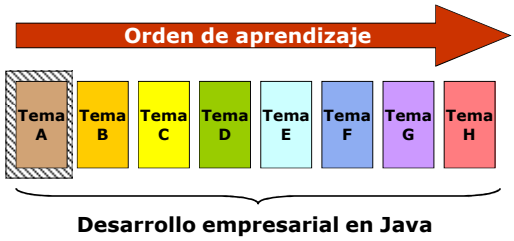
Un enfoque común para aprender Java (temas principales)

- 1. Fundamentos. Compilación, JDK, etc.
- 2. Aspectos procedimentales del lenguaje.
- 3. Programación orientada a objetos.
- 4. Excepciones, librerías, multithreading, etc.
- 5. Interfaz gráfica de escritorio.
- 6. Bases de datos: JDBC
- 7. Programación para el Web: servlets, JSP.
- 8. Arquitectura MVC, n-tier, etc.
- 9. Testing
- 10. Enterprise Javabeans.
- 11. Seguridad.
- 12. Análisis y diseño O-O.
- 13. Buenas prácticas. Patrones de diseño.

Este es un enfoque lineal

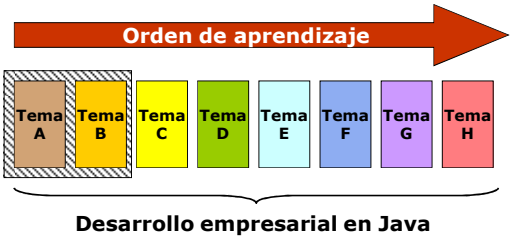


1. El enfoque común para aprender Java



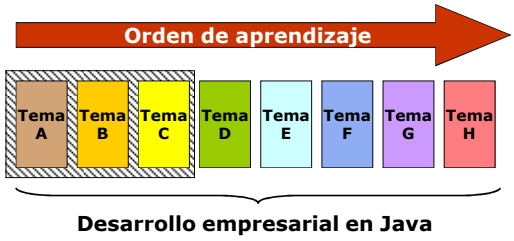
- El cuadrado con trama indica lo que se ha aprendido hasta el momento.

2. El enfoque común para aprender Java



- El cuadrado con trama indica lo que se ha aprendido hasta el momento.

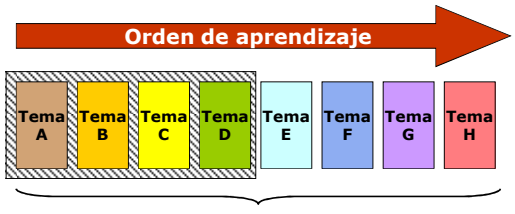
3. El enfoque común para aprender Java



- El cuadrado con trama indica lo que se ha aprendido hasta el momento.

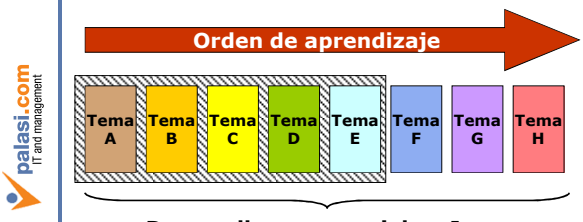
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

4. El enfoque común para aprender Java



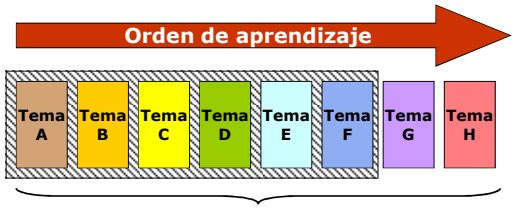
- El cuadrado con trama indica lo que se ha aprendido hasta el momento.

5. El enfoque común para aprender Java



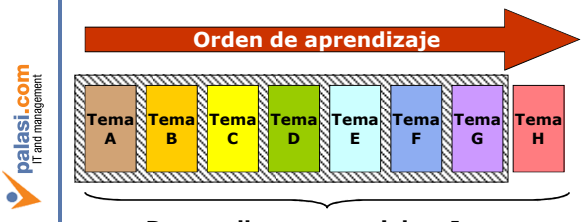
- El cuadrado con trama indica lo que se ha aprendido hasta el momento.

6. El enfoque común para aprender Java



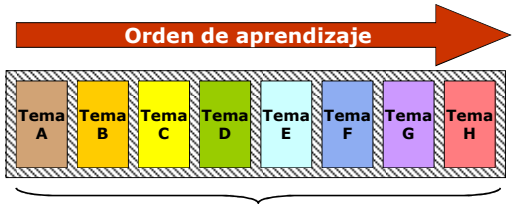
- El cuadrado con trama indica lo que se ha aprendido hasta el momento.

7. El enfoque común para aprender Java



- El cuadrado con trama indica lo que se ha aprendido hasta el momento.

8. El enfoque común para aprender Java



- El cuadrado con trama indica lo que se ha aprendido hasta el momento.

Problemas de este enfoque

- 1. Es un enfoque lineal: se aprende todo un tema y después otro tema.
- 2. El aprendiz tarda mucho tiempo en poder realizar una aplicación medianamente real (hasta JDBC).
- 3. El aprendiz aún tarda más tiempo en poder realizar una aplicación para el Web (el punto fuerte de Java).
- 4. El aprendiz tarda mucho tiempo en saber cómo está estructurada una aplicación Java (hasta arquitectura).
- 5. El aprendiz aprende a hacer las cosas mal y después aprende a hacerlas bien.
- 6. JDBC es difícil de gestionar (excepciones) y no permite fácilmente una metodología O-O.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

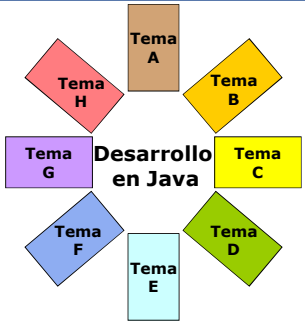
Problemas de este enfoque

- Este enfoque es el causante de que el conocimiento de Java en El Salvador sea tan bajo.
- Como es tan lento, la mayoría de aquellos que reciben cursos de Java sólo aprenden a hacer programas de juguete: con la consola, sin base de datos.
- Hay pocos que sean capaces de programar una aplicación empresarial de tamaño real.
- Aquí es lo que queremos aprender. ¿Cómo lo haremos?

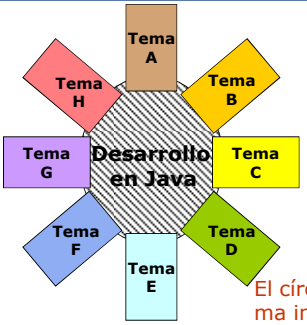
Lo haremos con un enfoque diferente

- 1. Repaso de HTML. Introducción a JSP.
- 2. Elementos de guión (Aspectos procedimentales).
- 3. Programación O-O.
- 4. Bases de datos: Motores de persistencia
- 5. Librerías, multithreading, ampliación O-O, etc.
- 6. Arquitectura MVC.
- 7. Más sobre el Web: cookies, seguridad.
- 8. Análisis y diseño O-O.
- 9. Testing.
- 10. Enterprise Javabeans.
- 11. Seguridad.

Nuestro enfoque es concéntrico

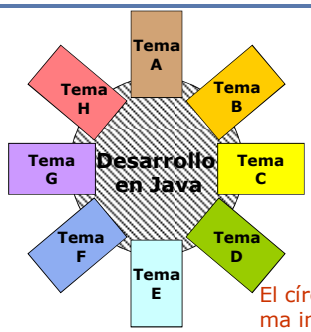


1. Nuestro enfoque es concéntrico



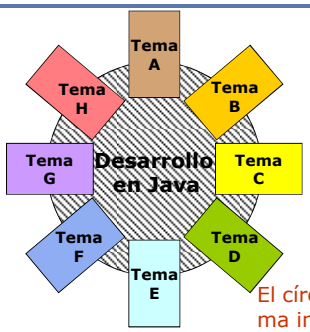
El círculo con trama indica los temas aprendidos hasta el momento

2. Nuestro enfoque es concéntrico



El círculo con trama indica los temas aprendidos hasta el momento

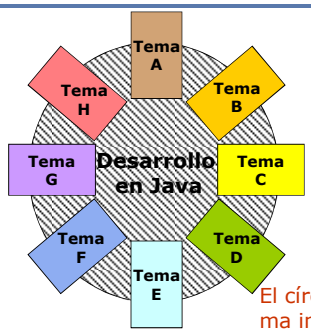
3. Nuestro enfoque es concéntrico



El círculo con trama indica los temas aprendidos hasta el momento

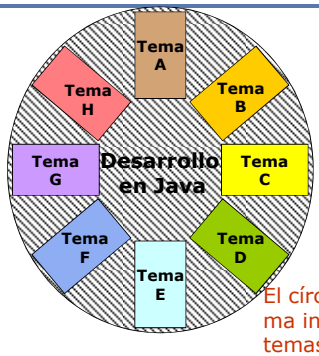
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

4. Nuestro enfoque es concéntrico



El círculo con trama indica los temas aprendidos hasta el momento

5. Nuestro enfoque es concéntrico



El círculo con trama indica los temas aprendidos hasta el momento

Ventajas de este enfoque

- 1. Es un enfoque concéntrico: se aprende lo principal de los temas y después se va ampliando.
- 2. El aprendiz sabe desde el principio programar una aplicación para el Web.
- 3. El aprendiz sabe tempranamente cómo está estructurada una aplicación Java.
- 4. El aprendiz aprende a hacer las cosas bien desde el principio (n-tier, buenas prácticas y patrones de diseño se aprenden sobre la marcha).
- 5. No se usa JDBC, sino motores de persistencia que son más sencillos y permiten fácilmente una metodología O-O.

El enfoque de este curso

- 1. Repaso de HTML. Introducción a JSP.
- 2. Elementos de guión (Aspectos procedimentales).
- 3. Programación O-O.
- 4. Bases de datos: Motores de persistencia
- 5. Librerías, multithreading, ampliación O-O, etc.
- 6. Arquitectura MVC.
- 7. Más sobre el Web: cookies, seguridad.
- 8. Análisis y diseño O-O.
- 9. Testing.
- 10. Enterprise Javabeans.
- 11. Seguridad.

Objetivos y metodología del curso

- 1. Motivación de aprender el lenguaje Java.
- 2. Objetivos del curso.
- 3. Enfoque que se usará.
- 4. Metodología.

Metodología utilizada

- Presentaciones en Powerpoint. Para transmitir los conceptos teóricos del curso.
- Ejercicios prácticos. Para asimilar los conceptos en programas prácticos.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ventajas de esta metodología

- Combinación de teoría y práctica.
- Enfoque que va de lo abstracto a lo concreto.
- Fomenta la participación del alumno.
- Asegura el aprendizaje

Algunas indicaciones

- Se valora la participación: tanto espontánea como programada.
- Se anima a hacer tantas preguntas como se desee: no hay ningún momento en que una pregunta sea inadecuada.

Prerrequisitos del curso

- El curso empieza desde nivel cero.
- Aun así, debido al ritmo, se necesita un conocimiento previo de un lenguaje de programación interactivo (Basic y var., C y var., COBOL, Pascal, FoxPro, etc)
- Éste es el único prerrequisito que se necesita.

Notación de estas transparencias

- Las instrucciones MS-DOS se escribirán en `Courier New negrita.`
- Los textos que aparezcan en Windows (textos de menús, de botones, etc) se escribirán en **Arial Negrita**.
- El código fuente en Java (o en general, cualquier contenido de un archivo de texto) se escribirá en letra `Courier New negrita` (normalmente con fondo amarillo).
- Si en alguna parte del código fuente, hay un dato que varía según el programa o el caso concreto (por ejemplo el nombre del programa) se escribirá en ***Arial negrita y cursiva***. Por ejemplo, `class nombreClase`

Notación de estas transparencias

- Cuando se trata de acciones de la computadora (Windows, programas, Web, etc.), la barra vertical | se utiliza para separar las acciones consecutivas. Significa "a continuación".
- Así **Inicio|Panel de control|Sistema** es la forma abreviada de decir.
  - Hacer clic en **Inicio.**
  - A continuación, hacer clic en **Panel de control.**
  - A continuación, hacer clic **en Sistema.**
- **Cuando hay algo entre corchetes [ ],** es que es opcional.

Programa del curso

- Objetivos y metodología del curso.
- Tema 1. Introducción a la programación Web en Java.
- Tema 2. Fundamentos de la programación orientada a objetos.
- Tema 3. Motores de persistencia.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

palasi.com
IT and management

Tema 1.
Introducción a la programación Web en Java

Programa del tema 1

- 1. Introducción al lenguaje Java.
- 2. Instalación de la plataforma Java.
- 3. Introducción a la programación Web.
- 4. Fundamentos de HTML.
- 5. Introducción a Eclipse.
- 6. Introducción a Tomcat.
- 7. Introducción a JSP.
- 8. Variables. Expresiones aritméticas y lógicas.
- 9. Sentencias algorítmicas: condicionales, ciclos.
- 10. Arreglos.

Programa del tema 1

- 1. Introducción al lenguaje Java.
- 2. Instalación de la plataforma Java.
- 3. Introducción a la programación Web.
- 4. Fundamentos de HTML.
- 5. Introducción a Eclipse.
- 6. Introducción a Tomcat.
- 7. Introducción a JSP.
- 8. Variables. Expresiones aritméticas y lógicas.
- 9. Sentencias algorítmicas: condicionales, ciclos.
- 10. Arreglos.

¿Qué es Java?

- Quien da la definición más acertada.

¿Qué es Java?

- Es un lenguaje de programación orientada a objetos desarrollado por Sun Microsystems.
- Es una plataforma para desarrollo de programas que incluye varios componentes que se interconectan entre sí.

¿Qué es Java?

- Es un lenguaje de programación orientada a objetos desarrollado por Sun Microsystems.
- Es una plataforma para desarrollo de programas que incluye varios componentes que se interconectan entre sí.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

El lenguaje Java

- Java es un lenguaje O-O desarrollado por Sun Microsystems en 1995 (muy reciente).
- Hereda muchos conceptos de C++, aunque tiene una filosofía diferente.
- Se ha extendido rápidamente y se ha convertido en un estándar para la programación en Internet.

Origen del lenguaje Java

- En 1991, Sun intentó introducirse en el mercado de electrodomésticos: se necesitaba un lenguaje independiente del chip.
- James Gosling desarrolló el lenguaje Oak, que fue ocupado en varios proyectos, pero la idea no prosperó.
- En 1995, Bill Joy, cofundador de Sun consideró que Oak (ahora Java) era el lenguaje ideal para la Internet por su independencia del hardware. Sun anuncia Java.

¿Por qué otro lenguaje de programación?

- Java surgió cuando ya había decenas de miles de lenguajes de programación comerciales y algunos muy extendidos.
- ¿Cuáles son las ventajas que justifican su creación?

Ventajas de Java

- Independiente de la plataforma.
- Orientado a objetos.
- Distribuido.
- Robusto.
- Seguro.
- Multihilo.
- Público.

La principal ventaja de Java

- Independencia de la plataforma: un programa escrito en Java puede ejecutarse en plataformas diferentes (siempre que éstas implementen la máquina virtual) sólo copiándolo, sin necesidad de recompilar.
- "Write once, run anywhere" es el mantra de Java.
- Facilita el desarrollo de software. Idóneo para redes heterogéneas, como Internet.

Orientado a objetos

- Organiza el programa como el mundo real: objetos con comportamientos definidos.
- Mejora la gestión de complejidad: divide los problemas en módulos sencillos.
- Mejor calidad y claridad del código: mayor facilidad para diseñar, modificar y mantener.
- Facilita la creación de bibliotecas de clases, y por lo tanto, la reutilización de código.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Distribuido

- Proporciona un soporte de alto nivel para redes.
- Así, por ejemplo, se pueden llamar a procedimientos en otra máquina tan fácilmente como si estuvieran en la misma (RMI, CORBA).
- Esto lo hace especialmente adecuado para Internet.

Robusto

- Java está diseñado para minimizar los errores de programación.
- No permite declaraciones implícitas, ni aritmética de punteros, como en C++.
- Permite un manejo elegante y robusto de las excepciones con la instrucción try/catch.

Seguro

- Java mejora la seguridad implementando una serie de políticas.
- El intérprete verifica el código "binario" de Java para detectar si hay código mal formado o inseguro. En ese caso, no lo ejecuta.
- Todo código indigno de confianza (como el bajado de Internet) se ejecuta en un "cajón de arena" (sandbox), sin acceso al sistema de archivos, protegiendo a la máquina de código malicioso.

Multihilo

- Las aplicaciones modernas realizan varias tareas a la vez, para ello necesitan manejar diferentes hilos de ejecución (thread).
- Java hace muy sencillo manejar diferentes hilos de ejecución gracias a la clase *java.lang.Thread* que incorpora.

Público

- Java es público, gratuito y descargable (pero no de código abierto).
- Una buena parte de herramientas para desarrollo en Java son gratuitas.
- En este curso, sólo utilizaremos herramientas gratuitas.
- Ahorro considerable en licencias.

Una opinión personal

- Para mí, lo mejor de Java, no es el mismo lenguaje, es todo lo que se ha construido alrededor de él.
- Cientos de miles de herramientas (la mayoría de código abierto) que lo cubren prácticamente todo.
- Metodologías que hacen el desarrollo más flexible (más importantes en el mundo Java): patrones de diseño, AOP, análisis y diseño con modelo de dominio, etc.
- Una vibrante comunidad de programadores.
- Toda una industria en constante movimiento, mejora y progreso.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

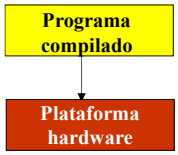
Desventajas de Java

- No permite el acceso directo al hardware de la PC (se puede hacer de forma indirecta, por ejemplo, con JNI). Aunque esto puede ser ventaja.
- Es un lenguaje que tiene una curva de aprendizaje más lenta que otros. Es por eso que aquí se toma un enfoque diferente, para acelerar el aprendizaje.

Independencia de la plataforma

- Como hemos visto, es una de las principales ventajas de Java.
- Permite que un mismo programa se ejecute en diferentes plataformas sin tener que cambiarlo.
- Se debe al procedimiento que se sigue para la compilación y ejecución de un programa Java.

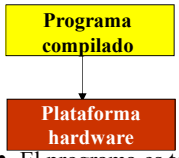
Ejecución de un programa escrito en un lenguaje tradicional



significa "se ejecuta sobre"

- La plataforma hardware es el conjunto del hardware y del sistema operativo.
- El programa está compilado específicamente para una determinada plataforma hardware, por lo tanto es ejecutable directamente sobre la plataforma.

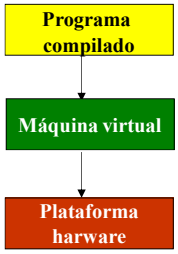
Desventajas de este enfoque



significa "se ejecuta sobre"

- El programa es totalmente dependiente de la plataforma hardware.
- Si escribimos un programa en una PC y lo queremos ejecutar otra plataforma (Linux, celular) no funciona.
- Esto es malo, pues cada vez las plataformas son más diversas (celulares, tablet PC, handhelds, etc.)

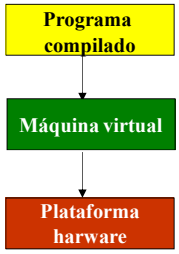
Ejecución de Java



significa "se ejecuta sobre"

- En Java el programa compilado no es ejecutado directamente por la plataforma hardware.
- Hay un programa, llamado Máquina virtual (JVM) que se encarga de ejecutar el programa compilado.
- Es decir, se añade una capa de indirección.

"Todos los problemas en Computación pueden resolverse con otro nivel de indirección"

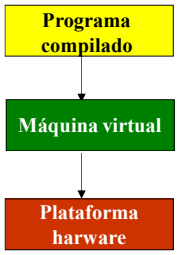


significa "se ejecuta sobre"

- Este es un aforismo famoso de Butler W. Lampson (de Microsoft) que es fundamentalmente cierto.
- Aquí, como veremos, el nivel de indirección de la máquina virtual nos permite aislarnos de la plataforma hardware.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

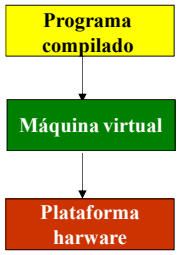
Hay una máquina virtual para cada plataforma hardware



significa "se ejecuta sobre"

- El programa compilado está en un lenguaje que entiende la máquina virtual, llamado "bytecodes".
- **Hay una máquina virtual para casi cada plataforma** y todas las máquinas virtuales entienden el mismo lenguaje de "bytecodes".

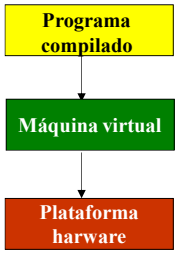
¿Qué ventajas da eso?



significa "se ejecuta sobre"

- **El programa se convierte en independiente de la plataforma.**
- El programa compilado en bytecodes se puede ejecutar en cualquier plataforma para la cual haya una máquina virtual (prácticamente todas)

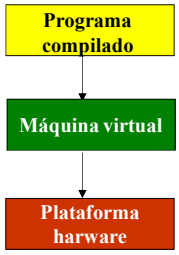
Esto es útil



significa "se ejecuta sobre"

- Ahora puedo compilar mi programa Java en una PC y copiarlo en una Mac, un Linux o una handheld.
- El programa funcionará sin cambios, pues todas estas plataformas tienen una máquina virtual que entiende los bytecodes.
- Ahora resulta tan fácil programar en todas las plataformas como en una plataformas.

Otra ventaja

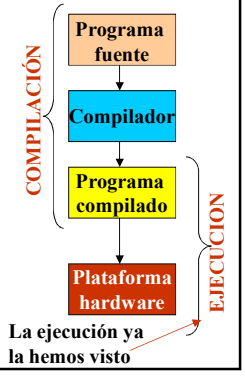


significa "se ejecuta sobre"

- Como ahora la ejecución está controlada por la máquina virtual, pueden detectarse algunos temas de seguridad.
- La máquina virtual no dejará ejecutar código malicioso.
- Además provee otros servicios útiles, como multithreading, recolección de basura, etc.
- Los programas son más seguros y de mejor calidad. Están mejor controlados

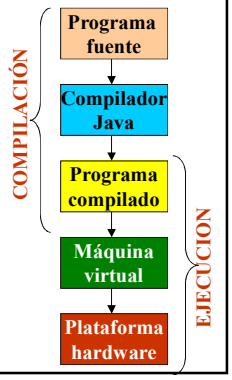
Eso en cuanto la ejecución, pero ¿qué hay de la compilación?

- En un lenguaje convencional, el compilador traduce el código fuente a un programa compilado específicamente para la plataforma.



Esto cambia en Java

- Ahora, el programa compilado no se ejecuta sobre la plataforma sino sobre un programa llamado máquina virtual.
- El compilador genera un programa en bytecodes, que puede ejecutarse en cualquier plataforma.



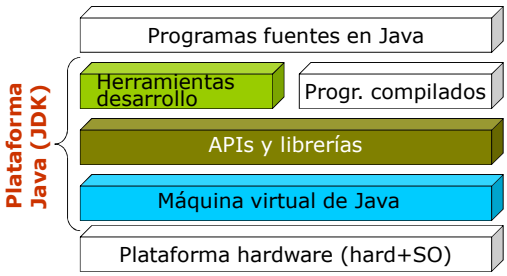
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

¿Qué es Java?

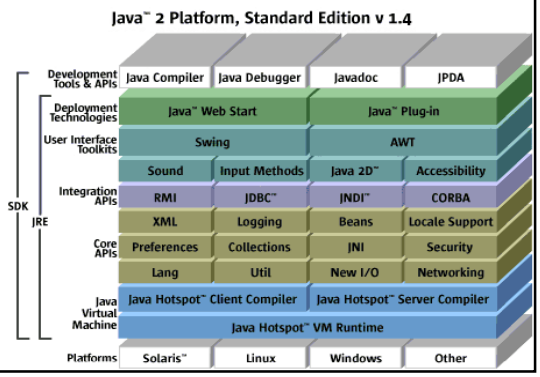
- Es un lenguaje de programación orientada a objetos desarrollado por Sun Microsystems.
- Es una plataforma para desarrollo de programas que incluye varios componentes que se interconectan entre sí.

La plataforma Java

- Java no es sólo un lenguaje, sino una plataforma (en color en la figura).



La plataforma Java con más detalle



Programa del tema 1

Instalación de la plataforma Java

- La plataforma Java se necesita:
  - Para desarrollar programas Java.
  - Para ejecutarlos (recordemos la máquina virtual).
- Se da en dos versiones:
  - **JRE.** Es el subconjunto mínimo para ejecutar programas Java.
  - **JDK** (también llamado SDK). Es la plataforma completa, que permite ejecutar y desarrollar programas Java. Incluye el JRE.

La plataforma Java es gratuita

- Algunas máquinas la tienen instalada de fábrica.
- En todo caso, puede descargarse del sitio oficial de Java (http://java.sun.com)

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Instalando y configurando la plataforma Java

- Deben seguirse tres pasos:
- 1. Descargar la plataforma de Internet.
- 2. Instalar la plataforma.
- 3. Configurar el sistema para usar la plataforma.

Instalando y configurando la plataforma Java

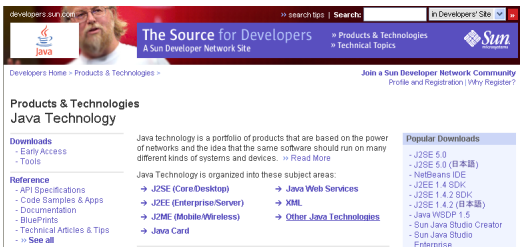
- Deben seguirse tres pasos:
- 1. Descargar la plataforma de Internet.
- 2. Instalar la plataforma.
- 3. Configurar el sistema para usar la plataforma.

Nota preliminar

- Para instalar Java desde Windows, la cuenta debe ser "Administrador del sistema".
- Si usted tiene ese tipo de cuenta, no hay problema.
- Si no lo tiene, un usuario que lo tenga lo tiene que autorizar desde **Inicio|Panel de control|Usuarios.**

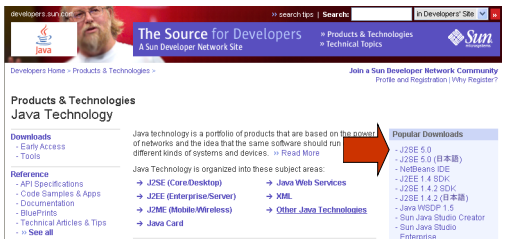
Descargando la plataforma de Internet (1)

- Primero, se accede al sitio oficial de Java, **http://java.sun.com**



Descargando la plataforma de Internet (2)

- Debajo de Popular Downloads se elige la última versión J2SE que no sea beta.



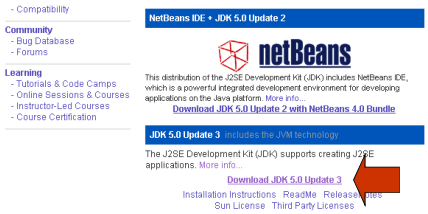
¿Por qué J2SE?

- **J**ava **2** **S**tandard **E**dition (es decir, lo que conocemos por el lenguaje Java con las librerías más comunes)

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Descargando la plataforma de Internet (3)

- Se elige el último JDK que se incluye en la página que aparece.



¿Por qué JDK?

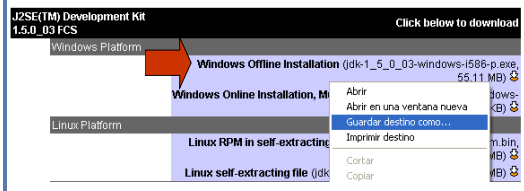
- Recordemos, la plataforma Java se da en dos versiones:
  - **JRE.** Es el subconjunto mínimo para ejecutar programas Java.
  - **JDK (también llamado SDK).** Es la plataforma completa, que permite ejecutar y desarrollar programas Java. Incluye el JRE.
- Como queremos ejecutar y desarrollar (programar) elegimos este último.

Descargando la plataforma de Internet (4)

- Se acepta el interminable contrato de licencia.

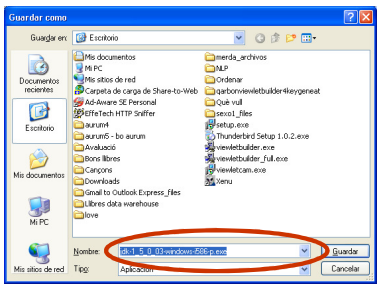
Descargando la plataforma de Internet (5)

- En nuestro caso, elegimos la versión para Windows (el tipo de instalación fuera de línea: "offline installation").
- Se hace clic derecho y se elige "Guardar destino como".



Descargando la plataforma de Internet (6)

- Aparece una descarga que guardamos en nuestra computadora.



Descargando la plataforma de Internet (7)

- Esperamos mientras la plataforma Java se descarga

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Instalando y configurando la plataforma Java

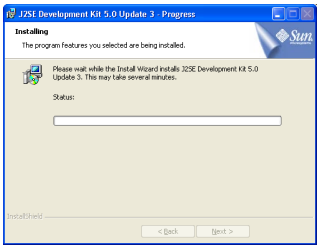
- Deben seguirse tres pasos:
- 1. Descargar la plataforma de Internet.
- 2. Instalar la plataforma.
- 3. Configurar el sistema para usar la plataforma.

Instalar la plataforma (1)

- Como resultado de la descarga, aparecerá un programa de instalación, sobre el cual se debe hacer doble clic.

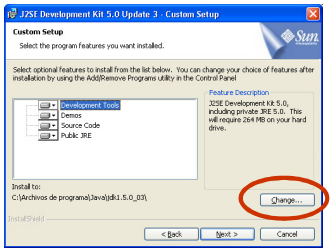
Instalar la plataforma (2)

- El programa de instalación es típico. Hay que hacer clic en "Next" cada vez que se presenta una opción. Hay que aceptar un contrato de licencia.



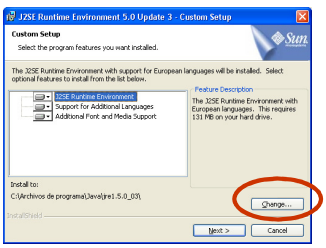
Instalar la plataforma (3)

- En esta ventana se puede cambiar el directorio de instalación del JDK, haciendo clic en "Change". Como veremos, es importante saber cuál es este directorio de instalación del JDK.



Instalar la plataforma (4)

- Antes de finalizar la instalación del JDK, se produce la instalación del JRE (recordemos que el JRE es parte del JDK). También el JRE se puede cambiar de directorio de instalación.



Instalando y configurando la plataforma Java

- Deben seguirse tres pasos:
- 1. Descargar la plataforma de Internet.
- 2. Instalar la plataforma.
- 3. Configurar el sistema para usar la plataforma.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

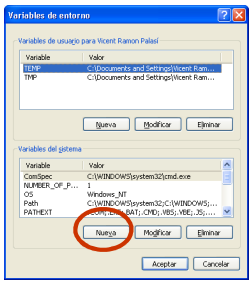
Configurar el sistema para usar la plataforma

- Hay que definir las siguientes variables de entorno para configurar el sistema para usar la plataforma Java. Es bueno reiniciar.

Nombre	Valor
JAVA_HOME	*directorioDeInstalacionDelJDK*
PATH	%PATH%;%JAVA_HOME%\bin

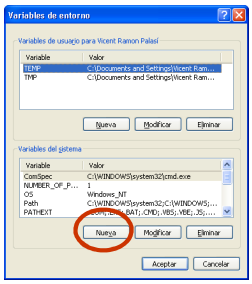
Para definir una variable de entorno

- Se accede a **Panel de Control|Sistema|Opciones avanzadas|Variables de entorno|Nueva**



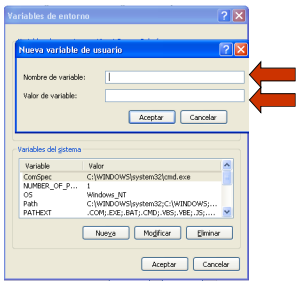
Para definir una variable de entorno

- Se accede a **Panel de Control|Sistema|Opciones avanzadas|Variables de entorno|Nueva**



En la ventana que aparece

- Se define el nombre y el valor de la variable de entorno.



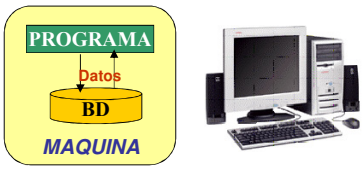
Programa del tema 1

Programación para el Web

- Este es un curso de programación para el Web.
- Es muy diferente a la programación monolítica, donde los programas se ejecutan en una sola máquina.
- También es diferente de una programación en una red tradicional (LAN o similares).
- ¿Cuáles son las diferencias?

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Programación monolítica: Arquitectura

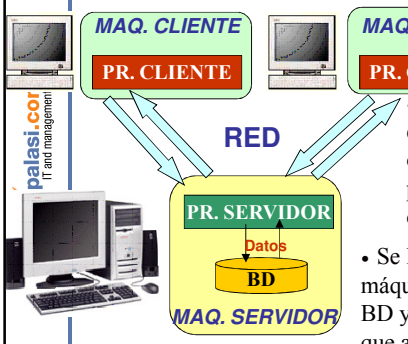


• Cada programa se ejecuta en una sola máquina

Esto no suele ser suficiente

- La mayoría de las veces deberemos ejecutar el programa en varias máquinas diferentes (pero accediendo a la misma base de datos).
- Puede haber tantas máquinas como se quiera.
- Para simplificar, dibujaremos dos máquinas, aunque todo lo que explicaremos será aplicable a más de dos.

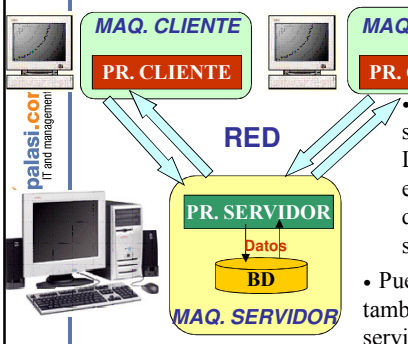
Programación en red tradicional: Arquitectura cliente-servidor



• Ahora la BD está en una máquina y las otras acceden a ella para recuperar los datos.

• Se llama **servidor** a la máquina que tiene la BD y **clientes** a las otras que acceden a ella.

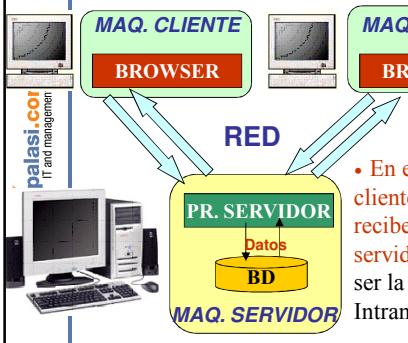
Tanto el servidor como el cliente pueden tener código



• Ahora el programa se ha partido en dos. La parte que está en el cliente y la parte que está en el servidor.

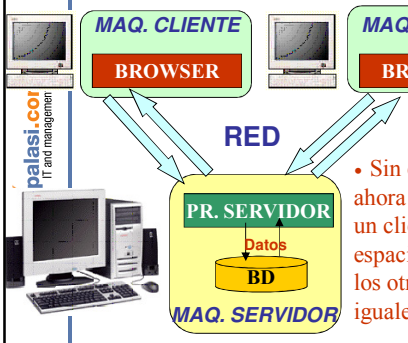
• Pueden llamarse también cliente y servidor respectivamente.

Programación para el Web



• En ella, el programa cliente es un "browser" y recibe de HTML del servidor. La red puede ser la Internet o una Intranet.

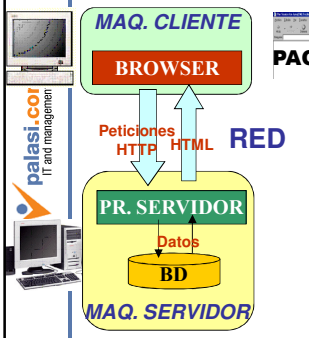
Los clientes pueden ser muchos



• Sin embargo, a partir de ahora sólo dibujaremos un cliente por falta de espacio y porque todos los otros clientes son iguales

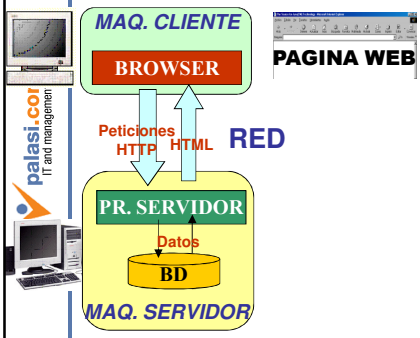
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

El servidor recibe peticiones HTTP y genera HTML



- El cliente es el navegador de Internet ("browser"). El servidor recibe peticiones HTTP y devuelve HTML que después el browser traduce en una página Web, que es la que ve el cliente.

Servidor genera HTML que crea la página Web en el "browser



En resumen

- En la programación para el Web, el cliente no tenemos que programarlo, sino que ya viene dado: es el navegador ("browser").
- Sólo tenemos que programar al servidor, pero éste debe ser compatible con el cliente (el "browser"), que ya está definido.
- Para que sea compatible, debe hablar el lenguaje que el browser entiende, el cual es HTML.
- Esto nos obliga a conocer HTML.

Programa del tema 1

- 1. Introducción al lenguaje Java.
- 2. Instalación de la plataforma Java.
- 3. Introducción a la programación Web.
- 4. Fundamentos de HTML.
- 5. Introducción a Eclipse.
- 6. Introducción a Tomcat.
- 7. Introducción a JSP.
- 8. Variables. Expresiones aritméticas y lógicas.
- 9. Sentencias algorítmicas: condicionales, ciclos.
- 10. Arreglos.

HTML

- Como hemos visto, en una aplicación Web el servidor envía HTML al "browser" del cliente.
- El browser usa este HTML para dibujar la página.
- Por ello, si queremos programar para el Web, debemos tener nociones de HTML.

Este no es un curso sobre HTML

- Por ello, no estudiaremos este lenguaje con profundidad, ni siquiera de forma general.
- Sólo explicaremos lo mínimo de HTML que hace falta para comprender la programación para el Web con Java.
- Esto hará que nuestras páginas no serán muy estéticas pero se verá el mecanismo que permite funcionar una aplicación por Internet. Todo lo demás es diseño.
- Si quieren crear una página Web con un bonito diseño, les recomendamos que tomen un curso de diseño de páginas Web.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

HTML

- HyperText Markup Language (Lenguaje de marcado de hipertexto). Es el lenguaje en el cual se diseñan las páginas Web. El código escrito en este lenguaje se le llama hipertexto.
- Creado por Tim Berners-Lee en el laboratorio de física nuclear CERN en Ginebra (Suiza) en 1989.
- Es el HTML el que creó el Web. Antes ya había Internet pero no era tan fácil de manejar.
- HTML es un lenguaje que sólo define el diseño estático de las páginas. Para definir su comportamiento dinámico, se necesita otro lenguaje. Por ejemplo, Java (como ya veremos).

HTML

- Es un lenguaje que se escribe en archivos de texto (llamados documentos HTML) y tiene marcas o etiquetas ("tags") que indican cuales son los elementos de la página Web .
- Un elemento de la página Web se escriben entre dos etiquetas: una de inicio y una de final. La de inicio se escribe <*ELEMENTO*> y la del final se escribe </*ELEMENTO*>. Se pueden leer como "principio de Elemento" y "fin de Elemento".
- Así, por ejemplo, un documento HTML comienza con **<HTML>** ("principio de HTML") y acaba con **</HTML>** ("fin de HTML").

Estructura de un documento HTML sencillo

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
 HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <TITLE>
            Título de la página
        </TITLE>
    </HEAD>
    <BODY>
        Diseño de la página
    </BODY>
</HTML>
```

Un documento HTML tiene tres partes (1)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <TITLE>
            Título de la págin
        </TITLE>
    </HEAD>
    <BODY>
        Diseño de la página
    </BODY>
</HTML>
```

Primera parte. Una línea indicando la versión de HTML que utilizamos

En nuestro caso, siempre usaremos la misma línea pues siempre usaremos HTML 4.0 Transicional

Un documento HTML tiene tres partes (2)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <TITLE>
            Título de la págin
        </TITLE>
    </HEAD>
    <BODY>
        Diseño de la página
    </BODY>
</HTML>
```

Segunda parte. La cabecera (HEAD) que contiene información general del documento como, por ejemplo, el título.

Un documento HTML tiene tres partes (3)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <TITLE>
            Título de la págin
        </TITLE>
    </HEAD>
    <BODY>
        Diseño de la página
    </BODY>
</HTML>
```

Tercera parte. El cuerpo (BODY) que contiene el diseño de la página Web.

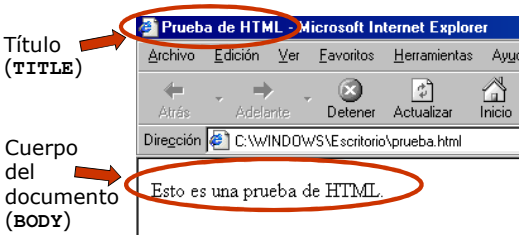
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejemplo: un documento HTML de prueba

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <TITLE>
           Prueba de HTML
        </TITLE>
    </HEAD>
    <BODY>
       Esto es una prueba de HTML.
    </BODY>
</HTML>
```

Como se ve la página de prueba

- Una vez abrimos con un navegador el documento HTML de prueba que acabamos de ver, lo que aparece es esto.



Algunas etiquetas comunes para el cuerpo del documento

- `<P> Aquí va un párrafo </P>` En medio de estas dos etiquetas va un párrafo. La segunda parte es opcional. Así:

```
<P> Este es el primer parrafo.
<P> Este es el segundo parrafo.
```

- `<H1> Aquí va un título </H1>` En medio de estas dos etiquetas va un título (es título porque tiene letra de mayor tamaño).
- `<H2> Aquí va un título más pequeño </H2>`
- `<H3> Aquí un título aún más pequeño </H3>`
- Títulos más pequeños se obtienen con `<H4>` hasta `<H6>`

Algunas etiquetas comunes para el cuerpo del documento

- `<EM> Aquí va letra enfatizada </EM>` Normalmente la letra enfatizada va en cursiva.

```
<EM>Hola</EM> es una palabra de
saludo
```

- `<STRONG> Aquí va letra muy enfatizada </STRONG>` Normalmente la letra muy enfatizada va en negrita.

```
La programación requiere
<STRONG>practica</STRONG>
```

- `<A HREF="URLdeHipervinculo">Texto de hipervínculo</A>` Especifica hipervínculos (enlaces) a otros documentos HTML.

```
<A HREF="http://www.google.com">
Buscador Google</A>
```

Algunos caracteres útiles en español

- En HTML los caracteres que no son números o letras del alfabeto inglés deben indicarse de una forma especial. Así:
  - `á` es la a acentuada (á).
  - `é` es la e acentuada (é).
  - `í` es la i acentuada (í).
  - `ó` es la o acentuada (ó).
  - `ú` es la u acentuada (ú).
  - `ñ` es la letra eñe (ñ).
  - `¿` es abrir interrogación (¿)
  - `¡` es abrir exclamación (¡)
  - ` ` es un espacio en blanco ( )

Nota

- Los saltos de línea del documento HTML no se reflejan en la página Web que éste produce.
- Para separar las líneas en la página Web, las separaremos con **
** (que salta de línea) o bien con **<P>** (que, como hemos visto, comienza un nuevo párrafo dejando una línea en blanco antes de él).
- Si no se ponen estas marcas, no se salta de línea.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejemplo: un segundo documento HTML de prueba

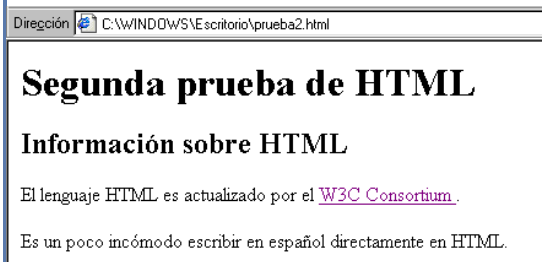
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
<HEAD><TITLE> Segunda prueba de
   HTML</TITLE></HEAD>
<BODY>
   <H1>Segunda prueba de HTML </H1>
   <H2>Informaci&oacute;n sobre HTML</H2>
   <P> El lenguaje HTML es actualizado por
   el <A HREF="http://www.w3.org/">W3C
   Consortium</A>.
   <P> Es un poco inc&oacutemodo escribir en
   espa&ntilde;ol directamente en HTML.
</BODY>
</HTML>
```

Pregunta

- ¿Qué es lo que aparecería en página cuando abrimos este documento en un navegador?
- Escriban este documento y compruébenlo por ustedes mismos.

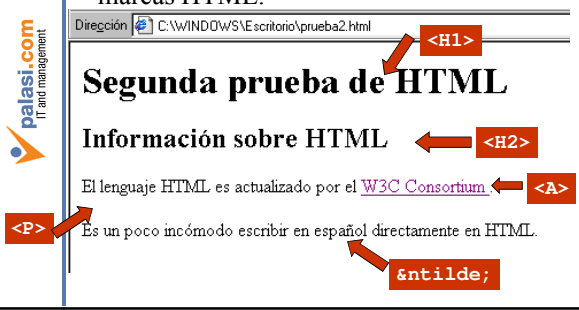
Respuesta

- Lo que aparece es esto.



Algunas marcas HTML y cómo se reflejan en la página

- Observen qué apariencia tienen algunas marcas HTML.



Ejercicio

- Creen un documento HTML que se corresponda a una página como ésta. (Pueden cambiar la dirección preferida o añadir otros elementos, si lo desean)

Solución

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
   <HEAD>
      <TITLE>Una p&aacute;gina Web sencilla</TITLE>
   <HEAD>
   <BODY>
      <H1>Una p&aacute;gina Web sencilla</H1>
      <P>Aqu&iacute; se practica el uso de las
         <STRONG>negritas</STRONG> y <EM>la
         cursiva</EM></P>
      <P>Mi direcci&oacute;n preferida en Internet
      es <A href="www.google.com">Google</A>.</P>
   </BODY>
</HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

¿Cómo introducimos datos en una aplicación Web?

- Hay veces en que queremos introducir datos en una página Web: por ejemplo, términos de búsqueda, contraseñas, nombres, etc.
- Si navegamos por Internet, estamos acostumbrados a unos rectángulos donde se escribe y unos botones para enviar la información. Ejemplo:

- ¿Cómo se hace esto en HTML?

Formularios

- Una página Web donde hay uno (o varios) controles de introducción de datos y un botón para enviarlos al servidor **se dice que contiene un formulario.**
- Nota: además de cuadros de texto pueden haber otros controles de introducción, como botones de radio, etc.
- **Los formularios son la forma que tiene HTML para que el usuario introduzca datos.**

Formularios

- Dicho de otra forma, un formulario es un fragmento de una página Web que sirve para introducir datos y enviarlos al servidor.
- Consta de controles de introducción de datos (sobre todo, cuadros de texto) para introducir los datos y (uno o varios) botones para enviarlos al servidor.
- Llamaremos elementos del formulario tanto a los controles de introducción de datos como los botones.

Código HTML para un formulario

```
<FORM>
    código HTML del primer elemento
    código HTML del segundo elemento
    . . .
    código HTML del último elemento
</FORM>
```

Importante

- Para los controles de introducción de datos, distinguiremos entre:
  - **El nombre.** Es el nombre de ese control
  - **El texto.** Que son las palabras que se ven por página en ese control.
  - **El valor**. Que son los datos que se envían al servidor, para ser procesados por el programa.
- Estos tres pueden ser el mismo o diferentes.

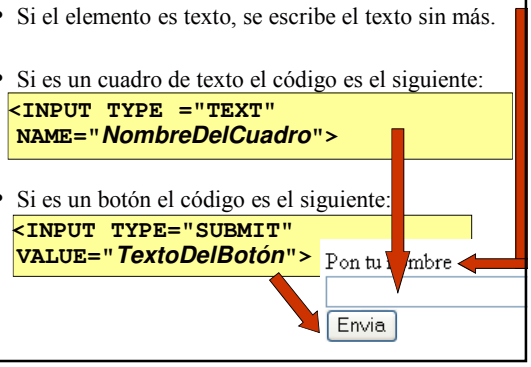
Código HTML para un elemento de formulario

- Si el elemento es texto, se escribe el texto sin más.
- Si es un cuadro de texto el código es el siguiente:

```
<INPUT TYPE ="TEXT"
NAME="NombreDelCuadro">
```

- Si es un botón el código es el siguiente:

```
<INPUT TYPE="SUBMIT"
VALUE="TextoDelBotón">
```



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Formulario: ejemplo anterior

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
   Transitional//EN">
<HTML>
  <HEAD>
    <TITLE>Cuadros de texto</TITLE>
  <HEAD>
  <BODY>
    <FORM>
    Pon tu nombre<BR>
    <INPUT TYPE="TEXT" NAME="nombre"><BR>
    <INPUT TYPE="SUBMIT" VALUE="Envia">
    </FORM>
  </BODY>
</HTML>
```

Pon tu nombre

Envia

Casillas de verificación

- Son casillas que indican opciones que pueden seleccionarse o no. Cada casilla puede seleccionarse o no de forma independiente de las otras.

¿Qué ingredientes quiere para su pizza?

Pepperoni ☐
Cebolla ☐
Salchicha ☑

- Se obtienen con el siguiente código:

```
<INPUT TYPE ="CHECKBOX"
NAME="NombreDeLaCasilla">
```

Si queremos que la casilla salga seleccionada por defecto, se coloca la palabra **CHECKED** antes del **>**

Casillas de verificación: ejemplo anterior

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
<HEAD><TITLE>Casillas de
  verificacion</TITLE><HEAD>
<BODY>
  <FORM>
    <P>¿Qué ingredientes quiere para su
  pizza?      <P>Pepperoni <INPUT
  TYPE="CHECKBOX"  NAME="pepperoni"><BR>

  Cebolla <INPUT TYPE="CHECKBOX"
  NAME="cebolla"> <BR>

  Salchicha <INPUT TYPE="CHECKBOX"
  NAME="salchicha" CHECKED>
      </FORM>
   </BODY></HTML>
```

Botones de radio

- Como las casillas de verificación, son casillas en las que se pueden seleccionar opciones. Pero, a diferencia de éstas, se dan en grupos y sólo se puede seleccionar un botón de cada grupo.

¿Cuál es su método de pago?

Mastercard ○
Visa ◉
American Express ○

-
```
<INPUT TYPE="RADIO" NAME="
NombreGrupo" VALUE="NombreBotón">
```

Si se quiere que aparezca el botón seleccionado por defecto, se pone **CHECKED** antes de **>**.

Botones de radio: ejemplo anterior

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
<HEAD><TITLE>Botones de radio</TITLE><HEAD>
<BODY>
  <FORM>
    <P>¿Cuál es su método de pago?
    <P>Mastercard <INPUT TYPE="RADIO"
  NAME="Tarjeta" VALUE="Mastercard"><BR>

  Visa <INPUT TYPE="RADIO" NAME="Tarjeta"
  VALUE="Visa" CHECKED> <BR>

  American Express <INPUT TYPE="RADIO"
  NAME="Tarjeta" VALUE="American Express">
      </FORM>
   </BODY></HTML>
```

Listas

- Controles de introducción de datos en los que se elige un valor de una lista predeterminada.

LISTA SIMPLE

¿Cuál es su método de pago?

Visa
Mastercard
Visa
American Express

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Código HTML de listas

```
<SELECT NAME="NombreLista">

<OPTION VALUE="ValorOpción1">TextoOpción1
<OPTION VALUE="ValorOpción2">TextoOpción2
...
<OPTION VALUE="ValorOpciónn">TextoOpciónn

</SELECT>
```

- Para preseleccionar una opción, hay que poner **SELECTED** después del > que cierra el **OPTION**

Listas: ejemplo anterior

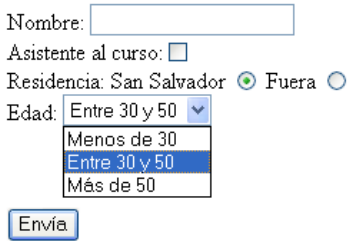
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
   Transitional//EN">
<HTML>
<HEAD><TITLE>Listas</TITLE><HEAD>
<BODY>
 <FORM>
<P>¿Cuál es su método de pago?
<P> <SELECT NAME="Tarjeta">
     <OPTION VALUE="Mastercard">Mastercard
     <OPTION VALUE="Visa" SELECTED>Visa
     <OPTION VALUE="American Express">American
   Express
   </SELECT>
    </FORM>
 </BODY></HTML>
```

Algunos detalles estéticos

- Como hemos visto, las líneas las separaremos con **
** (que salta de línea) o bien con **<P>** (que comienza un nuevo párrafo dejando una línea en blanco antes de él).
- Si queremos centrar uno o varios elementos, los encerraremos entre **<CENTER>** y **</CENTER>**

Ejercicio

- Crear un archivo llamado **saludar.html** que produzca este formulario



Solución (1)

```
<!DOCTYPE ...><HTML><HEAD>
   <TITLE>Formulario</TITLE><HEAD>
<BODY><FORM>
<P>Nombre: <INPUT TYPE="TEXT" NAME="nombre"><BR>
Asistente al curso:<INPUT TYPE="CHECKBOX"
   NAME="asistente"><BR>
Residencia: San Salvador <INPUT TYPE="RADIO"
   NAME="residencia" VALUE="San Salvador" CHECKED>
Fuera <INPUT TYPE="RADIO" NAME="residencia"
   VALUE="fuera"><BR>
```

Solución (2)

```
Edad:
<SELECT NAME="edad">
 <OPTION VALUE="menos de 30">Menos de 30
 <OPTION VALUE="entre 30 y 50" SELECTED>Entre 30
   y 50
 <OPTION VALUE="más de 50">Más de 50
</SELECT><BR>
<INPUT TYPE="SUBMIT" VALUE="Env&iacute;a">
</FORM>
</BODY>
</HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ahora ya sabemos como crear un formulario

- Sólo me falta saber cómo enviar la información del formulario al servidor para que la procese.
- Por ejemplo en Google, cuando busco algo, envío el término de búsqueda al servidor para que busque.

Esto lo veremos más adelante

- Sin embargo, adelantamos que, en vez de comenzar el código HTML de un formulario con **<FORM>**, como hemos visto.
- Hay que comenzarlo con

```
<FORM ACTION="URLDelPrograma">
```

donde ***URLDelPrograma*** es el URL del programa al que queremos enviar la información.

Ejercicio

- Queremos hacer una aplicación para el Web que sume dos números.
- La aplicación pedirá dos números mediante un formulario y dará la suma.
- Creen este formulario en HTML.

Programa del tema 1

- 1. Introducción al lenguaje Java.
- 2. Instalación de la plataforma Java.
- 3. Introducción a la programación Web.
- 4. Fundamentos de HTML.
- 5. Introducción a Eclipse.
- 6. Introducción a Tomcat.
- 7. Introducción a JSP.
- 8. Variables. Expresiones aritméticas y lógicas.
- 9. Sentencias algorítmicas: condicionales, ciclos.
- 10. Arreglos.

5. Introducción a Eclipse

- 5.1. Entornos de desarrollo.
- 5.2. Descargar Eclipse y Lomboz.
- 5.3. Instalar Eclipse y Lomboz.
- 5.4. Algunas tareas básicas en Eclipse.

5. Introducción a Eclipse

- 5.1. Entornos de desarrollo.
- 5.2. Descargar Eclipse y Lomboz.
- 5.3. Instalar Eclipse y Lomboz.
- 5.4. Algunas tareas básicas en Eclipse.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Java no necesita un entorno de desarrollo

- Todo se puede hacer con el JDK y con el bloc de notas.
- También se puede ir de San Salvador a Santa Ana a pie.

Entornos de desarrollo (IDEs)

- (**I**ntegrated **D**evelopment **E**nvironment) Aplicación integrada que automatiza la mayoría de tareas de desarrollo en un entorno gráfico amigable para el usuario.
- Incluye **como mínimo** un editor que permite editar, compilar y depurar dentro del mismo programa.
- Ejemplos de IDEs: Eclipse, Forté for Java, Netbeans (Sun), JBuilder (Borland), VisualAge for Java (IBM), JCreator, jEdit, y un largo etcétera.

IDEs respecto a programas del JDK

IDEs	JDK
Un único programa para compilar, depurar y ejecutar.	Varios programas independientes
Entorno gráfico.	Línea de comandos.
Más amigable al usuario	Menos amigable al usuario
Adecuado para grandes sistemas.	Incómodo para grandes sistemas.
Gratuitos o de pago.	Gratuito.
No estándar	Estándar
No disponible universalmente	Universalmente disponible.

Nosotros usaremos un IDE

- Debemos estar preparados para programar programas reales (de tamaño empresarial) que no es práctico programar sin IDE.
- Un IDE nos permite ser más productivos y mejora la calidad de nuestra programación.
- Pero, ¿qué IDE usaremos? Hay muchos.
- Usaremos Eclipse pues es el más extendido.
- ¿Las pruebas?

Oreilly.net

- Encuesta sobre el IDE más usado en el sitio de O'Reilly (editorial que vende muchos libros sobre Java)

POLL RESULTS

PollMaster: Chris DiBona

Which Java IDE do you use?

Eclipse.	35.6%
NetBeans.	18.7%
Emacs.	5.8%
IntelliJ.	13.9%
Other.	10.6%
Java?	15.1%

Manageability.org

- Encuesta sobre el IDE preferido en un sitio Web sobre proyectos Java de código abierto.

What is your favorite Java IDE?

Current result of this poll

CodeGuide	1% (13)
Eclipse	54% (438)
IDEA	22% (186)
JBuilder	5% (45)
JDeveloper	3% (28)
NetBeans	7% (58)
Weblogic WorkShop	0% (7)
Other (i.e. vi, Emacs, JEdit etc.)	4% (35)

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

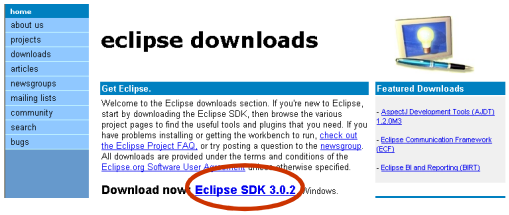
Eclipse: el IDE de Java más extendido

- IDE para varios lenguajes: Java, pero también Perl, PHP, Python, C/C++, Ruby e incluso .NET (a través de Mono) o edición hexadecimal.
- Iniciado por IBM, pero ya desde hace tiempo es de código abierto. Últimamente se han unido a él BEA, Sybase e incluso Borland (que vende "JBuilder", el cual están intentando basarlo en Eclipse).
- Rápido, eficiente y robusto. Muy completo: tiene cientos de plugins **www.eclipse-plugins.2y.net**, **www.eclipseplugincentral.com**
- Uno de estos plugins es Lomboz, para desarrollo Java para el Web, el cual utilizaremos.

5. Introducción a Eclipse

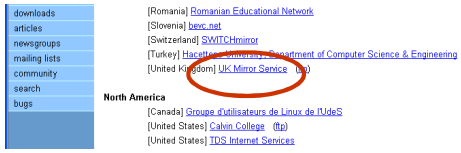
Descargar Eclipse

- En la página principal de Eclipse (www.eclipse.org) vamos a Downloads. En la página que aparece elegimos el enlace en grande que aparece al lado de la palabra "Download now" y al lado de Windows.



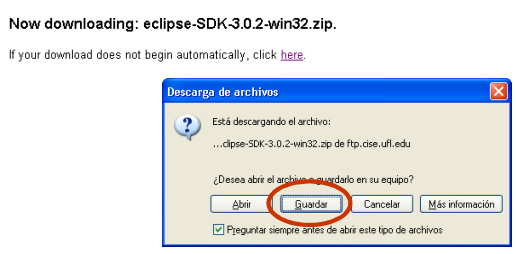
Aparecerá una serie de "mirrors"

- Se hace clic sobre el mirror desde el que queramos descargar el programa.



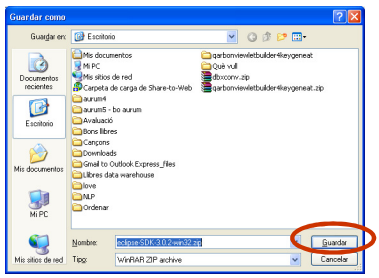
Aparecerá una ventana

- A los pocos segundos, aparecerá sobre esta ventana una ventana emergente para descargar el programa. Debemos hacer clic en **Guardar**



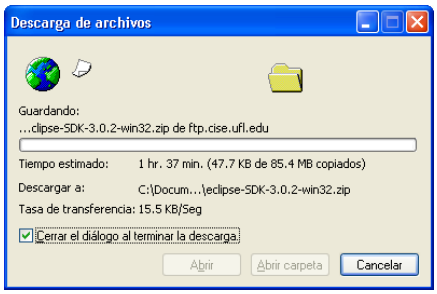
Se elige la ubicación para guardar el programa

- Y se hace clic en **Guardar**



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Se descargará Eclipse

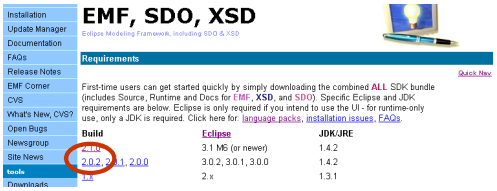


Ahora descargaremos EMF

- Es un "framework" de modelado y generación de código que se integra en Eclipse. Lomboz lo necesita para ejecutarse. Accederemos a su página principal con el URL www.eclipse.org/emf/ y haremos clic en Downloads.

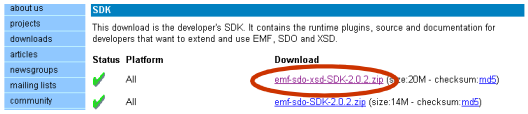
Descargando EMF

- Elegimos la versión adecuada para la versión de Eclipse (en nuestro caso, Eclipse es 3.0.2, por lo que se elige EMF 2.0.2)



Ahora descargaremos EMF

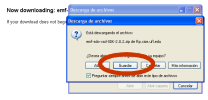
- Se elige el paquete completo, que incluye EMF, SDO y XSD. Se descarga este paquete.



Se elige un mirror

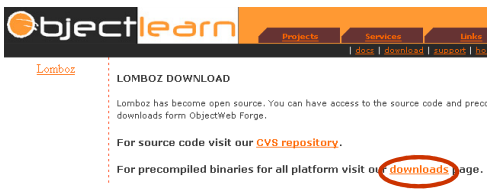
- De la lista de mirrors disponibles.

- Comenzará la descarga.



Descargar Lomboz

- Se va a la página de Lomboz www.objectlearn.com y se hace clic en download y otra vez en downloads.

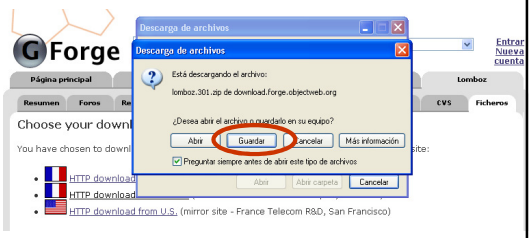


Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Aparecerá una página de descargas

- Se elige el apartado correspondiente a la versión de Eclipse que tenemos (en nuestro caso 3.0.2).
- Se elige el enlace de título corto (se trata de la versión estable), lo otro son compilaciones no estables.

En la página que aparece hacemos clic en un enlace y se descargará Lomboz



5. Introducción a Eclipse

- 5.1. Entornos de desarrollo.
- 5.2. Descargar Eclipse y Lomboz.
- 5.3. Instalar Eclipse y Lomboz.
- 5.4. Algunas tareas básicas en Eclipse.

Instalar Eclipse

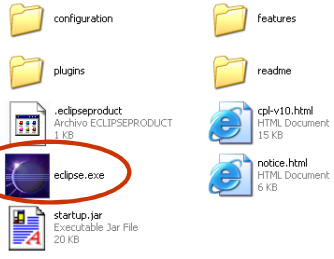
- Es fácil. Descomprimimos el archivo ZIP que hemos obtenido con la descarga. Se generará un directorio eclipse que podemos colocar donde queramos (por ejemplo en `C:\Archivos de programa`)

Nota importante

- El directorio donde hemos colocado Eclipse debe poder ser escrito por todos los usuarios que utilizarán dicho programa.
- Por lo tanto, si hay varios usuarios en la misma máquina que usan la misma instalación de Eclipse, deberemos compartir la carpeta entre ellos.

Dentro de la carpeta `eclipse` el ejecutable se llama `eclipse.exe`

- Podemos ejecutarlo directamente o, mejor, hacer un acceso directo para él.

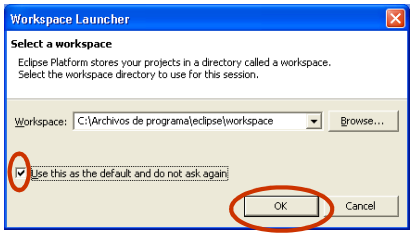


Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejecutemos este ejecutable.

- Se iniciará Eclipse.

En unos pocos momentos

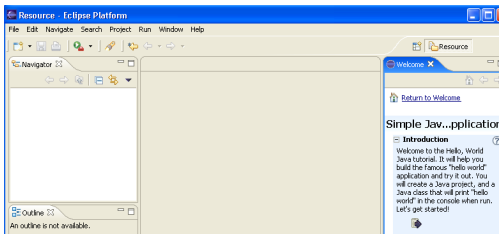


- Aparecerá un cuadro de diálogo. Seleccionamos la casilla de verificación y hacemos clic en OK

Aparece una pantalla de bienvenida

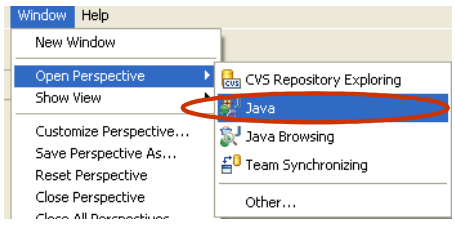
- Hacemos clic en la flecha en la esquina superior derecha.

Aparece el banco de trabajo (workbench)



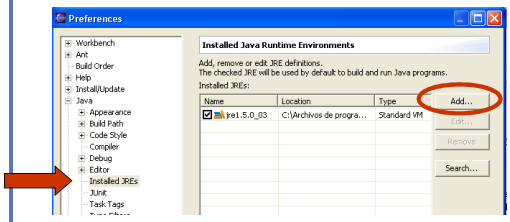
Cambiamos a perspectiva Java

- Usando **Window|Open perspective|Java**



Hacemos que Eclipse apunte al JDK que hemos instalado (1)

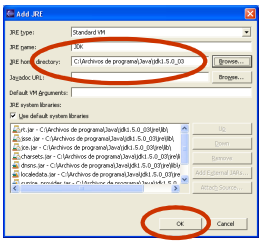
- Vamos a **Windows|Preferences|Java|Installed JREs** y hacemos clic en **Add**.



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

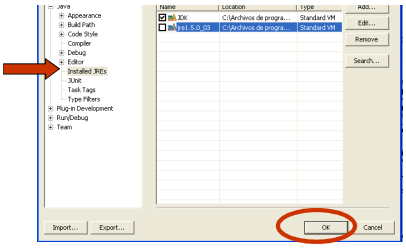
Hacemos que Eclipse apunte al JDK que hemos instalado (2)

- Se le da un nombre arbitrario y se coloca la ruta del JDK que hemos instalado. Después se hace clic en **OK**.



Hacemos que Eclipse apunte al JDK que hemos instalado (3)

- Se activa la casilla de verificación a la izquierda del JDK que acabamos de especificar y se hace clic en **OK**.

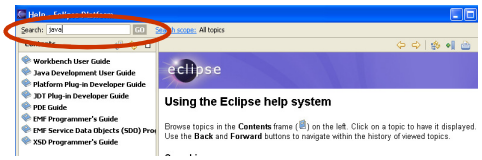


¿Por qué esto último?

- Por defecto, Eclipse busca un JRE de Java para ejecutarse.
- Sin embargo, para desarrollar y compilar programas se requiere un JDK. Por ello, debemos indicar a Eclipse el JDK que usaremos.

Es bueno también

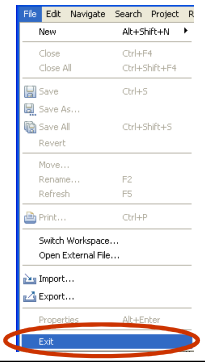
- Entrar a **Help|Help contents** y escribir una palabra en el cuadro de **Search**.



- Esto indexará la ayuda. Es largo pero sólo se hace una vez.

Salimos del programa

- Con **File|Exit**



Una observación

- Eclipse tarda bastante en iniciarse.
- Mientras trabajemos, lo mejor es mantenerlo abierto, en vez de entrar y salir de él continuamente.
- Sin embargo, aquí entramos y salimos frecuentemente porque es necesario para la instalación.

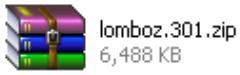
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Instalar EMF

- Como resultado de la descarga, aparecerá un archivo. Lo descomprimiremos en el directorio padre del directorio de instalación de Eclipse. Eclipse no debe estar iniciado.

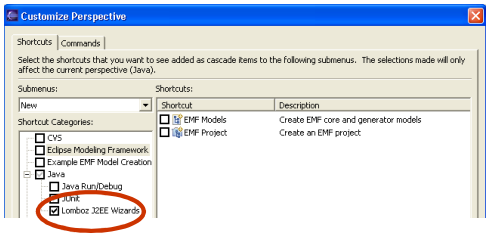
Instalar Lomboz

- Como resultado de la descarga, obtendremos un archivo ZIP. Lo debemos descomprimir en el directorio de Eclipse. Eclipse no debe estar iniciado.



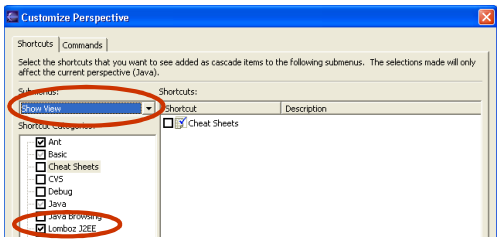
Iniciamos Eclipse

- Vamos a **Windows|Customize Perspective**. En el árbol de la izquierda, hacemos clic en la casilla de verificación **Lomboz J2EE Wizards**.



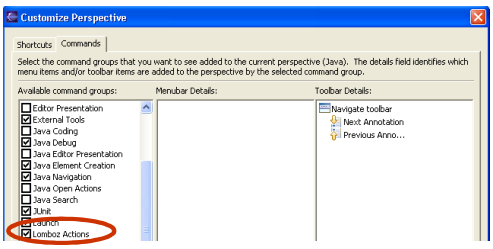
En la misma ventana

- En **Submenus** elegimos **Show View** y seleccionamos la casilla de verificación **Lomboz J2EE.**



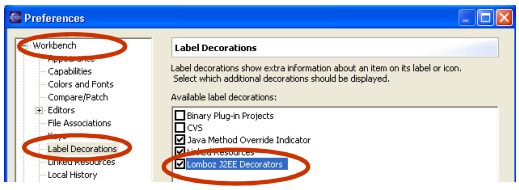
En la misma ventana

- Elegimos la pestaña **Commands** y seleccionamos la casilla de verificación **Lomboz Actions.**



Ahora entramos en Windows|Preferences

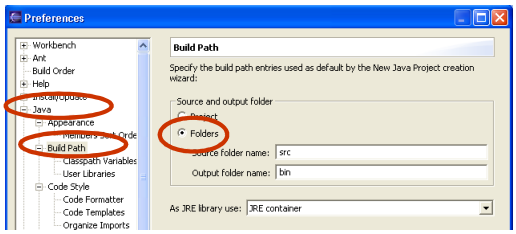
- En el árbol izquierda, expandimos **Workbench,** seleccionamos **Label Decorations** y seleccionamos la casilla de verificación **Lomboz J2EE Decorations.**



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

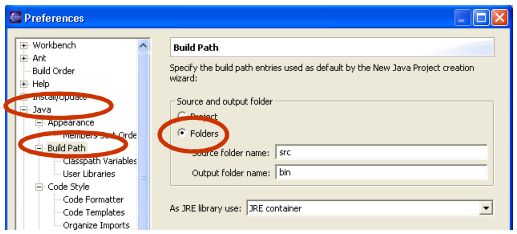
En la misma ventana

- En el árbol izquierda, expandimos **Java,** seleccionamos **Build Path** y seleccionamos el botón de radio **Folders.**



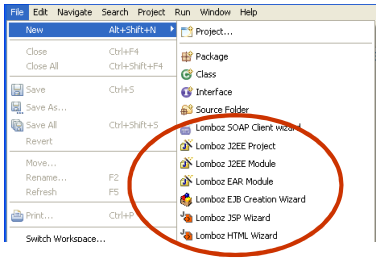
En la misma ventana

- En el árbol izquierda, expandimos **Java,** seleccionamos **Build Path** y seleccionamos el botón de radio **Folders.**



Si hacemos File|New

- Veremos una serie de opciones que comienzan por Lomboz.



5. Introducción a Eclipse

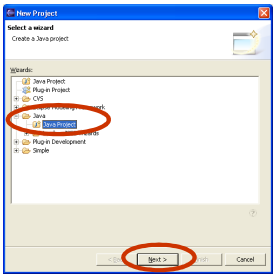
- 5.1. Entornos de desarrollo.
- 5.2. Descargar Eclipse y Lomboz.
- 5.3. Instalar Eclipse y Lomboz.
- 5.4. Algunas tareas básicas en Eclipse.

Proyectos

- Como la mayoría de entornos de desarrollo, Eclipse trabaja con proyectos.
- Un proyecto es el conjunto de archivos que son necesarios para desarrollar y ejecutar una aplicación.

Creando un nuevo proyecto en Eclipse

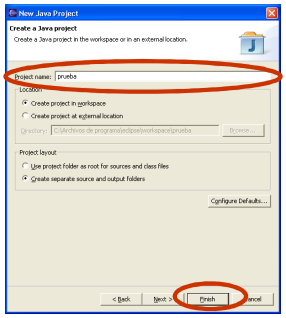
- Debemos hacer **File|New|Project**. Aparecerá una ventana como la siguiente, en la que expandiremos **Java,** seleccionaremos **Java Project** y haremos clic en **Next**



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

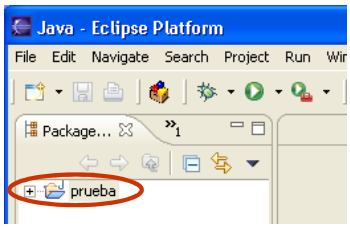
Creando un nuevo proyecto en Eclipse

- Ponemos el nombre del proyecto y hacemos clic en **Finish**. Todas las otras opciones se dejan por defecto.



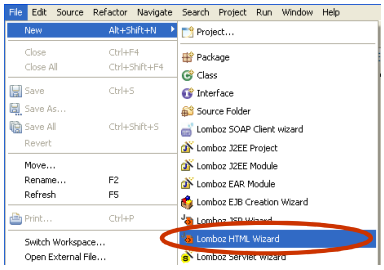
El nuevo proyecto aparecerá a la izquierda

- En un área llamada **Package Explorer** donde aparecerán todos nuestros proyectos para poderlos gestionar fácilmente.



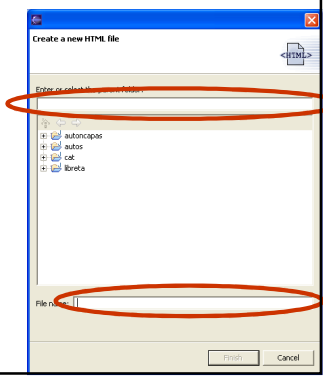
Creemos un archivo para incluir en este proyecto

- Por ejemplo, un documento HTML. Hacemos **File|New|Lomboz HTML Wizard**.



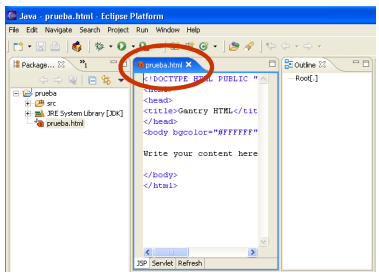
Aparecerá una ventana

- Pondremos el nombre del documento HTML en **File name**, el nombre de la carpeta en **parent folder**. Haremos clic en **Finish**



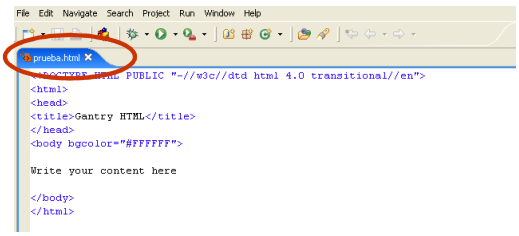
Aparece el documento a la derecha

- Aparece un editor HTML con una estructura de documento HTML. Hagamos doble clic en el título para poder verlo bien.



Así aparece una ventana de edición despejada, más cómoda

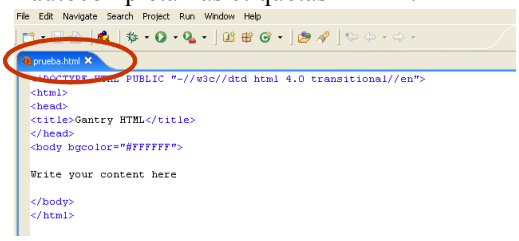
- Hagamos doble clic de nuevo en el título para retornar a la situación anterior.



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

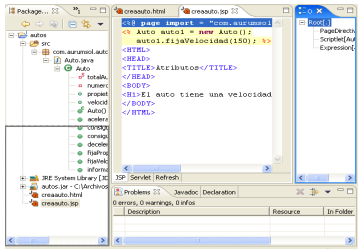
Una de las ventajas de esto

- Es que la sintaxis aparece coloreada, lo que nos permite detectar errores fácilmente. También nos da sugerencias para autocompletar las etiquetas HTML.



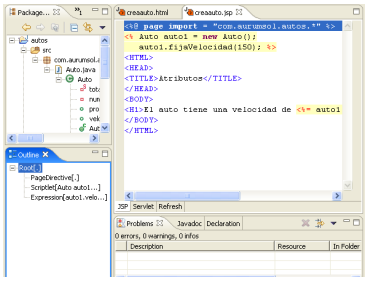
Poniéndonos cómodos

- Arrastren la ventana llamada Outline hasta la parte inferior del Package Explorer



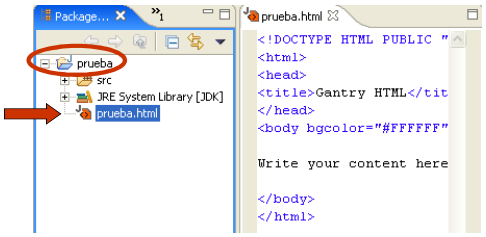
Poniéndonos cómodos

- Así vemos mejor el código



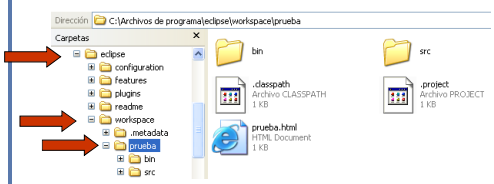
Expandamos el nombre del proyecto que aparece en el Package Explorer

- Debajo de él, aparecerá el nombre del archivo. Así, podremos seleccionar el archivo como si fuera el Explorador de Windows.



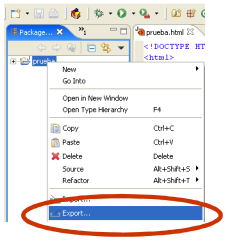
¿Dónde se guarda este archivo en disco?

- Si no hemos especificado lo contrario, se guarda en un directorio con el nombre del proyecto.
- Este directorio se encuentra dentro del directorio **`workspace`** de la instalación de Eclipse.



Hacer una copia de seguridad de un proyecto

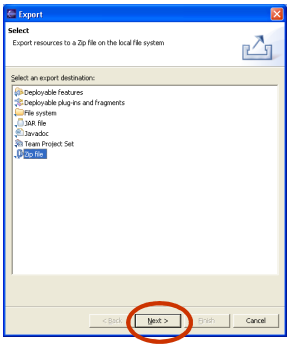
- Se hace clic derecho en el proyecto en el Package Explorer y se selecciona **Export...**



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

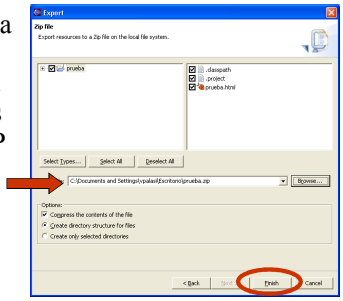
Aparece una ventana

- Elegimos Zip file y hacemos clic en el botón Next >



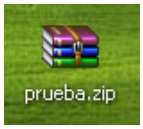
Aparece otra ventana

- Se pone la ruta (ubicación y nombre) en la que queremos guardar el ZIP y se hace clic en **Finish**.



En la ubicación seleccionada

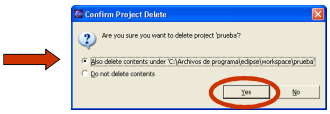
- Aparecerá un archivo ZIP que es la copia de seguridad del proyecto.



- Cuando se les pida una práctica o un ejercicio que entregar, lo que se les está pidiendo es este archivo ZIP de copia de seguridad.

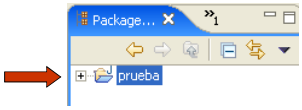
Ejercicio

- Hagan una copia de seguridad de un proyecto de prueba que tengan.
- Después borren el proyecto, haciendo clic en el nombre del proyecto en el Package Explorer y pulsando la tecla "Del" o "Supr". Elijan la opción "Also delete…" en el cuadro de diálogo que aparece.



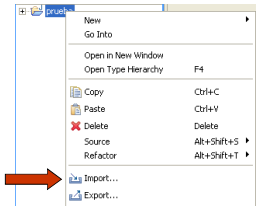
Recuperar una copia de seguridad

- Sólo pueden recuperarse copias de seguridad dentro de un proyecto.
- Lo primero es crear un proyecto con el nombre que queramos restaurar la copia de seguridad. Así:



Recuperar una copia de seguridad

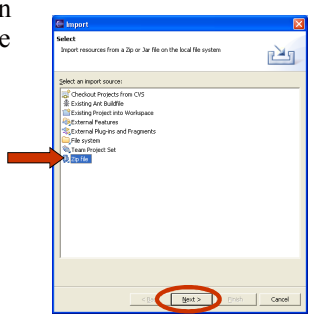
- Se hace clic derecho en el nombre del proyecto y se selecciona Import…



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

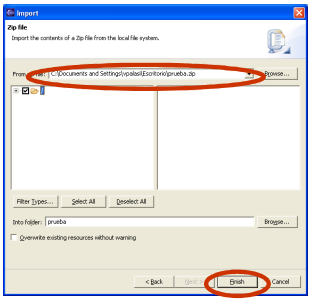
Recuperar una copia de seguridad

- Se elige la opción **Zip file** y se hace clic en **Next**



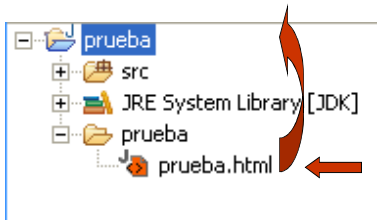
Recuperar una copia de seguridad

- Se elige la ruta del ZIP desde el cual vamos a restaurar y se hace clic en **Finish**.



Recuperar una copia de seguridad

- Los recursos no aparecen en el sitio correcto. Por ello, los arrastramos hasta la carpeta principal.

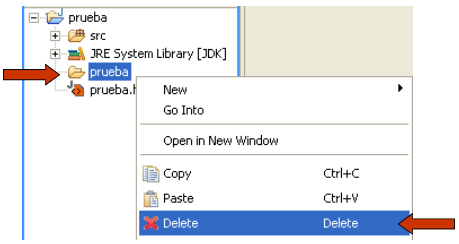


Concretamente, lo que debemos hacer

- 1. El directorio src **que está dentro de la subcarpeta** lo arrastramos hasta el nombre del proyecto.
- 2. El directorio bin lo borramos.
- 3. Si queda algún archivo Java, lo arrastramos a src.
- 4. Todos los archivos que queden los arrastramos hasta el nombre del proyecto.
- 5. Borramos la carpeta vacía que queda.
- 6. Se abren todos los archivos de clases Java. Para cada uno de ellos, si aparece una bombilla al lado de package, se hace clic en ella y se hace clic en la primera opción que aparece.

Recuperar una copia de seguridad

- Ahora borramos la subcarpeta y ya está



Ejercicio

- Recuperar la copia de seguridad que hemos hecho anteriormente.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Programa del tema 1

- 1. Introducción al lenguaje Java.
- 2. Instalación de la plataforma Java.
- 3. Introducción a la programación Web.
- 4. Fundamentos de HTML.
- 5. Introducción a Eclipse.
- 6. Introducción a Tomcat.
- 7. Introducción a JSP.
- 8. Variables. Expresiones aritméticas y lógicas.
- 9. Sentencias algorítmicas: condicionales, ciclos.
- 10. Arreglos.

Queremos poner esta "importante" página Web en la Internet

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
   <HEAD>
      <TITLE>Una p&aacute;gina Web sencilla</TITLE>
   <HEAD>
   <BODY>
      <H1>Una p&aacute;gina Web sencilla</H1>
      <P>Aqu&iacute; se practica el uso de las
         <STRONG>negritas</STRONG> y <EM>la
         cursiva</EM></P>
      <P>Mi direcci&oacute;n preferida en Internet
      es <A href="www.google.com">Google</A>.</P>
   </BODY>
</HTML>
```

Queremos poner esta "importante" página Web en la Internet

- Para poner esta página Web en la Internet, necesitamos un servidor Web.
- Servidor Web: Hardware y software que permite servir páginas Web a un cliente externo mediante Internet.
- Un servidor Web consta de:
  - Hardware. Una máquina conectada a la Internet.
  - Software. Un programa que sirva páginas Web.

Tenemos dos opciones

- 1. Contratar un servicio de alojamiento en Internet. Este nos permite poner nuestras páginas en Internet.
- 2. Implementar nuestro propio servidor Web. Lo que haremos aquí.
  - La máquina conectada a Internet ya la tenemos.
  - Necesitamos un programa para un servidor Web.

Necesitamos un programa que haga de servidor Web

- ¿Cualquier programa?
- En nuestro caso, necesitamos que este servidor soporte también Java. Así, cuando empecemos a programar Java para el Web podremos ejecutar los programas.
- A un programa servidor Web que soporta Java, se le llama "servidor de servlets".

¿Qué servidores de servlets hay disponibles?

- Recordemos, servidores de servlets son servidores Web que soportan Java.
- Hay de dos clases:
  - Los que soportan EJB (una tecnología que no veremos aquí).
  - Los que no soportan EJB.
- Los primeros no los vamos a considerar aquí ya que son muy "pesados" (consumen muchos recursos). Sólo veremos los segundos.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Servidores de servlets

- De código abierto:
- **Jakarta Tomcat.** El más extendido. http://jakarta.apache.org/tomcat/
- **Jetty.** http://jetty.mortbay.org/
- **Enhydra.** http://www.enhydra.org/
- **Resin** http://caucho.com/resin/index.xtp

Servidores de servlets

- Comerciales:
- Hay pocos que no soporten EJB. No están muy extendidos.
- **ServletExec** http://www.newatlanta.com/products/servletexec/index.jsp
- **Servetec Internet Server** http://www.servertec.com/products/iws/iws.html
- **Donde encontrar una lista de servidores de servlets (incluyendo los que soportan EJB):** http://servlets.com/engines/ http://www.javaskyline.com/serv.html

¿Cuál elegiremos? Criterios para escoger un servidor de servlets

- Que esté extendido y lo más estándar posible. Así, tenemos una comunidad que nos sirve para resolver dudas.
- Que sea gratuito. Por razones obvias.
- Que sea de código abierto. Más fiable. Sin problemas de discontinuidad. Podemos programar el código.
- Que sea potente (soporte un alto nivel de concurrencia).
- Que tenga las máximas características posibles (seguridad, logging, etc.)
- Que sea sencillo de instalar, configurar y usar.

And the winner is...

- Tomcat es parte del proyecto Jakarta de la fundación Apache. Es prácticamente el estándar en cuanto a servidores de servlets.
- Es gratuito, de código abierto, potente y sencillo.
- ¿Qué más se puede pedir?

En resumen

- Queremos un servidor Web para poner nuestra página HTML en la Internet.
- Queremos que soporte Java.
- Necesitamos dos cosas:
  - Una máquina conectada a Internet.
  - Un programa servidor de servlets. Tenemos a Tomcat.

Instalación y configuración de Tomcat

- **Nota preliminar e importante**: Para instalar y ejecutar Tomcat en Windows se debe tener una cuenta de administrador del sistema.
- Tomcat abre puertos que solo los puede abrir un administrador.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

1. Descargar la última versión de Tomcat

- Vamos a la página del proyecto Jakarta de la fundación Apache http://jakarta.apache.org y hacemos clic en el enlace **Downloads**.

1. Descargar la última versión de Tomcat

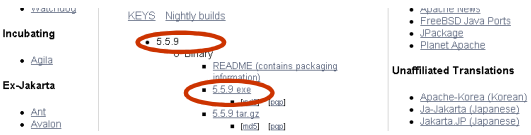
- En la lista de proyectos que aparece, se hace clic en Tomcat.

1. Descargar la última versión de Tomcat

- Se elige la última versión de Tomcat. En nuestro caso, Tomcat 5.

1. Descargar la última versión de Tomcat

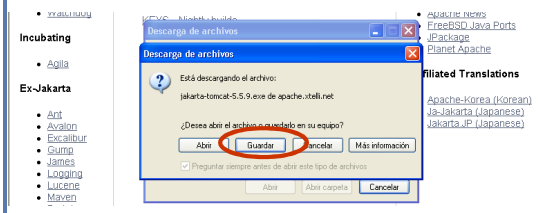
- En la lista que aparece, hacemos clic en la última versión no beta de Tomcat con extensión exe (ya que ésta incluye instalador).



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

1. Descargar la última versión de Tomcat

- Se descargará un archivo.



1. Descargar la última versión de Tomcat

- Más abajo en esta misma página, hay que elegir la descarga llamada "Admin" (mejor la versión ZIP). Se descargará otro archivo.

Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

Antes que nada

- Hay que ver si se tiene Internet Information Services (IIS), un servidor Web que traen por defecto.
- Para ello, hagamos http://localhost.
- Si sale una ventana de contraseña y, al hacer clic en Cancelar, se redirecciona a http://localhost/localstart.asp
- En ese caso, hay que deshabilitarlo.

Deshabilitando IIS

- En **Inicio|Panel de control|Agregar/quitar programas|Agregar o quitar componentes de Windows**, se desactiva la casilla de verificación **Servicios de Internet Information Server (IIS).** Con esto se desinstala.
- Si sólo queremos deshabilitarlo podemos ir a **Inicio|Panel de control|Herramientas administrativas|Servicios** y deshabilitar
  - IIS Admin Service
  - Simple Mail Transport Protocol (SMTP)
  - World Wide Web Publishing Service

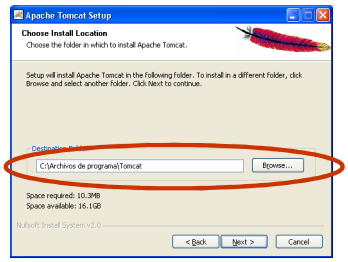
2. Instalar Tomcat

- Como resultado de la descarga, tenemos un programa ejecutable de instalación, el cual ejecutaremos.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

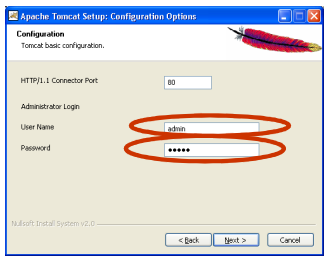
2. Instalar Tomcat

- Elegimos el directorio donde queremos instalar Tomcat.
- En los ejemplos posteriores es **`C:\Archivos de programa\Tomcat`**



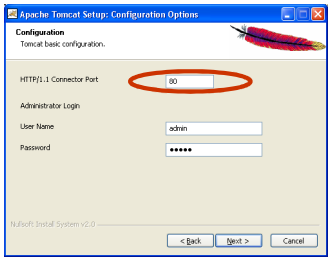
2. Instalar Tomcat

- Ponemos una clave de acceso y contraseña para el administrador.



2. Instalar Tomcat

- Seleccionamos el puerto en el que Tomcat va a recibir las peticiones de las páginas. Se recomienda 80, que es el estándar de HTTP.



¿Por qué 80?

- Por defecto, Tomcat utiliza el puerto 8080 para escuchar las peticiones de los clientes. Esto obliga a que los clientes, deban escribir las URLs de esta forma:

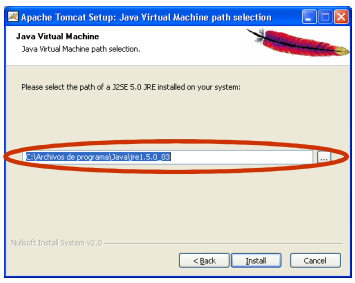
  http://*nombredominio*:8080

- Esto no es lo usual, de forma que si cambiamos el puerto a 80 (que es el defecto para HTTP) será posible escribir URLs como

  http://*nombredominio*

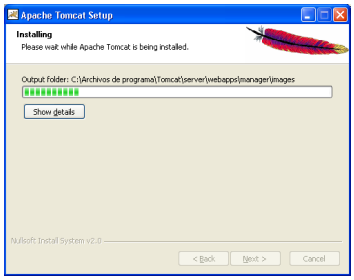
2. Instalar Tomcat

- Se debe especificar el directorio de instalación del JRE de Java.



2. Instalar Tomcat

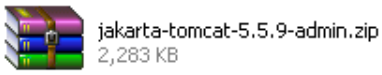
- Esperamos a que Tomcat se acabe de instalar.



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

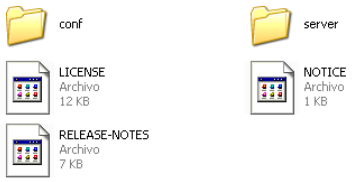
2. Instalar Tomcat

- Se descomprime el archivo ZIP que habíamos descargado y cuyo nombre contiene "Admin".



2. Instalar Tomcat

- Los archivos que resultan de la descompresión, se colocan en los directorios del mismo nombre de la instalación de Tomcat.

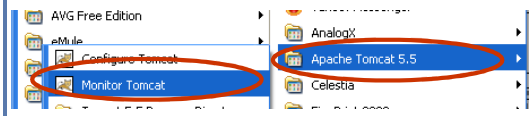


Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

3.1 Ejecutar Tomcat

- En la lista de programas, se selecciona **Apache Tomcat 5.5** y **Monitor Tomcat**.

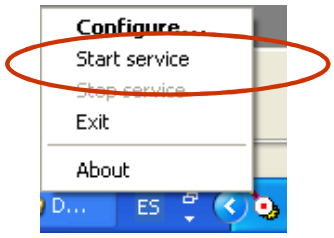


3.1 Ejecutar Tomcat

- En la bandeja del sistema aparece un icono representando a Tomcat.

3.1 Ejecutar Tomcat

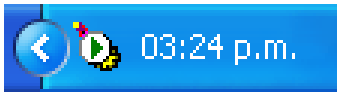
- Haciendo clic derecho en el icono, aparece un menú desde donde podemos iniciar Tomcat.



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

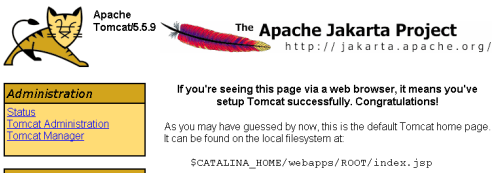
Se inicia Tomcat

- El icono cambia (tiene verde si Tomcat está iniciado y rojo si no lo está).
- Para cerrar Tomcat, basta con hacer clic derecho en el icono y seleccionar **Stop service.**



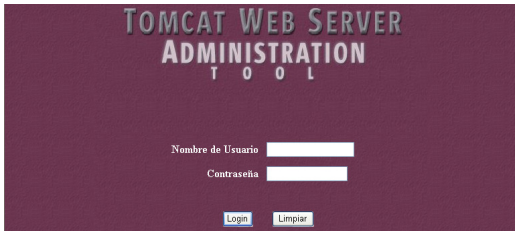
Reiniciamos Tomcat

- Abrimos un browser. Teclear el URL: http://localhost Debe aparecer una página como la siguiente.



Una vez iniciado Tomcat

- Se accede a http://localhost/admin y debe aparecer una página como esta: la administración de Tomcat.



Limpiar algunos archivos

- En **`webapps\ROOT`** borrar todos los archivos y directorios excepto **`WEB-INF`**.
- En **`webapps\ROOT\WEB-INF`** borrar todos los archivos y directorios excepto **`lib`**.
- En **`webapps\ROOT\WEB-INF\lib`** borrar todos los archivos y directorios que haya.

Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

3.2. Definir algunas variables de entorno

- Hay que definir las siguientes variables de entorno para configurar Tomcat. Es bueno reiniciar.

Nombre	Valor
`CATALINA_HOME`	***Directorio De Instalación De Tomcat***
`CLASSPATH`	**`.;%CATALINA_HOME%\common\lib\ servlet-api.jar; %CATALINA_HOME%\common\lib\jsp-api.jar;%CLASSPATH%`** (en una única línea, **sin espacios**)

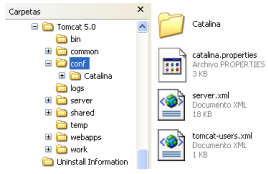
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

3.3. Activar los servlets

- Por ahora no utilizaremos los servlets directamente. Sin embargo, los dejaremos ya activados.
- Para ello editaremos el archivo **web.xml** del subdirectorio **conf** del directorio de instalación de Tomcat.



3.3. Activar los servlets

- En **web.xml** se descomenta, eliminando lo que está en rojo:

```
<!-
<servlet>
  <servlet-name>invoker</servlet-name>
  <servlet-class>
   org.apache.catalina.servlets.InvokerServlet
  </servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet> -->
```

3.3. Activar los servlets

- En **web.xml** se descomenta, eliminando lo que está en rojo:

```
<!-
<servlet-mapping>
  <servlet-name>invoker</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
-->
```

Además es bueno activar la recarga de servlets.

- De forma que Tomcat recompile los servlets que han cambiado desde la última vez que se ejecutaron (esto es automático para las JSPs).
- Es bueno activarlas, pues la recarga es útil para un entorno de desarrollo.
- Sin embargo, sería poco adecuado activar la recarga en un entorno de producción.

En el archivo conf\context.xml, cambiamos

<Context> por **<Context reloadable="true">**. El archivo queda así:

```
<!-- The contents of this file will be
  loaded for each web application -->
<Context reloadable="true">
    <!-- Default set of monitored
  resources -->
    <WatchedResource>WEB-
  INF/web.xml</WatchedResource>
      <!-- Uncomment this to disable
  session persistence across Tomcat
  restarts -->
    <!--
    <Manager pathname="" />
    -->
</Context>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

3.4. Definir un nombre de dominio

- Hasta ahora, se puede acceder a nuestro servidor Web a través de http://***direcciónIP*** (o http://localhost, http://127.0.0.1 desde la misma máquina)
- Sin embargo, si el servidor Web se puede acceder desde fuera de la empresa (no es una Intranet), es mejor acceder mediante un nombre de dominio.
- Eso se debe configurar como explicamos a continuación.

3.4. Definir un nombre de dominio

- Si sólo queremos un nombre de dominio para nuestro servidor (el caso más usual),es sencillo:
- 3.4.1.Nos aseguramos de que nuestra IP es pública y accesible desde el exterior (IPs privadas no sirven).
- 3.4.2.Nos aseguramos de contratar un nombre de dominio y hacerlo apuntar a la dirección IP de nuestra máquina. Esto se hace con una empresa registradora.

3.4.2. Haciendo apuntar el nombre de dominio a la dirección IP

- Cada empresa registradora es distinto. Para Godaddy, primero hacemos login|**Domain Names**|**Manage Domains** y seleccionamos el nombre de dominio:
- A) Si el dominio no está aparcado en Godaddy, hacemos que lo esté. Opciones: **Name Servers Summary**|**Default Parked Nameservers**|**Save changes**.
- B) Abrimos el control del DNS. Total DNS **Control**|**Manage DNS Zone File**.

Después de hacer lo anterior, en GoDaddy.com aparece

- En el record **A**, se pone la dirección IP y después se hace clic en **Save**

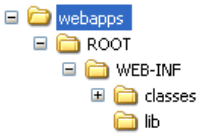
3.4. Definir varios nombres de dominio (1)

- Podemos tener varios nombres de dominio en la instalación de Tomcat. Se necesitan 4 acciones.
- 1. Comprobar que la IP de la máquina es pública y accesible.
- 2. Hacer que todos los nombres de dominio apunten a esta IP. Para ello, debemos contactar con la empresa registradora. Con GoDaddy es lo mismo que antes pero repetido para cada nombre de dominio.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

3.4. Definir varios nombres de dominio (2)

- 3. Dentro del directorio de instalación de Tomcat, se crea un directorio **webapps** para cada nombre, con la siguiente estructura.



- 4. Configurar el archivo **server**.xml del directorio **conf** de la instalación de Tomcat.

Configurar el archivo server.xml

- Para cada nombre de dominio que queramos definir, colocar el fragmento siguiente antes del **</Engine>** que cierra el **<Engine name="Catalina"**

```
<Host name="nombre" debug="0"
  appBase="carpeta" unpackWARs="true"
  autoDeploy="true" xmlValidation="false"
  xmlNamespaceAware="false">
  <Logger
    className="org.apache.catalina.logger.
    FileLogger" directory="logs"
    prefix="nombrearchivo" suffix=".txt"
    timestamp="true" />
</Host>
```

Datos de la transparencia anterior.

- Para cada nombre de dominio, se debe especificar en el fragmento anterior:
  - **nombre**: Nombre de dominio completo con www o bien dirección IP de la máquina (o localhost o 127.0.0.1).
  - **carpeta**: Nombre (sin ruta) de la carpeta webapps del directorio de instalación de Tomcat.
  - **nombrearchivo**: Nombre del archivo de log.
- Estos datos serán diferentes para cada nombre de dominio.

(Entre paréntesis. Forma de servir las peticiones de Tomcat)

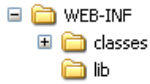
- 1. Sólo se sirven las peticiones que llegan a la máquina:
  - http://***numeroIPdelamáquina***.
  - http://***nombre de dominio*** (asignado por el DNS al IP de esta máquina).
  - http://localhost o http://127.0.0.1 desde la misma máq.
- 2.Estas peticiones son servidas por el **<Engine Catalina**. Si en **server.xml** hay definido un **<Host** con el nombre de dominio o IP de la petición, va al directorio **webapps** que se define en dicho **<Host**.
- 3.Si no hay definido ningún **<Host** así,se toma el **<Host** que indica el **defaultHost** definido en el **<Engine** Catalina.

Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

Crear un contexto para desarrollo

- Debemos tener directorios para desplegar las páginas y otros archivos que necesitemos que sirva el servidor. A estos directorios se les llama contextos.
- Un contexto debe contener una estructura de subdirectorios como la siguiente:



- Hasta ahora sólo hemos tenido un contexto: el directorio **webapps\ROOT** de la instalación de Tomcat. Sin embargo, esto es práctico. ¿Por qué?

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Crear un contexto para desarrollo

- En un entorno real es poco práctico estar desplegando las aplicaciones que están en desarrollo en el mismo directorio que las aplicaciones que están en producción.
- Esto puede producir que dejemos sin servicio accidentalmente una aplicación que los clientes estén utilizando.
- Normalmente, se necesitan dos contextos: uno de desarrollo (para programar) y uno de producción (para los clientes una vez programado).
- Así, podemos programar sin afectar nuestra página Web y sin tocar lo que los clientes están utilizando en estos momentos.

Crear un contexto para desarrollo

- Se necesitan dos contextos: uno para producción y otro para desarrollo.
- El de producción es el que utilizábamos hasta ahora. Está en el subdirectorio **`webapps\ROOT`** de la instalación de Tomcat. Tiene la estructura de subdirectorios de cualquier contexto.
- Pero nos falta el contexto para desarrollo. Este lo deberemos crear.

3.5. Crear un contexto para desarrollo

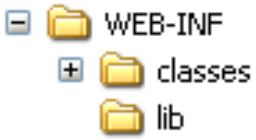
- 3.5.1. Crear un directorio con los subdirectorios adecuados.
- 3.5.2. Indicar a Tomcat que ese directorio es un contexto.

3.5. Crear un contexto para desarrollo

- 3.5.1. Crear un directorio con los subdirectorios adecuados.
- 3.5.2. Indicar a Tomcat que ese directorio es un contexto.

1. Crear un directorio con los subdirectorios adecuados

- Creamos un directorio dentro de **`webapps`** (p.ej., **`C:\Archivos de programa\Tomcat 5.0\webapps\prueba`**) e incluimos unos subdirectorios como los siguientes.



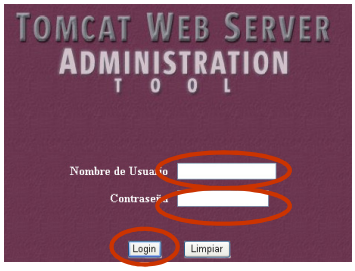
3.5. Crear un contexto para desarrollo

- 3.5.1. Crear un directorio con los subdirectorios adecuados.
- 3.5.2. Indicar a Tomcat que ese directorio es un contexto.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

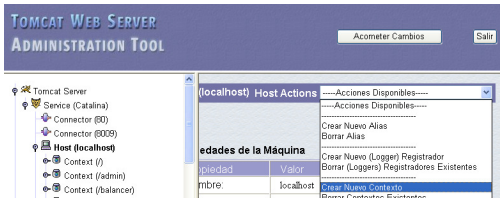
2. Indicar a Tomcat que ese directorio es un contexto

- Abrimos la Herramienta de Administración de Tomcat. Con Tomcat iniciado, en un navegador hacemos http://localhost/admin. Introducimos login y contraseña del administrador. Clic en **Login**.



2. Indicar a Tomcat que ese directorio es un contexto

- En la parte izquierda, expandimos **Service (Catalina)** y hacemos clic en **Host (localhost).** En la parte derecha, seleccionar la acción **"Crear un Nuevo Contexto".**



Configure las opciones remarcadas en rojo

- **Base del documento**. Directorio del contexto (despliegue). Es bueno ponerlo dentro de webapps.
- **Trayectoria.** Prefijo del contexto (con /).
- Después se hace clic en **Guardar**

¿Qué es la trayectoria?

- Es el prefijo del contexto que se colocan en las URLs. Así, en nuestro ejemplo es prueba. Eso significa que si desplegamos un archivo **`hola.html`** en nuestro contexto se podrá acceder con la siguiente URL

http://dominio/prueba/hola.html

- En general,

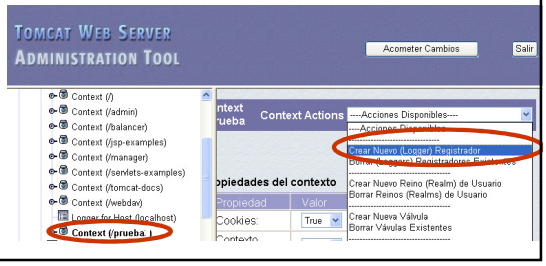
http://dominio/trayectoria/*archivo*

Instalación y configuración de Tomcat

- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

2. Indicar a Tomcat que ese directorio es un contexto

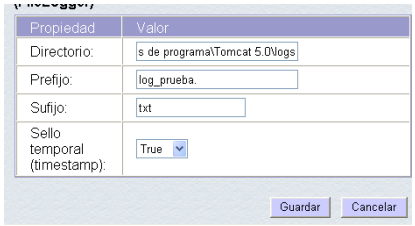
- Además, cada contexto tiene un archivo de log para guardar los mensajes. Para definirlo, se selecciona el contexto "/**prueba**" a la izquierda y a la derecha se hace clic en **Crear Nuevo(Logger) Registrador**



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

2. Indicar a Tomcat que ese directorio es un contexto

- Se le pone un nombre (prefijo) al archivo de log (acabado en punto). Se le pone un sufijo y un directorio (normalmente **`logs`** de la instalación de Tomcat).Se hace clic en **Guardar**.

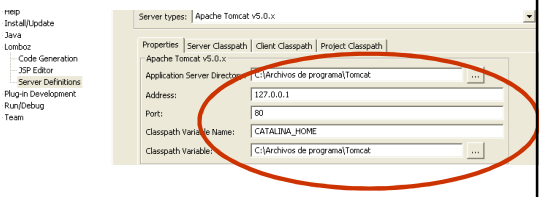


Instalación y configuración de Tomcat

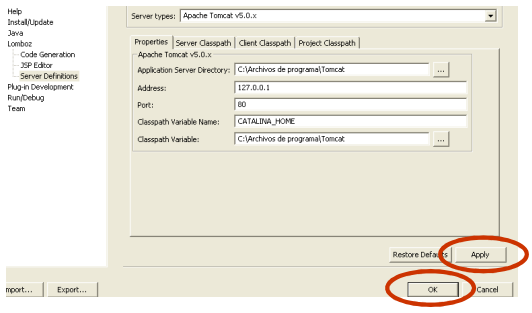
- 1. Descargar la última versión.
- 2. Instalar Tomcat.
- 3. Configurar Tomcat
  - 3.1. Ejecutar Tomcat
  - 3.2. Definir algunas variables de entorno.
  - 3.3. Activar los servlets.
  - 3.4. Definir un nombre de dominio (optativo)
  - 3.5. Crear un contexto para desarrollo (optativo).
- 4. Configurar Eclipse para Tomcat.
- 5. Probar Tomcat.

En Eclipse, se ejecuta Windows|Preferences y en el árbol Lomboz|Server Definitions

- De la lista desplegable **Server types**, se elige **Apache Tomcat v.5.0.x.**
- Después, se especifican los parámetros de configuración que aparecen en la siguiente figura.



Cuando acabamos hacemos clic en Apply y OK



Un experimento

- Hagan un archivo **`prueba.html`** con el contenido que aparece en el cuadro inferior (donde **`X`** sea su nombre).

```
<HTML>
<HEAD><TITLE>Prueba de
  Tomcat</TITLE></HEAD>
<BODY>
<H1>Pagina Web de X</H1>
</BODY>
</HTML>
```

Probar Tomcat

- Copien el archivo **`prueba.html`** al directorio **`webapps\ROOT`** de la instalación de Tomcat.
- En el navegador de Internet escriban **http://localhost/prueba.html**

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Probar el servidor

- Si todo sale bien, deberá aparecer una página Web como la siguiente.
- Ya tiene instalado Tomcat.

Importante

- Todo lo que usted ponga en el directorio **`webapps\ROOT`** de la instalación de Tomcat, será servido por el servidor Web.
- Cuando tenemos una aplicación Web y queremos ponerla a disposición del público (en producción), sólo debemos copiarla a este directorio.
- A esto se le llama **desplegar** una aplicación Web. Al paso se le llama **despliegue** ("deployment").
- A **`webapps\ROOT`** podemos llamarle **directorio de despliegue**.
- Una JSP se programa, se despliega y el usuario la ejecuta desde un navegador.

Ahora cambien el nombre del archivo `prueba.html` a `index.html`

- Y despliéguenlo, es decir, cópienlo al directorio **`webapps\ROOT`**
- Además borren el archivo **`prueba.html`** de ese directorio.
- Después hagan **http://localhost.**
- ¿Qué pasa?

index.html es el archivo por defecto

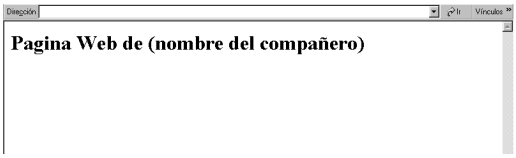
- Si no se especifica qué archivo se desea, el servidor Web supone que es index.html.
- Esto no sólo es con Tomcat o Java, sino que en toda la programación Web.
- Otro ejemplo. Hagan www.aurumsol.com o www.aurumsol.com/index.html y verán lo mismo.

Un experimento

- Abran un navegador y hagan http://localhost ¿Qué aparece?
- Ahora desactiven el firewall de Windows, haciendo clic en el icono con forma de escudo en la bandeja de entrada.
- Ahora hagan http://***direcciónIP***, donde ***direcciónIP*** es la dirección de la máquina del compañero. ¿Qué aparece?

Aparece la página del compañero

- Es decir, una página parecida a esta.



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

¿Qué hemos hecho?

- Cualquier persona del mundo que esté conectada a Internet, puede hacer http://***numeroIPdeNuestraMaquina*** y acceder a la página Web que hemos creado.
- En suma, hemos creado un servidor Web.
- Es decir, acabamos de crear un servicio de alojamiento de páginas Web en nuestra máquina.

Si queremos tener nuestro propio servidor Web

- En la vida real, sólo nos faltaría una cosa: comprar un nombre de dominio y asociarlo a nuestro número de IP. Esto es sencillo de hacer por Internet (aunque las instrucciones varían para cada proveedor de nombres de dominio).
- Ahora que podemos crear nuestro propio servidor Web, nos planteamos la pregunta. ¿Qué es mejor?
  - Tener un servidor Web propio como el que hemos configurado con Tomcat.
  - O bien contratar un servicio de alojamiento por la Internet.

Por el software no hay problema

- Tomcat (u otros servidores de servlets) soporta tantos usuarios concurrentes como deseemos.
- No habría problema en colocar un servidor web en la máquina de nuestro domicilio.

El problema está en el hardware

- Para atender gran cantidad de usuarios concurrentes, se necesitan servidores potentes, una línea de conexión con Internet con gran ancho de banda, etc.
- Esto suele ser más costoso que contratar un alojamiento en Internet.

La conclusión es

- Para grandes empresas, mejor tener un servidor Web propio. Da un mayor control
- Para empresas pequeñas y medianas, depende:
  - Para aplicaciones que no tengan un gran numero de usuarios concurrentes (Intranets o páginas de Internet no muy visitadas) es mejor tener un servidor Web propio, ya que nos da un mayor control y no es caro (como vemos, el software es gratuito).
  - Para aplicaciones con una gran carga de usuarios concurrentes, será más rentable contratar un alojamiento en Internet pues el hardware podría encarecer.

¿Cómo contratar un alojamiento en Internet?

- Busquen en los motores de búsqueda las palabras “Java Hosting”
- Elijan un alojamiento que soporte servlets y (si lo desean) Enterprise Javabeans.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Un alojamiento gratis

- En http://www.myjavaserver.com hay un alojamiento gratis con servlets que puede ser interesante considerar, como mínimo para pruebas.

Ejercicio

- Pongan esta "importante" página en la Internet, desplegándola sobre su servidor Tomcat. Comprueben que está accesible.

Programa del tema 1

1. Introducción al lenguaje Java.
2. Instalación de la plataforma Java.
3. Introducción a la programación Web.
4. Fundamentos de HTML.
5. Introducción a Eclipse.
6. Introducción a Tomcat.
7. Introducción a JSP.
8. Variables. Expresiones aritméticas y lógicas.
9. Sentencias algorítmicas: condicionales, ciclos.
10. Arreglos.

No basta con el HTML

- Con páginas Web como las que hemos visto hasta ahora no podemos hacer una aplicación Web.
- ¿Por qué?

Un ejemplo

- La empresa Cables Salvadoreños, tiene una aplicación Web para sus clientes.
- En esta aplicación, los clientes pueden consultar su saldo con la empresa.
- Para ello, deben introducir su nombre y clave.

Arquitectura de la aplicación de consulta de saldo

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Pasos para consultar el saldo

1. El cliente envía su nombre y clave a través de Internet.
2. Se consulta el saldo en la BD.
3. Se envía al cliente un doc. HTML que contiene el saldo.

Pasos para consultar el saldo

1. El cliente envía su nombre y clave a través de Internet.
2. Se consulta el saldo en la BD.
3. **Se envía al cliente un doc. HTML que contiene el saldo.**

¿Cómo crear 1 archivo HTML con los resultados de la búsqueda en la BD?

- Necesitaríamos construir un archivo HTML en ese mismo momento que recuperamos los datos.
- Esto no lo podemos hacer (hasta ahora). Todos los archivos HTML son fijos y no cambian. Son archivos de texto.
- El servidor Web sólo los envía tal cual están en el disco.

Dos tipos de páginas Web

- **Estáticas.**
  - **Su contenido es el mismo en todas las ocasiones.**
  - **Están en un archivo en el servidor y sólo se envían al cliente tal cual.**
  - **Ejemplo: la mayoría de páginas Web principales.**
- **Dinámicas.**
  - **Su contenido varía según los datos que ha introducido el cliente.**
  - **Se crean en el mismo momento que se envían al cliente.**
  - **Ejemplo: la página que devuelve el saldo.**

Hasta ahora sólo sabemos crear páginas estáticas

- Las ponemos en un archivo y el servidor las sirve siempre igual.
- Para una aplicación Web (como la de consulta de saldo) necesitamos páginas dinámicas.
- ¿Cómo las construimos?

Con HTML no podemos

- HTML sólo puede servir páginas estáticas (siempre las mismas). Esto se debe a razones históricas.
- Cuando se vio la necesidad de páginas dinámicas, aparecieron lenguajes para crearlas. Estos lenguajes son los lenguajes de programación para el Web.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Lenguajes de programación para el Web

- Permiten crear páginas dinámicas.
- Las más populares son:
- Perl. Fue la primera opción para programar páginas Web dinámicas.
- PHP, Python, Ruby. (son lenguajes de código abierto)
- ASP (tanto la versión tradicional como .NET) .
- JSP (y servlets). Son la solución Java para implementar páginas dinámicas.

Nosotros veremos JSP

- Java es un lenguaje mucho más potente que Perl, PHP o Python. Permite hacer aplicaciones más robustas, escalables y para empresas de mayor tamaño.
- Para programar en Web para Java hay servlets y JSP. Pero JSP es la mejor opción.
- JSP significa **J**ava **S**erver **P**ages.

Estructura de una página JSP sencilla

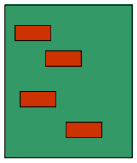
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
 HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <TITLE>
            Título de la página
        </TITLE>
    </HEAD>
    <BODY>
        Diseño de la página
    </BODY>
</HTML>
```

¿Está bromeando? ¡Esto es HTML!

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
 HTML 4.0 Transitional//EN">
<HTML>
    <HEAD>
        <TITLE>
            Título de la página
        </TITLE>
    </HEAD>
    <BODY>
        Diseño de la página
    </BODY>
</HTML>
```

La estructura de una página JSP es igual que la de una página HTML

- De hecho una página JSP no es más que una página HTML con fragmentos (“trocitos”) de Java dentro.
- Si el HTML es verde y el código Java es rojo, esto se podría representar así:



Los trocitos se llaman “Elementos de guión” (“Scripting elements”)

- Los elementos de guión son construcciones sintácticas que permiten insertar código Java en una JSP.
- Hay de tres clases (las veremos más adelante):
- 1. Expresiones JSP.
- 2. Scriptlets.
- 3. Declaraciones JSP.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Una página JSP sencilla

```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Escuela</TITLE>
  </HEAD>
  <BODY>
  <H1>Dos por dos son <%= 2*2 %>
  </H1>
   </BODY>
</HTML>
```

Todo lo que está en negro lo conocemos

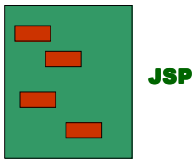
```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Escuela</TITLE>
  </HEAD>
  <BODY>
  <H1>Dos por dos son <%= 2*2 %>
  </H1>
   </BODY>
</HTML>
```

¿Qué es lo que está en rojo?

```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Escuela</TITLE>
  </HEAD>
  <BODY>
  <H1>Dos por dos son <%= 2*2 %>
  </H1>
   </BODY>
</HTML>
```

Es un trozo de Java

- Recordemos que una página JSP no es más que una página HTML con fragmentos de Java dentro, llamados elementos de guión.
- Si el HTML es verde y el código Java es rojo, esto se podría representar así:



Este elemento de guión es una expresión JSP

```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Escuela</TITLE>
  </HEAD>
  <BODY>
  <H1>Dos por dos son <%= 2*2 %>
  </H1>
   </BODY>
</HTML>
```

Expresiones JSP

- Permiten introducir el valor de una expresión Java entre el código HTML.
- Cuando se accede a la página JSP, la expresión es evaluada y su valor es insertado en el HTML (en la página).
- Las expresiones JSP se incluyen entre `<%=` y `%>`

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Expresiones JSP

- Permiten introducir el valor de una expresión Java entre el código HTML.
- Cuando se accede a la página JSP, la expresión es evaluada y su valor es insertado en el HTML (en la página).
- Las expresiones JSP se incluyen entre `<%=` y `%>`

Vemos que la expresión JSP es 2*2

```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Escuela</TITLE>
  </HEAD>
  <BODY>
  <H1>Dos por dos son <%= 2*2 %>
  </H1>
   </BODY>
</HTML>
```

<%= y %> sólo es para indicarnos que lo de dentro es una expresión JSP

Expresiones JSP

- Permiten introducir el valor de una expresión Java entre el código HTML.
- Cuando se accede a la página JSP, la expresión es evaluada y su valor es insertado en el HTML (en la página).
- Las expresiones JSP se incluyen entre `<%=` y `%>`

Cuando se accede a la página, la expresión 2*2 es evaluada

```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Escuela</TITLE>
  </HEAD>
  <BODY>
  <H1>Dos por dos son <%= 2*2 %>
  </H1>
   </BODY>
</HTML>
```

<%= y %> sólo es para indicarnos que lo de dentro es una expresión JSP

Expresiones JSP

- Permiten introducir el valor de una expresión Java entre el código HTML.
- Cuando se accede a la página JSP, la expresión es evaluada y su valor es insertado en el HTML (en la página).
- Las expresiones JSP se incluyen entre `<%=` y `%>`

El 4 es insertado en el HTML. Tendremos.

```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Escuela</TITLE>
  </HEAD>
  <BODY>
  <H1>Dos por dos son 4
  </H1>
   </BODY>
</HTML>
```

Es decir, generamos el HTML en el momento: página dinámica

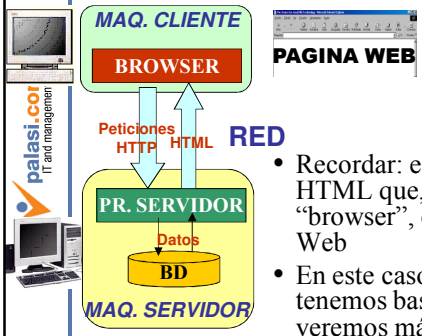
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

En la página Web tendremos lo siguiente

Dos por dos son 4

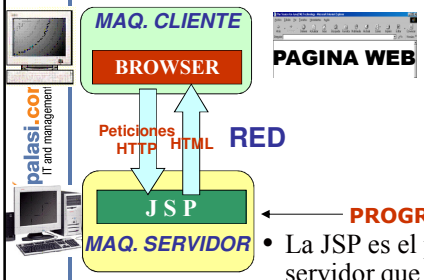
- Esto es lógico sabiendo el HTML que se ha generado (transparencia anterior).

Es decir, lo que hace la JSP es generar una página HTML (dinámicamente)



- Recordar: el servidor genera HTML que, en el "browser", crea la página Web
- En este caso, aún no tenemos base de datos (ya la veremos más adelante).

Es decir, lo que hace la JSP es generar una página HTML (dinámicamente)



- La JSP es el programa servidor que genera HTML.

Podemos ver la JSP como una "plantilla" que genera documentos HTML

- O como una fábrica de documentos HTML.
- En realidad, es un programa que genera HTML (pero, por ahora, no veremos esto con detalle).
- Después los documentos HTML son visualizados en el navegador como páginas.

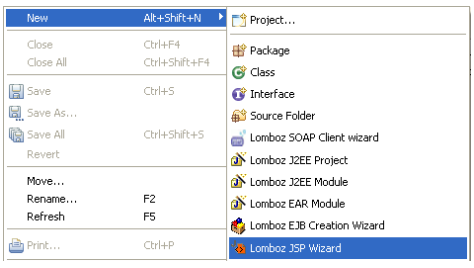
Veamos esto en la práctica

- Creen un archivo anterior con la página JSP **USANDO ECLIPSE** y guárdenlo como **escuela.jsp**

```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Escuela</TITLE>
  </HEAD>
  <BODY>
  <H1>Dos por dos son <%= 2*2 %>
 </H1>
  </BODY>
</HTML>
```

Crear una JSP usando Eclipse

- Se hace **File|New|Lomboz JSP Wizard**

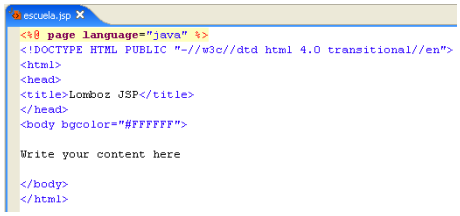


Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Crear una JSP usando Eclipse

- Se hace clic en el proyecto en el que queremos crear la JSP y se escribe el nombre de la misma.

Crear una JSP usando Eclipse



- Aparece un esqueleto de JSP. Lo bueno es que las palabras claves están coloreadas y se distingue por el color el código Java del código HTML, lo que es muy útil para detectar errores. Además, autocompleta etiquetas HTML.

¿Cómo hacemos para que los visitantes accedan a esta página? Despliegue

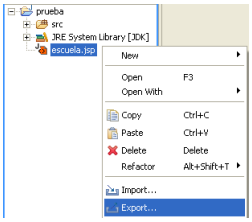
- Es igual que desplegar un documento HTML normal.
- La ponemos en el directorio de despliegue **`webapps\ROOT`** para colocar esta página en el servidor Web. Ahora ya está disponible para todos los visitantes de la página Web.

¿Cómo desplegar?

- Obviamente, copiando en el directorio de despliegue **`webapps\ROOT`**. Pero hay dos maneras.
- Copiando en el explorador de Windows (la más obvia).
- Haciéndolo desde Eclipse (la más práctica, pues estamos programando desde este entorno).

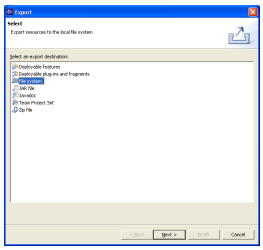
Desplegando desde Eclipse

- Se hace clic derecho en la JSP que queremos explorar y se selecciona Export.



Desplegando desde Eclipse

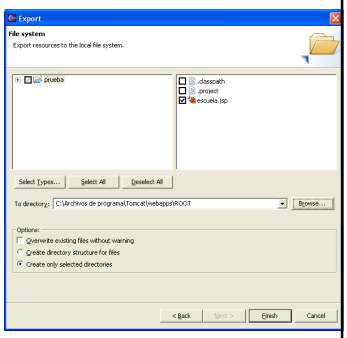
- Se selecciona File system.



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Desplegando desde Eclipse

- Se selecciona el directorio de despliegue (esto sólo se debe hacer una vez, ya que después sale por defecto).
- Se hace clic en **Finish** y ya está.



Despliegen la JSP anterior mediante Eclipse

- Para acceder a ella, con Tomcat iniciado, abrimos un navegador y escribimos http://localhost/escuela.jsp . Háganlo.
- En un caso real, escribiríamos el nombre de dominio en vez de localhost

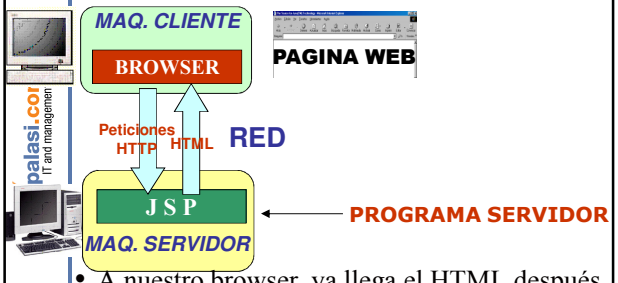
Como ven, es un poco lento

- Esto es porque la primera vez que se accede a una página JSP, Tomcat la compila.
- Las siguientes veces es rápido. Compruébenlo accediendo a esta página por segunda vez.
- Por ello, es una buena práctica que el programador acceda una primera vez a cada página JSP después de desplegarla. Así, los clientes externos no sufren ningún retraso posible.

Una vez obtengan la página en el browser

- En su navegador, ejecuten **Ver | Código fuente**
- ¿Qué es lo que ven?
- Como ven, la JSP produce un HTML después de evaluar la expresión.

Recordemos que la JSP genera dinámicamente una página HTML



- A nuestro browser, ya llega el HTML después de evaluar la expresión. Es lo que vemos con **Ver|Código fuente**

Ya hemos creado y desplegado nuestra primera página JSP

- Sin embargo, no es muy interesante. El resultado lo podríamos obtener con un documento HTML estático.
- Vamos a hacer un caso más interesante (e imposible con HTML).
- Queremos hacer una aplicación Web en la que introduzcamos un nombre y el programa nos salude por este nombre.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Preguntas

- ¿De cuantas páginas constará esta aplicación Web?
- ¿Cuáles serán HTML y cuáles JSP?
- ¿Cómo será la primera de las páginas?

Respuestas

- Debe haber dos páginas: una para introducir el nombre y otra para saludar con ese nombre.
- La primera es un formulario y, por lo tanto, es HTML. La segunda es dinámica (se genera en el momento) y, por lo tanto, es una JSP.
- Las llamaremos **`nombre.html`** y **`saludo.jsp`** respectivamente.

¿Cómo relacionamos estas dos páginas?

- Tenemos que decir al servidor que los datos recogidos en **`nombre.html`** sean enviados a **`saludo.jsp`** para ser procesados.
- Esto se indica poniendo en el formulario la URL de la página que procesa los datos. Así:

```
<FORM ACTION="http://localhost/
  saludo.jsp">
```

En general

- Para especificar qué programa procesará los datos de un formulario HTML, se escribe el URL de este programa en el atributo **`ACTION`** de la etiqueta **`FORM`**.

```
<FORM ACTION="URLDelPrograma">
```

- Esto no sólo se aplica a las JSP sino a cualquier lenguaje de programación Web.
- Recordemos que las JSP son un programa (que genera HTML).

Ejercicio

- Escriban **`nombre.html`** (el documento HTML que sirva para introducir el nombre que queremos saludar).

Solución

```
<!DOCTYPE HTML ...>
<HTML><HEAD><TITLE>Tu nombre</TITLE></HEAD>
<BODY>
<H1>Saludar es de bien educados</H1>
<FORM ACTION="http://localhost/saludo.jsp">
   <P><STRONG>Escribe tu
  nombre</STRONG><BR>
   <INPUT TYPE="TEXT" NAME="nombre"><BR>
   <INPUT TYPE="SUBMIT"
  VALUE="Saludar"></P>
</FORM>
</BODY>
</HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ahora veremos cómo es la página JSP (saludo.jsp)

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Saludo</TITLE>
    </HEAD>
    <BODY>
    <H1>Hola,
    <%= request.getParameter("nombre") %>.
     &iquest;C&oacute;mo amaneciste?</H1>
   </BODY>
   </HTML>
```

¿Qué es esto?

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Saludo</TITLE>
    </HEAD>
    <BODY>
    <H1>Hola,
    <%= request.getParameter("nombre") %>.
     &iquest;C&oacute;mo amaneciste?</H1>
   </BODY>
   </HTML>
```

request.getParameter ("*nombreCuadro*")

- Se supone que la JSP que contiene esta expresión ha recibido los datos de un formulario.
- Esta expresión sirve para recuperar los datos del formulario.

request.getParameter ("*nombreCuadro*")

- Devuelve el valor del cuadro de texto que se llama ***nombreCuadro***
- Es decir:
- En el formulario se ha definido un cuadro de texto así

```
<INPUT TYPE ="TEXT"
  NAME="nombreCuadro">
```

- El valor que hemos introducido en ese cuadro es el que se recuperará con

```
request.getParameter("nombreCuadro")
```

En nuestro caso

```
<!DOCTYPE HTML ...>

<HTML><HEAD><TITLE>Tu nombre</TITLE></HEAD>
<BODY>
<H1>Saludar es de bien educados</H1>
<FORM ACTION="http://localhost/saludo.jsp">
   <P><STRONG>Escribe tu nombre</STRONG><BR>
   <INPUT TYPE="TEXT" NAME="nombre"><BR>
   <INPUT TYPE="SUBMIT" VALUE="Saludar"></P>
</FORM>
</BODY>
</HTML>
```

- Hemos definido un cuadro de texto llamado "nombre", donde escribiremos el nombre de pila.

En nuestro caso

```
<!DOCTYPE HTML ...>

<HTML><HEAD><TITLE>Tu nombre</TITLE></HEAD>
<BODY>
<H1>Saludar es de bien educados</H1>
<FORM ACTION="http://localhost/saludo.jsp">
   <P><STRONG>Escribe tu nombre</STRONG><BR>
   <INPUT TYPE="TEXT" NAME="nombre"><BR>
   <INPUT TYPE="SUBMIT" VALUE="Saludar"></P>
</FORM>
</BODY>
</HTML>
```

- Es el valor que ponemos en este cuadro el que devolverá

```
request.getParameter("nombre")
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Cada vez que se acceda a esta página JSP

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
    <TITLE>Saludo</TITLE>
   </HEAD>
   <BODY>
   <H1>Hola,
   <%= request.getParameter("nombre") %>.
   &iquest;C&oacute;mo amaneciste?</H1>
  </BODY>
  </HTML>
```

- La expresión JSP evaluará al nombre que se introdujo y así se dará el saludo.

Ejercicio optativo

- Creen una aplicación Web que pida un nombre y lo repita tres veces (sin espacios en blanco).
- Así: Pedro debería dar
- PedroPedroPedro
- Nota: usen + para juntar nombre y apellido (concatenación de cadenas).
- Utilicen Eclipse para crear los dos archivos (con las opciones Lomboz HTML Wizard y Lomboz JSP Wizard).

Solución

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
    <TITLE>Eco</TITLE>
   </HEAD>
   <BODY>
   <H1>Tu nombre repetido es
     <%= request.getParameter("nombre")+
      request.getParameter("nombre")+
      request.getParameter("nombre") %>.
   </H1>
  </BODY>
  </HTML>
```

Comentarios en JSP (1)

- Hay dos tipos de comentarios en JSP.
- Podemos utilizar los comentarios HTML, que tienen el siguiente formato.

```
<!-- comentario -->
```

- Estos comentarios (como cualquier código HTML) se envían directamente al navegador del cliente, así que el usuario de la JSP puede observarlo con **Ver|Código fuente** en su navegador.

Comentarios en JSP (2)

- Si no queremos que el cliente pueda ver el comentario, podemos utilizar un comentario JSP, el cual tiene el siguiente formato.

```
<%-- comentario --%>
```

- Estos comentarios no son visibles por el cliente, pues no se traducen a HTML ni, por lo tanto, se envían al navegador.

Por ejemplo

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
    <TITLE>Eco</TITLE>
   </HEAD>
   <BODY>
   <H1>Tu nombre repetido es
     <%-- Aquí repetimos el nombre
      3 veces --%>
     <%= request.getParameter("nombre")+
      request.getParameter("nombre")+
      request.getParameter("nombre") %>.
   </H1>
  </BODY></HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Programa del tema 1

- 1. Introducción al lenguaje Java.
- 2. Instalación de la plataforma Java.
- 3. Introducción a la programación Web.
- 4. Fundamentos de HTML.
- 5. Introducción a Eclipse.
- 6. Introducción a Tomcat.
- 7. Introducción a JSP.
- 8. Variables. Expresiones aritméticas y lógicas.
- 9. Sentencias algorítmicas: condicionales, ciclos.
- 10. Arreglos.

8. Variables. Expresiones aritméticas y lógicas.

- 8.1. Variables.
- 8.2. Literales.
- 8.3. Operadores.
- 8.4. Precedencia de operadores.

8. Variables. Expresiones aritméticas y lógicas.

- 8.1. Variables.
- 8.2. Literales.
- 8.3. Operadores.
- 8.4. Precedencia de operadores.

Esto es muy engorroso de digitar

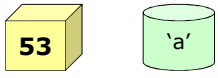
```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
    <TITLE>Eco</TITLE>
   </HEAD>
   <BODY>
   <H1>Tu nombre repetido es
     <%= request.getParameter("nombre")+
      request.getParameter("nombre")+
      request.getParameter("nombre") %>.
   </H1>
  </BODY>
  </HTML>
```

Sería mejor no repetir la misma expresión

- Podríamos almacenar el resultado de `request.getParameter("nombre")` en una variable (por ejemplo, **`nombre1`**) y hacer `<%= nombre1+nombre1+nombre1 %>`.
- Pero no sabemos manejar las variables en Java. Este es un tema importante y, por lo tanto, lo trataremos ahora.

Repasando el concepto de variables (1)

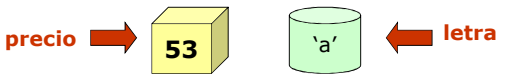
- Una variable es una o más direcciones de memoria que guardan algún contenido.
- A alto nivel, podemos imaginar una variable como un recipiente que guarda un dato o información.
- Hay variables de diferentes tipos según el tipo de dato que guardan



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

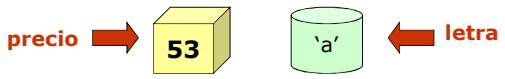
Repasando el concepto de variables (2)

- Cada variable tiene un nombre por la que la identificamos.
- De una variable, nos interesa:
  - Su nombre.
  - Su contenido.
  - El tipo: tipo de datos que puede contener.



Repasando el concepto de variables (3)

- Podemos hacer dos tipos de operaciones con las variables:
  - Guardar un valor en ellas.
  - Recuperar el valor que tienen guardado.
- Sólo se pueden guardar datos que sean del mismo tipo que la variable.



Tipos de datos básicos o primitivos

Nombre	Valores
byte	Números enteros de –128 a 127
short	Enteros de –32768 a 32767
int	Enteros de –2,147,483,648 a 2,147,483,647
long	Enteros de –922117036854775808 a 922117036854775807
float	Reales de ±3.40282347e+38 a ±1.40239846e-45
double	Reales de ±1. 79769313486231570e+308 a ±4.94065645841246544e-324
char	Cualquier carácter Unicode.
boolean	true o false

Los más usados están en rojo

Nombre	Valores
byte	Números enteros de –128 a 127
short	Enteros de –32768 a 32767
int	Enteros de –2147483648 a 2147483647
long	Enteros de –922117036854775808 a 922117036854775807
float	Reales de ±3.40282347e+38 a ±1.40239846e-45
double	Reales de ±1. 79769313486231570e+308 a ±4.94065645841246544e-324
char	Cualquier carácter Unicode.
boolean	true o false

Tamaño de tipos primitivos

Nombre	Tamaño
byte	1 byte
short	2 bytes
int	4 bytes
long	8 bytes
float	4 bytes
double	8 bytes
char	2 bytes
boolean	1 byte

Un tipo de datos no primitivo

- `String`
- Contiene cadenas de caracteres.
- Es diferente a los tipos primitivos, como veremos.
- Recuerden: se escribe con mayúscula inicial.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

En Java, como en la mayoría de lenguajes

- Antes de usar una variable, la tenemos que declarar.
- Declarar una variable es decirle al programa el nombre y el tipo de la variable
- Así se le "avisa" al programa de que vamos a usar esa variable.

Declaración de variables

- En Java, el formato de declaración de variables es
  tipo nombre;
- Ejemplo:

```
int precio;
char letra;
boolean aplicaIVA;
```

- Quiere decir vamos a usar:
  - Una variable entera llamada **precio**.
  - Una variable de tipo carácter llamada **letra**.
  - Una variable booleana llamada **aplicaIVA**

Nombres de las variables e identificadores

- Reglas aplicables a cualquier identificador: variables, nombres de clases, de métodos, etc.
- El primer carácter debe ser una letra y los otros deben ser letras o dígitos.
- Se considera letra:
  - Cualquier carácter Unicode que sea una letra (también otros alfabetos).
  - El signo de dólar ($) y el de subrayado (_).
- Se considera dígito del 0 al 9.
- Ningún identificador puede ser una palabra reservada por Java.

Palabras reservadas por Java

abstract	default	goto	operator	synchronized
boolean	do	if	outer	this
break	double	implements	package	throw
byte	else	import	private	throws
byValue	extends	inner	protected	transient
case	false	instanceof	public	true
cast	final	int	rest	try
catch	finally	interface	return	var
char	float	long	short	void
class	for	native	static	volatile
const	future	new	super	while
continue	generic	null	switch	

Nombres de las variables

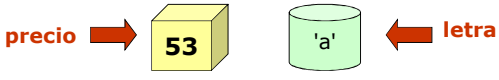
- Ejemplos de nombres de variables:

```
marañón
ejemploDeVariable
otroEjemplo
ooEjemlo
_Ídéntïficador_nô_334343
```

- Java es "case sensitive" la variable **precio** no es la misma que la variable **Precio**.
- **Por convención, las variables Java comienzan en minúsculas y las siguientes palabras tienen su inicial en mayúsculas.**

Repasando el concepto de variables

- Podemos hacer dos tipos de operaciones con las variables:
  - Guardar un valor en ellas.
  - Recuperar el valor que tienen guardado.



Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Guardando un valor en una variable

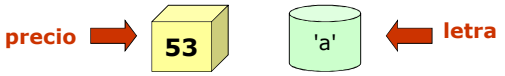
- Es una sentencia llamada "asignación" con la siguiente sintaxis.

```
nombre = valor;
```
(mismo tipo)

- Así, por ejemplo,

```
precio = 53;
letra = 'a';
```

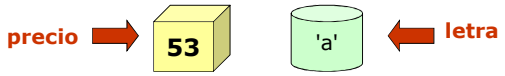
- Es una sentencia (acción), no una igualdad (relación).



Recuperando el valor de una variable

- Para recuperar el valor de una variable, sólo debe escribirse su nombre
- Así, por ejemplo,

```
<%= precio %>
costo = precio;
```



Más sobre declaración de variables

- El formato es ***tipo nombre;***

```
int precio;
```

- Pero también pueden encadenarse variables del mismo tipo.

```
int precio, edad, costo;
```

- También puede darse un valor inicial a la variable cuando se declare.

```
int precio = 10;
```

En Java, se puede dar un valor inicial a la variable cuando se declara

- El formato para hacerlo es

```
tipo nombre = valorinicial;
```

- Ejemplo:

```
int precio = 10;
char letra = 'a';
boolean aplicaIVA = true;
```

- Como vemos, esto es hacer una declaración de variables y una asignación al mismo tiempo.

Es lo mismo

- El código siguiente

```
int precio;
precio = 10;
```

- Que el código siguiente:

```
int precio = 10;
```

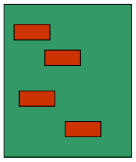
¿Dónde ponemos la declaración de variables y la asignación en las JSP?

- Claramente, no dentro de las expresiones JSP, pues éstas son sólo para expresiones y la declaración y la asignación son instrucciones.
- Para ello, usamos una construcción sintáctica llamada **scriptlets.**

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Recordemos: "Elementos de guión" ("Scripting elements")

- Los elementos de guión son construcciones sintácticas que permiten insertar código Java en una JSP.
- Hay de tres clases:
- 1. Expresiones JSP. (ya la hemos visto)
- 2. Scriptlets.
- 3. Declaraciones JSP.



Scriptlets

- Sirven para introducir instrucciones de código Java directamente dentro de una JSP.
- Las instrucciones se colocan entre `<%` y `%>`.
- Las instrucciones de los scriptlets son ejecutadas en el punto en que se insertan dentro del código HTML.
- Mientras las expresiones JSP sólo podían incluir expresiones, los scriptlets pueden incluir instrucciones (sentencias, comandos) Java enteras.

Diferencia entre las expresiones JSP y los scriptlets.

- Las expresiones JSP contienen expresiones: devuelven un valor. Este valor es insertado en el código HTML.
- Los scriptlets contienen instrucciones (sentencias, comandos): se ejecutan en un punto del documento HTML pero no devuelven nada.

Ejemplo. Recordemos el ejercicio anterior.

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
    <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Tu nombre repetido es
     <%= request.getParameter("nombre")+
      request.getParameter("nombre")+
      request.getParameter("nombre") %>.
    </H1>
  </BODY>
  </HTML>
```

Se puede escribir así

```
<% String nombre1 =
  request.getParameter("nombre"); %>
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
    <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Tu nombre repetido es
     <%= nombre1+nombre1+nombre1 %>. </H1>
  </BODY>
  </HTML>
```

Scriptlet donde declaramos la variable nombre1 y le asignamos su valor inicial

```
<% String nombre1 =
  request.getParameter("nombre"); %>
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
    <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Tu nombre repetido es
     <%= nombre1+nombre1+nombre1 %>. </H1>
  </BODY>
  </HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

nombre1 tendrá el valor que hemos introducido en el formulario

```
<% String nombre1 =
  request.getParameter("nombre"); %>
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Tu nombre repetido es
     <%= nombre1+nombre1+nombre1 %>. </H1>
   </BODY>
   </HTML>
```

En la expresión JSP lo concatenamos tres veces

```
<% String nombre1 =
  request.getParameter("nombre"); %>
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Tu nombre repetido es
     <%= nombre1+nombre1+nombre1 %>. </H1>
   </BODY>
   </HTML>
```

También podríamos haber hecho esto.

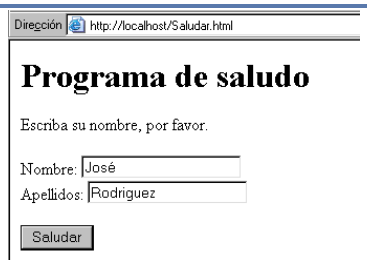
```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
   <% String nombre1 =
    request.getParameter("nombre"); %>
    <H1>Tu nombre repetido es
     <%= nombre1+nombre1+nombre1 %>. </H1>
   </BODY>
   </HTML>
```

Pero es mejor que el código Java y HTML se mezclen lo menos posible

Fíjense que el `request.getParameter("nombre");` es de tipo `String`

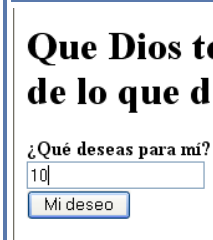
```
<% String nombre1 =
  request.getParameter("nombre"); %>
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Tu nombre repetido es
     <%= nombre1+nombre1+nombre1 %>. </H1>
   </BODY>
   </HTML>
```

En general, todo lo que introduzcamos en un formulario HTML será de tipo `String`



- Incluso si ponemos `TYPE="NUMERIC"`

¿Qué pasa si queremos tratar la entrada al formulario como un numero (entero)?



- Esta es una aplicación de cálculo del doble.
- Queremos considerar la entrada del formulario como un entero y no como una cadena de caracteres.
- Así podremos hacer cálculos con ella.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Se utiliza esta expresión para convertir tipo

- Para convertir las variables de tipo **`String`** en variables de tipo **`int`**.

```
Integer.parseInt(exprString)
```

Así, la aplicación de doble anterior, tendría este formulario (`doble.html`)

```
<!DOCTYPE HTML ...>
<HTML><HEAD><TITLE>El doble</TITLE></HEAD>
<BODY>
<H1>Que Dios te de el doble de lo que
  desees para m&iacute;.</H1>
<FORM
  ACTION="http://localhost/resultado.jsp">
    <P><STRONG>&iquest;Qu&eacute; deseas
  para  m&iacute;?</STRONG><BR>
    <INPUT TYPE="TEXT" NAME="numero"><BR>
    <INPUT TYPE="SUBMIT" VALUE="Mi
  deseo"></P>
</FORM>
</BODY></HTML>
```

Y esta página JSP (`resultado.jsp`)

```
<% String entrada =
  request.getParameter("numero");
  int numero= Integer.parseInt(entrada); %>
<!DOCTYPE HTML ...>
   <HTML><HEAD>
     <TITLE>Mi deseo</TITLE>
    </HEAD>
    <BODY>
    <H1>&iexcl;Que Dios te de
     <%= numero*2 %>!</H1>
   </BODY>
</HTML>
```

O también podría escribirse

```
<% int numero= Integer.parseInt(
  request.getParameter("numero")); %>
<!DOCTYPE HTML ...>
   <HTML><HEAD>
     <TITLE>Mi deseo</TITLE>
    </HEAD>
    <BODY>
    <H1>&iexcl;Que Dios te de
     <%= numero*2 %>!</H1>
   </BODY>
</HTML>
```

Ustedes mismos

- Despliegen esta aplicación y ejecútenla.

Ejercicio

- Programen una aplicación Web que pida dos números y devuelva la suma de estos dos números.
- Despliéguenla.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

8. Variables. Expresiones aritméticas y lógicas.

- 8.1. Variables.
- 8.2. Literales.
- 8.3. Operadores.
- 8.4. Precedencia de operadores.

Recordando la sentencia de asignación

- Habíamos dicho que la sintaxis era
  nombre = valor;
- En realidad, la sintaxis es
  nombre = expresion;
- Una expresión es un enunciado en Java que retorna un valor. Por ejemplo,
  - El literal **`53`**
  - La variable **`precio`**.
  - La expresión **`precioSinIVA*1.13`**

Una expresión es un enunciado que retorna un valor

- Una expresión puede estar compuesta por:
  - Un literal **`53 'a'`**
  - El nombre de una variable **`precio`**
  - Varias expresiones unidas por operadores.
    `precioSinIVA*1.13`

    (Un operador es un símbolo que simboliza una operación)

Una expresión es un enunciado que retorna un valor

- Una expresión puede estar compuesta por:
  - Un literal **`53 'a'`**
  - El nombre de una variable **`precio`**
  - Varias expresiones unidas por operadores.
    `precioSinIVA*1.13`

    (Un operador es un símbolo que simboliza una operación)

Literales

- Hemos visto que las variables contienen valores.
- Los literales son la forma de escribir estos valores en un programa.
- Así, por ejemplo, el carácter correspondiente a la letra a minúscula se escribe en un programa como 'a'

Literales enteros

- Los valores enteros se pueden escribir:
  - En decimal (de forma normal). 21
  - En octal (precedido por un 0). 025.
  - En hexadecimal (precedido por 0x). 0x15
- Para indicar que el número es un **`long`**, puede acabarse con una ele ("L"). En caso contrario, se considera de tipo **`int`**. Ejemplo: 21L

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Literales reales

- Los valores reales se pueden escribir:
  - En notación normal (siempre con un punto decimal). Así, 21.53, 21. ,.54 , 0.63
  - En notación exponencial (siempre con una "e"). 5e4, que quiere decir 5*10 4
- Para indicar que el número es un **`float`** se acaba con una F y si es **`double`** con una D. Si no se especifica nada, se supone que es un **`double`**.

Literales booleanos

- Sólo dos:

```
true
false
```

- Siempre en minúscula.

Literales de carácter

- Cualquier carácter Unicode encerrado entre comillas simples. **`'a'`**, **`'ñ'`**, **`'π'`**.
- También puede especificarse por su código Unicode en octal o hexadecimal. Así, **`\102`** en octal y **`\u00A3`** en hexa.

`\b`	Retroceso	`\"`	Comillas dobles
`\t`	Tabulador	`\'`	Comillas simples
`\n`	Nueva línea	`\\`	Barra invertida
`\r`	Retorno		

Literales de cadena

- Corresponden al tipo de datos **`String`**, que contiene textos (cadenas de caracteres).
- Se encierran entre comillas dobles, así

```
"Este es un ejemplo de texto"
```

- Pueden contener literales de caracteres:

```
"La palabra \"rosa\" es un nombre"
```

8. Variables. Expresiones aritméticas y lógicas.

Una expresión es un enunciado que retorna un valor

- Una expresión puede estar compuesta por:
  - Un literal `53 'a'`
  - El nombre de una variable `precio`
  - Varias expresiones unidas por operadores.
    `precioSinIVA*1.13`

    (Un operador es un símbolo que simboliza una operación)

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

8.3. Operadores

- 8.3.1. Operadores aritméticos.
- 8.3.2. Operadores lógicos.
- 8.3.3. Operadores de comparación.
- 8.3.4. Operadores de asignación.
- 8.3.5. Operadores de tipo String.
- 8.3.6. Operadores de conversión de tipos.
- 8.3.7. Otros operadores

8.3. Operadores

- 8.3.1. Operadores aritméticos.
- 8.3.2. Operadores lógicos.
- 8.3.3. Operadores de comparación.
- 8.3.4. Operadores de asignación.
- 8.3.5. Operadores de tipo String.
- 8.3.6. Operadores de conversión de tipos.
- 8.3.7. Otros operadores

Tipos

- Una expresión puede estar compuesta por:
  - Un literal `53 'a'`
  - El nombre de una variable `precio`
  - Varias expresiones unidas por operadores.
    `precioSinIVA*1.13`
    (Un operador es un símbolo que simboliza una operación)

Operadores aritméticos (enteros y reales)

+	expr1+expr2	Suma expr1 más expr2	ADITIVOS
–	expr1-expr2	Resta expr1 de expr2	
*	expr1*expr2	Multiplica expr1 por expr2	
/	expr1/expr2	Divide expr1 por expr2	MULTIPLICATIVOS
%	expr1%expr2	Calcula el resto de dividir expr1 por expr2	
+	+expr1	Hace int la expresión	
–	-expr1	Devuelve el valor de expr1 canviado de signo	UNARIOS
++	var++	Incrementa valor de var	INCREM. Y
--	var--	Decrementa valor de var	DECREM.

¿División entera o división real?

- **División entera.** El resultado es un entero que es el cociente de la división.
  4/5 = 0
- **División real.** El resultado es un real que es exactamente el valor de la división.
  4.0/5.0=0.8
- En Java si el dividendo y divisor son ambos enteros (**`short, int, long`**) es entera. Si hay un real (**`float, double`**), la división es real.

Ejercicio optativo

- Programen una aplicación Web que dado un número entero introducido en un formulario HTML, devuelva el número siguiente.
- Utilicen para ello el operador de incremento.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

palasi.com
IT and management

Ejercicio

- Escribir una aplicación Web que lea el sueldo por hora normal, el número de horas normales y extras en un mes. La aplicación debe escribir por página el salario mensual del empleado.
- Nota:
  - Las horas extras se pagan el doble que las normales
  - Utilicen sólo números enteros.

8.3. Operadores

Operadores lógicos

`!`	`!expr1`	NO lógico. Si `expr1` es `true`, retorna `false`. Si `expr1` es `false`, retorna `true`
`&&`	`expr1&&expr2`	Y lógico. Sólo retorna `true`, si las dos expresiones son `true`. Si no, retorna `false`.
`||`	`expr1||expr2`	O lógico. Si una de las dos expresiones es `true`, retorna `true`. Si no, retorna `false`.

8.3. Operadores

Operadores de comparación

`==`	`expr1==expr2`	Igual. Retorna `true` si y solo si las dos expr. tienen mismo valor. Tipo primitivo (no String).
`!=`	`expr1!=expr2`	Diferente. Ret. `true` si y solo si las 2 expr. tienen valor diferente Tipo primitivo (no String).
`<`	`expr1<expr2`	Menor. Retorna `true` si y solo si el valor de la primera expresión es menor que el de la segunda.
`<=`	`expr1<=expr2`	Menor o igual. Como el anterior, pero con menor o igual.
`>`	`expr1>expr2`	Mayor. Retorna `true` si y solo si el valor de la primera expresión es mayor que el de la segunda.
`>=`	`expr1>=expr2`	Mayor o igual. Como el anterior, pero con mayor o igual.

Operadores de comparación

Tipos de los parámetros

`==`	`expr1==expr2`	Tipo primitivo (no `String`).
`!=`	`expr1!=expr2`	Tipo primitivo (no `String`).
`<`	`expr1<expr2`	Tipo aritmético (enteros y reales).
`<=`	`expr1<=expr2`	Tipo aritmético (enteros y reales).
`>`	`expr1>expr2`	Tipo aritmético (enteros y reales).
`>=`	`expr1>=expr2`	Tipo aritmético (enteros y reales).

Tipo del valor devuelto

`boolean`

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Nota

- Es importante diferenciar la instrucción de asignación.

```
precio = precioSinIVA *1.13
```

- de la comparación de igualdad

```
(precio == precioSinIVA*1.13)
```

La sintaxis es muy parecida. El significado es muy diferente.

- La asignación es una instrucción. Es una **acción**. "Guardar en una variable un valor".
- La comparación de igualdad es una condición. Es una **expresión**. Devuelve un resultado (`true` o `false`) según si las dos partes de la comparación son iguales.

8.3. Operadores

Operadores de asignación

- Sirven para abreviar algunas expresiones comunes.

Op.	Uso	Equivalente	Descripción
`+=`	`var+=expr`	`var=var+expr`	Aumenta `var` con `expr`
`-=`	`var-=expr`	`var=var-expr`	Disminuye `var` con `expr`
`*=`	`var*=expr`	`var=var*expr`	Multiplica `var` con `expr`
`/=`	`var/=expr`	`var=var/expr`	Divide `var` con `expr`
`%=`	`var%=expr`	`var=var%expr`	Que `var` tenga el resto de dividir `var` con `expr`

Pregunta

- ¿Cómo calcularíamos el precio con IVA a partir del precio sin IVA, usando el operador de asignación `*=`?
- Nota: usen una sola variable para ambos precios.

8.3. Operadores

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Operadores de tipo `String`

- `s1.equals(s2)` Muestra si dos cadenas s1 y s2 son iguales. NO UTILIZAR ==
- `+` Concatena dos cadenas.
  "Juan "+"Carlos" → "Juan Carlos "
- Si ***var*** es una variable de tipo `String`
  `var.length()` devuelve la longitud de la cadena.
  `var.charAt(i)` devuelve carácter en la posición `i`
  `var.substring(i,j)` Devuelve la subcadena que empieza en la posición `i` y acaba en la posición `j-1`

(Atención: el primer carácter está en la posición 0)

Ejercicio

- Para llamar al extranjero desde El Salvador, uno debe marcar el carrier de la compañía telefónica (155 o 144, tres dígitos), el código internacional (00), el código del país (507, supondremos que tiene siempre tres dígitos), el código de área (que se supone de 2 dígitos) y el número del abonado (supongamos de siete dígitos).
- Escribir una aplicación Web que, dado un número telefónico completo, imprima por página el carrier, código internacional, código del país, código de área y código de abonado.

8.3. Operadores

Operador de conversión de tipos o casting (1)

- Hay tipos que tienen valores en común. Por ejemplo, `int` y `long`.
- A veces, necesitamos convertir un tipo en otro (por ejemplo el `5 int` en `5 long`).
- Hay dos tipos de conversión:
  - Conversión ascendente. Del tipo más específico al tipo más general. De `int` a `long`
  - Conversión descendente. Del tipo más general al tipo más específico. De `long` a `int`

Operador de conversión de tipos o casting (2)

- La conversión ascendente la realiza automáticamente Java.

```
<%
int enteroNormal;
long enteroLargo;
enteroNormal = 56;
enteroLargo = enteroNormal;
%>
```

- En este caso, el entero `int` se convierte automáticamente en `long`

Operador de conversión de tipos o casting (3)

- La conversión descendente debe realizarla explícitamente el programador con el operador de conversión de tipos, que tiene la sintaxis.

`(tipo)expr`

- Donde `expr` es la expresión a convertir y `tipo` el tipo al que se quiere convertir.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Operador de conversión de tipos o casting (4)

- Uso del operador de casting para hacer conversión descendente.

```
<%
int enteroNormal;
long enteroLargo;
enteroLargo = 56L;
enteroNormal = (int)enteroLargo;
%>
```

Otros operadores de conversión de tipos (1)

- El operador de conversión que hemos visto sólo funciona entre tipos relacionados (uno es un subconjunto del otro).
- Pero a veces se necesitan convertir tipos que no están relacionados.
- Para ello, se utilizan otros operadores.

Otros operadores de conversión de tipos (2)

`String.valueOf(expr)`

Convierte en tipo **`String`** las variables de tipo **`boolean`**, **`int`**, **`char`**, **`long`**, **`float`**, **`double`**.

Ejemplo:

String.valueOf(53) → "53"

Otros operadores de conversión de tipos (3)

- Para convertir las variables de tipo **`String`** en variables de tipo **`int`**, **`long`**, **`float`**, **`double`**, **`char`**, **`boolean`**.

```
Integer.parseInt(exprString)
Long.parseLong(exprString)
Float.parseFloat(exprString)
Double.parseDouble(exprString)
exprString.charAt(n)
Boolean.getBoolean(exprString)
```

Algo obvio

- Todas estas conversiones no siempre son posibles.
- Debemos estar pendiente de los datos que se le pasan son convertibles. Ejemplo:

```
Integer.parseInt("50")
```
Correcto

```
Integer.parseInt("Java")
```
Error

8.3. Operadores

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Otros operadores

- Operador condicional ternario.

```
exprbool?expr1:expr2
```

Si exprbool es **true**, devuelve el valor de expr1.
En caso contrario, devuelve el valor de expr2.

- Operadores de bits. Poco usados. No los veremos.
- Operadores relacionados con objetos. Los veremos más adelantes.

8. Variables. Expresiones aritméticas y lógicas.

- 8.1. Variables.
- 8.2. Literales.
- 8.3. Operadores.
- 8.4. Precedencia de operadores.

Una expresión es un enunciado que retorna un valor

- Una expresión puede estar compuesta por:
  - Un literal `53 'a'`
  - El nombre de una variable `precio`
  - Varias expresiones unidas por operadores.
    `precioSinIVA*1.13`
    (Un operador es un símbolo que simboliza una operación)

Si hay varios operadores, ¿cuál es el Que se aplica primero?

Precedencia de operadores

- El orden en que se aplican los operadores en una expresión es importante.
- Ejemplo. `precio = 200*4+50`
- Primero el producto.
  - 200*4=800
  - 800+50 = 850
- Primero la suma.
  - 4+50=54
  - 54*200=10800

La precedencia de operadores en Java

1. Paréntesis, de dentro a fuera.
2. Incremento y decremento (postfijos) (++,--)
3. Unarios (+, -) y NO lógico (!).
4. Operador de casting.
5. Multiplicativos (*, /, %) , de izqda. a derecha
6. Aditivos (+, -), de izquierda a derecha.
7. De comparación relacionales (<,>,<=,>=)
8. De comparación de igualdad (==,!=)
9. Y lógico (&&)
10. O lógico (||)
11. Condicional ternario (?:)
12. De asignación (=, +=,-=,*=,/=,%=).

Precedencia de operadores

- Siguiendo la lista, el producto tiene precedencia sobre la suma.

```
precio = 200*4+50
```

- Primero, el producto.
  - 200*4=800
  - 800+50 = 850
- Si quisiéramos que se evaluara de otra forma de la que establece la precedencia, utilizaríamos **paréntesis**:

```
precio = 200*(4+50)
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Programa del tema 1

- 1. Introducción al lenguaje Java.
- 2. Instalación de la plataforma Java.
- 3. Introducción a la programación Web.
- 4. Fundamentos de HTML.
- 5. Introducción a Eclipse.
- 6. Introducción a Tomcat.
- 7. Introducción a JSP.
- 8. Variables. Expresiones aritméticas y lógicas.
- 9. Sentencias algorítmicas: condicionales, ciclos.
- 10. Arreglos.

9. Estructuras de control

- 9.1. Instrucciones de generación de texto.
- 9.2. Estructuras condicionales.
- 9.3. Paréntesis: otros controles de entrada.
- 9.4. Estructuras repetitivas.
- 9.5. Sentencias de salto.

9. Estructuras de control

- 9.1. Instrucciones de generación de texto.
- 9.2. Estructuras condicionales.
- 9.3. Paréntesis: otros controles de entrada.
- 9.4. Estructuras repetitivas.
- 9.5. Sentencias de salto.

Esto es un paréntesis

- Antes de empezar con las estructuras de control, queremos hacer un paréntesis para explicar la instrucción de generación de texto.
- Esta instrucción nos será útil a partir de ahora.
- Se llama: `out.print`

La instrucción `out.print`

- Sirve para incluir un texto dentro de la página Web.
- Su sintaxis es:

```
out.print("texto");
```

- El texto se incluirá en una línea de la página Web.
- El texto puede tener etiquetas HTML si queremos darle algún tipo de formato. Así, por ejemplo:

```
out.print("Esto es <B>negrita</B>");
```

La instrucción no sólo incluye texto

- Si le quitamos las comillas, incluye el valor de cualquier expresión (de las que acabamos de ver).
- Su sintaxis es:

```
out.print(expresión);
```

- La expresión se evaluará, se convertirá en `String` y se incluirá en una línea de la página Web.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejemplo

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Dos por dos son
     <% out.print(2*2); %>. </H1>
  </BODY>
</HTML>
```

`out.print` es una instrucción y aparece en un scriptlet

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Dos por dos son
     <% out.print(2*2); %>. </H1>
  </BODY>
</HTML>
```

La JSP anterior podría haberse escrito así

```
<!DOCTYPE HTML ...>
<HTML>
   <HEAD>
     <TITLE>Eco</TITLE>
    </HEAD>
    <BODY>
    <H1>Dos por dos
     <%= 2*2 %>. </H1>
  </BODY>
</HTML>
```

Si queremos

- Que una expresión se evalúe y se incluya en una página Web, tenemos dos formas.
  - Con una expresión JSP `<%= 2*2 %>`
  - Con un out.print dentro de un scriptlet

    `<% out.print(2*2); %>`
- ¿Por qué dos en vez de una?
- A veces, nos resultará más conveniente una de ellas en cuestión de legibilidad.

Por ejemplo

- Si estamos en medio de un scriptlet, será mejor
  - Emplear `out.print`

  que
  - Cerrar el scriptlet.
  - Escribir una expresión JSP.
  - Volver a abrir el scriptlet.
- Esto lo veremos con ejemplos más adelante.

9. Estructuras de control

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Aplicaciones que hemos visto hasta ahora

- Demasiado simples
  - Secuencia de instrucciones
  - Se ejecutan una después de otra sin ninguna variación
- Siempre hacen lo mismo, independientemente de los datos que les entrábamos

A veces, nos interesa

- Que la aplicación responda diferentemente dependiendo de los datos.
- Para ello, debemos poder hacer que se ejecuten diferentes instrucciones dependiendo de los datos.

Solución: Instrucción `if`

- Sintaxis:

```
if (condicion) {
    sentencias
}
```

OJO: NO LLEVA THEN

- La condición es una expresión que evalúa a un booleano.
- Si la expresión evalúa a **`true`** (la condición se cumple), se ejecutan las sentencias.
- Si la expresión evalúa a **`false`**, no se ejecuta nada.

Un ejemplo de `if`

```
<% int precio= Integer.parseInt(
  request.getParameter("precio")); %>
<!DOCTYPE HTML ...>
   <HTML><HEAD>
     <TITLE>Mi deseo</TITLE>
     </HEAD>
     <BODY>
     <H1>
     <% if (precio>1000){
         out.print("Demasiado caro");
        }
      %>
     </H1>
   </BODY>
</HTML>
```

También podría ser

```
<% int precio= Integer.parseInt(
   request.getParameter("precio")); %>
<!DOCTYPE HTML ...>
   <HTML><HEAD>
     <TITLE>Mi deseo</TITLE>
     </HEAD>
     <BODY>
     <% if (precio>1000){ %>
       <H1>Demasiado caro</H1>
     <% } %>
   </BODY>
</HTML>
```

Fíjense que un `if` puede encerrar código HTML

```
<% int precio= Integer.parseInt(
   request.getParameter("precio")); %>
<!DOCTYPE HTML ...>
   <HTML><HEAD>
     <TITLE>Mi deseo</TITLE>
     </HEAD>
     <BODY>
     <% if (precio>1000){ %>
       <H1>Demasiado caro</H1>
     <% } %>
   </BODY>
</HTML>
```

En general, se puede mezclar código Java con HTML, pero es mejor evitarlo SI SE PUEDE.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Pregunta: ¿qué pasaría si el número es menor que 1000?

- Respondan.

Respuesta

- Presentaría una página en blanco, pues la página que generaría sería la siguiente.

```
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Mi deseo</TITLE>
   </HEAD>
   <BODY>
   </BODY>
  </HTML>
```

- Compruébenlo ejecutándolo y haciendo **Ver|Código fuente**.

Condicionales alternativos

- Como hemos visto, el condicional **`if`** nos permite ejecutar un conjunto de sentencias si se cumple una condición. En caso contrario, no hace nada.
- A veces, necesitamos que, si no se cumple la condición, se haga una cosa en específico, en vez de no hacer nada.
- Para ello, utilizaremos el condicional alternativo o **`if-else`**

Instrucción "if-else"

- Sintaxis:

```
if (condicion) {
    sentencias1
} else {
    sentencias2
}
```

OJO: NO LLEVA THEN

- La condición es una expresión booleana.
- Si la expresión evalúa a **`true`** (la condición se cumple), se ejecuta el conjunto ***sentencias1***.
- Si la expresión evalúa a **`false`**, se ejecuta el conjunto ***sentencias2***.

Pregunta: ¿Qué hace esta página? ¿Cómo se podría expresar de forma más elegante?

```
<% int precio= Integer.parseInt(
   request.getParameter("precio")); %>
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Mi deseo</TITLE>
   </HEAD>
   <BODY>
    <% if (precio>1000){ %>
     <H1>Demasiado caro</H1>
   <% } else { %>
    <H1>Lo compro</H1>
   <% } %>
  </BODY>
</HTML>
```

De forma más elegante

```
<% int precio= Integer.parseInt(
   request.getParameter("precio")); %>
<!DOCTYPE HTML ...>
  <HTML><HEAD>
    <TITLE>Mi deseo</TITLE>
   </HEAD>
   <BODY>
    <H1>
    <% if (precio>1000){
       out.print("Demasiado caro");
     } else {
       out.print("Lo compro");
     } %>
   </H1>
  </BODY>
</HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejercicio

- Dado un número que representa un mes (así 1 es enero, 2 es febrero, etc.) hacer una aplicación que escriba si el mes pertenece a la primera parte del año o a la segunda.

Condicionales anidados

- Hasta ahora, hemos considerado dos posibilidades:
  - Que se cumpla la condición.
  - Que no se cumpla.
- Pero hay problemas que necesitan distinguir entre más de dos posibilidades.
- Para ello, se coloca una estructura **`if-else`** y dentro de la rama **`else`** se coloca otra estructura **`if-else`** .
- A esto se le llama condicionales anidadas.

Condicionales anidados

- Por claridad, se suelen escribir de esta forma:

```
if (condicion1) {
     sentencias1
} else if (condicion2) {
     sentencias2
} else if...
....
} else {
    sentenciasn
}
```

Si se cumple la *condicion1*, se ejecutan las *sentencias1*.

Si se cumple la *condicion2*, se ejecutan las *sentencias2*, *etc.*

Si no se cumple ninguna cond., se ejecutan las *sentenciasn*

Condicionales anidados

- Por claridad, se suelen escribir de esta forma:

```
if (condicion1) {
     sentencias1
} else if (condicion2) {
     sentencias2
} else if...
....
} else {
    sentenciasn
}
```

Si se cumplen varias condiciones, sólo se ejecutan las sentencias de la primera condición que las cumple. El orden es impor.

Pregunta: ¿Qué hace esta página? ¿Cómo se podría expresar de forma más elegante?

```
<% int precio= Integer.parseInt(
   request.getParameter("precio")); %>
<!DOCTYPE HTML ...>
   <HTML><HEAD>
     <TITLE>Regateo</TITLE>
     </HEAD>
     <BODY>
      <% if (precio>1000){ %>
        <H1>Demasiado caro</H1>
      <% } else if (precio>800) {%>
        <H1>Rebájemelo</H1>
      <% } else { %>
        <H1>Lo compro</H1>
      <% } %>
   </BODY>
</HTML>
```

¿Qué hace esta aplicación?

```
<%int precio = Integer.parseInt(
request.getParameter("precio")); %>
<!DOCTYPE HTML ...>
   <HTML>
   <HEAD><TITLE>Regateo</TITLE></HEAD>
   <BODY>
     <% if (precio > 1000) {
          out.print("¡Demasiado caro!");
        } else if (precio > 800) {
          out.print("Rebájemelo");
        } else
          out.print("Lo compro");
        } %>
   </BODY>
</HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Solución

- **Si el precio que se pasa es mayor que 1000, sale una página**

  Demasiado caro!

- **Si está entre 800 y 1000, escribe**

  Rebájemelo

- **Si es menor o igual que 800,**

  ¡Lo compro!

- **Despliegen la JSP para comprobarlo**

Ejercicio

- En una empresa hay tres clases de clientes: A, B y C. A los primeros se les hace un descuento del 10%, a los segundos del 20% y a los terceros de 30%.
- Queremos tener una aplicación Web que se le introduzca el tipo de cliente y nos escriba el descuento que se le aplica a este tipo de cliente. Hacerlo lo más robusto posible.
- Utilicen la función **`equals`** de cadenas.

Condicional `switch`

- Es una estructura de selección múltiple, como los condicionales anidados que acabamos de ver.
- Puede contemplar tantas opciones como se desee.
- La diferencia es que la elección de las sentencias que se ejecutan depende del valor de una expresión que se evalúa.

Instrucción "switch"

- Sintaxis:

```
switch (expresion) {
   case valor1:
      sentencias1
      break;
   case valor2:
      sentencias2
      break;
   ...
   default:
      sentenciasn
      break;
}
```

- Si la expresión evalúa al *valor1*, se ejecutan las *sentencias1*.
- Si la expresión evalúa al *valor2*, se ejecutan las *sentencias2*.
- Si el valor no es contemplado por ningún case, se ejecuta *sentenciasn* (`default`)
- La parte `default` es opcional.

Instrucción "switch"

- Sintaxis:

```
switch (expresion) {
   case valor1:
      sentencias1
      break;
   case valor2:
      sentencias2
      break;
   ...
   default:
      sentenciasn
      break;
}
```

- *valor1*, *valor2*... deben ser literales.
- Tanto la expresión como *valor1*, *valor2*... deben ser de tipo `byte`, `char`, `short` e `int`.

Nota

- A partir de ahora, no incluiremos en cada JSP la expresión

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.0
Transitional//EN">
```

- Pues no cabe en las transparencias. Pero se supone que está allí.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejemplo

```
<%int dia = Integer.parseInt(
request.getParameter("dia")); %>
<HTML><HEAD><TITLE>Semana</TITLE></HEAD>
<BODY>
    <% switch (dia) {
      case 1:
         out.print("Domingo");break;
           ...
      case 7:
         out.print("Sábado");break;
      default:
         out.print("No día de semana");
    }%>
   </BODY>
</HTML>
```

Ejemplo

```
<%int mes = Integer.parseInt(
request.getParameter("mes")); %>
<HTML><HEAD><TITLE>Semana</TITLE></HEAD>
<BODY><% switch (mes) {
      case 4:
      case 6:
      case 8:
      case 10:
         out.print("30 días");break;
      case 2:
         out.print("28 días");break;
      default:
         out.print("31 días");}%>
   </BODY>
</HTML>
```

Una misma sentencia se ejecuta con valores diferentes

Atención

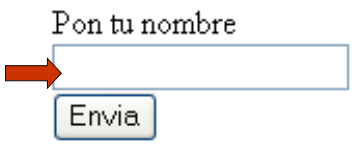
- El condicional **`switch`** sólo se puede utilizar con tipos de datos **`byte`**, **`char`**, **`short`** e **`int`**.
- Para otros tipos de datos (por ejemplo, **`float`**, **`double`**, **`boolean`** y **`String`**) hay que utilizar condicionales anidados.
- Todo lo que puede hacerse con **`switch`** puede hacerse con condicionales anidados (pero no al revés).

Ejercicio

- Tenemos un semáforo inteligente que cambia de color según un código que genera el centro de control. Si el código es 0, el semáforo se pone en verde. Si el código es 1, el semáforo se pone en amarillo. Si el código es 2, el semáforo se pone en rojo.
- Escribir una aplicación Web que use una sentencia **`switch`**, que, dado el número de un color, imprima el nombre del color correspondiente. Hacer la aplicación lo más robusta posible.

9. Estructuras de control

Hasta ahora sólo hemos usado cuadros de texto para introducir los datos.

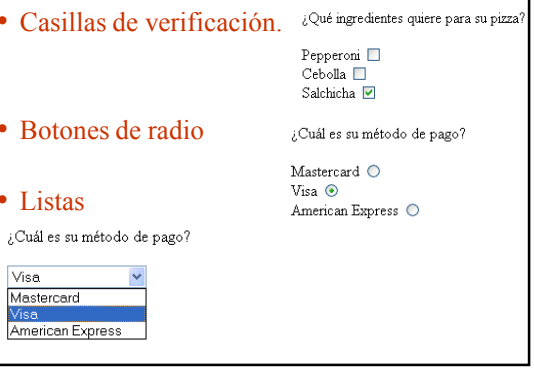


- Sin embargo, hay otros controles que nos permiten introducir datos.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Otros controles de introducción de datos que se pueden usar

- Casillas de verificación.
- Botones de radio
- Listas



Para obtener los datos de un cuadro de texto desde la JSP

- Sabemos que se usa

```
request.getParameter("nombreCuadro")
```

- ¿Qué se usa para obtener los datos de los otros controles de introducción?

Para los botones de radio

- Código HTML:

```
<INPUT TYPE="RADIO" NAME="
  NombreGrupo"
  VALUE="NombreBotón">
```

- Obtener el nombre del botón seleccionado del grupo con

```
request.getParameter("nombreGrupo")
```

Botones de radio: ejemplo anterior

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
<HEAD><TITLE>Botones de radio</TITLE><HEAD>
<BODY>
  <FORM>
    <P>¿Cuál es su método de pago?
    <P>Mastercard <INPUT TYPE="RADIO"
 NAME="Tarjeta" VALUE="Mastercard"><BR>

 Visa <INPUT TYPE="RADIO" NAME="Tarjeta"
 VALUE="Visa" CHECKED> <BR>

 American Express <INPUT TYPE="RADIO"
 NAME="Tarjeta" VALUE="American Express">
      </FORM>
  </BODY></HTML>
```

En este caso

- Para saber la tarjeta de crédito seleccionada será:

```
request.getParameter("Tarjeta")
```

- Por ejemplo, se podría hacer algo así

```
String tarjeta =
  request.getParameter("Tarjeta");
if (tarjeta.equals("Visa")){
     numeroTarjeta = 1;
} else if (tarjeta.equals("Mastercard")) {
     numeroTarjeta = 2;
} else {
     numeroTarjeta = 3;
}
```

Para las casillas de verificación

- Código HTML:

```
<INPUT TYPE ="CHECKBOX"
  NAME="nombreCasilla">
```

- Obtener si la casilla de verificación está activada o no

```
request.getParameter("nombreCasilla")
!= null
```

- Devuelve `true` si está activada y `false` si no lo está.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Casillas de verificación: ejemplo anterior

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
  Transitional//EN">
<HTML>
<HEAD><TITLE>Casillas de
  verificacion</TITLE><HEAD>
<BODY>
  <FORM>
    <P>¿Qué ingredientes quiere para su
  pizza?    <P>Pepperoni <INPUT
  TYPE="CHECKBOX"  NAME="pepperoni"><BR>

  Cebolla <INPUT TYPE="CHECKBOX"
  NAME="cebolla"> <BR>

  Salchicha <INPUT TYPE="CHECKBOX"
  NAME="salchicha" CHECKED>
      </FORM>
   </BODY></HTML>
```

En este caso

- Para saber si se ha elegido cebolla:

```
request.getParameter("cebolla")!=null
```

- Por ejemplo, se podría hacer algo así

```
int precio = 10;
if (request.getParameter("cebolla")!= null){
  precio += 5;
}
if (request.getParameter("pepperoni")!= null){
  precio += 8;
}
if (request.getParameter("salchicha")!= null){
  precio +=10;
}
```

Listas

- Código HTML para una lista:

```
<SELECT NAME="nombreLista">
<OPTION VALUE="ValorOpción1">TextoOpción1
<OPTION VALUE="ValorOpción2">TextoOpción2
...
<OPTION VALUE="ValorOpciónn">TextoOpciónn
</SELECT>
```

- Para obtener el valor de la opción seleccionada se usa

```
request.getParameter("nombreLista")
```

Listas: ejemplo anterior

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
   Transitional//EN">
<HTML>
<HEAD><TITLE>Listas</TITLE><HEAD>
<BODY>
 <FORM>
<P>¿Cuál es su método de pago?
<P> <SELECT NAME="Tarjeta">
    <OPTION VALUE="Mastercard">Mastercard
    <OPTION VALUE="Visa" SELECTED>Visa
    <OPTION VALUE="American Express">American
  Express
  </SELECT>
  </FORM>
 </BODY></HTML>
```

En este caso

- Para saber la tarjeta de crédito seleccionada será:

```
request.getParameter("Tarjeta")
```

- Por ejemplo, se podría hacer algo así

```
String tarjeta =
  request.getParameter("Tarjeta");
if (tarjeta.equals("Visa")){
     numeroTarjeta = 1;
} else if (tarjeta.equals("Mastercard")) {
     numeroTarjeta = 2;
} else {
     numeroTarjeta = 3;
}
```

Ejercicio

- Habíamos creado un archivo llamado **`saludar.html`** que producía este formulario

- Programar una JSP que escriba todas las opciones que hemos seleccionado.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Solución

```
<% String nombre =
  request.getParameter("nombre");
String asistencia;
if (request.getParameter("asistente")!=null){
  asistencia = "Es asistente al curso";
} else {
  asistencia = "No es asistente al curso";}
String residencia =
  request.getParameter("residencia");
String edad = request.getParameter("edad");%>
<!DOCTYPE...>
<HTML><HEAD><TITLE>Saludo</TITLE><HEAD><BODY>
Usted se llama <%=nombre%>.<br><%=asistencia%>
<br>Usted vive en <%=residencia%><br>
Su edad es <%=edad%></BODY></HTML>
```

9. Estructuras de control

Estructuras repetitivas

- Hasta ahora, las aplicaciones que hemos visto ejecutan cada instrucción una sola vez como máximo.
- Esto puede no ser conveniente para ciertos tipos de aplicación.
- Ejemplo: aplicación Web que imprime por página cien dígitos 1 seguidos.

Posible solución

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <%
    out.print("1");
    out.print("1");
    out.print("1");

     ...
    out.print("1"); }%>
   </BODY>
</HTML>
```

Problema

- No es práctica una aplicación tan larga.
- Este problema es general. Por ejemplo, escribir la planilla de mil empleados de una empresa.
- Lo que se necesita es que el lenguaje provea instrucciones que permitan repetir una misma tarea (un mismo bloque de sentencias) muchas veces.

Repitiendo sentencias

- Existen tres instrucciones en Java que permiten repetir bloques de sentencias
- Instrucción **`while`**
- Instrucción **`do-while`**
- Instrucción **`for`**
- A estas intrucciones se les puede llamar **estructuras repetitivas, bucles o ciclos.**

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Repitiendo sentencias

- Existen tres instrucciones en Java que permiten repetir bloques de sentencias
- Instrucción **`while`**
- Instrucción **`do-while`**
- Instrucción **`for`**
- A estas intrucciones se les puede llamar **estructuras repetitivas, bucles o ciclos.**

Sintaxis de la instrucción `while`

```
while (condicion) {
  sentencias
}
```

- ***condicion*** es una expresión que evalúa a **`boolean`**
- ***sentencias*** es el bloque de sentencias que debe repetirse (se le llama cuerpo del bucle).

Semántica de la instrucción `while`

```
while (condicion) {
  sentencias
}
```

- Mientras se cumple ***condicion***, se repiten ***sentencias.***

Ejecución de la instrucción *`while`*

```
while (condicion)
  sentencias
}
```

1
2

- La primera vez que se ejecuta el bucle, se evalúa la ***condicion***.
- Si evalúa a **`false`**, se salta el bucle.
- Si evalúa a **`true`**, se ejecutan ***sentencias*** y se vuelve a evaluar ***condicion***.
- Si evalúa a **`false`**, se salta el bucle.
- Si evalúa a **`true`** se ejecutan ***sentencias*** y se vuelve a evaluar ***condicion***.
- **Y así sucesivamente.**

Ejecución de la instrucción *`while`*

```
while (condicion) {
  sentencias
}
```

es equivalente a

```
if (condicion) {
  sentencias
  if (condicion) {
    sentencias
    if (condicion) {
      sentencias
      ...
    }
  }
}
```

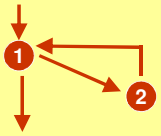
Posible solución al problema anterior

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <% int i = 1;
    while (i <= 100) {
       out.print("1");
       i++;
    } %>
   </BODY>
</HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Posible solución al problema anterior

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <% int i = 1;
    while (i <= 100) {
       out.print("1");
       i++;
    } %>
   </BODY>
</HTML>
```



- Se inicializa i con 1.
- i==1, 1<=100, true, impr. 1, incrementa i
- i==2, 2<=100, true, impr. 1, incrementa i
- ...
- i==100, 100<=100, true, impr.1, incr. i
- i==101, 101<=100, false, acaba bucle

100 veces

Partes de un bucle

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <% int i = 1;
    while (i <= 100) {
       out.print("1");
       i++;
    } %>
   </BODY>
</HTML>
```

INICIALIZACIÓN
CONDICIÓN
CUERPO (sentencias)

- Cada vez que se ejecuta el cuerpo del bucle, se le llama **iteración o pasada** del bucle.
- Cuando un bucle deja de ejecutarse porque la condición evalúa a **false**, se dice que la aplicación **sale** del bucle.

Esquema de un bucle

```
inicialización
while (condicion) {
   cuerpo
}
```

- con más detalle:

```
inicialización
while (condicion) {
   tratar el elemento actual
   obtener elemento siguiente
}
instrucciones finales (opcional)
```

Esquema de un bucle

```
int i = 1;
while (i <= 100) {
   out.print("1");
   i++;
}
```

INICIALIZACIÓN
CONDICIÓN
Tratar el. actual
Obtener siguiente elemento
Instrucciones finales (no existen)

- Pregunta: ¿Qué pasaría si omitiéramos la parte de obtener el siguiente elemento?

Solución a la pregunta anterior

```
int i = 1;
while (i <= 100) {
   out.print("1");
}
```

- i==1, true, imprime 1
- i==1, true, imprime 1
- ...
- Y así hasta el infinito, **bucle infinito** ("se traba")

Ejercicio

- Escribir una aplicación Web que escriba por página la suma de los n primeros números naturales, donde n es un número que entra el usuario.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejercicio optativo

- La secuencia de Fibonacci se define de esta manera. El primer término de la secuencia es 1, el segundo 1 y, a partir del tercero, cada término es la suma de los dos anteriores. En términos matemáticos, esto se escribe así:
- $a_1=1$
- $a_2=2$
- $a_n=a_{n-1}+a_{n-2}$ para n>=3
  Así, los primeros términos de la secuencia de Fibonacci son los siguientes:
- 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377
- Escribir una aplicación Web que, dado un número *n*, obtenga el término número *n* de la secuencia de Fibonacci.

Solución del ejercicio anterior

```
<%int n = Integer.parseInt(
request.getParameter("n"));
int i=1; int suma=0;
while (i<=n) {
     suma+=i;
     i++;
}%>
<HTML><HEAD><TITLE>Semana</TITLE></HEAD>
<BODY>
    La suma de los <%= n %> primeros
naturales
es <%= suma %>.
</BODY>
</HTML>
```

Pregunta

- ¿Qué pasaría si introdujéramos 0 en esta aplicación Web?
- Queremos saber la suma de los 0 primeros naturales.

Pasando 0 como entrada

```
<%int n = Integer.parseInt(
request.getParameter("n"));
int i=1; int suma=0;
while (i<=n) {
     suma+=i;
     i++;
}%>
<HTML><HEAD><TITLE>Semana</TITLE></HEAD>
<BODY>
    La suma de los <%= n %> primeros
naturales
es <%= suma %>.
</BODY>
</HTML>
```

n es 0. La condición (1<0) evalúa a **false** . El bucle no se ejecuta. Se escribe 0 como resultado.

El resultado de la aplicación Web sería 0

- Este es el resultado correcto. La suma de los 0 primeros naturales es 0.
- ¿Qué lecciones podemos sacar de este caso?

Lecciones que podemos obtener

- 1. El bucle **while** puede ejecutarse 0, 1 o varias veces (se dice que puede tener 0, 1 o varias iteraciones)
- 2. Si la aplicación está bien diseñada, el caso en que el bucle no se ejecute (0 iteraciones) se comportará bien.
- 3. Esto es muy bueno en la mayoría de casos pero algunas pocas veces es un inconveniente.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

A veces es inconveniente el `while`

- A veces, se necesita que el bucle se ejecute como mínimo una vez.
- Por ejemplo, en operaciones sobre arrays (aún no vistos).
- Esto es una minoría de veces, pero a veces se necesita.
- Esto es debido a que **`while`** no está diseñado para casos en que el bucle se debe ejecutar una vez como mínimo.
- Para ello existe el **`do-while`**.

`do-while`, una nueva estructura repetitiva

Sintaxis

```
do {
   sentencias
} while (condicion);
```

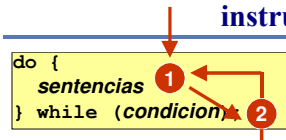
- ***condicion*** es una expresión que evalúa a **`boolean`**
- ***sentencias*** es el bloque de sentencias que debe repetirse (el cuerpo del bucle).

Semántica de la estructura *`do-while`*

```
do {
   sentencias
} while (condicion);
```

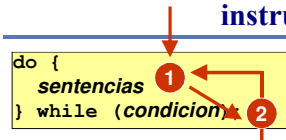
- "Se repiten las sentencias mientras se cumpla la condición"
- Al contrario de la instrucción **`while`**, la condición se evalúa DESPUÉS de ejecutarse el cuerpo del bucle.
- Esto hace que el bucle como mínimo se ejecute una vez.

Ejecución de la instrucción *`do-while`*



- Se ejecutan las **`sentencias`** y se evalúa la ***condicion***.
- Si evalúa a **`false`**, se acaba el bucle.
- Si evalúa a **`true`**, se ejecutan las ***sentencias*** y se vuelve a evaluar ***condicion***.
- Si evalúa a **`false`**, se salta el bucle.
- Si evalúa a **`true`** se ejecutan las ***sentencias*** y se vuelve a evaluar ***condicion***.
- **Y así sucesivamente.**

Ejecución de la instrucción *`do-while`*



- Se ejecutan las **`sentencias`** y se evalúa la ***condicion***.
- Si evalúa a **`false`**, se acaba el bucle.

Las **`sentencias`** se ejecutan al menos una vez

- Si evalúa a **`false`**, se salta el bucle.
- Si evalúa a **`true`** se ejecutan las ***sentencias*** y se vuelve a evaluar ***condicion***.
- **Y así sucesivamente.**

Esquema de un bucle *`do-while`*

```
inicialización
do {
   tratar el elemento actual
   obtener elemento siguiente
} while (condicion);
instrucciones finales (opcional)
```

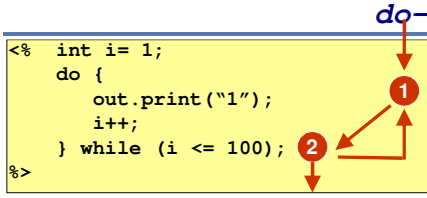
Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejecución de la instrucción *do-while*

```
<%  int i= 1;
    do {
       out.print("1");
       i++;
    } while (i <= 100);
%>
```

Ejecución de la instrucción *do-while*

```
<%  int i= 1;
    do {
       out.print("1");
       i++;
    } while (i <= 100);
%>
```



- Se inicializa `i` con valor `1`
- Impr. 1, increm. i, `i==2,2<=100,true`
- Impr. 1, increm. i, `i==3,3<=100,true`, ...
- Impr. 1, increm.i, `i==100,100<=100,true`,
- Impr.1, increm.i, `i==101,101<=100,false`, acaba

100 veces

Solución del ejercicio anterior con *do-while*

```
<%int n = Integer.parseInt(
request.getParameter("n"));
int i=1; int suma=0;
do {
 suma = suma+i
 i++;
} while (i<=n);%>
<HTML><HEAD><TITLE>Semana</TITLE></HEAD>
<BODY>
La suma de los <%= n %> primeros
naturales es <%= suma %>.
</BODY>
</HTML>
```

El problema es que no funciona bien con 0 (daría 1), pues siempre ejecuta 1 iteración.

Solución del ejercicio anterior con *do-while*

```
<%int n = Integer.parseInt(
request.getParameter("n"));
int i=1; int suma=0;
do {
 suma = suma+i
 i++;
} while (i<=sumandos);%>
<HTML><HEAD><TITLE>Semana</TITLE></HEAD>
<BODY>
    La suma de los <%= n %> primeros
naturales
es <%= suma %>.
</BODY>
</HTML>
```

Habría que hacer un **`if-else`** que tratara el caso 0 separadamente. El código sería más complicado.

Conclusión

- Utilizamos la estructura **`while`** cuando hay casos en que el cuerpo del bucle puede no ejecutarse (la mayoría de los casos).
- Utilizamos **`do-while`**, cuando el cuerpo del bucle se ejecuta como mínimo una vez (una pequeña parte de los casos).

Ejercicio

- El factorial de un número *n* (que se representa por *n!*) es el producto de todos los números naturales desde uno hasta n. Es decir,

  $n! = 1 * 2 * 3 * ... * n$

- Escribir una aplicación que dado un número *n* (que debe ser mayor o igual que 1), escriba por página el factorial de *n*. Hacer dos versiones: una con **`do`** y otra con **`while`**. ¿Cuál es preferible?

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Esquema de un bucle *while*

```
inicialización
while (condicion) {
   tratar el elemento actual
   obtener elemento siguiente (actualización)
}
instrucciones finales (opcional)
```

- Vamos a prescindir de las instrucciones finales. A la acción de obtener el elemento siguiente le llamaremos "actualización"

Esquema de un bucle *while*

```
inicialización
while (condicion) {
   tratar el elemento actual
   actualización
}
```

- A veces, la inicialización, la condición y la actualización son sencillas, están muy relacionadas y conviene agruparlas para mayor claridad.

Inicialización, condición y actualización relacionadas

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <% int i = 1;
    while (i <= 100) {
       out.print("1");
       i++;
    } %>
   </BODY>
</HTML>
```

- La inicialización asigna a la variable `i` su valor inicial.
- La actualización incrementa la variable `i`.
- La condición comprueba que `i` llega a su valor final.

Inicialización, condición y actualización relacionadas

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <% int i = 1;
    while (i <= 100) {
       out.print("1");
       i++;
    } %>
   </BODY>
</HTML>
```

- Las tres acciones tratan de la variación del valor variable i: están muy relacionadas. Además, son sencillas.
- Sería conveniente tenerlas agrupadas, pero están dispersas por el código.

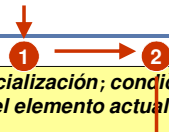
Estructura *for*

Sintaxis

```
for (inicialización; condicion; actualización) {
   tratar el elemento actual
}
```

- ***inicialización*** es una sentencia
- ***actualización*** es otra sentencia
- ***condicion*** es una expresión que evalúa a `boolean`
- ***tratar el elemento actual*** es un bloque de sentencias
- Cualquiera de estas cuatro puede dejarse en blanco.

Semántica de la estructura *for*

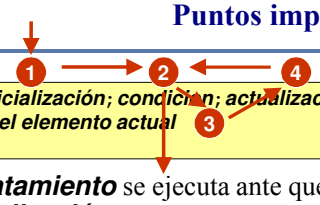



```
for (inicialización; condicion; actualización) {
   tratar el elemento actual
}
```

- Se ejecuta la sentencia de ***inicialización*** al principio del bucle.
- Se evalúa la ***condicion***. Si es `false`, se acaba el bucle. Si es `true`, se ejecuta el ***tratamiento*** y la ***actualización***.
- Se evalúa la ***condicion***. Si es `false`, se acaba el bucle. Si es `true`, se ejecuta el ***tratamiento*** y la ***actualización***.
- Y así sucesivamente.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Puntos importantes




```
for (inicialización; condicion; actualización) {
   tratar el elemento actual
}
```

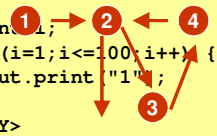
- El ***tratamiento*** se ejecuta ante que la ***actualización***.
- La ***condicion*** se evalúa al principio del bucle, como en el **`while`** y al contrario de **`do-while`**.
- Por lo tanto, puede ser que el cuerpo del bucle no se ejecute (0 iteraciones)

Ejemplo del bucle *for*

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <% int i;
    for (i=1;i<=100;i++) {
       out.print("1");
    }%>
   </BODY>
</HTML>
```

Ejemplo del bucle `for`

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <% int i;
    for (i=1;i<=100;i++) {
       out.print("1");
    }%>
   </BODY>
</HTML>
```

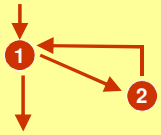


- 1. Se inicializa `i` con valor `1`
- `i==1`, `true`, imprime 1, incrementa i
- `i==2`, `true`, imprime 1, incrementa i ...
- `i==100`, `true`, imprime 1, incrementa i
- `i==101`, `false`, salta bucle, acaba

100 veces

Esta ejecución es idéntica a la de la estructura `while`

```
<HTML><HEAD><TITLE>Muchos unos</TITLE></HEAD>
<BODY>
    <% int i = 1;
    while (i <= 100) {
       out.print("1");
       i++;
    } %>
   </BODY>
</HTML>
```



- `Se inicializa i con 1.`
- `i==1,1<=100,true`, impr. 1, incrementa i
- `i==2,2<=100,true`, impr. 1, incrementa i ...
- `i==100,100<=100,true`, impr.1, incr. i
- `i==101,101<=100,false`, acaba bucle

100 veces

La estructura `for` es equivalente a la `while`

```
for (inicialización; condicion; actualización) {
   tratar el elemento actual
}
```

es equivalente a

```
inicialización
while (condicion) {
   tratar el elemento actual
   actualización
}
```

Entonces, ¿qué motivo tiene tener dos estructuras que hacen lo mismo?

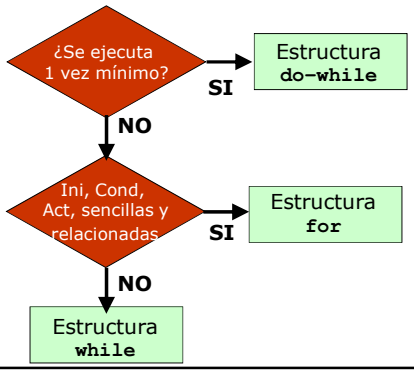
El motivo de la estructura *for*

```
for (inicialización; condicion; actualización) {
   tratar el elemento actual
}
```

- El motivo es por claridad. Cuando la inicialización, la condición y actualización son sencillas y están relacionadas se usa **`for`**.
- En caso contrario, se usa **`while`**.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Uso de las estructuras repetitivas



Algunas peculiaridades de la estructura *for* (1)

```
for (int i=1;i<=100;i++) {
  out.print("1");
}
```

- En la parte de inicialización se puede declarar una variable.
- Esta variable sólo queda declarada dentro del bucle. No existe fuera de él.

Algunas peculiaridades de la estructura *for* (2)

```
for (i=1, j=1;i<=3;i++,j++) {
  out.print(i+"-"+j);

}
```

- En la parte de inicialización y de actualización pueden colocarse varias sentencias (separadas por coma) que se ejecutan secuencialmente.
- Es posible pero es más recomendable usar un **while** en estos casos

Ejercicios

- Dada una cadena, escribir la cadena inversa por página. Así, por ejemplo, "Juan" debe producir "nauJ". Utilizar un bucle **for** y las funciones sobre cadenas **charAt** y **length**.

Solución

```
<%
String cadena = request.getParameter("cadena");
String inversa ="";
for (int i=cadena.length(); i>=0; i--){
  inversa += String.valueOf(charAt(i));
}%>
<HTML>
<HEAD><TITLE>Cadena inversa</TITLE></HEAD>
<BODY>
La cadena inversa de <%=cadena%> es <%=inversa%>
</BODY>
</HTML>
```

Bucles anidados

- Es cuando un bucle está dentro de otro.
- Se llaman también bucles imbricados o "embedded loops".
- Se utilizan cuando se quiere hacer un recorrido de los datos por varias dimensiones.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Ejemplo de bucles anidados

```
<HTML>
<HEAD><TITLE>Casillas de ajedrez</TITLE></HEAD>
<BODY>
<%
for (int fila = 1; fila<=8;  fila++){
  for (int col = 1; col<=8; col++){
    out.print("["+fila+","+col+"] ");
  }
  out.print();
}%>
</BODY>
</HTML>
```

Salida por la página de la aplicación Web anterior

```
[1,1] [1,2] [1,3] [1,4] [1,5] [1,6] [1,7] [1,8]
[2,1] [2,2] [2,3] [2,4] [2,5] [2,6] [2,7] [2,8]
[3,1] [3,2] [3,3] [3,4] [3,5] [3,6] [3,7] [3,8]
[4,1] [4,2] [4,3] [4,4] [4,5] [4,6] [4,7] [4,8]
[5,1] [5,2] [5,3] [5,4] [5,5] [5,6] [5,7] [5,8]
[6,1] [6,2] [6,3] [6,4] [6,5] [6,6] [6,7] [6,8]
[7,1] [7,2] [7,3] [7,4] [7,5] [7,6] [7,7] [7,8]
[8,1] [8,2] [8,3] [8,4] [8,5] [8,6] [8,7] [8,8]
```

- Como ven el bucle de fuera controla la fila y el bucle de dentro controla la columna.

Ejercicio

- Escribir una aplicación Web que, imprima los resultados de las tablas de multiplicar del 1 al 5. Así el aplicación debe producir algo como esto:

1,	2,	3,	4,	5,	6,	7,	8,	9,	10
2,	4,	6,	8,	10,	12,	14,	16,	18,	20
3,	6,	9,	12,	15,	18,	21,	24,	27,	30
4,	8,	12,	16,	20,	24,	28,	32,	36,	40
5,	10,	15,	20,	25,	30,	35,	40,	45,	50

9. Estructuras de control

Estructuras de salto

- Son instrucciones que permiten interrumpir el flujo de control normal de un bucle según la voluntad del programador.
- Son dos:
  - La instrucción **`break`**
  - La instrucción **`continue`**

Estructuras de salto

- Son instrucciones que permiten interrumpir el flujo de control normal de un bucle según la voluntad del programador.
- Son dos:
  - La instrucción **`break`**
  - La instrucción **`continue`**

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

La instrucción *break*

- Cuando se ejecuta sale del bucle actual.
- Ejemplo: Aplicación Web que busca la primera 'a' en una cadena introducida por el usuario.

Solución al ejemplo de la instrucción break

```
<% int i = 0;
String cadena n = request.getParameter("cadena");
   while (i<=cadena.length()-1){
      if (cadena.charAt(i) =='a') {
          break;}
      i++;
    }%><HTML>
<HEAD><TITLE>Buscando as</TITLE></HEAD>
<BODY>
   <% if (i > cadena.length()-1){
      out.print("No hay ninguna a");
    } else {
      out.print("A en posición"+ i);
    }%> </BODY></HTML>
```

Prueba

- Despliegen el ejemplo anterior para comprobar que es lo que hace

Solución al ejemplo de la instrucción break

Si encuentra la 'a' sale del bucle

Si no, sale cuando acaba la cadena

```
<% int i = 0;
String cadena n = request.getParameter("cadena");
   while (i<=cadena.length()-1){
      if (cadena.charAt(i) =='a') {
          break;}
      i++;
    }%><HTML>
<HEAD><TITLE>Buscando as</TITLE></HEAD>
<BODY>
   <% if (i > cadena.length()-1){
      out.print("No hay ninguna a");
    } else {
      out.print("A en posición"+ i);
    }%> </BODY></HTML>
```

La instrucción break es poco legible

Si encuentra la 'a' sale del bucle

Si no, sale cuando acaba la cadena

```
<% int i = 0;
String cadena n = request.getParameter("cadena");
   while (i<=cadena.length()-1){
      if (cadena.charAt(i) =='a') {
          break;}
      i++;
```

- Como vemos ahora hay dos condiciones de salida dispersas por el bucle, **que puede ser muy largo.**
- Esto es poco legible y se presta a errores de comprensión y mantenimiento

La instrucción break no es necesaria

```
   while (i<=cadena.length()-1){
     if (cadena.charAt(i) =='a') {
         break;
     }
     i++;
   }
```

- se podría haber escrito de esta forma

```
while (i<=cadena.length()-1 &&
       cadena.charAt(i) !='a') {
   i++;
}
```

- Más legible, pues la condición de salida del bucle queda explícita.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

La instrucción `break` no es necesaria

```
while (i<=cadena.length()-1){
  if (cadena.charAt(i) =='a') {
      break;
  }
  i++;
}
```

- También se habría podido expresar en un **`for`**

```
for (i=0;i<=cadena.length()-1 &&
      cadena.charAt(i) !='a';i++) {    }
```

- Aunque esto es menos legible que el **`while`** es preferible al **`break`**

Prueba

- Sustituyan en la aplicación anterior la instrucción **`break`** por el bucle **`while`** modificado y comprueben que funciona.

Conclusión

- Hemos visto que:
  - La instrucción **`break`** es poco elegante y dificulta la comprensión y, por lo tanto, el mantenimiento del código.
  - La instrucción **`break`** no es necesaria, pues hay otras formas más elegantes de hacer las cosas.
- Por eso, se aconseja no usar esta instrucción (con excepción de la estructura **`switch`** donde es necesaria)

Instrucciones de salto

- Son instrucciones que permiten interrumpir el flujo de control normal de un bucle según la voluntad del programador.
- Son dos:
  - La instrucción **`break`**
  - La instrucción **`continue`**

La instrucción `continue`

- Cuando se encuentra,
  - se deja ejecutar el resto del bucle
  - se evalúa de nuevo la condición
  - si esta es **`true`**, se continua con la siguiente iteración.
- Ejemplo: Aplicación Web que imprime todas las letras que no sean 'a' en una cadena introducida por el usuario.

Solución al ejemplo de la instrucción `continue`

```
<%int i = 0; char letra;
String cadena n = request.getParameter("cadena");
%><HTML><HEAD><TITLE>Buscando as</TITLE></HEAD>
<BODY><%
while (i<=cadena.length()-1){
  letra = cadena.charAt(i);
  i++;
  if (letra=='a') {
     continue;
  }
  out.print(letra);
}
%>
</BODY></HTML>
```

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Prueba

- Despliegen el ejemplo anterior para comprobar que es lo que hace

Solución al ejemplo de la instrucción `continue`

```
<%int i = 0; char letra;
String cadena n = request.getParameter("cadena");
%><HTML><HEAD><TITLE>Buscando as</TITLE></HEAD>
<BODY><%
while (i<=cadena.length()-1){
  letra = cadena.charAt(i);
  i++;
  if (letra=='a') {
     continue;
  }
  out.print(letra);
}
%>
</BO
```

Si la letra es 'a', salta a la siguiente iteración y no la imprime con el print.

La instrucción `continue` no es elegante

- Se necesita un razonamiento complicado para comprender qué es el que hace el bucle.
- Cuando más complicado el bucle, menos legible

```
while (i<=cadena.length()-1){
  letra = cadena.charAt(i);
  i++;
  if (letra=='a') {
     continue;
  }
  out.print(letra);
}
%>
</BO
```

Si la letra es 'a', salta a la siguiente iteración y no la imprime con el print.

La instrucción `continue` no es necesaria

```
while (i<=cadena.length()-1){
  letra = cadena.charAt(i);
  i++;
  if (letra=='a') {
    continue;
  }
  out.print(letra);
}
```

es equivalente al siguiente

```
while (i<=args[0].length()-1){
   letra = cadena.charAt(i);
   if (letra=!'a') {
     out.print(letra);
   }
   i++;
}
```

La instrucción `continue` no es necesaria

- Este último bucle es más claro, más legible y más fácil de mantener.
- No necesita razonamientos sobre las iteraciones

```
while (i<=cadena.length()-1){
   letra = cadena.charAt(i);
   if (letra=!'a') {
     out.print(letra);
   }
   i++;
}
```

Prueba

- Sustituyan en la aplicación anterior la instrucción **`continue`** por el bucle **`while`** modificado y comprueben que funciona.

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com

Conclusión

- Hemos visto que:
  - La instrucción **`continue`** es poco elegante y dificulta la comprensión y, por lo tanto, el mantenimiento del código.
  - La instrucción **`continue`** no es necesaria, pues hay otras formas más elegantes de hacer las cosas.
- Por eso, se aconseja no usar esta instrucción en nuestras aplicaciones

También existen `break` y `continue` etiquetados

- No los vamos a explicar: son aún peores que el **`break`** y **`continue`** sin etiqueta que hemos visto.
- El problema de estas dos instrucciones es que realizan un salto en el código sin ser estructuras. Se parecen al **`goto`** (que no existe en Java).
- La conclusión es que no vamos a utilizar ninguna de las dos.

Sobre el profesor

Dr. Vicent-Ramon Palasí Lallana.

- **Consultas y dudas a:**
  Teléfono: 275-4254.
  E-mail: vpalasi@aurumsol.com
  Web: www.aurumsol.com

Vicent Palasí, PhD, MBA, MEd. Todos los derechos reservados.
Mail: palasi@palasi.com Web: www.palasi.com