ELECTRONICS AND COMPUTER ENGINEERING

LECTURE NOTES

Web Technologies

J.B. INSTITUTE OF ENGINEERING AND TECHNOLOGY
(UGC AUTONOMOUS)
Bhaskar Nagar, Moinabad Mandal , R.R. District, Hyderabad -500075

Lecture notes

UNIT – 1

OVER VIEW:

Unit-1 demonstrates how to create static web pages using basic HTML tags and presentation of content using CSS. This unit focuses on how to create tables, forms, and lists to display the content in web pages.

CONTENTS:

1. Basic HTML Tags
a. List
b. Tables
c. Images
d. Forms
e. frames
2. Cascading Style Sheets
a. Three mechanisms by which we can apply styles
b. Forms of CSS

HTML stands for Hypertext Markup Language. It is used to display the document in the web browsers. HTML pages can be developed to be simple text or to be complex multimedia program containing sound, moving images and java applets. HTML is considered to be the global publishing format for Internet. It is not a programming language. HTML was developed by Tim Berners-Lee. HTML standards are created by a group of interested organizations called W3C (world wide web consortium). In HTML formatting is specified by using tags. A tag is a format name surrounded by angle brackets. End tags which switch a format off also contain a forward slash.

Basic HTML tags

1. Body tag :

Body tag contain some attributes such as bgcolor, background etc. bgcolor is used for background color, which takes background color name or hexadecimal number and #FFFFFF and background attribute will take the path of the image which you can place as the background image in the browser.

<body bgcolor=”#F2F3F4” background= “c:\amer\imag1.gif”>

2. Paragraph tag:

Most text is part of a paragraph of information. Each paragraph is aligned to the left, right or center of the page by using an attribute called as align.

<p align=”left” | “right” | “center”>

3. Heading tag:

HTML is having six levels of heading that are commonly used. The largest heading tag is <h1> . The different levels of heading tag besides <h1> are <h2>, <h3>, <h4>, <h5> and <h6>. These heading tags also contain attribute called as align.

<h1 align=”left” | “right” | “center”> . . . . <h2>

4. hr tag:

This tag places a horizontal line across the system. These lines are used to break the page. This tag also contains attribute i.e., width which draws the horizontal line with the screen size of the browser. This tag does not require an end tag.

<hr width=”50%”>.

5. base font:

This specify format for the basic text but not the headings.
<basefont size=”10”>
6. font tag:
This sets font size, color and relative values for a particular text.
<font size=”10” color=”#f1f2f3”>
7. bold tag:
This tag is used for implement bold effect on the text
 …….
8. Italic tag:
This implements italic effects on the text.
…….
9. strong tag:
This tag is used to always emphasized the text
…….
10. tt tag:
This tag is used to give typewriting effect on the text
<tt>……..</tt>
11. sub and sup tag:
These tags are used for subscript and superscript effects on the text.
_{…….}
^{……..}
12. Break tag:
This tag is used to the break the line and start from the next line.

13. &amp &lt &gt   &quot
These are character escape sequence which are required if you want to display
characters that HTML uses as control sequences.
Example: < can be represented as &lt.
14. Anchor tag:
This tag is used to link two HTML pages, this is represented by <a>
 some text
href is an attribute which is used for giving the path of a file which you want to
link.

Lists:

One of the most effective ways of structuring a web site is to use lists. Lists provides
straight forward index in the web site. HTML provides three types of list i.e.,

1. unordered list,

2. ordered list and

3. definition list.

Lists can be easily embedded easily in another list to provide a complex but readable structures.
The different tags used in lists are
explained below.
 …..
The ordered(numbered) and unordered(bulleted) lists are each made up of sets of list
items. This tag is used to write list items

1. unordered list,

<ul type=”disc” | “square” | ”circle” > …..
This tag is used for basic unordered list which uses a bullet in front of each tag, every
thing between the tag is encapsulated within <li> tags.

2. ordered list,

<ol type=”1” | ”a” | “I” start=”n”>…..
This tag is used for unordered list which uses a number in front of each list item or it uses any element which is mentioned in the type attribute of the <ol> tag, start attribute is used for indicating the starting number of the list.

3. definition list.

<dl>….. </dl>
This tag is used for the third category i.e., definition list, where numbers or bullet is not used in front of the list item, instead it uses definition for the items.
<dt>…..</dt>
This is a sub tag of the <dl> tag called as definition term, which is used for marking the items whose definition is provided in the next data definition.
<dd> ….</dd>
This is a sub tag of the <dd> tag, definition of the terms are enclosed within these tags. The definition may include any text or block.

Tables:

Table is one of the most useful HTML constructs. Tables are find all over the web application. The main use of table is that they are used to structure the pieces of information and to structure the whole web page. Below are some of the tags used in table.

Links

The HTML code for a link is simple. It looks like this: 19
Link text

JAVASCRIPT

demonstrates on client side validations using JavaScript. It focuses on how to handle events, exceptions, etc. This unit also focuses on DHTML concepts.

CONTENTS:

1) JavaScript concepts
2) Objects in java scripts
3) DHTML with JavaScript

JavaScript (also called JScript) is a scripting language with the primary aim of giving life to our web pages.It is very powerful, flexible, and easy to learn.

Features

- Imperative and structured
- Dynamic
  - dynamic typing
  - object based
  - run-time evaluation
- Functional
  - first-class functions
  - nested functions
  - closures
- Prototype-based
- Miscellaneous
- Vendor-specific extensions

There are three ways by which we can place Javascript for use in a web page.
1. Inside the head section.
2. Within the body section.
3. In an external file.

Events

The Date object

The Date class is used to store and retrieve dates in JavaScript.

Array

The Array object is used to holding a set of data or values in a single variable name.
var urArray=new Array()

DOM (document object model)

A DOM (document object model) is an application programming interface (API) for representing a document (such as an HTML document) and accessing and manipulating the various elements (such as HTML tags and strings of text) that make up that document. Java Script-enabled web browsers have always defined a document object model; a web-browser DOM may specify, for example: that the forms in an HTML document are accessible through the forms[] array of the Document object.

form validation

Form validation is the process of checking that a form has been filled in correctly or not before it is processed.
There are two methods for the form validation.
1: Client-Side validation
In Java Script Client-side form validation is an important part of a web site where data needs to be collected from the user. Users are innately ignorant, and will mess up data entry in a web form if given the chance. It is the job of the web programmer, then, to make sure his pages which use forms include client-side form validation using JavaScript.
2: Server-side validation
In Java Script the server also benefits from client-side validation since it saves a number of round-trips between the visitor and the server owing to typos and easily spotted mistakes. This advantage does not alleviate the neccessity of doing server-side validation.

Unit 2

Overview :
This unit focuses on creating XML documents that are designed to carry data. XML is all about **describing** information. XML was designed to transport and store data. We can create web documents from XML using XSLT to transform our documents into HTML. We can then send our XML to an XSLT processor on the web server and serve that result to the web browser. This makes our documentation available in whatever format we need it to be in.
Contents :
Introduction to XML
DTD
XML Schema
XSLT
DOM
SAX
Introduction to XML
XML describes and focuses on the data while HTML only displays and focuses on how data looks. HTML is all about displaying information but XML is all about describing information. The tags used to mark up HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tags that are defined in the HTML standard.
In HTML some elements can be improperly nested within each other like this:
This text is bold and italic
In XML all elements must be properly nested within each other like this:
An XML document is composed of
1. Declarations (prolog, dtd reference)

2. Elements

3. Comments

4. Entities (predefined, custom defined, character entities)

The XML declaration: Always the first line in the xml document:
The XML declaration should always be included. It defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.
<?xml version="1.0" encoding="ISO-8859-1"?>
Root Element: The next line defines the first element of the document . It is called as the root element
<E-mail>
Child Elements: The next 4 lines describe the four child elements of the root (To, From, Subject and Body). And finally the last line defines the end of the root element .
</E-mail>
</E-mail>
syntax-rules
- All XML elements must have a closing tag
- XML tags are case sensitive
- XML Elements Must be Properly Nested
- XML Documents Must Have a Root Element
- Always Quote the XML Attribute Values
- With XML, White Space is Preserved

 Comments in XML <!-- This is a comment -->
 XML Elements have Relationships
- Elements in a xml document are related as parents and children.

XML elements must follow these naming conventions:

Names must not start with a number or punctuation character but it can contain letters, numbers, and other characters without spaces. Names must not start with the letters xml (or XML, or Xml, etc)

XML Attributes

 XML elements can have attributes in the start tag, just like HTML.

 Attributes are used to provide additional information about elements.

 Attribute values must always be enclosed in quotes. Use either single or double quotes eg. <color="red"> or <color='red'>

 If the attribute value itself contains double quotes it is necessary to use single quotes, like in this example: <name='Rose "India" Net'>

 : If the attribute value itself contains single quotes it is necessary to use double quotes, like in this example: <name="Rose 'India' Net">

DTD(Document Type Definition**)**

A Document Type Definition (DTD) defines the legal building blocks of an XML document.It defines the document structure with a list of legal elements and attributes. A DTD can be defined inside a XML document, or a external reference can be declared .

Internal DTD If the DTD is defined inside the XML document, it should be wrapped in a DOCTYPE definition with the following syntax:

<!DOCTYPE root-element [element-declarations]>

External DTD

If the DTD is defined in an external file, it should be wrapped in a DOCTYPE definition with the following syntax:

<!DOCTYPE root-element SYSTEM "filename">

Importance of a DTD

 With a DTD, a XML file carries a description of its own format.

 With a DTD, independent groups of people can agree to use a standard DTD for interchanging data.

 User application can use a standard DTD to verify that the data he receives from the outside world is valid.

 User can also use a DTD to verify his own data.

Building blocks of XML DTD Documents:

 Elements
 Attributes
 Entities
 PCDATA
 CDATA

PCDATA

 PCDATA means **parsed character data.** It can be thought as the character data (text) found between the start tag and the end tag of a XML element.

 PCDATA is a text to be parsed by a parser. **The text is checked by the parser for entities and markup**.

 Tags inside the text will be treated as markup and entities will be expanded. However, parsed character data should not contain any **&, <,** or **>** characters. These should be represented by the **&amp** , **&lt,** and **&gt** entities, respectively.

CDATA:
 CDATA is character data that will NOT be parsed by a parser. Tags inside the text will NOT be treated as markup and entities will not be expanded.

DTD-Elements: Elements are the **main constituent components** of both XML documents. Elements can contain text, other elements, or be empty.
Syntax:
<!ELEMENT element-name category> or
<!ELEMENT element-name (element-content)>
EX: **Elements with Parsed Character Data**
<!ELEMENT To (#PCDATA)> <!ELEMENT From (#PCDATA)>
Elements with Children (sequences)
Elements with one or more children are declared with the name of the children elements inside the parentheses as :
<!ELEMENT element-name (child1)>
or
<!ELEMENT element-name (child1,child2,...)>
EX:
<!ELEMENT E-mail (To,From,Subject,Body)>
When children are declared in a sequence separated by commas, the children must appear in the same sequence in the document.
In a full declaration, the children must also be declared.
Children can have children.
Tag qualifiers
* Indicates zero or more occurrence.
<!ELEMENT color (Fill-Red*)>
? Indicates Zero or one time occurrence.
<!ELEMENT color (Fill-Red?)>
+ Indicates one or more occurrence
<!ELEMENT color (Fill-Red+)>
( ) Indicates a group of expressions to be matched together.
EX:<!ELEMENT E-mail(#PCDATA|To|From|Subject|Body)*>
| Indicates an option.
<!ELEMENT E-mail (To,From,Subject,(Message|Body))>
Special tag Values in DTD
• Tag definition can have following instead of sub-tags:
• ANY
- Indicates that the tag can contain any other defined element or PCDATA.
- Usually used for the root element.
- Elements can occur in any order in such a document.
- Not recommended to be used.

• EMPTY
- It says that the element contains no contents (and consequently no corresponding end-tag)
- Ex: to allow a tag flag used as <flag/> in a xml file the DTD entry should be

<!ELEMENT flag EMPTY>
DTD-Attributes:
Attributes provide **extra information about elements**.

In a DTD, attributes are declared with an ATTLIST declaration.
Declaring Attributes
The ATTLIST declaration defines the element having a attribute with attribute name , attribute type , and attribute default value. An attribute declaration has the following syntax:
<!ATTLIST element-name attribute-name attribute-type default-value>
DTD example:
<!ATTLIST reciept type CDATA "check">
XML example:
<reciept type="check" />
DTD-Entities
Entities are variables used to define shortcuts to standard text or special characters. Entity references are references to entities Entities can be declared internally or externally.
Internal Entity Declaration
Syntax:
<!ENTITY entity-name "entity-value">
XML Schema
XML Schemas are more powerful than DTDs.
XML Schema is a W3C Standard. It is an XML-based alternative to DTDs.
It describes the structure of an XML document.
The XML Schema language is also referred to as XML Schema Definition (XSD).
We think that very soon XML Schemas will be used in most Web applications as a replacement for DTDs. Here are some reasons:
- XML Schemas are extensible to future additions
- XML Schemas are richer and more powerful than DTDs
- XML Schemas are written in XML, supports data types and namespaces.

What is an XML Schema?
- XML Schema is used to define the legal building blocks of an XML document, just like a DTD.
- An XML Schema defines user-defined integrants like elements, sub-elements and attributes needed in a xml document.
- It defines the data types for elements and attributes along with the occurrence order .
- It defines whether an element is empty or can include text.
- It also defines default and fixed values for elements and attributes

Features of XML Schemas :
XML Schemas Support Data Types
One of the greatest strengths of XML Schemas is its support for data types. With support for data types:
- It is easier to describe allowable document content
- It is easier to validate the correctness of data
- It is easier to work with data from a database
- It is easier to define data facets (restrictions on data)
- It is easier to define data patterns (data formats)
- It is easier to convert data between different data types

XML Schemas use XML Syntax
Another great strength about XML Schemas is that they are written in XML. Simple XML editors are used to edit the Schema files. Even the same XML parsers can be used to parse the Schema files.
XML Schemas are Extensible

XML Schemas are extensible, because they are written in XML.So a user can reuse a Schema in other Schemas and can also refer multiple schemas in the same document. He can also create his own data types derived from the standard types

XML Schemas Secure Reliable Data Communication

When sending data from a sender to a receiver, it is essential that both parts have the same "expectations" about the content. With XML Schemas, the sender can describe the data in a way that the receiver will understand. A date like: "03-11-2004" will, in some countries, be interpreted as 3.November and in other countries as 11.March.However, an XML element with a data type like this: <datetype="date">2004-03-11</date> ensures a mutual understanding of the content, because the XML data type "date" requires the format "YYYY-MM-DD".

XSLT

XSLT (**Extensible Stylesheet Language Transformations**) is a declarative, XML-based language used for the transformation of XML documents. The original document is not changed; rather, a new document is created based on the content of an existing one.[2] The new document may be serialized (output) by the processor in standard XML syntax or in another format, such as HTML or plain text.[3] XSLT is most often used to convert data between different XML schemas or to convert XML data into web pages or PDF documents.

Simple API for XML (SAX)

SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as callbacks to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

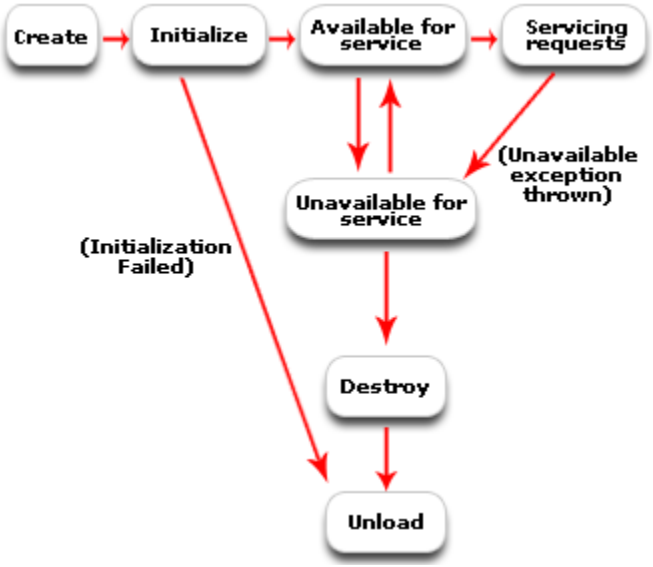
Document Object Model (DOM)

DOM (Document Object Model) is an interface-oriented Application Programming Interface that allows for navigation of the entire document as if it were a tree of "Node" objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM Nodes are abstract; implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed.

Unit 3

Overview :
This unit describes how to generate dynamic content in webpages using servlets and procedure for installing web servers procedure for creating web applications.
Contents :
Introduction to servlets
Lifecycle of a servlet
JSDK server
Servlet API
Session tracking
Security issues
Introduction to servlets:
Servlets are java programs that run on web or application servers, acting as middle layer between request coming from the web browsers or other HTTP clients and database servers. Servlets process user request, produce the results and sends the results as a response to the user
Advantages of Java Servlets
1. Portability
2. Powerful
3. Efficiency
4. Safety
5. Extensibilty
6. Inexpensive

Life cycle of Servlet



Loading: The servlet container loads the servlet during startup or when the first request is made. The loading of the servlet depends on the attribute <load-on-startup> of web.xml file. If the attribute <load-on-startup> has a positive value then the servlet is load with loading of the container otherwise it load when the first request comes for service. After loading of the servlet, the container creates the instances of the servlet.

 Initialization: After creating the instances, the servlet container calls the init() method and passes the servlet initialization parameters to the init() method. The init() must be called by the

servlet container before the servlet can service any request. The initialization parameters persist untill the servlet is destroyed. The init() method is called only once throughout the life cycle of the servlet. The servlet will be available for service if it is loaded successfully otherwise the servlet container unloads the servlet.

 Servicing the Request: After successfully completing the initialization process, the servlet will be available for service. Servlet creates seperate threads for each request. The sevlet container calls the service() method for servicing any request. The service() method determines the kind of request and calls the appropriate method (doGet() or doPost()) for handling the request and sends response to the client using the methods of the response object.

 Destroying the Servlet: If the servlet is no longer needed for servicing any request, the servlet container calls the destroy() method . Like the init() method this method is also called only once throughout the life cycle of the servlet. Calling the destroy() method indicates to the servlet container not to sent the any request for service and the servlet releases all the resources associated with it. Java Virtual Machine claims for the memory associated with the resources for garbage collection.

Servlet API Provides the following two packages

 Javax.servlet

The javax.servlet package contains a number of classes and interfaces that describe and define the contracts between a servlet class and the runtime environment provided for an instance of such a class by a conforming servlet container.

 Javax.servlet.http

The javax.servlet.http package contains a number of classes and interfaces that describe and define the contracts between a servlet class running under the HTTP protocol and the runtime environment provided for an instance of such a class by a conforming servlet container.

1) javax.servlet.Servlet interface

 A servlet is a small Java program that runs within a Web server. Servlets receive and respond to requests from Web clients, usually across HTTP, the HyperText Transfer Protocol.

 To implement this interface, you can write a generic servlet that extends `javax.servlet.GenericServlet` or an HTTP servlet that extends `javax.servlet.http.HttpServlet`.

 This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server.

 In addition to the life-cycle methods(init(),service(),destroy()), this interface provides the `getServletConfig` method, which the servlet can use to get any startup information, and the `getServletInfo` method, which allows the servlet to return basic information about itself, such as author, version, and copyright.

Methods:

 init

public void **init**(ServletConfig config)
throws ServletException
Called by the servlet container to indicate to a servlet that the servlet is being placed into service. The servlet container calls the `init` method exactly once after instantiating the servlet. The `init` method must complete successfully before the servlet can receive any requests.
The servlet container cannot place the servlet into service if the `init` method
1. Throws a ServletException
2. Does not return within a time period defined by the Web server

Parameters:

config - a ServletConfig object containing the servlet's configuration and initialization parameters
Throws:
ServletException - if an exception has occurred that interferes with the servlet's normal operation
getServletConfig
public ServletConfig **getServletConfig**()
Returns a ServletConfig object, which contains initialization and startup parameters for this servlet. The ServletConfig object returned is the one passed to the init method.
 service
public void **service**(ServletRequest req,ServletResponse res) throws ServletException, java.io.IOException
Called by the servlet container to allow the servlet to respond to a request.
This method is only called after the servlet's `init()` method has completed successfully.
Parameters:
req - the ServletRequest object that contains the client's request
res - the ServletResponse object that contains the servlet's response
Throws:
ServletException - if an exception occurs that interferes with the servlet's normal operation
java.io.IOException - if an input or output exception occurs
 getServletInfo
public java.lang.String **getServletInfo**()
Returns information about the servlet, such as author, version, and copyright.

destroy
public void **destroy**()
Called by the servlet container to indicate to a servlet that the servlet is being taken out of service. This method is only called once all threads within the servlet's service method have exited or after a timeout period has passed. After the servlet container calls this method, it will not call the service method again on this servlet.
This method gives the servlet an opportunity to clean up any resources that are being held (for example, memory, file handles, threads) and make sure that any persistent state is synchronized with the servlet's current state in memory.
2) javax.servlet.ServletConfig Interface
A servlet configuration object used by a servlet container to pass information to a servlet during initialization.
Methods:
 getServletName
public java.lang.String **getServletName**()
Returns the name of this servlet instance. The name may be provided via server administration, assigned in the web application deployment descriptor, or for an unregistered (and thus unnamed) servlet instance it will be the servlet's class name.
 getServletContext
public ServletContext **getServletContext**()
Returns a reference to the ServletContext in which the caller is executing.
Returns:
a ServletContext object, used by the caller to interact with its servlet container
 getInitParameter
public java.lang.String **getInitParameter**(java.lang.String name)
Returns a String containing the value of the named initialization parameter, or null if the parameter does not exist.
Parameters:
name - a String specifying the name of the initialization parameter
Returns:

a String containing the value of the initialization parameter

- **getInitParameterNames**

public java.util.Enumeration **getInitParameterNames**()
Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.

Returns: an Enumeration of String objects containing the names of the servlet's initialization parameters.

3) javax.servlet.ServletContext Interface

Defines a set of methods that a servlet uses to communicate with its servlet container, for example, to get the MIME type of a file, dispatch requests, or write to a log file
There is one context per "web application" per Java Virtual Machine
The ServletContext object is contained within the ServletConfig object.

Mehods:

java.lang.String **getInitParameter**(java.lang.String name) Returns a String containing the value of the named context-wide initialization parameter, or null if the parameter does not exist.
java.util.Enumeration **getInitParameterNames**() Returns the names of the context's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the context has no initialization parameters.
java.lang.String **getMimeType**(java.lang.String file) Returns the MIME type of the specified file, or null if the MIME type is not known.
RequestDispatcher **getNamedDispatcher**(java.lang.String name) Returns a RequestDispatcher object that acts as a wrapper for the named servlet.
RequestDispatcher **getRequestDispatcher**(java.lang.String path) Returns a RequestDispatcher object that acts as a wrapper for the resource located at the given path
java.lang.String **getServerInfo**() Returns the name and version of the servlet container on which the servlet is running.
java.lang.String **getServletContextName**() Returns the name of this web application corresponding to this ServletContext as specified in the deployment descriptor for this web application by the display-name element.
Void **setAttribute**(java.lang.String name, java.lang.Object object) Binds an object to a given attribute name in this servlet context.
java.lang.Object **getAttribute**(java.lang.String name) Returns the servlet container attribute with the given name, or null if there is no attribute by that name.

4) javax.servlet. RequestDispatcher Interface

Defines an object that receives requests from the client and sends them to any resource (such as a servlet, HTML file, or JSP file) on the server
This interface is intended to wrap servlets, but a servlet container can create RequestDispatcher objects to wrap any type of resource

Methods:

- **forward**

public void **forward**(ServletRequest request,
ServletResponse response)
throws ServletException,
java.io.IOException

Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server .

include

public void **include**(ServletRequest request,
ServletResponse response)
throws ServletException,

java.io.IOException
Includes the content of a resource (servlet, JSP page, HTML file) in the response. In essence, this method enables programmatic server-side includes.

5) javax.servlet.ServletRequest Interface

Defines an object to provide client request information to a servlet. The servlet container creates a ServletRequest object and passes it as an argument to the servlet's service method
A ServletRequest object provides data including parameter name and values, attributes, and an input stream.

1) javax.servlet.ServletResponse Interface

Defines an object to assist a servlet in sending a response to the client. The servlet container creates a `ServletResponse` object and passes it as an argument to the servlet's `service` method.
To send binary data in a MIME body response, use the ServletOutputStream returned by getOutputStream().
To send character data, use the PrintWriter object returned by getWriter().

1) javax.servlet.GenericServlet class

Defines a generic, protocol-independent servlet.
`GenericServlet` implements the `Servlet` and `ServletConfig` interfaces.
`GenericServlet` may be directly extended by a servlet, although it's more common to extend a protocol-specific subclass such as `HttpServlet`.

Interfaces and classes of javax.servlet.http package

1. javax.servlet.http. HttpServletRequest Interface

Extends the `ServletRequest` interface to provide request information for HTTP servlets.
The servlet container creates an `HttpServletRequest` object and passes it as an argument to the servlet's service methods (`doGet`, `doPost`, etc).
Methods

- **getCookies**

  public Cookie[] **getCookies**()

Returns an array containing all of the Cookie objects the client sent with this request. This method returns null if no cookies were sent.

- **getSession**

public HttpSession **getSession**()
Returns the current session associated with this request, or if the request does not have a session, creates one.
public HttpSession **getSession**(boolean create)
Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.
If `create` is `false` and the request has no valid `HttpSession`, this method returns `null`.

- **getRequestURI**

public java.lang.String **getRequestURI**()
Returns the part of this request's URL from the protocol name up to the query string in the first line of the HTTP request.
For example POST /some/path.html HTTP/1.1 then it returns /some/path.html

- **getRequestURL**

public java.lang.StringBuffer **getRequestURL**()
Reconstructs the URL the client used to make the request. The returned URL contains a protocol, server name, port number, and server path, but it does not include query string parameters.

- **getServletPath**

public java.lang.String **getServletPath**()

Returns the part of this request's URL that calls the servlet. This path starts with a "/" character and includes either the servlet name or a path to the servlet, but does not include any extra path information or a query string.

- **getContextPath**

public java.lang.String **getContextPath**()
Returns the portion of the request URI that indicates the context of the request. The context path always comes first in a request URI. The path starts with a "/" character but does not end with a "/" character.

- **getHeader**

public java.lang.String **getHeader**(java.lang.String name)
Returns the value of the specified request header as a String.

- **getHeaders**

public java.util.Enumeration **getHeaders**(java.lang.String name) Returns all the values of the specified request header as an Enumeration of String objects.

- **getHeaderNames**

public java.util.Enumeration **getHeaderNames**()
Returns an enumeration of all the header names this request contains. If the request has no headers, this method returns an empty enumeration.

2) javax.servlet.http. HttpServletResponse Interface

Extends the `ServletResponse` interface to provide HTTP-specific functionality in sending a response. For example, it has methods to access HTTP headers and cookies.
The servlet container creates an HttpServletResponse object and passes it as an argument to the servlet's service methods

- **addCookie**

public void **addCookie**(Cookie cookie)
Adds the specified cookie to the response. This method can be called multiple times to set more than one cookie

- **sendError**

public void **sendError**(int sc)
throws java.io.IOException
Sends an error response to the client using the specified status code and clearing the buffer.
If the response has already been committed, this method throws an IllegalStateException

- **sendRedirect**

public void **sendRedirect**(java.lang.String location)
throws java.io.IOException
Sends a temporary redirect response to the client using the specified redirect location URL. This method can accept relative URLs;
location - the redirect location URL

- **setHeader**

public void **setHeader**(java.lang.String name,
java.lang.String value)
Sets a response header with the given name and value. If the header had already been set, the new value overwrites the previous one.

addHeader

public void **addHeader**(java.lang.String name, java.lang.String value)
Adds a response header with the given name and value. This method allows response headers to have multiple values.

- **addIntHeader**

public void **addIntHeader**(java.lang.String name,
int value)

Adds a response header with the given name and integer value. This method allows response headers to have multiple values.

 setStatus

public void **setStatus**(int sc)

Sets the status code for this response. This method is used to set the return status code when there is no error

3) javax.servlet.http. HttpSession Interface

HttpSession provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

A session usually corresponds to one user, who may visit a site many times. The server can maintain a session in many ways such as using cookies or rewriting URLs.

This interface allows servlets to

 View and manipulate information about a session, such as the session identifier, creation time, and last accessed time

 Bind objects to sessions, allowing user information to persist across multiple user connections

Methods:

 getAttribute

public java.lang.Object **getAttribute**(java.lang.String name)

Returns the object bound with the specified name in this session, or null if no object is bound under the name.

 setAttribute

public void **setAttribute**(java.lang.String name,
java.lang.Object value)

Binds an object to this session, using the name specified. If an object of the same name is already bound to the session, the object is replaced.

 getAttributeNames

public java.util.Enumeration **getAttributeNames**()

Returns an Enumeration of String objects containing the names of all the objects bound to this session.

 removeAttribute

public void **removeAttribute**(java.lang.String name)

Removes the object bound with the specified name from this session.

 invalidate

public void **invalidate**()

Invalidates this session then unbinds any objects bound to it.

 isNew

public boolean **isNew**()

returns true if the server has created a session, but the client has not yet joined

 getMaxInactiveInterval

public int **getMaxInactiveInterval**()

Returns the maximum time interval, in seconds, that the servlet container will keep this session open between client accesses.

 setMaxInactiveInterval

public void **setMaxInactiveInterval**(int interval)

Specifies the time, in seconds, between client requests before the servlet container will invalidate this session. A negative time indicates the session should never timeout.

 getLastAccessedTime

public long **getLastAccessedTime**()

Returns the last time the client sent a request associated with this session,

 getId

public java.lang.String **getId**()

Returns a string containing the unique identifier assigned to this session.

 getCreationTime

public long **getCreationTime**()

Returns the time when this session was created

Session Tracking:

Http is a *stateless* protocol, means that it can't persist the information. It always treats each request as a new request. In Http client makes a connection to the server, sends the request., gets the response, and closes the connection.

In session management client first make a request for any servlet or any page, the container receives the request and generate a unique session ID and gives it back to the client along with the response. This ID gets stores on the client machine. Thereafter when the client request again sends a request to the server then it also sends the session Id with the request. There the container sees the Id and sends back the request.

Session Tracking can be done in Four ways:

1. Hidden Form Fields:

2. Cookies

3. HttpSession

4. **URL Rewriting**

UNIT – IV

Overview :

In this unit we focuses on JSP Technologies. JavaServer Pages (JSP) is a Java technology that helps software developers serve dynamically generated web pages based on HTML, XML, or other document types. JSP pages are loaded in the server and are operated from a structured special installed Java server packet called a Java EE Web Application, often packaged as a `.war` or `.ear` file archive. JSP allows Java code and certain pre-defined actions to be interleaved with static web markup content, with the resulting page being compiled and executed on the server to deliver an HTML or XML document. The compiled pages and any dependent Java libraries use Java bytecode rather than a native software format, and must therefore be executed within a Java virtual machine (JVM) that integrates with the host operating system to provide an abstract platform-neutral environment.

Contents

Problems with servlets

The Anatomy of JSP Page

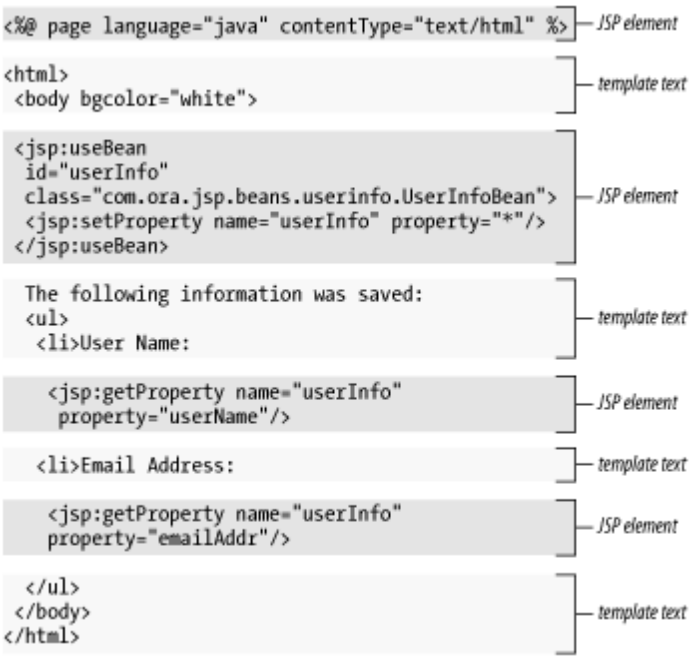
JSP Processing

MVC

JSDK

The Anatomy of a JSP Page

A JSP page is simply a regular web page with JSP elements for generating the parts that differ for each request,

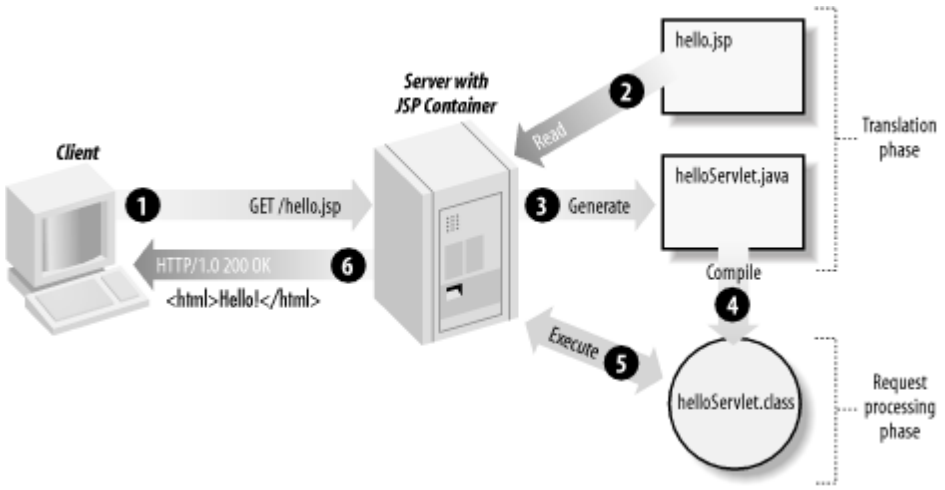


Everything in the page that isn't a JSP element is called *template text*. Template text can be any text: HTML, WML, XML, or even plain text. Since HTML is by far the most common web-page language in use today, most of the descriptions and examples in this book use HTML, but keep in mind that JSP has no dependency on HTML; it can be used with any markup language. Template text is always passed straight through to the browser.
When a JSP page request is processed, the template text and dynamic content generated by the JSP elements are merged, and the result is sent as the response to the browser.

JSP Processing

Just as a web server needs a servlet container to provide an interface to servlets, the server needs a *JSP container* to process JSP pages.
The JSP container is responsible for intercepting requests for JSP pages. To process all JSP elements in the page, the container first turns the JSP page into a servlet (known as the *JSP page implementation class*).
The conversion is pretty straightforward; all template text is converted to `println( )` statements and all JSP elements are converted to Java code that implements the corresponding dynamic behavior. The container then compiles the servlet class.

Converting the JSP page to a servlet and compiling the servlet form the *translation phase*.
The JSP container initiates the translation phase for a page automatically when it receives the first request for the page. Since the translation phase takes a bit of time, the first user to request a JSP page notices a slight delay.
The translation phase can also be initiated explicitly; this is referred to as *precompilation* of a JSP page. Precompiling a JSP page is a way to avoid hitting the first user with this delay.
The JSP container is also responsible for invoking the JSP page implementation class (the generated servlet) to process each request and generate the response. This is called the *request processing phase*.
JSP page translation and processing phases



As long as the JSP page remains unchanged, any subsequent request goes straight to the request processing phase (i.e., the container simply executes the class file). When the JSP page is modified, it goes through the translation phase again before entering the request processing phase. The JSP container is often implemented as a servlet configured to handle all requests for JSP pages. In fact, these two containers--a servlet container and a JSP container--are often combined in one package under the name *web container*. a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet: it's loaded once and called repeatedly, until the server is shut down. By virtue of being an automatically generated servlet, a JSP page inherits all the advantages of a servlet: platform and vendor independence, integration, efficiency, scalability, robustness, and security.

JSP Application Design with MVC

. The key point of using MVC is to separate logic into three distinct units: the Model, the View

As long as the JSP page remains unchanged, any subsequent request goes straight to the request processing phase (i.e., the container simply executes the class file). When the JSP page is modified, it goes through the translation phase again before entering the request processing phase. The JSP container is often implemented as a servlet configured to handle all requests for JSP pages. In fact, these two containers--a servlet container and a JSP container--are often combined in one package under the name *web container*. a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet: it's loaded once and called repeatedly, until the server is shut down. By virtue of being an automatically generated servlet, a JSP page inherits all the advantages of a servlet: platform and vendor independence, integration, efficiency, scalability, robustness, and security.

JSP Application Design with MVC

. The key point of using MVC is to separate logic into three distinct units: the Model, the View and the Controller.

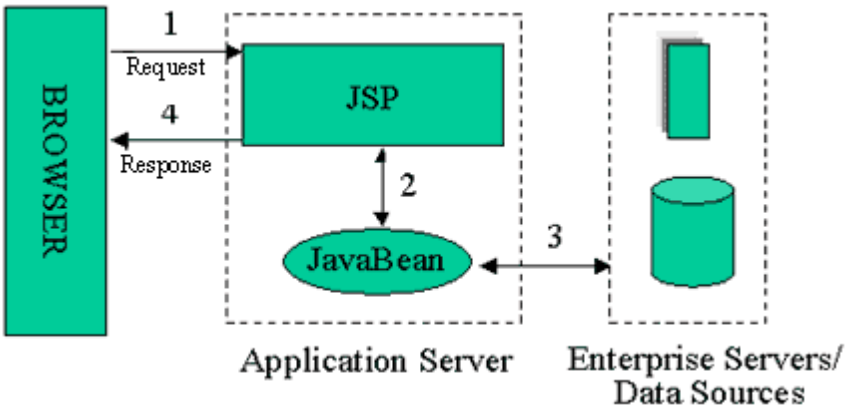
In a server application, we commonly classify the parts of the application as business logic, presentation, and request processing.

Business logic is the term used for the manipulation of an application's data, such as customer, product, and order information.

Presentation refers to how the application data is displayed to the user, for example, position, font, and size.

And finally, *request processing* is what ties the business logic and presentation parts together.

JSP Model 1 Architecture



In MVC terms, the Model corresponds to business logic and data, the View to the presentation, and the Controller to the request processing.
Why use this design with JSP? The answer lies primarily in the first two elements. Remember that an application data structure and logic (the Model) is typically the most stable part of an application, while the presentation of that data (the View) changes fairly often. Just look at all the face-lifts many web sites go through to keep up with the latest fashion in web design. Yet, the data they present remains the same.
Another common example of why presentation should be separated from the business logic is that you may want to present the data in different languages or present different subsets of the data to internal and external users.

Access to the data through new types of devices, such as cell phones and personal digital assistants (PDAs), is the latest trend. Each client type requires its own presentation format. It should come as no surprise, then, that separating business logic from the presentation makes it easier to evolve an application as the requirements change; new presentation interfaces can be developed without touching the business logic.

JSDK
1 Set classpath to <home dir>/jsdk2.0/lib/jsdk.jar;
2. Set path to <home dir>/jsdk2.0/bin;
3. Compile the servlet program (.java file).
4. Copy the .class file to <home dir>/jsdk2.0/examples
5. Execute servletrunner
6. Open web browser
7. Enter the url as follows in the address bar: http://host:8080/servlet/<servlet name>

JSP Elements

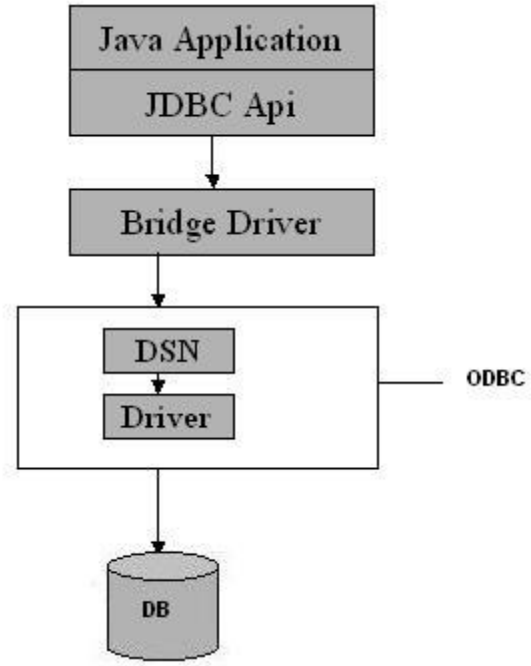
There are three types of JSP elements you can use: *scripting*, *action* and *directive*.

Scripting elements

Scripting elements allow you to add small pieces of code (typically Java code) in a JSP page, such as an `if` statement to generate different HTML depending on a certain condition. Like actions, they are also executed when the page is requested. You should use scripting elements with extreme care: if you embed too much code in your JSP pages, you will end up with the same kind of maintenance problems as with servlets embedding HTML.

Scripting elements **Element**	**Description**
`<% ... %>`	Scriptlet, used to embed scripting code.
`<%= ... %>`	Expression, used to embed scripting code expressions when the result shall be added to the response. Also used as request-time action attribute values.
`<%! ... %>`	Declaration, used to declare instance variables and methods in the JSP page implementation class.

JDBC Driver Types



Type 1: JDBC-ODBC Bridge driver (Bridge)
Type 2: Native-API/partly Java driver (Native)
Type 3: AllJava/Net-protocol driver (Middleware)
Type 4: All Java/Native-protocol driver (Pure)

Type 1 JDBC Driver
JDBC-ODBC Bridge driver

The Type 1 driver translates all JDBC calls into ODBC calls and sends them to the ODBC driver. ODBC is a generic API. The JDBC-ODBC Bridge driver is recommended only for experimental use or when no other alternative is available.

Advantage

The JDBC-ODBC Bridge allows access to almost any database, since the database's ODBC drivers are already available.

Disadvantages

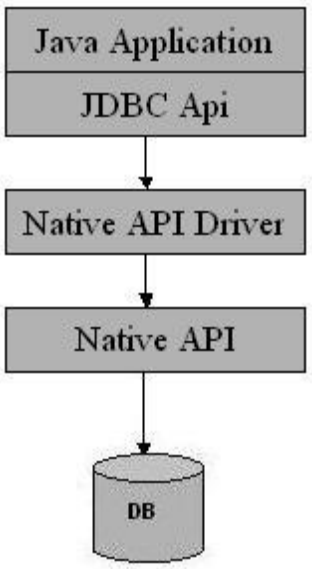
1. Since the Bridge driver is not written fully in Java, Type 1 drivers are not portable. 2. A performance issue is seen as a JDBC call goes through the bridge to the ODBC driver, then to the database, and this applies even in the reverse process. They are the slowest of all driver types. 3. The client system requires the ODBC Installation to use the driver. 4. Not good for the Web.

Type 2 JDBC Driver

Native-API/partly Java driver

The distinctive characteristic of type 2 jdbc drivers are that Type 2 drivers convert JDBC calls into database-specific calls i.e. this driver is specific to a particular database. Some

distinctive characteristic of type 2 jdbc drivers are shown below. Example: Oracle will have oracle native api.



Advantage

The distinctive characteristic of type 2 jdbc drivers are that they are typically offer better performance than the JDBC-ODBC Bridge as the layers of communication (tiers) are less than that of Type 1 and also it uses Native api which is Database specific.

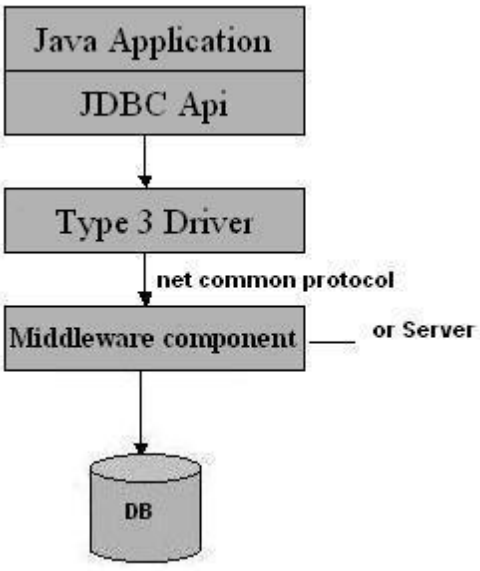
Disadvantage

1. Native API must be installed in the Client System and hence type 2 drivers cannot be used for the Internet. 2. Like Type 1 drivers, it's not written in Java Language which forms a portability issue. 3. If we change the Database we have to change the native api as it is specific to a database 4. Mostly obsolete now 5. Usually not thread safe.

Type 3 JDBC Driver

All Java/Net-protocol driver

Type 3 database requests are passed through the network to the middle-tier server. The middle-tier then translates the request to the database. If the middle-tier server can in turn use Type1, Type 2 or Type 4 drivers.

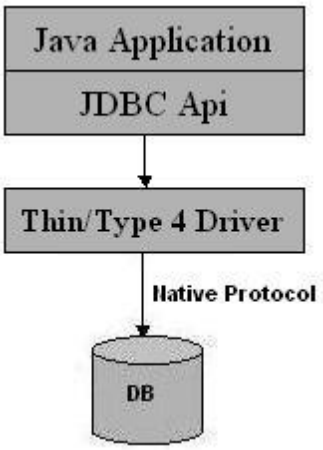


Advantage
1. This driver is server-based, so there is no need for any vendor database library to be present on client machines. 2. This driver is fully written in Java and hence Portable. It is suitable for the web. 3. There are many opportunities to optimize portability, performance, and scalability. 4. The net protocol can be designed to make the client JDBC driver very small and fast to load. 5. The type 3 driver typically provides support for features such as caching (connections, query results, and so on), load balancing, and advanced system administration such as logging and auditing. 6. This driver is very flexible allows access to multiple databases using one driver. 7. They are the most efficient amongst all driver types. **Disadvantage**
It requires another server application to install and maintain. Traversing the recordset may take longer, since the data comes through the backend server.

Type 4 JDBC Driver

Native-protocol/all-Java driver

The Type 4 uses java networking libraries to communicate directly with the database server.



Advantage
1. The major benefit of using a type 4 jdbc drivers are that they are completely written in Java to achieve platform independence and eliminate deployment administration issues. It is most suitable for the web. 2. Number of translation layers is very less i.e. type 4 JDBC drivers don't have to translate database requests to ODBC or a native connectivity interface or to pass the request on to another server, performance is typically quite good. 3. You don’t need to install special software on the client or server. Further, these drivers can be downloaded dynamically.

Disadvantage With type 4 drivers, the user needs a different driver for each database.

List of Drivers
• Bridge driver
• sun.jdbc.odbc.JdbcOdbcDriver
 jdbc:odbc:<dsn>
• Cloudscape
• COM.cloudscape.core.JDBCDriver
 jdbc:cloudscape:[database name and location]
• PostGRESQL

• org.postgresql.Driver
 jdbc:postgresql://[host]:[port]/[database name]
• MySQL
• com.mysql.jdbc.Driver
 jdbc:mysql://[host]:3306/[databasename]
• Oracle
• oracle.jdbc.driver.OracleDriver

jdbc:oracle:thin:@[host]:1521:[sid]

Steps to using Bridge driver

1. Create a data source name using ODBC
2. Load the database driver
3. Establish a Connection to the database
4. Create a Statement object
5. Execute SQL Query statement(s)
6. Retrieve the ResultSet Object
7. Retrieve record/field data from

8. object for processing
9. Close ResultSet Object
10. Close Statement Object
11. Close Connection Object

javax.sql.RowSet

The interface that adds support to the JDBC API for the JavaBeans™ component model. A rowset, which can be used as a JavaBeans component in a visual Bean development environment, can be created and configured at design time and executed at run time.

The RowSet interface provides a set of JavaBeans properties that allow a RowSet instance to be configured to connect to a JDBC data source and read some data from the data source. A group of setter methods (setInt, setBytes, setString, and so on) provide a way to pass input parameters to a rowset's command property. This command is the SQL query the rowset uses when it gets its data from a relational database, which is generally the case.

The RowSet interface supports JavaBeans events, allowing other components in an application to be notified when an event occurs on a rowset, such as a change in its value.

javax.sql.DataSource

A factory for connections to the physical data source that this DataSource object represents. An alternative to the DriverManager facility, a DataSource object is the preferred means of getting a connection. An object that implements the DataSource interface will typically be registered with a naming service based on the Java™ Naming and Directory (JNDI) API.

The DataSource interface is implemented by a driver vendor. There are three types of implementations:

1. Basic implementation -- produces a standard Connection object
2. Connection pooling implementation -- produces a Connection object that will automatically participate in connection pooling. This implementation works with a middle-tier connection pooling manager.
3. Distributed transaction implementation -- produces a Connection object that may be used for distributed transactions and almost always participates in connection pooling. This implementation works with a middle-tier transaction manager and almost always with a connection pooling manager.

A DataSource object has properties that can be modified when necessary. For example, if the data source is moved to a different server, the property for the server can be changed. The benefit is that because the data source's properties can be changed, any code accessing that data source does not need to be changed.

A driver that is accessed via a DataSource object does not register itself with the DriverManager. Rather, a DataSource object is retrieved though a lookup operation and then used to create a Connection object. With a basic implementation, the connection obtained through a DataSource object is identical to a connection obtained through the DriverManager facility.

Specific database actions

<sql: transaction>
<sql: update>
UPDATE Account SET Balance = Balance - 1000
WHERE AccountNumber = 1234
</sql: update>
<sql: update>
UPDATE Account SET Balance = Balance + 1000
WHERE AccountNumber = 5678
</sql: update>
</sql: transaction>

All SQL actions that make up a transaction are placed in the body of a <sql:transaction> action element. This action tells the nested elements which database to use, so if you need to specify the database with the dataSource attribute, you must specify it for the <sql:transaction> action. The isolation attribute can specify special transaction features. When the Data Source is made available to the application through JNDI or by another application component, it's typically already configured with an appropriate isolation level. This attribute is therefore rarely used. The details of the different isolation levels are beyond the scope of this book. If you believe you need to specify this value, you can read up on the differences in the JDBC API documents or in the documentation for your database. You should also be aware that some databases and JDBC drivers don't support all transaction isolation levels.

UNIT V

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a recursive acronym for "PHP: Hypertext Preprocessor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.

- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

Common uses of PHP

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

Characteristics of PHP

Five important characteristics make PHP's practical nature possible −

- Simplicity
- Efficiency
- Security
- Flexibility
- Familiarity

"Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this −

Live Demo

```
<html>

   <head>
      <title>Hello World</title>
   </head>

   <body>
      <?php echo "Hello, World!";?>
   </body>

</html>
```

It will produce following result −

```
Hello, World!
```

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognised by the PHP Parser.

```
<?php PHP code goes here ?>

<?    PHP code goes here ?>

<script language = "php"> PHP code goes here </script>
```

A most common tag is the <?php...?> and we will also use the same tag in our tutorial.

PHP is whitespace insensitive

Whitespace is the stuff you type that is typically invisible on the screen, including spaces, tabs, and carriage returns (end-of-line characters).

PHP whitespace insensitive means that it almost never matters how many whitespace characters you have in a row.one whitespace character is the same as many such characters.

For example, each of the following PHP statements that assigns the sum of 2 + 2 to the variable $four is equivalent −

```
$four = 2 + 2; // single spaces
$four <tab>=<tab2<tab>+<tab>2 ; // spaces and tabs
$four =
2+
2; // multiple lines
```

PHP is case sensitive

Yeah it is true that PHP is a case sensitive language. Try out following example −

Live Demo

```
<html>
   <body>

      <?php
         $capital = 67;
         print("Variable capital is $capital<br>");
         print("Variable CaPiTaL is $CaPiTaL<br>");
      ?>

   </body>
</html>
```

This will produce the following result −

```
Variable capital is 67
```

```
Variable CaPiTaL is
```

Statements are expressions terminated by semicolons

A *statement* in PHP is any expression that is followed by a semicolon (;).Any sequence of valid PHP statements that is enclosed by the PHP tags is a valid PHP program. Here is a typical statement in PHP, which in this case assigns a string of characters to a variable called $greeting −

```
$greeting = "Welcome to PHP!";
```

Expressions are combinations of tokens

The smallest building blocks of PHP are the indivisible tokens, such as numbers (3.14159), strings (.two.), variables ($two), constants (TRUE), and the special words that make up the syntax of PHP itself like if, else, while, for and so forth

Braces make blocks

Although statements cannot be combined like expressions, you can always put a sequence of statements anywhere a statement can go by enclosing them in a set of curly braces.

Here both statements are equivalent −

```
if (3 == 2 + 1)
   print("Good - I haven't totally lost my mind.<br>");

if (3 == 2 + 1) {
   print("Good - I haven't totally");
   print("lost my mind.<br>");
}
```

Running PHP Script from Command Prompt

Yes you can run your PHP script on your command prompt. Assuming you have following content in test.php file

Live Demo

```
<?php
   echo "Hello PHP!!!!!";
?>
```

Now run this script as command prompt as follows −

```
$ php test.php
```

It will produce the following result −

```
Hello PHP!!!!!
```

Hope now you have basic knowledge of PHP Syntax.

PHP - Variable Types

The main way to store information in the middle of a PHP program is by using a variable.

Here are the most important things to know about variables in PHP.

- All variables in PHP are denoted with a leading dollar sign ($).
- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables in PHP do not have intrinsic types - a variable does not know in advance whether it will be used to store a number or a string of characters.
- Variables used before they are assigned have default values.
- PHP does a good job of automatically converting types from one to another when necessary.
- PHP variables are Perl-like.

PHP has a total of eight data types which we use to construct our variables −

- **Integers** − are whole numbers, without a decimal point, like 4195.
- **Doubles** − are floating-point numbers, like 3.14159 or 49.1.
- **Booleans** − have only two possible values either true or false.
- **NULL** − is a special type that only has one value: NULL.
- **Strings** − are sequences of characters, like 'PHP supports string operations.'
- **Arrays** − are named and indexed collections of other values.
- **Objects** − are instances of programmer-defined classes, which can package up both other kinds of values and functions that are specific to the class.
- **Resources** − are special variables that hold references to resources external to PHP (such as database connections).

The first five are *simple types*, and the next two (arrays and objects) are compound - the compound types can package up other arbitrary values of arbitrary type, whereas the simple types cannot.

We will explain only simple data type in this chapters. Array and Objects will be explained separately.

Integers

They are whole numbers, without a decimal point, like 4195. They are the simplest type .they correspond to simple whole numbers, both positive and negative. Integers can be assigned to variables, or they can be used in expressions, like so −

```
$int_var = 12345;
$another_int = -12345 + 12345;
```

Integer can be in decimal (base 10), octal (base 8), and hexadecimal (base 16) format. Decimal format is the default, octal integers are specified with a leading 0, and hexadecimals have a leading 0x.

For most common platforms, the largest integer is (2**31 . 1) (or 2,147,483,647), and the smallest (most negative) integer is . (2**31 . 1) (or .2,147,483,647).

Doubles

They like 3.14159 or 49.1. By default, doubles print with the minimum number of decimal places needed. For example, the code −

Live Demo

```
<?php
   $many = 2.2888800;
   $many_2 = 2.2111200;
   $few = $many + $many_2;

   print("$many + $many_2 = $few <br>");
?>
```

It produces the following browser output −

```
2.28888 + 2.21112 = 4.5
```

Boolean

They have only two possible values either true or false. PHP provides a couple of constants especially for use as Booleans: TRUE and FALSE, which can be used like so −

```
if (TRUE)
   print("This will always print<br>");

else
   print("This will never print<br>");
```

Interpreting other types as Booleans

Here are the rules for determine the "truth" of any value not already of the Boolean type −

- If the value is a number, it is false if exactly equal to zero and true otherwise.
- If the value is a string, it is false if the string is empty (has zero characters) or is the string "0", and is true otherwise.
- Values of type NULL are always false.
- If the value is an array, it is false if it contains no other values, and it is true otherwise. For an object, containing a value means having a member variable that has been assigned a value.
- Valid resources are true (although some functions that return resources when they are successful will return FALSE when unsuccessful).
- Don't use double as Booleans.

Each of the following variables has the truth value embedded in its name when it is used in a Boolean context.

```
$true_num = 3 + 0.14159;
$true_str = "Tried and true"
$true_array[49] = "An array element";
$false_array = array();
$false_null = NULL;
$false_num = 999 - 999;
$false_str = "";
```

NULL

NULL is a special type that only has one value: NULL. To give a variable the NULL value, simply assign it like this −

```
$my_var = NULL;
```

The special constant NULL is capitalized by convention, but actually it is case insensitive; you could just as well have typed −

```
$my_var = null;
```

A variable that has been assigned NULL has the following properties −

- It evaluates to FALSE in a Boolean context.
- It returns FALSE when tested with IsSet() function.

Strings

They are sequences of characters, like "PHP supports string operations". Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = 'This is a somewhat longer, singly quoted string';
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

Live Demo

```
<?php
   $variable = "name";
   $literally = 'My $variable will not print!';

   print($literally);
   print "<br>";

   $literally = "My $variable will print!";
   print($literally);
?>
```

This will produce following result −

```
My $variable will not print!
My name will print
```

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP −

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are −

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character
- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

Here Document

You can assign multiple lines to a single string variable using here document −

Live Demo

```
<?php
   $channel =<<<_XML_

   <channel>
      <title>What's For Dinner</title>
      <link>http://menu.example.com/ </link>
      <description>Choose what to eat tonight.</description>
   </channel>
   _XML_;

   echo <<<END
   This uses the "here document" syntax to output multiple lines with
variable
   interpolation. Note that the here document terminator must appear on a
line with
   just a semicolon. no extra whitespace!


   END;

   print $channel;
?>
```

This will produce following result −

```
This uses the "here document" syntax to output
multiple lines with variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon. no extra whitespace!

<channel>
<title>What's For Dinner<title>
<link>http://menu.example.com/<link>
<description>Choose what to eat tonight.</description>
```

Variable Scope

Scope can be defined as the range of availability a variable has to the program in which it is declared. PHP variables can be one of four scope types −

- Local variables
- Function parameters
- Global variables
- Static variables

Variable Naming

Rules for naming a variable is −

- Variable names must begin with a letter or underscore character.
- A variable name can consist of numbers, letters, underscores but you cannot use characters like + , - , % , (,) . & , etc

There is no size limit for variables.

PHP - Operator Types

What is Operator? Simple answer can be given using expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator. PHP language supports following type of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Lets have a look on all operators one by one.

Arithmetic Operators

There are following arithmetic operators supported by PHP language −

Assume variable A holds 10 and variable B holds 20 then −

Show Examples

Operator	Description	Example
+	Adds two operands	A + B will give 30
-	Subtracts second operand from the first	A - B will give -10
*	Multiply both operands	A * B will give 200
/	Divide numerator by de-numerator	B / A will give 2
%	Modulus Operator and remainder of after an integer division	B % A will give 0
++	Increment operator, increases integer value by one	A++ will give 11
--	Decrement operator, decreases integer value by one	A-- will give 9

Comparison Operators

There are following comparison operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then −

Show Examples

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Logical Operators

There are following logical operators supported by PHP language

Assume variable A holds 10 and variable B holds 20 then −

Show Examples

Operator	Description	Example
and	Called Logical AND operator. If both the operands are true then	(A and B) is

	condition becomes true.	true.
or	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A or B) is true.
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.	(A && B) is true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is false.

Assignment Operators

There are following assignment operators supported by PHP language −

Show Examples

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B into C
+=	Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Conditional Operator

There is one more operator called conditional operator. This first evaluates an expression for a true or false value and then execute one of the two given statements depending upon the result of the evaluation. The conditional operator has this syntax −

Show Examples

Operator	Description	Example
? :	Conditional Expression	If Condition is true ? Then value X : Otherwise value Y

Operators Categories

All the operators we have discussed above can be categorised into following categories −

- Unary prefix operators, which precede a single operand.
- Binary operators, which take two operands and perform a variety of arithmetic and logical operations.
- The conditional operator (a ternary operator), which takes three operands and evaluates either the second or third expression, depending on the evaluation of the first expression.
- Assignment operators, which assign a value to a variable.

Precedence of PHP Operators

Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator −

For example x = 7 + 3 * 2; Here x is assigned 13, not 20 because operator * has higher precedence than + so it first get multiplied with 3*2 and then adds into 7.

Here operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Unary	! ++ --	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=	Right to left

PHP - Arrays

An array is a data structure that stores one or more similar type of values in a single value. For example if you want to store 100 numbers then instead of defining 100 variables its easy to define an array of 100 length.

There are three different kind of arrays and each array value is accessed using an ID c which is called array index.

- **Numeric array** − An array with a numeric index. Values are stored and accessed in linear fashion.
- **Associative array** − An array with strings as index. This stores element values in association with key values rather than in a strict linear index order.
- **Multidimensional array** − An array containing one or more arrays and values are accessed using multiple indices

NOTE − Built-in array functions is given in function reference PHP Array Functions

Numeric Array

These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.

Example

Following is the example showing how to create and access numeric arrays.

Here we have used **array()** function to create array. This function is explained in function reference.

```
<html>
   <body>

      <?php
         /* First method to create array. */
         $numbers = array( 1, 2, 3, 4, 5);

         foreach( $numbers as $value ) {
            echo "Value is $value <br />";
         }

         /* Second method to create array. */
         $numbers[0] = "one";
         $numbers[1] = "two";
         $numbers[2] = "three";
         $numbers[3] = "four";
         $numbers[4] = "five";

         foreach( $numbers as $value ) {
            echo "Value is $value <br />";
         }
      ?>

   </body>
</html>
```

This will produce the following result −

```
Value is 1
Value is 2
Value is 3
Value is 4
Value is 5
Value is one
```

```
Value is two
Value is three
Value is four
Value is five
```

Associative Arrays

The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.

To store the salaries of employees in an array, a numerically indexed array would not be the best choice. Instead, we could use the employees names as the keys in our associative array, and the value would be their respective salary.

NOTE − Don't keep associative array inside double quote while printing otherwise it would not return any value.

Example

Live Demo

```
<html>
   <body>

      <?php
         /* First method to associate create array. */
         $salaries = array("mohammad" => 2000, "qadir" => 1000, "zara" =>
500);

         echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
         echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
         echo "Salary of zara is ".  $salaries['zara']. "<br />";

         /* Second method to create array. */
         $salaries['mohammad'] = "high";
         $salaries['qadir'] = "medium";
         $salaries['zara'] = "low";

         echo "Salary of mohammad is ". $salaries['mohammad'] . "<br />";
         echo "Salary of qadir is ".  $salaries['qadir']. "<br />";
         echo "Salary of zara is ".  $salaries['zara']. "<br />";
      ?>

   </body>
</html>
```

This will produce the following result −

```
Salary of mohammad is 2000
Salary of qadir is 1000
Salary of zara is 500
Salary of mohammad is high
Salary of qadir is medium
Salary of zara is low
```

Multidimensional Arrays

A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

Example

In this example we create a two dimensional array to store marks of three students in three subjects −

This example is an associative array, you can create numeric array in the same fashion.

Live Demo

```
<html>
   <body>

      <?php
         $marks = array(
            "mohammad" => array (
               "physics" => 35,
               "maths" => 30,
               "chemistry" => 39
            ),

            "qadir" => array (
               "physics" => 30,
               "maths" => 32,
               "chemistry" => 29
            ),

            "zara" => array (
               "physics" => 31,
               "maths" => 22,
               "chemistry" => 39
            )
         );

         /* Accessing multi-dimensional array values */
         echo "Marks for mohammad in physics : " ;
         echo $marks['mohammad']['physics'] . "<br />";

         echo "Marks for qadir in maths : ";
         echo $marks['qadir']['maths'] . "<br />";

         echo "Marks for zara in chemistry : " ;
         echo $marks['zara']['chemistry'] . "<br />";
      ?>

   </body>
</html>
```

This will produce the following result −

```
Marks for mohammad in physics : 35
Marks for qadir in maths : 32
```

```
Marks for zara in chemistry : 39
```

PHP - Strings

They are sequences of characters, like "PHP supports string operations".

NOTE − Built-in string functions is given in function reference PHP String Functions

Following are valid examples of string

```
$string_1 = "This is a string in double quotes";
$string_2 = "This is a somewhat longer, singly quoted string";
$string_39 = "This string has thirty-nine characters";
$string_0 = ""; // a string with zero characters
```

Singly quoted strings are treated almost literally, whereas doubly quoted strings replace variables with their values as well as specially interpreting certain character sequences.

Live Demo

```
<?php
   $variable = "name";
   $literally = 'My $variable will not print!\\n';

   print($literally);
   print "<br />";

   $literally = "My $variable will print!\\n";

   print($literally);
?>
```

This will produce the following result −

```
My $variable will not print!\n
My name will print!\n
```

There are no artificial limits on string length - within the bounds of available memory, you ought to be able to make arbitrarily long strings.

Strings that are delimited by double quotes (as in "this") are preprocessed in both the following two ways by PHP −

- Certain character sequences beginning with backslash (\) are replaced with special characters
- Variable names (starting with $) are replaced with string representations of their values.

The escape-sequence replacements are −

- \n is replaced by the newline character
- \r is replaced by the carriage-return character
- \t is replaced by the tab character

- \$ is replaced by the dollar sign itself ($)
- \" is replaced by a single double-quote (")
- \\ is replaced by a single backslash (\)

String Concatenation Operator

To concatenate two string variables together, use the dot (.) operator −

Live Demo

```
<?php
   $string1="Hello World";
   $string2="1234";

   echo $string1 . " " . $string2;
?>
```

This will produce the following result −

```
Hello World 1234
```

If we look at the code above you see that we used the concatenation operator two times. This is because we had to insert a third string.

Between the two string variables we added a string with a single character, an empty space, to separate the two variables.

Using the strlen() function

The strlen() function is used to find the length of a string.

Let's find the length of our string "Hello world!" −

Live Demo

```
<?php
   echo strlen("Hello world!");
?>
```

This will produce the following result −

```
12
```

The length of a string is often used in loops or other functions, when it is important to know when the string ends. (i.e. in a loop, we would want to stop the loop after the last character in the string)

Using the strpos() function

The strpos() function is used to search for a string or character within a string.

If a match is found in the string, this function will return the position of the first match. If no match is found, it will return FALSE.

Let's see if we can find the string "world" in our string −

Live Demo

```
<?php
   echo strpos("Hello world!","world");
?>
```

This will produce the following result −

```
 6
```

As you see the position of the string "world" in our string is position 6. The reason that it is 6, and not 7, is that the first position in the string is 0, and not 1.

PHP - Files & I/O

functions related to files −

- Opening a file
- Reading a file
- Writing a file
- Closing a file

Opening and Closing Files

The PHP **fopen()** function is used to open a file. It requires two arguments stating first the file name and then mode in which to operate.

Files modes can be specified as one of the six options in this table.

Sr.No	Mode & Purpose
	r
1	Opens the file for reading only. Places the file pointer at the beginning of the file.
	r+
2	Opens the file for reading and writing. Places the file pointer at the beginning of the file.
	w
3	Opens the file for writing only.

	Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file. **w+** Opens the file for reading and writing only.	
4	Places the file pointer at the beginning of the file. and truncates the file to zero length. If files does not exist then it attempts to create a file. **a** Opens the file for writing only.	
5	Places the file pointer at the end of the file. If files does not exist then it attempts to create a file. **a+** Opens the file for reading and writing only.	
6	Places the file pointer at the end of the file. If files does not exist then it attempts to create a file.	

If an attempt to open a file fails then **fopen** returns a value of **false** otherwise it returns a **file pointer** which is used for further reading or writing to that file.

After making a changes to the opened file it is important to close it with the **fclose()** function. The **fclose()** function requires a file pointer as its argument and then returns **true** when the closure succeeds or **false** if it fails.

Reading a file

Once a file is opened using **fopen()** function it can be read with a function called **fread()**. This function requires two arguments. These must be the file pointer and the length of the file expressed in bytes.

The files length can be found using the **filesize()** function which takes the file name as its argument and returns the size of the file expressed in bytes.

So here are the steps required to read a file with PHP.

- Open a file using **fopen()** function.

- Get the file's length using **filesize()** function.
- Read the file's content using **fread()** function.
- Close the file with **fclose()** function.

The following example assigns the content of a text file to a variable then displays those contents on the web page.

```
<html>

   <head>
      <title>Reading a file using PHP</title>
   </head>

   <body>

      <?php
         $filename = "tmp.txt";
         $file = fopen( $filename, "r" );

         if( $file == false ) {
            echo ( "Error in opening file" );
            exit();
         }

         $filesize = filesize( $filename );
         $filetext = fread( $file, $filesize );
         fclose( $file );

         echo ( "File size : $filesize bytes" );
         echo ( "<pre>$filetext</pre>" );
      ?>

   </body>
</html>
```

It will produce the following result −

Writing a file

A new file can be written or text can be appended to an existing file using the PHP **fwrite()** function. This function requires two arguments specifying a **file pointer** and the string of data that is to be written. Optionally a third integer argument can be included to specify the length of the data to write. If the third argument is included, writing would will stop after the specified length has been reached.

The following example creates a new text file then writes a short text heading inside it. After closing this file its existence is confirmed using **file_exist()** function which takes file name as an argument

```
<?php
   $filename = "/home/user/guest/newfile.txt";
   $file = fopen( $filename, "w" );
```

```
   if( $file == false ) {
      echo ( "Error in opening new file" );
      exit();
   }
   fwrite( $file, "This is  a simple test\n" );
   fclose( $file );
?>
<html>

   <head>
      <title>Writing a file using PHP</title>
   </head>

   <body>

      <?php
         $filename = "newfile.txt";
         $file = fopen( $filename, "r" );

         if( $file == false ) {
            echo ( "Error in opening file" );
            exit();
         }

         $filesize = filesize( $filename );
         $filetext = fread( $file, $filesize );

         fclose( $file );

         echo ( "File size : $filesize bytes" );
         echo ( "$filetext" );
         echo("file name: $filename");
      ?>

   </body>
</html>
```

PHP - Functions

PHP functions are similar to other programming languages. A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value.

You already have seen many functions like **fopen()** and **fread()** etc. They are built-in functions but PHP gives you option to create your own functions as well.

There are two parts which should be clear to you −

- Creating a PHP Function
- Calling a PHP Function

In fact you hardly need to create your own PHP function because there are already more than 1000 of built-in library functions created for different area and you just need to call them according to your requirement.

Please refer to PHP Function Reference for a complete set of useful functions.

Creating PHP Function

Its very easy to create your own PHP function. Suppose you want to create a PHP function which will simply write a simple message on your browser when you will call it. Following example creates a function called writeMessage() and then calls it just after creating it.

Note that while creating a function its name should start with keyword **function** and all the PHP code should be put inside { and } braces as shown in the following example below −

Live Demo

```
<html>

   <head>
      <title>Writing PHP Function</title>
   </head>

   <body>

      <?php
         /* Defining a PHP Function */
         function writeMessage() {
            echo "You are really a nice person, Have a nice time!";
         }

         /* Calling a PHP Function */
         writeMessage();
      ?>

   </body>
</html>
```

This will display following result −

```
You are really a nice person, Have a nice time!
```

PHP Functions with Parameters

PHP gives you option to pass your parameters inside a function. You can pass as many as parameters your like. These parameters work like variables inside your function. Following example takes two integer parameters and add them together and then print them.

Live Demo

```
<html>

   <head>
      <title>Writing PHP Function with Parameters</title>
   </head>

   <body>

      <?php
         function addFunction($num1, $num2) {
            $sum = $num1 + $num2;
```

```
            echo "Sum of the two numbers is : $sum";
         }

         addFunction(10, 20);
      ?>

   </body>
</html>
```

This will display following result −

```
Sum of the two numbers is : 30
```

Passing Arguments by Reference

It is possible to pass arguments to functions by reference. This means that a reference to the variable is manipulated by the function rather than a copy of the variable's value.

Any changes made to an argument in these cases will change the value of the original variable. You can pass an argument by reference by adding an ampersand to the variable name in either the function call or the function definition.

Following example depicts both the cases.

Live Demo

```
<html>

   <head>
      <title>Passing Argument by Reference</title>
   </head>

   <body>

      <?php
         function addFive($num) {
            $num += 5;
         }

         function addSix(&$num) {
            $num += 6;
         }

         $orignum = 10;
         addFive( $orignum );

         echo "Original Value is $orignum<br />";

         addSix( $orignum );
         echo "Original Value is $orignum<br />";
      ?>

   </body>
</html>
```

This will display following result −

```
Original Value is 10
Original Value is 16
```

PHP Functions returning value

A function can return a value using the **return** statement in conjunction with a value or object. return stops the execution of the function and sends the value back to the calling code.

You can return more than one value from a function using **return array(1,2,3,4)**.

Following example takes two integer parameters and add them together and then returns their sum to the calling program. Note that **return** keyword is used to return a value from a function.

Live Demo

```
<html>

   <head>
      <title>Writing PHP Function which returns value</title>
   </head>

   <body>

      <?php
         function addFunction($num1, $num2) {
            $sum = $num1 + $num2;
            return $sum;
         }
         $return_value = addFunction(10, 20);

         echo "Returned value from the function : $return_value";
      ?>

   </body>
</html>
```

This will display following result −

```
Returned value from the function : 30
```

Setting Default Values for Function Parameters

You can set a parameter to have a default value if the function's caller doesn't pass it.

Following function prints NULL in case use does not pass any value to this function.

Live Demo

```
<html>

   <head>
      <title>Writing PHP Function which returns value</title>
   </head>

   <body>
```

```
      <?php
         function printMe($param = NULL) {
            print $param;
         }

         printMe("This is test");
         printMe();
      ?>

   </body>
</html>
```

This will produce following result −

```
This is test
```

Dynamic Function Calls

It is possible to assign function names as strings to variables and then treat these variables exactly as you would the function name itself. Following example depicts this behaviour.

Live Demo

```
<html>

   <head>
      <title>Dynamic Function Calls</title>
   </head>

   <body>

      <?php
         function sayHello() {
            echo "Hello<br />";
         }

         $function_holder = "sayHello";
         $function_holder();
      ?>

   </body>
</html>
```

This will display following result −

```
Hello
```

PHP - Form

What is the Form?

A Document that containing black fields, that the user can fill the data or user can select the data.Casually the data will store in the data base

Example

Below example shows the form with some specific actions by using post method.

```
<html>

   <head>
      <title>PHP Form Validation</title>
   </head>

   <body>
      <?php

         // define variables and set to empty values
         $name = $email = $gender = $comment = $website = "";

         if ($_SERVER["REQUEST_METHOD"] == "POST") {
            $name = test_input($_POST["name"]);
            $email = test_input($_POST["email"]);
            $website = test_input($_POST["website"]);
            $comment = test_input($_POST["comment"]);
            $gender = test_input($_POST["gender"]);
         }

         function test_input($data) {
            $data = trim($data);
            $data = stripslashes($data);
            $data = htmlspecialchars($data);
            return $data;
         }
      ?>

      <h2>Tutorials Point Absolute classes registration</h2>

      <form method = "post" action = "/php/php_form_introduction.htm">
         <table>
            <tr>
               <td>Name:</td>
               <td><input type = "text" name = "name"></td>
            </tr>

            <tr>
               <td>E-mail:</td>
               <td><input type = "text" name = "email"></td>
            </tr>

            <tr>
               <td>Specific Time:</td>
               <td><input type = "text" name = "website"></td>
            </tr>

            <tr>
               <td>Class details:</td>
               <td><textarea name = "comment" rows = "5" cols =
"40"></textarea></td>
            </tr>

            <tr>
               <td>Gender:</td>
               <td>
```

```
                  <input type = "radio" name = "gender" value =
"female">Female
                  <input type = "radio" name = "gender" value = "male">Male
               </td>
            </tr>

            <tr>
               <td>
                  <input type = "submit" name = "submit" value = "Submit">
               </td>
            </tr>
         </table>
      </form>

      <?php
         echo "<h2>Your Given details are as :</h2>";
         echo $name;
         echo "<br>";

         echo $email;
         echo "<br>";

         echo $website;
         echo "<br>";

         echo $comment;
         echo "<br>";

         echo $gender;
      ?>

   </body>
</html>
```

13. Unit wise questions

Important Questions From Each Unit (WT)

Unit I

1. Explain Basic Html tags.

Ans: **Basic HTML tags**
1. Body tag :
Body tag contain some attributes such as bgcolor, background etc. bgcolor is used for background color, which takes background color name or hexadecimal number and #FFFFFF and background attribute will take the path of the image which you can place as the background image in the browser.
<body bgcolor=”#F2F3F4” background= “c:\amer\imag1.gif”>
2. Paragraph tag:
Most text is part of a paragraph of information. Each paragraph is aligned to the left, right or center of the page by using an attribute called as align.
<p align=”left” | “right” | “center”>
3. Heading tag:
HTML is having six levels of heading that are commonly used. The largest heading tag is <h1> . The different levels of heading tag besides <h1> are <h2>, <h3>, <h4>, <h5> and <h6>. These heading tags also contain attribute called as align.
<h1 align=”left” | “right” | “center”> . . . . <h2>
4. hr tag:
This tag places a horizontal line across the system. These lines are used to break the page. This tag also contains attribute i.e., width which draws the horizontal line with the screen size of the browser. This tag does not require an end tag.
<hr width=”50%”>.
5. base font:
This specify format for the basic text but not the headings.
<basefont size=”10”>
6. font tag:
This sets font size, color and relative values for a particular text.
<font size=”10” color=”#f1f2f3”>
7. bold tag:
This tag is used for implement bold effect on the text
 …….
8. Italic tag:
This implements italic effects on the text.
…….
9. strong tag:
This tag is used to always emphasized the text
…….
10. tt tag:
This tag is used to give typewriting effect on the text
<tt>……..</tt>
11. sub and sup tag:

These tags are used for subscript and superscript effects on the text.
_{…….}
^{……..}
12. Break tag:
This tag is used to the break the line and start from the next line.

13. &amp &lt &gt   &quot
These are character escape sequence which are required if you want to display characters that HTML uses as control sequences.
Example: < can be represented as &lt.
14. Anchor tag:
This tag is used to link two HTML pages, this is represented by <a>
 some text
href is an attribute which is used for giving the path of a file which you want to link.

2. Explain the Advantages JavaScript with suitable examples.

Ans: **i. JavaScript is a client side language:** The JavaScript code is executed on the user's processor instead of the web server thus it saves bandwidth and load on the web server.

ii. JavaScript is an easy language to learn: The JavaScript language is easy to learn and offers syntax that is close to English. It uses the DOM model that provides plenty of predefined functionalities to the various objects on pages making it a breeze to develop a script to solve a custom purpose.

iii. No compilation needed: JavaScript does not require compilation process so no compiler is needed. The browser interprets JavaScript as it HTML tags.

Iv. Easy to debug and test: The understanding syntax of JavaScript is easy. Any person can learn it very easily and use it to develop dynamic and scalable websites.

v. Procedural programming capabilities: JavaScript language encompasses all the capabilities of a procedural language. Branching, looping, condition checking are some of those capabilities that can be executed on a web page.

Example:

```
<html>

<head>

<title>Event!!!</title>

<script type="text/javascript">

function trigger()

{

document.getElementById("hover").addEventListener("mouseover",
popup);
```

```
function popup()

{

alert("Welcome to my WebPage!!!");

}

}

</script>

</head>

<body  onload="trigger();">

<p id="hover">Welcome!!!</p>

</body>

</html>
```

3. Write short notes on the following:
   a) HTML Frames

Ans: a) HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document. A collection of frames in the browser window is known as a frameset. The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames

There are few drawbacks with using frames, so it's never recommended to use frames in your webpages −

- Some smaller devices cannot cope with frames often because their screen is not big enough to be divided up.
- Sometimes your page will be displayed differently on different computers due to different screen resolution.
- The browser's *back* button might not work as the user hopes.

- There are still few browsers that do not support frame technology.

4. Explain how events are handled in JavaScript.

Ans: An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

Often, when events happen, you may want to do something.

JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, **with JavaScript code**, to be added to HTML elements.

An event occurs when something happens in a browser window. The kinds of events that might occur are due to:

- A document loading
- The user clicking a mouse button
- The browser screen changing size

When a function is assigned to an event handler, that function is run when that event occurs.

A handler that is assigned from a script used the syntax '`[element].[event] = [function];`', where `[element]` is a page element, `[event]` is the name of the selected event and [function] is the name of the function that occurs when the event takes place.

For example:

```
document.onclick = clickHandler;
```

This handler will cause the function `clickHandler()` to be executed whenever the user clicks the mouse anywhere on the screen. Note that when an event handler is assigned, the function name does not end with parentheses. We are just pointing the event to the name of the function. The `clickHandler()` function is defined like this:

```
function clickHandler(event) {
  //some code here
}
```

By convention the event is represented by the variable '`event()`'. In some browsers the event must be explicitly passed to the function, so as a precaution it's often best to include a

conditional to test that the `event()` variable has been passed, and if it hasn't then to use an alternative method that works on those other browsers:

```
function clickHandler(event) {
  event = event || window.event;
  //some code here
}
```

Elements within a document can also be assigned event handlers. For example:

```
document.getElementsByTagName('a')[0].onclick = linkHandler;
```

This will cause the `linkHandler()` function to be executed when the user clicks the first link on the page.

Keep in mind that this style of handler assignment depends on the link's position inside the page. If another link tag is added before this one, it will take over the handler from the original link. A best practice is to maintain the separation of code and page structure by assigning each link an identifier by using the id attribute.

```
<a id="faqLink" href="faq.html">Faq</a>
```

A handler assignment can then work regardless of where the element is positioned.

```
document.getElementById('faqLink').onclick = linkHandler;
```

5. Explain how the Math object used in JavaScript with examples.

Ans:

The **math** object provides you properties and methods for mathematical constants and functions. Unlike other global objects, **Math** is not a constructor. All the properties and methods of **Math** are static and can be called by using Math as an object without creating it.

Thus, you refer to the constant **pi** as **Math.PI** and you call the *sine* function as **Math.sin(x)**, where x is the method's argument.

Syntax

The syntax to call the properties and methods of Math are as follows

```
var pi_val = Math.PI;
var sine_val = Math.sin(30);
```

6. List the Math methods.

Ans: *Math Methods*

Here is a list of the methods associated with Math object and their description

Sr.No	Method & Description
1	abs() Returns the absolute value of a number.
2	acos() Returns the arccosine (in radians) of a number.
3	asin() Returns the arcsine (in radians) of a number.
4	atan() Returns the arctangent (in radians) of a number.
5	atan2() Returns the arctangent of the quotient of its arguments.
6	ceil() Returns the smallest integer greater than or equal to a number.
7	cos() Returns the cosine of a number.
8	exp() Returns E^N, where N is the argument, and E is Euler's constant, the base of the natural logarithm.
9	floor() Returns the largest integer less than or equal to a number.
10	log() Returns the natural logarithm (base E) of a number.
11	max() Returns the largest of zero or more numbers.
12	min() Returns the smallest of zero or more numbers.
13	pow()

Returns base to the exponent power, that is, base exponent.

14 random()

Returns a pseudo-random number between 0 and 1.

15 round()

Returns the value of a number rounded to the nearest integer.

16 sin()

Returns the sine of a number.

17 sqrt()

Returns the square root of a number.

18 tan()

Returns the tangent of a number.

19 toSource()

Returns the string "Math".

Unit II

1. What is a namespace? Describe how a namespace is created with an example.

Ans: A **Namespace** is a set of unique names. Namespace is a mechanisms by which element and attribute name can be assigned to a group. The Namespace is identified by URI(Uniform Resource Identifiers).

Namespace Declaration

A Namespace is declared using reserved attributes. Such an attribute name must either be **xmlns** or begin with **xmlns:** shown as below −

```
<element xmlns:name = "URL">
```

Syntax

- The Namespace starts with the keyword **xmlns**.
- The word **name** is the Namespace prefix.
- The **URL** is the Namespace identifier.

Example

Namespace affects only a limited area in the document. An element containing the declaration and all of its descendants are in the scope of the Namespace. Following is a simple example of XML Namespace −

```
<?xml version ="1.0" encoding ="UTF-8"?>
<cont:contactxmlns:cont="www.tutorialspoint.com/profile">
<cont:name>Tanmay Patil</cont:name>
<cont:company>TutorialsPoint</cont:company>
<cont:phone>(011) 123-4567</cont:phone>
```

</cont:contact>

Here, the Namespace prefix is **cont**, and the Namespace identifier (URI) as *www.tutorialspoint.com/profile*. This means, the element names and attribute names with the **cont** prefix (including the contact element), all belong to the *www.tutorialspoint.com/profile* namespace.

2. Give the uses of DTD.

Ans: The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements. A DTD can be declared inline in your XML document, or as an external reference.

Internal DTD
This is an XML document with a Document Type Definition: (Open it in IE5, and select view source)

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note    (to,from,heading,body)>
<!ELEMENT to      (#PCDATA)>
<!ELEMENT from    (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body    (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

The DTD is interpreted like this:
!ELEMENT note (in line 2) defines the element “note” as having four elements: “to,from,heading,body”.
!ELEMENT to (in line 3) defines the “to” element to be of the type “CDATA”.
!ELEMENT from (in line 4) defines the “from” element to be of the type “CDATA”
and so on…..

External DTD

This is the same XML document with an external DTD: (Open it in IE5, and select view source)

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

This is a copy of the file “note.dtd” containing the Document Type Definition:

3. Define XML Schema. Show how it is created.

Ans: An XML Schema describes the structure of an XML document.
The XML Schema language is also referred to as XML Schema Definition (XSD).
XSD Example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

The purpose of an XML Schema is to define the legal building blocks of an XML document:

- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

4. What are the goals of XML.

Ans: Design goals for XML

- XML shall be straightforwardly usable over the Internet.
- XML shall support a wide variety of applications.
- XML shall be compatible with SGML.
- It shall be easy to write programs which process XML documents.
- The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
- XML documents should be human-legible and reasonably clear.
- The XML design should be prepared quickly.
- The design of XML shall be formal and concise.
- XML documents shall be easy to create.
- Terseness in XML markup is of minimal importance.

5. Give the syntax of an XML document.

Ans: The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.

XML Documents Must Have a Root Element
XML documents must contain one **root** element that is the **parent** of all other elements:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

In this example **<note>** is the root element:

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

The XML Prolog
This line is called the XML **prolog**:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The XML prolog is optional. If it exists, it must come first in the document.
XML documents can contain international characters, like Norwegian øæå or French êèé.
To avoid errors, you should specify the encoding used, or save your XML files as UTF-8.
UTF-8 is the default character encoding for XML documents.

Unit III

1. Explain the functionality of javax.servlet package for web servers. Discuss about various classes and interfaces of the package.

Ans:

A servlet always work in a framework and this framework is supported by various classes and interfaces of javax.servlet package.Various classes and interfaces used by this package are enlisted as follows

Class name	Description
Generic Servlet	This class implements the Servlet and ServletConfig Interfaces
ServletInputStream	Fro reading the client request the input stream is generated using this class
ServletOutputStream	For writing the client response the outputstream is generated using this class
ServletException	Handles the servlet error

Interface name	Description
Servlet	This is very important interface which is responsible for implementing the lifecycle methods such as init,service,destroy methods of servlet
ServletConfig	Using this interface ,the initialization parameters can be obtained by servlets
ServletRequest	When client makes some request then the data within that request can be read using this interface
ServletResponse	When the client gets the response then the data for that response is written using this interface
ServletContext	Using this interface servlet can log the events and access the information about it

2. Write about Servlet Lifecycle.
Ans:

Servlet Life Cycle

Servlet is a Java programming language class that provides a component-based, platform-independent method to create web based applications to enhance the functionality of web server.

If we talk about Servlet Life Cycle, we would need to cover the entire process starting from its initialization till the destruction which is basically wrapped in four stages that are:

- **Loading and Instantiation**
- **Initialization**
- **Servicing the Request**
- **Destroying the Servlet**

1. **Loading and Installation:**

   This is the initial stage of Servlet Life Cycle in which servlet container loads the servlet from web.xml, which is done either during initiation or when the first request is made. If the attribute <load-on-startup> of web.xml file has a positive value then the Servlet will be loaded else it will load when the first request comes for service. After loading of the servlet, the container creates the instances of the Servlet by using *Class.forName(ServletName).newInstance()*.

2. **Initialization:**

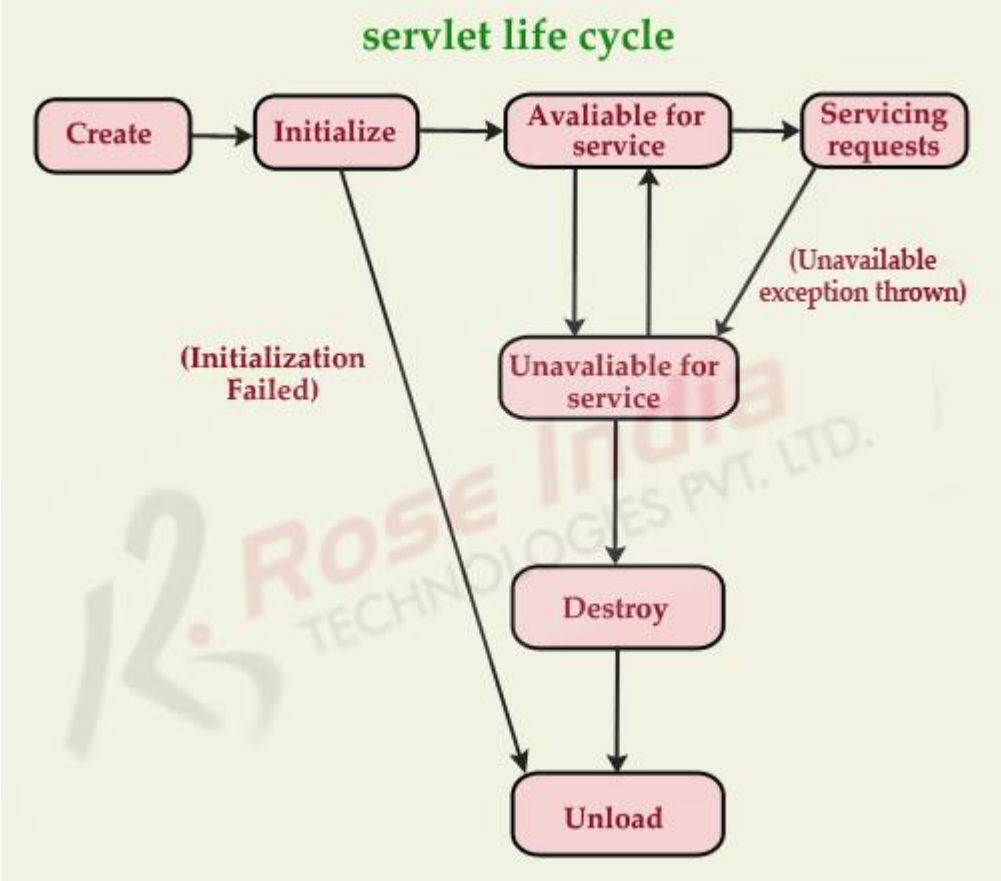
   Once the instances are created the second stage called Initialization begins in which the servlet container calls the ***init()*** method and passes the servlet initialization parameters to the ***init()*** method. One thing that is to be known is, the ***init()*** must be called by the servlet container prior to the Servlet can serve any request. The initialization parameter continues until the Servlet is destroyed. The ***init()*** method is called only once during the entire life cycle of the servlet. The servlet will only be available for service if it is servlet code is loaded successfully without any error otherwise it will be not available for service.

3. **Servicing the Request:**

   Once the initialization process has been completed successfully, the Servlet becomes available for service. Moreover, servlet creates a separate thread for each of the request made and the the servlet container calls the ***service()*** method for servicing any request. The ***service()*** method evaluates the kind of request and calls the suitable method (***doGet()*** or ***doPost()***) to execute the request and send response to the client using the methods of the response object.

4. **Destroying the Servlet:**

If the Servlet is no longer needed to serve any request further, the servlet container calls for the ***destroy()*** method. Like the ***init()*** method this ***destroy()*** method is also called only once in the entire life cycle of the servlet. Once the ***destroy()*** method is called, it indicates to the servlet container not to send any request for service and the Servlet releases all the resources related to it. Java Virtual Machine is responsible for the memory associated with the resources for garbage collection.



3.Give the advantages of Servlets over CGI.
Ans:

Servlets have a number of advantages over CGI and other API's. They are:

1. **Platform Independence**
   Servlets are written entirely in java so these are platform independent. Servlets can run on any Servlet enabled web server. For example if you develop an web application in windows machine running Java web server, you can easily run the same on apache web server (if Apache Serve is installed) without modification or compilation of code. Platform independency of servlets provide a great advantages over alternatives of servlets.
2. **Performance**
   Due to interpreted nature of java, programs written in java are slow. But the java servlets runs very fast. These are due to the way servlets run on web server. For any program initialization takes significant amount of time. But in case of servlets initialization takes place first time it receives a request and remains in memory till times out or server shut downs. After servlet is loaded, to handle a new request it

simply creates a new thread and runs service method of servlet. In comparison to traditional CGI scripts which creates a new process to serve the request.

3. **Extensibility**
   Java Servlets are developed in java which is robust, well-designed and object oriented language which can be extended or polymorphed into new objects. So the java servlets take all these advantages and can be extended from existing class to provide the ideal solutions.
4. **Safety**
   Java provides very good safety features like memory management, exception handling etc. Servlets inherits all these features and emerged as a very powerful web server extension.
5. **Secure**
   Servlets are server side components, so it inherits the security provided by the web server. Servlets are also benefited with Java Security Manager

4. Make a comparison of servlet and applets.

Ans:

BASIS FOR COMPARISON	**APPLET**	**SERVLET**
Execution	Applet is always executed on the client side.	Servlet is always executed on the server side.
Packages	import java.applet.*; import java.awt.*;	import javax.servlet.*; import java.servlet.http.*;
Lifecycle methods	init(), stop(), paint(), start(), destroy().	init( ), service( ), and destroy( ).

User interface	Applets use user interface classes like AWT and Swing.	No User interface required.
Requirement	Requires java compatible browser for execution.	It processes the input from client side and generates the response in terms of the HTML page, Javascript, Applets.
Resources	As it arrives at the client, it uses the resources of the client to produce graphical interface and run complex computation.	It utilizes the resources of the server to process request and response of the client.
Bandwidth Utilization	Applets utilize more network bandwidth as it executes on the client machine.	Servlets are executed on the servers and hence require less bandwidth.
Security	More prone to risk as it is on the client machine.	It is under the server security.

5. What are cookies in Servlets?

Ans: **Cookies are text files that are sent by Servlet to the Web Browsers and are used to track various information's about the user.**

Cookies in Servlet

Cookies are text files that are sent by Servlet to the Web Browsers that uniquely identifies a client. Browsers store cookies on local computer.

Cookies are used to track various informations. Whenever a browser sends a request to browser, it also sends the information stored in the local computer so that server can identify the user.

A cookie has a name, a value, optional attributes and a version number.

HttpServletResponse.addCookie(javax.servlet.http.Cookie) method is used by servlet to send cookies to the browser

A browser can store up to 20 cookies for each Web server and a total of 300 cookies.

HttpServletRequest.getCookies() method is used to retrieve cookies.

Cookies can have same name but they always have different path attributes.

Cookies affect the Web pages cache.

Servlet Cookie Methods:

java.lang.Object clone(): overrides java.lang.Object.clone method and returns a copy of this cookie

java.lang.String getComment(): returns the comment attached to this cookie

java.lang.String getDomain(): returns the domain name set for this cookie

int getMaxAge(): returns the maximum age of the cookie in seconds

java.lang.String getName(): returns the cookie's name

java.lang.String getPath(): returns the path on the server where the browser will return this cookie

boolean getSecure(): if the browser sends cookies on a secure protocol, it returns true else false

java.lang.String getValue(): returns cookie's value

int getVersion(): returns the version of the protocol cookie follows

void setComment(java.lang.String purpose): specifies cookie's purpose

void setDomain(java.lang.String pattern): specifies the domain in which cookie must be presented

void setMaxAge(int expiry): sets cookie's maximum age in seconds

void setPath(java.lang.String uri): specifies a path for the client to return the cookie

void setSecure(boolean flag): indicates that the cookie must be sent using a secure protocol

void setValue(java.lang.String newValue): When a cookie has been created, it assigns new value to cookie.

void setVersion(int v): Sets the version of the cookie.

6. What are different methods in Servlets?

Ans:

A Generic servlet contains the following five methods:

init()

public void init(ServletConfig config) throws ServletException

The **init() method** is called only once by the servlet container throughout the life of a servlet. By this init() method the servlet get to know that it has been placed into service.

The servlet cannot be put into the service if

- The init() method does not return within a fix time set by the web server.
- It throws a ServletException

Parameters - The init() method takes a **ServletConfig** object that contains the initialization parameters and servlet's configuration and throws a **ServletException** if an exception has occurred.

service()

public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException

Once the servlet starts getting the requests, the service() method is called by the servlet container to respond. The servlet services the client's request with the help of two objects. These two objects **javax.servlet.ServletRequest** and **javax.servlet.ServletResponse** are passed by the servlet container.

The status code of the response always should be set for a servlet that throws or sends an

error.

Parameters - The service() method takes **the ServletRequest** object that contains the client's request and the object **ServletResponse** contains the servlet's response. The service() method throws **ServletException and IOExceptions** exception.

getServletConfig()

public ServletConfig getServletConfig()

This method contains parameters for initialization and startup of the servlet and returns a **ServletConfig object.** This object is then passed to the init method. When this interface is implemented then it stores **the ServletConfig object** in order to return it. It is done by the generic class which implements this inetrface.

Returns - the ServletConfig object

getServletInfo()

public String getServletInfo()

The information about the servlet is returned by this method like version, author etc. This method returns a string which should be in the form of plain text and not any kind of markup.

Returns - a string that contains the information about the servlet

destroy()

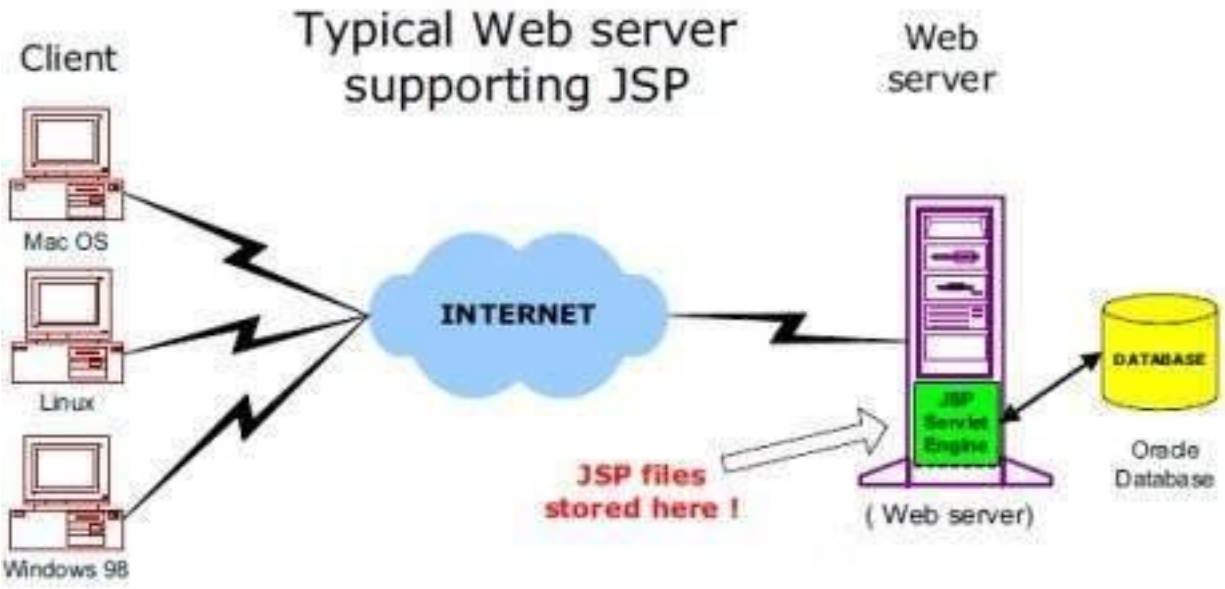
public void destroy()

This method is called when we need to close the servlet. That is before removing a servlet instance from service, the servlet container calls the destroy() method. Once the servlet container calls the destroy() method, no service methods will be then called . That is after the exit of all the threads running in the servlet, the destroy() method is called. Hence, the servlet gets a chance to clean up all the resources like memory, threads etc which are being held.

Unit IV

1. Explain how JSP processing is handled. Also show how JSP are better than servlet.

Ans:

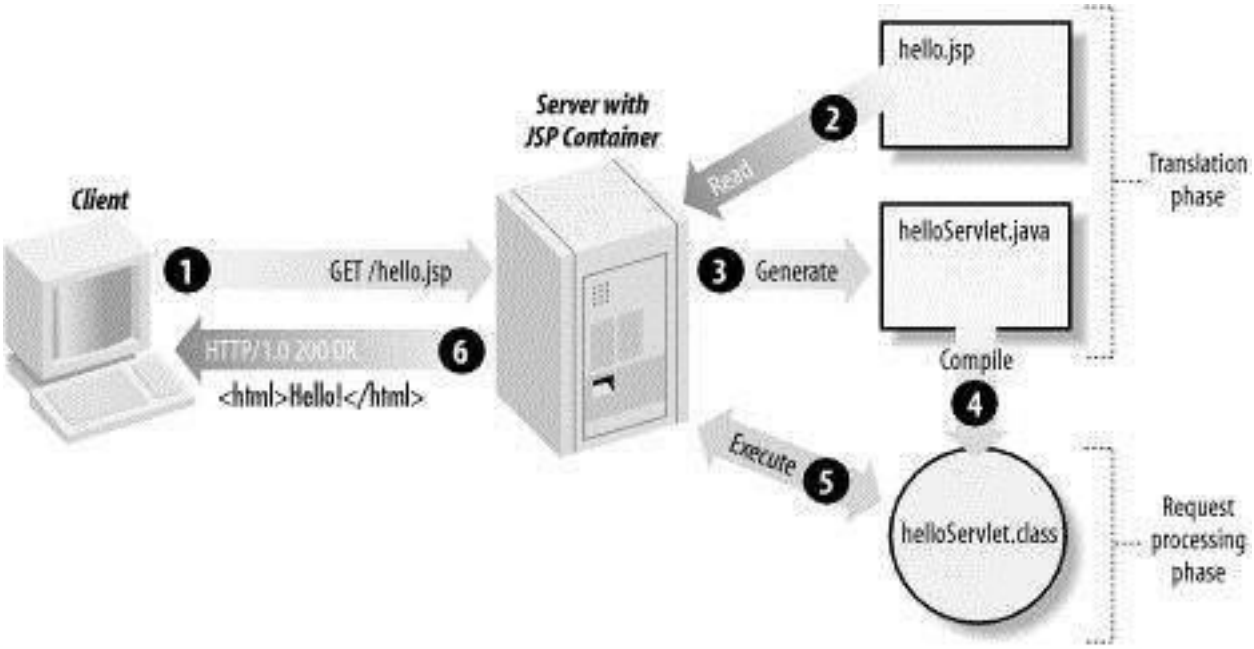


JSP Processing

The following steps explain how the web server creates the Webpage using JSP −

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with **.jsp** instead of **.html**.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements and all JSP elements are converted to Java code. This code implements the corresponding dynamic behavior of the page.
- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format. The output is furthur passed on to the web server by the servlet engine inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.

All the above mentioned steps can be seen in the following diagram −



Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with the other scripting languages (such as PHP) and therefore faster.
So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet.

2. Compare and contrast cookies and sessions with suitable examples

Cookies	**Sessions**
Cookies are stored in browser as text file format.	Sessions are stored in server side.
It is stored limit amount of data.	It is stored unlimited amount of data
It is only allowing 4kb[4096bytes].	It is holding the multiple variable in sessions.
It is not holding the multiple variable in cookies.	It is holding the multiple variable in sessions.

we can accessing the cookies values in easily. So it is less secure.	we cannot accessing the session values in easily.So it is more secure.
setting the cookie time to expire the cookie.	using session_destory(), we we will destroyed the sessions.
The setcookie() function must appear BEFORE the <html> tag.	The session_start() function must be the very first thing in your document. Before any HTML tags.

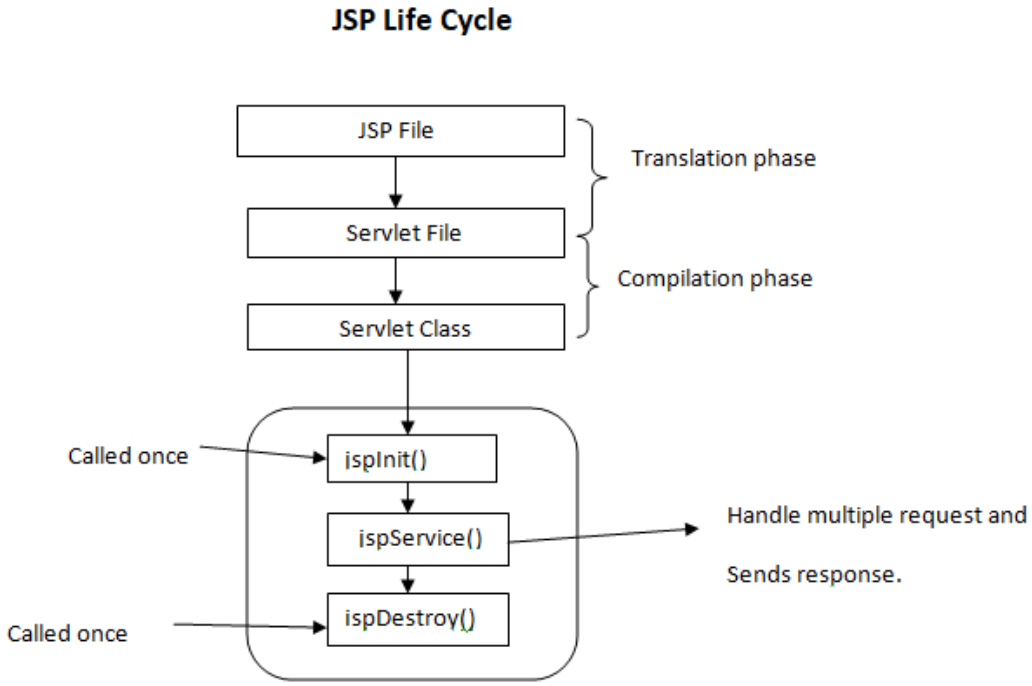
3. Describe the problems associated with servlets? And how to overcome?

Ans:

1. Servlets are difficult to code which are overcome in JSP. Other way, we can say, JSP is almost a **replacement** of Servlets, (by large, the better word is **extension** of Servlets), where coding decreases more than half.
2. In Servlets, both **static code** and **dynamic code** are put together. In JSP, they are separated.
3. In JSP, the **static content** and **dynamic content** is separated. Static content is written in HTML and dynamic content in JSP. As much of the response comprises of static content (nearly 70%) only, the JSP file more looks as a HTML file.
4. A Web site comes with both static content and dyanamic content. Static content involves displaying company logo, company building, company lietrature like company present and back history etc. which will never change. This is done by a (Graphic) **Designer** with the knowledge of just HTML and some Photoshop.
   The dynamic content changes every day and every minute like displaying running share prices, clearing client doubts and accessing database to fulfil client requested data etc. This is done by by a **JSP Programmer**..
5. Both **presentation layer** and **business logic layer** put together in Servlets. In JSP, they can be separated with the usage of JavaBeans.
6. JSP can easily be integrated with JavaBeans.
7. JSP is much used in frameworks like Sturts etc.
8. With JSP, Programmer can build **custom tags** that can be called in JavaBeans directly. Servlets do not have this advantage. Reusability increases with tag libraries and JavaBean etc.
9. Writing alias name in <**url-pattern**> tag of **web.xml** is **optional** in JSP but mandatory in Servlets.

4. Explain about the Lifecycle of a jsp.

Ans:Life cycle of JSP
A Java Server Page life cycle is defined as the process started with its creation which later translated to a servlet and afterward servlet lifecycle comes into play. This is how the process goes on until its destruction.



Following steps are involved in JSP life cycle:

- Translation of JSP page to Servlet
- Compilation of JSP page(Compilation of JSP into test.java)
- Classloading (test.java to test.class)
- Instantiation(Object of the generated Servlet is created)
- Initialization(jspInit() method is invoked by the container)
- Request processing(_jspService()is invoked by the container)
- JSP Cleanup (jspDestroy() method is invoked by the container)

We can override jspInit(), jspDestroy() but we can't override _jspService() method.

Translation of JSP page to Servlet :

This is the first step of JSP life cycle. This translation phase deals with Syntactic correctness of JSP. Here test.jsp file is transllated to test.java.

Compilation of JSP page :

Here the generated java servlet file (test.java) is compiled to a class file (test.class).

Classloading :

Servlet class which has been loaded from JSP source is now loaded into container.

Instantiation :

Here instance of the class is generated. The container manages one or more instance by providing response to requests.

Initialization :

jspInit() method is called only once during the life cycle immediately after the generation of Servlet instance from JSP.

Request processing :

_jspService() method is used to serve the raised requests by JSP.It takes request and response object as parameters.This method cannot be overridden.

JSP Cleanup :

In order to remove the JSP from use by the container or to destroy method for servlets jspDestroy()method is used. This method is called once, if you need to perform any cleanup task like closing open files, releasing database connections jspDestroy() can be overridden.

5. Describe various steps that are needed for accessing a database
from a JSP page?
Ans:

Declaring Variables: Java is a strongly typed language which means, that variables must be explicitly declared before use and must be declared with the correct data types. In the above example code we declare some variables for making connection. Theses variables are-

```
Connection con=null;
ResultSet rst=null;
Statement stmt=null;
```

The objects of type `Connection`, `ResultSet` and `Statement` are associated with the Java `sql`. "con" is a Connection type object variable that will hold Connection type object. "rst" is a ResultSet type object variable that will hold a result set returned by a database query. "stmt" is a object variable of Statement .Statement Class methods allow to execute any query.

Connection to database: The first task of this programmer is to load database driver. This is achieved using the single line of code :-

```
String driver = "org.gjt.mm.mysql.Driver";
Class.forName(driver).newInstance();
```

The next task is to make a connection. This is done using the single line of code :-

```
String url="jdbc:mysql://localhost/books?user=<userName>&password=<password>";
con=DriverManager.getConnection(url);
```

When url is passed into getConnection() method of DriverManager class it returns connection object.

Executing Query or Accessing data from database:

This is done using following code :-

```
stmt=con.createStatement(); //create a Statement object
rst=stmt.executeQuery("select * from books_details");
```

stmt is the Statement type variable name and **rst** is the RecordSet type variable. A query is always executed on a Statement object.
A Statement object is created by calling createStatement() method on connection object **con.**

The two most important methods of this Statement interface are executeQuery() and executeUpdate(). The executeQuery() method executes an SQL statement that returns a single ResultSet object. The executeUpdate() method executes an insert, update, and delete SQL statement. The method returns the number of records affected by the SQL statement execution.

After creating a Statement ,a method executeQuery() or executeUpdate() is called on Statement object **stmt** and a SQL query string is passed in method executeQuery() or executeUpdate().
This will return a ResultSet **rst** related to the query string.

Reading values from a ResultSet:

```
while(rst.next()){

  %>

  <tr><td><%=no%></td><td><%=rst.getString("book_name")%></td><td><%=rst.getStri
ng("author")%></td></tr>

 <%

}
```

The database in example consists of a single table of three columns or fields. The database name is "books" and it contains information about **books names & authors**.

Table:books_details

ID	Book Name	Author
1.	Java I/O	Tim Ritchey
2.	Java & XML,2	Brett

	Edition	McLaughlin
3.	Java Swing, 2nd Edition	Dave Wood, Marc Loy,

Start MYSQL prompt and type this SQL statement & press Enter-

```
MYSQL>CREATE DATABASE `books` ;
```

This will create "books" database.
Now we create table a table "books_details" in database "books".

```
MYSQL>CREATE TABLE `books_details` (
`id` INT( 11 ) NOT NULL AUTO_INCREMENT ,
`book_name` VARCHAR( 100 ) NOT NULL ,
`author` VARCHAR( 100 ) NOT NULL ,
PRIMARY KEY ( `id` )
) TYPE = MYISAM ;
```

This will create a table "books_details" in database "books"

The `ResultSet` represents a table-like database result set. A `ResultSet` object maintains a cursor pointing to its current row of data. Initially, the cursor is positioned before the first row. Therefore, to access the first row in the `ResultSet`, you use the `next()` method. This method moves the cursor to the next record and returns `true` if the next row is valid, and `false` if there are no more records in the `ResultSet` object.

Other important methods are `getXXX()` methods, where `XXX` is the data type returned by the method at the specified index, including `String`, `long`, and `int`. The indexing used is 1-based. For example, to obtain the second column of type String, you use the following code:

```
resultSet.getString(2);
```

You can also use the `getXXX()` methods that accept a column name instead of a column index. For instance, the following code retrieves the value of the column LastName of type `String`.

```
resultSet.getString("book_name");
```

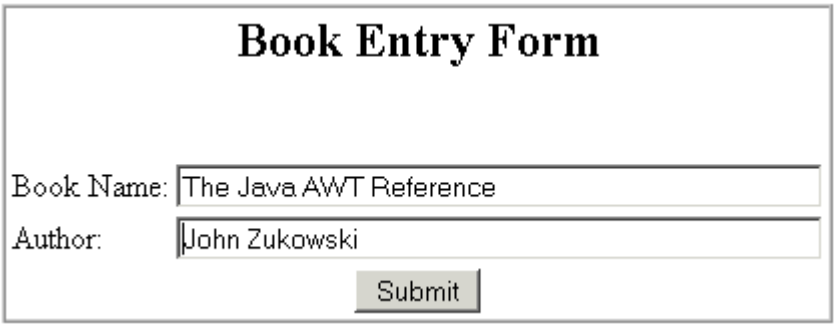
The above example shows how you can use the `next()` method as well as the `getString()` method. Here you retrieve the 'book_name' and 'author' columns from a table called 'books_details'. You then iterate through the returned `ResultSet` and print all the book name and author name in the format " book name | author " to the web page.

Summary:

This article presents JDBC and shows how you can manipulate data in a relational database from your JSP page. To do this, you need to use the java.sql package: `DriverManager`, `Connection`, `Statement`, and `ResultSet`. Keep in mind, however, that this is only an

introduction. To create a Web application, you need JDBC to use more features such as prepared statements and connection pooling.

When you click on the above link a Book Entry Form will open



Fill the book name and author fields and press Submit button. A page will open and show a table of book name and authors like...

Books List

S.No	Book Name	Author
1	Java I/O	Tim Ritchey
2	Java & XML,2 Edition	Brett McLaughlin
3	Java Swing, 2nd Edition	Dave Wood, Marc Loy, James Elliott, Brian Cole, Robert Eckstein
4	Java Cookbook, 2nd Edition	Ian F. Darwin
5	Java Web Services Unleashed	Robert J Brunner, Frank Cohen
6	Core Java Data Objects	Sameer Tyagi, Michael Vorburger
7	Java in a Nutshell	David Flanagan
8	Java Web Services in a Nutshell	Kim Topley
9	The Java AWT Reference	John Zukowski

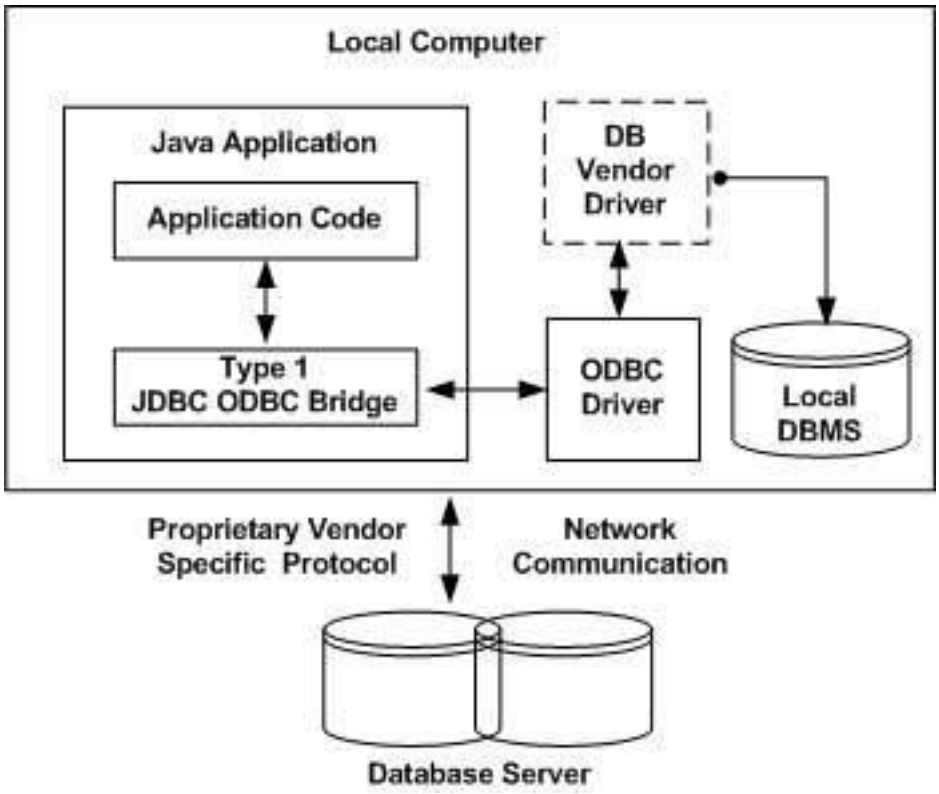
6. Explain JDBC type 1 to 4 drivers.

Ans: JDBC drivers implement the defined interfaces in the JDBC API, for interacting with your database server.
For example, using JDBC drivers enable you to open database connections and to interact with it by sending SQL or database commands then receiving results with Java.
The *Java.sql* package that ships with JDK, contains various classes with their behaviours defined and their actual implementaions are done in third-party drivers. Third party vendors implements the *java.sql.Driver* interface in their database driver.
JDBC Drivers Types
JDBC driver implementations vary because of the wide variety of operating systems and hardware platforms in which Java operates. Sun has divided the implementation types into four categories, Types 1, 2, 3, and 4, which is explained below −

Type 1: JDBC-ODBC Bridge Driver

In a Type 1 driver, a JDBC bridge is used to access ODBC drivers installed on each client machine. Using ODBC, requires configuring on your system a Data Source Name (DSN) that represents the target database.

When Java first came out, this was a useful driver because most databases only supported ODBC access but now this type of driver is recommended only for experimental use or when no other alternative is available.

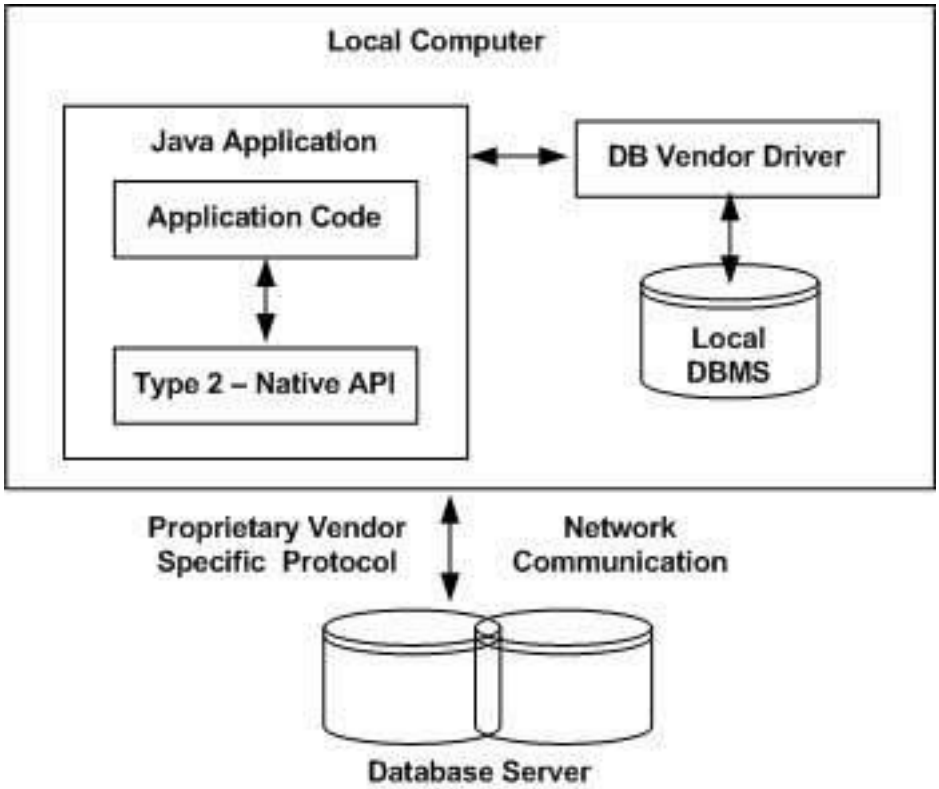


The JDBC-ODBC Bridge that comes with JDK 1.2 is a good example of this kind of driver.

Type 2: JDBC-Native API

In a Type 2 driver, JDBC API calls are converted into native C/C++ API calls, which are unique to the database. These drivers are typically provided by the database vendors and used in the same manner as the JDBC-ODBC Bridge. The vendor-specific driver must be installed on each client machine.

If we change the Database, we have to change the native API, as it is specific to a database and they are mostly obsolete now, but you may realize some speed increase with a Type 2 driver, because it eliminates ODBC's overhead.

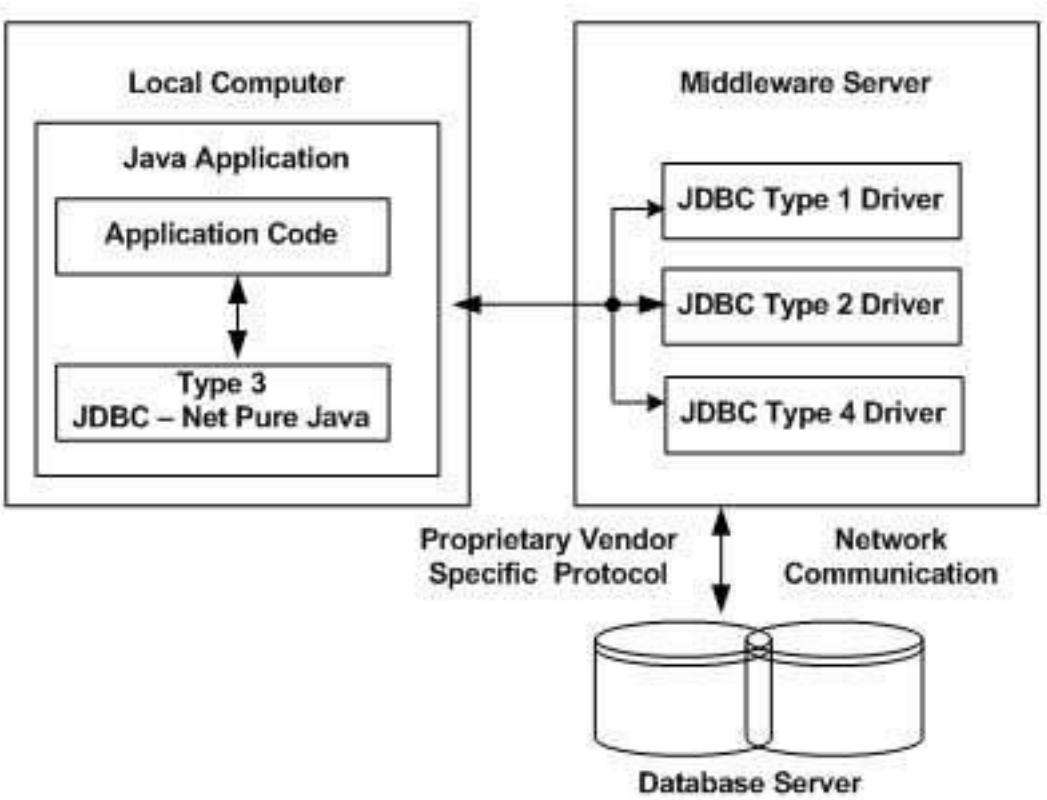


The Oracle Call Interface (OCI) driver is an example of a Type 2 driver.

Type 3: JDBC-Net pure Java

In a Type 3 driver, a three-tier approach is used to access databases. The JDBC clients use standard network sockets to communicate with a middleware application server. The socket information is then translated by the middleware application server into the call format required by the DBMS, and forwarded to the database server.

This kind of driver is extremely flexible, since it requires no code installed on the client and a single driver can actually provide access to multiple databases.



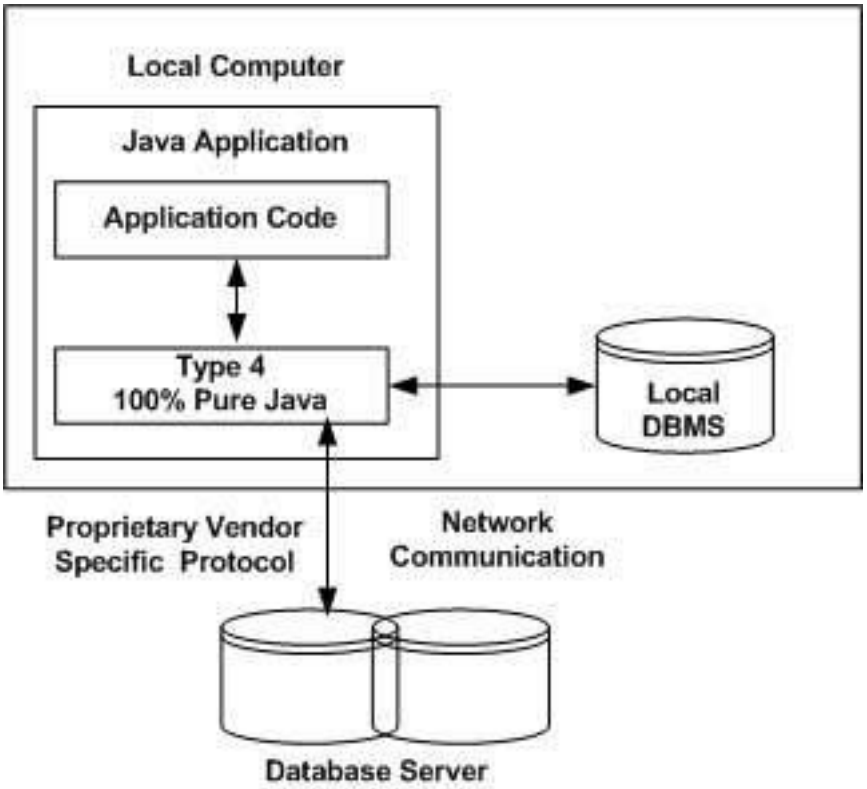
You can think of the application server as a JDBC "proxy," meaning that it makes calls for the client application. As a result, you need some knowledge of the application server's configuration in order to effectively use this driver type.

Your application server might use a Type 1, 2, or 4 driver to communicate with the database, understanding the nuances will prove helpful.

Type 4: 100% Pure Java

In a Type 4 driver, a pure Java-based driver communicates directly with the vendor's database through socket connection. This is the highest performance driver available for the database and is usually provided by the vendor itself.

This kind of driver is extremely flexible, you don't need to install special software on the client or server. Further, these drivers can be downloaded dynamically.



MySQL's Connector/J driver is a Type 4 driver. Because of the proprietary nature of their network protocols, database vendors usually supply type 4 drivers.

Which Driver should be Used?

If you are accessing one type of database, such as Oracle, Sybase, or IBM, the preferred driver type is 4.

If your Java application is accessing multiple types of databases at the same time, type 3 is the preferred driver.

Type 2 drivers are useful in situations, where a type 3 or type 4 driver is not available yet for your database.

The type 1 driver is not considered a deployment-level driver, and is typically used for development and testing purposes only.

Unit V

1. What the the different control structures in PHP.

Ans:

1. If..else
2. If..elseif
3. Switch
4. For loop
5. While loop
6. Do while loop
7. For each loop
8. Break statement
9. Continue statement

2. Explain Arrays in PHP with its different types with suitable examples.

Ans:
In PHP, there are three types of arrays:

- **Indexed arrays** - Arrays with a numeric index
- **Associative arrays** - Arrays with named keys
- **Multidimensional arrays** - Arrays containing one or more arrays

PHP Indexed Arrays
There are two ways to create indexed arrays:
The index can be assigned automatically (index always starts at 0), like this:
$cars = array("Volvo", "BMW", "Toyota");
or the index can be assigned manually:
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
The following example creates an indexed array named $cars, assigns three elements to it, and then prints a text containing the array values:
Example

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo"I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

PHP Associative Arrays
Associative arrays are arrays that use named keys that you assign to them.
There are two ways to create an associative array:
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
or:
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
The named keys can then be used in a script:
Example

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
echo"Peter is " . $age['Peter'] . " years old.";
?>
```

PHP - Multidimensional Arrays
A multidimensional array is an array containing one or more arrays.
PHP understands multidimensional arrays that are two, three, four, five, or more levels deep.
However, arrays more than three levels deep are hard to manage for most people.

Example

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

3. Discuss any five String functions in PHP.

Ans: Basic PHP String Functions

In this section, we will discuss a few basic string functions which are most commonly used in PHP scripts.

1. Getting length of a String

PHP has a predefined function to get the length of a string. Strlen() displays the length of any string. It is more commonly used in validating input fields where the user is limited to enter a fixed length of characters.

Syntax

```
Strlen(string);
```

Example

```
<?php
echo strlen("Welcome to Cloudways");//will return the length of given string
?>
```

Output

```
20
```

2. Counting of the number of words in a String

Another function which enables display of the number of words in any specific string is str_word_count(). This function is also useful in validation of input fields.

Syntax

```
Str_word_count(string)
```

Example

```
<?php
echo str_word_count("Welcome to Cloudways");//will return the number of words in a string
?>
```

Output

```
3
```

3. Reversing a String

Strrev() is used for reversing a string. You can use this function to get the reverse version of any string.

Syntax

```
Strev(string)
```

Example

```
<?php
echo strrev("Welcome to Cloudways");// will return the string starting from the end
?>
```

Output

```
syawduolC ot emocleW
```

4. Finding Text Within a String

Strpos() enables searching particular text within a string. It works simply by matching the specific text in a string. If found, then it returns the specific position. If not found at all, then it will return "False". Strops() is most commonly used in validating input fields like email.

Syntax

```
Strpos(string,text);
```

Example

```
<?php
echo strpos("Welcome to Cloudways","Cloudways");
?>
```

Output

11

5. Replacing text within a string

Str_replace() is a built-in function, basically used for replacing specific text within a string.

Syntax

Str_replace(string to be replaced,text,string)

Example

```
<?php
echo str_replace(“cloudways”, “the programming world”, “Welcome to cloudways”);
?>
```

Output

Welcome to the programming world

4. Explain the data types in PHP.

Ans: Data Types defines the type of data a variable can store. PHP allows eight different types of data types. All of them are discussed below. The first five are called simple data types and the last three are compound data types:

1. **Integer** : Integers hold only whole numbers including positive and negative numbers, i.e., numbers without fractional part or decimal point. They can be decimal (base 10), octal (base 8) or hexadecimal (base 16). The default base is decimal (base 10). The octal integers can be declared with leading 0 and the hexadecimal can be declared with leading 0x. The range of integers must lie between -2^31 to 2^31.Example:

```
1.<?php

  // decimal base integers
  $deci1 = 50;
  $deci2 = 654;

  // octal base integers
  $octal1 = 07;

  // hexadecimal base integers
  $octal = 0x45;

  $sum = $deci1 + $deci2;
  echo $sum;

  ?>

2.
```

Output:
704

2. **Double**: Can hold numbers containing fractional or decimal part including positive and negative numbers. By default, the variables add a minimum number of decimal places. Example:

```
1.<?php

  $val1 = 50.85;
  $val2 = 654.26;

  $sum = $val1 + $val2;

  echo $sum;

  ?>

2.
```

Output:
705.11

3. **String** : Hold letters or any alphabets, even numbers are included. These are written within double quotes during declaration. The strings can also be written within single quotes but it will be treated differently while printing variables. To clarify this look at the example below.Example:

```
1.<?php

  $name = "Krishna";
  echo "The name of the Geek is $name \n";
  echo 'The name of the geek is $name';

  ?>

2.
```

Output:
The name of the Geek is Krishna

The name of the geek is $name

4. **NULL**: These are special types of variables that can hold only one value i.e., NULL. We follow the convention of writing it in capital form, but its case sensitive.
Example:

```
1.<?php

  $nm = NULL;
  echo $nm;    // This will give no output

  ?>

2.      33
```

5. **Boolean**: Hold only two values, either TRUE or FALSE. Successful events will return true and unsuccessful events return false. NULL type values are also treated as false in Boolean. Apart from NULL, 0 is also consider as false in boolean. If a string is empty then it is also considered as false in boolean data type.
Example:

```
1.<?php

  if(TRUE)
     echo "This condition is TRUE";
  if(FALSE)
     echo "This condition is not TRUE";
  ?>

2.

  Output:
  This condition is TRUE
```

This condition is not TRUE

6. **Arrays**: Array is a compound data-type which can store multiple values of same data type. Below is an example of array of integers.

```
1.<?php

  $intArray = array( 10, 20 , 30);

  echo "First Element: $intArray[0]\n";
  echo "Second Element: $intArray[1]\n";
  echo "Third Element: $intArray[2]\n";

  ?>

2.

  Output:
  First Element: 10
```

Second Element: 20
Third Element: 30

7. **Objects**: Objects are defined as instances of user defined classes that can hold both values and functions. This is an advanced topic and will be discussed in details in further articles.
8. **Resources**: Resources in PHP are not an exact data type. These are basically used to store references to some function call or to external PHP resources. For example, consider a database call. This is an external resource.

5. How can you handle forms in PHP? Explain with suitable program..

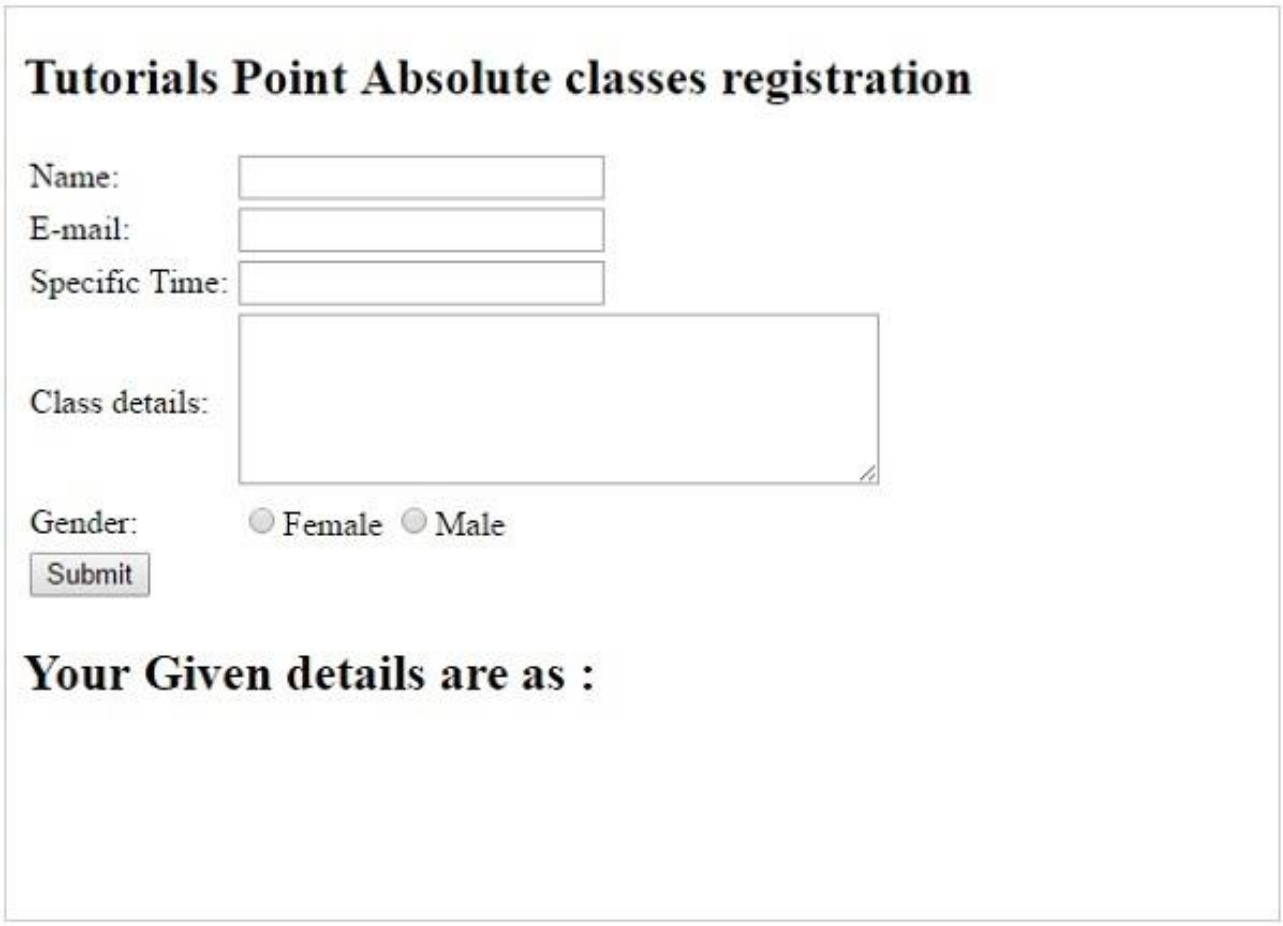
Ans:

A Document that containing black fields, that the user can fill the data or user can select the data.Casually the data will store in the data base
Example
Below example shows the form with some specific actions by using post method.

```
<html>

<head>
<title>PHP Form Validation</title>
</head>

<body>
<?php

// define variables and set to empty values
        $name = $email = $gender = $comment = $website ="";

if($_SERVER["REQUEST_METHOD"]=="POST"){
            $name = test_input($_POST["name"]);
            $email = test_input($_POST["email"]);
            $website = test_input($_POST["website"]);
            $comment = test_input($_POST["comment"]);
            $gender = test_input($_POST["gender"]);
}

function test_input($data){
            $data = trim($data);
            $data = stripslashes($data);
            $data = htmlspecialchars($data);
return $data;
}
?>

<h2>Tutorials Point Absolute classes registration</h2>

<formmethod="post"action="/php/php_form_introduction.htm">
<table>
<tr>
<td>Name:</td>
<td><inputtype="text"name="name"></td>
</tr>

<tr>
<td>E-mail:</td>
<td><inputtype="text"name="email"></td>
</tr>

<tr>
<td>Specific Time:</td>
<td><inputtype="text"name="website"></td>
</tr>
```

```
<tr>
<td>Class details:</td>
<td><textareaname="comment"rows="5"cols="40"></textarea></td>
</tr>

<tr>
<td>Gender:</td>
<td>
<inputtype="radio"name="gender"value="female">Female
<inputtype="radio"name="gender"value="male">Male
</td>
</tr>

<tr>
<td>
<inputtype="submit"name="submit"value="Submit">
</td>
</tr>
</table>
</form>

<?php
      echo "<h2>Your Given details are as :</h2>";
      echo $name;
      echo "<br>";

      echo $email;
      echo "<br>";

      echo $website;
      echo "<br>";

      echo $comment;
      echo "<br>";

      echo $gender;
?>

</body>
</html>
```

It will produce the following result −



6. How can you establish a connection to database in PHP. What are the functions involved in it?

```
<?php

// Five steps to PHP database connections:

// 1. Create a database connection
// (Use your own servername, username and password if they are different.)
// $connection allows us to keep referring to this connection after it is established
$connection =mysql_connect("localhost","root","myPassword");
if (!$connection) {
```

```
die("Database connection failed: ".mysql_error());
}

// 2. Select a database to use
$db_select =mysql_select_db("widget_corp",$connection);
if (!$db_select) {
die("Database selection failed: ".mysql_error());
}

// 3. Perform database query
$result =mysql_query("SELECT*FROM subjects", $connection);
if (!$result) {
die("Database query failed: ".mysql_error());
}

// 4. Use returned data
while ($row =mysql_fetch_array($result)) {
echo $row["menu_name"]."".$row["position"]."<br />";
}

// 5. Close connection
mysql_close($connection);
```

1. Create connection

It is very important when you want to start a dynamic website to create connection with your SQL (we are talking about mysql) by using mysql_connect() function. In mysql_connect() arguments should be string and it's important to use die() function with mysql_error() function to check if there is a problem with the connection or not.

2. Select database

Now we have to select the database which we are creating and saving our data by using mysql_select_db() function which the arguments are the name of database and connection we made earlier.

3. Perform database query

In this step we have to select out data from database and bring it into our web page. The best decision to use mysql_query() function which it will Send a MySQL query with 2 arguments as displayed in the above code.

4. Use return data

To display the result on your web page, you have to use while() loop function in addition to mysql_fetch_array() function which fetch a result row as an associative array, a numeric array, or both.

5. Close connection

To close connection, it is better to use mysql_close() function to close MySQL connection. This is a very important function as it closes the connection to the database server. Your script will still run if you do not include this function. And too many open MySQL connections can cause problems for your account. This it is a good practice to close the MySQL connection once all the queries are executed.