

IT230 WEB TECHNOLOGIES

FINAL STUDY GUIDE

WE WILL COVER ALL THE CHAPTERS

Before Midterm

- HTTP
- XHTML
- CSS
- JavaScript
- DOM

After Midterm

- PHP 5 introduction
- PHP 5 OOP
- PHP database & errors
- Servlets
- XML
- JSP
- Database & JSP

QUESTIONS TYPES:

These are just examples of the major questions types:

- ✓ MCQs,
- ✓ True/False,
- ✓ Essay
- ✓ Short Answer,
- ✓ What is the output of the program or the function,
- ✓ Write a code.

Question Type	Count	Marks
MCQs	20	10
T/F	20	10
Essay	2	8
Short Answer	3	6
Find Output ¹	3	6
Coding ²	2	10
	Total	50

¹ What is the output questions will be on (JavaScript, Servlets and PHP)

² Write a code question will be on (HTML, PHP)

TOPICS

Week	
2	TCP,IP,DNS,UDP, HTTP request / response (URI, MIME), client caching
3	HTML tags: div, p, span, table, ol, ul, img, a, form, input, ... (coding)
4	Different methods of embedding CSS, Selector Strings (id, class, ...), CSS Box Model
5	Window.alert, prompt / Functions, variables, if, loop, operators, Array, Build-in objects
6	Events: onclick, onmouseover, onmouseout, onfocus, onblur / Event Propagation / Modifying Element Style/ Document tree : Node / Document tree : document
8	Managing Variables/ Data types/SuperGlobals/ Arrays/Operators/Control structure (coding)
9	Design patterns (Strategy pattern, singleton pattern, factory pattern, observer pattern) (concept) Read user input (coding) TECHNIQUES TO MAKE SCRIPTS "SAFE" (concept) Cookies & sessions(coding)
10	Mysql Buffered Versus Unbuffered Queries (concept) Connect to mysql (coding) Error versus Warning (concept)
11	Server side programming/Servlets vs Applications/Servlet Life Cycle (concept)/ Sessions/Cookies
12	XML syntax/ XML Document (how to write XML document) / Java based DOM (concept)
13	Overview of JSP page components /JSP & servlet (the concept) Web Applications (Concept)/JSP Expression Language MVC (Concept)
14	JDBC Seven Steps for Database Access(Concept) Prepared Statements (Concept) Callable Statements (Concept) Database Transactions (Concept)



week 8 [coding]

IT230

Dr Mohamed Habib

fppt.com



PHP 5 Basic Language

- Managing Variables

```
if (isset($first_name)) {  
    print '$first_name is set';  
}  
  
$name = "John Doe";  
unset($name);  
if (isset($name)) {  
    print '$name is set';  
}
```

fppt.com



PHP 5 Basic Language

- Managing Variables

```
if (empty($name)) {  
    print 'Error: Forgot to specify a value for $name';  
}
```

fppt.com



PHP 5 Basic Language

- SuperGlobals

- ☛ `$_GET[]`. An array that includes all the `GET` variables that PHP received from the client browser.
- ☛ `$_POST[]`. An array that includes all the `POST` variables that PHP received from the client browser.
- ☛ `$_COOKIE[]`. An array that includes all the cookies that PHP received from the client browser.
- ☛ `$_ENV[]`. An array with the environment variables.
- ☛ `$_SERVER[]`. An array with the values of the web-server variables.

fppt.com



Example

```
<html>  
<body>
```

HTML page

```
<form action="welcome.php" method="post">  
Name: <input type="text" name="name"><br>  
E-mail: <input type="text" name="email"><br>  
<input type="submit">  
</form>
```

```
</body>  
</html>
```

Name:
E-mail:

fppt.com

PHP page [welcom.php]

```
<html>  
<body>
```

```
Welcome <?php echo $_POST["name"]; ?><br>  
Your email address is: <?php echo $_POST["email"]; ?>
```

```
</body>  
</html>
```

Name:
E-mail:

Result:

```
Welcome SEU  
Your email address is: contact@seu.edu.sa
```

page1 in HTML form

```
<form action="third.php" method="get">
  <!-- Choices -->
  Red    <input type="checkbox" name="color[]" id="color" value="Red">
  Green  <input type="checkbox" name="color[]" id="color" value="Green">
  Blue   <input type="checkbox" name="color[]" id="color" value="Blue">
  Cyan   <input type="checkbox" name="color[]" id="color" value="Cyan">
  Magenta <input type="checkbox" name="color[]" id="color" value="Magenta">
  Yellow <input type="checkbox" name="color[]" id="color" value="Yellow">
  Black  <input type="checkbox" name="color[]" id="color" value="Black">
  <!-- Submit -->
  <input type="submit" value="submit">
</form>
```

```
<?php

$name = $_GET['color'];

if(isset($_GET['color'])) {

echo "You chose the following color(s): <br>";

foreach ($name as $color){
echo $color."<br />";
}

} // end brace for if(isset

else {

echo "You did not choose a color.";

}

?>
```

third.php



```
<?php  
  
$name = $_GET['color'];  
  
if(isset($_GET['color'])) {  
  
    echo "You chose the following color(s): <br>";  
    echo "<ul>";  
  
    foreach ($name as $color){  
  
        echo "<li>" . $color . "</li>";  
  
    }  
  
    echo "</ul>";  
  
} // isset  
  
else {  
  
    echo "You did not choose a color.";  
  
}
```

another
code
third.php



PHP 5 Basic Language

- Basic Data Types
 - Integer
 - Floating Point
 - Strings
 - Booleans
 - Null



PHP 5 Basic Language

- Arrays

- ❏ `array(1, 2, 3)` is the same as the more explicit `array(0 => 1, 1 => 2, 2 => 3)`.
- ❏ `array("name" => "John", "age" => 28)`
- ❏ `array(1 => "ONE", "TWO", "THREE")` is equivalent to `array(1 => "ONE", 2 => "TWO", 3 => "THREE")`.
- ❏ `array()` an empty array.

Here's an example of a nested `array()` statement:

```
array(array("name" => "John", "age" => 28), array("name" =>
↳ "Barbara", "age" => 67))
```

fppt.com

```
$arr1 = array(1, 2, 3);
$arr2[0] = 1;
$arr2[1] = 2;
$arr2[2] = 3;
```

```
print_r($arr1);
print_r($arr2);
```

```
$arr1 = array("name" => "John", "age" => 28);
$arr2["name"] = "John";
$arr2["age"] = 28;
```

```
if ($arr1 == $arr2) {
    print '$arr1 and $arr2 are the same' . "\n";
}
```




PHP 5 Basic Language

- Reading array values

```
print $arr2["name"];  
if ($arr2["age"] < 35) {  
    print " is quite young\n";  
}
```

fppt.com



PHP 5 Basic Language

- Accessing Nested Arrays

```
$arr = array(1 => array("name" => "John", "age" => 28), array("name"  
=> "Barbara", "age" => 67))
```

You could achieve the same result with the following statements:

```
$arr[1]["name"] = "John";  
$arr[1]["age"] = 28;  
$arr[2]["name"] = "Barbara";  
$arr[2]["age"] = 67;
```

fppt.com



PHP 5 Basic Language

- **Traversing Arrays Using foreach**

Here's a short example of the `foreach()` loop:

```
$players = array("John", "Barbara", "Bill", "Nancy");

print  "The players are:\n";
foreach ($players as $key => $value) {
    print "#$key = $value\n";
}
```

The output of this example is

```
The players are:
#0 = John
#1 = Barbara
#2 = Bill
#3 = Nancy
```



PHP 5 Basic Language

- **Traversing Arrays Using list() and each()**

```
$players = array("John", "Barbara", "Bill", "Nancy");

reset($players);

while (list($key, $val) = each($players)) {
    print "#$key = $val\n";
}
```

The output of this example is

```
#0 = John
#1 = Barbara
#2 = Bill
#3 = Nancy
```



PHP 5 Basic Language

- Operators
 - Numerical Operators
 - Assignment Operators
 - Comparison Operators
 - Logical Operators
 - Bitwise Operators
 - Negation Operators
 - Increment/Decrement Operators
 - The Cast Operators
 - The Silence Operator

fppt.com



PHP 5 Basic Language

- CONTROL STRUCTURES

Statement

```
if (expr)
    statement
elseif (expr)
    statement
elseif (expr)
    statement
...
else
    statement
```

Statement List

```
if (expr):
    statement list
elseif (expr):
    statement list
elseif (expr):
    statement list
...
else:
    statement list
endif;
```

fppt.com



PHP 5 Basic Language

- CONTROL STRUCTURES

Statement

```
switch (expr){  
    case expr:  
        statement list  
    case expr:  
        statement list  
    ...  
    default:  
        statement list  
}
```

Statement List

```
switch (expr):  
    case expr:  
        statement list  
    case expr:  
        statement list  
    ...  
    default:  
        statement list  
endswitch;
```

fppt.com



PHP 5 Basic Language

- Loop Control Structures

```
$result = 1;  
while ($n > 0) {  
    $result *= $n--;  
}  
print "The result is $result";
```

```
break;  
break expr;  
continue;  
continue expr;
```

fppt.com



PHP 5 Basic Language

- **Loop Control Structures**

Here's an example:

```
for ($i = 0; $i < 10; $i++) {  
    print "The square of $i is " . $i*$i . "\n";  
}
```

The result of running this code is

```
The square of 0 is 0  
The square of 1 is 1  
...  
The square of 9 is 81
```

fppt.com



PHP 5 Basic Language

- **Code Inclusion Control Structures**

```
include file_name;
```

```
include $_SERVER["DOCUMENT_ROOT"] . "/myscript.php";
```

fppt.com



week9

IT230
Dr Mohamed Habib

fppt.com



Advanced OOP and Design Patterns

- **DESIGN PATTERNS**
- When designing software, certain programming patterns repeat themselves. Some of these have been addressed by the software design community and have been given accepted general solutions. These repeating problems are called **design patterns**.

fppt.com

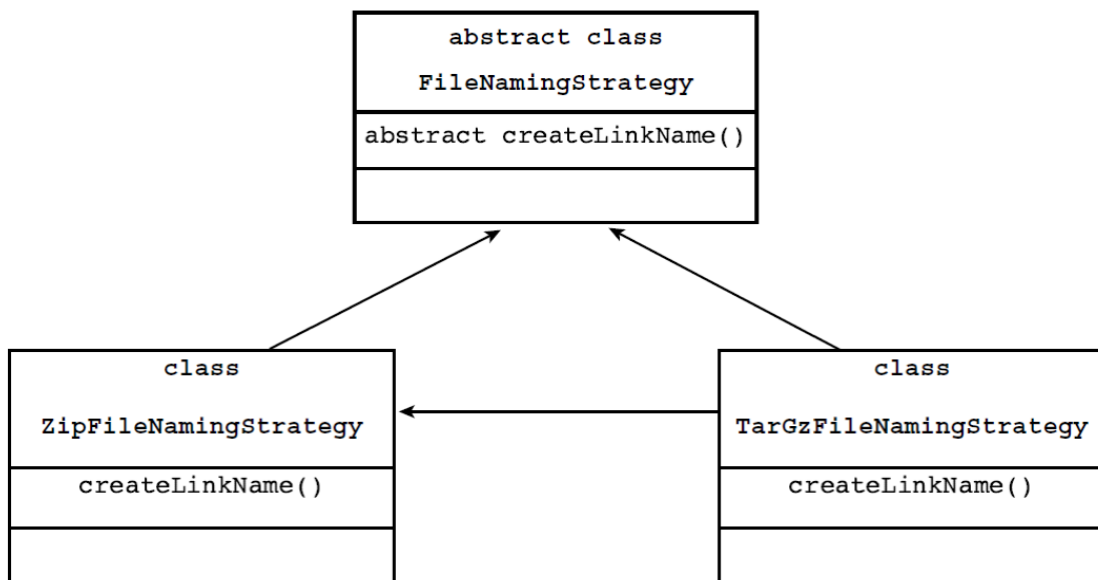
Advanced OOP and Design Patterns

- **Strategy Pattern**
- The **strategy pattern** is typically used when your programmer's algorithm should be interchangeable with different variations of the algorithm. For example, if you have code that creates an image, under certain circumstances, you might want to create JPEGs and under other circumstances, you might want to create GIF files.
- The strategy pattern is usually implemented by declaring an abstract base class with an algorithm method, which is then implemented by inheriting concrete classes. At some point in the code, it is decided what concrete strategy is relevant; it would then be instantiated and used wherever relevant.

fppt.com

Advanced OOP and Design Patterns

- **Strategy Pattern**



fppt.com



Advanced OOP and Design Patterns

- **Singleton Pattern**
- The **singleton pattern** is probably one of the best-known design patterns. You have probably encountered many situations where you have an object that handles some centralized operation in your application, such as a logger object. In such cases, it is usually preferred for only one such application-wide instance to exist and for all application code to have the ability to access it.
- Specifically, in a logger object, you would want every place in the application that wants to print something to the log to have access to it, and let the centralized logging mechanism handle the filtering of log messages according to log level settings. For this kind of situation, the singleton pattern exists.


fppt.com



Advanced OOP and Design Patterns

- **Factory Pattern**
- Polymorphism and the use of base class is really the center of OOP. However, at some stage, a concrete instance of the base class's subclasses must be created. This is usually done using the **factory pattern**. A Factory class has a static method that receives some input and, according to that input, it decides what class instance to create (usually a subclass).

fppt.com



Advanced OOP and Design Patterns

- **Factory Pattern**

- Say that on your web site, different kinds of users can log in. Some are guests, some are regular customers, and others are administrators. In a common scenario, you would have a base class User and have three subclasses: GuestUser, CustomerUser, and AdminUser. Likely User and its subclasses would contain methods to retrieve information about the user (for example, permissions on what they can access on the web site and their personal preferences).
- The best way for you to write your web application is to use the base class User as much as possible, so that the code would be generic and that it would be easy to add additional kinds of users when the need arises.

fppt.com



Advanced OOP and Design Patterns

- **Observer Pattern**

- PHP applications, usually manipulate data. In many cases, changes to one piece of data can affect many different parts of your application's code. For example, the price of each product item displayed on an e-commerce site in the customer's local currency is affected by the current exchange rate. Now, assume that each product item is represented by a PHP object that most likely originates from a database; the exchange rate itself is most probably being taken from a different source and is not part of the item's database entry. Let's also assume that each such object has a display() method that outputs the HTML relevant to this product.
- The **observer pattern** allows for objects to register on certain events
- and/or data, and when such an event or change in data occurs, it is automatically
- notified. In this way, you could develop the product item to be an observer
- on the currency exchange rate, and before printing out the list of items, you
- could trigger an event that updates all the registered objects with the correct
- rate. Doing so gives the objects a chance to update themselves and take the
- new data into account in their display() method

fppt.com



Advanced OOP and Design Patterns

- **Observer Pattern**
- The **observer pattern** allows for objects to register on certain events and/or data, and when such an event or change in data occurs, it is automatically notified. In this way, you could develop the product item to be an observer on the currency exchange rate, and before printing out the list of items, you could trigger an event that updates all the registered objects with the correct rate. Doing so gives the objects a chance to update themselves and take the new data into account in their display() method

fppt.com



How to Write a Web Application with PHP

- **USER INPUT**

coding

Registration

E-mail address:

First name:

Last name:

Password:

fppt.com



How to Write a Web Application with PHP

- **USER INPUT**

coding

```
<html>
<head><title>Register</title></head>
<body>
  <h1>Registration</h1>
  <form method="get" action="register.php">
    <table>
      <tr><td>E-mail address:</td>
        <td><input type='text' name='email' /></td></tr>
      <tr><td>First name:</td>
        <td><input type='text' name='first_name' /></td></tr>
      <tr><td>Last name:</td>
        <td><input type='text' name='last_name' /></td></tr>
      <tr><td>Password:</td>
        <td><input type='password' name='password' /></td></tr>
      <tr>
        <td colspan='2'>
          <input type='submit' name='register' value='Register' />
        </td>
      </tr>
    </table>
  </form>
</body>
</html>
```

fppt.com



How to Write a Web Application with PHP

- **USER INPUT**

coding

```
<?php
    if (!isset ($_POST['register']) || ($_POST['register'] !=
        ↪'Register')) {
?>

<?php
    } else {
?>
E-mail: <?php echo $_POST['email']; ?><br />
Name: <?php echo $_POST['first_name']. ' '. $_POST['last_name'];
    ↪?><br />
Password: <?php echo $_POST['password']; ?><br />
<?php
    }
?>
```

fppt.com



How to Write a Web Application with PHP

- **TECHNIQUES TO MAKE SCRIPTS “SAFE”**
- **Input Validation**

concept

- For different kinds of input, you can use different methods. For instance, if you expect a parameter passed with the HTTP GET method to be an integer, force it to be an integer in your script:

```
<?php
$product_id = (int) $_GET['prod_id'];
?>
```

fppt.com



How to Write a Web Application with PHP

- **TECHNIQUES TO MAKE SCRIPTS “SAFE”**
- **Input Validation**

concept

- Everything other than an integer value is converted to 0. But, what if `$_GET['prod_id']` doesn't exist? You will receive a notice because we turned the `error_level` setting up. A better way to validate the input would be

```
<?php
if (!isset($_GET['prod_id'])) {
    die ("Error, product ID was not set");
}
$product_id = (int) $_GET['prod_id'];
?>
```

fppt.com



How to Write a Web Application with PHP

- **TECHNIQUES TO MAKE SCRIPTS “SAFE”** concept
- **HMAC Verification**
- If you need to prevent bad guys from tampering with variables passed in the URL (such as for a redirect as shown previously, or for links that pass special parameters to the linked script), you can use a hash, as shown in the following script:



How to Write a Web Application with PHP

- **TECHNIQUES TO MAKE SCRIPTS “SAFE”** concept
- **Input Filter**

The `filter_input()` function gets an external variable (e.g. from form input) and optionally filters it.

This function is used to validate variables from insecure sources, such as user input.

Example

Check if the external variable "email" is sent to the PHP page, through the "get" method, and also check if it is a valid email address:

```
<?php
if (!filter_input(INPUT_GET, "email", FILTER_VALIDATE_EMAIL)) {
    echo("Email is not valid");
} else {
    echo("Email is valid");
}
?>
```



How to Write a Web Application with PHP

- TECHNIQUES TO MAKE SCRIPTS “SAFE”
- Working with Passwords

concept

Another appliance of hash functions is authenticating a password entered in a form on your web site with a password stored in your database. For obvious reasons, you don't want to store unencrypted passwords in your database. You want to prevent evil hackers who have access to your database (because the sysadmin blundered) from stealing passwords used by your clients. Because hash functions are not at all reversible, you can store the password hashed with a function like `md5()` or `sha1()` so the evil hackers can't get the password in plain text.

fppt.com



How to Write a Web Application with PHP

- COOKIES

coding

What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the `setcookie()` function.

Syntax

`setcookie(name, value, expire, path, domain, secure, httponly);`

Only the *name* parameter is required. All other parameters are optional.

fppt.com



The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable `$_COOKIE`). We also use the `isset()` function to find out if the cookie is set:

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() +
(86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is
not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!
<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

coding

fppt.com



coding

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the `setcookie()` function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30),
"/");
?>
<html>
<body>
```

fppt.com



coding

Delete a Cookie

To delete a cookie, use the `setcookie()` function with an expiration date in the past:

Example

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>

<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

fppt.com



How to Write a Web Application with PHP

- COOKIES

coding

```
<?php
    ob_start();
?>
<html>
<head><title>Login</title></head>
<body>
<?php
    if (isset ($_POST['login']) && ($_POST['login'] == 'Log in') &&
        ($uid = check_auth($_POST['email'], $_POST['password'])))
    {
        /* User successfully logged in, setting cookie */
        setcookie('uid', $uid, time() + 14400, '/');
        header('Location: http://kossu/crap/0x-examples/index.php');
        exit();
    } else {
?>
```

fppt.com



How to Write a Web Application with PHP

- **SESSIONS**

coding

What is a PHP Session?

When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.

Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser. So; Session variables hold information about one single user, and are available to all pages in one application.



How to Write a Web Application with PHP

- **SESSIONS**

A session is started with the **session_start()** function.

Session variables are set with the PHP global variable: **\$_SESSION**.

coding

```
<?php
// Start the session
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```



How to Write a Web Application with PHP

- **SESSIONS**

Get PHP Session Variable Values

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

coding

```
<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . "<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

fppt.com



Modify a PHP Session Variable

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>
```

coding

```
<?php
// to change a session variable, just overwrite
it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>

</body>
</html>
```

fppt.com



How to Write a Web Application with PHP

- **SESSIONS**

Destroy a PHP Session

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

coding



week 10

IT230
Dr Mohamed Habib

fppt.com



Objectives

- **Chapter 6: Databases with PHP 5**
Chapter 5: Error Handling

fppt.com

Databases with PHP 5

• Connections

coding

Function Name	Description
<code>mysqli_connect(...)</code> <code>\$mysqli = new mysqli(...)</code>	Opens a connection to the MySQL server. Parameters (all are optional) <ul style="list-style-type: none">• host name (string)• user name (string)• password (string)• database name (string)• TCP port (integer)• UNIX domain socket (string)
<code>mysqli_init()</code> <code>\$mysqli = new mysqli</code> <code>mysqli_options(...)</code> <code>\$mysqli->options(...)</code> <code>mysqli_real_connect(...)</code> <code>\$mysqli->real_connect(...)</code>	Initializes MySQLi and returns an object for use with <code>mysqli_real_connect</code> Set various connection options Opens a connection to the MySQL server
<code>mysqli_close(...)</code> <code>\$mysqli->close()</code>	Closes a MySQL server connection The parameter is connection object (function only)
<code>mysqli_connect_errno()</code>	Obtains the error code of the last failed connect
<code>mysqli_connect_error()</code>	Obtains the error message of the last failed connect
<code>mysqli_get_host_info(...)</code> <code>\$mysqli->host_info</code>	Returns a string telling what the connection is connected to

fppt.com

Example

coding

```
<?php

$conn = mysqli_connect("localhost", "test", "", "world");
if (empty($conn)) {
    die("mysqli_connect failed: " . mysqli_connect_error());
}

print "connected to " . mysqli_get_host_info($conn) . "\n";
mysqli_close($conn);
```

Here, the `mysqli_connect()` function connects to "localhost" with the user name "test", an empty password, and selects the "world" database as the default database. If the connect fails, `mysqli_connect()` returns `FALSE`, and `mysqli_connect_error()` returns a message saying why it could not connect.

.com



Example

When using the object-oriented interface, you can also specify your connection parameters by passing them to the constructor of the `mysqli` object:

```
<?php

$mysqli = new mysqli("localhost", "test", "", "world");
if (mysqli_connect_errno) {
die("mysqli_connect failed: " . mysqli_connect_error());
}
print "connected to " . $mysqli->host_info . "\n";
$mysqli->close();
```

fppt.com



Databases with PHP 5

- **Buffered Versus Unbuffered Queries**
- **Buffered queries** will retrieve the query results and store them in memory on the client side, and subsequent calls to get rows will simply spool through local memory.
- Buffered queries have the advantage that you can seek in them, which means that you can move the “current row” pointer around in the result set freely because it is all in the client. Their disadvantage is that extra memory is required to store the result set, which could be very large, and that the PHP function used to run the query does not return until all the results have been retrieved.

fppt.com



Databases with PHP 5

- **Buffered Versus Unbuffered Queries**
- **Unbuffered queries**, on the other hand, limit you to a strict sequential access of the results but do not require any extra memory for storing the entire result set. You can start fetching and processing or displaying rows as soon as the MySQL server starts returning them. When using an unbuffered result set, you have to retrieve all rows with `mysqli_fetch_row` or close the result set with `mysqli_free_result` before sending any other command to the server.

fppt.com



Error Handling

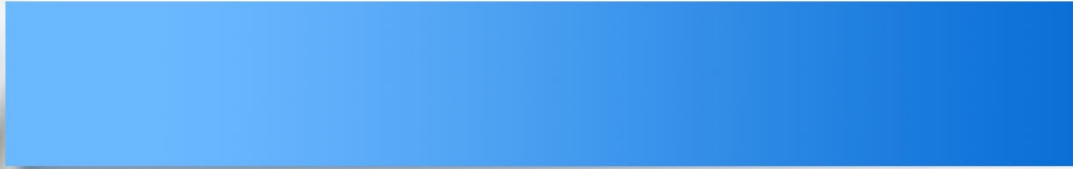
• TYPES OF ERRORS

– Runtime Errors

–PHP Errors

- **E_ERROR**: This is a fatal, unrecoverable error. Examples are out-of-memory errors, uncaught exceptions, or class redeclarations
- **E_WARNING**: This is the most common type of error. It normally signals that something you tried doing went wrong. Typical examples are missing function parameters, a database you could not connect to, or division by zero.

fppt.com



The `mysqli_real_connect()` function opens a new connection to the MySQL server.

The `mysqli_real_connect()` function differs from [mysqli_connect\(\)](#) in the following ways:

- `mysqli_real_connect()` requires a valid object created by [mysqli_init\(\)](#)
- `mysqli_real_connect()` can be used with `mysqli_options()` to set different options for the connection
- `mysqli_real_connect()` has a flag parameter

Server-side Programming

- The combination of
 - HTML
 - JavaScript
 - DOMis sometimes referred to as **Dynamic HTML** (DHTML)
- Web pages that include scripting are often called **dynamic** pages (vs. **static**)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Server-side Programming

- Similarly, web server response can be static or dynamic
 - **Static**: HTML document is retrieved from the file system and returned to the client
 - **Dynamic**: HTML document is generated by a program in response to an HTTP request
- Java servlets are one technology for producing dynamic server responses
 - **Servlet** is a class instantiated by the server to produce a dynamic response

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Servlets vs. Java Applications

- Servlets **do not have a main()**
 - The main() is in the server
 - Entry point to servlet code is via call to a method (doGet() in the example)
- Servlet **interaction with end user is indirect** via request/response object APIs
 - Actual HTTP request/response processing is handled by the server
- Primary servlet **output is typically HTML**

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Servlet Life Cycle

- Servlet API life cycle methods
 - **init()**: called when servlet is instantiated; must return before any other methods will be called
 - **service()**: method called directly by server when an HTTP request is received; default service() method calls doGet() (or related methods covered later)
 - **destroy()**: called when server shuts down

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Servlet Life Cycle

```
public void init()  
    throws ServletException  
{  
    try {  
        BufferedReader br =  
            new BufferedReader(new FileReader("aFile"));  
        visits = (new Integer(br.readLine())).intValue();  
    }  
    catch (FileNotFoundException fnfe) {  
        throw new UnavailableException("File not found: " +  
            fnfe.toString());  
    }  
    catch (Exception e) {  
        throw new UnavailableException("Data problem: " +  
            e.toString());  
    }  
}
```

Example life cycle method:
attempt to initialize visits variable
from file

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Servlet Life Cycle

```
public void init()  
    throws ServletException  
{  
    try {  
        BufferedReader br =  
            new BufferedReader(new FileReader("aFile"));  
        visits = (new Integer(br.readLine())).intValue();  
    }  
    catch (FileNotFoundException fnfe) {  
        throw new UnavailableException("File not found: " +  
            fnfe.toString());  
    }  
    catch (Exception e) {  
        throw new UnavailableException("Data problem: " +  
            e.toString());  
    }  
}
```

Exception to be thrown
if initialization fails and servlet
should not be instantiated

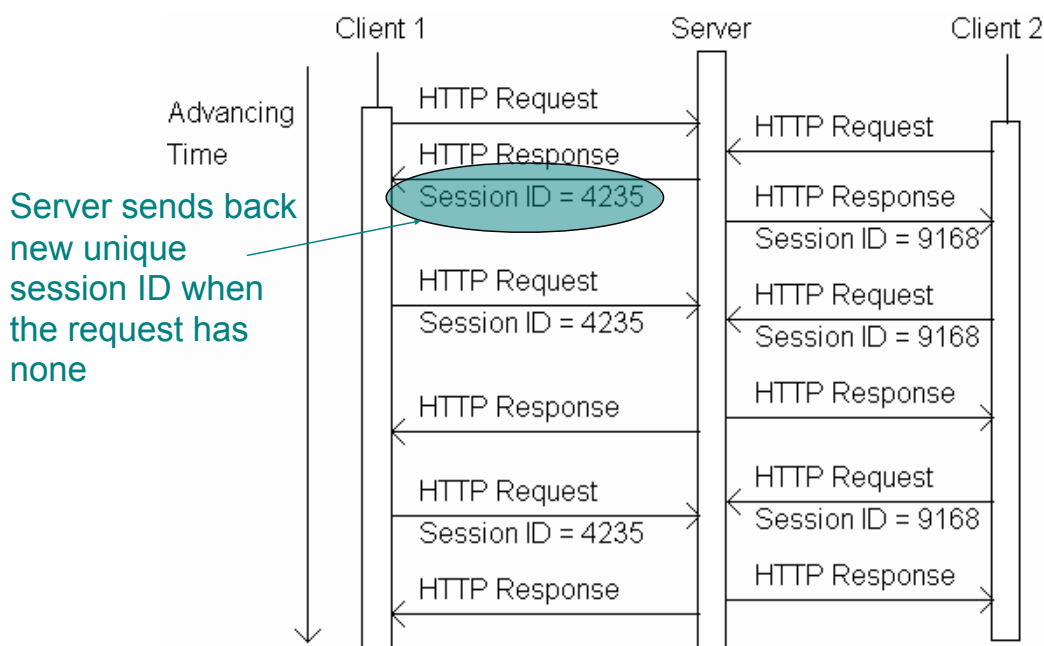
Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

- Many interactive Web sites spread user data entry out over **several pages**:
 - Ex: add items to cart, enter shipping information, enter billing information
- Problem: how does the server know which users generated which HTTP requests?
 - Cannot rely on standard HTTP headers to identify a user

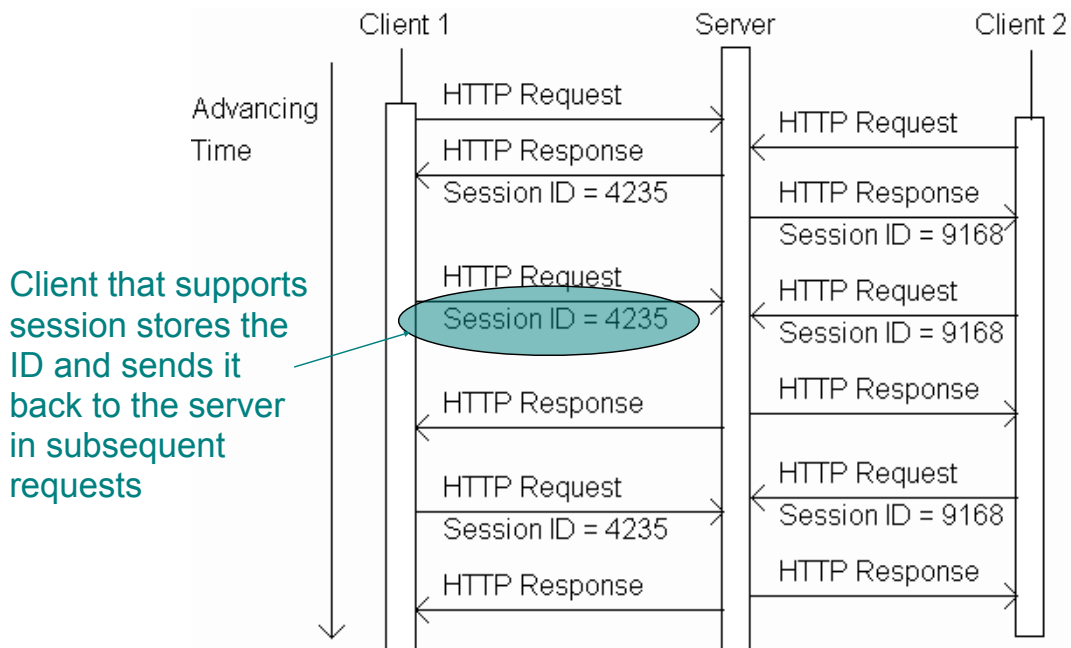
Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions



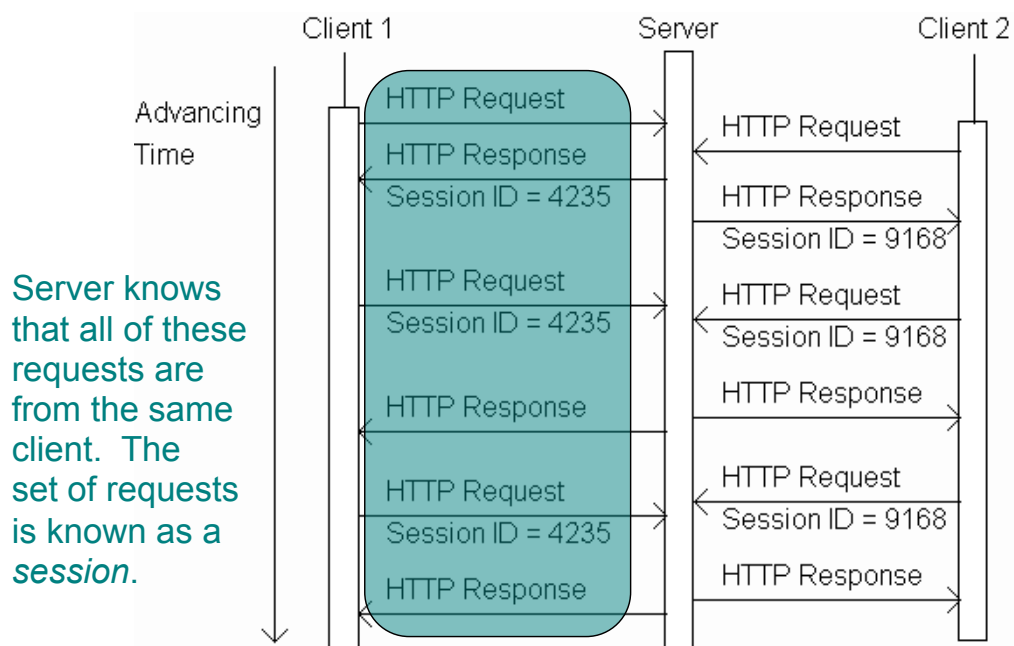
Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions



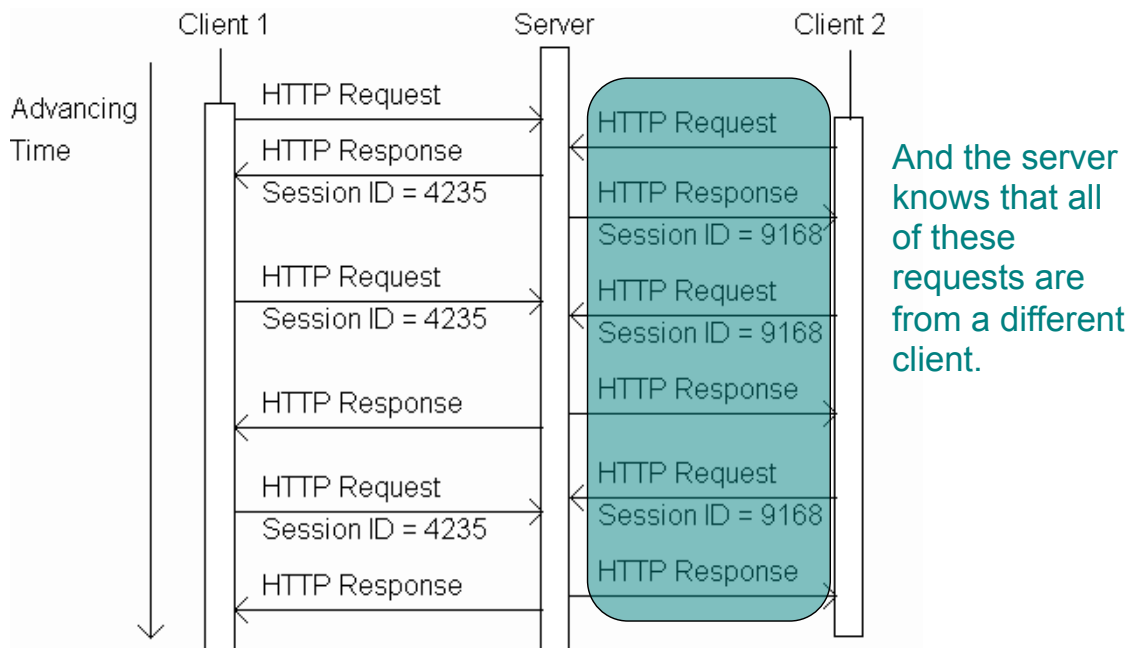
Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
HttpSession session = request.getSession();  
if (session.isNew()) {  
    visits++;  
}
```

Returns HttpSession object associated with this HTTP request.

- Creates new HttpSession object if no session ID in request or no object with this ID exists
- Otherwise, returns previously created object

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
HttpSession session = request.getSession();  
if (session.isNew()) {  
    visits++;  
}
```

Boolean indicating whether returned object was newly created or already existed.

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

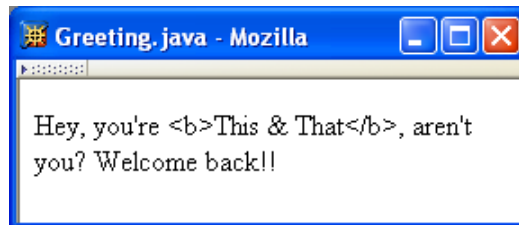
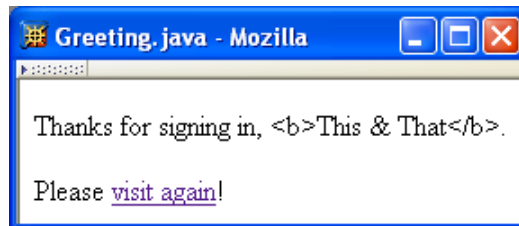
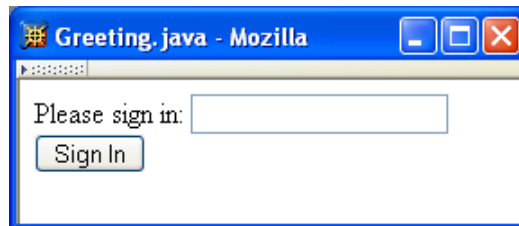
```
HttpSession session = request.getSession();  
if (session.isNew()) {  
    visits++;  
}
```

Incremented once per session

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

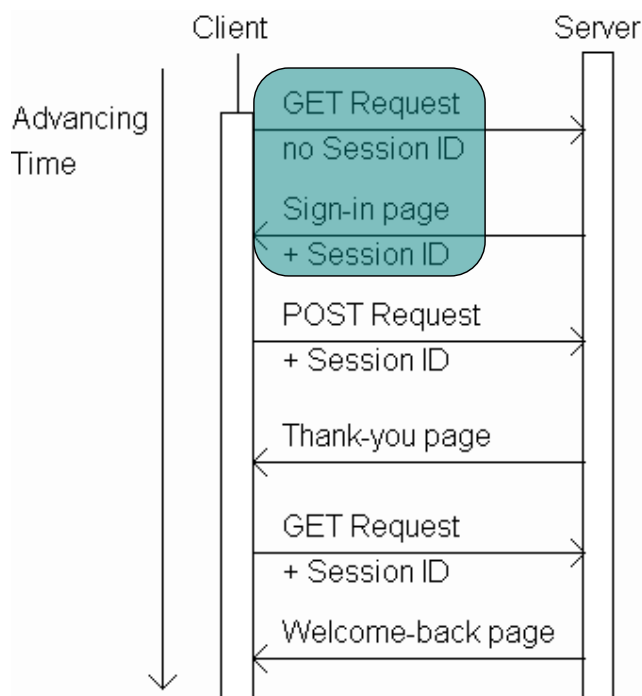
Sessions

Three web pages produced by a single servlet



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
public void doGet (HttpServletRequest request,
                  HttpServletResponse response)
{
    HttpSession session = request.getSession();
    String signIn = (String)session.getAttribute("signIn");
    if (session.isNew() || (signIn == null)) {
        printSignInForm(servletOut, "Greeting");
    } else {
        printWelcomeBack(servletOut, signIn);
    }
}
```

Session attribute is a name/value pair

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
public void doGet (HttpServletRequest request,
                  HttpServletResponse response)
{
    HttpSession session = request.getSession();
    String signIn = (String)session.getAttribute("signIn");
    if (session.isNew() || (signIn == null)) {
        printSignInForm(servletOut, "Greeting");
    } else {
        printWelcomeBack(servletOut, signIn);
    }
}
```

Session attribute will have null value until a value is assigned

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

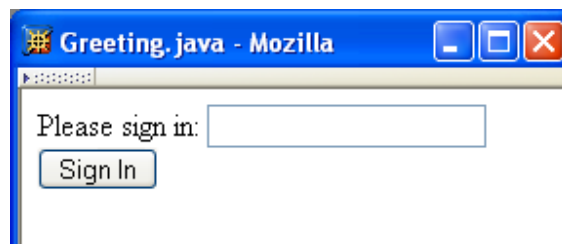
```
public void doGet (HttpServletRequest request,
                  HttpServletResponse response)
{
    HttpSession session = request.getSession();
    String signIn = (String)session.getAttribute("signIn");
    if (session.isNew() || (signIn == null)) {
        printSignInForm(servletOut, "Greeting");
    } else {
        printWelcomeBack(servletOut, signIn);
    }
}
```

Generate sign-in form if session is new or signIn attribute has no value, welcome-back page otherwise.

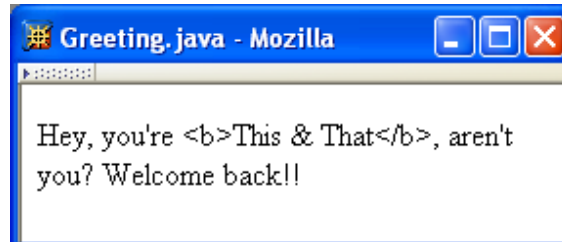
Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

Sign-in form



Welcome-back page



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
private void printSignInForm(PrintWriter servletOut,  
                             String action)  
{  
    ...  
    servletOut.println(  
" <form method='post' action='" + action + "'><div> \n" +  
" <label> \n" +  
"     Please sign in: <input type='text' name='signIn' /> \n" +  
    ...  
    ...
```

Second argument
("Greeting") used as
action attribute value
(relative URL)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
private void printSignInForm(PrintWriter servletOut,  
                             String action)  
{  
    ...  
    servletOut.println(  
" <form method='post' action='" + action + "'><div> \n" +  
" <label> \n" +  
"     Please sign in: <input type='text' name='signIn' /> \n" +  
    ...  
    ...
```

Form will be sent using POST HTTP
method (doPost() method will be called)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
private void printSignInForm(PrintWriter servletOut,
                             String action)
{
    ...
    servletOut.println(
"    <form method='post' action='" + action + "'><div> \n" +
"    <label> \n" +
"    Please sign in: <input type='text' name='signIn' /> \n" +
    ...

```

Text field containing
user name is named
signIn

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
public void doPost (HttpServletRequest request,
                   HttpServletResponse response)
{
    ...
    String signIn = request.getParameter("signIn");
    HttpSession session = request.getSession();
    if (signIn != null) {
        printThanks(servletOut, signIn, "Greeting");
        session.setAttribute("signIn", signIn);
    } else {
        printSignInForm(servletOut, "Greeting");
    }
}

```

Retrieve
signIn
parameter value

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
public void doPost (HttpServletRequest request,
                   HttpServletResponse response)
    ...
    String signIn = request.getParameter("signIn");
    HttpSession session = request.getSession();
    if (signIn != null) {
        printThanks(servletOut, signIn, "Greeting");
        session.setAttribute("signIn", signIn);
    } else {
        printSignInForm(servletOut, "Greeting");
    }
}
```

Normal
processing:
signIn
parameter
is present in
HTTP request

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

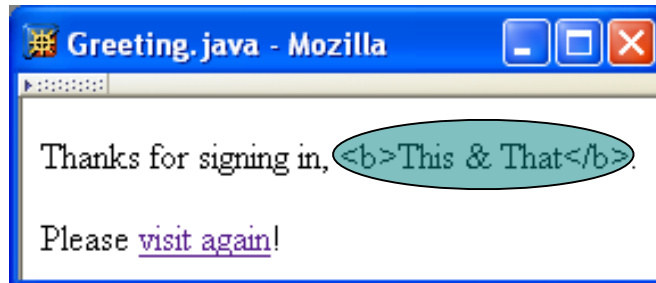
```
public void doPost (HttpServletRequest request,
                   HttpServletResponse response)
    ...
    String signIn = request.getParameter("signIn");
    HttpSession session = request.getSession();
    if (signIn != null) {
        printThanks(servletOut, signIn, "Greeting");
        session.setAttribute("signIn", signIn);
    } else {
        printSignInForm(servletOut, "Greeting");
    }
}
```

Generate
HTML for
response

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

Thank-you page



Must escape XML special characters in user input

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
public void doPost (HttpServletRequest request,
                   HttpServletResponse response)
...
String signIn = request.getParameter("signIn");
HttpSession session = request.getSession();
if (signIn != null) {
    printThanks(servletOut, signIn, "Greeting");
    session.setAttribute("signIn", signIn);
} else {
    printSignInForm(servletOut, "Greeting");
}
```

Assign a value to the signIn session attribute {

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

- Session attribute methods:
 - `setAttribute(String name, Object value)`: creates a session attribute with the given name and value
 - `Object getAttribute(String name)`: returns the value of the session attribute named name, or returns null if this session does not have an attribute with this name

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

```
public void doPost (HttpServletRequest request,
                   HttpServletResponse response)
    ...
    String signIn = request.getParameter("signIn");
    HttpSession session = request.getSession();
    if (signIn != null) {
        printThanks(servletOut, signIn, "Greeting");
        session.setAttribute("signIn", signIn);
    } else {
        printSignInForm(servletOut, "Greeting");
    }
```

Error
processing
(return user
to sign-in form)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Sessions

- By default, each session **expires** if a server-determined length of time elapses between a session's HTTP requests
 - Server destroys the corresponding session object
- Servlet code can:
 - **Terminate** a session by calling `invalidate()` method on session object
 - Set the **expiration time-out duration** (secs) by calling `setMaxInactiveInterval(int)`

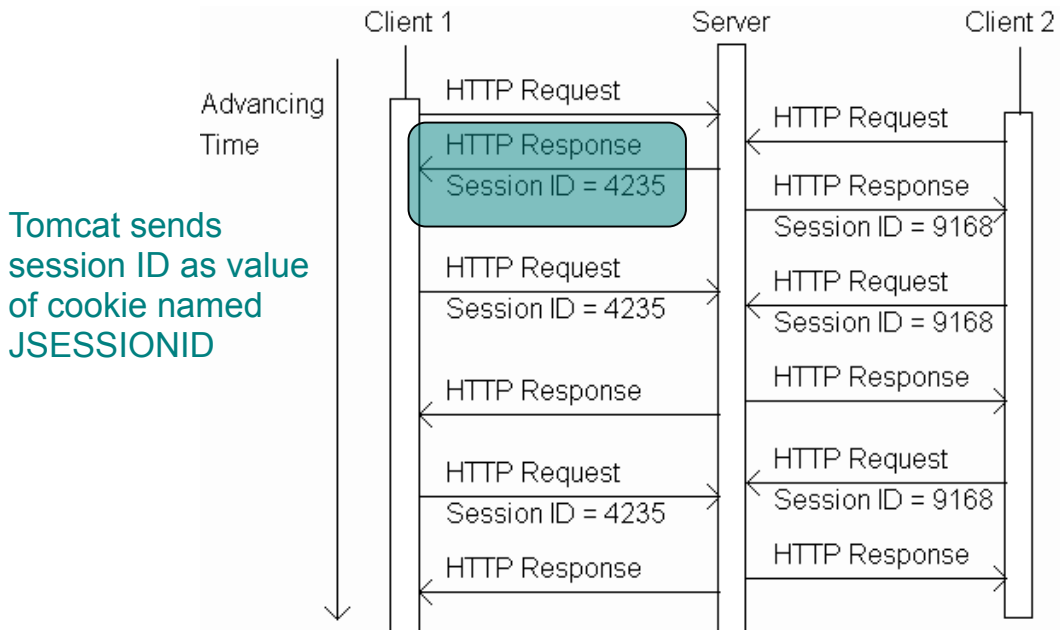
Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

- A **cookie** is a name/value pair in the Set-Cookie header field of an HTTP response
- Most (not all) clients will:
 - **Store** each cookie received in its file system
 - **Send** each cookie back to the server that sent it as part of the Cookie header field of subsequent HTTP requests

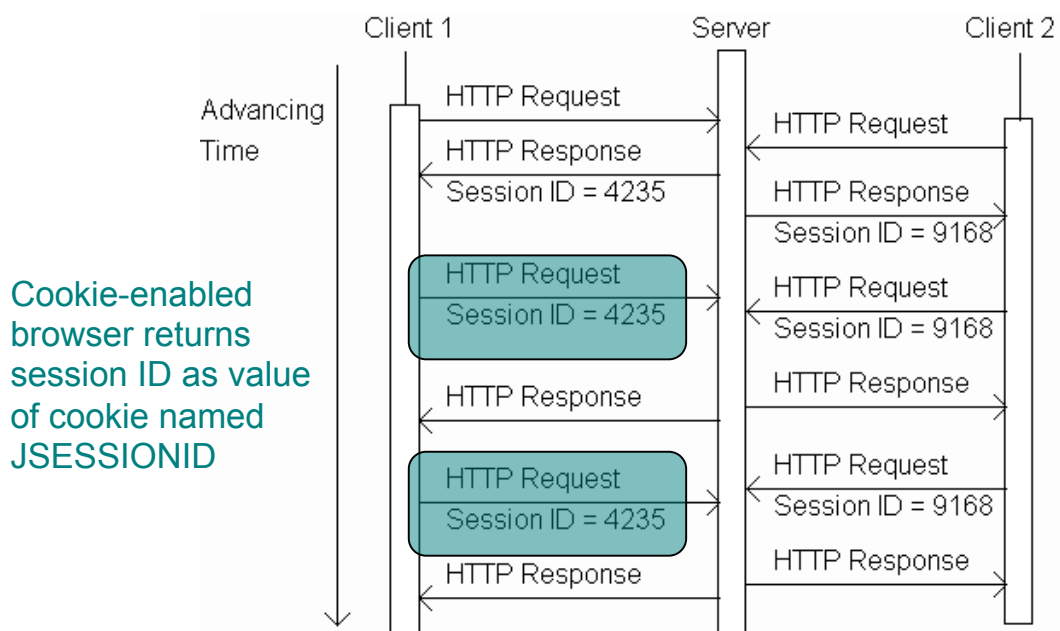
Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

- Servlets can set cookies explicitly
 - Cookie class used to represent cookies
 - `request.getCookies()` returns an array of Cookie instances representing cookie data in HTTP request
 - `response.addCookie(Cookie)` adds a cookie to the HTTP response

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

TABLE 6.3: Key Cookie class methods.

Method	Purpose
<code>Cookie(String name, String value)</code>	Constructor to create a cookie with given name and value
<code>String getName()</code>	Return name of this cookie
<code>String getValue()</code>	Return value of this cookie
<code>void setMaxAge(int seconds)</code> Cookies are expired by client (server can request expiration date)	Set delay until cookie expires. Positive value is delay in seconds, negative value means that the cookie expires when the browser closes, and 0 means delete the cookie.

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

```
public void doGet (HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException
{
    // Get count from cookie if available, otherwise
    // use initial value.
    int count = 0;
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (int i=0; (i<cookies.length) && (count==0); i++) {
            if (cookies[i].getName().equals("COUNT")) {
                count = Integer.parseInt(cookies[i].getValue());
            }
        }
    }
}
```

Return array of cookies contained in HTTP request

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

```
public void doGet (HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException
{
    // Get count from cookie if available, otherwise
    // use initial value.
    int count = 0;
    Cookie[] cookies = request.getCookies();
    if (cookies != null) {
        for (int i=0; (i<cookies.length) && (count==0); i++) {
            if (cookies[i].getName().equals("COUNT")) {
                count = Integer.parseInt(cookies[i].getValue());
            }
        }
    }
}
```

Search for cookie named COUNT and extract value as an int

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

Send replacement cookie value to client (overwrites existing cookie)

```
// Increment the count and add request to client to store it
// for one year.
count++;
Cookie cookie = new Cookie("COUNT",
                           new Integer(count).toString());
cookie.setMaxAge(oneYear);
response.addCookie(cookie);

// Set the HTTP content type in response header
response.setContentType("text/html; charset=\"UTF-8\"");
. . .
" <body> \n" +
"   <p>You have visited this page " + count + " time(s) \n" +
"     in the past year, or since clearing your cookies.</p> \n" +
" </body> \n" +
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

Should call addCookie() before writing HTML

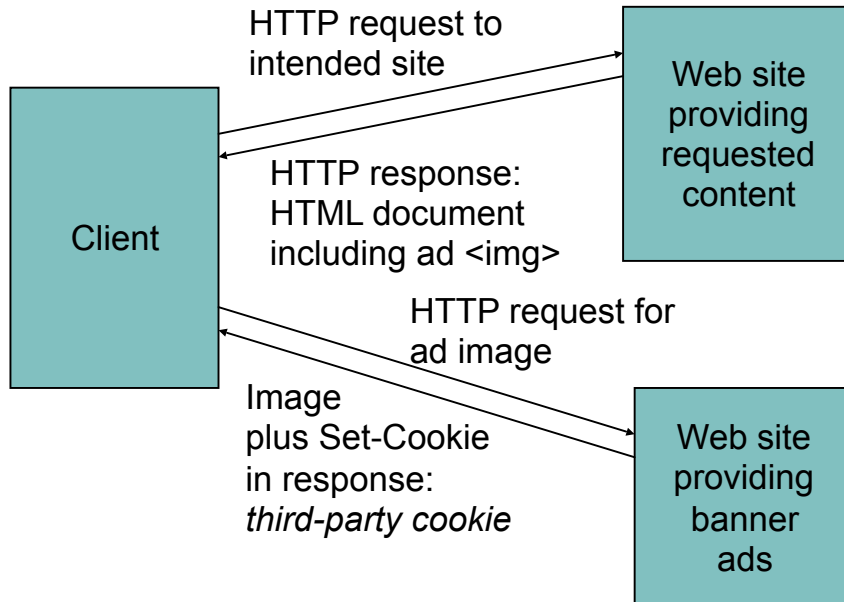
```
// Increment the count and add request to client to store it
// for one year.
count++;
Cookie cookie = new Cookie("COUNT",
                           new Integer(count).toString());
cookie.setMaxAge(oneYear);
response.addCookie(cookie);

// Set the HTTP content type in response header
response.setContentType("text/html; charset=\"UTF-8\"");
. . .
" <body> \n" +
"   <p>You have visited this page " + count + " time(s) \n" +
"     in the past year, or since clearing your cookies.</p> \n" +
" </body> \n" +
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

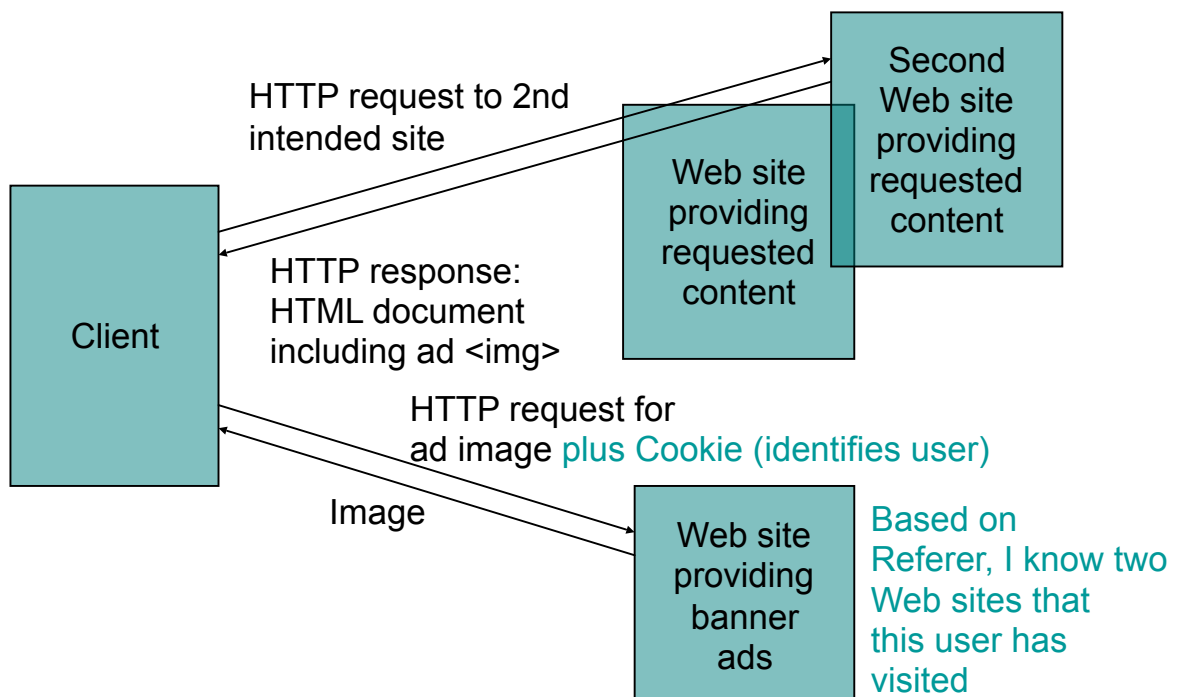
Privacy issues



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

Privacy issues



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Cookies

Privacy issues

- Due to privacy concerns, many users **block** cookies
 - Blocking may be fine-tuned. Ex: Mozilla allows
 - Blocking of third-party cookies
 - Blocking based on on-line privacy policy
- Alternative to cookies for maintaining session: **URL rewriting**

XML Syntax

- An XML document consists of
 - Markup
 - Tags, which begin with `<` and end with `>`
 - References, which begin with `&` and end with `;`
 - Character, e.g. ` `
 - Entity, e.g. `<`
 - » The entities `lt`, `gt`, `amp`, `apos`, and `quot` are recognized in every XML document.
 - » Other XHTML entities, such as `nbsp`, are only recognized in other XML documents if they are defined in the DTD
 - Character data: everything not markup

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Syntax

- Comments
 - Begin with `<!--`
 - End `-->`
 - Must not contain –
- **CDATA section**
 - Special element the entire content of which is interpreted as character data, even if it appears to be markup
 - Begins with `<![CDATA[`
 - Ends with `]]>` (illegal except when ending CDATA)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Syntax

- The CDATA section

```
<![CDATA[  
    <message>This & that</message>  
]]>
```

is equivalent to the markup

```
&lt;message&gt;This &amp; that&lt;/message&gt;
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Syntax

- < and & must be represented by references except
 - When beginning markup
 - Within comments
 - Within CDATA sections

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Syntax

- Element tags and elements
 - Three types
 - Start, e.g. <message>
 - End, e.g. </message>
 - Empty element, e.g.

 - Start and end tags must properly nest
 - Corresponding pair of start and end element tags plus everything in between them defines an **element**
 - Character data may only appear within an element

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Syntax

- Start and empty-element tags may contain attribute specifications separated by white space
 - Syntax: *name = quoted value*
 - *quoted value* must not contain <, can contain & only if used as start of reference
 - *quoted value* must begin and end with matching quote characters (‘ or “)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Syntax

- Element and attribute names are case sensitive
- XML white space characters are space, carriage return, line feed, and tab

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Documents

- A **well-formed XML document**
 - follows the XML syntax rules and
 - has a single root element
- Well-formed documents have a tree structure
- Many **XML parsers** (software for reading/writing XML documents) use tree representation internally

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Documents

- An XML document is written according to an **XML vocabulary** that defines
 - Recognized element and attribute names
 - Allowable element content
 - Semantics of elements and attributes
- XHTML is one widely-used XML vocabulary
- Another example: **RSS** (rich site summary)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Documents

```
<rss version="0.91">
  <channel>

    <title>www.example.com</title>
    <link>http://www.example.com/</link>
    <description>
      www.example.com is not a site that changes often...
    </description>
    <language>en-us</language>

    <item>
      <title>Announcing a Sibling Site!</title>
      <link>http://www.example.org/</link>
      <description>
        Were you aware that example.com is not the
        only site in the example family?
      </description>
    </item>

    <item>
      <title>We're Up!</title>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Documents

```
<link>http://www.example.net/</link>
<description>
  Our new RSS feed is up. Visit us today!
</description>
</item>
</channel>
</rss>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Documents

- Valid names and content for an XML vocabulary can be specified using
 - Natural language
 - XML DTDs (Chapter 2)
 - XML Schema (Chapter 9)
- If DTD is used, then XML document can include a document type declaration:

```
<!DOCTYPE rss
  SYSTEM "http://my.netscape.com/publish/formats/rss-0.91.dtd">
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Documents

- Two types of XML parsers:
 - Validating
 - Requires document type declaration
 - Generates error if document does not
 - Conform with DTD and
 - Meet XML validity constraints
 - » Example: every attribute value of type ID must be unique within the document
 - Non-validating
 - Checks for well-formedness
 - Can ignore external DTD

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Documents

- Good practice to begin XML documents with an **XML declaration**
 - Minimal example: `<?xml version="1.0"?>`
 - If included, < must be very first character of the document
 - To override default UTF-8/UTF-16 character encoding, include **encoding declaration** following version:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

XML Documents

- Internal subset of DTD

```
<!DOCTYPE rss
  SYSTEM "http://my.netscape.com/publish/formats/rss-0.91.dtd"
  [
    <!ENTITY vsn "0.91">
    <!ENTITY unused "This entity is not used.">
  ]
>
<rss version="&vsn;">
```

Declaration of
internal subset of DTD

- Entity vsn will be defined by any XML parser, validating or not

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Java-based DOM

- Java DOM API defined by org.w3c.dom package
- Semantically similar to JavaScript DOM API, but many small syntactic differences
 - Nodes of DOM tree belong to classes such as Node, Document, Element, Text
 - Non-method properties accessed via methods
 - Ex: parentNode accessed by calling getParentNode()

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Java-based DOM

- Methods such as `getElementsByTagName()` return instance of `NodeList`
 - `getLength()` method returns # of items
 - `item()` method returns an item

```
document.getElementsByTagName("link").item(0)
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Java-based DOM

- Example: program to count link elements in an RSS document:

```
DocumentBuilderFactory docBuilderFactory =
    DocumentBuilderFactory.newInstance();
DocumentBuilder parser = docBuilderFactory.newDocumentBuilder();

Document document = parser.parse(new File(args[0]));
NodeList links = document.getElementsByTagName("link");
System.out.println("Input document has " +
    links.getLength() +
    " 'link' elements.");
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Java-based DOM

- Imports:

From Java
API for XML
Processing
(JAXP)

```
// JAXP classes
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;

// DOM classes
import org.w3c.dom.Document;
import org.w3c.dom.NodeList;

// JDK classes
import java.io.File;
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Java-based DOM

- Default parser is non-validating and non-namespace-aware.
- Overriding:

```
DocumentBuilderFactory docBuilderFactory =
    DocumentBuilderFactory.newInstance();
docBuilderFactory.setNamespaceAware(true);
DocumentBuilder parser = docBuilderFactory.newDocumentBuilder();
```

- Also setValidating(true)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Java-based DOM

- Namespace-aware versions of methods end in NS:

```
NodeList links = Namespace name  
document.getElementsByTagNameNS(null, "link");  
Local name
```

WEB TECHNOLOGIES

A COMPUTER SCIENCE PERSPECTIVE

JEFFREY C. JACKSON

Chapter 8

Separating Programming and Presentation: JSP Technology

[all concept]

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Why JSP?

Overview of JSP page components

- Servlet/CGI approach: server-side code is a program with HTML embedded
- **JavaServer Pages** (and PHP/ASP/ColdFusion) approach: server-side “code” is a document with program embedded
 - Supports cleaner separation of **program logic** from **presentation**
 - Facilitates **division of labor** between developers and designers

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  <jsp:directive.page contentType="text/html" />
  <jsp:output
    omit-xml-declaration="yes"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />

  <head>
    <title>
      HelloCounter.jspx
    </title>
  </head>
```

Default namespace is XHTML

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  <jsp:directive.page contentType="text/html" />
  <jsp:output
    omit-xml-declaration="yes"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />

  <head>
    <title>
      HelloCounter.jspx
    </title>
  </head>
```

Also uses two JSP-defined namespaces

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  <jsp:directive.page contentType="text/html" />
  <jsp:output
    omit-xml-declaration="yes"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />
  <head>
    <title>
      HelloCounter.jspx
    </title>
  </head>
```

JSP-defined
markup (initialization)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  <jsp:directive.page contentType="text/html" />
  <jsp:output
    omit-xml-declaration="yes"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />
  <head>
    <title>
      HelloCounter.jspx
    </title>
  </head>
```

Standard XHTML

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<body>
<jsp:scriptlet>
  /* Initialize and update the "visits" variable. */
</jsp:scriptlet>
<c:if test="${empty visits}">
  <c:set var="visits" scope="application" value="0" />
</c:if>
<c:set var="visits" scope="application" value="${visits+1}" />

<p>
  Hello World!
</p>
<p>
  This page has been viewed
    ${visits}
  times since the most recent
  application restart.
</p>
</body>
</html>
```

JSP scriptlet

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<body>

<jsp:scriptlet>
  /* Initialize and update the "visits" variable. */
</jsp:scriptlet>
<c:if test="${empty visits}">
  <c:set var="visits" scope="application" value="0" />
</c:if>
<c:set var="visits" scope="application" value="${visits+1}" />

<p>
  Hello World!
</p>
<p>
  This page has been viewed
    ${visits}
  times since the most recent
  application restart.
</p>
</body>
</html>
```

JSP-based program logic:
initialize and increment variable

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<body>

  <jsp:scriptlet>
    /* Initialize and update the "visits" variable. */
  </jsp:scriptlet>
  <c:if test="{empty visits}">
    <c:set var="visits" scope="application" value="0" />
  </c:if>
  <c:set var="visits" scope="application" value="{visits+1}" />

  <p>
    Hello World!
  </p>
  <p>
    This page has been viewed
    {visits} Replaced with value of variable
    times since the most recent
    application restart.
  </p>
</body>
</html>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

Output
XHTML
document
after 3 visits

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"><head><title>
HelloCounter.jsp</title></head><body><p>
Hello World!</p><p>
This page has been viewed
  3
times since the most recent
application restart.</p></body></html>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

<html

```
xmlns="http://www.w3.org/1999/xhtml"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:c="http://java.sun.com/jsp/jstl/core">
<jsp:directive.page contentType="text/html" />
<jsp:output
  omit-xml-declaration="yes"
  doctype-root-element="html"
  doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
  doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />

<head>
  <title>
    HelloCounter.jspx
  </title>
</head>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

- Used html as root element
 - Can use HTML-generating tools, such as Mozilla Composer, to create the HTML portions of the document
 - JSP can generate other XML document types as well

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  <jsp:directive.page contentType="text/html" />
  <jsp:output
    omit-xml-declaration="yes"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />

  <head>
    <title>
      HelloCounter.jspx
    </title>
  </head>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

- Namespaces
 - JSP (basic elements, normal prefix jsp)
 - Core JSP Standard Tag Library (JSTL) (prefix c)
 - **Tag library**: means of adding functionality beyond basic JSP
 - JSTL included in with JWSDP 1.3 version of Tomcat
 - JSTL provides tag libraries in addition to core (more later)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  <jsp:directive.page contentType="text/html" />
  <jsp:output
    omit-xml-declaration="yes"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />

  <head>
    <title>
      HelloCounter.jspx
    </title>
  </head>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

- JSP elements
 - directive.page: typical use to set HTTP response header field, as shown (default is text/xml)
 - output: similar to XSLT output element (controls XML and document type declarations)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  <jsp:directive.page contentType="text/html" />
  <jsp:output
    omit-xml-declaration="yes"
    doctype-root-element="html"
    doctype-public="-//W3C//DTD XHTML 1.0 Strict//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd" />

  <head>
    <title>
      HelloCounter.jspx
    </title>
  </head>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<body>

  <jsp:scriptlet>
    /* Initialize and update the "visits" variable. */
  </jsp:scriptlet>
  <c:if test="${empty visits}">
    <c:set var="visits" scope="application" value="0" />
  </c:if>
  <c:set var="visits" scope="application" value="${visits+1}" />

  <p>
    Hello World!
  </p>
  <p>
    This page has been viewed
      ${visits}
    times since the most recent
    application restart.
  </p>
</body>
</html>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

- **Template data:** Like XSLT, this is the HTML and character data portion of the document
- **Scriptlet:** Java code embedded in document
 - While often used in older (non-XML) JSP pages, we will avoid scriptlet use
 - One use (shown here) is to add comments that will not be output to the generated page

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

```
<body>

  <jsp:scriptlet>
    /* Initialize and update the "visits" variable. */
  </jsp:scriptlet>
  <c:if test="${empty visits}">
    <c:set var="visits" scope="application" value="0" />
  </c:if>
  <c:set var="visits" scope="application" value="${visits+1}" />

  <p>
    Hello World!
  </p>
  <p>
    This page has been viewed
      ${visits}
    times since the most recent
    application restart.
  </p>
</body>
</html>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Example

- Core tag library supports simple programming
 - if: conditional
 - empty: true if variable is non-existent or undefined
 - set: assignment
 - **application scope** means that the variable is accessible by other JSP documents, other users (sessions)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP and Servlets

- JSP documents are not executed directly
 - When a JSP document is first visited, Tomcat
 1. Translates the JSP document to a servlet
 2. Compiles the servlet
 - The servlet is executed
- Exceptions provide traceback information for the servlet, *not* the JSP
 - The servlets are stored under Tomcat work directory

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP and Servlets

- A JSP-generated servlet has a `_jspService()` method rather than `doGet()` or `doPost()`
 - This method begins by automatically creating a number of **implicit object** variables that can be accessed by scriptlets

Object name	Instance of
<code>request</code>	<code>javax.servlet.http.HttpServletRequest</code>
<code>response</code>	<code>javax.servlet.http.HttpServletResponse</code>
<code>session</code>	<code>javax.servlet.http.HttpSession</code>
<code>out</code>	<code>javax.servlet.jsp.JspWriter</code>

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP and Servlets

- Translating template data:

```
out.write("<head>");
out.write("<title>");
out.write("\n      HelloCounter.jsp");
out.write("</title>");
out.write("</head>");
out.write("<body>");
```

- Scriptlets are copied as-is to servlet:

```
/* Initialize and update the "visits" variable. */
```

JSP and Servlets

- Scriptlets can be written to use the implicit Java objects:

```
<jsp:scriptlet>
    out.write("<p>Hello " +
              request.getParameter("username") +
              "!</p>");
</jsp:scriptlet>
```

- We will avoid this because:
 - It defeats the separation purpose of JSP
 - We can incorporate Java more cleanly using JavaBeans technology and tag libraries

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP and Servlets

- JSP elements translate to:

```
response.setContentType("text/html charset=UTF-8");
out.write("<!DOCTYPE html PUBLIC
\"-//W3C//DTD XHTML 1.0 Strict//EN\"
\"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">\n");
```

- `#{visits}` in template code translates to `out.write()` of value of variable
- Core tags (e.g., `if`) normally translate to a method call

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- A **web application** is a collection of resources that are used together to implement some web-based functionality
- Resources include
 - **Components**: servlets (including JSP-generated)
 - Other resources: HTML documents, style sheets, JavaScript, images, non-servlet Java classes, *etc.*

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- Sharing data between components of a web application
 - Tomcat creates one `ServletContext` object per web application
 - Call to `getServletContext()` method of a servlet returns the associated `ServletContext`
 - `ServletContext` supports `setAttribute()/getAttribute()` methods

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- Within Tomcat, all of the files of a simple web app are placed in a directory under webapps
 - JSP documents can go in the directory itself
 - “Hidden” files--such as servlet class files--go under a WEB-INF subdirectory (more later)
- Once the web app files are all installed, used Tomcat Manager to **deploy** the app

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- Deploying a web app consisting of a single JSP document HelloCounter.jspx:
 - Create directory webapps/HelloCounter
 - Copy JSP doc to this directory
 - Visit localhost:8080/manager/html
 - Enter HelloCounter in “WAR or Directory URL” box and click Deploy button
- Web app is now at URL localhost:8080/
HelloCounter/
HelloCounter.jspx

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- Manager app:
 - Stop: web app becomes unavailable (404 returned)
 - Start: web app becomes available again
 - Reload: stop web app, restart with latest versions of files (no need to restart server)
 - Undeploy: stop app and ***remove all files!***
 - Always keep a copy of app outside webapps

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- Set parameters of a web application by
 - Creating a deployment descriptor (XML file)
 - Saving the descriptor as WEB-INF/web.xml
- Simple example web.xml:

```
<web-app
  xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">

  <display-name>HelloCounter</display-name>
</web-app>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

TABLE 8.2: Some elements of web application deployment descriptors.

Element	Use (as child of web-app)
<code>display-name</code>	Provides name to be displayed for application (for example, in Manager's Display Name field)
<code>description</code>	Provides text describing the web application for documentation purposes
<code>context-param</code>	Provides parameter value that can be used by components for initialization
<code>servlet</code>	Associates a name with either a servlet class or a JSP document and optionally sets other options and parameters for the servlet/JSP document
<code>servlet-mapping</code>	Associates a URL (or a set of URL's) with one of the servlet names defined by a <code>servlet</code> element
<code>session-config</code>	Specifies the default for the length of time that a session can be idle before being terminated
<code>mime-mapping</code>	Associates file extensions with MIME types

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

<code>welcome-file-list</code>	Specifies a list of files. If an HTTP request is mapped to a directory within this application, the server will search for within the directory for one of these files and respond with the first file found. If no file is found, the directory contents are displayed by default.
<code>error-page</code>	Specifies a resource (static web page or application component) that will provide the HTTP response when either a specified HTTP error status code is generated or a specified Java exception is thrown to the container.
<code>jsp-config</code>	Associates certain information with the JSP documents of an application, such as the location of tag library files and settings for certain JSP options
<code>security-role</code>	Defines a "role" (e.g., manager, customer) to be used for purposes of allowing or denying access to certain resources of a web application
<code>security-constraint</code>	Specifies application resources that should be access-protected and indicates which user roles will be granted access to these resources
<code>login-config</code>	Specifies how the container should request user name and password information (which will subsequently be mapped to one or more roles) when a user attempts to access a protected resource

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- Some examples:
 - Setting an initial value accessible by `application.getInitParameter()`:

```
<context-param>
  <param-name>initialVisitsValue</param-name>
  <param-value>527</param-value>
</context-param>
```

- Setting the length of time (in minutes) before a session times out:

```
<session-config>
  <session-timeout>1</session-timeout>
</session-config>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- Mapping URLs to app components:

```
<servlet>
  <servlet-name>visit_count</servlet-name>
  <jsp-file>/HelloCounter.jsp</jsp-file>
</servlet>
<servlet-mapping>
  <servlet-name>visit_count</servlet-name>
  <url-pattern>*.jsp</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>visit_count</servlet-name>
  <url-pattern>/visitor/*</url-pattern>
</servlet-mapping>
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- There are four URL patterns (from high to low precedence)

TABLE 8.3: Forms of URL Patterns

Name	Example	Post-context path matched
Exact	/HelloCounter.jspx	The path /HelloCounter.jspx
Path-prefix	/visitor/*	The path /visitor or any path beginning with /visitor/
Extension	*.jsp	Any path ending in .jsp
Default	/	Any path

- If no URL pattern matches, Tomcat treats path as a relative file name

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

Web Applications

- Methods on request object for obtaining path information:
 - Example: /HelloCounter/visitor/test.jsp
 - getContextPath(): returns /HelloCounter
 - getServletPath(): returns /visitor
 - getPathInfo(): returns /test.jsp

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- `${visits+1}` is an example of an **EL expression** embedded in a JSP document
 - `${...}` is the syntax used in JSP documents to mark the contained string as an EL expression
 - An EL expression can occur
 - In **template data**: evaluates to Java String
 - As (part of) the value of certain JSP **attributes**: evaluates to data type that depends on context

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- **EL literals**:
 - true, false
 - decimal integer, floating point, scientific-notation numeric literals
 - strings (single- or double-quoted)
 - null

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- EL **variable names**: like Java
 - Can contain letters, digits, `_`, and `$`
 - Must not begin with a digit
 - Must not be reserved:

```
and   div   empty eq   false ge   gt   instanceof
le    lt    mod   ne    not   null  or    true
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- EL **operators**:
 - Relational: `<`, `>`, `<=`, `>=`, `==`, `!=`
 - Or equivalents: `lt`, `gt`, `le`, `ge`, `eq`, `ne`
 - Logical: `&&`, `||`, `!`
 - Or equivalents: `and`, `or`, `not`
 - Arithmetic:
 - `+`, `-` (binary and unary), `*`
 - `/`, `%` (or `div`, `mod`)
 - `empty`: true if arg is null or empty string/array/Map/Collection
 - Conditional: `? :`
 - Array access: `[]` (or object notation)
 - Parentheses for grouping

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- EL **automatic type conversion**
 - Conversion for + is like other binary arithmetic operators (+ does *not* string represent concatenation)
 - Otherwise similar to JavaScript

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- EL provides a number of **implicit objects**
- Most of these objects are related to but **not the same as** the JSP implicit objects
 - JSP implicit objects cannot be accessed directly by name in an EL expression, but can be accessed indirectly as properties of one of the EL implicit objects

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

TABLE 8.4: EL implicit objects.

EL Implicit Object Name	Represents
<code>pageContext</code>	Container for JSP implicit objects
<code>pageScope</code>	Values accessible via calls to <code>page.getAttribute()</code>
<code>requestScope</code>	Values accessible via calls to <code>request.getAttribute()</code>
<code>sessionScope</code>	Values accessible via calls to <code>session.getAttribute()</code>
<code>applicationScope</code>	Values accessible via calls to <code>application.getAttribute()</code>
<code>param</code>	Values accessible via <code>request.getParameter()</code>
<code>paramValues</code>	Values accessible via <code>request.getParameterValues()</code>
<code>header</code>	Values accessible via <code>request.getHeader()</code>
<code>headerValues</code>	Values accessible via <code>request.getHeaders()</code>
<code>cookie</code>	Map from cookie names to their associated Cookie values (data obtained via <code>request.getCookies()</code>)
<code>initParam</code>	Values accessible via <code>application.getInitParameter()</code>

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- `pageContext`: provides access to JSP implicit objects
 - Ex: EL expression `pageContext.request` is reference to the JSP request object
- `page`: JSP implicit object representing the servlet itself
- JSP objects `page`, `request`, `session`, and `application` all have `getAttribute()` and `setAttribute()` methods
 - These objects store **EL scoped variables** (e.g., visits)

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- Reference to non-implicit variable is resolved by looking for an EL scoped variable in the order:
 - page
 - request
 - session
 - application
- If not found, value is null
- If found, value is Object
 - JSP automatically casts this value as needed

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- All EL implicit objects except pageContext implement Java Map interface
- In EL, can access Map using array or object notation:
 - Servlet: `request.getParameter("p1")`
 - EL:
 - `param['p1']`
 - or
 - `param.p1`

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- Array/List access:

If EL scoped variable aVar represents

- Java array; or
- java.util.List

and if EL scoped variable index can be cast to integer

then can access elements of aVar by

- aVar[index]
- aVar.index

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

JSP Expression Language (EL)

- Function call

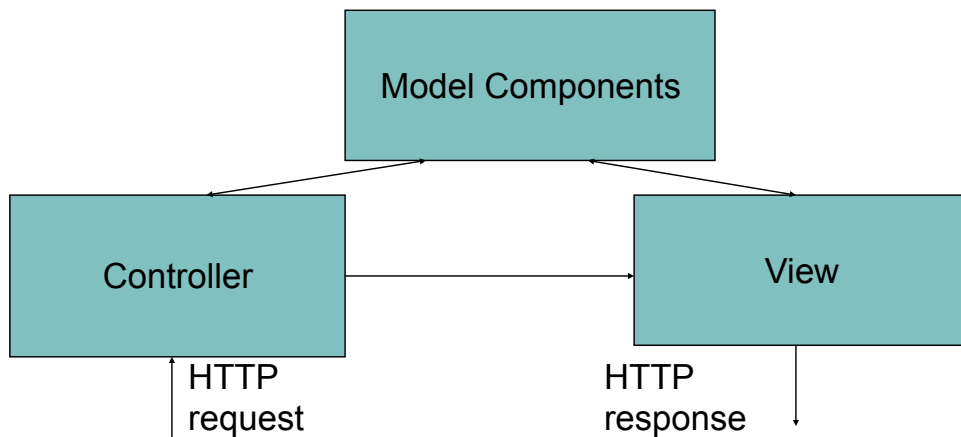
- Function name followed by parenthesized, comma-separated list of EL expression arguments
- Tag libraries define all functions
- Function names usually include a namespace prefix associated with the tag library

```
fn:toLowerCase(param['username'])
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

MVC

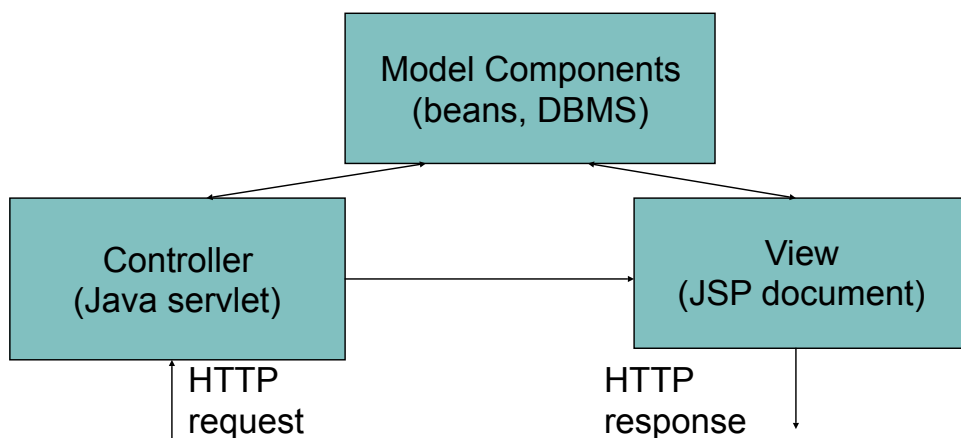
- Many web apps are based on the Model-View-Controller (MVC) architecture pattern



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

MVC

- Typical JSP implementation of MVC



Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

MVC

- Forwarding an HTTP request from a servlet to another component:
 - By URL

```
RequestDispatcher dispatcher =  
    getServletContext().getRequestDispatcher(contextRelativeURL);
```

Ex: /HelloCounter.jspx

- By name

```
<servlet>  
    <servlet-name>visit_count/servlet-name>  
    <jsp-file>/HelloCounter.jspx</jsp-file>  
</servlet>  
  
RequestDispatcher dispatcher =  
    getServletContext().getNamedDispatcher("visit_count");
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

MVC

```
public class Controller extends HttpServlet  
{  
    /**  
     * If session is new then increment and display the application  
     * visit counter. Otherwise (this is the continuation of an  
     * active session), display a message.  
     */  
    public void doGet (HttpServletRequest request,  
                      HttpServletResponse response)  
        throws ServletException, IOException  
    {  
        HttpSession session = request.getSession();  
        if (session.isNew()) {  
            RequestDispatcher visitDispatch =  
                getServletContext().getNamedDispatcher("visit_count");  
            visitDispatch.forward(request, response);  
        }  
        else {  
            RequestDispatcher laterDispatch =  
                getServletContext().getNamedDispatcher("visit_later");  
            laterDispatch.forward(request, response);  
        }  
    }  
}
```

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

MVC

- How does the controller know which component to forward to?
 - getPathInfo() value of URL's can be used
 - Example:
 - servlet mapping pattern in web.xml:

- URL ends with: `/controller/*`
- `getPathInfo()` returns: `/controller/help?prod=324324`
- `getPathInfo()` returns: `/help`

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

MVC

- JSP include action (not the same as the include directive!)

```
<table style="width:100%" border="0">
  <tbody>
    <tr>
      <td style="width:20%"
        ><jsp:include page="/navbar.jsp" /></td>
      <td style="width:80%"
        ><jsp:include page="/mainContent.jsp" /></td>
    </tr>
  </tbody>
</table>
```

Execute specified component and include its output in place of the include element

Jackson, Web Technologies: A Computer Science Perspective, © 2007 Prentice-Hall, Inc. All rights reserved. 0-13-185603-0

MVC

- Adding parameters to the request object seen by an included component:

```
<jsp:include page="/navbar.jspx">  
  <jsp:param name="currentPage" value="home" />  
</jsp:include>
```

request object seen by navbar.jspx will include
parameter named currentPage with value home



Week 14 [just concept]

IT230

Dr Mohamed Habib

fppt.com



JDBC

JDBC provides a standard library for accessing relational databases. By using the JDBC API, you can access a wide variety of SQL databases with exactly the same Java syntax. It is important to note that although the JDBC API standardizes the approach for connecting to databases, the syntax for sending queries and committing transactions, and the data structure representing the result, JDBC does *not* attempt to standardize the SQL syntax. So, you can use any SQL extensions your database vendor supports. However, since most queries follow standard SQL syntax, using JDBC lets you change database hosts, ports, and even database vendors with minimal changes to your code.

fppt.com



JDBC

Seven Steps for Database Access

concept

- Load the JDBC driver
- Define the connection URL
- Establish the connection
- Create a Statement object
- Execute a query or update
- Process the result set
- Close the statement and connection

fppt.com



JDBC

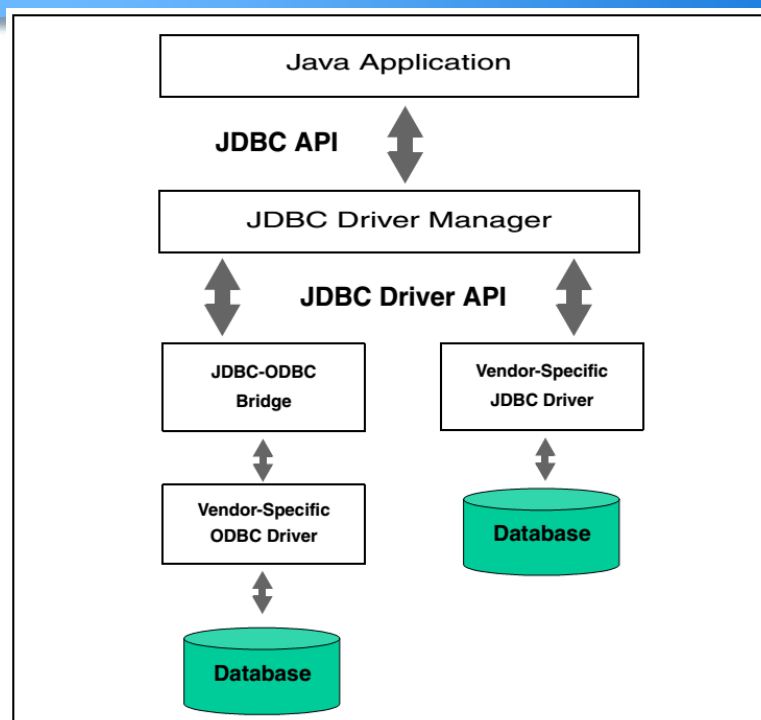
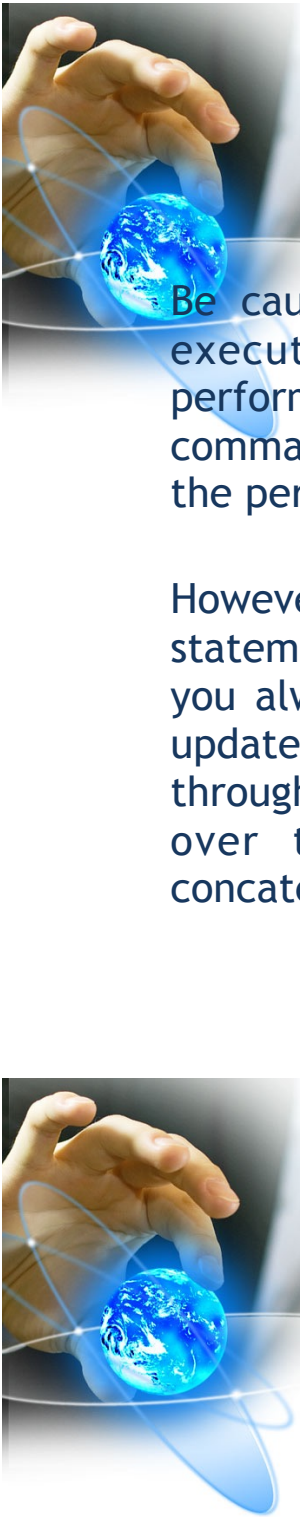


Figure 17-1 Two common JDBC driver implementations. JDK 1.4 includes a JDBC-ODBC bridge; however, a pure JDBC driver (provided by the vendor) yields better performance.

fppt.com



Adv. Of Prepared Statements concept

Be cautious though: a prepared statement does not always execute faster than an ordinary SQL statement. The performance improvement can depend on the particular SQL command you are executing. For a more detailed analysis of the performance for prepared statements in Oracle.

However, performance is not the only advantage of a prepared statement. Security is another advantage. We recommend that you always use a prepared statement or stored procedure to update database values when accepting input from a user through an HTML form. This approach is strongly recommended over the approach of building an SQL statement by concatenating strings from the user input values.

fppt.com

Adv. Of Prepared Statements

Core Approach

To avoid an SQL Injection Attack when accepting data from an HTML form, use a prepared statement or stored procedure to update the database.

fppt.com



Creating Callable Statements concept

With a CallableStatement, you can execute a stored procedure or function in a database. For example, in an Oracle database, you can write a procedure or function in PL/SQL and store it in the database along with the tables. Then, you can create a connection to the database and execute the stored procedure or function through a CallableStatement.

fppt.com



Adv. Of Callable Statements

A stored procedure has many advantages. For instance, syntax errors are caught at compile time instead of at runtime; the database procedure may run much faster than a regular SQL query; and the programmer only needs to know about the input and output parameters, not the table structure. In addition, coding of the stored procedure may be simpler in the database language than in the Java programming language because access to native database capabilities (sequences, triggers, multiple cursors) is possible.

fppt.com



DisAdv. Of Callable Statements

One disadvantage of a stored procedure is that you may need to learn a new database-specific language (note, however, that Oracle8i Database and later support stored procedures written in the Java programming language).

A second disadvantage is that the business logic of the stored procedure executes on the database server instead of on the client machine or Web server.

fppt.com



Creating Callable Statements

1. **Define the call to the database procedure.** As with a prepared statement, you use special syntax to define a call to a stored procedure. The procedure definition uses escape syntax, where the appropriate ? defines input and output parameters.
2. **Prepare a CallableStatement for the procedure.** You obtain a CallableStatement from a Connection by calling prepareCall.
3. **Register the output parameter types.** Before executing the procedure, you must declare the type of each output parameter.
4. **Provide values for the input parameters.** Before executing the procedure, you must supply the input parameter values.
5. **Execute the stored procedure.** To execute the database stored procedure, call execute on the CallableStatement.
6. **Access the returned output parameters.** Call the corresponding getXxx method, according to the output type.

fppt.com



Using Database Transactions concept

When a database is updated, by default the changes are permanently written (or *committed*) to the database. However, this default behavior can be programmatically turned off. If autocommitting is turned off and a problem occurs with the updates, then each change to the database can be backed out (or rolled back to the original values). If the updates execute successfully, then the changes can later be permanently committed to the database. This approach is known as *transaction management*.

fppt.com



Using Database Transactions

The default for a database connection is autocommit; that is, each executed statement is automatically committed to the database. Thus, for transaction management you first need to turn off autocommit for the connection by calling `setAutoCommit(false)` .

fppt.com



Using Database Transactions

- Here, the statement for obtaining a connection from the DriverManager is outside the try/catch block. That way, rollback is not called unless a connection is successfully obtained. However, the getConnection method can still throw an SQLException and must be thrown by the enclosing method or be caught in a separate try/catch block.