

Unit 4. CRM - Web Marketing

What This Unit Is About

Identify/utilize the components of the framework to build and run Web Marketing solutions

What You Should Be Able to Do

After completing this unit, you should be able to:

- Define the Web Marketing solution space
- Identify the technologies used for creating a Web Marketing solution
- Identify the framework components that help create a Web Marketing solution
- Utilize WebSphere Development Studio to create a Web site
- Utilize Domino Designer to create a Web site

How You Will Check Your Progress

Accountability:

- Complete the labs

References

- WebSphere Development Studio
<http://www.ibm.com/software/webservers/studio/>
- Domino Server
<http://lotus.com/home.nsf/welcome/dominodesigner>
- IBM HTTP Server
<http://www.ibm.com/software/webservers/httpservers/index.html>
- Domino Application Server
<http://www.lotus.com/home.nsf/welcome/domino>

Objectives

- Define Customer Relationship Management - Web Marketing
- Identify the components of a Web Marketing solution
- Understand key technologies used by Web Marketing clients
- Discuss important web protocols
- Understand logical session support
- Understand the role of an HTTP server

Figure 4-1. Objectives (in212400)

Notes:

4.1 The Solution Space

Agenda

- The Need
 - How do I get my message to prospective clients/partners?
 - How do I provide personalization, consistent look & feel?

- Technology

- Client

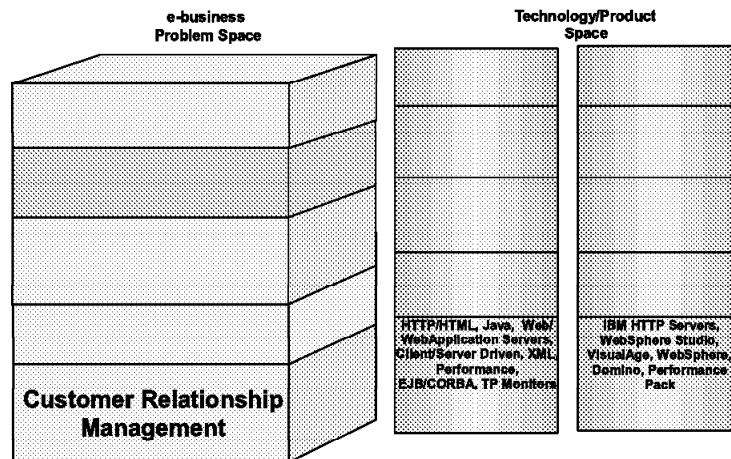
- Web browsers
- HTML
- DHTML
- Script
- CSS
- DOM
- Java applets

- Network

- Web protocols
- Session state
- Cookies

- Server

- HTTP server



- Products

- WebSphere Studio
- Domino Designer
- IBM HTTP Server

Figure 4-2. Agenda (in212405)

Notes:

We'll be looking at the technologies involved in application that run on the World Wide Web. These center around the use of HTML documents delivered via the HTTP protocol. We'll examine Web browsers and the application code that runs on them, and we'll look at the network protocols used to deliver HTML content. Finally we'll map the products from the framework into these technology blocks.

The Need

- Web Marketing = 'brochureware'
- Client-driven application model
 - Request/response only
 - Limited reuse of resources
 - Limited access to existing applications and data
- Static content, dynamic presentation
- Content can be updated via off-line processes

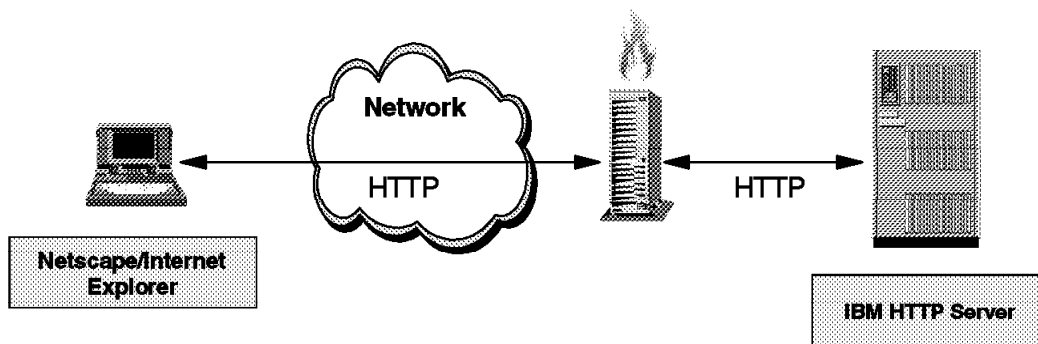


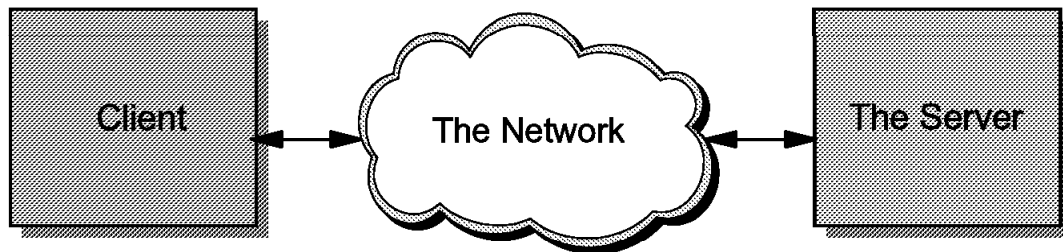
Figure 4-3. The Need (in212410)

Notes:

Customer Relationship Management - Web Marketing attempts to satisfy the need to make information available to customers, prospects and partners via the Internet. Like a television commercial or a magazine ad, the content is fixed. However, unlike a these types of information delivery, the navigation of content is controlled by the intended target (the client) and may be presented in a customized format. Although a Web Marketing site may "feel" interactive, there is very little provision for interaction with other applications or data sources.

4.2 The Technology

The Technology - 3 Building Blocks



- ▶ The "Browser"
 - ▶ Web browser
 - ▶ "Personal" device
 - ▶ Makes Request
 - ▶ Controls the "session"
 - ▶ Universal Client
- ▶ Everything between the server and the User Agent
 - ▶ HTTP
 - ▶ TCP/IP
 - ▶ Ethernet, TR, ATM, and so forth
- ▶ The HTTP Server
 - ▶ Apache
 - ▶ Netscape
 - ▶ Microsoft
 - ▶ Responds to request

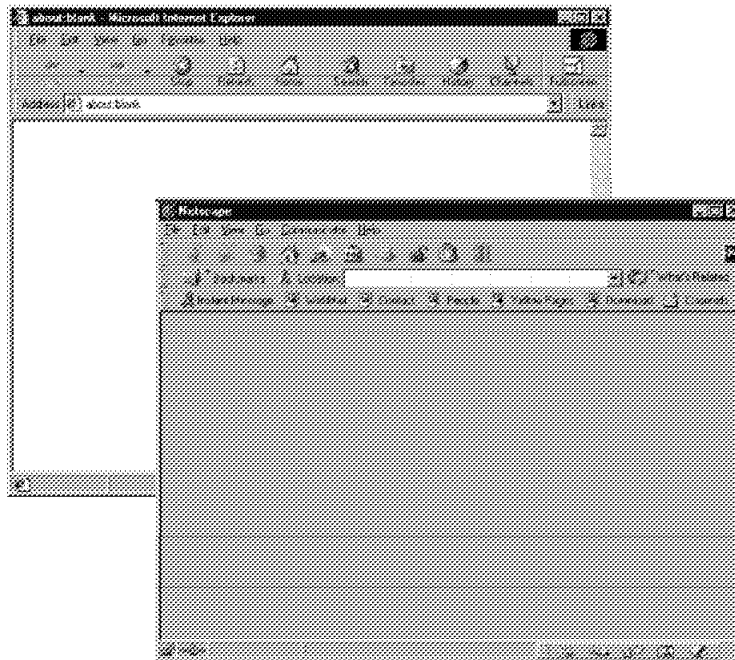
Figure 4-4. The 3 Building Blocks (in212415)

Notes:

The "client" is the program that runs on the user's computer - typically a Web browser. However, it may also be a thin Java client or a "personal" (Tier 0) device, such as a PDA or cell phone. The network is everything required to facilitate communication between the client and the server. The server is typically a Web server.

The Web Browser

- "Universal Client"
- Formats static and dynamic content
- Applies styles
- Executes script
- Exposes object model for document
- Provides Java applet context
- Readily available



- May render tags differently
- Incompatible object models
- Varying CSS support
- JDK level

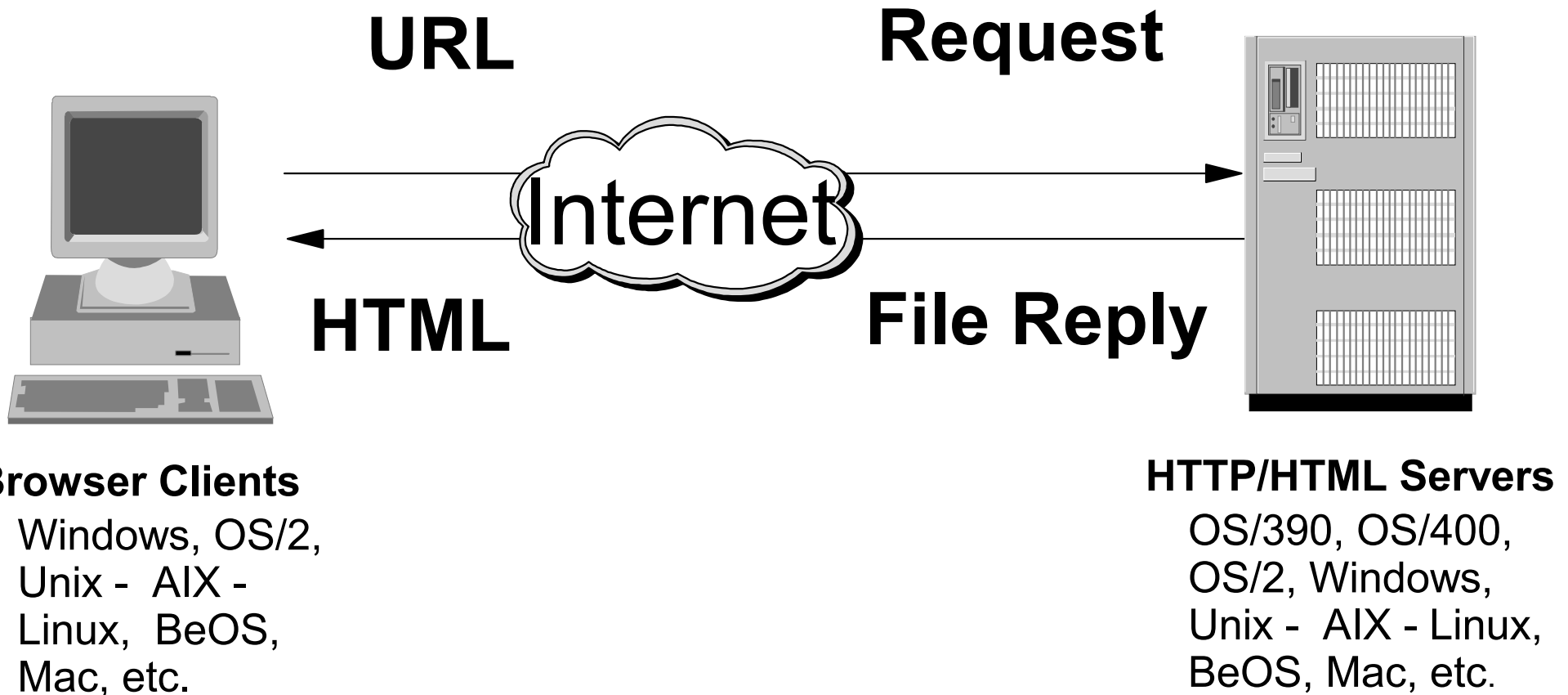
Figure 4-5. The Web Browser (in212420)

Notes:

The Web browser is rapidly becomes the freely and readily available "universal client" in the e-Business world. It is responsible for controlling the session, formatting the data returned by the server, and executing part of the code involved in an end to end application. Be aware of the potential problems listed in the lower 4 bullets!

What Is HTML?

HyperText Markup Language is an authoring language used to create documents for the World Wide Web.



HTML Processing

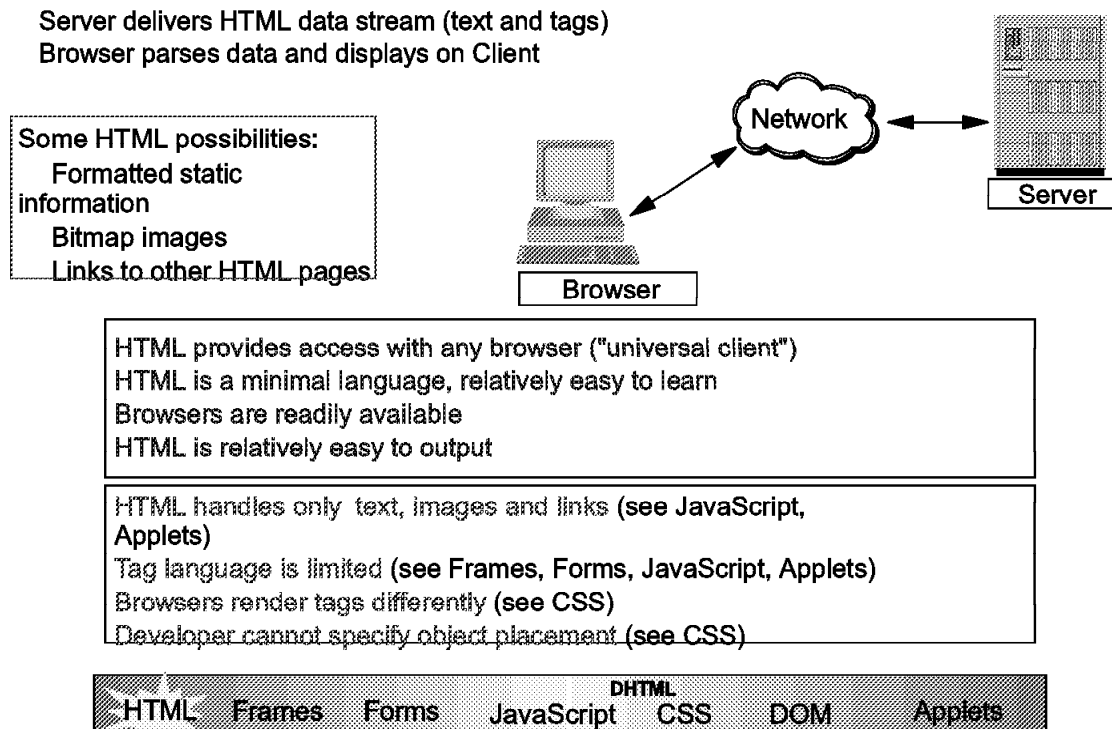


Figure 4-6. HTML Processing (in212425)

Notes:

The original version of HTML could only display simply formatted text, display images, and create links to other HTML pages. While this is very limited, it is also highly portable, and currently is available on virtually any platform. The middle two boxes list the advantages and disadvantages of HTML. In the disadvantage box, items in parentheses indicate technologies that address that particular disadvantage.

Simple HTML

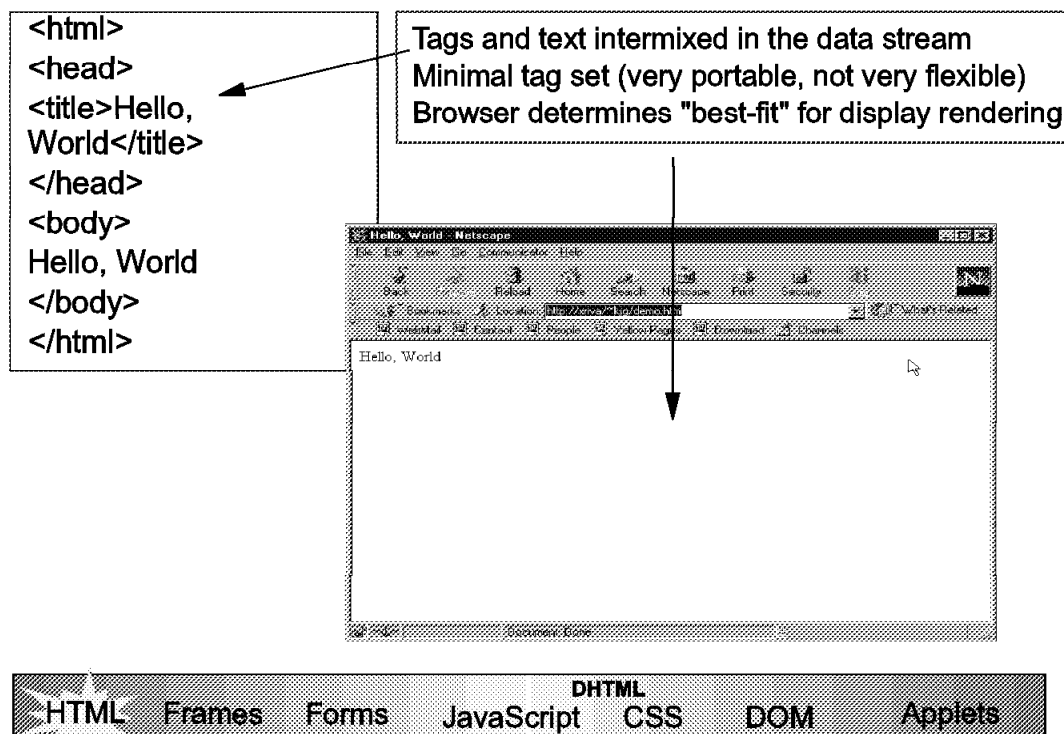


Figure 4-7. Simple HTML (in212430)

Notes:

The format of an HTML tag is a keyword enclosed in angle brackets. Keywords are not case-sensitive. Most tags accept modifiers, or attributes formatted as name/value pairs within the tag. For example, `<BODY BGCOLOR="blue">` is a body tag with an attribute named BGCOLOR set to a value of blue. This particular attribute sets the background color for the entire document to blue.

Some tags stand alone and do not require a closing tag. An example of this is the `
` tag, which inserts a new line into the output text. Most tags affect the text contained between an opening tag and a closing tag. The closing tag is defined as the same keyword as the opening tag preceded with a forward slash character.

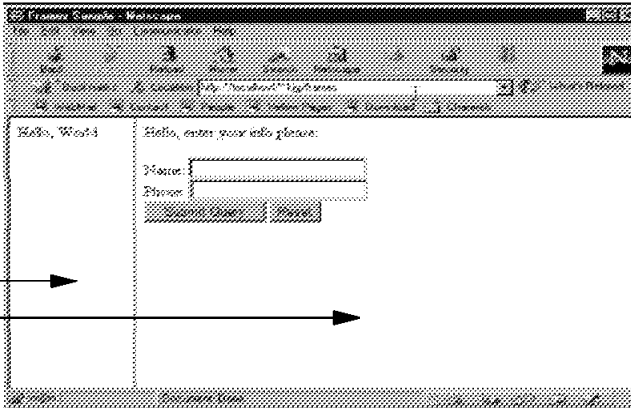
All of the text to be processed by the browser must be contained between the opening and closing HTML tags. Within this block there are two sections - the Head and the Body. The head contains information about the page such as its title, search engine keywords, caching characteristics, etc. The body contains the text to be displayed on the page.

Frames (Positioning Support)

```

<html>
<head>
<title>Frames Sample</title>
</head>
<frameset
cols="20%,80%"
border=2>
<frame src="demo.html">
<frame src="form.html">
</frameset>
</html>

```



Frames provide some placement control
 Frames allow multiple pages on one screen

Frames do not provide object placement (see CSS)
 Older browsers may not support frames

HTML Frames Forms JavaScript DHTML CSS DOM Applets

Figure 4-8. Frames (Positioning Support) (in212435)

Notes:

Frames allow two HTML documents to be display simultaneously within the browser window. The source code shown controls the layout of the frames. The content of each of the frames is specified in the document listed in each <FRAME> tag. The most typical use of a frameset is to create a navigation bar that is displayed in the left frame while specific content is displayed in the right frame.

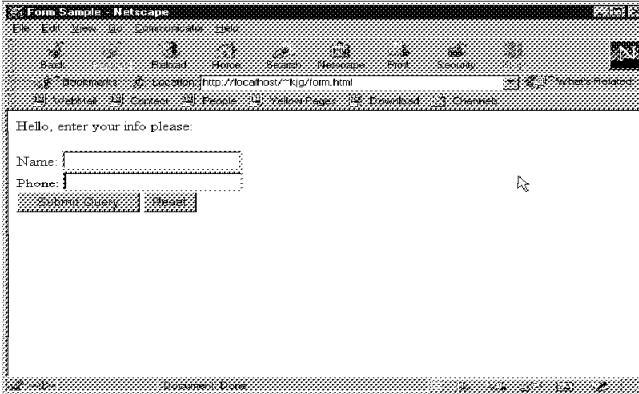
Forms (Processing Support)

```

<html>
<head>
<title>Form Sample</title>
</head>
<body>
Hello, enter your info please:
<form action="process.cgi">
Name:
<input type="text" name="Name">
<br>
Phone:
<input type="text" name="Phone">
<br>
<input type="Submit">
<input type="Reset">
</form>
</body>
</html>

```

Server program



Forms allow data input from HTML pages
Standard GUI controls are supported

HTML still cannot perform client-side processing
(see JavaScript, Applets)

HTML Frames Forms JavaScript DHTML CSS DOM Applets

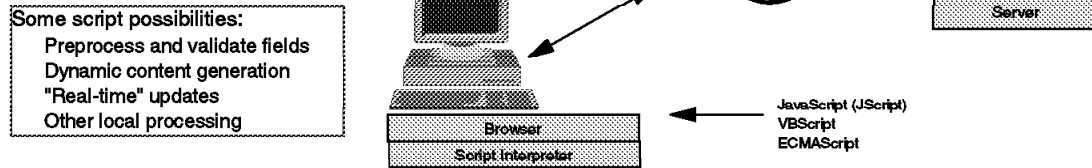
Figure 4-9. Forms (Processing Support) (in212440)

Notes:

Forms allow the user to enter data and submit it to a server for processing. The program that receives the data is specified in the ACTION attribute of the FORM tag. This is the only thing that can be done by the browser using just HTML processing.

JavaScript Processing

Server delivers HTML data stream (text and tags)
 Script is embedded or referenced in the HTML
 Browser parses data and displays on Client
 Browser interprets script



Field pre-processing possible
 Relatively quick downloads, since only text is sent
 Provides access to page objects (see DOM for another solution)

There are multiple scripting language choices and versions (see DOM)
 Older browsers may not support scripting
 User may have scripting disabled
 No APIs available to access back-end systems (see Applets)



Figure 4-10. JavaScript Processing (in212445)

Notes:

Scripting allows access to the elements within the page through the Document Object Model (DOM). This allows great flexibility in creating pages. For instance, a developer might create a navigation item as a listbox of sites, and cause the browser to go to the page selected in the list.

The predominate scripting language available today is JavaScript from Netscape. Microsoft has their own version of JavaScript called JScript and they also support VBScript in Internet Explorer. ECMAScript is the standardization of JavaScript by the European Computer Manufacturers Association.

The script is delivered as text intermixed with the HTML data. This provides relatively quick download. Scripting is designed to work within an HTML page and provides many features designed to enhance HTML processing. The main limitations are the inability to access the operating system or file system, and no ability to access traditional Enterprise assets such as databases.

JavaScript Example

```
<head>
<title>Hello, World</title>
<script>
function printMessage

    document.write("Hello, World");

</script>
</head>
<body>
<script>
printMessage();
</script>
</body>
</html>
```

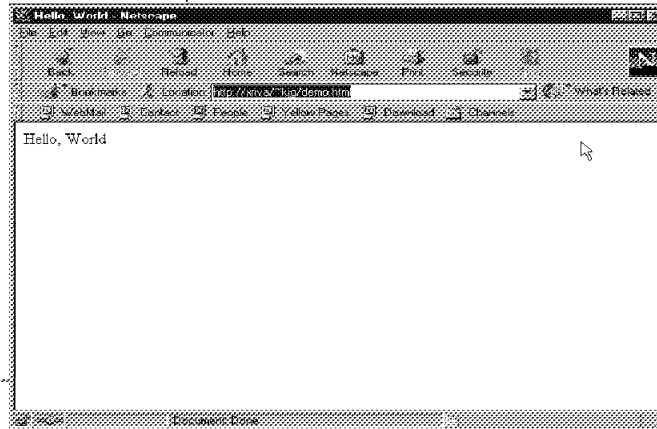


Figure 4-11. JavaScript example (in212450)

Notes:

An HTML page may contain multiple script blocks. The browser parses the text sequentially, so any reference that is executed must have already been parsed at execution time. This typically means that functions are contained within the head of the document and called in the body. The syntax of the language is identical to Java and very similar to C++.

Why Client-Side Processing?

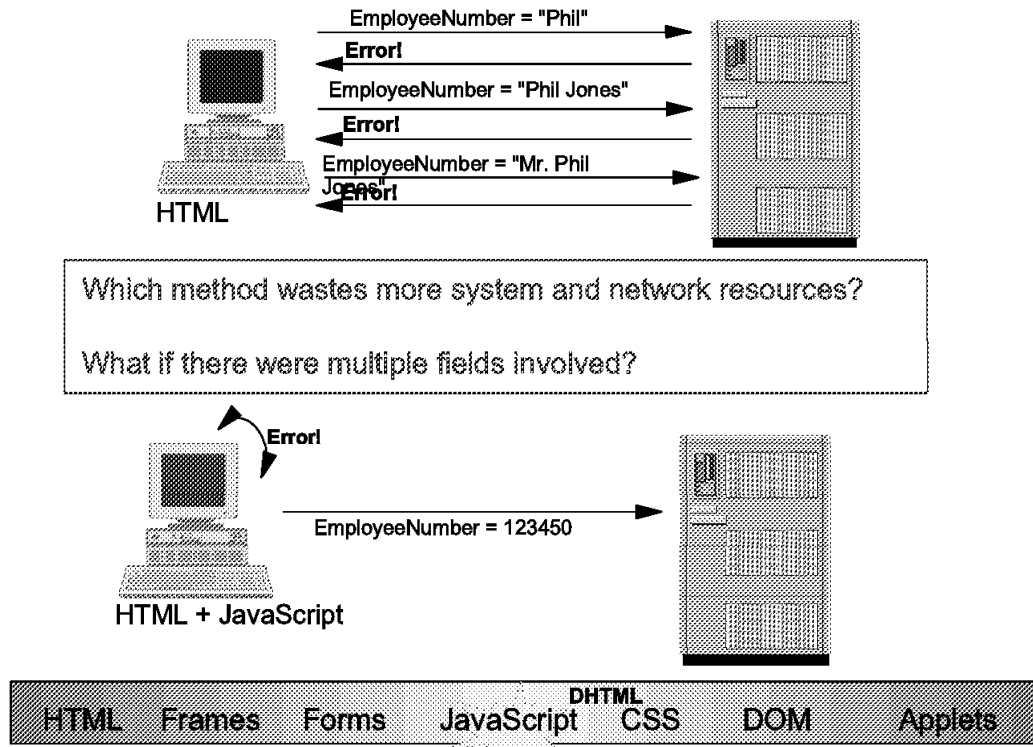


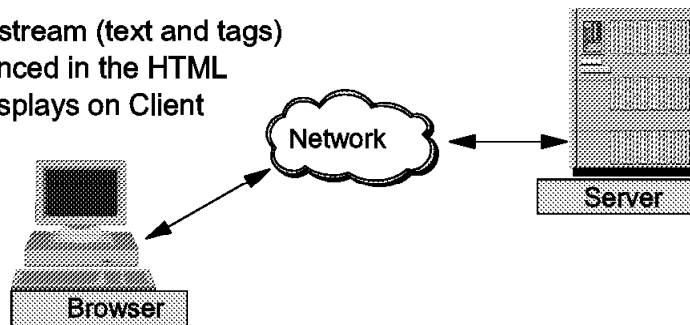
Figure 4-12. Why Client-Side Processing (in212455)

Notes:

If all of the processing occurs on the server, then a connection must be established with the server and the data sent prior to it being validated. This eats up network bandwidth and server resources. By moving the field validation to the client, we can alleviate this problem. JavaScript allows us to do this.

Cascading Style Sheet Processing

Server delivers HTML data stream (text and tags)
 CSS is embedded or referenced in the HTML
 Browser parses data and displays on Client



CSS provides X, Y, Z placement of HTML objects
 Relatively quick downloads, since only text is sent
 Site-wide "look and feel" easily established and maintained

Older browsers may not support CSS
 Browsers may not implement standard correctly

HTML Frames Forms JavaScript ^{DHTML} CSS DOM Applets

Figure 4-13. Cascading Style Sheet Processing (in212460)

Notes:

Cascading Style Sheets are delivered to the browser as simple text just like HTML and JavaScript. Like JavaScript, they can be embedding in an HTML document or external files. They can be used to create a consistent look and feel on all pages in a Web site. CSS provides complete X, Y, and Z axis placement of objects on an HTML page. This allows content to overlap if the design calls for it. Unfortunately, not all browsers support CSS, and even worse, not all browser interpret CSS identically.

CSS Example

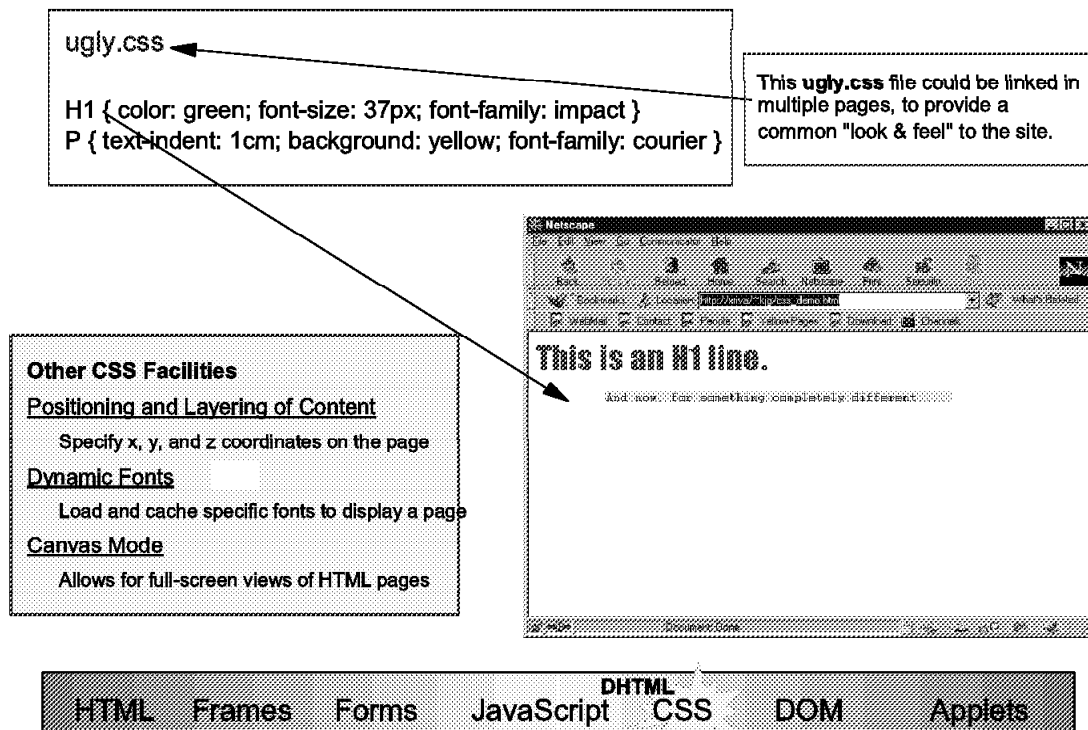


Figure 4-14. CSS Example (in212465)

Notes:

This example changes the look of text modified by HTML formatting tags. The style sheet is contained in a separate file, so multiple HTML pages can use the same set of style modifiers.

Document Object Model

- Browser-independent
- Language-independent
 - JavaScript, Java, Others
- Exposes HTML elements and their properties to programming languages
- Provides
 - A standard set of objects to represent HTML and XML documents
 - A standard model of how these objects can be combined
 - A standard interface for accessing and manipulating them

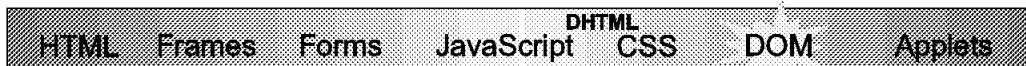
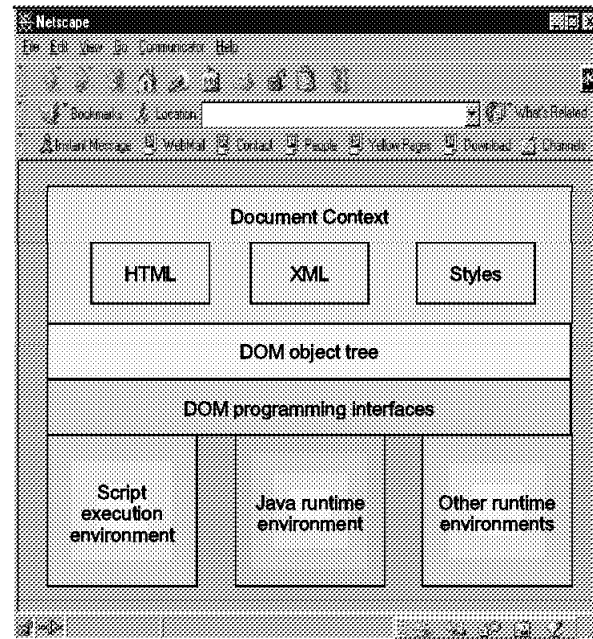


Figure 4-16. Document Object Model (in212475)

Notes:

The Document Object Model is a browser, language, and platform independent means of allowing a programming language to access the elements of an HTML or XML document through a set of standard interfaces. This allows applications to interact with a Web page in a consistent manner similar to the JavaScript Object Hierarchy previously mentioned.

Java Applet Processing

Server delivers HTML data stream (text and tags)

Java applet is referenced in the HTML

Browser parses data and displays on Client

Browser requests applet class file(s)

Browser loads applet into JVM

JVM executes applet code

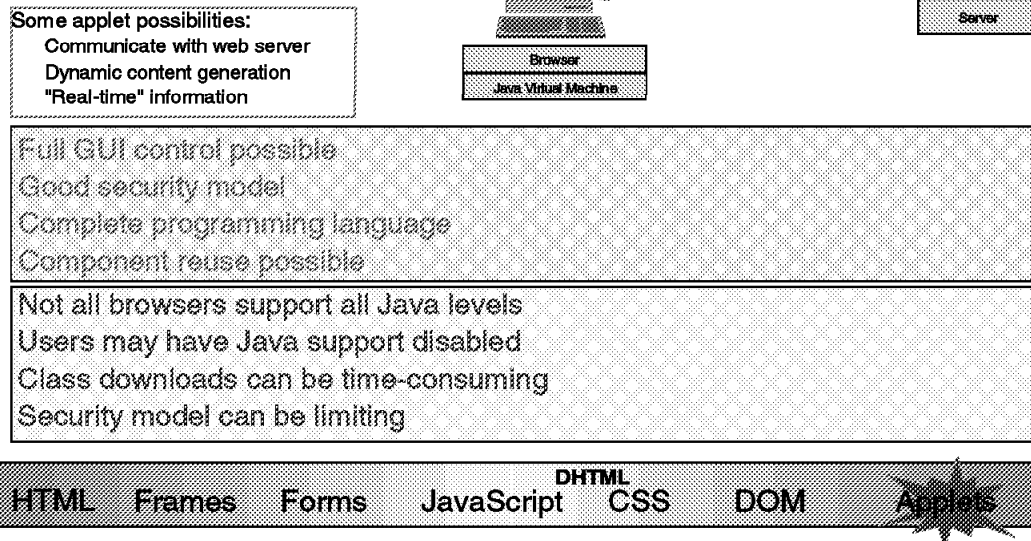


Figure 4-17. Java Applet Processing (in212480)

Notes:

Java applets provide full programming capabilities within a Web page. An applet is referenced via tags in an HTML page which causes the browser to download the applet class file and load it into the JVM. The JVM then executes the applet.

There are limitations in the security model for an applet which can impact application design. The primary restrictions are that the applet cannot access the operating system or file system of the browser platform, and it cannot establish network connection to any other machine on the network other than the machine from which the code was downloaded. This is referred to as the "sandbox".

It is possible to relax these restrictions by placing the applet class files into an archive and signing the archive with a digital certificate. The browser displays a notification to the user when a signed applet is loaded, and acceptance of the certificate is a prerequisite to the applet accessing protected resources.

HTML Evolution

Netscape Navigator



HTML 1.0
HTML 2.0
HTML 3.0

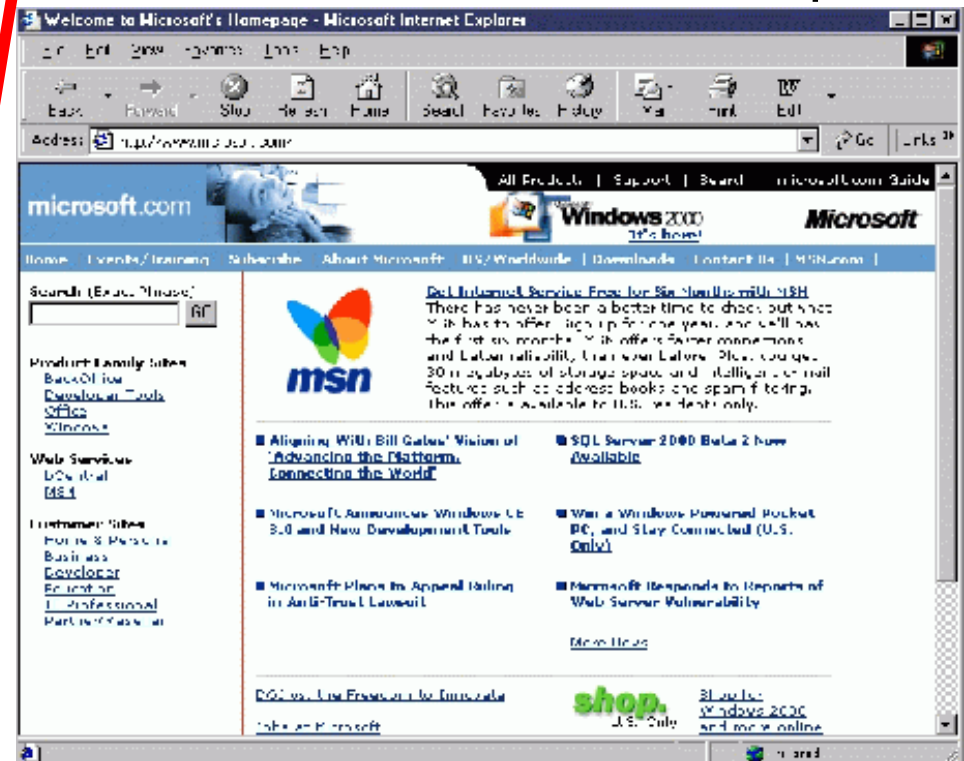
HTML 4.0

HTML 4.01, XHTML 1.0

HTML 3.6

HTML 3.2

Microsoft Internet Explorer



e-business Technologies Overview - Java

Standard Communications

TCP/IP

Standard User Interface

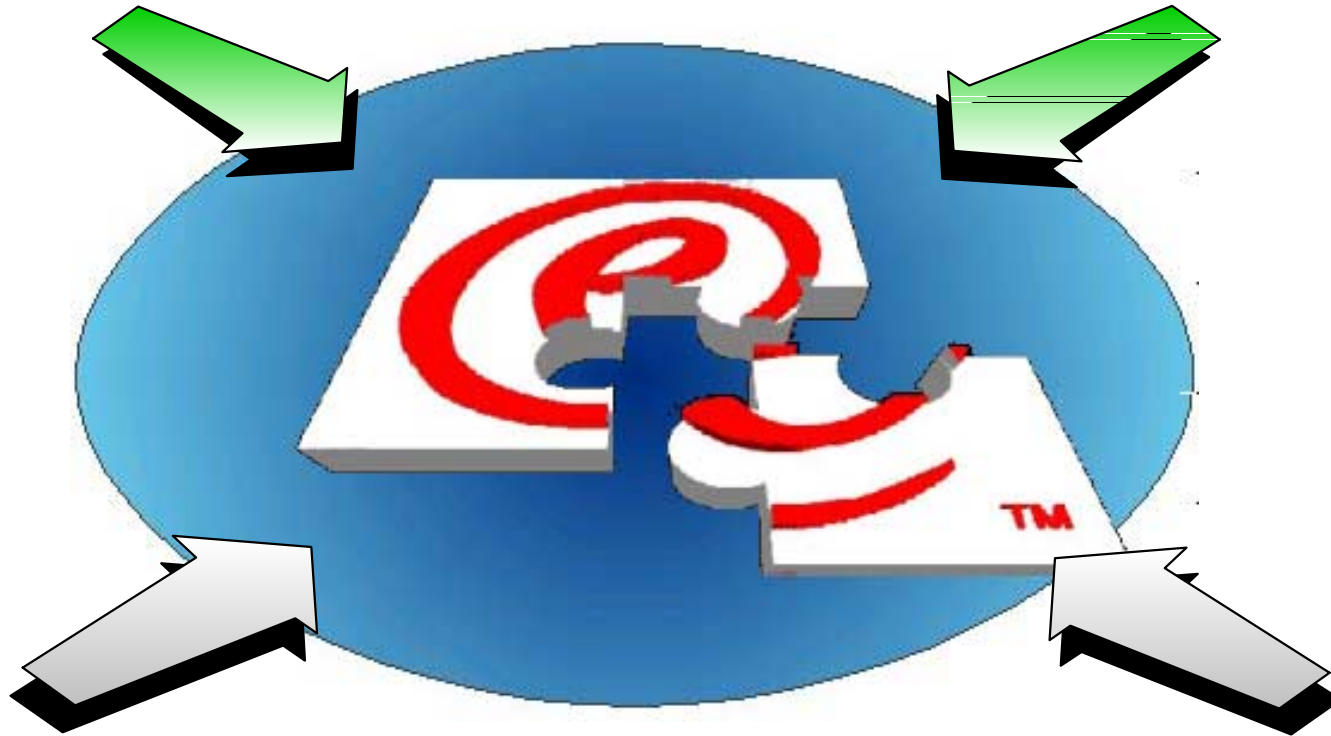
HTML

JAVA

Portable Code

XML

Portable Data



What Is Java?

Java is a object-oriented programming language.

Java is a platform that includes:

- Java Virtual Machine (JVM)
- Application Programming Interface (API)

Common types of Java programs include:

- Applets
- Applications
- Servlets



Why Is Java Important?

Java can write once and run anywhere. It also:

- Is an operating system, and platform-independent
- Has wide industry support

Enables client processing with:

- GUI (through a browser)
- Applets
- Applications

Enables the thin-client and server-side model with:

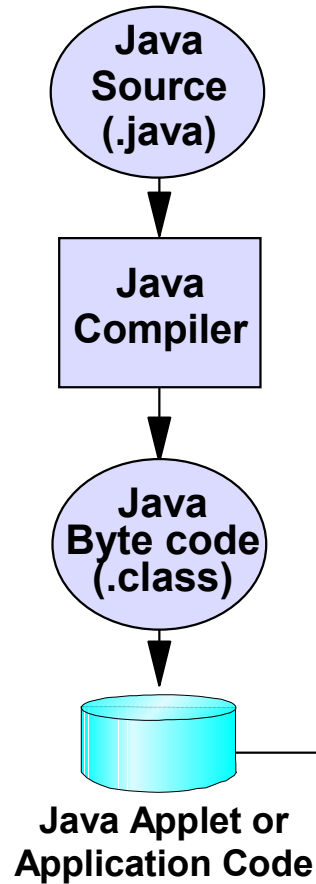
- Servlets
- JSPs
- EJBs



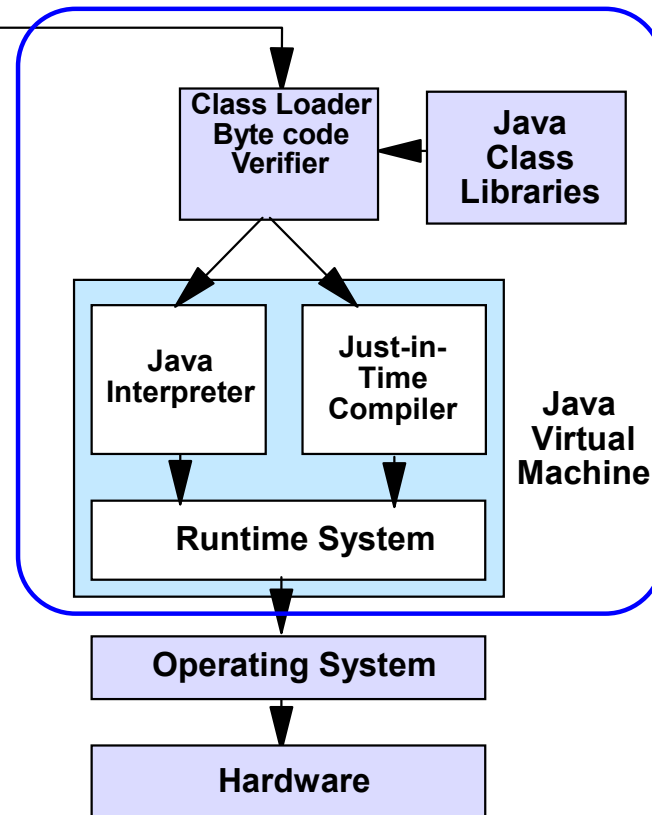
Write Once and Run Anywhere

Java is an interpretive language

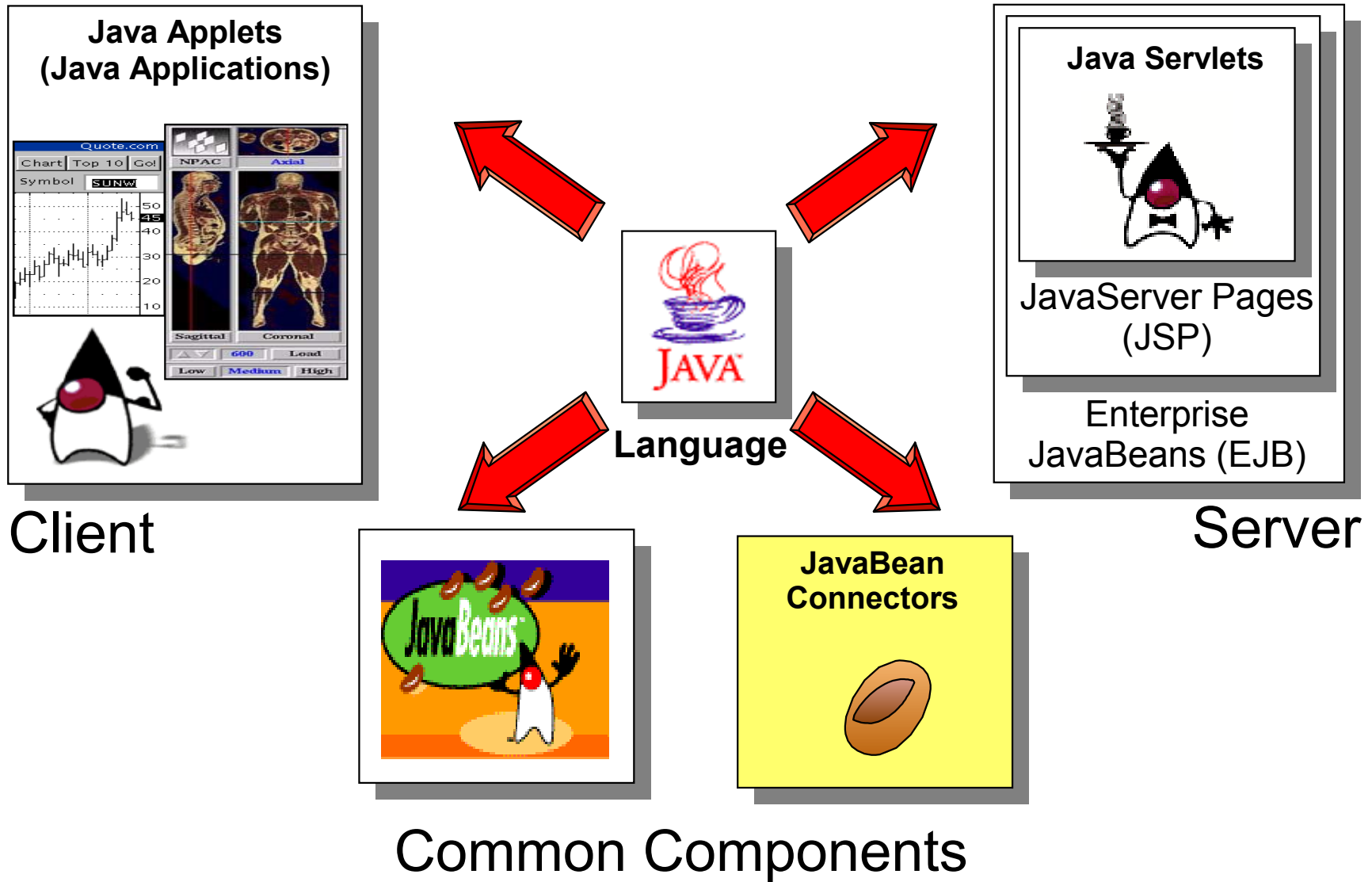
Compile-time Environment



Run-time Environment (Java Platform)



Java Everywhere



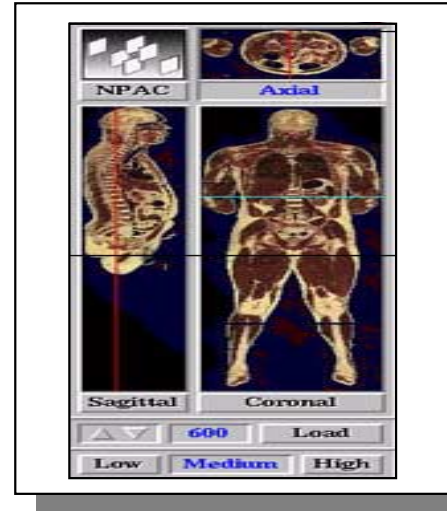
Client-Side Processing with Java Applets

"Eye Candy"



- Animation
- "Tickertape"
- Audio/Video
- Games

Interactive Business Function



- Calculators
- Stock/Portfolio Analysis
- Merchandising
- Medical Analysis

Unit 5. CRM - Data Access

What This Unit Is About

This unit discusses the technologies and products to access data.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Describe CRM with data access
- Describe the role of Application Servers
- Describe Servlets, JSPs, and data access beans
- Describe the Common Connector Framework and its use within VA Java and WebSphere
- Utilize JDBC, SQLJ, connector beans, connection pooling
- Describe IBM WebSphere AS Standard and Advanced editions
- Describe Domino R5 and data access via DECS

How You Will Check Your Progress

Accountability:

- Labs

Objectives

- Describe CRM with data access
- Describe the role of Application Servers
- Describe Servlets, JSPs, and data access beans
- Describe the Common Connector Framework and its use within VA Java and Websphere
- Utilize JDBC, SQLJ, connector beans, connection pooling
- Describe IBM Websphere AS Standard and Advanced editions
- Describe Domino R5 and data access via DECS

Figure 5-1. Unit Objectives (in212500)

Notes:

5.1 The Solution Space

Objectives

- Describe CRM with data access
- Describe the role of Application Servers
- Describe Servlets, JSPs, and Data Access beans
- Describe the Common Connector Framework and its use within VA Java and Websphere
- Utilize JDBC, SQLJ, connector beans, connection pooling
- Describe IBM Websphere AS Standard and Advanced editions
- Describe Domino R5 and data access via DECS

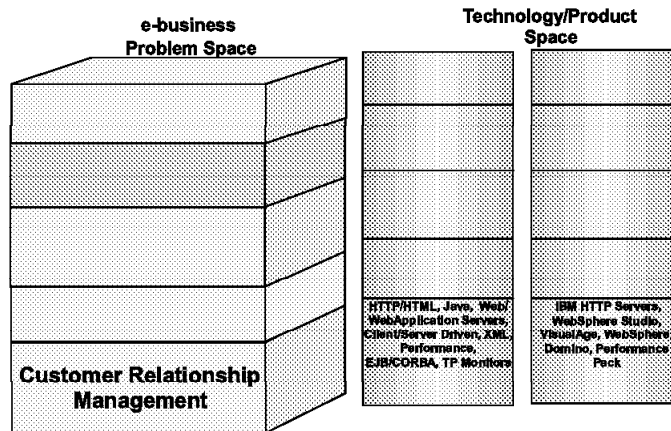
Figure 5-2. Objectives (in212505)

Notes:

Agenda

- The need
 - How do I centralize control of my application on the server yet provide universal access?

- Technology
 - Server-driven applications
 - CGI
 - Server API
 - Java Servlets
 - JSP
 - JavaBeans
 - XML
 - Connectors
 - JDBC
 - SQLJ
 - Connection Pooling
 - Application Servers



- Products
 - WebSphere Studio
 - VisualAge for Java
 - WebSphere Application Server
 - Domino Server



Figure 5-3. Agenda (in212510)

Notes:

We'll be looking at the technologies involved in Web applications that provide dynamic content. These typically involve actual applications working with the HTTP server to access enterprise data sources and generate the content which is forwarded to the client. We will compare the alternative approaches for server-side processing, examine the enabling application technologies and discuss the standards available for gaining database access. We will also examine the role of the application server in data access applications.

Finally, we will discuss the features of WebSphere Studio and VisualAge for Java which enhance the development of data access applications and the services provided by WebSphere Application Server and Domino Server.

The Need

- Customer/Partner self-service
- Server-driven application model
 - Business logic resides on server
 - Ready access to enterprise data sources
- Dynamic content generation
 - Server controls session flow
- Access "live" or "near-live" data

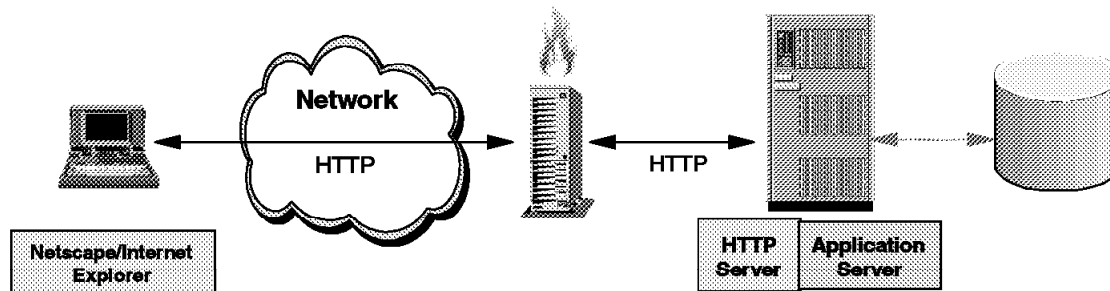
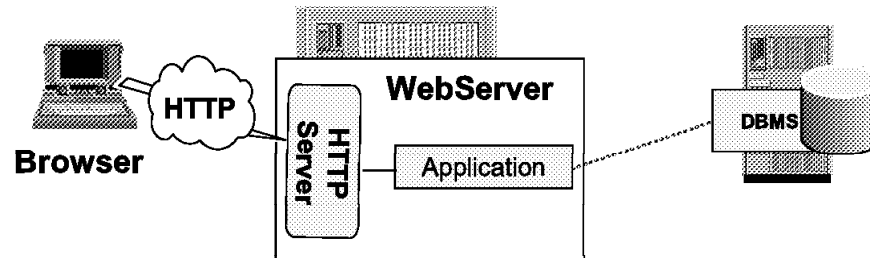


Figure 5-4. The Need (in212515)

Notes:

Customer Relationship Management - data access provides the ability for customers and partners to help themselves to information related to your business activities. It is often referred to as self-service. Typically, this involves providing access via the Internet to application data residing INSIDE your enterprise. For this reason, security can certainly become an issue. However, since the application logic and flow is controlled by the content generated on the server, the client's options for interaction with your data are limited.

Moving From WebServers



WebServer:

- A tightly integrated, single platform client/server model
- Single-platform applications developed and deployed quickly
- Use gateways to connect -- not integrate

Limitations for e-business:

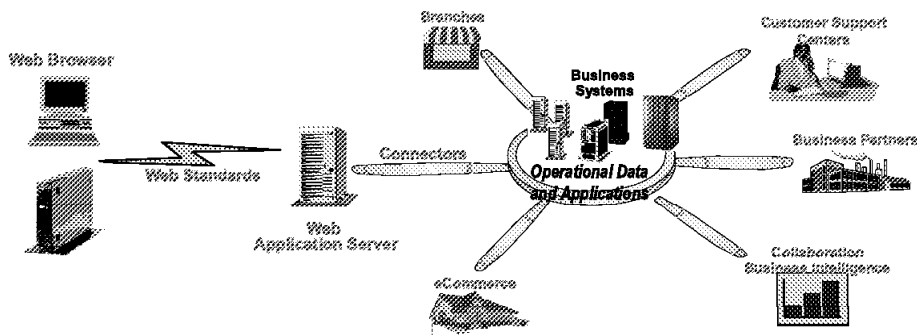
- Does not scale to enterprise level (performance/capacity)
- Limited reuse of components across applications
- Limited integration with existing applications and data
- Difficult to integrate with other enterprises (for example, for supply chain business integration)

Figure 5-5. Moving From WebServers (in212520)

Notes:

An HTTP server is great for serving static pages. However, as you see from the visual, it has some limitations which prevent it from satisfying our requirements without some assistance.

To Web Application Servers



Web Application Servers:

- Standards-based development and deployment approach
- Clients can access data and services anywhere in the system

Advantages:

- Workload can be optimized at any server
- Existing applications and data can be fully integrated
- Meets needs of users, developers and administrators

Services Provided:

- Work Load Management
- Reliability, Availability, Security
- Management and Monitoring
- Location and Service Transparency (Naming Service)
- Resource Management and Pooling (e.g. Database Connections)
- State/Session Management
- Caching of results
- Integration (Connectors) to backend applications/data

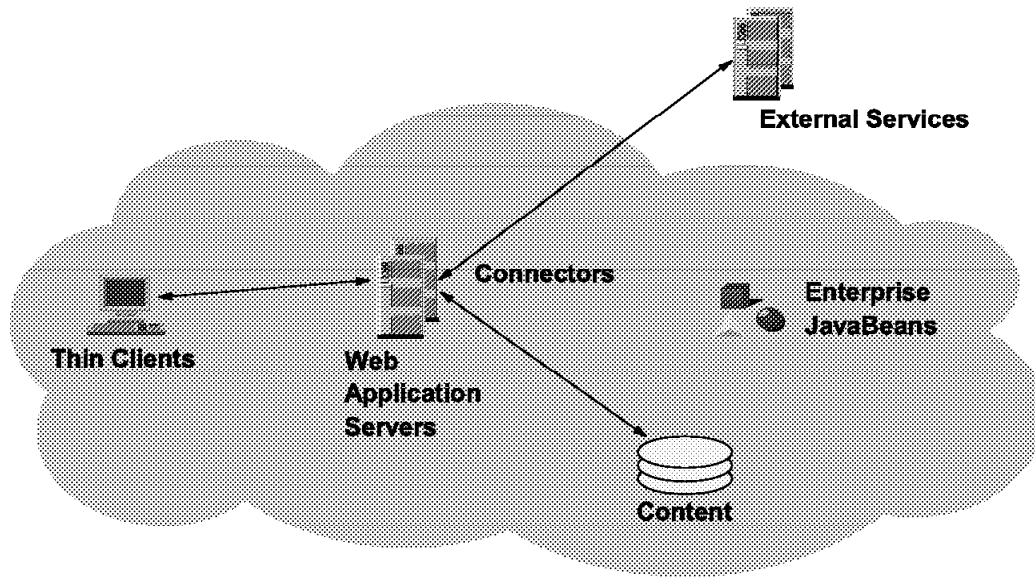
Figure 5-6. To Web Application Servers (in212525)

Notes:

The Web application server operates along with the HTTP server to provide a complete application environment. It is designed for reliability, scalability and security.

5.2 The Technology

Server-driven CRM Applications



The server application:

- Controls application flow
- Accesses enterprise resources
- Generates the UI seen by the client

Figure 5-9. Server-driven CRM Applications (in212540)

Notes:

Data access applications, by their very nature, imply integration with external content and services. They also may make use of existing Java components, enabling the reuse of business logic and data access policies.

Application Implementation Options

Common Gateway Interface	Server API	Java Servlet	Java Server Page
<ul style="list-style-type: none"> ✓ Separate process ✓ Access server resources ✓ Standard across servers ✓ Many source languages supported 	<ul style="list-style-type: none"> ✓ Part of Web server process ✓ Access server resources ✓ Less overhead than CGI 	<ul style="list-style-type: none"> ✓ Can be run in-process or out-of-process ✓ Standard AND portable across servers ✓ Access server resources ✓ Multithreaded 	<ul style="list-style-type: none"> ✓ Can be run in-process or out-of-process ✓ Standard AND portable across servers ✓ Access server resources ✓ Multithreaded ✓ Separates UI and application source
<ul style="list-style-type: none"> ✗ Separate process ✗ One process per invocation ✗ Code not portable ✗ Mixes UI code with application logic 	<ul style="list-style-type: none"> ✗ Error can crash Web server ✗ Interface not portable across web servers ✗ Code not portable ✗ Mixes UI code with application logic 	<ul style="list-style-type: none"> ✗ May have to be loaded upon request ✗ Mixes UI code with application logic 	<ul style="list-style-type: none"> ✗ Compiled upon first request ✗ Source change triggers recompile

Figure 5-10. Application implementation options (in212545)

Notes:

CGI and Server API programs have been around for some time, and can be very useful. However, they have some obvious limitations. CGI programs typically suffer the performance penalty of process activation overhead for each request. Server API programs may perform better, but error could potentially crash the Web server. Servlets and Java Server Pages have the advantage in these areas.

Web applications can rapidly become difficult to maintain. A server program which generates HTML content will typically contain quite a large amount of hard-coded HTML text. While this makes the program very straight-forward to create and test, it places a burden on those responsible for maintaining the code. A simple change to the appearance of the sight or the location of a data source can require modifications to a large number of source files. If correctly used, JSP can provide separation of the user interface definition and the business logic.

What Is a Java Server Page?

- *HTML/XML document containing special JSP tags*
- *Can utilize JavaBeans to access external applications and data sources*
- *If inactive (or not current), page compilation occurs*
 - *JSP Source is parsed*
 - *Java servlet code is generated*
 - *This "JSP Servlet" is compiled, loaded and run*
- *Can be used in conjunction with servlets*

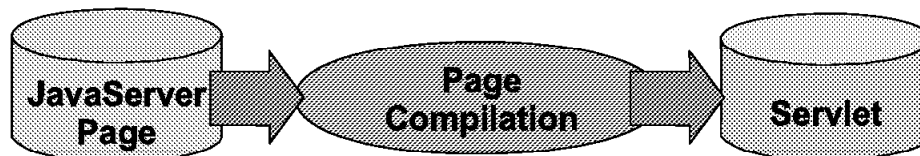


Figure 5-13. What is a Java Server Page? (in212560)

Notes:

JSP offers a variety of tags:

Directives

```
<%@ directive {attribute="value"}* %>
```

1.0 defines page, include, and taglib

Declarations

```
<%! declaration %>
```

jspInit() and jspDestroy() methods may be defined here

Scriptlets

```
<% valid_code_fragment %>
```

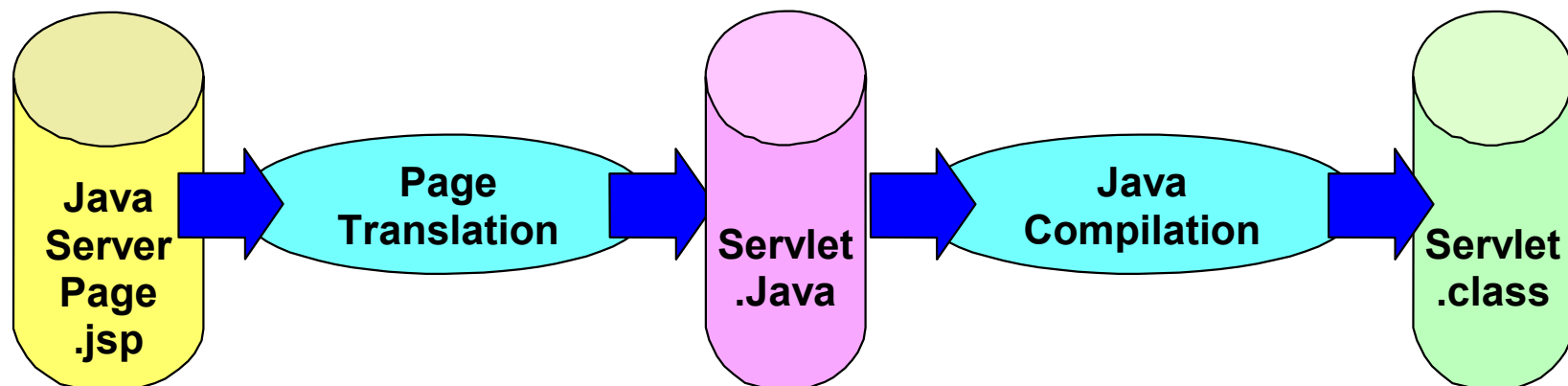
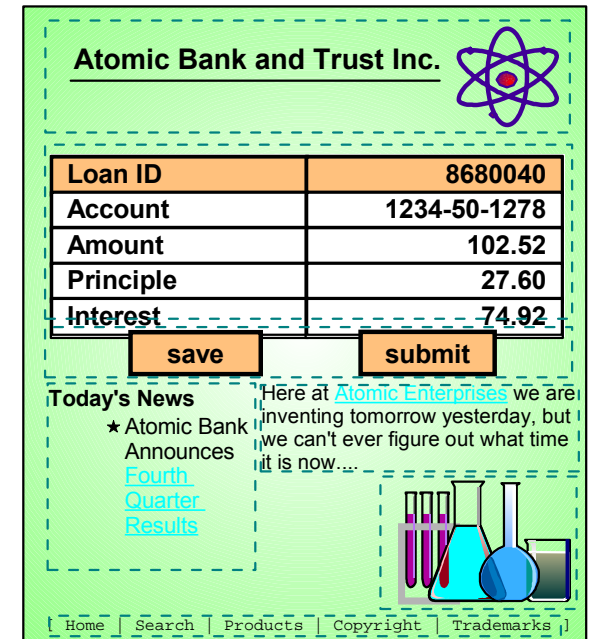
Java code makes up body of generated "method"

Expressions

.

What Is a JavaServer Page?

- HTML document that enables fill-in-the-blanks
- Embedded JSP specific tags
- Inline Java Code
- Can separate presentation from content when used with Java components such as JavaBeans
- Can be built by Web page editor and WebSphere Studio Page Designer
- On the server, JSP Page is parsed and translated into a Java servlet
- Provides extended programming support



<%= expression %>

Semantics:

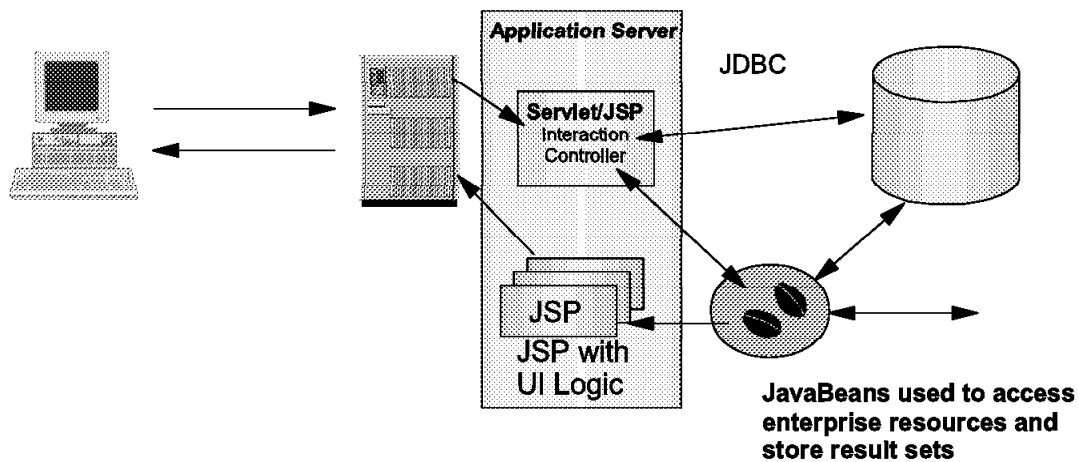
The expression is evaluated

Result is converted to a String and displayed

Here is a sample JSP:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML><HEAD><TITLE>Please confirm your purchase</TITLE>
</HEAD><BODY>
<%@ page errorPage="/pages/jktoys_error_redirector.jsp" %>
<%@ page import="com.jktoys.waslab.common.*" %>
<% // Get customer information from the session
    JKToysCustomer customer = (JKToysCustomer)
        session.getValue(JKToysUtil.CUSTOMER);
    // Get toy order from the session
    JKToyOrder toyOrder = (JKToyOrder)
        session.getValue(JKToysUtil.TOYORDER); %>
<P align="center"><APPLET code="JKToysMastHead.class"
width="650" height="250" archive="JKToysMastHead.jar"
codebase="/JKToysInternet"></APPLET></P>
<H2 align="center"><%= customer.getFirstName() %>, please make
sure these are the toys you picked:</H2><CENTER><BR>
<FORM NAME="JKPurchaseForm"
action="/JKToysInternet/JKApplyCharges" method="POST">
<TABLE border="1"><TR bgcolor="#993366"><FONT color="#FFFFFF">
    <TH>Picture</TH><TH>Name</TH><TH>Description</TH>
    <TH>Price</TH></FONT></TR>
<% // Need to iterate over the items in OrderedToys
    Toy aToy = null;
    Vector orderedToys = toyOrder.getOrderedToys();
    for(int i = orderedToys.size() - 1; i >= 0; i-- ) {
        aToy = (Toy) orderedToys.elementAt(i);      %>
<TR><TD><IMG border="0"
src="<%= "/JKToysInternet" + aToy.getThumbnailFile() %>"></TD>
<TD><%= aToy.getShortDescription() %></TD>
<TD><%= aToy.getFullDescription() %></TD>
<TD><%= aToy.getPriceAsString() %></TD></TR>
<%} /* end for loop */ %> <TR><TD>
<INPUT type="submit" name="ConfirmPurchaseButton"
value="Confirm"></TD>
<TD><INPUT type="submit" name="CancelButton" value="Cancel
Purchase"></TD>
<TD><STRONG>Your total purchase amount is:</STRONG></TD>
<TD><%= toyOrder.getTotalPriceAsString() %></TD></TR></TABLE>
</FORM></CENTER><H2 align="center">Thank you for your interest
in JKToys!!!</H2></BODY></HTML>
```

What Does a JSP Do?



- Enables abstraction of logic
- Positions development effort for EJBs
- Enables graphic artist to focus on UI and Java programmer to focus on JavaBeans, EJBs,...
- Servlets and JSPs can utilize prebuilt components

Figure 5-14. What Does a JSP Do? (in212565)

Notes:

Using JSPs (and possibly servlets) an application can be built which provides abstraction of the user interface, the business logic and the data model. Commonly known as Model/View/Controller architecture, this involves components which accept requests (servlets/JSPs) and route them appropriately, components which access and represent application data (JavaBeans) and components which define and generate the user interface (JSPs). Although a JSP can be used as an interaction controller, the real strength of JSPs is realized when used to define the user interface using JavaBeans as containers for result data.

There are some implicit objects provided by the application server to the JSP at execution time:

```
request -- HttpServletRequest object
response -- HttpServletResponse object
session -- the current HttpSession*
out -- the JspWriter which writes into the output stream
pageContext, application (ServletContext), config (ServletConfig), page exception --
Instance of Throwable (available to Error Page)
```

What Is JDBC ?

- A Java API for executing SQL statements
 - A trademark (often thought of as an acronym standing for "Java DataBase Connectivity")
- The Java way of accessing databases
- Allows Java Applications, Applets, and Servlets to:
 - Establish a connection with a database
 - Send dynamic SQL statements
 - No Analysis of SQL is done until execution time
 - Process the results returned from the SQL

Figure 5-17. What is JDBC (in212580)

Notes:

The Common Connector Framework

Before CCF

- Connectors all look different, but are semantically very similar
- Additional learning is required for a developer to work with a new connector
- A developer has to deal with the pure aspect of driving an interaction as well as with infrastructure services like security, transactions, and so forth
- For a component environment enabling a connector means to adapt its infrastructure services like security, transactions, ... to the respective services of the component environment
- Developing higher-level tools ends up to be a never ending story, you have to deal with any new connector that comes along

Common Connector Framework (CCF)

- A common client programming model for connectors
- A common infrastructure programming model for connectors, that gives a component environment a standard view for a connector and vice-versa

Figure 5-28. The Common Connector Framework (in2125d5)

Notes:

The CCF is a common approach for providing connectors which integrate external systems with Java applications.