Servlets and JavaServer Pages™

Lee Jean Man
Sun Microsystems

Agenda

- Servlets
  - Introduction
  - Basics
  - Advanced topics
- JavaServer Pages™ (JSP)
  - Introduction
  - JSP Syntax
- Current Status and Roadmap
- Conclusions, Q&A

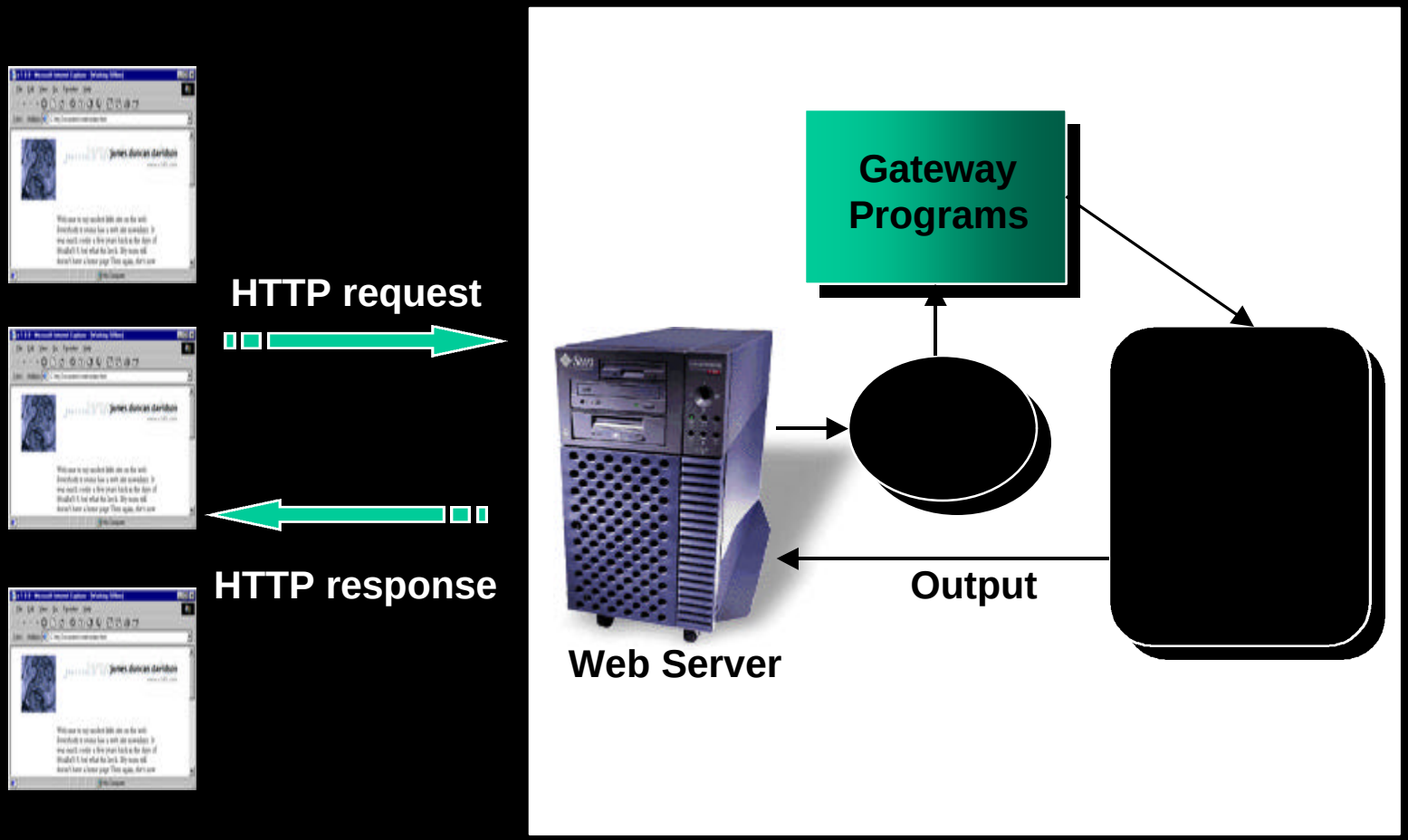
What are Servlets?

- Java™ objects which extend a HTTP server
- A Java Platform Standard Extension
- Alternative for CGI, NSAPI, ISAPI, etc.
- Multiple threads of execution

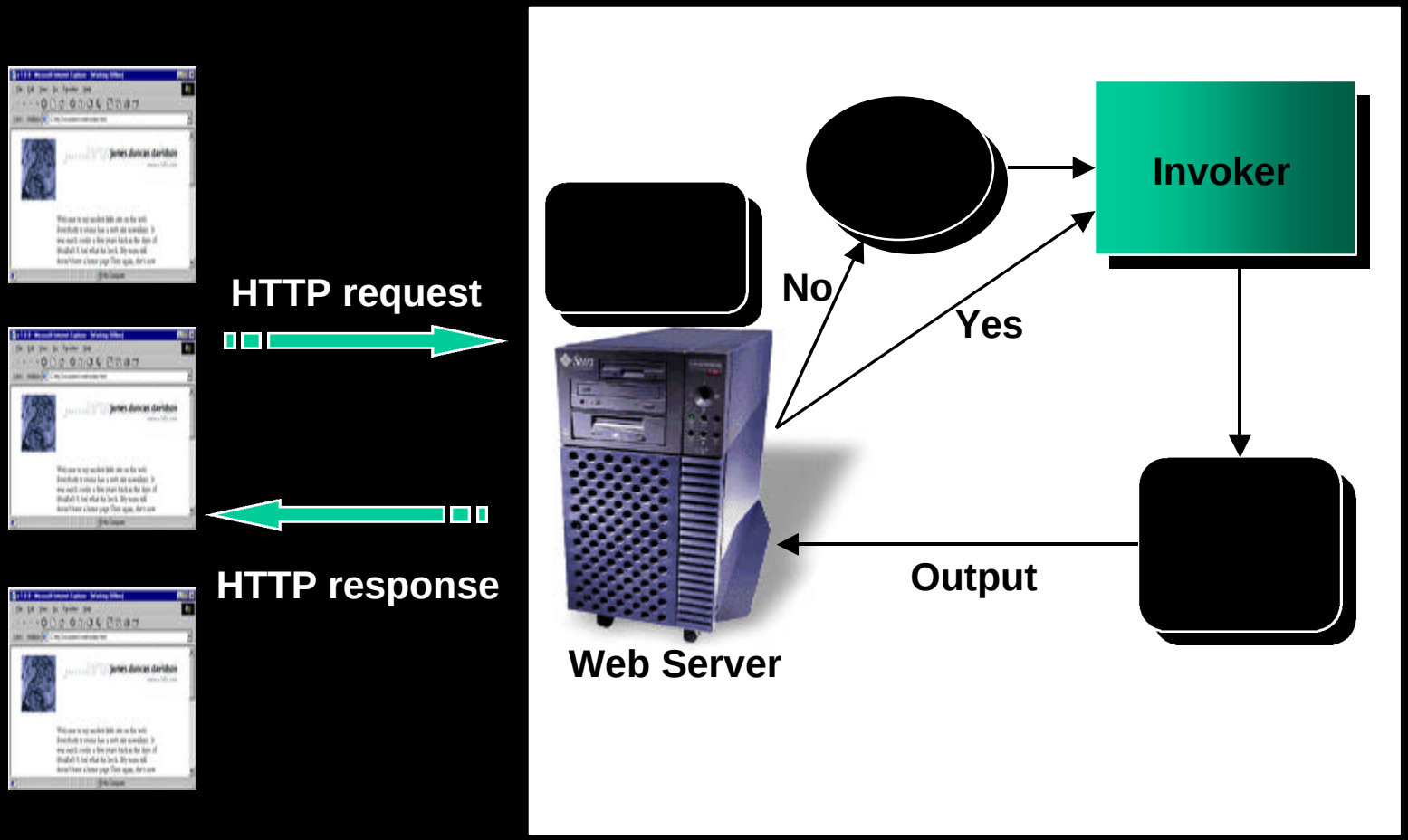
Why Servlets?

- HTTP is a ubiquitous protocol on the web today
- Much less overhead than the CGI model
- Superior alternative to proprietary HTTP server APIs
- Protocol and Platform Independent
- Write Once, Serve Anywhere!

Servlets vs. CGI



Servlets vs. CGI



Servlets are Lightweight

- Can run in the same process as host HTTP Server
- Are fully threadable
- Can be deployed into distributed server environments where RAS (Reliability, Availability, Scalability) are critical

Easy to Develop

- It's Java technology!
- Extensive availability of Java Platform libraries such as JDBC, RMI, EJB, JMS, JavaMail™, etc.
- Extensive third party libraries
- Can develop with the smallest of servlet engines on a laptop, deploy on the most mission critical servers on enterprise class hardware

Easy to Maintain

- Most servers allow reloading of a servlet by administrative action
- Servlets can be remotely loaded allowing several servers to share code

Possible Servlet Applications

- Site wide document management
- Electronic Commerce
- HR Applications
- Conference and chat applications
- Anything else you can put on the Web!

Code for a Simple Servlet

```
// ExampleServlet.java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ExampleServlet extends HttpServlet
{
    public void doGet(HttpServletRequest
      request,HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("Good Day!<BR>");
        Date rightNow = new Date();
        out.println("The time is: " + rightNow);
    }
}
```

Servlets Usage

- Results of JavaWorld™ Poll (December 1998)
  - 600 respondents
  - Top reasons for using servlets
    - Platform Independence
    - Power
    - Performance
    - Ease of use
    - Database access
    - Talk to applets
    - Security

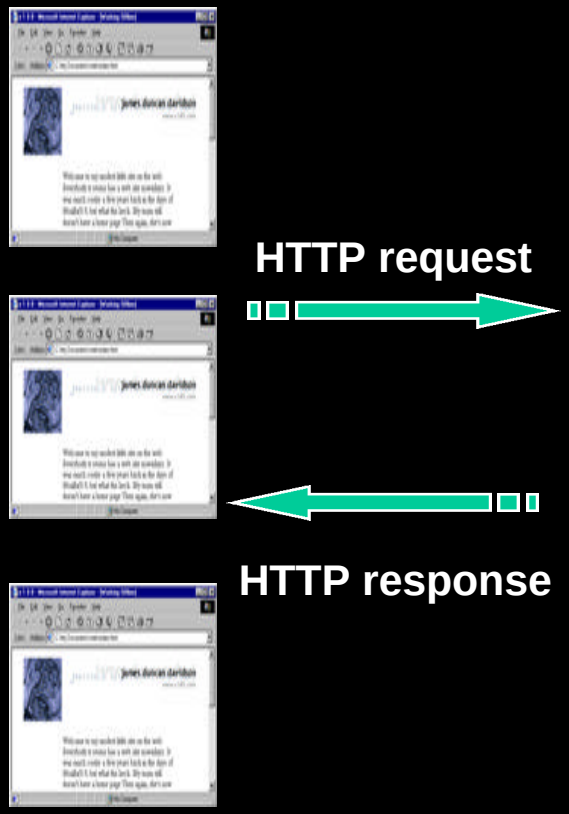
Agenda

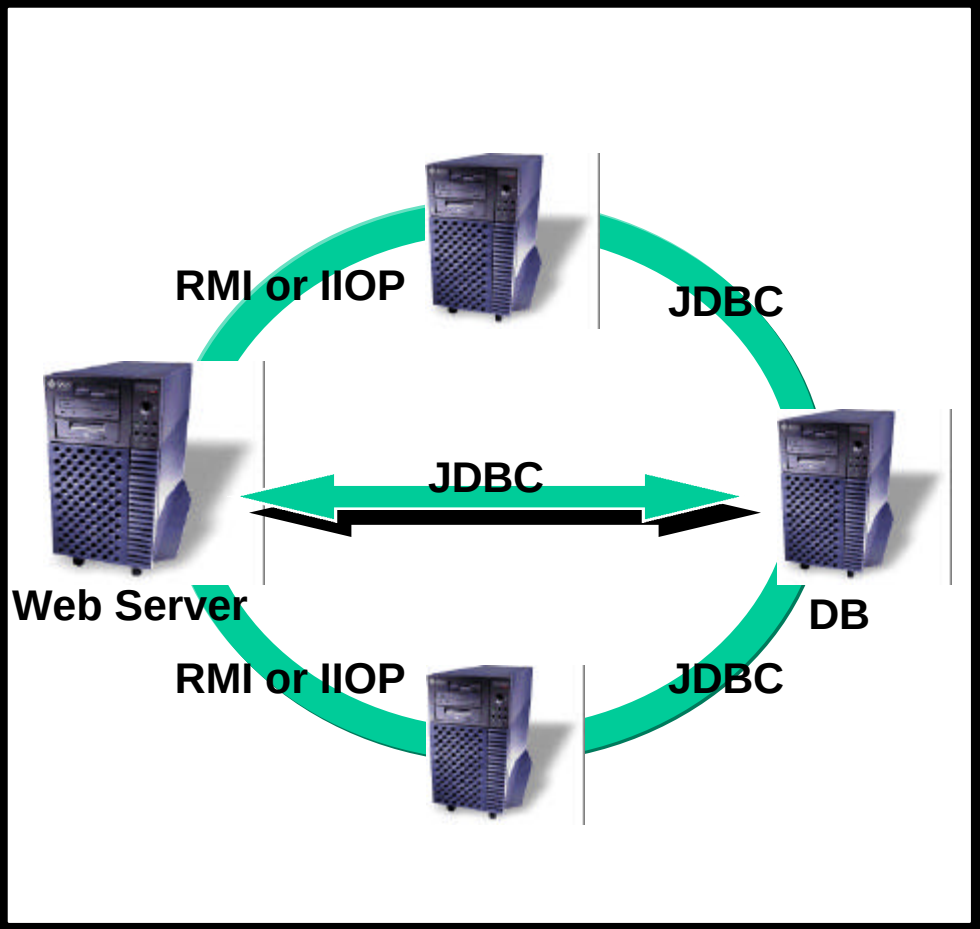
- Servlets
  - Introduction
  - Basics
  - Advanced topics
- JavaServer Pages (JSP)
  - Introduction
  - JSP Syntax
- Current Status and Roadmap
- Conclusions, Q&A

Servlet Basics

- Servlets have well defined lifecycle
- Servlets are managed objects that are loaded on demand and can be unloaded by the server at any time
- Servlets can be mapped to any part of the URL namespace that a servlet engine has control over
- Multiple threads of execution can run through a servlet unless otherwise specified

Illustration





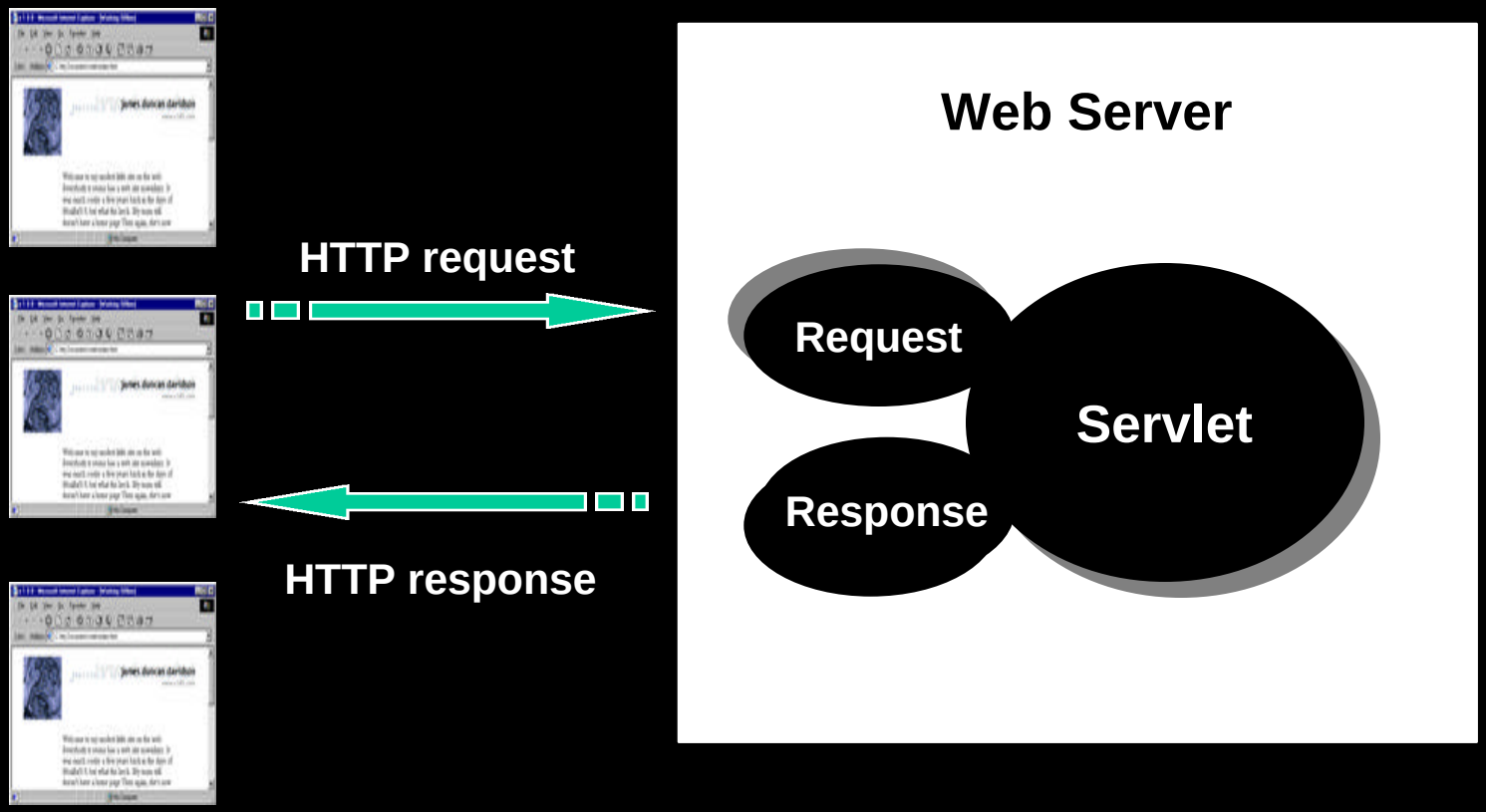
The Servlet Lifecycle

- Servlet is instantiated by the server
- It is initialized via the init() method
- The service() method is called each of client request
- Servlet is given a chance to clean up when being unloaded via a destroy() method

Anatomy of a Request

- A client makes a request on a server
- The request is resolved to a servlet by the server
- The servlet is invoked via the service method with a Request object and Response object
- The servlet provides a response to the request

Illustration



The Request Object

- Encapsulates all information from the client
- Provides access to
  - request headers
  - InputStream or Reader containing data from the client
  - CGI Like information
  - form data and query parameters
  - server specific parameters such as SSL information

Request Methods

```
public interface ServletRequest {

  Enumeration getParameterNames();
  String getParameter(String parameterName);
  String getRemoteAddr();
}

public interface HttpServletRequest extends
ServletRequest {

  String getRequestURI();
  Enumeration getHeaderNames();
  String getHeader(String headerName);
  HttpSession getSessiOn();
  Cookie[] getCookies();
}
```

The Response Object

- Encapsulates all communication back to the client
- Provides access to
  - response headers
  - OutputStream to write data to the client
  - setting cookies
- Convenience method for sending redirects, error pages, etc.

Response Methods

```
public interface ServletResponse {

    PrintWriter getWriter();
    ServletOutputStream getOutputStream();
    void setContentType(String type);
    void setContentLength(int length);
}

public interface HttpServletRequest
extends ServletResponse {

    void addCookie(Cookie cookie);
    void setStatus(int statusCode);
    void sendError(int statusCode);
    void sendRedirect(String url);
}
```

Agenda

- Servlets
  - Introduction
  - Basics
  - Advanced topics
- JavaServer Pages (JSP)
  - Introduction
  - JSP Syntax
- Current Status and Roadmap
- Conclusions, Q&A

Character Encoding Sample

```
// KoreanTest.java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class KoreanTest extends HttpServlet{
public void doGet(HttpServletRequest
                  req,HttpServletResponse res )
   throws ServletException, IOExcep
   res.setContentType("text/html;
   PrintWriter = res.getWriter();
   //PrintWriter out = new
   //PrintWriter(newOutputStreamWriter(
   //res.getOutputStream(),"KSC5601"));
   ...
   out.println( "<H1>" + title + "</H1>" );
   out.println( "</BODY></HTML>" );
   out.close();
} }
```

Session Tracking

- Sessions are a series of requests made by a specific client during a specific time period
- There have been many approaches to session tracking
- Unified in the Servlet API - Sessions are completely managed by the server
- Can use cookies, URL rewriting, or some other future mechanism

Session Tracking Sample

```
// LoginServlet.java

    public void init(ServletConfig config){
        database.put("java","java");
    }


    public void doGet( HttpServletRequest req,
                       HttpServletResponse res )
     throws ServletException, IOException{
     res.setContentType("text/html;charset=euc-kr");
     PrintWriter out = res.getWriter();
     out.write( first );
     out.write( second );
     out.close();
    }

// continue...
```

Session Tracking Sample (contd.)

```
  public void doPost(...) throws Exception {
  //Check ID and password,or Connect Database
  //After sessioncreate,redirect SessionServlet
  HttpSession session = req.getSession(true);
  String url = req.getRequestURI();
  int ind = url.lastIndexOf('/');
  res.sendRedirect(url.substring(0, ind+1) +
                          "SessionServlet");
  return;
  }
}  // end class
```

Session Tracking Sample (contd.)

```
//SessionServlet.java

 // Session retrieve or creation
 HttpSession session = req.getSession(false);
 // The first session redirect login.html.
 if ( session == null ) {
     String url = req.getRequestURI();
     int ind = url.lastIndexOf('/');
     res.sendRedirect(url.substring(0, ind+1)
                          +"LoginServlet");
     return;
 }
 out.println("<head><title>"+"SessionServlet"+
                 "</title></head><body>");
 out.println("<h1>SessionServlet result</h1>");

//continue ...
```

Session Tracking Sample (contd.)

```
 // Session ID, Session Data
    ...
    out.println("<br>Session ID:"+
    session.getId());
    Integer ival=
    (Integer)session.getValue("counter");
    if (ival == null)
        ival = new Integer(1);
    else
        ival = [illegible]er(ival.intValue() + 1);
        session.p[illegible]("counter", ival);
    ...
    }
}// end class
```

Cookie Tracking Sample

```
//LoginServlet.java

public void doPost(...) throws Exception{
    if(...){
        Cookie                    Cookie("web","logged");
        res                       ook);
         out.                  n("</H3></body></html>");
                    out.close();
    } else {
        res.sendRedirect(“... login.html");
    }
  ...
}
```

Cookie Tracking Sample (cont'd)

```
//CheckLogin.java

public void doGet(.....)                    ...ption{
 Cookie[] cookies=req.g...                  ...();
 if(cookies!=null){
   for( int i=0;i<cookies.length;i++){
    if(cookies[i].getName().equals("web") ||
      (cookies[i].getValue().equals("logged")){
      res.sendRedirect("http://www.sun.co.kr");
    }
   }
 }
 res.sendRedirect(”...login.html");
}
```

URL rewriting Sample I

```
//URLServlet.java

 public void doGet(...) throws Exception{
   HttpSession session=req.getSession(false);
   res.setHeader("pragma","no-cache");
    ...
   // URL rewriting ...
   out.println("<form name=\"form\"
   method=\"POST\" action=
   res.encodeURL(req.                    ())+"\">");
     ...
 }

continue...
```

URL rewriting Sample I (cont'd)

```
public void doPost(...) throws Exception{
 HttpSession session = req.getSession(true);
 if(session.isNew()){
   session.putValue("Book1", new int[] { 0 });
   session.putValue("Book2", new int[] { 0 });
 }
 ...
 if(...){ book1[0]++; ...
 }else if(...){  book2[0]++; ...
 } else if(req.getParameter("buy") != null){
  session.invalidate();  // session list remove
  ...
  res.setHeader(”Cache-Control", "no-cache");
} // end class
```

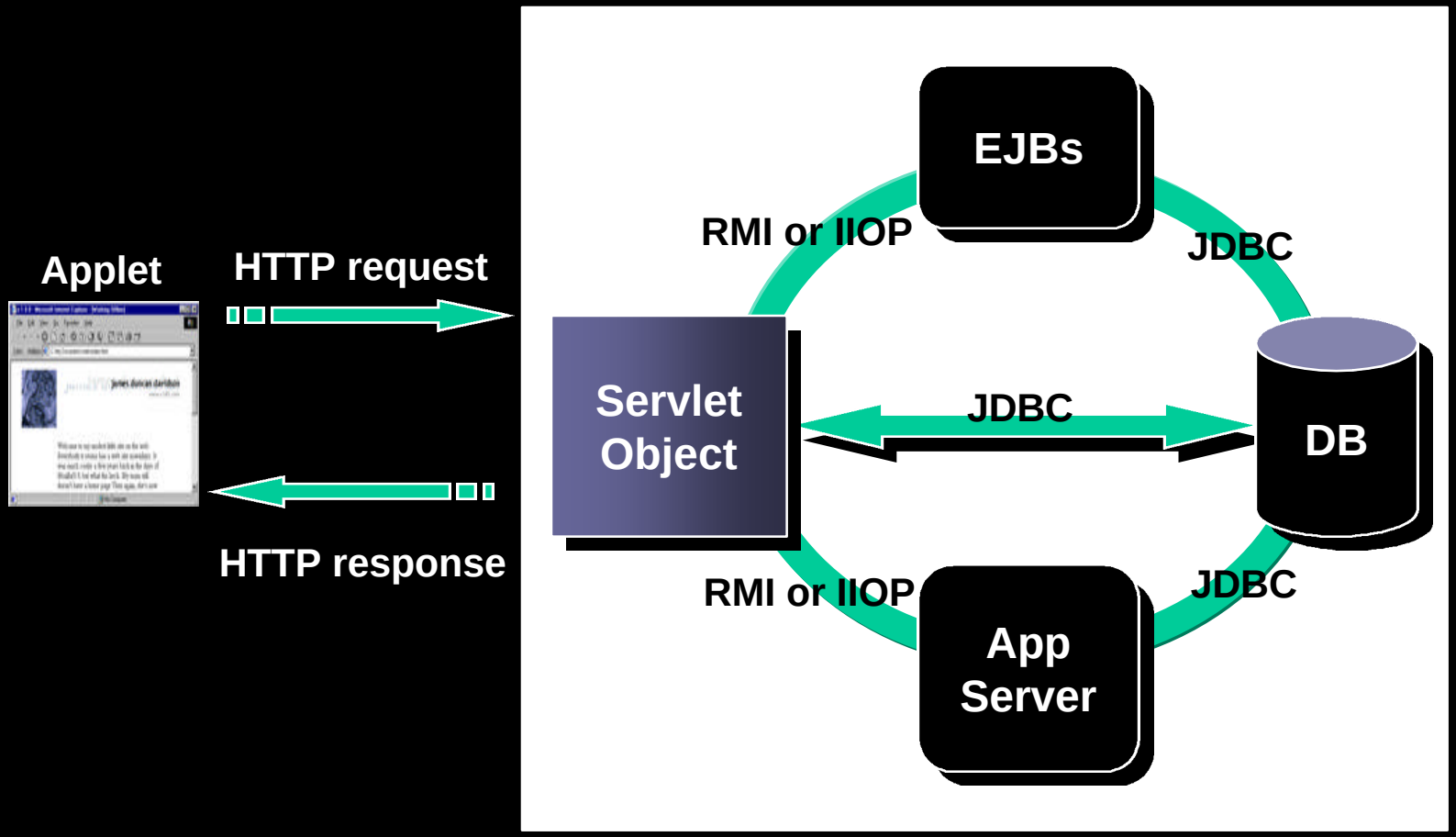
URL rewriting Sample II

```
//URLServlet2.java

public void doGet(...) throws Exception {
  HttpSession session=req.getSession(false);
  String session_id=null;
  if(session !=null){
        session_id=session.getId();
  }
  // URL rewriting ...
  out.println("<form name=\"form\"
      method=\"POST\" action=\"”+
      res.encodeURL(req.getRequestURI())+”
                    +session_id+"\">");
}
```

Client Servlet Communication

- Two primary methods of applet/servlet communication:
  - Text based using java.util.Properties
  - Binary based using object serialization
- The benefit of using Properties is that non Java Platform based clients can interoperate
- The benefit of using serialization is that complex data structures can be transmitted easily

Illustration



Client Sample

```
// PhoneApplet.java

public void callServlet() {
URL phoneServlet=new URL(getDocumentBase(),
                         "servlet/PhoneServlet");
   ServletMessage phone=
   new ServletMessage(phoneServlet);
   InputStream result =
   phone.sendMessage(args,ServletMessage.POST);
   DataInputStream in=
   new DataInputStream(result);
  ...
}
```

Client Sample (contd.)

```
// ServletMessage.java


    if (method == GET) {
            URL url = new URL(servlet.toExternalForm() + "?" +
                    URLEncoder.encode(args));
            return url.openStream();
     } else {
            URLConnection conn = servlet.openConnection();
            conn.setDoOutput(true);
            conn.setUseCaches(false);
            // POST the request data (html form encoded)
            PrintStream out =
                        new PrintStream(conn.getOutputStream());
             if (args != null && args.size() > 0) {
                  out.print(URLEncoder.encode(args));
             }
            out.close(); // ESSENTIAL for this to work!

            // Read the POST response data
            return conn.getInputStream();
     }
```

Server Sample

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException{
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();
        String name = req.getParameter("name");
        String title = "Phones";
        out.println("<HEAD><TITLE> Phone Servlet Output
                    </TITLE></HEAD><BODY>");
        out.println("<h1> PhoneServlet output </h1>");
        out.println("<p>");

        if (phones == null) {
            out.println("No Phone List!");
        } else if (name != null) {
            String phone = (String) phones.get(name);
            if (phone == null) phone = "Not listed";
            out.println(name + ": " + phone);
        } else
            out.println(phones.toString());

       out.println("</BODY>");
}
```

Using Serialization

```
// servlet side
{
  Result result = getResult(); // generate object
  ServletOutputStream out = response.getOutputStream();
  ObjectOutputStream oOut = new ObjectOutputStream(out);
  oOut.writeObject(result);

}

// client side
{
  URL url = new URL("http://myserver/Resultservlet");
  InputStream in = url.openStream();
  ObjectInputStream oIn = new ObjectInputStream(in);
  Result result = (Result)oIn.readObject();
}
```

Enterprise Role of Servlets

- Unfortunately RMI and IIOP don't go over firewalls well
- Every client has a browser, not every client has the ability to run your application code
- Servlets serve as the middle tier speaking HTTP between any kind of thin client and large enterprise services made available via EJB

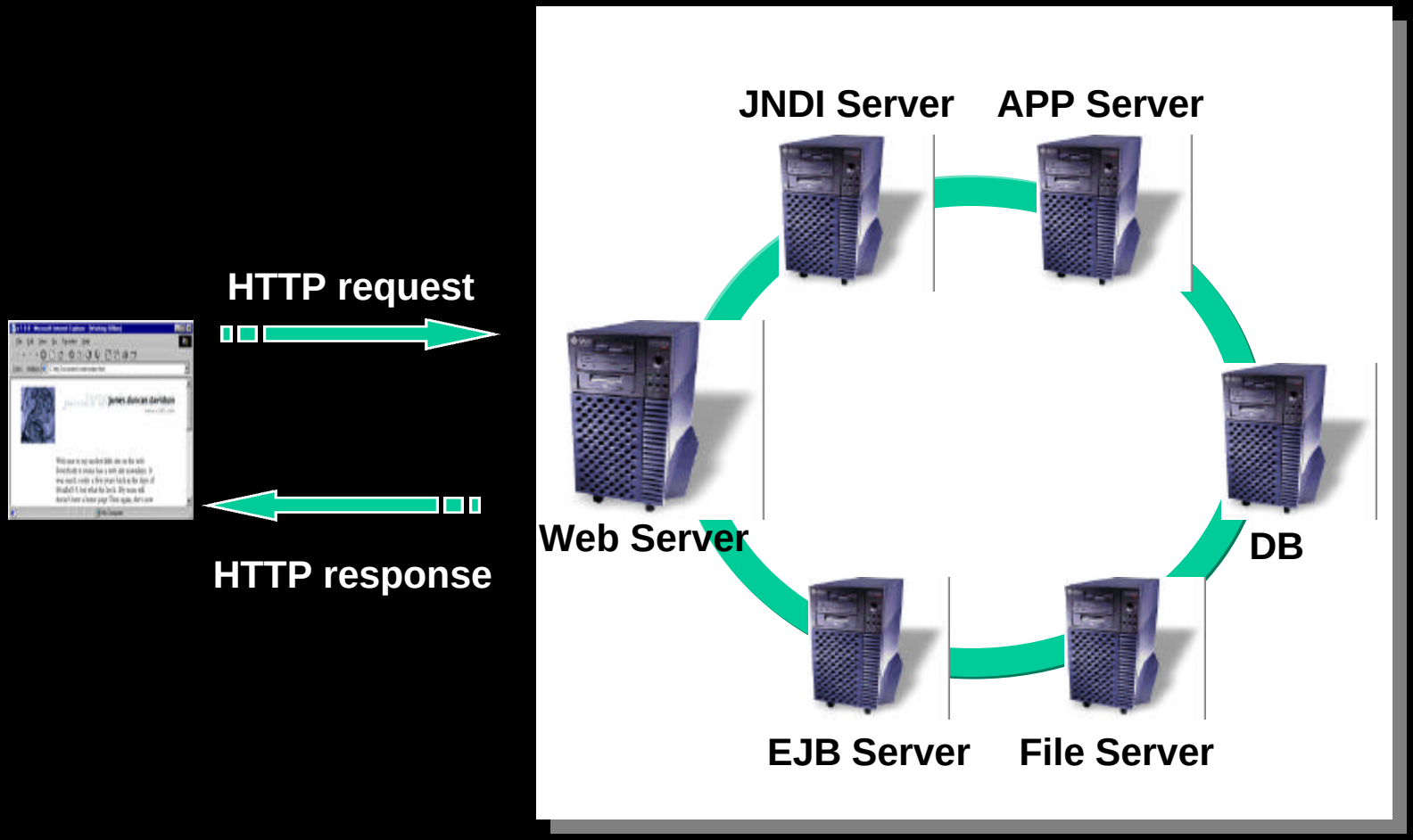
JDBC Sample

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class DbServlet extends HttpServlet{
    static final String url = "jdbc:oracle:thin:@203.234.247.1:1521:DEMO";
    static final String jdbcclass = "oracle.jdbc.driver.OracleDriver";
    static final String query = "SELECT * FROM Customer";
    Connection con = null; Statement stmt = null; ResultSet rs = null;

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        try {
          Class.forName(jdbcclass);
        } catch (Exception e) {}
    }
    public void doGet(HttpServletRequest req,HttpServletResponse res){
        res.setContentType ("text/html; charset=euc-kr");
        try {
              con = DriverManager.getConnection (url,"scott","tiger");
              stmt = con.createStatement();
              rs = stmt.executeQuery(query);
              PrintWriter out = res.getWriter ();
              writeHeader(out);  writeBody(out,rs);  writeEnd(out);
               out.close ();
        } catch (Exception ioe){}
    }
continue ...
```

JDBC Sample (contd.)

```
    public void writeHeader(PrintWriter out){
         out.println("<html><head>");
         out.println("<title>DB Servlet</title>”);
         out.println("</head><body>");
    }
    public void writeBody(PrintWriter out,ResultSet rset)
    {
         try{
            while(rset.next()) {
                  out.println(“Name   : “+rset.getString(1));
                  out.println(“ID     : “+rset.getString(2));
                  out.println(“Passwd : “+rset.getString(3));
            }
         }catch(Exception e){}
    }

    public void writeEnd(PrintWriter out)
    {
              out.println("</body></html>");
              out.println("</body></html>");
    }
} //end class
```

Illustration



Agenda

- Servlets
  - Introduction
  - Basics
  - Advanced topics
- JavaServer Pages (JSP)
  - Introduction
  - JSP Syntax
- Current status and Roadmap
- Conclusions, Q&A

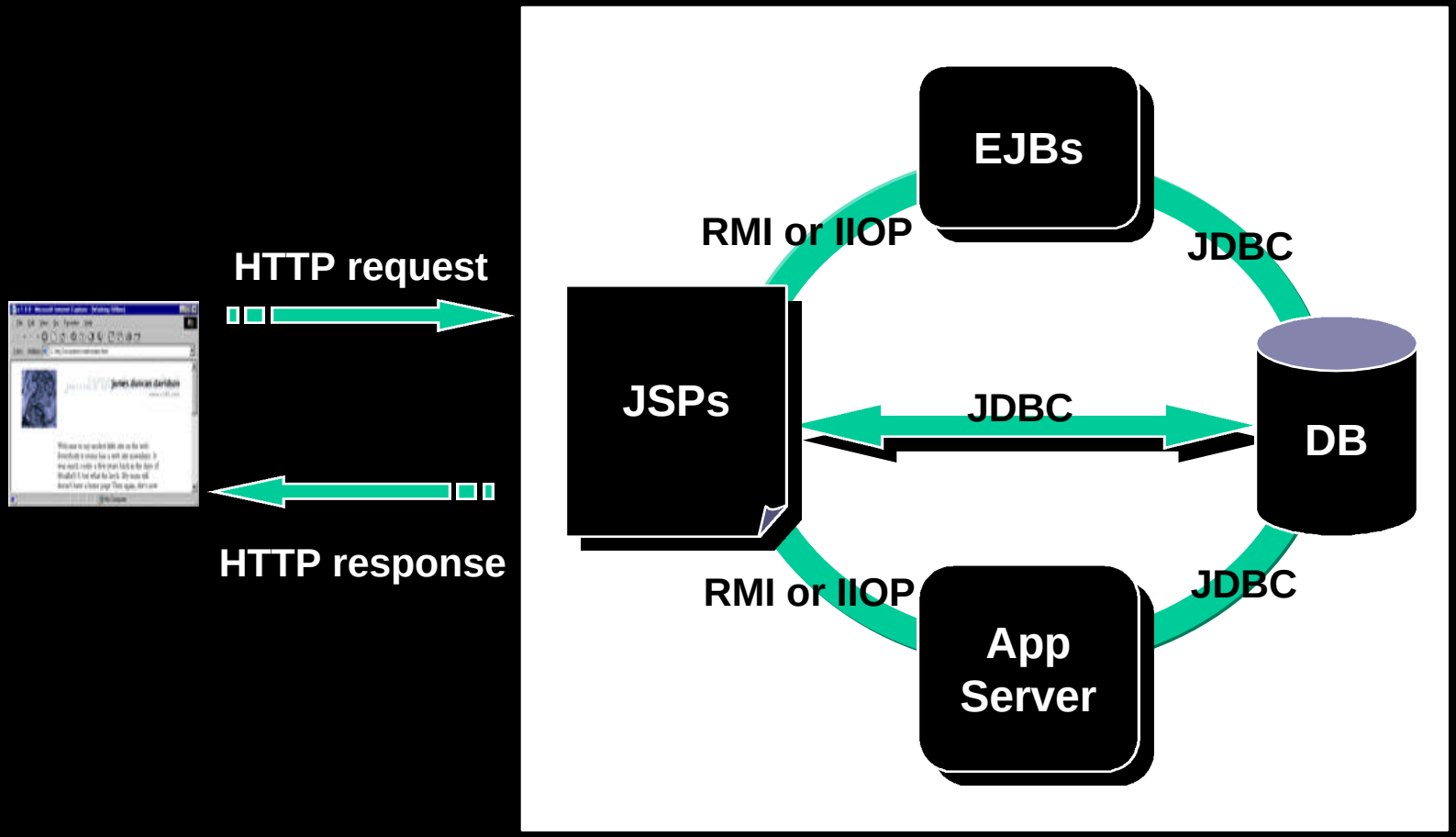
What is JSP

- Web pages containing a combination of HTML and code
- Standard Web Access Layer to J2EE
- Builds on Java Servlet technology
- Adds dynamic content generation capabilities to static templates
- Leverages JavaBeans ™

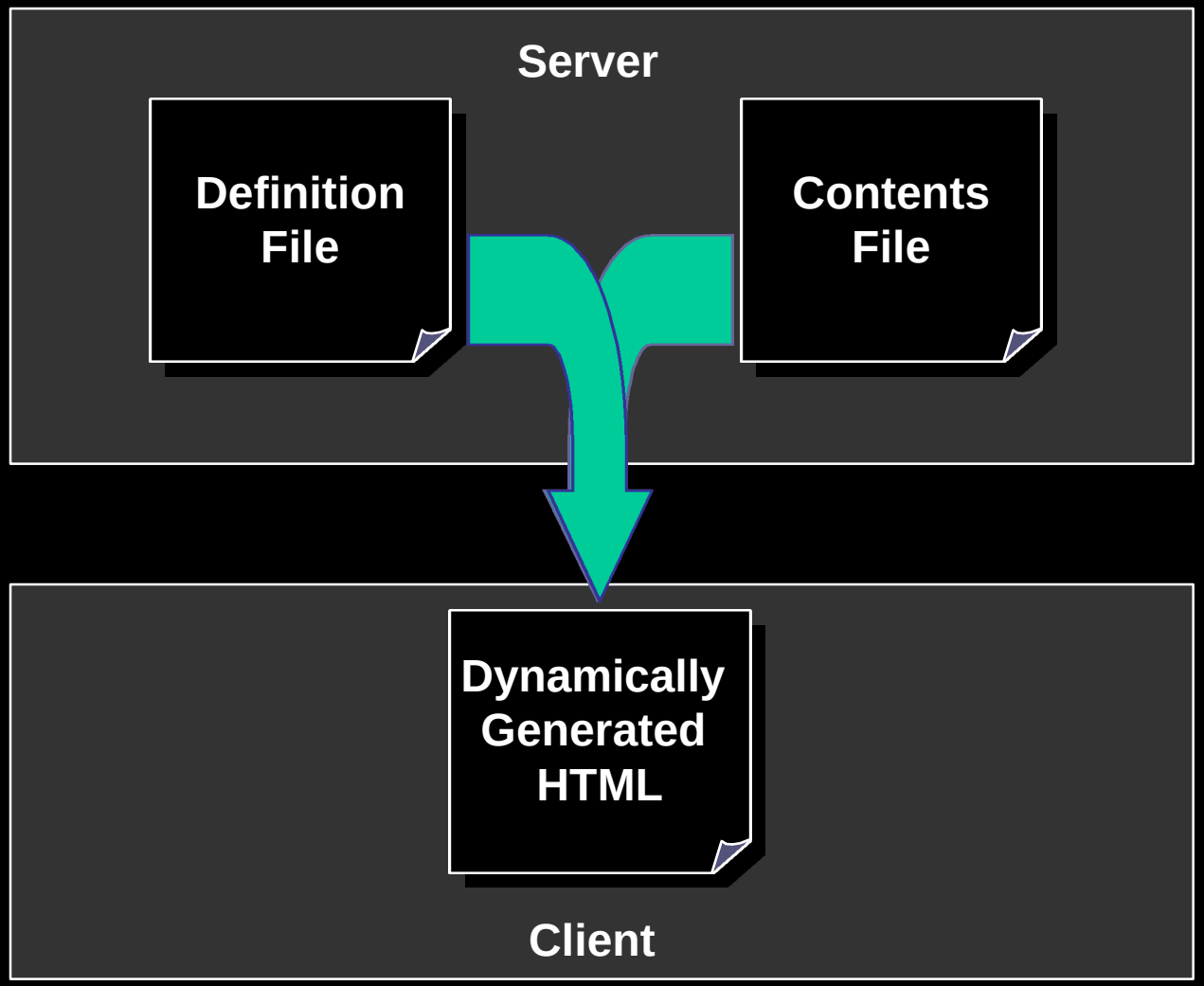
Benefits of using JSP

- Cleanly separate presentation from content
- Are automatically recompiled when source file changes
- Extensive availability of Java Platform libraries such as JDBC, RMI, EJB, JMS, JavaMail™, etc.

Illustration



Illustration

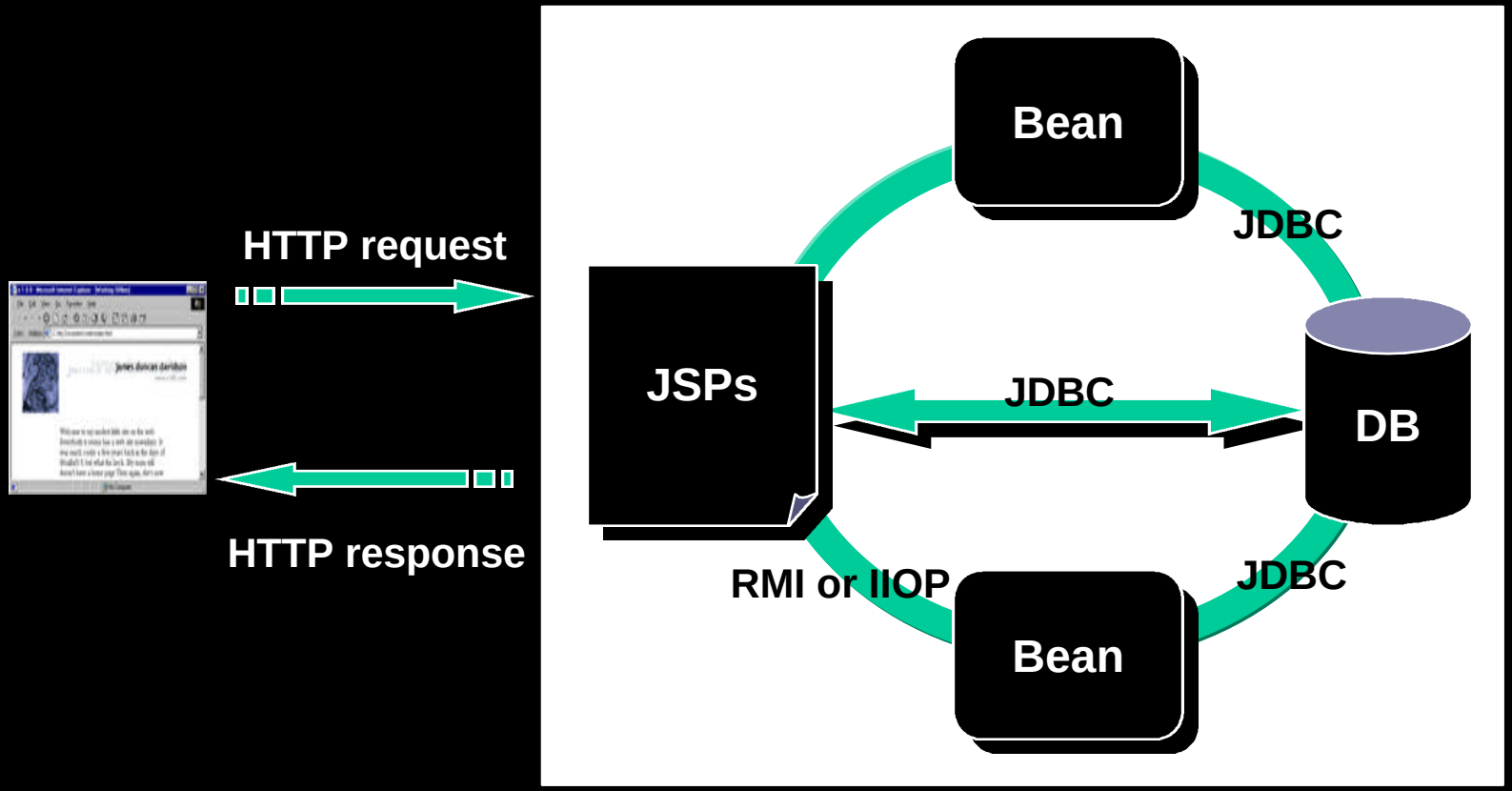


JSP and Servlets

- JSP turns Servlets inside out
  - JSP is compiled into a servlet by the server
  - JSP is a scripting language
  - JSP is for text output such as HTML and XML
  - Java-like code is embedded in a HTML page
  - Uses JavaBeans as reusable server component

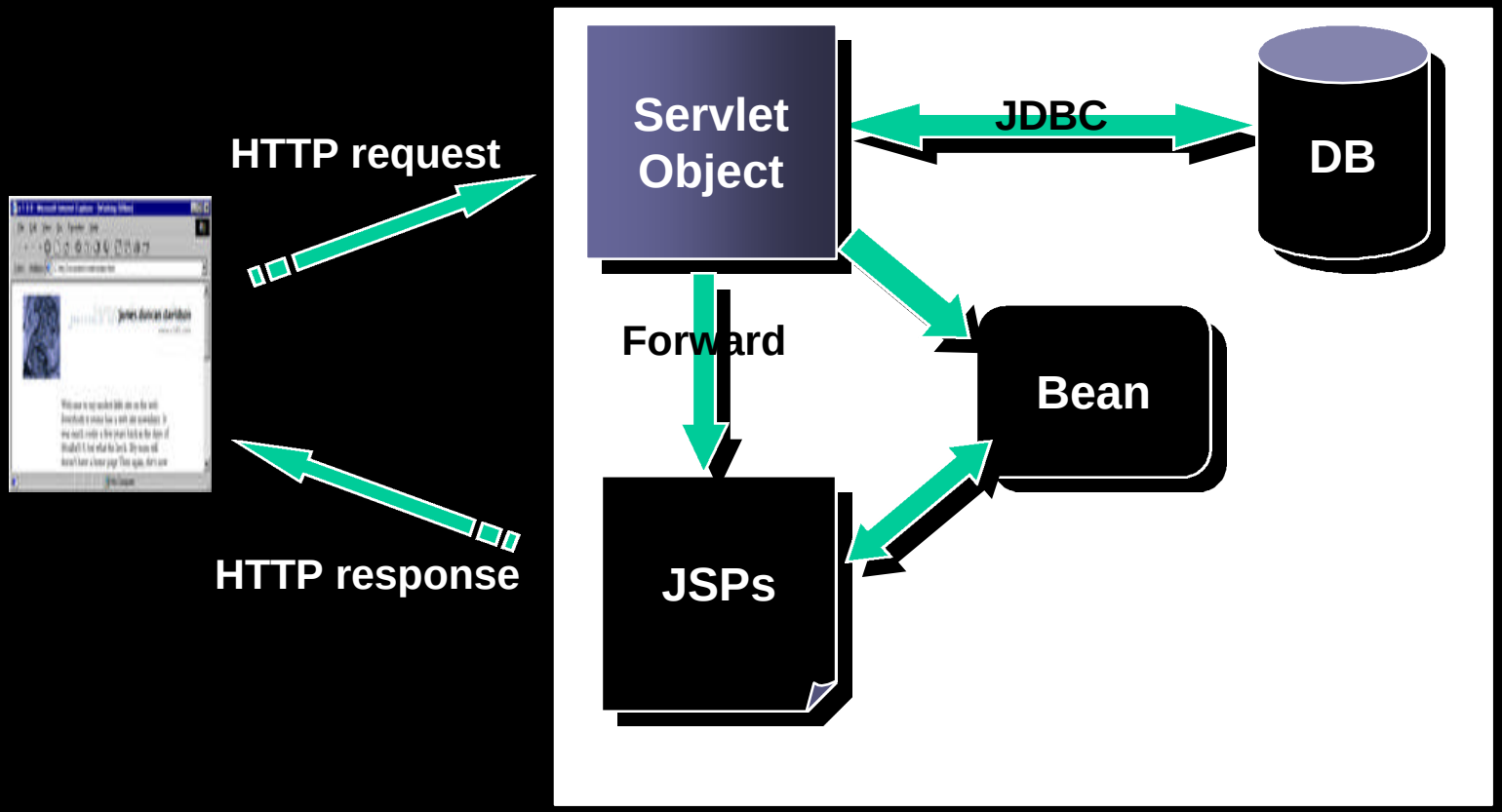
JSP Usage - Scenario 1

- The JSP page directly services the request



JSP Usage - Scenario 2

- The request is first handled by a Servlet, then forwarded to a JSP



JSP Elements

- Static text
- Scripting elements
  - Java Code
  - JavaScript™, others
- Core tags and Directives
- Custom tags

Code for a Simple JSP

```
<HTML><BODY><H1>JSP Form</H1>
    <%@ include file="index.html" %>
    <%@ page  language="java" import = "java.lang.*" %>
    <% if (request.getParameter("first") == null) { %>
<FORM action="jspform.jsp" method="get">
<p> First Name <p>
<INPUT type="text" name="first">
<INPUT type=submit value="submit">
<INPUT type=reset>
</FORM>
    <%} else {
        if (request.getParameter("first").equals("")) { %>
<p>You MUST fill in each field before submitting the form:
Press the Back button and fill in the empty fields in
your form.
    <%} else { String foo = request.getParameter("first");
        out.println(foo); }%>
    <%}%>
</HTML>
</BODY>
```

Agenda

- Servlets
  - Introduction
  - Basics
  - Advanced topics
- JavaServer Pages (JSP)
  - Introduction
  - JSP Syntax
- Current Status and Roadmap
- Conclusions, Q&A

Core Syntax

- JSP Standard directives
- JSP Standard tags
- Script language declarations
- Script language scriptlets & expressions
- Implicit Object
- Experimental tag extension

JSP Directives

- Controls the behavior of the JSP Engine

```
<%@ page directive="value"...%>
```

- Directive Types
  - Language
  - Page
  - Include
  - Tag Libraries

JSP Directives (contd.)

- language tag

```
<%@ page language="java" %>
```

- method variable

```
<%@ page method="doPost" %>
```

- import variable

```
<%@ page import="java.net.*,java.io.*" %>
```

- content Type variable

```
<%@ page
contentType="text/html;charset=KSC5601" %>
```

- extends variable

```
<%@ page
extends="java.lang.Thread" %>
```

JSP Directives (contd.)

- Specify page-specific attributes
  - session, buffering, threading, info, error handling
- Example:

```
<%@ page buffer="none" threadsafe="yes"
  errorpage="/murphy.jsp" %>
```

- Substitute text or code at JSP page processing time

```
<%@ include file="copyright.html %>
```

JSP Tag Library Directive

- Identify custom tag library
- Example:

```
<%@ taglib uri="http://www.wombat.com/taglib"
prefix="wombat" %>

<public:wombat>
.........
</public:wombat>
```

JSP Scripting Elements

- Declarations

```
<%! QuoteBean q; %>
```

- Scriptlets

```
<% q = new QuoteBean(); %>
```

- Expressions

```
<%= q.getQuote("SUNW"); %>
```

- Example:

```
<% int number=20;
   int i = number + 1; %>
<p> The result of the computation is:<%= i%>
```

Implicit Objects

- Objects defined implicitly for use in scripting tags
  - request, response
  - pagecontext, appcontext
  - session
  - out
  - in error pages: exception

Example:

```
<%if(request.getParameter("first")==null){ %>
<% out.println("String"); %>
```

Standard Tags

- USEBEAN
  - Declare the usage of an instance of a JavaBean
  - Locates or instantiates a Bean with a specific name and scope
- Example:

```
<jsp:useBean id="cart" scope="session”
class="sessions.DummyCart” />
```

Standard Tags (contd.)

- SETPROPERTY & GETPROPERTY
  - Manipulate the properties of a declared bean instance

```
<jsp:setProperty name="cart" property="*" />
<jsp:getProperty name="cart" property="*"
/>
```

- REQUEST
  - forward to other URI
  - include the output from other URI

```
<jsp:forward page=”pathToFile”/>
<jsp:include page=”pathToFile”/>
```

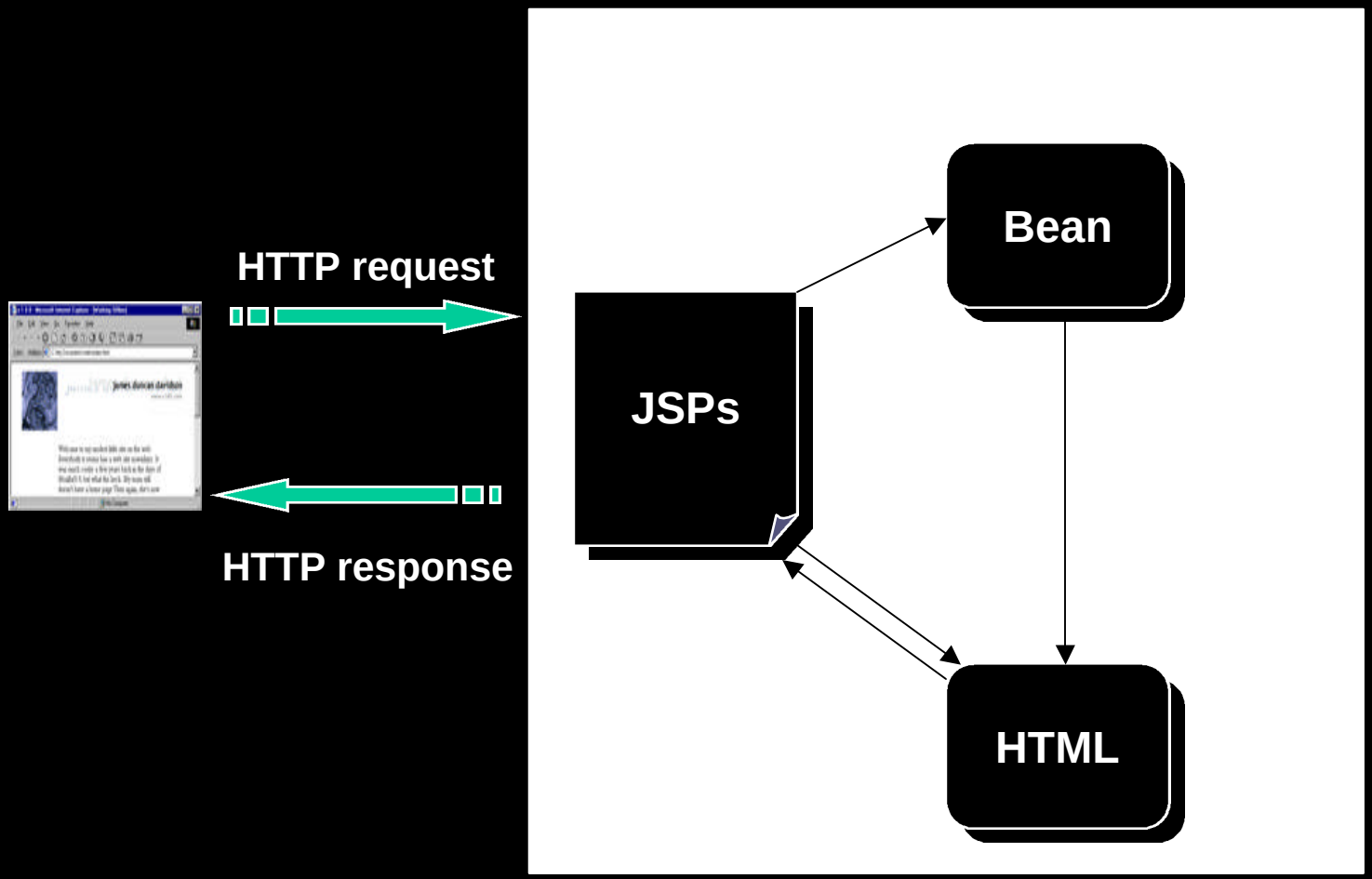
Standard Tags (contd.)

- PLUGIN
  - Download a Java plugin to the client Web browser to execute an applet or Bean.

Example:

```
<jsp:plugin type="bean|applet" code="objectCode"
codebase="pathObjectCode"
        [name="instanceName"][archive="archiveList"]
        [height="displayHeight"][width="displayWidth"]
        [jreversion="JREVersionNumber|1.1"]
        [<params><param name="name"
value="value"></params>
        </jsp:plugin>
```

Standard Tags Examples



Standard Tags Examples (contd.)

```
<html>
<jsp:useBean id="cart" scope="session"
                class="sessions.DummyCart" />
<jsp:setProperty name="cart" property="*" />
      <% cart.processRequest(request); %>
<FONT size = 5 COLOR="#CC0000">
<br> You have the following items in your cart:
<ol>
      <% String[] items = cart.getItems();
         for (int i=0; i<items.length; i++) {
      %>
<li>  <%= items[i] %>
      <%  } %>
</ol>
</FONT>
<hr>
      <%@ include file="carts.html" %>
</html>
```

Standard Tags Examples (contd.)

```
<html><head><title>carts</title></head>
<body bgcolor="white">
<font size = 5 color="#CC0000">
<form type=POST action=carts.jsp>
<BR>
Please enter item to add or remove:
<br>
Add Item:
<SELECT NAME="item">
<OPTION>Beavis & Butt-head Video collection
<OPTION>X-files movie
<OPTION>Twin peaks tapes
<OPTION>NIN CD
<OPTION>JSP Book
<OPTION>Concert tickets
<OPTION>Love life
<OPTION>Switch blade
<OPTION>Rex, Rugs & Rock n' Roll
</SELECT>
<br> <br>
<INPUT TYPE=submit name="submit" value="add">
<INPUT TYPE=submit name="submit" value="remove">
</form></FONT></body></html>
```

Standard Tags Examples (contd.)

```
package sessions;
import javax.servlet.http.*;
import java.util.*;
import java.io.*;
public class DummyCart implements Serializable{
    Vector v = new Vector();
    String submit = null;
    String item = null;
    private void addItem(String name) { v.addElement(name); }
    private void removeItem(String name) { v.removeElement(name); }
    public void setItem(String name) {  item = name;    }
    public void setSubmit(String s) {   submit = s;    }
    public String[] getItems() {
          String[] s = new String[v.size()];
          v.copyInto(s);
          return s;
    }
    public void processRequest(HttpServletRequest request) {
          if (submit == null) addItem(item);
          if (submit.equals("add")) addItem(item);
          else if (submit.equals("remove"))  removeItem(item);
          reset();
    }
    private void reset() {
          submit = null;
          item = null;
    }
}//end class
```

Custom Tags

- Encapsulate functionality in tags
- Easy to
  - author manually
  - manipulate by tool
- Portable semantics
- Portable tool support

Custom Tags Example

```
<connection id="con01" ref="foo.xml"
        userid="<%=session.getUserid()%>"
        password="<%=session.getPassword() %>"/>
<%con01.getSomeConnectionAttribute();%>

<query id="q" connection="con01">
    SELECT account, balance FROM acct_table WHERE
    customer_number= <%= request.getCustom() %>
</query>
```

```
<ul>
<foreach row="row" in="q">
    <li> The balance for <%= row.account%> is
<%=row.balance%>
</foreach>
</ul>
```

Session Tracking Example

```
<html>
<head><title>Session Tracking Test</title></head>
<body>
  <%
    session = request.getSession (true);
  %>
<h1>Session Tracking Test</h1>
  <%
    Integer ival =
    (Integer)session.getValue("counter");
    if (ival == null) ival = new Integer (1);
    else ival = new Integer (ival.intValue () +  1);
    session.putValue ("counter", ival);
  %>
This page accessed <%= ival %> times.
</body></html>
```

Agenda

- Servlets
  - Introduction
  - Basics
  - Advanced topics
- JavaServer Pages (JSP)
  - Introduction
  - JSP Syntax
- Current Status and Roadmap
- Conclusions, Q&A

Current Servlet API Status

- Servlet API 2.1 Specification released November 4th 1998
- Java Servlet Development Kit Version 2.1 Released
- Third party support for 2.1 arriving soon!
- Java Servlet Specification Version 2.2 Public Released 25-August-99

Servers Supporting Servlets

- Java Web Server™
- Sun WebServer™
- Netscape Application Server
- The Apache Project
- Oracle 8i
- IBM WebSphere
- Zues Web Server
- BEA Weblogic Tengah
- KonaSoft Enterprise Server
- ATG Dynamo Application Server
- Novacode NetForge
- W3C Jigsaw
- More!

Servlet Engines

- Support for IIS and Netscape enterprise
  - NewAtlanta ServletExec (www.newatlanta.com)
  - LiveSoftware Jrun (www.livesoftware.com)
  - WAI Cool Runner(www.gefionsoftware.com)

Web Server Support Servlet 2.1

Liteweb server (www.gefionsoftware.com)
jo! web server (www.webapp.de)
Jetty web server (www.mortbay.com)
Jigsaw web server (www.w3.org/Jigsaw)

Servlet Partners

- Sun Microsystems, Inc.
- IBM
- Oracle
- Netscape
- BEA Weblogic
- Art Technology Group
- The Apache Group
- Live Software
- New Atlanta Communications
- Other Individual Industry Leaders

Servlet Futures

- Deployment Descriptors for deploying servlets, their related class files, and content into any server
- Servlet Engine Compatibility tests
- Integration with the J2EE Reference Implementation

JSP Specification Status

- Designed using the Java Platform process
- Reassessment of spec.
- Public Release of the JSP 1.0 Specification
- JavaServer™ Web Development Kit (JSWDK1.0: JSP1.0 & Servlet 2.1)
- Public Release of the JSP 1.1 Specification(25-August-99)

JSP Implementation Status

- Complete compliance with JSP 1.0 spec
- Experimental tag extensions
- Integration with the J2EE Reference Implementation
- Integration with Apache
  - The Jakarta Project (http://jakarta.apache.org)

JSP and XML

- JSP 1.0 pages
  - can be transformed into XML docs
  - will enable using XML concepts/tools
- All standard tags
  - have an XML acceptable equivalent

JSP 1.1

- Improved integration in J2EE
  - packaging
  - deployment
  - security
- Support for Application
- Add standard custom tags
- Use Servlet2.2 instead of Servlet 2.1

Agenda

- Servlets
  - Introduction
  - Basics
  - Advanced topics
- JavaServer Pages (JSP)
  - Introduction
  - JSP Syntax
- Current Status and Roadmap
- Conclusions

Servlet Resources

- Websites
  http://java.sun.com/products/servlet/
  http://developer.java.sun.com
- Book(s)
  - Java Servlet Programming by Jason Hunter
- servlet-interest mailing list
  servlet-feedback@eng.sun.com

JSP Resources

- Websites
  http://java.sun.com/products/jsp
  http://developer.java.sun.com
- jsp-interest mailing list
  jsp-feedback@eng.sun.com

Q & A