INGENIERÍA INDUSTRIAL
UNIVERSIDAD DE CHILE

FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

IN3501 - Tecnologías de Información y Comunicaciones para la Gestión

CAPA DE NEGOCIOS O DE APLICACION

PROFESORES

Evelyn Andaur
Juan D. Velásquez
Gastón L'Huillier
Víctor Rebolledo Lorca

Temario

- Introducción.
- Redes, Internet y Web.
- Cliente Servidor de Múltiples Capas.
- La capa de datos.
- La capa de negocios.
- La capa de presentación.
- Administración de proyectos informáticos.

Capítulo V

CAPA DE NEGOCIOS O DE APLICACIÓN

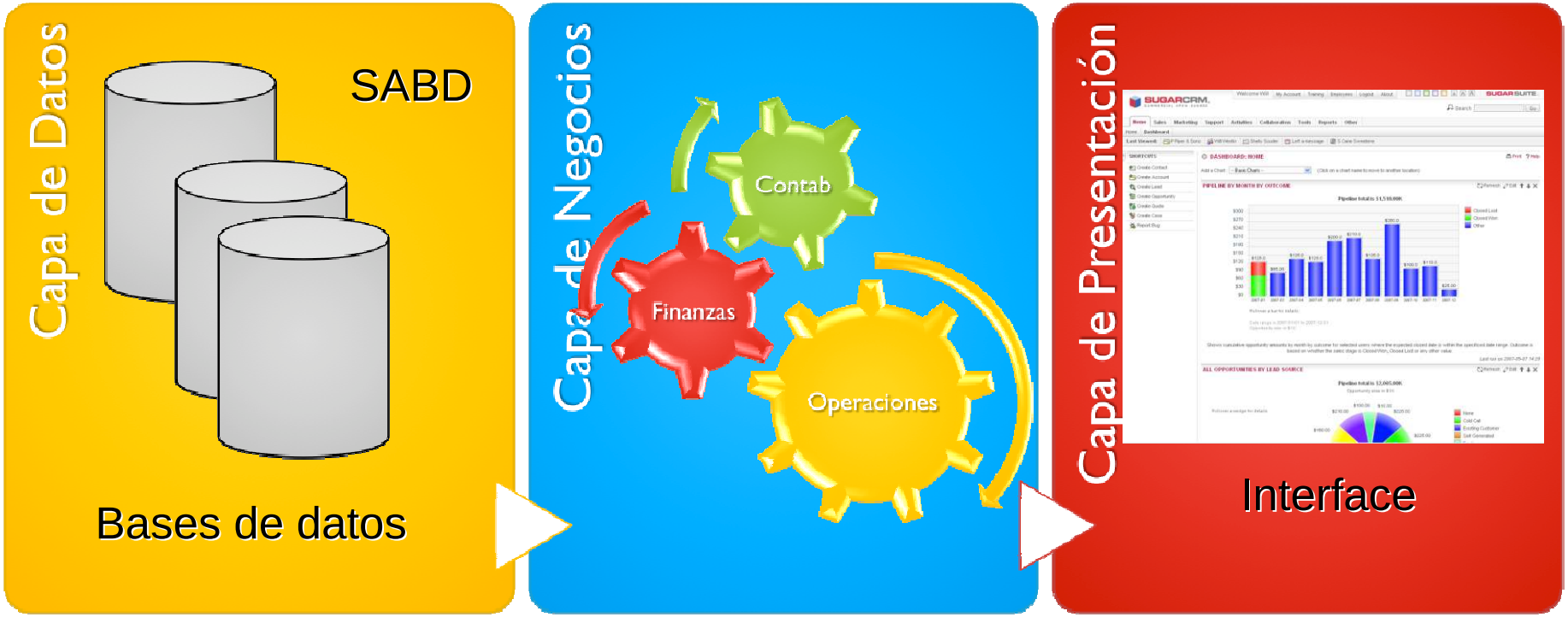
Temario

- Introducción a los procesos de negocio
- Soluciones para modelar la lógica de negocio:
  - Solución a medida
  - Usar un BPM System
  - Usar un Framework MVC

La Capa Media o de Negocio

- Lugar donde *residen los programas* que hacen el **procesamiento de la aplicación**
  - Implementa la **lógica y reglas del negocio** que son soportadas por la aplicación informática.
  - Todos aquellos programas o algoritmos que manipulan el intercambio de información entre la interfaz de usuario y las bases de datos
  - Responde a las solicitudes de los usuarios desde la capa de presentación.
    - Recibe los parámetros para ejecutar los programas.
    - Entrega los resultados a las peticiones
  - Mantiene conexión con la capa de datos
    - Pide los datos necesarios para responder a las solicitudes.
    - Envía las operaciones CRUD que deben ser realizadas en la base de datos

La Capa Media o de Negocio



Procesos de negocio

- Secuencia relacionada de **actividades** *cuyo propósito es entregar* ***un producto o servicio*** *determinado* ***para uno o varios clientes***.
  - *Pueden ser medidos, cualificados y cuantificados.*
  - *Siempre asociados a un cliente final*
    - *Interno*
    - *Externo*
  - *Deben ser alineados a la estrategia de la compañía.*
- *Ejemplos:*
  - *Adquisición de materiales*
  - *Proceso de Post-Venta de productos*
  - *Aprobación de crédito financiero*

Lógica de Negocio

Definición	Ejemplo: *Sistema de producción*
Modela los objetos de negocio	Cliente, proveedores, pedido, materias primas, productos, bodegas, plantas, distribuidores, flota de transporte, etc.
Define la interacción entre estos objetos	- Cliente solicita pedido - Proveedores suministran materias primas. - Plantas producen productos que serán llevados a distribuidores mediante flota de transporte
Modela las reglas del negocio (eventualmente pueden ser automatizadas)	- Los clientes que facturan más de 10.000 al año son premium y por tanto se les aplica un 10% de descuento a sus pedidos
Comprende workflows o flujos de trabajo: secuencia de acciones, actividades o tareas enmarcadas en el trabajo colaborativo de un conjunto de personas	-El cliente llena formulario web para hacer pedido. -Su ejecutivo comercial corrobora información, fija urgencia y plazo con el cliente y envía a Producción. También solicita facturación a Cobranza -El agente de producción toma el pedido, ejecuta orden de producción y fija distribuidor y medio de transporte. -Jefe de logística recibe notificación de producción terminada y despacha a bodega. -Jefe de bodega despacha a cliente y envía factura.

Diseño de la Capa Media

- Proceso complejo
- ¿Quién se encargará de modelar el proceso de negocio?
- ¿Quién implementará la solución informática?
- ¿Cuánta gente necesito?
- ¿Cuánto me va a costar?
- ¿Qué soluciones tengo?

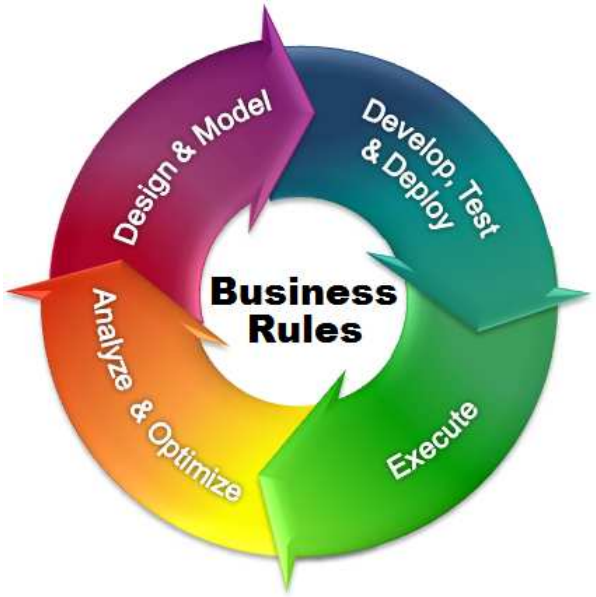
Antecedentes generales

- Los expertos del negocio y los expertos en tecnología no hablan el mismo "idioma".
- ¿Cómo encontrar una solución que satisfaga ambas partes?
- Los expertos del negocio especificarán requerimientos.
- Los expertos en tecnología diseñarán y construirán la solución.
- Los usuarios finales del sistema eventualmente no serán ni los expertos del negocio ni los expertos en tecnología.

Primera Solución:
Solución a medida

- ¿Quién implementa?
  - Consultora externa
  - Área informática interna
- Especificación de requerimientos compleja.
- La solución se programa desde cero
  - O bien usando ciertas librerías para funcionalidades básicas y estándares.
- Es específica al negocio y al proceso particular.
- No siempre incorpora *"Mejores prácticas"*

Segunda Solución:
Usar un BPM System



- Paquete informático
- Sueño del usuario de negocio, pues permite:
  - Modelar procesos.
  - Implementar procesos con "cero" código (lenguaje entendible para usuarios no técnicos)
  - Ejecutar Reglas de Negocio.
  - Orquestar procesos
  - Monitorear indicadores
- Incorpora "Mejores prácticas"

Tercera Solución:
Usar un framework MVC

- Conjunto de librerías que facilitan el desarrollo de aplicaciones web.
- Incorporan el patrón de diseño de software *Model-View-Controller MVC*
- Habilitan el desarrollo *"agil"* de aplicaciones.
- Solución intermedia entre *"Aplicación a medida"* e *"implementación de un BPMS"*
- *Acelera el desarrollo a medida de modo de construir soluciones efectivas y eficientes.*

Solución I

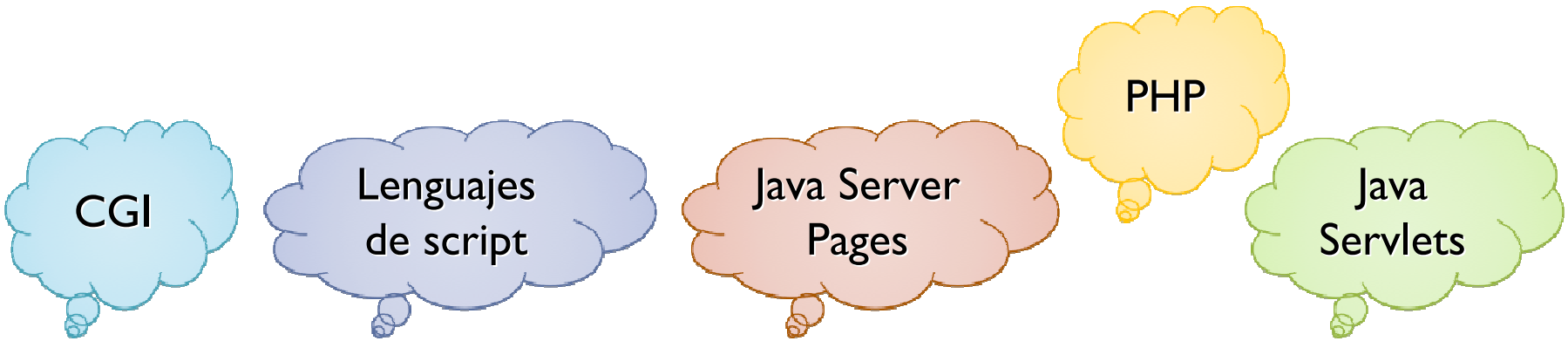
SOLUCIÓN A MEDIDA

Temario

- ¿Cómo resolver lógica del negocio?
- Lenguajes del lado del servidor:
  - CGI
  - Lenguajes interpretados
    - Perl, Python y Ruby
  - ASP.NET
  - PHP
  - Java
    - Servlets
    - Java Server Pages JSP
- Costo / Beneficio

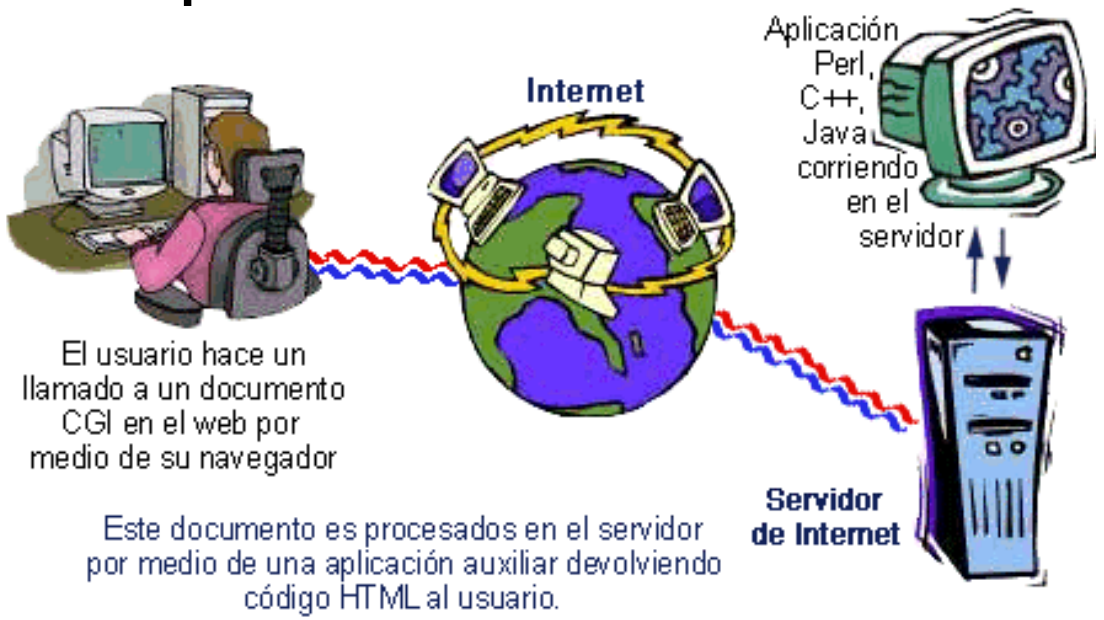
¿Cómo resolver lógica del negocio?

- **RESPUESTA**: ***Programar***
  - *Entender los problemas y necesidades de un usuario y, a partir de dicho conocimiento, construir un programa de computador que lo resuelva o ayude a resolver.*
- *Si el entorno será **Internet o una Intranet**, este "programa" deberá ser construido con una* ***tecnología ad-hoc***

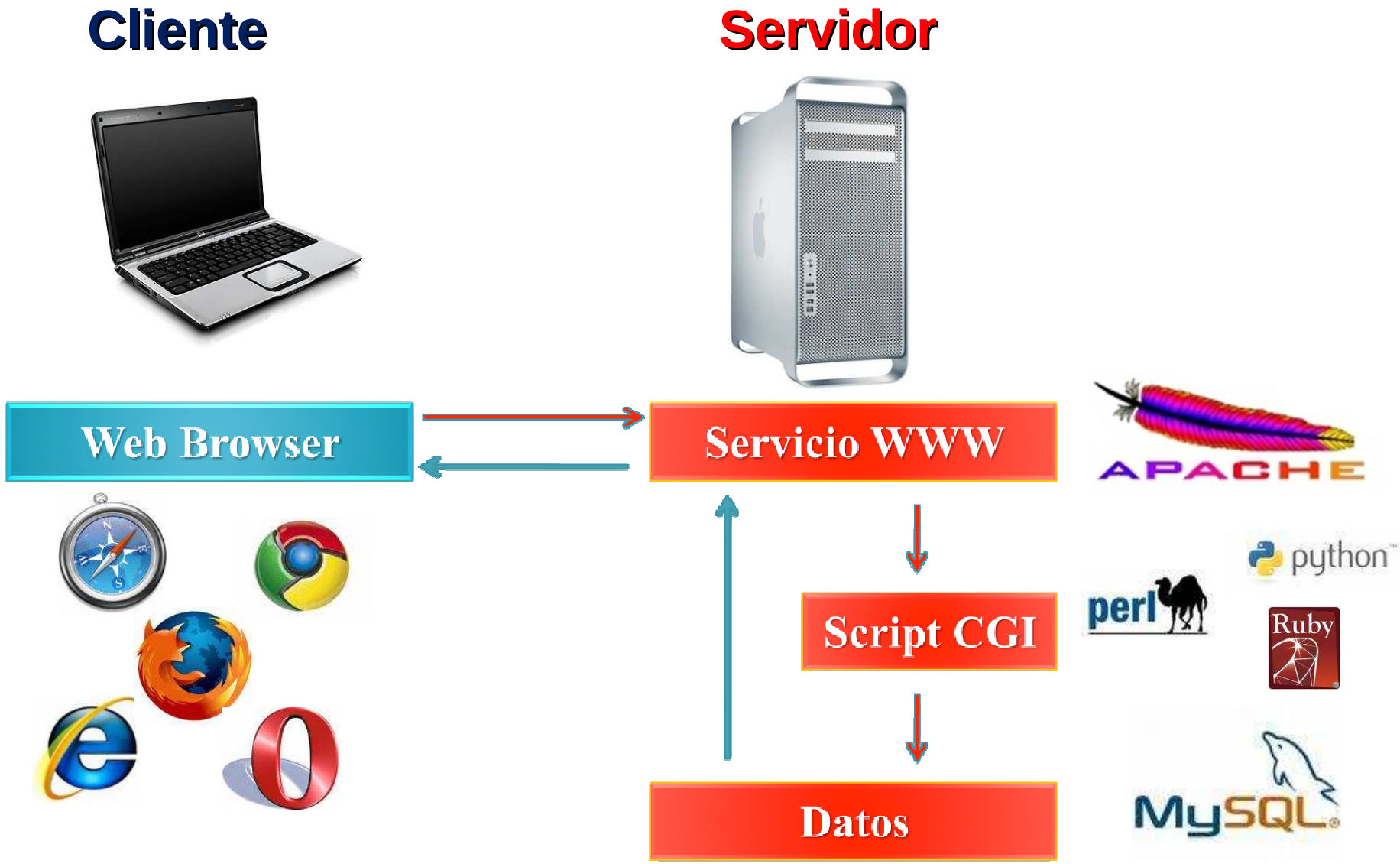


CGI – Common Gateway Interface

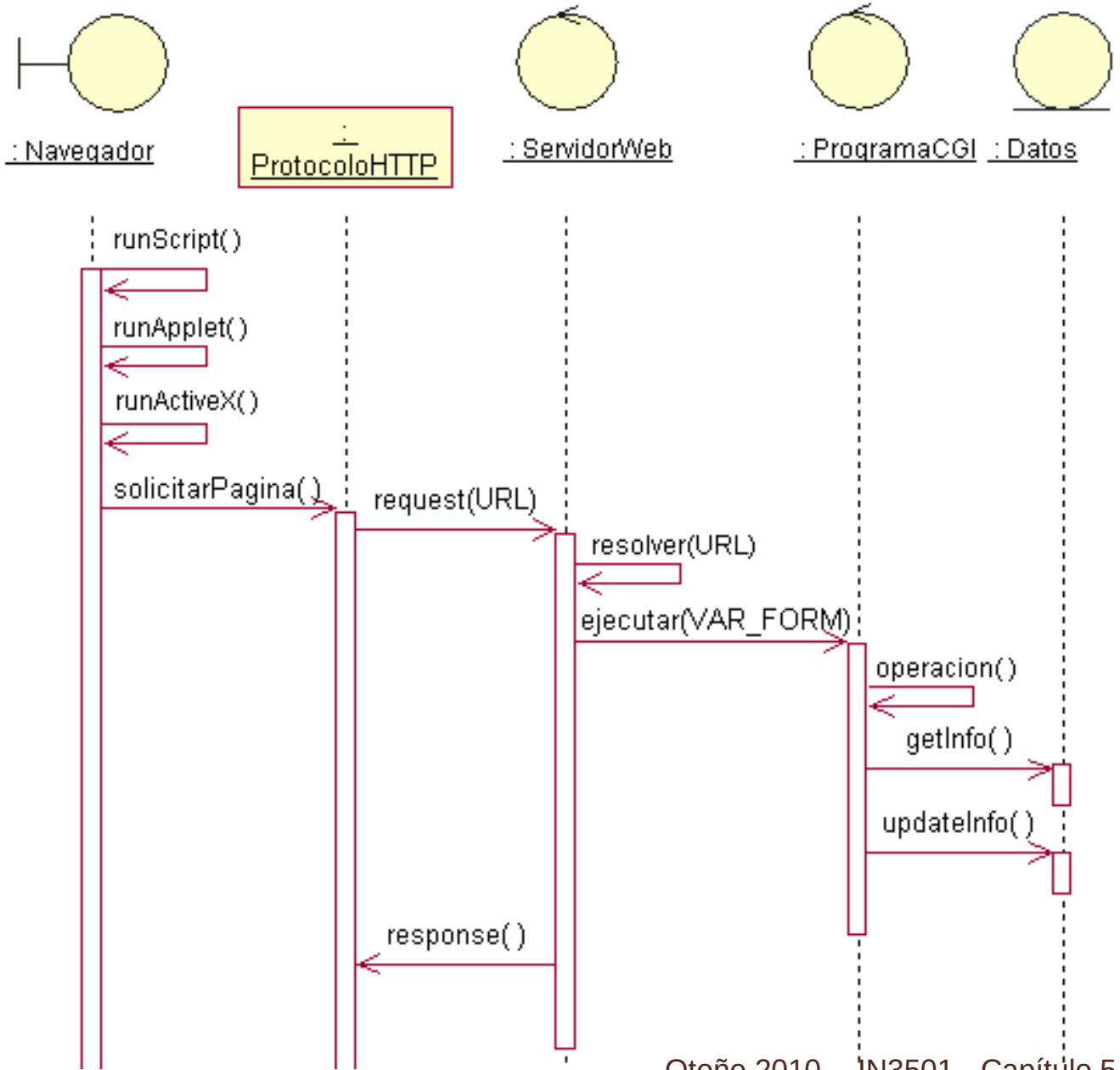
- Interfaz **simple y potente** para la ejecución de procesos
- Especifica:
  - Cómo se pasan los datos desde el servicio WWW a una aplicación externa.
  - Cómo se recuperan los resultados.



El proceso CGI



Llamada CGI


: Navegador
: ProtocoloHTTP
: ServidorWeb
: ProgramaCGI
: Datos
runScript( )
runApplet( )
runActiveX( )
solicitarPagina( )
request(URL)
resolver(URL)
ejecutar(VAR_FORM)
operacion( )
getInfo( )
updateInfo( )
response( )

CGI – Envío de datos

- **FORMULARIOS**
  - Dos métodos de envío
  - Los formularios electrónicos sirven para ***recibir datos desde el usuario y almacenarlos en algún dispositivo.***
  - Un formulario tiene la siguiente estructura:

```
<form action="url" method={Post|Get} >
<input type=... Name="var1">
.....
<input type=submit value=Enviar>
<input type=reset value=Borrar>
</form>
```

CGI – Método GET

1. El cliente WWW solicita un servicio de una aplicación CGI.
2. El servidor HTTP recibe la solicitud y los datos de entrada.
3. El servidor crea un ambiente y variables con los datos de entrada.
4. El servidor ejecuta la aplicación CGI en este ambiente.
5. El CGI procesa las variables de ambiente y recupera los datos de entrada.
6. La aplicación CGI se ejecuta produciendo un resultado sobre su salida estándar.
7. El servidor HTTP redirecciona la salida estándar de la aplicación CGI hacia el cliente WWW.
8. El cliente WWW recibe el resultado de su consulta.

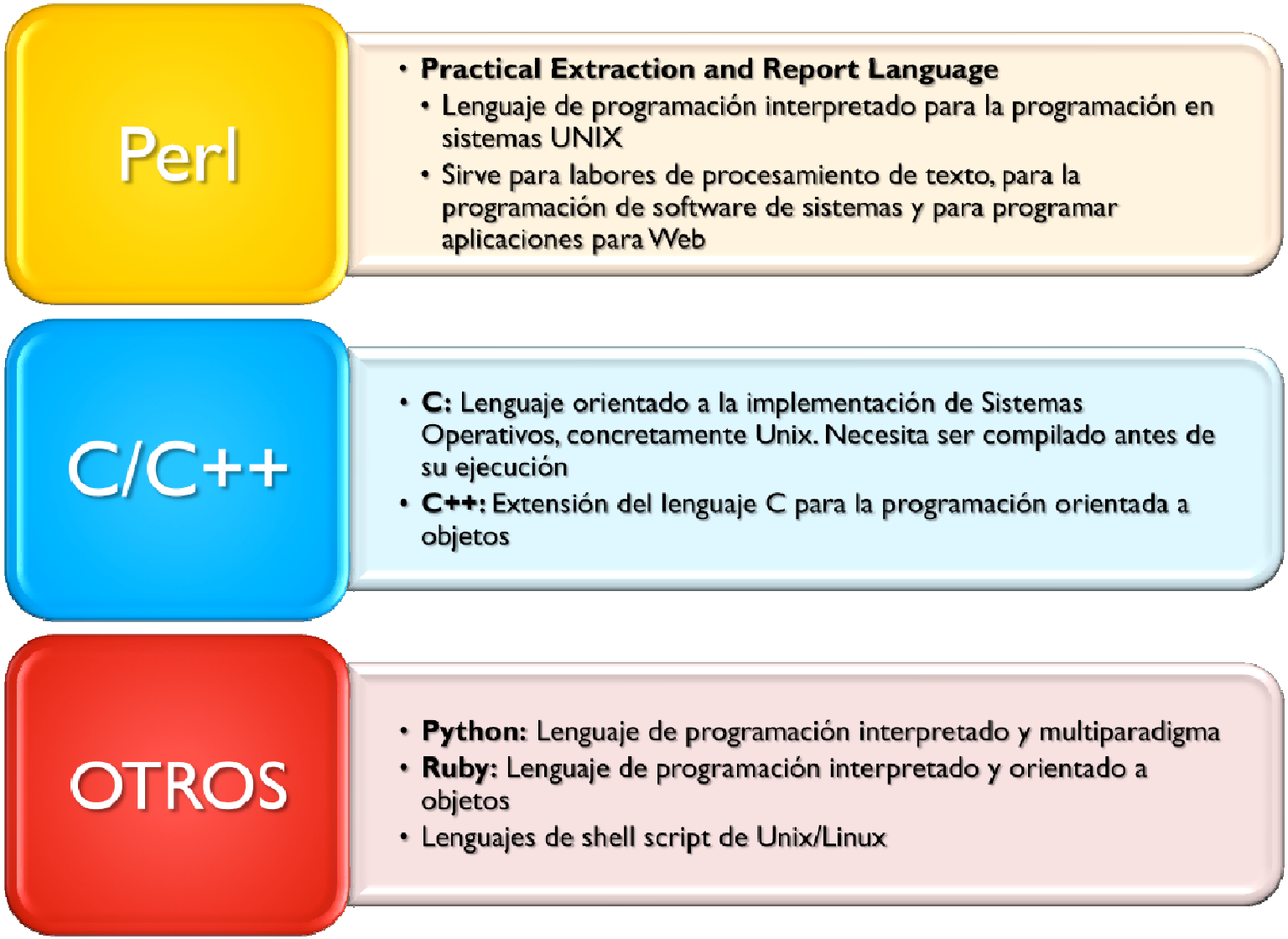
CGI – Método POST

1. El cliente WWW solicita un servicio de una aplicación CGI.
2. El servidor HTTP recibe la solicitud y los datos de entrada.
3. El servidor ejecuta la aplicación CGI pasándole la información a través de la entrada estándar.
4. La aplicación CGI procesa su entrada estándar y recupera los datos de entrada.
5. La aplicación CGI se ejecuta produciendo un resultado sobre su salida estándar.
6. EL servidor HTTP redirecciona la salida estándar de la aplicación CGI hacia el cliente WWW.
7. El cliente WWW recibe el resultado de su consulta.

Lenguajes de Programación CGI

- El Web Server llama a un **CGI creador de una página Web**.
- Luego toma el código de la página y lo envía al browser que lo solicitó.
- Desde un punto de vista funcional, la transacción anterior fue un **intercambio de archivos** entre dos procesos.
- El intercambio clásico de archivos entre procesos es a través de la **I/O estándar**.
- Entonces, *los lenguajes de programación de CGI deben soportar I/O*

Lenguajes de Programación CGI



CGI: Ventajas y Desventajas

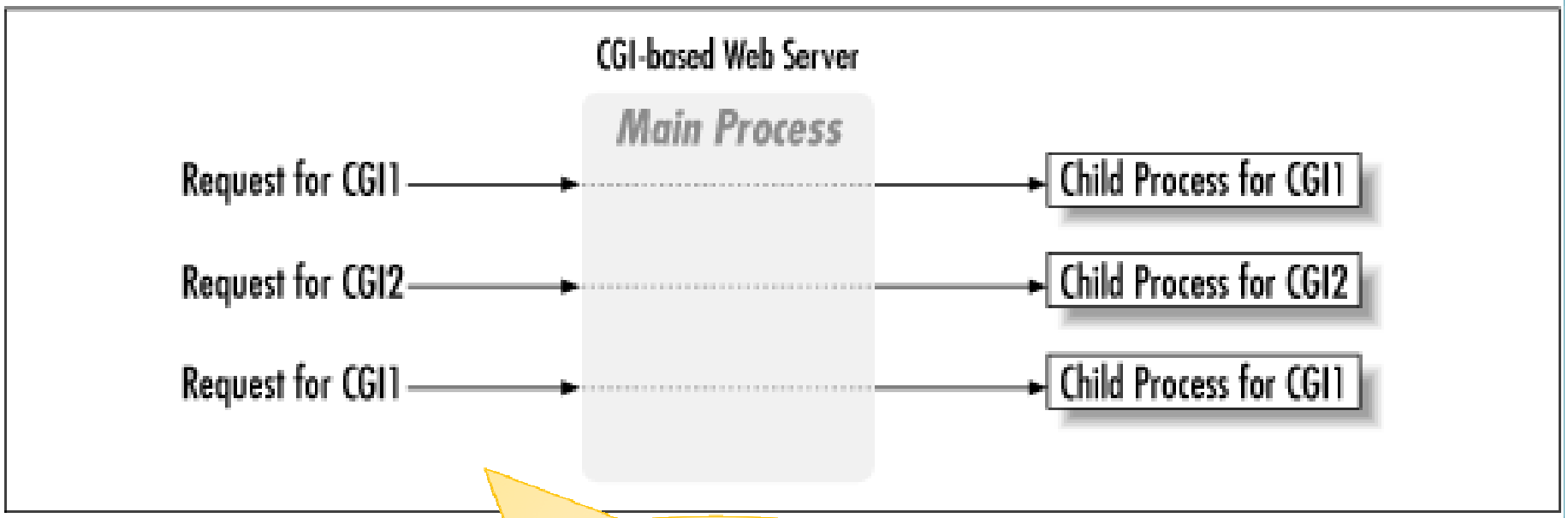
Ventajas

- Ejecuta un **PROCESO PESADO** por cada conexión lo cual lo hace útil para usuarios que requieran de efectuar "grandes" procesamientos vía web.
- Arquitectura primitiva pero simple.
- No tiene un lenguaje definido.
- Requiere de poca configuración.

Desventajas

- **Ineficiente para sitios con muchos usuarios conectados** (Memoria y tiempo de procesador).
- Usualmente los CGI scripts son dependientes de la plataforma

Ciclo de vida de un CGI



1 proceso por cliente

Lenguajes Interpretados o de Script

Lenguajes interpretados

- También llamados **lenguajes de Script**
  - *Serie de instrucciones interpretadas* por el computador **linealmente**
- Nacieron para **administrar sistemas operativos** como Unix
  - Shell script
- Evolucionaron para *controlar el manejo de archivos* y la creación de aplicaciones para la Web
  - Perl
- De acuerdo a necesidades particulares y la evolución natural surgieron nuevas alternativas
  - Python, Ruby, Tcl, Smalltalk, Awk, VBScript, etc.
- Corresponden a **lenguajes de alto nivel**

Lenguajes interpretados - Perl

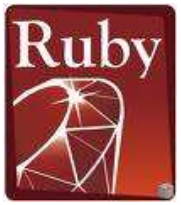
- Lenguaje **open source** de alto nivel, de propósito general y multiplataforma.
- Creado por Larry Wall (lingüista) en 1987
- Nace como **lenguaje de script para Unix**
- Se distingue por la **potente manipulación de archivos de texto**
- Es llamado "The Swiss Army chainsaw of programming languages" por su flexibilidad y adaptabilidad.
- Repositorio CPAN
- **Filosofía**: There's more than one way to do it

Lenguajes interpretados - Python

- Lenguaje **open source** de alto nivel, de propósito general, multiplataforma y multiparadigma.
- Creado por Guido Van Rossum (matemático) en 1991
- Lenguaje **sencillo y fácil de aprender**
  - Pseudocódigo
- Su escritura implica *reglas que facilitan su lectura posterior*
  - Por ejemplo: *Indentation*
- **Filosofía**: Debe haber sólo una manera lógica de hacer las cosas

Lenguajes interpretados - Ruby

- Ruby es un lenguaje de programación **open source** interpretado, reflexivo y fuertemente orientado a objetos.
- Creado por Yukihiro "Matz" Matsumoto en 1995
  - Ruby estaría diseñado para la *productividad y la diversión del desarrollador.*
- ***Killer App:*** *Ruby on Rails*
  - Framework MVC para el desarrollo fácil de aplicaciones web.
    - Muy publicitado
    - Framework referente para el desarrollo de los posteriores
  - *Se suele confundir el lenguaje* ***Ruby*** *con el framework* ***Ruby on Rails***



ASP.NET

ASP.NET

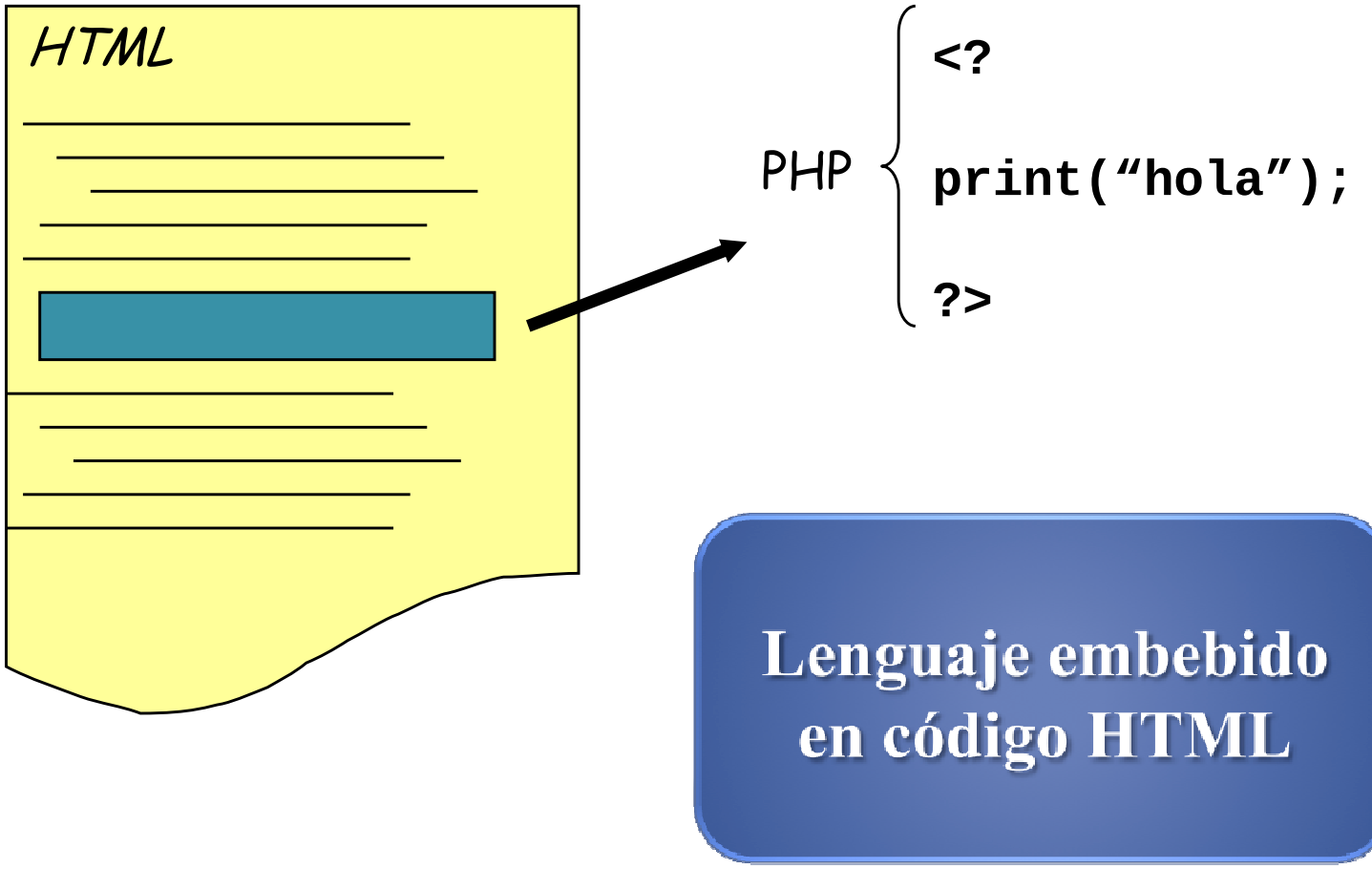
- **Active Server Pages**
- Tecnología ***desarrollada por Microsoft*** para la creación de páginas dinámicas del servidor
- Es interpretado mediante el servidor web por defecto del Sistema Operativo Windows
  - *Internet Information Services (IIS)*
- ASP se escribe en la misma página web, utilizando el lenguaje Visual Basic Script (Javascript de Microsoft).
- Sólo funciona en plataformas Microsoft
  - Costos de licencia
- *Su evolución ha implicado incompatibilidades entre versiones*

Lenguaje PHP

PHP – Hipertext Pre-Processor

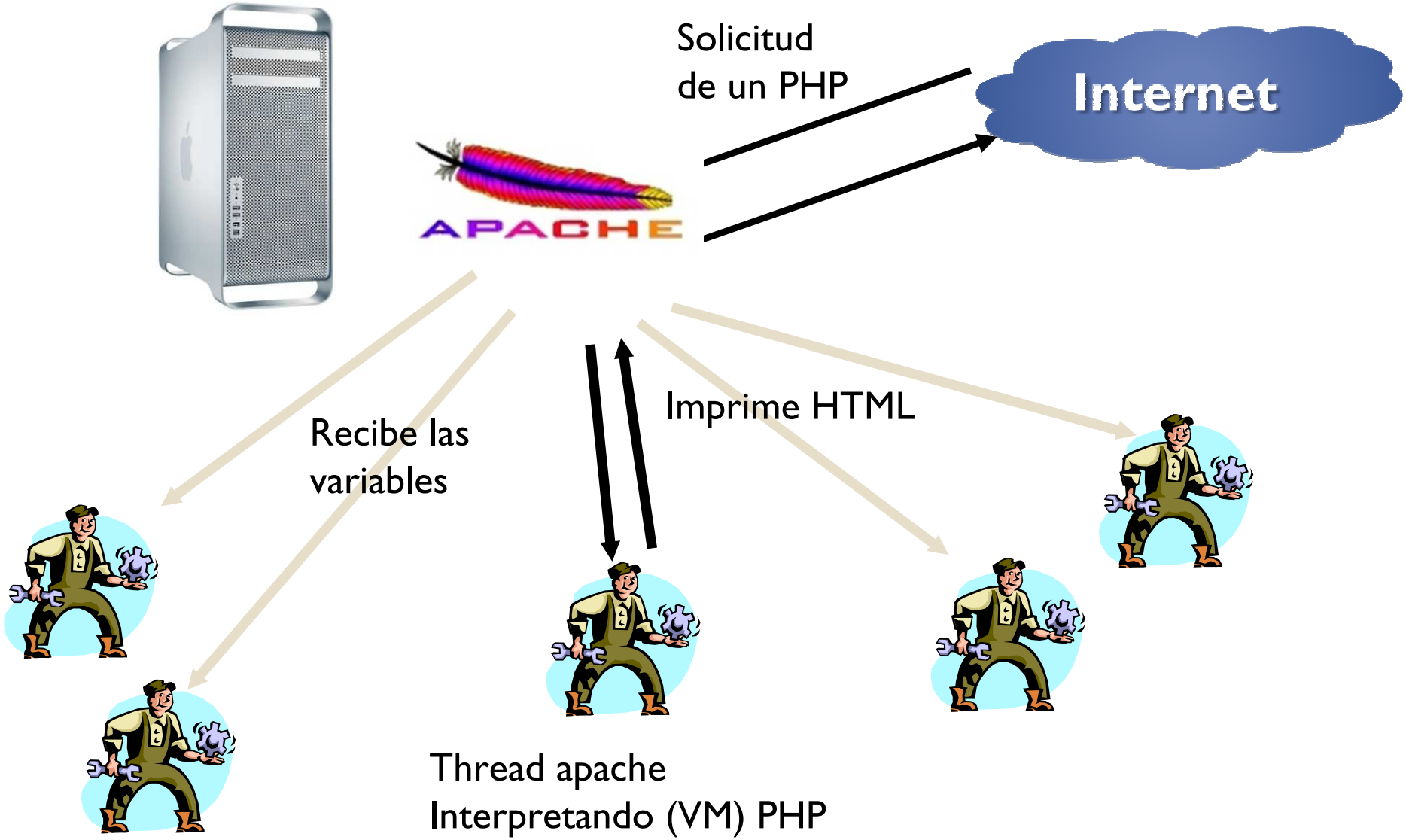
- *Lenguaje de programación interpretado*
- Lenguaje de script
  - Conjunto de instrucciones que se interpretan
- Diseñado para la construcción de páginas dinámicas
- **Se ejecuta en el servidor** y su *salida es interpretada en el Browser del Cliente*
  - Server side scripting
- Open source bajo PHP license
- Lenguaje de **paradigma Imperativo**
  - PHP5 soporta programación **orientada a objetos**

PHP – Hipertext Pre-Processor

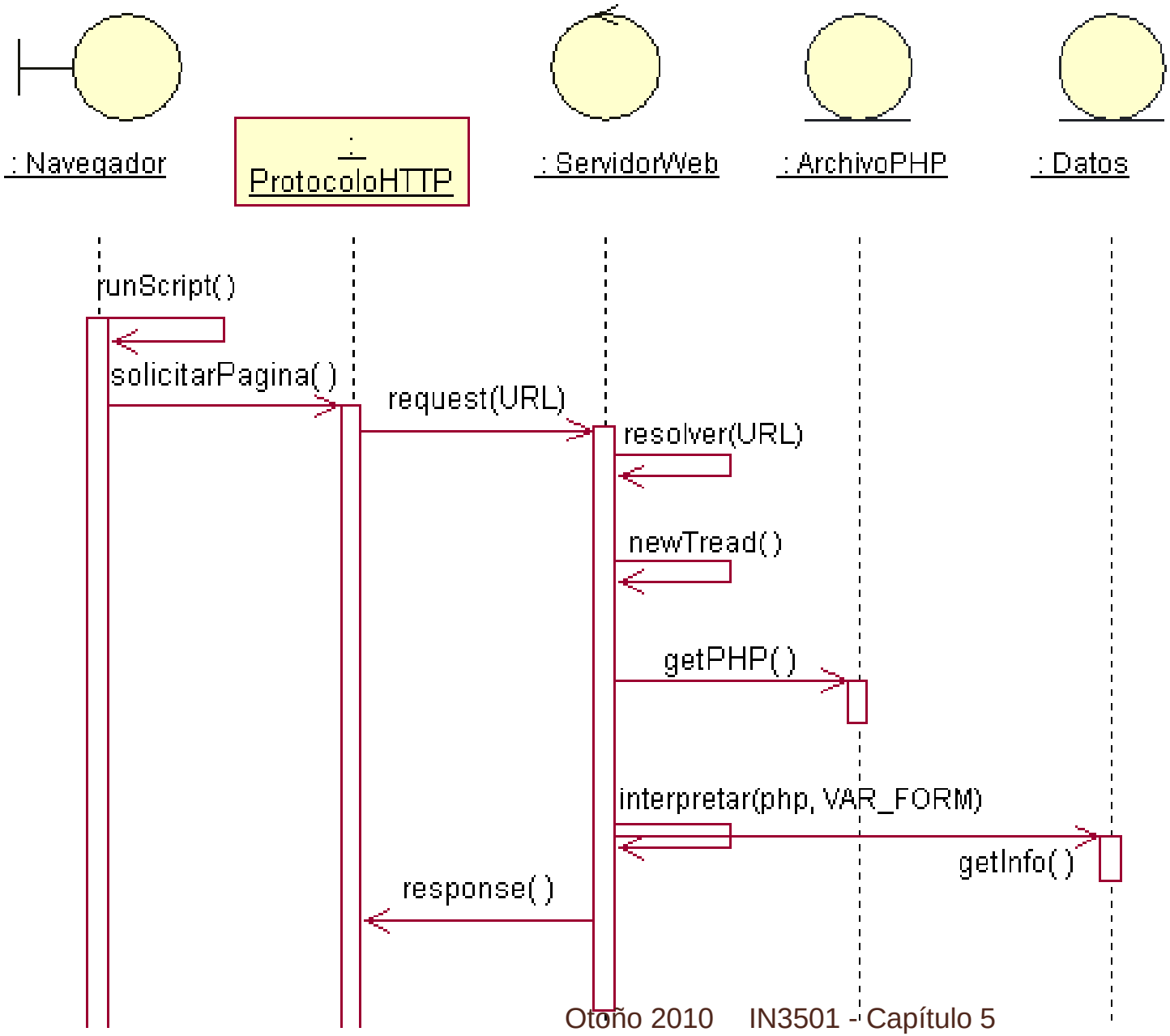


Lenguaje embebido en código HTML

PHP – Hipertext Pre-Processor



PHP – Hipertext Pre-Processor



PHP – Hipertext Pre-Processor

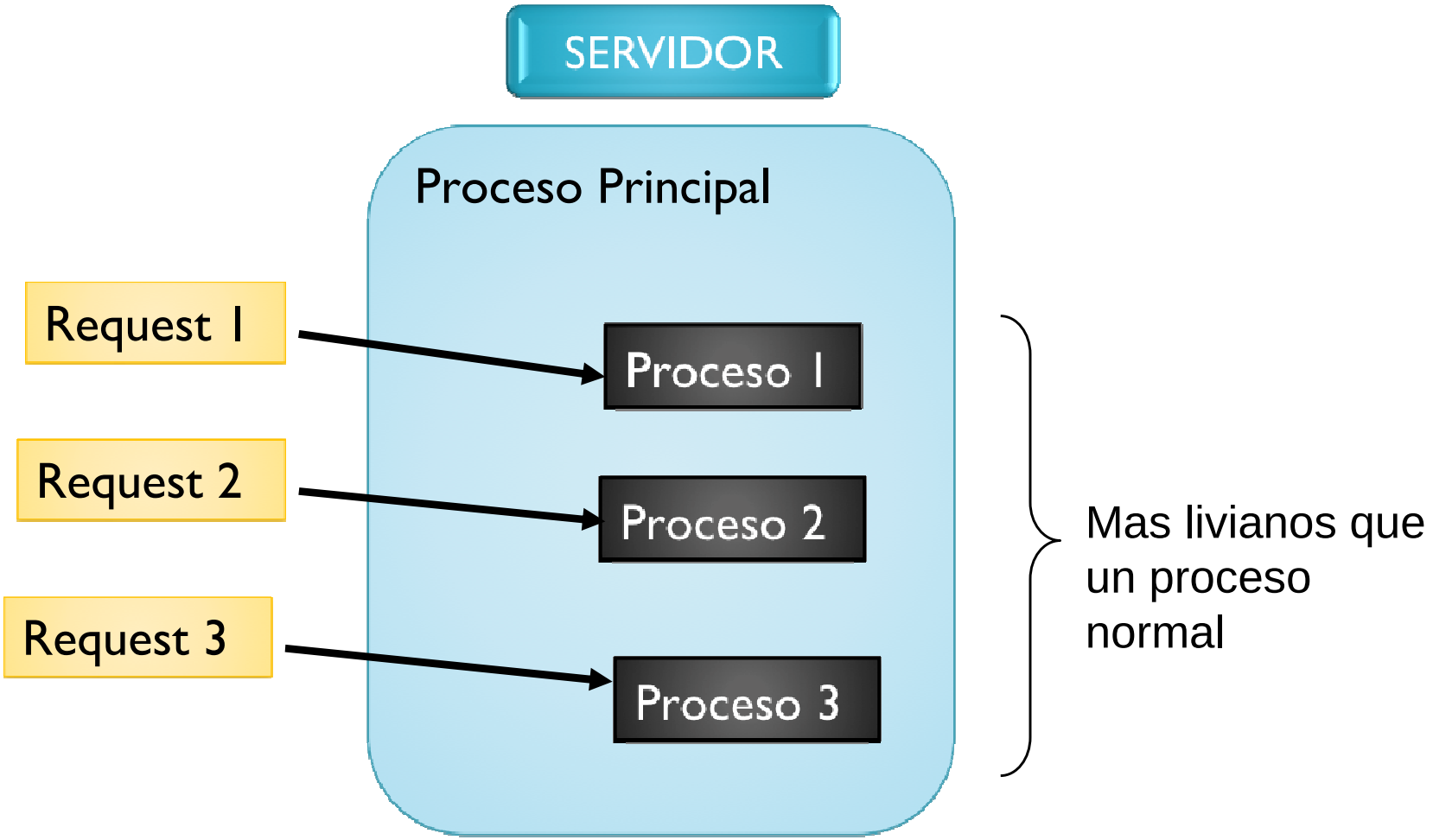
1. Fácil de aprender.
   - En 2 semanas aprendizaje se pueden lograr aplicaciones Web con uso de bases de datos.
2. No tiene tipos de datos. Se puede hacer "hola"+1 sin arrojar errores.
3. Software Libre.
4. Programas son textos.
5. Al encontrarse como módulo de apache se evita el overhead que significa levantar un nuevo proceso en el caso de los CGI. (=mas liviano que un CGI)
6. Apache además provee de un manejo de sesiones. (CGI no tiene un manejo directo de ellas)

PHP – Hipertext Pre-Processor

Desventajas de PHP

- **Grandes cantidades de clientes**, como sitios de e-commerce muy solicitados pueden sobrecargar al servidor. (miles de conexiones por minuto)
- Se puede buscar mejorar la situación por el lado de un balanceador de carga u optimizaciones del software (compilar PHP, cache del interprete, …).
- PHP como lenguaje no restringe al programador a un esquema ordenado y Orientado a Objetos. Aunque si lo soporta.
- No tiene garantía (Zend).

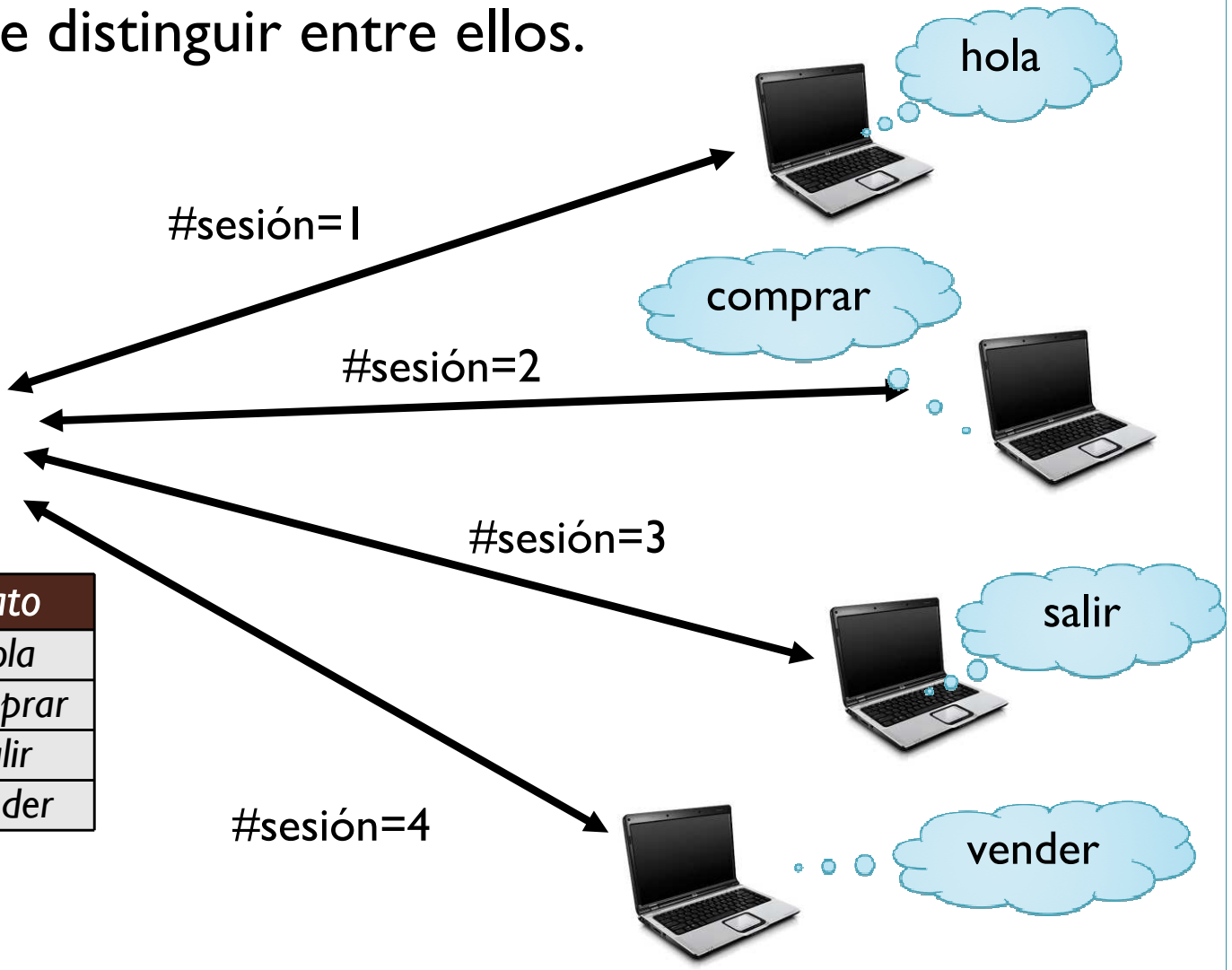
Ciclo de vida PHP

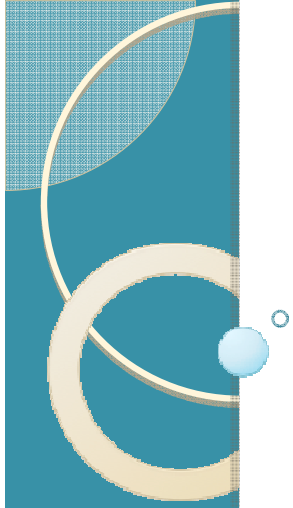


PHP - Sesiones

- Conjunto de datos asociados a cada cliente conectado. De forma de distinguir entre ellos.

Sesion	Dato
1	Hola
2	Comprar
3	Salir
4	Vender





Java Servlets

Java

- Lenguaje de programación de *alto nivel, orientado a objetos, multiplataforma y multiparadigma.*
- Desarrollado por **Sun Microsystems** a principios de los 90
- Corre sobre una máquina virtual Java Virtual Machine
  - JRE → Ejecución de programas Java
  - JVM → Desarrollo y/o Ejecución
- Es **pseudocompilado**, pues no compila a lenguaje máquina como C, sino que a un lenguaje para la **máquina virtual**
  - **Menor performance que C**
- Lenguaje **ampliamente utilizado** en conjunto a C y C++
  - Particularmente en soluciones open source gratuitas y comerciales.
  - Desarrollo de API para acceso a SABD
- **Comprende múltiples librerías** y patrones de diseño de software para una infinidad de propósitos.
  - J2EE
  - **Servlets, Java Server Pages**, etc.

Servlets

- Programas escritos en Java que se *ejecutan en el servidor*
- **Son "equivalentes" a los CGI.**
- Al usarlos con una base de datos, generan una sola conexión y atienden a los clientes a través de **thread**.
- Necesitan de un **contenedor de servlets** para su funcionamiento
  - Apache Tomcat
  - **NOTA:** Apache Tomcat ≠ Apache

- Devuelven páginas HTML
- Se conectan los clientes en la **URL del Servlet**
- Ejecutan distintos métodos según el tipo de requerimiento
  - GET
  - POST
  - etc.

Servlets

Aspectos destacables

- Maneja en forma separada cada tipo de Petición
- Tiene un **manejo muy eficiente** de los threads
- Maneja Sesiones
- Genera Páginas HTML legibles por cualquier Browser
- Permite el desarrollo de aplicaciones utilizando Java, visibles desde cualquier Browser
- Cualquier modificación del código se actualiza automáticamente
- Permite el **desarrollo de aplicaciones realmente escalables**

Ciclo de Vida de un Servlet (Modelo Request/Response)

- Inicialización
  - Se realiza cuando el primer cliente hace una petición del Servlet
  - Se define mediante el método **init**
    - Se pueden especificar parámetros de configuración
    - Se pueden definir variables que son visibles para cualquier proceso de manejo de peticiones

Ciclo de Vida de un Servlet (2)

- Manejo de Petición
  - Para cada cliente que hace una petición del Servlet, se crea un proceso que lo maneja
  - Se ejecutan distintos métodos según el tipo de petición
    - Petición GET (URL's): método **doGet**
    - Petición POST (Formularios): método **doPost**
    - Petición PUT (Upload): método **doPut**

Ciclo de Vida de un Servlet (2)

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest req,
HttpServletResponse res)
                    throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD><TITLE>Hello
World</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<BIG>Hello World</BIG>");
    out.println("</BODY></HTML>");
  }
}
```

Ciclo de Vida de un Servlet (3)

- Destrucción
  - Se ejecuta cuando se baja el servidor de Aplicación
  - Antes el servidor llama el método **destroy**
  - Sirve para realizar operaciones de finalización de la aplicación para evitar corrupción (desconexión de la BD, eliminación de archivos, etc.)

Objeto Request

- Objeto que contiene la información relacionada con la petición que realiza el cliente
  - Parámetros del Formularios HTML.
  - Cookies.
  - Sesiones.
  - Tipo MIME de la petición
  - Información del Browser

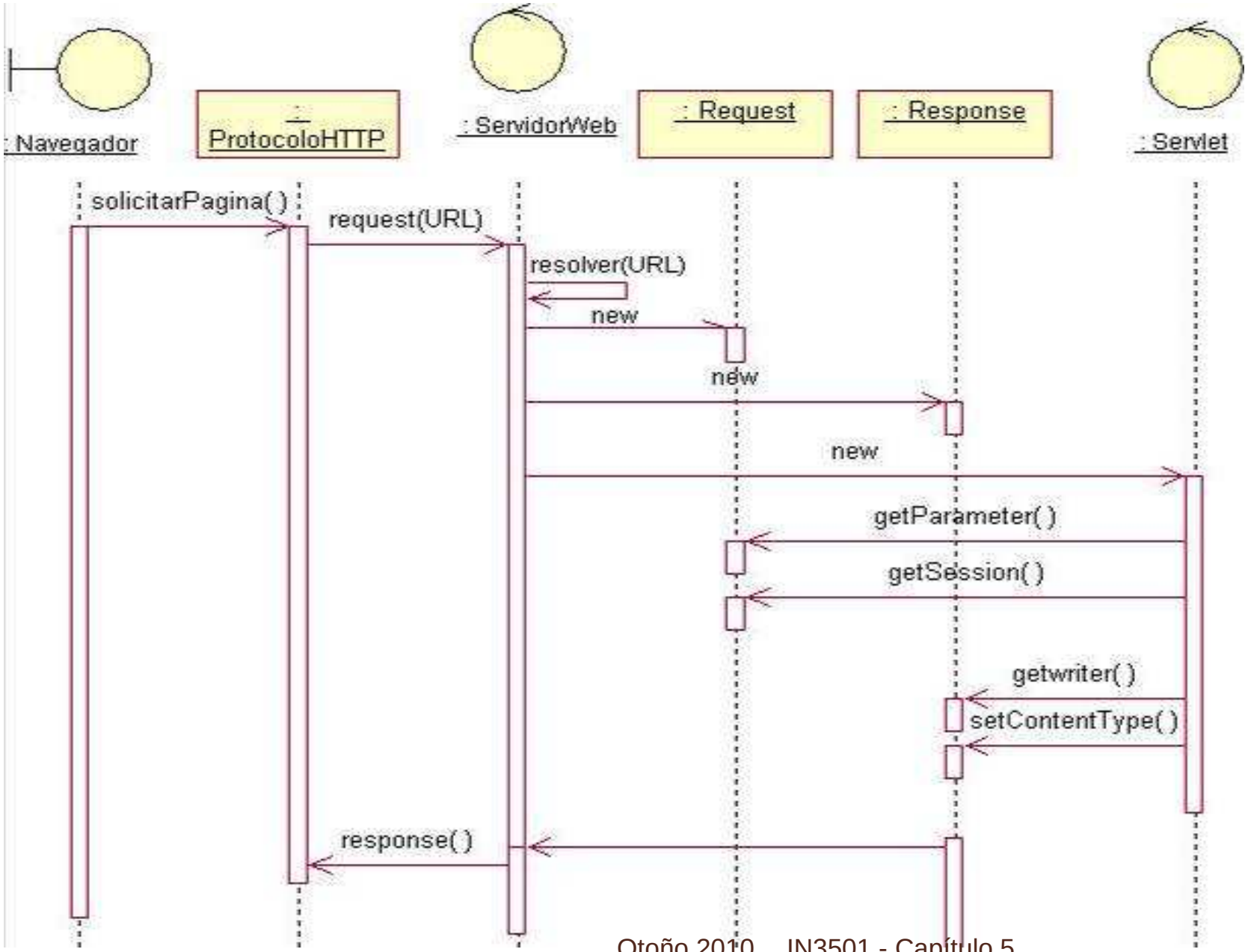
Objeto Response

- Objeto que contiene la información que será enviada al cliente
  - Código HTML a enviar.
  - Cookies a setear.
  - Tipo MIME de la respuesta.

Manejo de Sesiones

- Se realiza mediante el objeto **HttpSesion**
  - Se obtiene del objeto **Request**, con el método **getSession()**
  - Se puede asignar cualquier objeto al objeto **HttpSesion**, que podrá ser visto por cualquier otro método de manejo de peticiones (mientras dure la sesión)

Servlet: El Proceso

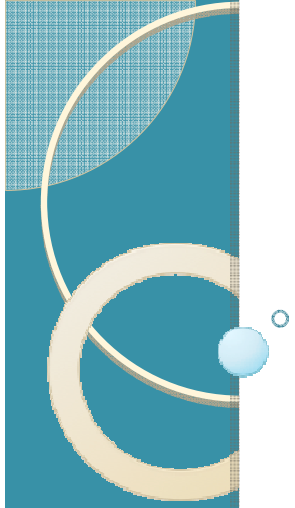


Servlets como Solución Transaccional

- A través de los servlets es posible construir sistemas de alto rendimiento y en entornos transaccionales
- La posibilidad de manejar objetos comunes permite una máxima utilización de recursos, como por ejemplo conexiones a la Base de Datos

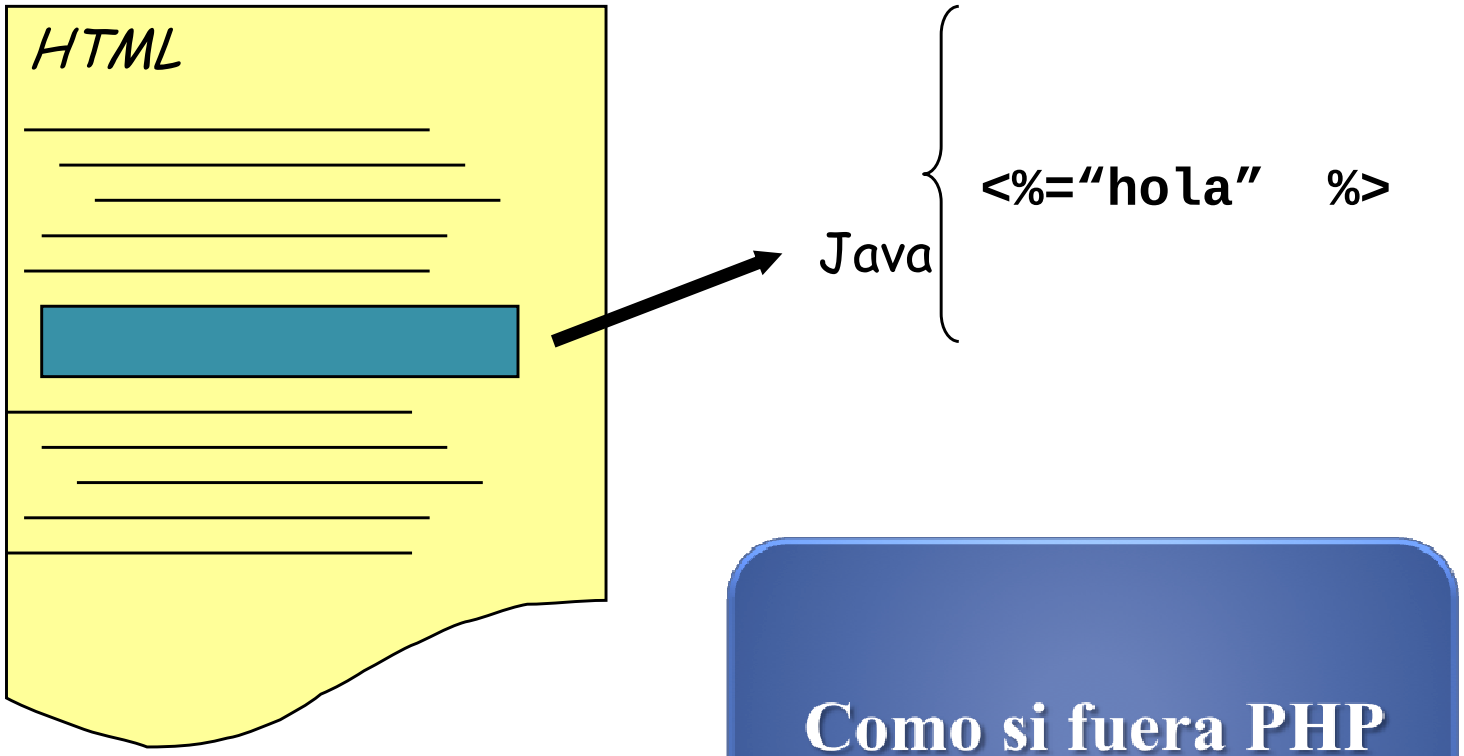
Servlets como Solución Transaccional (2)

- Su esquema de manejo de procesos es muy superior a otras tecnologías desarrolladas para este tipo de aplicaciones
- Existe un gran soporte del mercado, de forma que existen interfaces para casi la totalidad de los sistemas comerciales que están en operación



Java Server Pages (JSP)

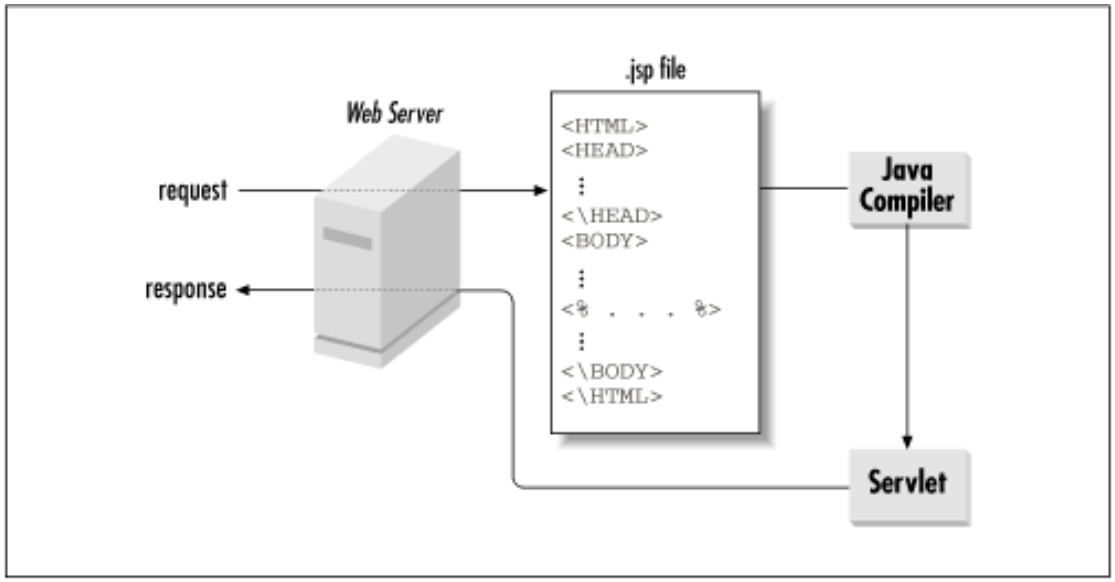
Java Server Pages - JSP



Java Server Pages - JSP

- Respuesta de Java a PHP para generar páginas dinámicas mediante instrucciones embebidas en HTML
- Manera alternativa y simplificada de construir servlets
- Pueden funcionar en conjunto o reemplazando a los Servlets

Java Server Pages - JSP



Java Server Pages - JSP

request: HttpServletRequest object

response: HttpServletResponse object

out: PrintWriter object

in:BufferedReader object

Java Server Pages - JSP

```
<HTML>
  <HEAD>
  <TITLE>Hello</TITLE>
  </HEAD>
<BODY>
<H1>
<%
     if (request.getParameter("name") == null) {
%>
Hello World
<% } else { %>
Hello,
<%= request.getParameter("name") %>
<% } %>
</H1>
</BODY>
</HTML>
```

JSP vs Servlets

- ¿Cuándo usar jsp vs servlet?
  - **Jsp:** Su fuerte es el manejo de interface dinámica en el servidor. Puede ser editado en Dreamweaver
  - **Servlet:** Lógica de negocio.
  - **Jsp:** Mas rápido de implementar, pero menos estructurado.
  - **Servlet:** Mas lento de implementar por la estructuración orientada a objetos a realizar.