CC5002 - Desarrollo de Aplicaciones Web

Clase 3: Servlets y Tomcat

Plan de clases proyectado (1era Mitad)

1. Intro
2. HTTP y HTML
3. Servlets y Tomcat
4. JSP y Threads
5. Sesiones y Cookies
6. Bases de datos
7. JavaBeans
8. JSP
9. MVC
10. JSTL
11. Seguridad
12. CSS
13. Javascript
14. AJAX
15. Control

Esquema de Hoy

- Preguntas
- Web dinamica
- Servlets
- Tomcat
- Tarea 2

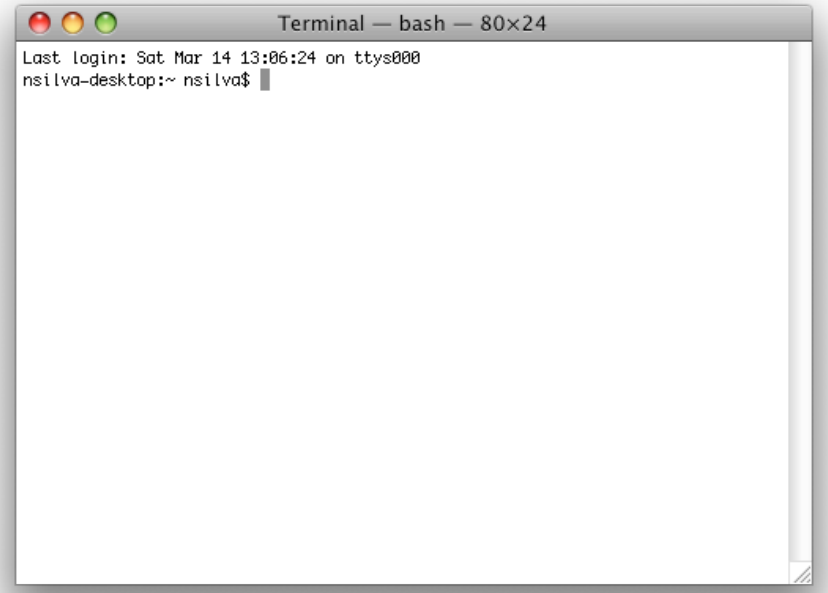
Pregunta para uds

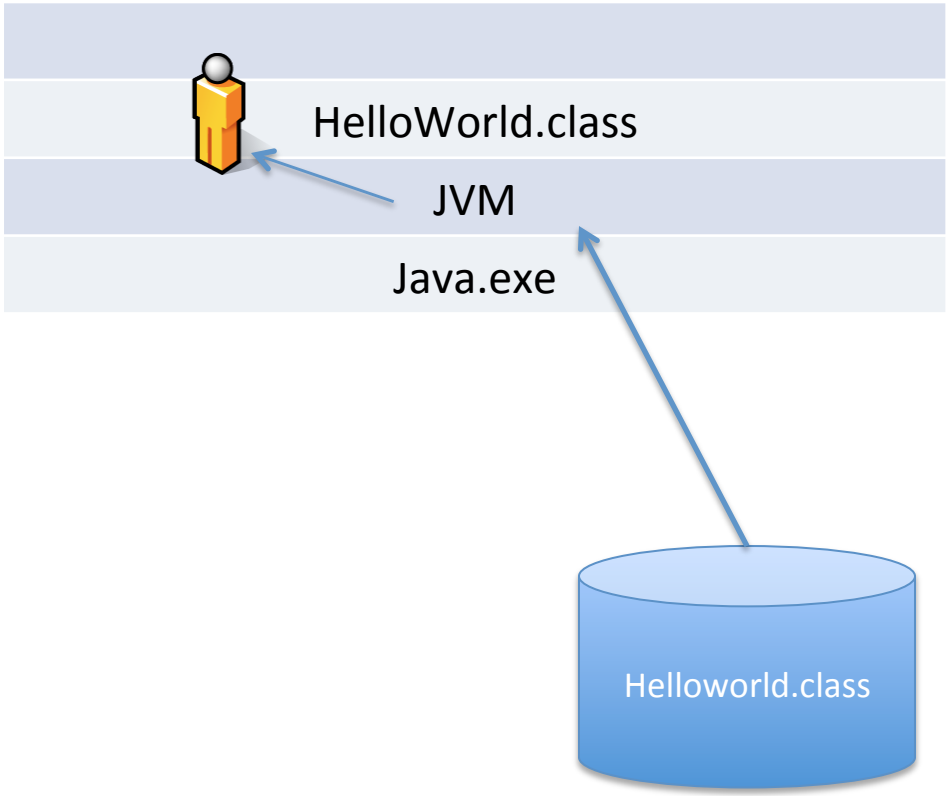
- ¿En cuantos lugares han visto correr Java Runtime Environments (JREs)?

Runtime Environments

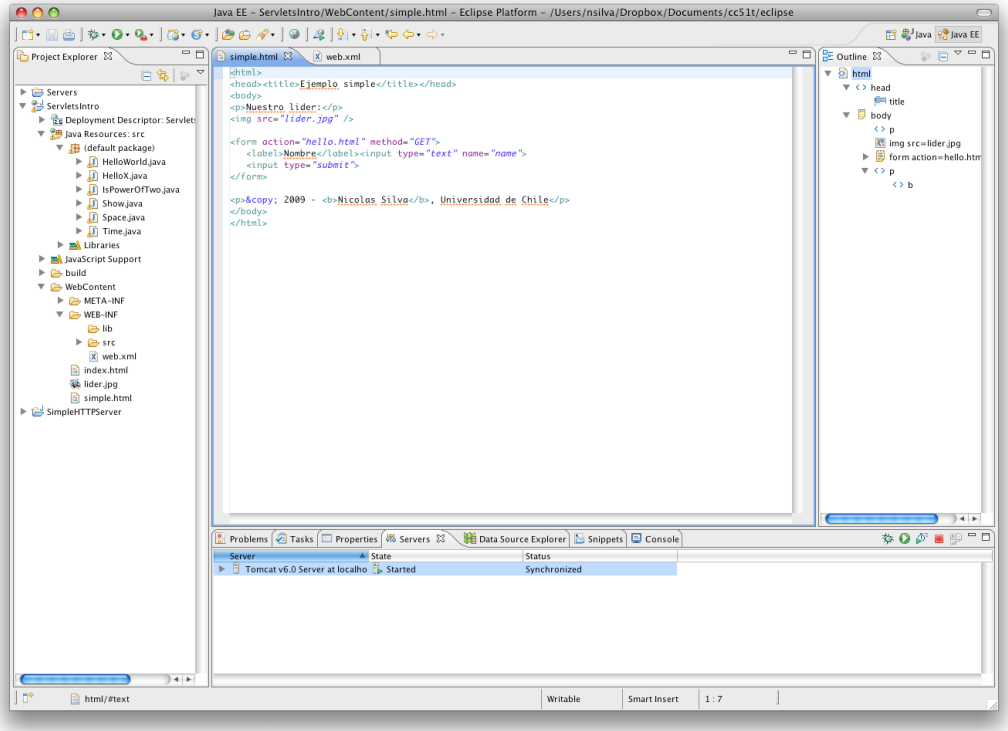
- Desde el terminal (shell)
- En web browsers
  - Usando applets
- En servidores
- En smart cards
- En IDEs de programacion, ej: eclipse

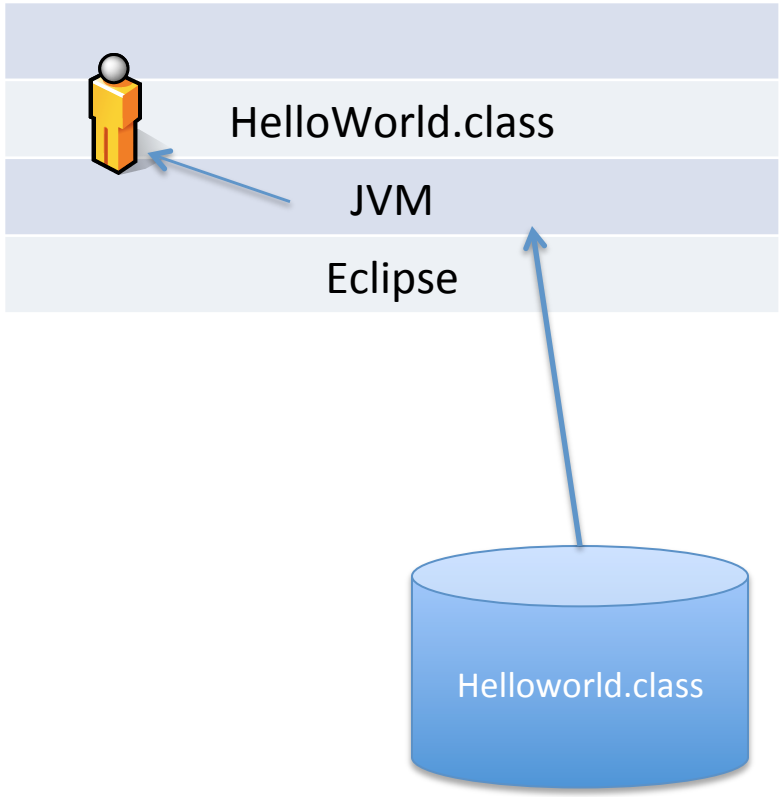
Java en el terminal



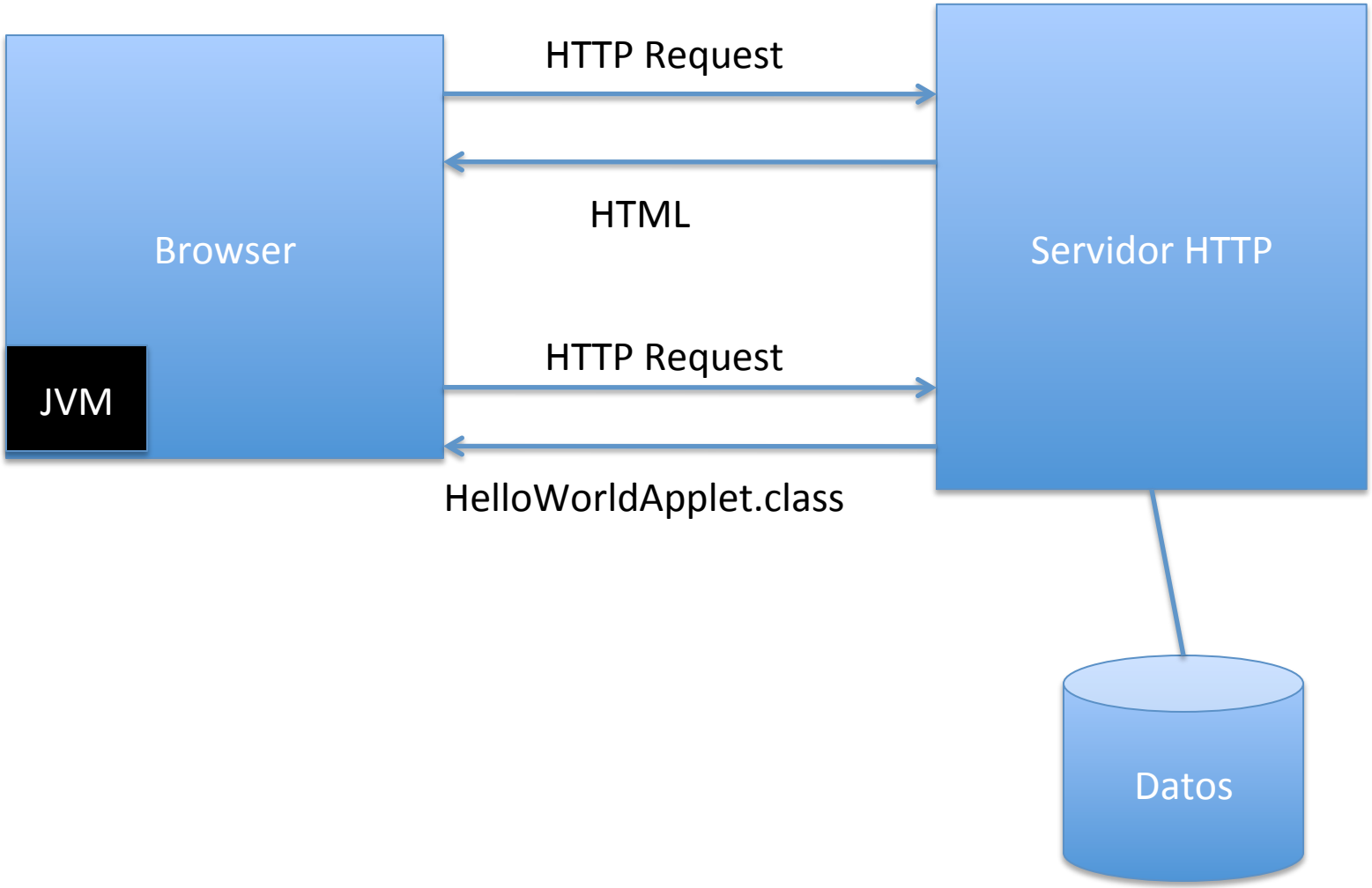


Java en un IDE

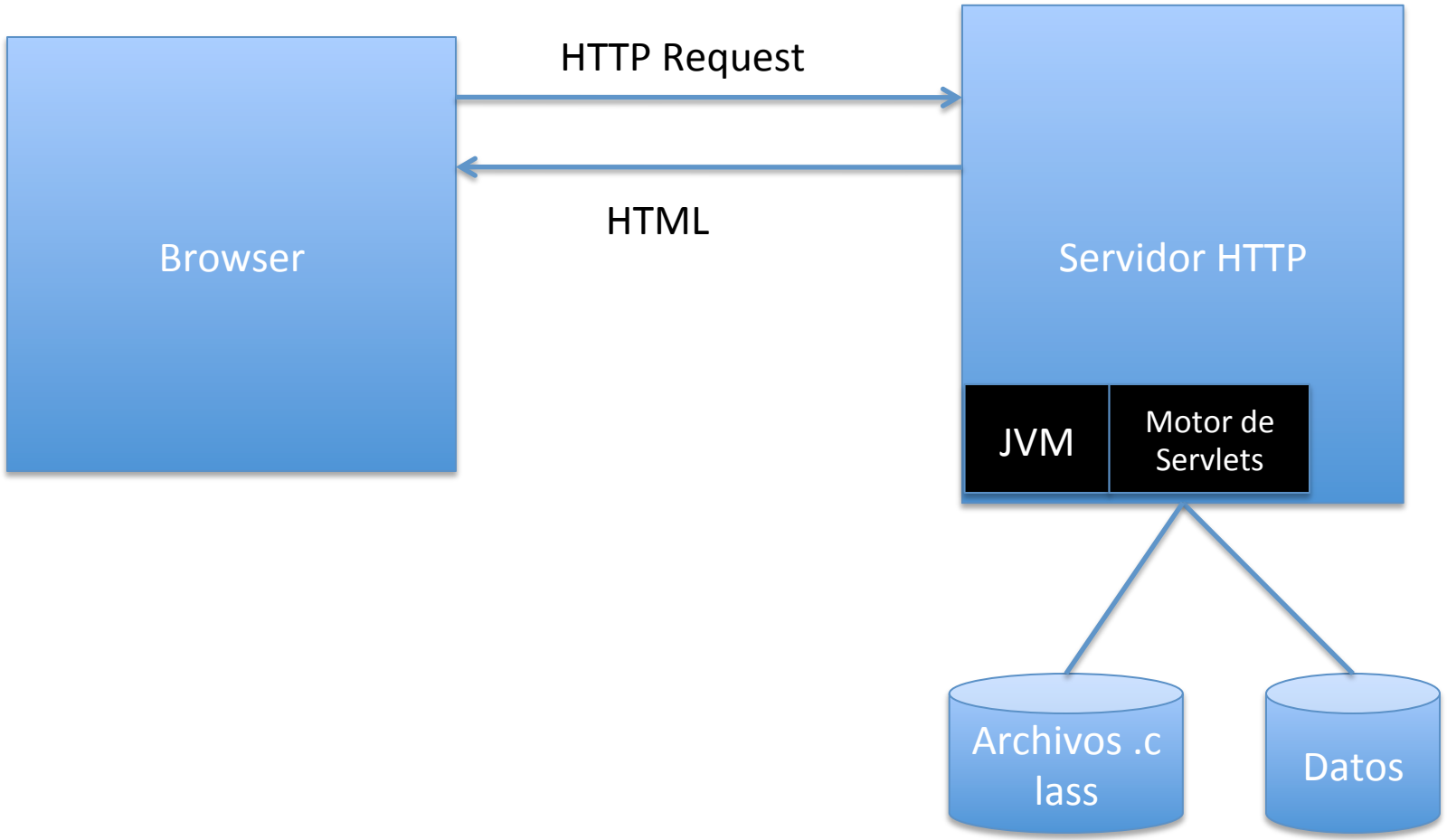




Java en un applet



Java en el Servidor



Ventajas de Java en el Servidor

- No hay problemas de versiones de Java
  - Solo generamos HTML
    - Versionamiento en HTML no es tan grave - ¿por qué?
- No hay problemas de bajar archivos muy grandes
  - Las aplicaciones reales tienen archivos .class muy grandes.
- La manera correcta de acceder a los datos
  - Seguro: Tu programa, corriendo en el servidor
  - Rapido: Tu programa, corriendo en el servidor

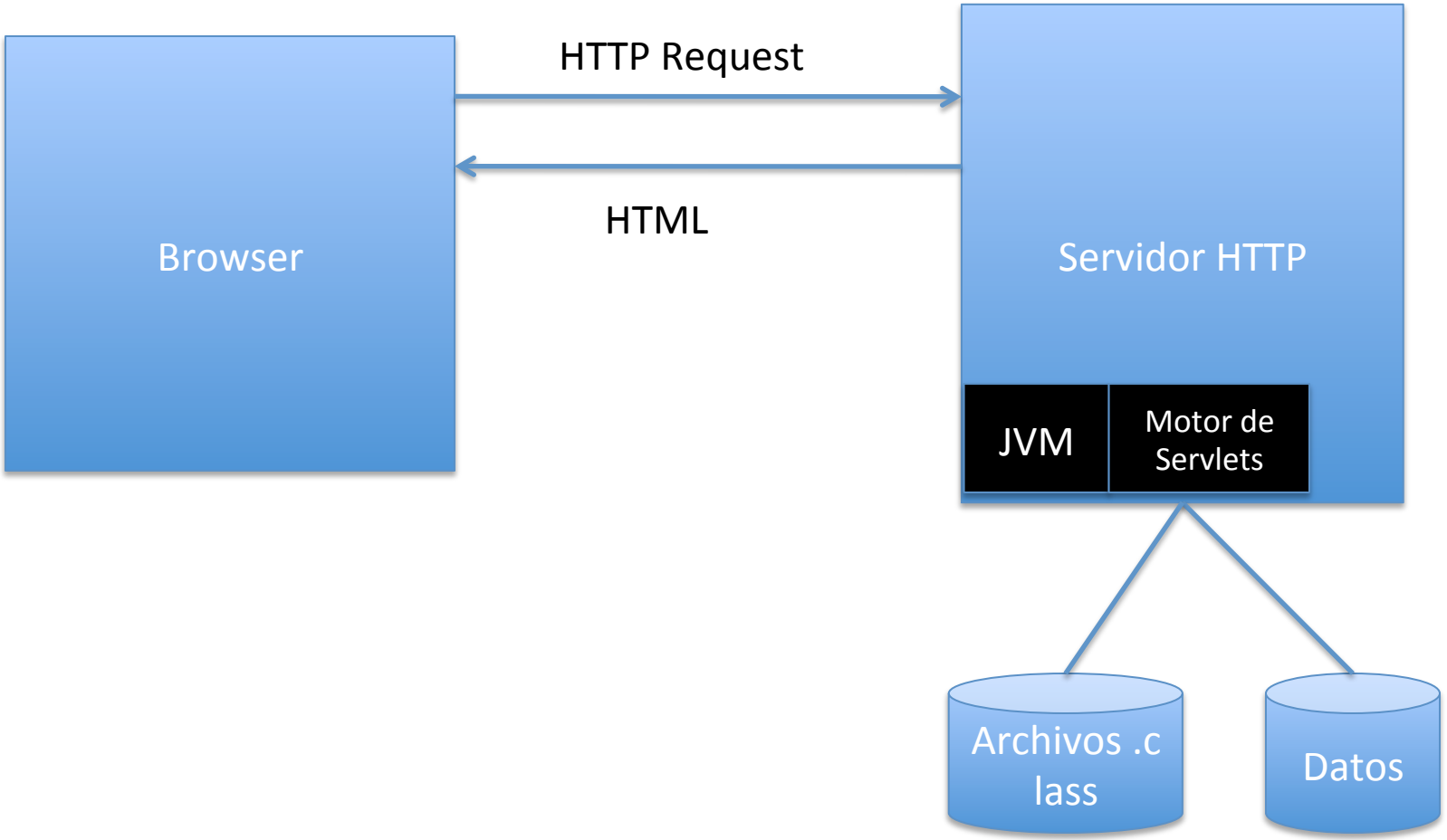
Tomcat

- En este curso usaremos Tomcat como contenedor de Servlets
  - Es una implementacion Open-Source, de Java Servlets y JSP
  - Info, docs, y downloads, en http://tomcat.apache.org
- Recomiendo que usen Tomcat 6.0
  - Funciona con Java 5 y Java 6
  - La version actual es 6.0.32
    - Bajen la version Core, zip
    - Lo usaremos con Netbeans.
- Tomcat lo pueden instalar como un Windows Service si lo desean.

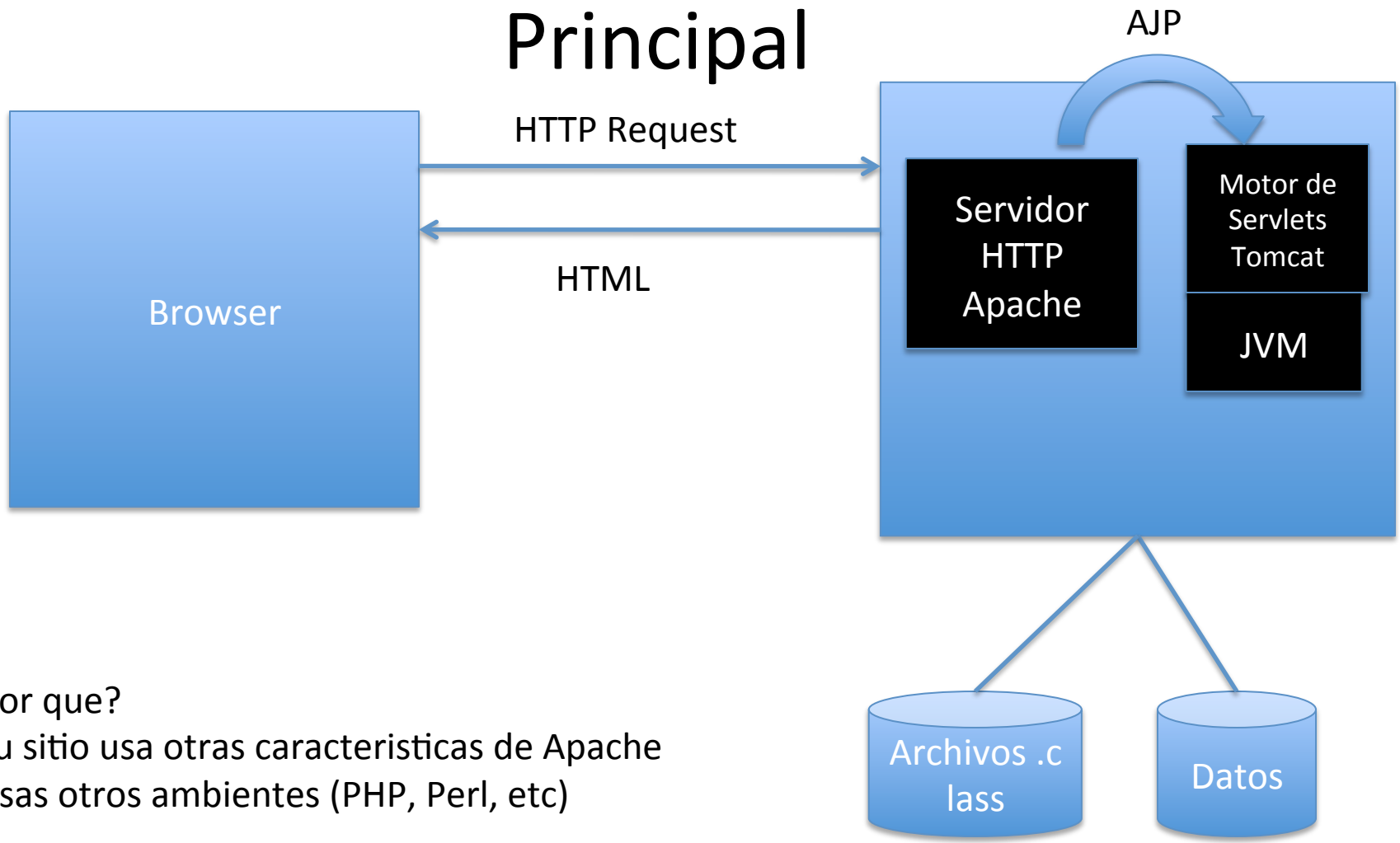
Ejemplos

- Mi laptop corre Tomcat en el puerto 8080
  - http://numeroip:8080
- Esta instalado HtmlDemos
  - http://numeroip:8080/HtmlDemos

Nuestro objetivo es el contenido dinamico



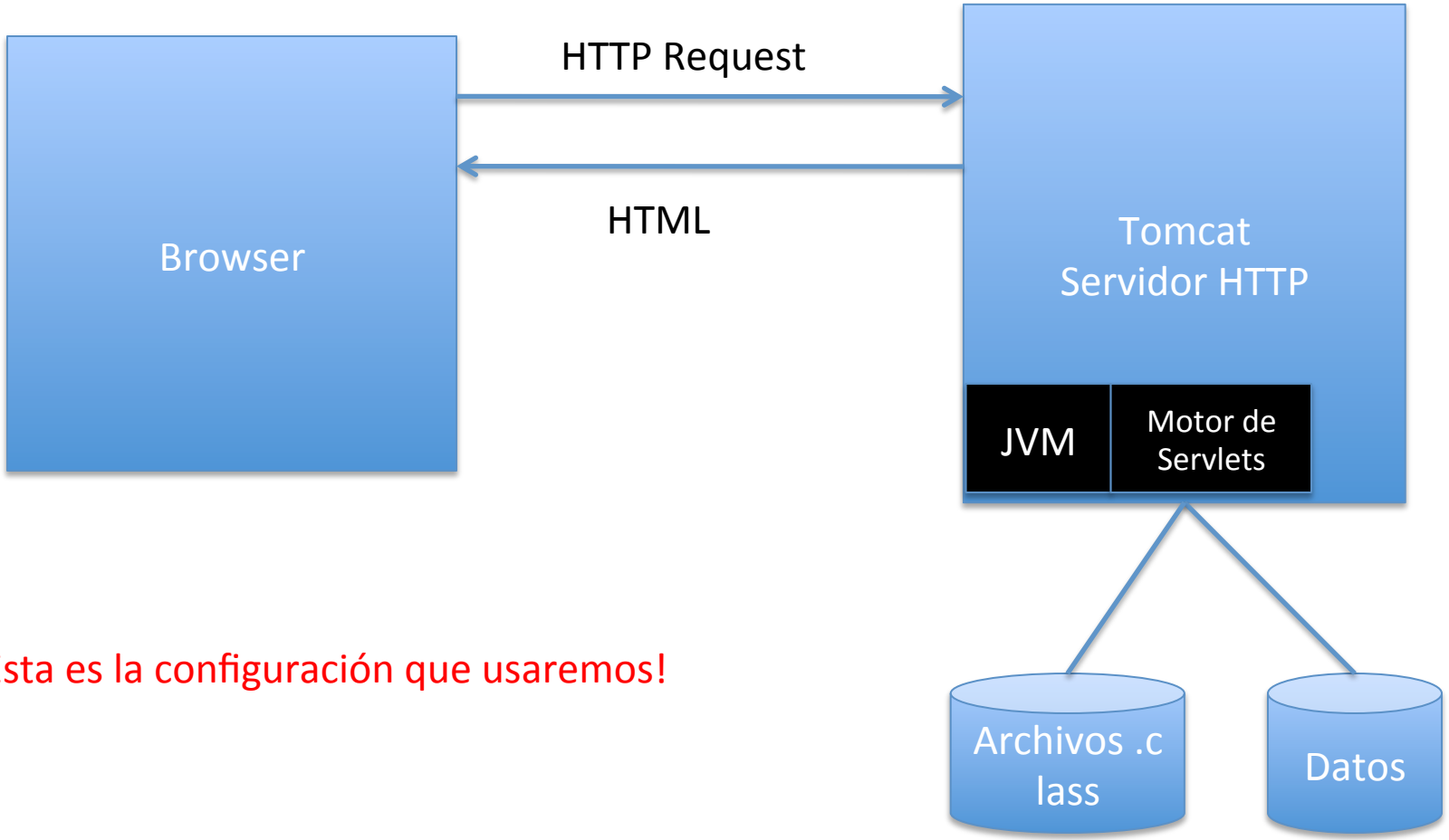
Servidor Apache HTTP como Servidor Principal



¿Por que?

- Tu sitio usa otras caracteristicas de Apache
- Usas otros ambientes (PHP, Perl, etc)

O Usar Apache Tomcat como Servidor HTTP



Esta es la configuración que usaremos!

Estructura de un servidor Tomcat

- Es el mismo loop que cualquier servidor HTTP.
  - `while (true) { request = readHttpRequest(…); response = processHttpRequest (request); sendHttpResponse (…,response); }`
- Un Request es un objeto con toda la informacion de un HTTP Request
- Un Response es un objeto con toda la informacion de un HTTP Response

Procesando un Request

- processHttpRequest() hace un monton de trabajo
  - Determina el Servlet correcto
  - Lo carga e inicializa si es necesario
  - Invoca al Servlet, pasandole los objetos de Request y Response
  - El Servlet procesa los inputs del formulario, mira las cookies, y genera un Response (Tipicamente

Paquetes Java Importantes

- javax.servlet
- javax.servlet.http
- Estos paquetes no estan dentro del JDK o el JRE
- Estan en:
  - %CATALINA_HOME%\common\servlet-api.jar

Interfaz: javax.servlet.Servlet

- Implementado por javax.servlet.GenericServlet
  - init(ServletConfig)
    - Inicializa el Servlet
  - service(ServletRequest, ServletResponse)
    - Procesa un request del cliente
  - destroy()
    - Permite limpiar tus recursos (cerrar archivos, bases de datos, etc)
  - getServletConfig()
    - Retorna un objeto de configuracion, con parametros de inicializacion
  - getServletInfo()
    - Retorna un string con informacion de este Servlet

Clase: javax.servlet.http.HttpServlet

- Este es el objeto que usaran
- Es una subclase de GenericServlet
- Uds deben extender HttpServlet
- HttpServlet tiene metodos “do” para cada metodo HTTP
- Reimplementar los metodos que uno desea usar
  - doDelete(HttpServletRequest, HttpServletResponse)
  - doGet(HttpServletRequest, HttpServletResponse)
  - doHead(HttpServletRequest, HttpServletResponse)
  - doOptions(HttpServletRequest, HttpServletResponse)
  - doPost(HttpServletRequest, HttpServletResponse)
  - doPut(HttpServletRequest, HttpServletResponse)
  - doTrace(HttpServletRequest, HttpServletResponse)

Ejemplo HolaMundo

```
import javax.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HolaMundo extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("    <head>");
        out.println("        <title>Hola Mundo!</title>");
        out.println("    </head>");
        out.println("    <body>");
        out.println("        <h1>Hola Mundo!</h1>");
        out.println("    </body>");
        out.println("</html>");
    }
}
```

Accediendo a Servlets desde el Browser

- Típicamente tendrá el Servlet tendrá un nombre
  - http://localhost:8080/ServletsIntro/HolaMundo

- Se puede definir un Servlet principal que se acceda en forma directa
  - http://locahost:8080/ServletsIntro

- Desde una ubicación remota
  - http://direccionip:8080/ServletsIntro/HolaMundo

Ejemplo Time

```
public class Time extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException{
     response.setContentType("text/html");
     PrintWriter out = response.getWriter();
     out.println("<html>");
     out.println("<head>");
     out.println("<title>Ejemplo Time</title>");
     out.println("</head>");
     out.println("<body>");
     out.println("  <h1>La hora actual en milisegundos es " +
                System.currentTimeMillis() + "</h1>");
     out.println("  <hr/>");
     java.util.Date d = new java.util.Date();
     out.println("  <h1>La hora local es "+d+"</h1>");
     out.println("</body>");
     out.println("</html>");
    }
}
```

Interfaz: javax.servlet.http.HttpServletRequest

- El contenedor de Servlets provee su propia implementacion
- Metodos importantes de : HttpServletRequest
  - getHeaderNames()
  - getHeader(name)
  - getQueryString()
  - getParameterNames()
  - getParameter(name)
  - getCookie(), getSession(),…
  - Descritos en cualquier libro de Servlets o en Java

Interfaz: javax.servlet.http.HttpServletResponse

- El contenedor de Servlets provee su propia implementacion
- Metodos importantes de : HttpServletResponse
  - setContentType()
  - getWriter()
  - getOutputStream()
  - sendError()
  - sendRedirect()
  - addCookie(cookie)
  - Descritos en cualquier libro de Servlets o en Java

Otros ejemplos

- HolaX
- EsPotenciaDeDos
- Show

Estructura de las aplicaciones web

- Cada aplicación tiene su propia carpeta en webapps
  - El nombre de la aplicación y de la carpeta son el mismo
  - Los archivos HTML pueden ir directamente en la raiz de la carpeta
  - La carpeta WEB-INF contiene:
    - Un archivo web.xml con la cofiguracion de la aplicación
    - Un directorio de clases para los binarios de la aplicación
- Ej:
  - HTMLDemos
  - ServletsIntro

Estructura de las aplicaciones web

- Netbeans almacena las aplicaciones internamente de forma diferente
  - Existe una carpeta src para el codigo fuente de los Servlets
  - Existe una carpeta build para los binarios de los Servlets
  - Existe una carpeta dist con un archivo .war listo para hacer deploy en cualquier server.
  - Los archivos HTML van en la carpeta web
  - La carpeta WEB-INF esta dentro de web:
    - Un archivo web.xml con la cofiguracion de la aplicación

Miren el archivo web.xml

- Miren los mapeos que se hacen

Administracion de Tomcat

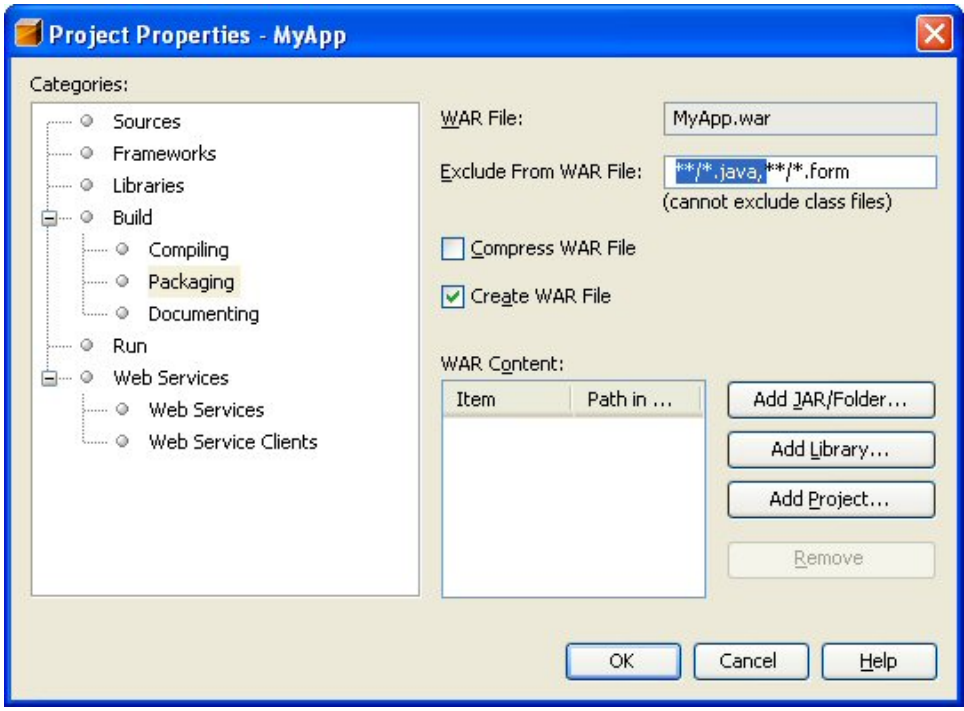
- Interfaz de administracion
  - Al instalar el servidor, corre por defecto como homepage en el puerto 8080
- Los archivos xml contienen la configuracion
  - En el directorio conf y WEB-INF
- %CATALINA_HOME%/logs
  - Stdout*.log contiene la salida de System.out
  - Manager*.log contiene la salida de la aplicación de administracion
  - Catalina*.log contiene info de las aplicaciones cargadas

Archivos .war

- Las aplicaciones web, normalmente estan empacadas en archivos
  - El mismo formato que los archivos .jar y .zip
- Ej:
  - Los ejemplos de hoy estaran como ServletsIntro.war
  - Se pueden crear con winzip o con el mismo netbeans
- Para instalar una aplicación en el servidor, basta copiar el .war al directorio webapps.

Archivos .war (Importante)

- Al entregar sus tareas, deberán enviar el archivo .war
- El .war que genera netbeans (En la carpeta dist) no incluye el codigo fuente.
- Para que lo incluya:
  - Click derecho al proyecto
  - ->Properties
  - ->Build->Packaging
  - En exclude from WAR file remover la sentencia: **/*.java



Archivos .war (Importante)

- Para que Netbeans actualice el .war que genera en la carpeta ./dist
  - Menu Run -> Clean and Build Project

Como construir sus servlets

- Descomprimir Apache en alguna carpeta a su eleccion.
- Setear el usuario manager
  - apache-tomcat/conf/tomcat-users.xml
  - Agregar esta linea:
  - `<user username=”manager" password=”manager" roles=”manager"/>`
- Configurar Servidor Tomcat en Netbeans
  - Pestaña Services
  - Click derecho a Servers
  - Add Server
  - Tomcat 6.0
  - Seleccionan el directorio donde alojaron Tomcat

Como construir sus servlets

- Abrir Netbeans
- File->new Project-> Java Web -> Web Application
- Opcion 1:
  - Crear una nueva clase java que extienda de HTTPServlet y editar el codigo de sus servlets
  - Editar el archivo web.xml consecuentemente

  Opcion 2:

  Click derecho al proyecto => new Servlet.
- Para ejecutar, hacer click al icono de “Play”

Servlets: Lo bueno y lo malo

- Malo
  - Java es un ambiente relativamente complejo
    - Editor, javac, debugger, etc
  - Habituarse a la programacion OO
    - Metodos, interfaces, clases, convenciones, etc
  - Sevlets no son particularmente una buena API para generar HTML
- Bueno
  - Lenguaje moderno y poderoso
  - Facilmente extendible por los conceptos de Java de clases, herencia, etc
  - Multiplataforma
  - Muy conocido

Ventajas de Java en el Servidor

- No hay problemas de versiones de Java
  - Versionamiento en HTML es un problema no tan grave
- No hay problemas de bajar archivos muy grandes
  - Las aplicaciones reales tienen archivos .class muy grandes.
- La manera correcta de acceder a los datos
  - Seguro: Tu programa, corriendo en el servidor
  - Rápido: Tu programa, corriendo en el servidor

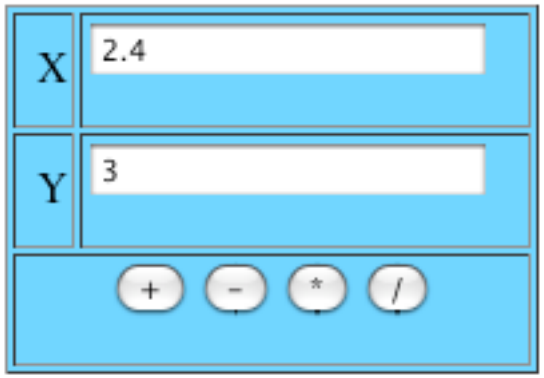
Donde leer mas

- Libro de Servlets
  - Head First Servlets & JSP (O'Reilly)
- Documentacion de Tomcat
  - http://tomcat.apache.org

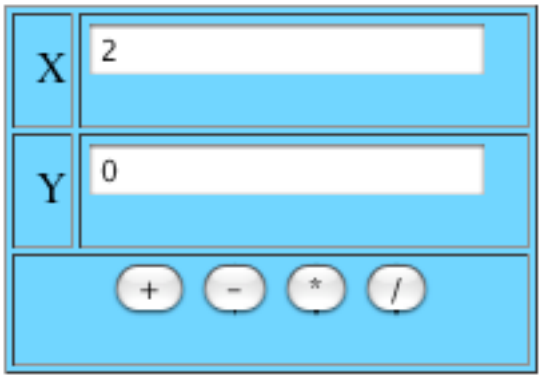
Tarea 2

- Instalar Java y Tomcat en su Computador
  - Pueden correr Tomcat dentro de Netbeans si lo desean
- Crear un servlet que implemente la calculadora de la Tarea 1
  - Cuando se invoque sin parametros, desplegar la calculadora
  - Cuando se invoque con parametros, desplegar la respuesta
  - Desplegar un error si los inputs no son numeros
  - Separar la respuesta del formulario
  - Redesplegar los inputs en el formulario
  - La calculadora debe estar siempre visible

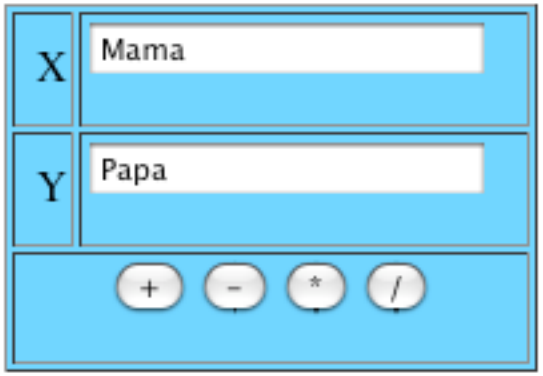
2.40 + 3.00 =5.40



No se puede dividir por 0



X no es un numero
Y no es un numero



Entrega de la Tarea 2

- Entregar por u-cursos
  - Fecha de entrega: Lunes 21 de Marzo
- La tarea debe llamarse tarea2.war

Proxima clase

- JSPs y Threads!
  - Traigan sus laptops la proxima clase para que puedan ser un ejemplo de carga en los websites de demo. (Es opcional, si es que es mucho problema)