IT230 Summary week 6 to 14

IT244
CS140
داتا 1
الجامعة السعودية الالكترونية
CS141
ويب230
IT230
جافا 2
جافا 1
وليد عمر
خبرة 10 سنوات
شروحات فيديو
خصومات للمجموعات
بأسعار رمزية
جوال - واتس : 00966509114811
http://www.egycbt.com/

Chapter 6

Server-side Programming

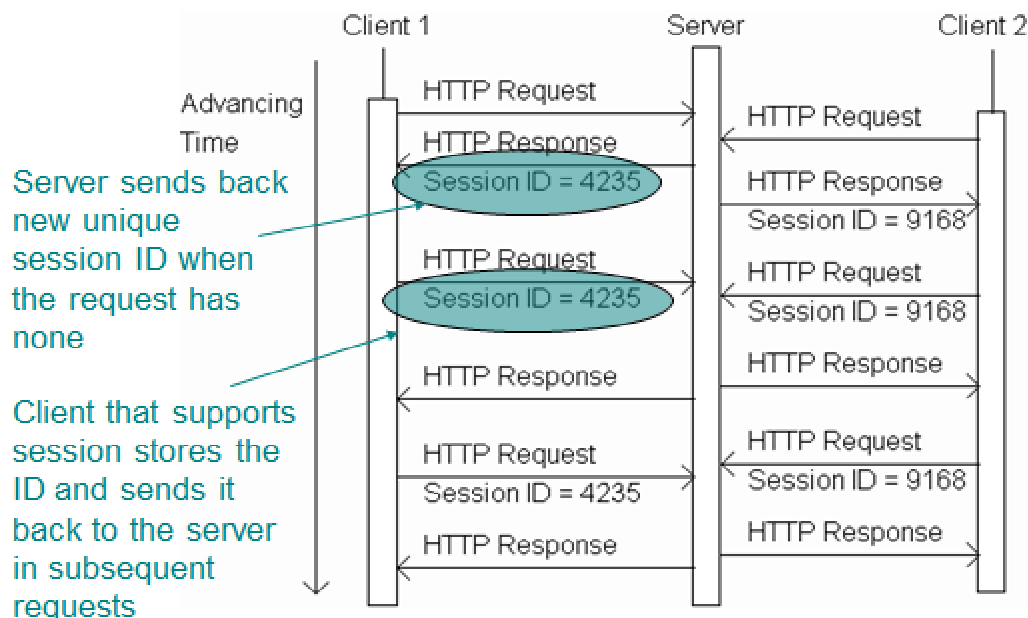
- The combination of
  - HTML
  - JavaScript
  - DOM

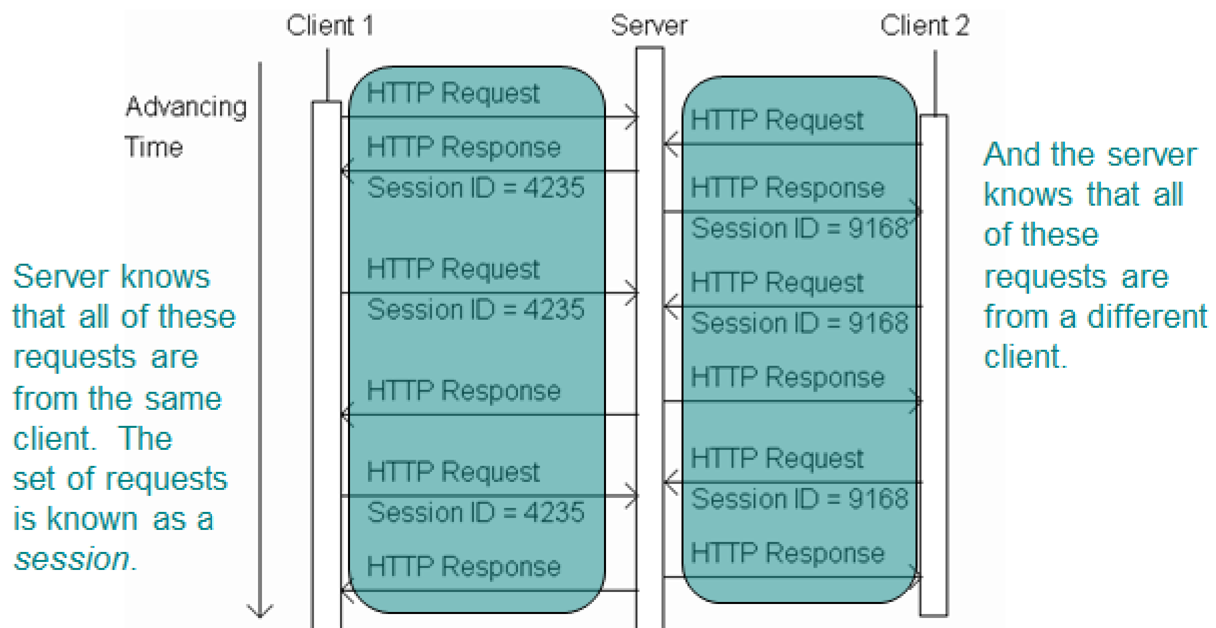
  is sometimes referred to as Dynamic HTML (DHTML)
- Web pages that include scripting are often called dynamic pages (vs. static)
- Similarly, web server response can be static or dynamic
  - Static: HTML document is retrieved from the file system and returned to the client
  - Dynamic: HTML document is 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

Id Omar - 0509114811

Session

- 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 use





- **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

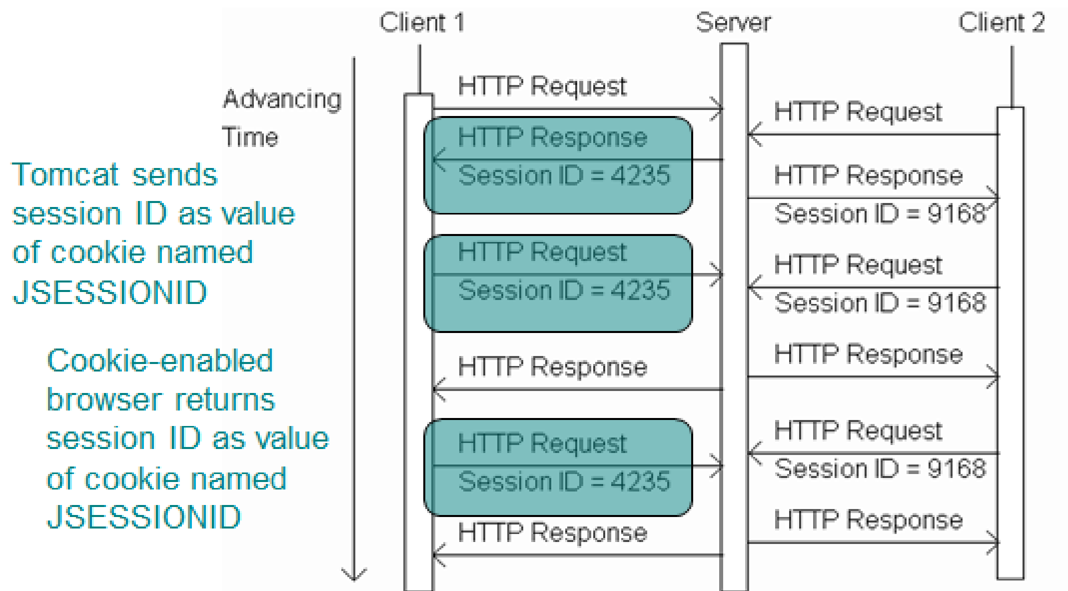
- By default, each session expires if a server-determined length of time elapses between a session's HTTP requests
  - Server destroys the corresponding session object

- Servlet code can:
  - Terminate a session by calling invalidate() method on session object
  - Set the expiration time-out duration (secs) by calling setMaxInactiveInterval(int)

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

Walid Omar - 05001148



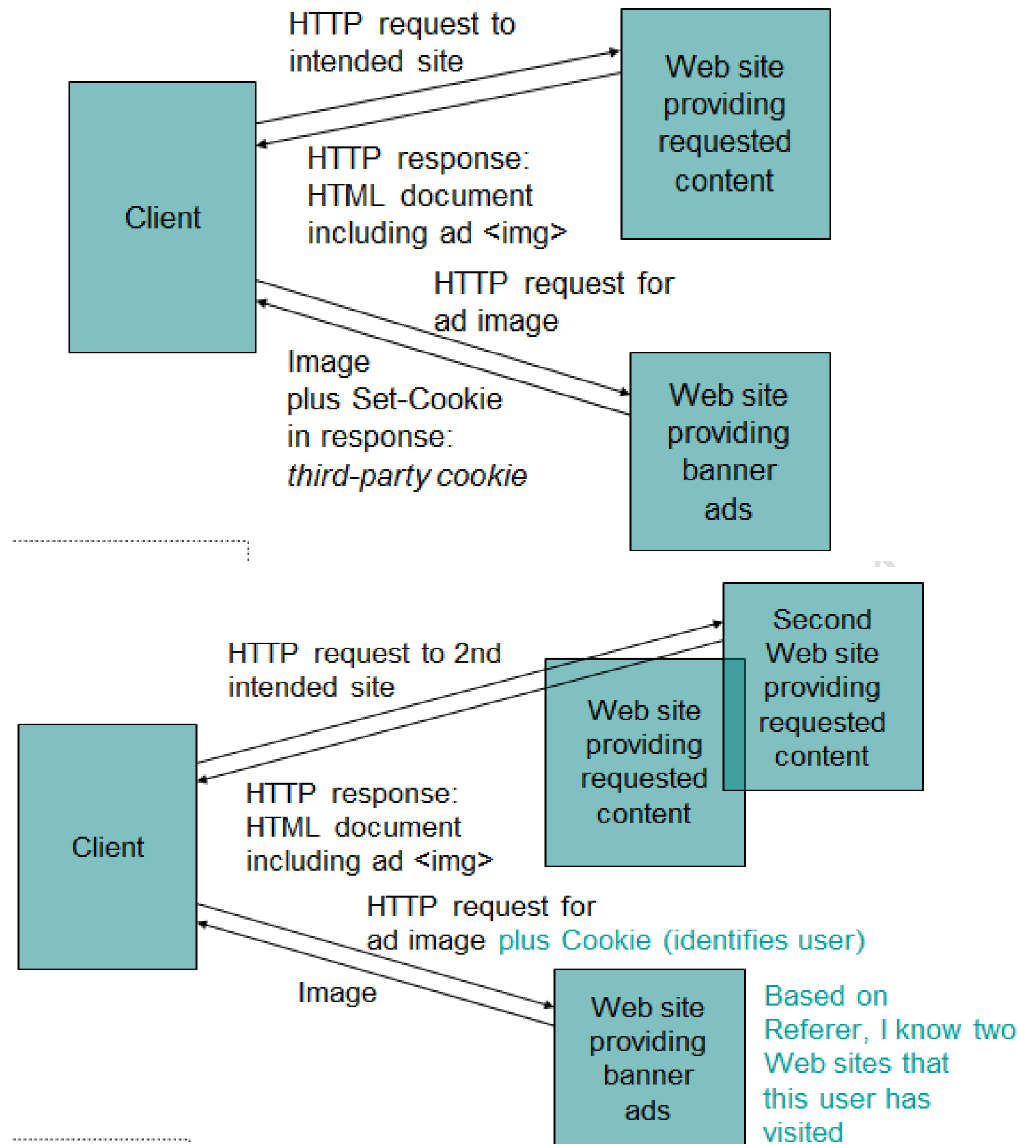
- **Servlets can set cookies explicitly**
  - Cookie class used to represent cookies
  - request.getCookies() returns an array of Cookie instances representing cookie data in HTTP request
  - response.addCookie(Cookie) adds a cookie to the HTTP response

TABLE 6.3 Key Cookie Class Methods

Method	Purpose
`Cookie(String name, String value)`	Constructor to create a cookie with given name and value.
`String getName()`	Return name of this cookie.
`String getValue()`	Return value of this cookie.
`void setMaxAge(int seconds)`	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.

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



Chapter 7

An XML document is one that follows certain syntax rules (most of which we followed for XHTML)

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

• 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)

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

• 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

Walid Omar - 0509114811

• Start and empty-element tags may contain attribute specifications separated by white space
  – *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 “)
• Element and attribute names are case sensitive
• XML white space characters are space, carriage return, line feed, and tab

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

• 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)

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

• 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:

• 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

• Good practice to begin XML documents with an XML declaration

– Minimal example:

– 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"?>
```

• 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

Java-based DOM

• Java DOM API defined by org.w3c.dom package

• Semantically similar to JavaScript DOM API, but many small syntactic differences

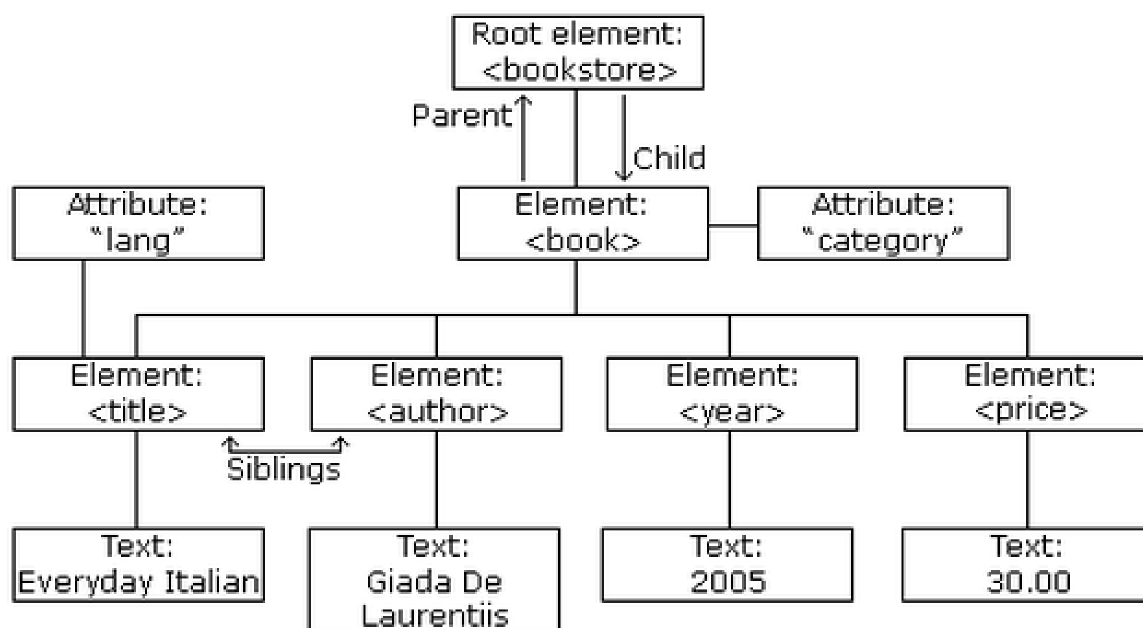
– Nodes of DOM tree belong to classes such as Node, Document, Element, Text

– Non-method properties accessed via methods • Ex: parentNode accessed by calling getParentNode()

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

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

Example



Convert tree to XML?

Answer

```
<bookstore>
  <book category="c">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
</bookstore>
```

Chapter8

Why JSP?

• Servlet/CGI approach: server-side code is a program with HTML embedded

• Java Server Pages (and PHP/ASP) approach: serverside "code" is a document with program embedded – Supports cleaner separation of program logic from presentation – Facilitates division of labor between developers and designers

• 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

• 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)

• 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

Walid Omar - 0509114811

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
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
session	javax.servlet.http.HttpSession
out	javax.servlet.jsp.JspWriter

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

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

Walid Omar - 0509114811

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

- EL variable names: like Java
  - Can contain letters, digits, _ , and $
  - Must not begin with a digit
  - Must not be reserved:
- EL automatic type conversion
  - Conversion for + is like other binary arithmetic operators (+ does *not* string represent concatenation)
  - Otherwise similar to JavaScript

- 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

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

Walid O.

- 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

- 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

• Function call

– Function name followed by parenthesized, comma-separated list of EL expression arguments

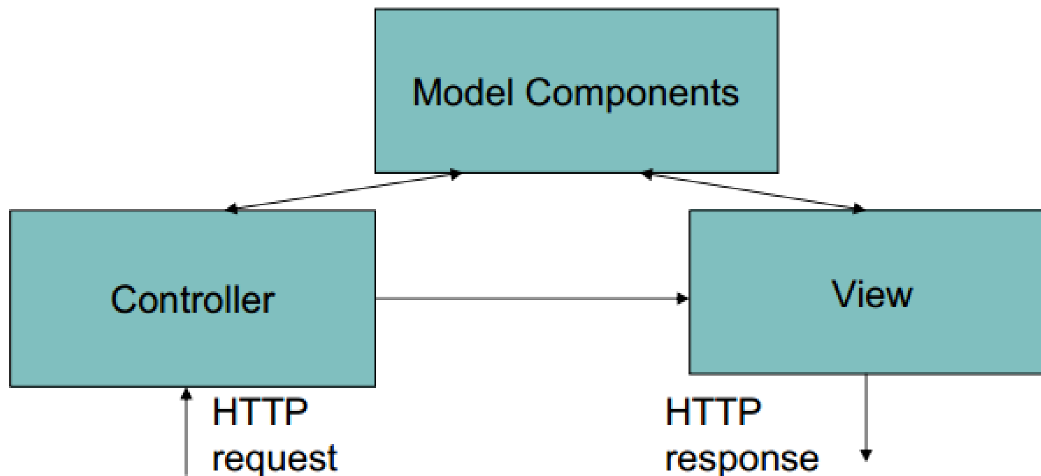
– Tag libraries define all functions

– Function names usually include a namespace prefix associated with the tag library

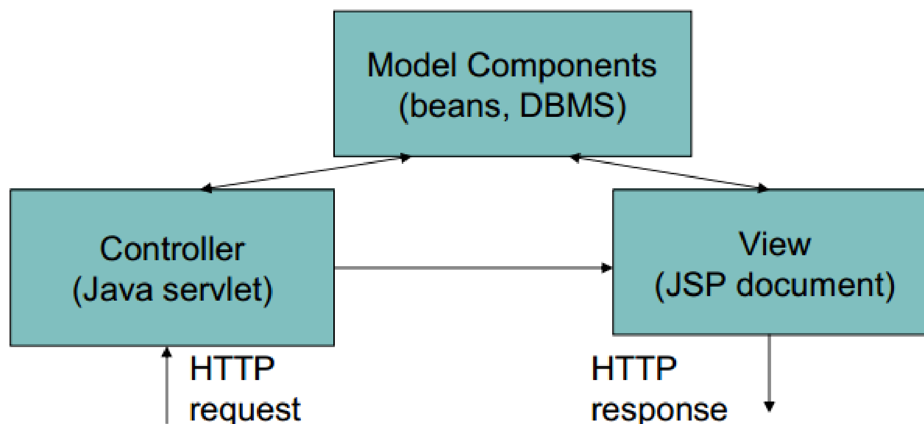
```
fn:toLowerCase(param['username'])
```

MVC

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



• Typical JSP implementation of 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: `/controller/*`
- URL ends with: `/controller/help?prod=324324`
- getPathInfo() returns: `/help`

• JSP include action (not the same as the include directive!)
• Adding parameters to the request object seen by an included component:

Ch17-18 JDBC

DBC 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.

Walid Omar 0509114811

Seven Steps for Database Access

– Load the JDBC driver

```java
public class DBExample {
  private static final String dbURL =
      "jdbc:derby:MyDB;create=true";
  private static final String driver =
      "org.apache.derby.jdbc.EmbeddedDriver";
  private static String username = "dbuser";
  private static String password = "p@55w0rd";

  private Connection connection = null;

  // continued...
}
```

– Define the connection URL – Establish the connection

```
public class DBExample {

 private Connection getConnection() {
    try {
      if (connection == null ||
            connection.isClosed()) {
        Class.forName(driver).newInstance();
        connection = DriverManager.getConnection(
            dbURL, username, password);
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
    return connection;
  }
  // continued...
}
```

– Create a Statement object
– Execute a query or update

```
public class DBExample {

 protected ResultSet executeQuery(String sql) {
    try {
      Statement statement =
          getConnection().createStatement();
      ResultSet results =
          statement.executeQuery(sql);
      statement.close();
      return results;
    } catch (SQLException e) {
      e.printStackTrace();
    }
    return null;
  }

  // continued...
}
```

Creating tables

```
CREATE SCHEMA GUESTBOOK

CREATE TABLE GUESTBOOK.GUESTS (
  id INTEGER NOT NULL
    GENERATED ALWAYS AS IDENTITY
    (START WITH 1, INCREMENT BY 1),
  name VARCHAR(50),
  message VARCHAR(255),
  PRIMARY KEY (id)
)
```

```
public class DBExample {
  public void createTables() {
    String schemaSql = ...; // DB specific SQL
    String tableSql = ...;  // DB specific SQL
    try {
      DatabaseMetaData dbmd =
          getConnection().getMetaData();
      ResultSet rs = dbmd.getTables(
          null, "GUESTBOOK", "GUESTS", null);
      if (!rs.next()) {
        executeSql(schemaSql);
        executeSql(tableSql);
      }
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }
  // continued...
}
```

Inserting data

```
public class DBExample {

 public void insertData() {
    String insertSql =
        "INSERT INTO GUESTBOOK.GUESTS " +
        "    (name, message)" +
        "VALUES ";
    String[] values = {
        "('Abdullah', 'Nice site!')",
        "('Ahmed', 'Nice to see.')",
        "('Omar', 'How did you make this?')"
    };
    for (String value : values) {
      executeSql(insertSql + value);
    }
  }
  // continued...
}
```

– Process the result set

Querying

```
public class DBExample {
  public void queryData() {
     try {
       String querySql =
           "SELECT * FROM GUESTBOOK.GUESTS";
       ResultSet results = executeQuery(querySql);
       while (results.next()) {
         int id = results.getInt("id");
         String name = results.getString("name");
         String message =
             results.getString("message");
         System.out.printf("%8d %-15s %-50s\n",
             id, name, message);
       }
     } catch (SQLException e) {
       e.printStackTrace();
     }
   }
   // continued...
}
```

```
1 Abdullah    Nice site!
2 Ahmed       Nice to see.
3 Omar        How did you make this?
```

– Close the statement and connection

Closing the connection

```
public class DBExample {

  private void closeConnection() {
    try {
      if (connection != null &&
          !connection.isClosed()) {
        DriverManager.getConnection(dbURL
            + ";shutdown=true");
        connection.close();
        connection = null;
      }
    } catch (SQLException e) {
      // ignore
    }
  }

}
```

Advantage of Prepared Statements

Performance:

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.

Security:

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 to avoid SQL injection attack. This approach is strongly recommended over the approach of building an SQL statement by concatenating strings from the user input values.

Creating Callable Statements

With a Callable Statement, 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 Callable Statement.

Walid omar - 0509114811

Advantage 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.

• 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.

disadvantage 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.

Using Database Transactions

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*.**

The default for a database connection is **autocommit**; 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)**

Ch2-3-4-5-6-7 PHP

كله مهم – يجي منه

- **T/F**
- **MCQ**
- **Output**
- **Code**
- **Essay**

مهم تشوفوا شرح ويك ١٢-١٣-١٤ كلهم و مراجعات php extra

Walid Omar - 0509114811

Ch4-5 PHP

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**.

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.

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.

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).

Walid Omar - 0509114811

- 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.

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

- **MySQL Strengths and Weaknesses**
  - **Strength: Great Market Penetration**
  - **Strength: Easy to Get Started**
  - **Strength: Open-Source License for Most Users**
  - **Strength: Fast**
  - **Weakness: Commercial License for Commercial Redistribution**
  - **Strength: Reasonable Scalability**

Buffered Versus Unbuffered Queries

Buffered Queries	UnBuffered Quesies
• **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	• **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.

Walid Omar - 0509114811

- **TYPES OF ERRORS**

– **Programming Errors**

• **Syntax/Parse Errors**

Syntax errors and other parse errors are caught when a file is compiled, before PHP starts executing it at all

• **Eval**
All syntax or parse errors are caught during compilation, except errors in code executed with eval(). In the case of eval, the code is compiled during the execution of the script.

• **Include / Require:** If your script includes another file that has a parse error, compilation will stop at the parse error. Code and declarations preceding the parse error are compiled, and those following the error are discarded.

- Undefined Symbols:

• When PHP executes, it may encounter names of variables, functions, and so on that it does not know. Because PHP is a loosely typed interpreted language, it does not have complete knowledge about all symbol names, function names, and so on during compilation.

1-Variables and Constants
Variables and constants are not dramatic, and they go by with just a notice.
2-Array Indexes
3-**Functions and Classes**
Although PHP keeps executing after running across an undefined variable or constant, it aborts whenever it encounters an undefined function or class

Waid Omar - 0509114811

• Logical Errors

• A more subtle type of programming error is a logical error , errors that are in the structure and logic of the code rather than just the syntax. The best ways to find logical errors is testing combined with code reviews.

– Portability Errors

- •Operating System Differences
- •PHP Configuration Differences
- •SAPI Differences

– 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.
- • Error Reporting Several php.ini configuration settings control which errors should be displayed and how.