Internet Technologies Notes

Contributor: Riya Goel

[KMV (DU)]

Computer Science Notes

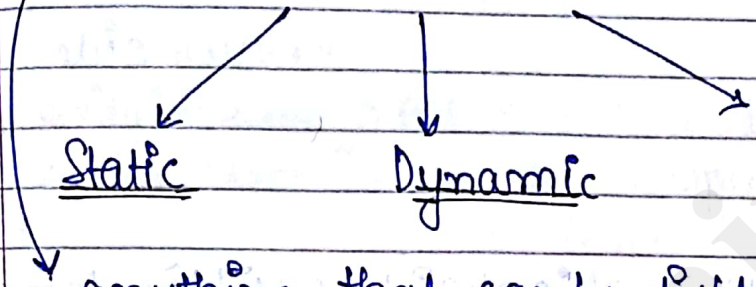
13/8/18

CLASSTIME | Page No.
Date / /

JavaScript

Core Java - 8 marks. } weightage.
JDBC - 10 marks }

- Web Pages



anything that can be displayed on browser in terms of document or file.

* Static - Something that is picked up i.e content without any understanding is painted on page.
(no change)

after this, everything gets flushed out from memory.

(changes can be encorporated) Dynamic - info. changes after rendering/ refreshing browser.

Interactive - User can simultaneously do changes. (Content validation)
(user Interaction)

eg :- Name : [ ] → only alphabets allowed.

Submit

To implement this, we have 2 cases

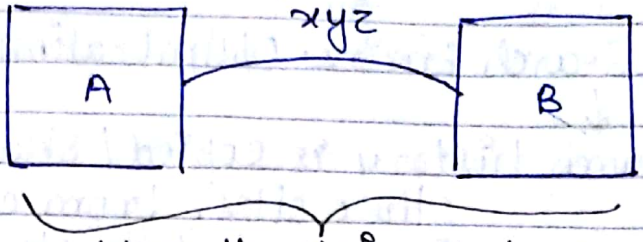
Errors/ I/P can be checked while user is entering data.
↓
Client side scripting.

All data is entered & then by clicking submit, we check for valid I/P.
↓
server side scripting.

NOTE: JavaScript is clientside scripting language.

Whenever we load a website, it takes a lot of time. Why?
↓
First of all it doesn't know who is the client. It checks in its cache for the client.
And to know which client requests a webpage, server uses cookies.
↓
random string
Just to authenticate/ differentiate that user is first time or ntime.

String of cookies generally flushes out after 30 days (by default).



Here the string to be searched is xyz.
If we search for it again, webpage opens fast.

But if we delete this string xyz, then the user is newly identified and thus webpage takes time in loading.

→ When we search for IPV4 config.
↓
{can be manually changed} ← Static IP } by default.
dynamic IP. }

→ DNS of Google server
↓
{ 4 4 4 4 }
{ 8.8.8.8 }

NOTE: JS objective is to make webpages more interactive i.e modifiable.
↓
Server basically maintains a priority based upon choices of user, i.e max. searched things shown on top & min. at bottom.

CLASSTIME | Page No.
Date / /

NOTE:- SEO - Search Engine Optimization (URL)
↓
Search history is copied / shared to other sites, from company's webpage that is searched to other companies.

→ How to create webpages :-
↓
HTML (extended version - XML)

Hyper Text
(i.e links)

HTML → webpages + script
↓ ↓
(use of tags) Interactive webpages.

<head> </head>
<body> </body>
To define tags.

{Script tag can be defined anywhere in HTML acc. to need}.

To define script ↓ {need to explicitly specify}.

<head>
<script = "JavaScript"> </script> </head>
<body> </body>

• DOM :-
(Document

? Why do

as

Defau

→ If w

NOTE: There

NOTE: Initi

• Jav

→ Ad

CLASSTIME | Page No.
Date / /

- DOM :- Model that uses script & webpages. (Document Object Model).

? Why do we need to explicitly use script as JavaScript ?

as this script is Netscape (designed by Netscape).

Default script → JavaScript → Netscape scripts. (Google, mozilla, safari).

→ If we don't explicitly specify, it interprets default script.

NOTE:- There is no relation in Java & Javascript.

- Both deals with objects (similarity). (compiled language)
- Object oriented scripting language (interpreted language)

NOTE:- Initially, VBScript was default script for Internet Explorer.

- JavaScript - It is an object oriented language that allows creation of interactive webpages

→ Advantages of JS :-

(1) It is an interpreted language {no compilation phase req}

(2) Procedural capabilities - Loops, conditions, checking can be implemented.

(3) User dependent programming. {depends on user}.

imp. (4) Platform independent. {can run on any Browser}. (client)

→ File on Browser can vary by extension. But platform independent
(Windows → .exe
Linux → .dz).

NOTE:- Inspect element - used for a debugging tool in browser.

* Data Types :-

(1) Number (integer, floating pt. and Nan).
(2) Boolean (True / False)
(3) String (' ' or " ")
(4) Null. (explicitly a DT)

Nan ↙ not a number (eg:- abc10) mix of both number & alphabet.

NOTE:- It we

* Type Ca

eg:

eg:

→ In

→ Fl

NOTE:- Othe

* Crea

CLASSTIME | Page No.
Date / /

NOTE:- If we check for Integer value corresponding to ↴ (automatic conversion).

True - 1
False - 0.

* Type Casting :- (depending on order of DT).

eg :- "Hello" + 10 ⟶ Hello10
↙
Depending upon predefined DT, typecasting takes place.

eg :- 20 + "10" ⟶ 30
↙
as previous type was Int.

⟶ Int + String
⟶ Float + String. } 2 types of type casting (Default Types)

NOTE:- Other DT have to be explicitly typecasted.

* Creation of variables in JS :-
↓
Using var keyword, we can define any DT (derived / primitive)

var a = 10; (can be int / float).
↙
automatically gets converted to a number } specify in

terms of broad category; don't use a is integer X}.

var a = "abc";
var a = 10.5;

* Arrays :-

array a = new Array();

array → keyword
a → name of Aarry.
new Array() → class available in JS

(OR)

{array a = new Array(length);}

doesn't work but is a version in JS.

length → size of array

eg:- a = new Array (10);

0-9

But a[9] = 20; —
a[10] = 30; —

This is acceptable as in JS by default, array grows dynamically.

Dense A
created
value
Der
a =
2t
<htm
<he
<sc
Considers
I eg

(only for JS)

CLASSTIME | Page No.
Date / /

- Dense Array :- An array that has been created and initialized with a specific value simultaneously, then it is known as Dense array.

```
a = new Array ("10", "20", "30");
```

It is like all other arrays in all languages but in JS it is provided a specific name.

```
<html>
 <head>
 <script type = "Javascript">
        var a = 10;
    document.write (a);
 </script> </head> </html>
```

To display values

func. for o/p.

Considers screen space as a document on which o/p is to be written.

Document → space / webpage.

I eg:-
```
<script>
    a = new Array (2);
    document.write (a[0]);
    document.write (a[1]);
</script>
```

{ To output an array using looping }

II using join()

```
a[0] = 10;
a[1] = 20;
a[3] = 30;
```

```
ab = a.join();
document.write(a);
```

ab = new Array (a);

This is also an array.

O/P → 10 20 30.

Multi Dimensional Arrays

We can define multiple DT in arrays and also an array within an array.

```
eg:- a = new Array (10, "a",
         20, "d", new Array(10,20));
         0   1   2  3   4
```

```
d.w(a[0]);  → 10
d.w(a[1]);  → a
d.w(a[4][0]); → 10
d.w(a[4][0] + a[4][1])
```

Here O/P is 30 as they have been concatenated.

→ 10 20 only possible when both string.

NOTE:- Join() only not m

NOTE:- All types o

Q Write a J

<he
<s

NOTE:-
In case of null, we don't have m memory location

eg:-

Va

Othe

Page No.
Date / /

30.

DT. In arrays
ithin an array.

0 1
(10, "a",
new 4
Array(10,20));

→ 10
→ a
]); → 10

[4][1])
only possible
0} when both strings.

CLASSTIME Page No.
Date / /

NOTE:- Join() only deals with one-D arrays & not multidimensional arrays.

NOTE:- All types of looping is possible in JS.

Q Write a JS code to display a no. from 1 to 10.

```
<head>
<script>
    var i;
for( i=0; i<10; i++)
    d.w(i);
```

NOTE:- {In case of null, we don't have new memory locations}.

var i = null; → i [null]
i = 10; override
(no new memory).

eg:- var a = "Thursday";
↓
To show it is weekday or weekend.
we can use if-else

Otherwise, best way is through ternary ↓ operator.

Thursday ? "Weekday" : "Weekend".

CLASSTIME | Page No.
Date / /

* Operators in JS :-

(1) Arithmetic (+, -, %, /)

(2) Logical (&&, ||, !)

(3) Comparison ↓

(equals) == — compares values & not DT

eg :- 10 == "10" → For equals case, it automatically converts DT and thus value is 10 ∴ returns True.

=== (strictly equals) — checks both DT and value

eg :- 10 === "10" ↓ returns false.

→ Similarly, != (not equals)

!== (strictly not equals)

(4) Ternary Operator.

(5) delete () → delete a[4] // deletes value at index 4

(6) new() → used for allocation of objects.

Date / /

')

= (strictly equals)

hecks both DT and value

eg :- 10 = = = "10"
↓
returns false.

! = = (strictly not equals)

// deletes value at index 4

ction
jects.

CLASSTIME | Page No.
Date / /

(7) void → Generally it is a func. but in JS it is an operator.
↓
doesn't return value.

→ To find length of array ↓

We have a property, not a func.

arr.length
↓ ↓
name of array. returns property of length.

- Built In Functions :-

(1) parseInt() ↓

parseInt(abc);

Although in Java, it gives an error, but in JS it returns NaN. (not a no.)

eg :- parseInt(10abc);

It only returns the integer value of associated String.

O/P = 10.

var b = parseFloat(abc10.5)
↓
(10.5)

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

eg:- parseInt(abc);

Here it will not return 0 X
It always return NaN in int or float.

(2) parseFloat().

(3) eval() :- Automatically gives value when expression is fed in parameters.
evaluation.

var a = eval("10 * (0+5)");

only works with expressions.

Calculates on basis of BODMAS.

NOTE:- All these functions can be directly passed in document.write().

eg:- parseInt(abc2)

only returns 1 and not 12 or 2 X
{ first entered no. }.

Whenever we have '+' (concatenation) we have case of typecasting.

eg:- pars

"10"+20 —

But parse

* Dialouge Bo

(1) Alert Dialou
don't wo

To make an

→

<script>

wr

We

→ eg :- parseInt ("10" + 20);

↓

O/P → 10

"10" + 20 → this gets typecasted into String 1020.
But parseInt automatically converts it into an integer
∴ O/P is not NaN.

* Dialouge Box :-
(default design)

(1) Alert Dialouge Box - Things at backend don't work. It is like a pop-up box.

To make an alert ↓

→ alert ("Hello");

```
<script>
    alert ("Goodmorning");
```

↓

We are not using document ⊙ write as we are displaying nothing on webpage.
We display it on another dialouge box.

→ Alert Dialouge Box.

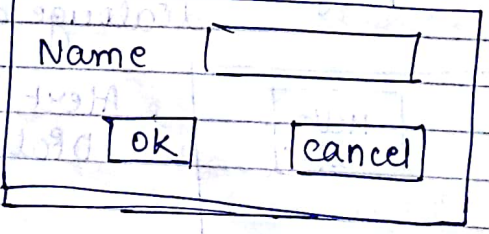
NOTE:

- If whole of code is in JSP, then extension is .jsp and can be included as a stylesheet in program.
- Using JSP with HTML, extension → .html.
- We can display multiple dialouge boxes one over other.

```
<script>
    alert ("Hello");
    alert ("GM");
</script>
```

When we create alert dialouge box, automatically a box and Ok button comes. (-We can't change font of Ok).

(2) Prompt Dialouge Box -



can fetch user I/P.

(3) Confirm Dia

If ok is pressed, t execution to some page

NOTE: In alert

TUTORIALSDUNIYA.COM

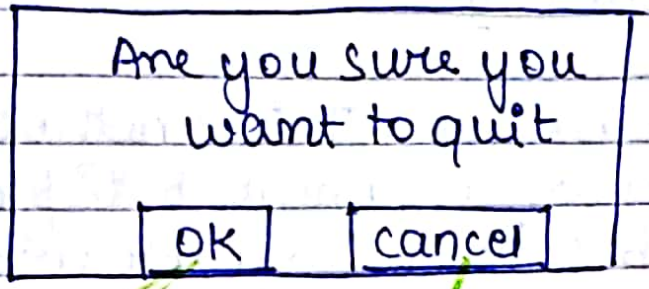
Get FREE Compiled Books, Notes, Programs, Question Papers with Solution etc of the following subjects at https://www.tutorialsduniya.com

- C and C++
- Java
- Data Structures
- Computer Networks
- Android Programming
- PHP Programming
- JavaScript
- Java Server Pages
- Python
- Microprocessor
- Artificial Intelligence
- Machine Learning
- Computer System Architecture
- Discrete Structures
- Operating Systems
- Algorithms
- DataBase Management Systems
- Software Engineering
- Theory of Computation
- Operational Research
- System Programming
- Data Mining
- Computer Graphics
- Data Science

- DU Programs: https://www.tutorialsduniya.com/programs
- TutorialsDuniya App: http://bit.ly/TutorialsDuniyaApp
- C++ Tutorial: https://www.tutorialsduniya.com/cplusplus
- Java Tutorial: https://www.tutorialsduniya.com/java
- JavaScript Tutorial: https://www.tutorialsduniya.com/javascript
- Python Tutorial: https://www.tutorialsduniya.com/python
- Kotlin Tutorial: https://www.tutorialsduniya.com/kotlin
- JSP Tutorial: https://www.tutorialsduniya.com/jsp

CLASSTIME | Page No.
Date / /

(3) Confirm Dialouge Box -



If OK is pressed, then execution goes to some other page.

If we press Cancel, execution is still on last line / last page.

NOTE:- In alert, we can also have

↓

alert (User I/P + " ");

for fetching user I/P.

eg:- var a = "Riya";
alert (a + "Hello");

CLASSTIME | Page No.
Date / /

Dialouge Boxes

(1) Alert Dialouge Box - This method takes a string as an argument & displays an alert dialouge Box in the browser window when invoked by corresponding JavaScript.

→ To Generate Alert Dialouge Box ↴

```
alert ("          ");
```

i/p as a string.

→ popup subwindow.

→ Functionality :- Alert Dialouge Box can be used for following.

(1) A message is displayed to the user when incorrect info. is entered.
(2) For any invalid result as the output of a calculation.
(3) Any warning message for a specific service that is not available for that given time.

eg :-

Here alert dialouge box can be used to check by user.

One alert box is generated one at a time.

If we

it

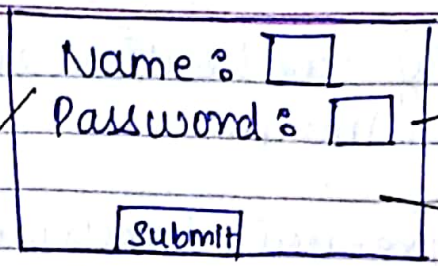
} used to

(2) Promp
Instan
displ

- It also
accep

CLASSTIME | Page No.
Date / /

eg:-



① alert → login success

② login unsuccess.

Here alert dialouge box can be used to check I/P by user.

```
alert ("Hi");
alert ("Hello");
alert ("bye");
```

one alert box is generated one at a time.

can't show multiple alert boxes.

doesn't work outside <script>

If done so, it displays a plain text (like HTML) (browser can't interpret it)

If we press OK of Hi, then only execution jumps to next stmt., otherwise it stucks to that instn only.

{used to display user info.} → Alert

(2) Prompt Dialouge Box - Prompt method instantiates the prompt Dialouge Box which displays a specific message

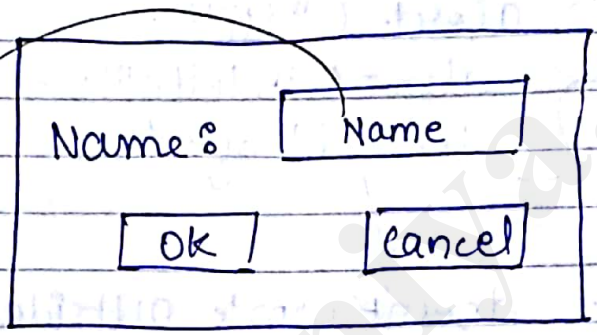
- It also provides single data entry which accepts user input.

CLASSTIME | Page No.
Date / /

- Functionality can be discussed in following manner

(1) displays a pre-defined message
(2) displays a text box & accepts user I/P
(3) displays the [OK] and [Cancel] button.

Prompt DB generates a single textbook.



Value overrides if new value is entered & pressed Okay.

prompt ("Enter your name", "Name");

value written beside textbox (label)

placeholder value in a textbox

If we want no placeholder value

prompt ("——", "");

passing empty string.

Hi : [Name] → although this name flushes out when we type our own value.

But in prompt we first need to use backspace, erase value & then write our value.

Hi : [Name]

OK

Hi : Name

O/P is this as we didn't erase value of textbox.

cancel.

displays a null value when cancel is clicked.

To fetch value entered by user

↓

var(a) = prompt(" ", " ");

Q Generate a prompt Box. Take a user I/P and compare whether it's weekday or weekend.

<Script>

var a = prompt("Enter day", "Day");

if (a == "Sat" || a == "Sun")

Include cond. for invalid i/p.

```
alert("weekend");
else
alert("weekday");
```

var (a) = Evaluate [10*10+5/20] [Ok] [Cancel]

This gets passed in eval & it evaluates expression.

NOTE:- If we are using prompt, then we can't use HTML input tags.

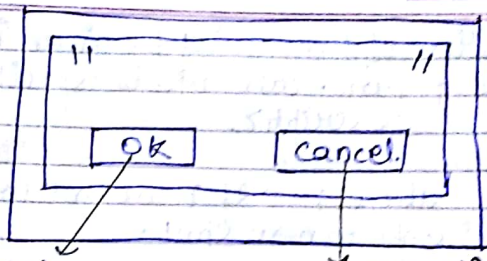
NOTE:- If not prompt, by default <input type=-->

(3) Confirm Dialouge Box :-

- This dialouge box displays a predefined message with a Ok and Cancel button.
- Clicking on the [Ok] causes a true value to be passed otherwise false value is passed and halts the execution.

Syntax-

```
confirm("Are you sure you want to continue / quit");
```



It causes the working at backend and thus we are able to execute other stmts. (returns true).

execution is on confirm dialouge box i.e same state (no next stmt of script executed)

eval() :- always takes i/p as string & returns a string.
To use as integer, you need to use parseInt().

* User Defined functions :-

```
<script>
function abc() {
    Alert("Hello");
}
abc();      // func. call.
</script></head>
```

execution not started as there is no calling.

Keyword

↳ func. name

CLASSTIME | Page No.
Date / /

21 Aug, 18

NOTE:- Definition of func. will be given in head & func. call will be specified in <script>.

abc(); → If it doesn't work, then it jumps to next Stmt.

* Event Driven mapping ↓

On execution of an event, then we call some functionality on it.
{DOM}

event - some change.

Built-

eg:- <head>
func

<body

If we
nee

Paramet

Here we ne
use var to
assign
value

Functions

Built-in User defined.

ex:- <head>
function abc() { alert(); }
</head>
<body>
abc(); // non parameterised func.

if we execute functions in event, no need to use ; at last and also func. is not in ""

abc(); ⟶ normal stmt. requires ;

Parameterized func.

function abc(int)
{
var a = int;
alert(a);
}
abc(10);

Here we need to use var to assign value.

CLASSTIME | Page No.
Date / /

Function with a function

```
function abc()
{
    var a = "abc";
    var b = "def";
    xyz(a,b);
}

function xyz(p,q)
{
    return(p+q);
}
```

→ Here if we define variables in diff. / 2 seperate lines then ; not req in both lines.

But if we write in same line, we need to use ;

```
var a = "abc"; var b = "def"
```

; required.

Only for variable declaration.

```
var a = "abc", b = "def";
```

; req ✓

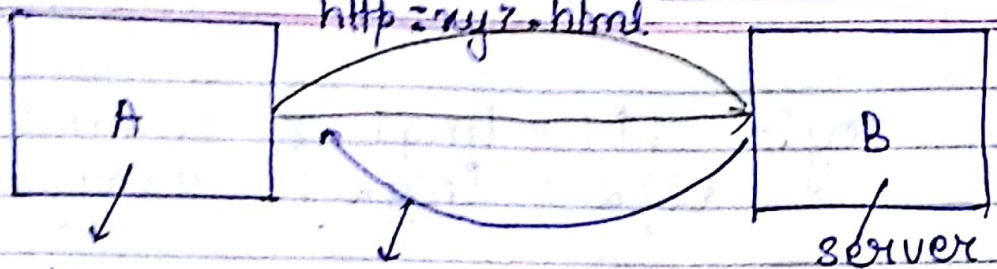
* DOM → Document Object Model.

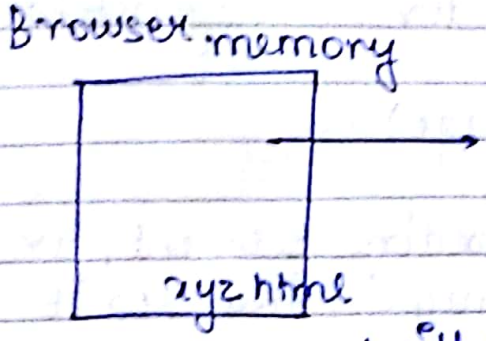
TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well





Now server searches for xyz.html & prints to window i.e Render without any changes.

Now browser memory has no context in it due to which browser will not do any changes in this {static}.

Text
Images
TextBox
Buttons.

things in webpage which are to be biforcated to allow changes.

elements of a document (webpage)

In DOM, we wants elements converted to Objects so that all elements get a memory and then can be referenced for further changes.

CLASSTIME | Page No.
Date / /

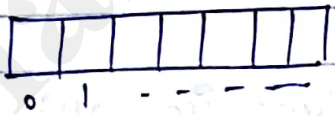
eg:- Obj1 = img1
Obj2 = img2
} elements accessed through obj.

* Object defined at Top level
↓
Navigator (highest hierarchy)
(Object of a page)

For every corresponding element, we have an array defined for it

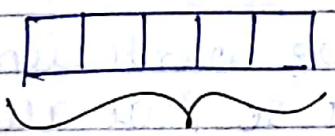
eg:- If we have 10 images & 3 buttons

ar_image[10] ar_button[3]



{image will be accessed by index}.

eg:- If we have no video clip in document



null values are present.

CLASSTIME | Page No.
Date / /

NOTE:- Array is 1-D and not 2-D and array is automatically defined.

- Hierarchy Sequence in DOM :-

(1) Navigator (Browser) (doesn't include CSS)
(2) Window (window in browser)
(3) Document (page in window).
    (i) Anchor
    (ii) ~~Anchor~~ Link
    (iii) Form
    (iv) Button
    (v) Text
    (vi) TextArea
    (vii) Radio Button

Objects are initialized on basis of priority/ hierarchy.

(iii) to (vii) Inside form hierarchy.

Now for every element, an array gets initialised.

Now if some changes take place, array gets changed & can be retrieved.

* Dom doesn't include CSS but we need to have dynamic webpages
  ∴ We introduced
  ↓
  JSSS (Java Script extended Stylesheet).
  (DOM model)
  (Java assistant extended stylesheet)

Hierarchy becomes ↓

(1) Document
- Tags
- P
- DIV
- SPAN

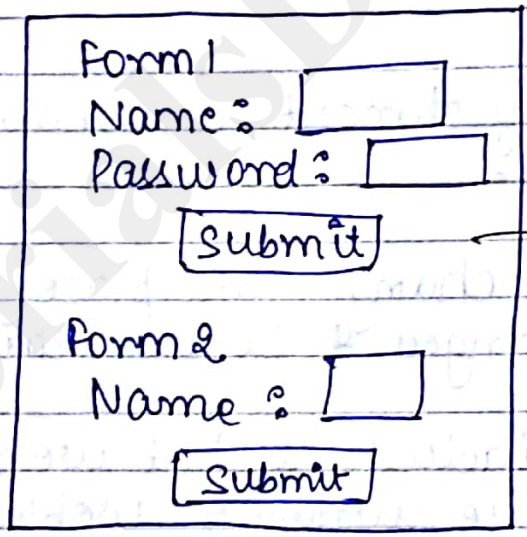
(2) Classes

(3) ID's

→ Document.getElementById(" ");

fetches an object on basis of ID.

eg:-



→ Acc. to DOM hierarchy first of all form object will be created

Form | 0 | 1 |

0 and 1 are addresses of form 1 & 2.

CLASSTIME | Page No.
Date / /

Form 1 [0][1][2] → Submit.

Form 2 [0][1]

Now to access Name of Form1

↓

Form1[0] and not Form[0]

This specifies 2 addresses for Form1 & form2.

OR

Form[0].Form1[0].

Hierarchy to access Name

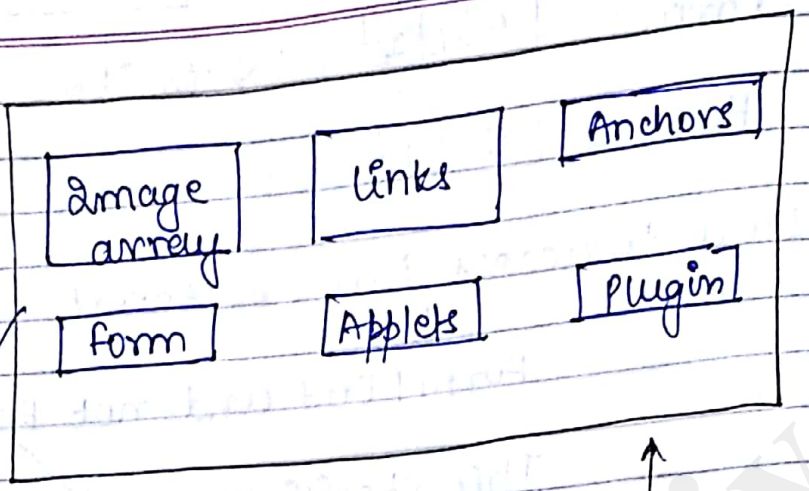
→ We can access div by its name throughout the program as it is an object.

Form.
- Buttons
- Radio Buttons
- :
- :
- :

elements of form

We can give these elements inside form only as they are

∴ First main array is made of form and then these elements are accessed using index.



Objects of Browser Memory

These are diff. arrays of objects & thus Button can't be made as a single array. It needs to be accessed using form array.

Objects for Navigator

- Window
- location
- history
- event
- screen.

} For showing search history we need to call object navigator & not document object.

We have 3 things defined for an object (element)

Anchors

Plugin

emory

of objects & thus
ade as a single
eds to be accessed
ay.

) For showing
search history
we need to
call object
navigator
& not document
object.

d
(element)

(1) Property (2) Methods (3) Events.

↓

things that are
specific to an
element.

eg :- <input type = " text" value = " " id = " "
name = " " default value = " "
length = " " />

These properties are by default defined
& properties differ acc. to elements

↓

→ To access ~~value~~ element of Document

document.getElementById("1");

This fetches address
of a textBox whose
id = 1.

To access its value

↓

vara = document.getElementById("1").value
alert(a);

→ Events are used for changing State of a
method

eg :- OnMouseClick()
method.

CLASSTIME | Page No.
Date / /

Name : Name

* Here we want to change content of this, we can do this by.

Onclick (when we click textbox) — Onchange (if we press backspace)

To implement onclick.

```
function abc()
{
    document.getElementById("1").value = "";
}
```

to change value.

* <input type = "text" value = " "
onclick = "abc()" />.

To focus on a textbox

We use .focus() & not onFocus()

as it is an event.

document.ge

Thi
element as
proper

→ Properties &

For one pa
more th
ev

* Form elem

- Text
- TextArea
- Radio (
- Checkbox
- Button (pl
- Submit (by
onsubr

Fo

for both

document.getElementById.focus()

This serves as a property of element as for an element we have properties, method & events.

→ Properties & methods are used at time of manipulation.

For one particular i/p type, we can have more than 1 events provided that event is registered with that Object.

* Form elements to be worked upon

- Text
- TextArea
- Radio (for RadioButton)
- Checkbox
- Button (plain tab)
- Submit (by default calls onSubmit in form)
- Reset (default resets)
- Select } works
- Option } together as drop-down.
- Password
- Hidden
- File Upload.

For making a radio button

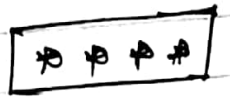
male : <input type = "male">
female : <input type = "female">.

Here, we need to have the same value for both radio buttons as otherwise without value, both can be selected.

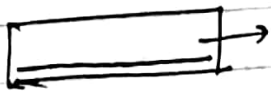
→ In checkboxes, we give value as null.

NOTE:- <input type = "password">

↓



NOTE:- <input type = "hidden">



ue value as null.

password">

idden">

→ no text visible.

27 Aug, 18

→ Anything displayed on webpage is an element for each element, we have a 1-D array through which values can be fetched.

To define anything like button, radiobutton or checkbox, we use <form> ... </form>.

We do this so as to create a super array which stores references of diff. form elements.

<form action = " " method = " "
Where to send page. GET or POST defined in http.

* Event Handlers available in JS

(1) onAbort event → With event handlers, we always use "on"

(2) onBlur

(3) onChange. event handler.

(4) onClick

(5) onDoubleclick → onDblClick().

(6) onDragDrop.

(7) onFocus

(8) onkeyDown.

(9) onKeyPress / onKeyRelease.

(10) onKeyUp

(11) onMouseDown / onMouseClick

(12) onMouseOut

(13) onMouseOver

(14) onReset

(15) onSubmit

(16) onload. {when page gets loaded}

This specifies that whenever something occurs, "on" registers the corresponding listener to capture its action.

* onAbort → When loading pages, images/ videos don't get loaded initially & in that cases we get/use onAbort.

```
<img src = "          "
    onAbort = "abc();" />
```

On doing this, abc() gets executed which is a user defined func.

Now, if usr or password is wrongly entered we use onfocus so as to highlight the textbox color.

document.getElementById(" ").
value == usrv
↓
if matched, then display alert.

→ If this doesn't happen, then we use focus method defined by onfocus.
↓
document.getElementById(" ").
focus
focusses the textbox given by particular id.

NOTE:- Focus() doesn't work for buttons. And focus() works for only 1 element at a time.

NOTE:- Removing "on" from all of them makes them a method.

→ onClick captures an event and click() method provides implementation.

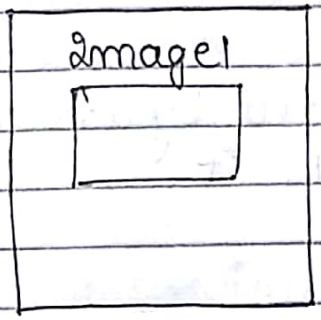
NOTE:- Mouse handling events are needed to be given implementation to methods.

→ onReset() and onSubmit() have a by default implementation & they redirect us to formaction.

→ If for diff. submits (more than 1) then we need to check for submit using Id.

→ While using submit button, we don't need to explicitly use onSubmit().

Q




```
<img src = "    " >
onMouseOver = "abc()";
onMouseOut = "def()";
```

```
abc()
{ <img src = "    "> }
```

```
def()
{ ———— }
```

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

CLASSTIME | Page No.
Date / /

* TextBox

Name: <input type = "Text" Id = "1" />

Name: [ ]

"TextArea"

The textarea can be extended by dragging. Even if we don't drag it, it

* For text & textbook, available properties are :-

(1) name
(2) value
(3) default value (original display of value).

} By default available.

* Methods

(1) onFocus()
(2) onBlur()
(3) onChange()
(4) onSelect() ⟶ to select data.

CLASSTIME | Page No.
Date / /

onChange()

already available data has to be deleted

already available data has to be appended.

Q Create a Textbox with a default value. As soon as Textbox is clicked flush out the default value.

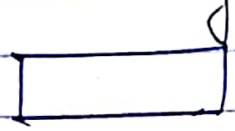
. value = " ".

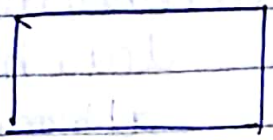
Name : < input type - - - - - - />

```
function abc()
{
    ("2"). value = document. getelement
                    By Id ("1"). value;

    document. getElement By Id ("1").
                    value = "   ";
}

onclick = "abc";
```

CLASSTIME | Page No.
Date / /

Q Stack using JS.





Stack

Push Pop

default implementation in JS

a.push(" ");
a.pop();

underflow.

28 Aug 18

* Properties :- attributes used while defining i/p type

eg:- name, value.....

→ To access those properties, we have event listeners / methods.

Event

Trigger → captures change of state

But its working depends on method.

* Radio Buttons :-

(1) Properties - Checked (by default selected) (id.checked)

- Index
- length
- name

When we group radio buttons, the name must be same as we need to

select only one of them.

(2) Methods - clicked()
event listener → onClicked().

On click of a radio button, property gets checked from unchecked & we can have diff. functionality.

eg: Enter Num [ ]

{ ○ Square ○ Squareroot }

use of onClicked().

Result [ ]

Square : <input type = "radio" name = "xyz" onClicked = "square()" />

Squareroot : <input type = "radio" name = "xyz" onClicked = "root()" />.

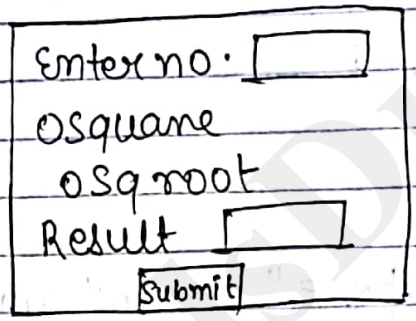
We keep name property same so as to group them together.

```
function square()
{
  var a = document.getElementById("1").value;
  var b = a*a;
  document.getElementById("3").value = b;
}
```

```
function root()
{
    var a = document.getElementById("1").value;

    var b = math.sqrt(a);
```

This is an inbuilt square root func.

```
document.getElementById("3").value = b;
}
```

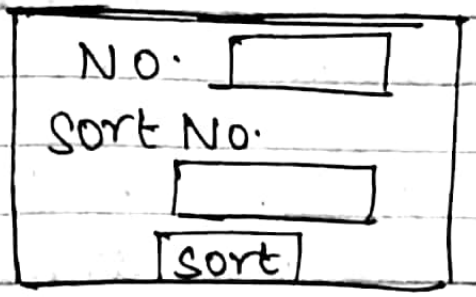


On submit, calculate square or square root.

```
<input type="radio" id="2" - -/>
<input type="radio" id="3" - -/>

Result: <- - - - id="4"/>

<input type="button" value="submit"/>
        onclick="check()"

function check()
{
    if (id.checked == "2")
        abc();
```

```
else if (id.checked == "3")
    def();
}
```

Q



CLASSTIME | Page No.
Date / /

JavaScript

* **CheckBox** :-

Maths : <input type = "checkbox" name = "abc"/>
Eng : <input type = "checkbox" name = "def"/>

grouping :∴ Here name is diff [i.e diff multiple options can be selected.

Maths ☐
Eng ☐

more than 1 option is available.

→ If we need to select only 1 checkbox, group them together keeping the name same.

Properties-

checked.
{ Eg: checked }

* **Selected Option.** { like dropdown menu }

<input type = "select" >
<option selected> → When we have 1st option, it is compulsory to write selected.
<options>
<select>

Format:

format 2 → <
<option sele
<optio
<
abc

If not used selected, we won't be able to select any option.

To do thi

<select nam

{ like radio buttons, we use checked

for dropdown we need to check Selected option }.

T
s

* { . innerH

<p

CLASSTIME | Page No.
Date / /

heckbox" name="abc"/>
ckbox" name="def"/>

is diff (i.e diff
options can be
lected.

on is available.
only 1 checkbox,
ether keeping the
ne.

{ id:checked}

ropdown menu}

"/>
→ When we have
1st option, it is
compulsory
to write
selected.
t>

CLASSTIME | Page No.
Date / /

format 2 → < select name = "abc" >
<option selected > xyz
< options > pqr
< select > def.

If not used
selected, we
won't be
able to
select any
option.

abc [v]
xyz
pqr
def.

} can select
more than 1
option in
dropdown.

To do this, we have a property.

< select name = "abc" size = "0" >
by default.

{ Like radio buttons,
we use checked
for
dropdown we
need to check
selected
option }.

size = "2"

This means that 2 items can be
selected and will have
position as 0 and 1

xyz	pqr
(0)	(1)

* { . innerHTML } → fetches value from an HTML tag or element.

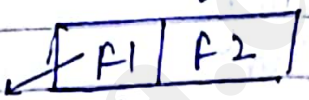
<p> ≡ { provide p a name &
then use inner
HTML }.

* To access values / elements

document.getElementById("1").value.

Using arrays

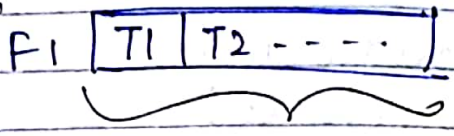
eg:- We have 2 forms



name of form

{<form name = "F1" />

In form 1 we have



F1.elements[0].value — elements array

fetches value of textbox T1 in form 1

- Built-in Objects Defined in JavaScript

(1) String - has only 1 property i.e length.

- Default methods

(a) blink()
(b) bold()
(c) charat(Index)
(d) Italics()

(e) tolower() → converts to lowercase.
(f) toUppercase() → converts to uppercase.
(g) substring (Start index, last index)

var a = "xyz" // same as a = new String();

only 3 DT in JS (String, number, boolean)

xyz.length (✗) → length is a property not a method.

a.blink()

blink is a tag in HTML used to continously flash out / highlight it.

No change in data but continously flashes it.

a.bold() → makes font bold.

a.charAt(0) → fetches character at index 0

{small a}

a.italics() → makes font italics.

a.substring(0, 2) → what part as a substring is to be considered

eg :- var a = "xyz"

a. substring (0,1) → 2nd value is exclusive
1 value is inclusive ↓

O/P → 'x'

a. substring (0,2) → O/P → xy.

(2) math-

Default methods -

(a) abs() → returns +ve value
(b) ceil() → returns upper value
eg:- ceil (2.5) ⟹ 3
(c) floor()
(d) pow (base, power)
(e) cos()
(f) tan()
(g) sin()
(h) sqrt.

} returns trigonometric values from quadrant.

All these methods are to be accessed using objects.

math.abs() ✓
or
math m = m.abs();

V. V. imp.

Date - This is the only object which needs to be instantiated first.

```
a = new Date();
dow(a);
```

↓ executes system's current date and time. (static time, changes only when we refresh webpage)

In Java, date is defined in util package.

eg:-
```
java.util.*;
a = newDate();
S.O.P(a);
```
} prints current date & time.

Default methods -

(a) getDate() — returns a value from 1 to 31

```
var abc = a.getDate()
dow(abc);
```

29 ← fetches numeric value from current Date.

(b) getHours() - { follows 24 hr format)
↓ returns a value from 0 to 23.

```
var b = a.getHours();
dow(b+1);
```

(+1) is req. as hrs is from 0 to 23.

So, +1 is req° to give exact no. of hrs.

(c) getTime() → returns no. of milliseconds from 1 Jan 1970 to the current time specified as parameter.

eg :- a.getTime(2017)

works from 1 Jan 1970 to 2017

If no parameter is provided, it takes the system's time.

Implicit work.

NOTE:- getTime() can return -ve values also.

To print date in YYYY/MM/DD format

We can use default methods of Date object

To fetch date, we can use
getDate() → DD
From getTime() → we fetch millisec. and convert it into years &

CLASSTIME Page No. Date / /

ive exact no. of
hrs.

rns no. of milliseconds
to the current time
ameter.

ime (2017)

orks from 1 Jan
1970 to 2017

vided, it takes
me.

rk.

n -ve values

format

ods of Date

use
→ DD
ch millisec.
ans &

CLASSTIME Page No. Date / /

remainder will provide us months.

- setTimeOut()
  - has same parameters as setInterval() but setTimeOut() executes things only once.

  eg:- used to set values to colours after a given time interval.

- setInterval().
  - we specify a time interval in milliseconds & after each ms, time gets updated

  a = new Date();

  To call it recursively

  function abc()
  {
  a = new Date();
  }

  setInterval(abc(),1000)
  ↓
  after each second, time gets updated.

  - used to make a moving/working clock.

* for alphanumeric values ↘ We have match (a, R.Exp).
  If matched, then exp. is ✓

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

4 Sept 18

Regular Expressions

for matching alphabets.

var a = (/^[a-z A-Z]+$/)
to match with Regular expression we use match()

<input type = "text" id = "1" />
var b = document.getElementById("1")
value.match(a);

returns True / False

for comparing nos ↓

use 0-9 instead of a-z.

Q Now enter a value in textbox and simultaneously check whether the entered chrs valid as a regular expression or not ↓

OnKeyPress / OnKeyUp / OnKeyDown has to be used.

Regular Exp. ↓

/ [↓] + $/
- [↓] : value
- + : to concatenate other values
- $: end of expression

Advantage

no. of requ
increased
& resource
reduced &
time comple
xity ↓ ex.

Server A

CLASSTIME | Page No.
Date / /

JSP

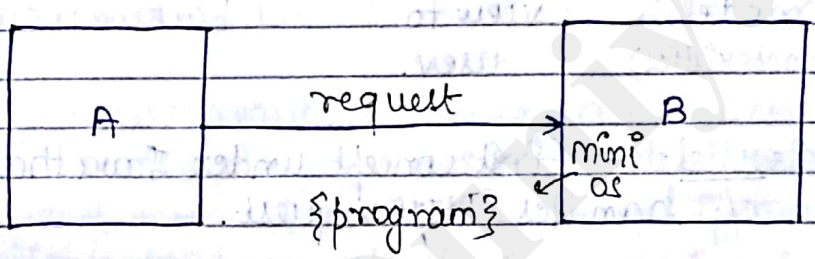
(Java Server Page)

↓

executed on server ∴ pages formed will be webpages.

Architecture of JSP is purely Java based i.e use of classes & objects.
Then this code gets embedded to HTML.

JVM – OS dependent (platform dependent)



Server A sends a request to server B ∴ B must have some resources to access the request.

Total memory is 4K & each program requires 1K ∴ total programs that will be executed is 4

Advantages

∴ JSP divided programs into threads and thus things work in terms of threads rather than mini OS/program

no. of requests increased & resources reduced & time complexity ↓ es.

ServerA → ServerB is called CGI (common architecture

{ MVC - Model View & Controller }

CLASSTIME | Page No.
Date / /

Gateway Interface)

* CGI
  - dynamic webpages
  - interactive webpages.

* Before JSP, we had Servelets.

• MVC → {divides program in diff. parts}
  - basic architecture/ model (characteristics)
  - view to user.
  - controls processing (combines M & V) (backend work).

→ Servelets - first concept under Java that handles HTTP request.
  - w/o this can't handle any browser webpage.

* basic technology upon which we make our webpages using Java programming.

Java Servelets are programs that run on a Web/ Application Server that acts as a middle layer between a request coming from the Browser to the databases/ applications available on HTTP Server.

CLASSTIME | Page No.
Date / /

* Server
(provides services).

→ Depending upon services, we classify servers.

- If a server provides DB, then it is dB server
- If we deploy, host something then it is a Web server.
- If a server provides particular type of services regarding an appn, it is called Application server
(Broader Term).

NOTE:- When we request a server, a port gets created at server's side which is sent to client. Similarly, if many clients send requests, then buffer of server becomes full and thus it is not able to accept more requests.

Thus, sometimes we are not able to receive info. on webpage

Imp. Server never crashes → its buffer becomes full but it never crashes.

→ Webpage provides services b/w web Browser & a server.

requires PCB as it executes things seperately.

CLASSTIME | Page No.
Date / /

* Advantages of Servelets over CGI → gateway that is used to process webpage {mini os}.

(1) Performance is significantly better

(2) Servelets execute within the space of a Web server (no need to create seperate process to handle each flying request).

(3) Set of restrictions can be encorporated on a Server machine

(4) Full functionality of Java classes is available to a Servelete.

* Packages req. for Servelets

→ javax.servlet
→ Java.servlet.http.

} Normal Java & servelet program have no diff. Difference is just of packages used.

* Structure :-

→ Req^m for servelets

(1) Server

In netBeans, we have 2 default servers

Tomcat / Glassfish.

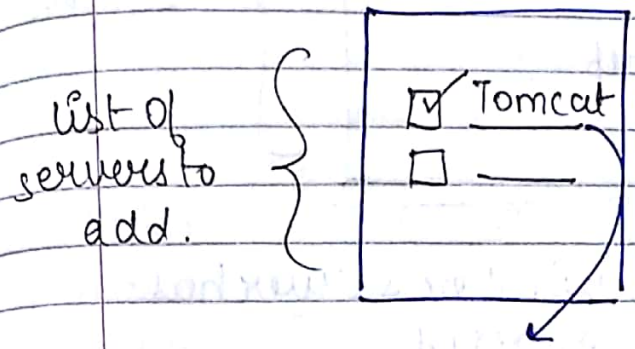
Repositories

CLASSTIME | Page No.
Date / /

project | services | DB } after creation of project in NetBeans

Server
- Tomcat
- Glassfish } must be available.

If not available, Right click → Add server.



If package is already installed, then it will run.

* If not, then go to apache.org (download from internet) .exe ⇒ (Download).
  - .gz
  - .dmg

{You can also download from local directory provided .exe must be available}

When we install Apache
- bin
  - startup.bat {batch file}
  - Shutdown.bat
- root
- web

→ Run startup.bat. 2. want to close it execute shutdown.bat

* using cmd ↓

C:/> (Specify path of bin file)

C:/bin> Startup

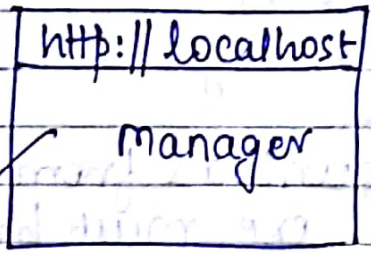
} to deploy server.

↓

How to check whether server has started.

http://localhost ↓

If any of the server is working It displays that default page regarding particular server.



Sometimes provide authentication regarding user & password.

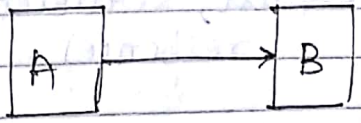
(2) Java Compiler (for servelets). (JDK)

CLASSTIME | Page No.
Date / /

→ How does a servet Work
(Lifecycle of Servelet)

① init() → Initialisation. (allocates resources for particular request)
② service() → (3 methods)
③ destroy() → free port (destroy conn).

Includes initialisation of variables & methods.



handling requests. (includes HTTP services)

• init() - init() is called only once during the lifecycle of a servelete. It normally provides / created when a user first invokes a URL corresponding to the servelet.

* Syntax - public void init() throws Servelet Exception {

(URL) www.abc.com }

{ xyz.html } servelete

as soon as this servelet hits URL init() is called.

• service() - It is the main method of servelet that provides a container to

handle requests coming from the client browser & to write the formatted response back to the user. Each time it receives a request, the server creates a new thread and executes multiple HTTP request type. (get, POST, PUT, DELETE and all other HTTP methods).

* Syntax - public void service(ServletRequest request, ServletResponse response).

service → basically implements HTTP methods.
by default → get ()

to implement HTTP methods. has & methods

public void doget { } method name.
keyword

doPut { }
doPost { }

public void doGet(HttpServletRequest hp, HttpServletResponse hr).

capture request

Specification of type of service.

capture response & send it in a formatted way.

CLASSTIME | Page No.
Date / /

- destroy() -

```
public void destroy() {

}
```

* Container - (Web container)

↓

box that contains files of same extension seperately
(biforcates all files stored on server)

eg :- A —abc.jsp→ B

To check for this file, a container gets called & thus each request will be carried out by a web container which will further go to a specific container.

Container processes file, generates response and returns back.

→ jsp pages → JSP container
→ servlet pages → Servlet container.

Program:-

```
import java.io.*; // We use this package as we need to process o/p in a formatted manner.
import javax.servlet.*;
import javax.servlet.http.*;

public class abc extends HttpServlet.
```

→ main class will be where we will implement http methods.

```
{
  String message = "Helloworld";
        public void init(){ }
public void doGet(HttpServletRequest request, HttpServlet.Response response)
{
      // now we need to specify which type of response is req.
Execution must return to client side
∴ response is generated by our class.

response.setContentType("text/html");
   // things displayed on server can be text or some html tags.

PrintWriter.out = response.getWriter();
```

takes / displays content in one go.
{ Buffer Writer class take content chr. by chr.

```
out.println(message);
      public void destroy(){ }
```

NOTE:

- Here we don't have a main class. But a main class for a Servelet is service
  ↙
  { doGet --- }

- It is not necessary to implement init() & destroy() methods.

Response
Plain Text
HTML.

S.O.P ⟶ prints on command prompt.

Here we use Response Writer as we need to write on browser window.

PrintWriter.out = response.getWriter().
↙
after this stmt, out becomes an obj. of Response Writer.

To compile ↓

javac abc.java

After this, the code won't work on cmd as it won't identify all these services. To make it work, deploy your code to WebResource folder in Apache → Tom-Cat

and use localhost/abc.

NOTE:- The no. of services we implement the no. of containers keeps on ↑ing.
↓ not a hardware but a logical connection.

container does the work of interpretation & executing the code.
↑

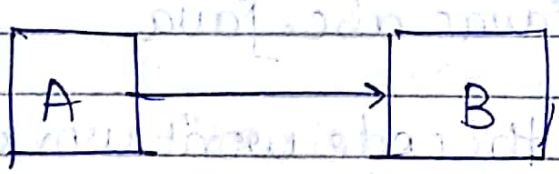
gives implementation of service().

NOTE:- in out.println
↓
We can use any HTML tags.

out.println("<h1>" + "hello ..." + "</h1>");

Using this model, we define everything at a single place and thus implementation is req.
In MVC, we have seperate components & thus implementation is not necessary.
↓

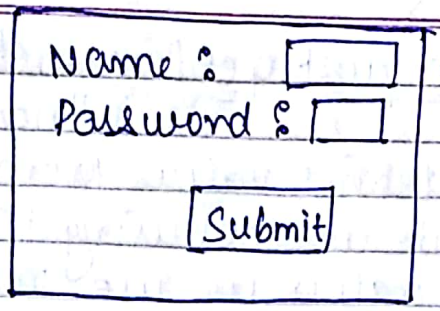
method-2.



abc.com/xyz.html.

Client always generates request
Server " " response.

CLASSTIME | Page No.
Date / /



Here <form action = method = /> is necessary to use unlike JS.

as we need to know which file will handle request and what type of HTTP method has to be used.

HTML document ↓

<form action = "ABC" method = GET/POST />.

<input type = "text"

⟹ public class ABC extends HttpServlet {

public void doGet ( )

{ response.setContentType ("text/html");
PrintWriter out = response.getWriter();

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

only request obj. is used

CLASSTIME | Page No.
Date / /

out.println(request.getParameter ("name"));

O/P.

This method fetches values from our form. And here we are not using 'id' attribute but rather we use 'name' as for multiple pages we don't have 'id'.

In one servlet mode only, we can print multiple stmts using concatenation

{although we can use multiple print stmts}

out.println ("<html>"Name:" + ...);

NOTE:- request.getParameter ("name", "pass");

X

One getParameter can capture only 1 value.

<form action = /ABC>

no extension req.

(ABC is a URL upon which we send our data).

NOTE:- doPost() is not defined in servlets as functionality is only defined for doGet and we also dont know what method to use.

doGet() is a default method and thus we

necessarily need to provide implementation.

```
public void doPost ( )
{
    doGet ( request, response);
}
```

NOTE:- Here we are doing everything on clientside both forms & servelets and client requests for that particular file.

→ JSP works on MVC.

- getParameter() → Servlets handle form data using the foll. methods :-

(1) getParameter() -

(2) getParameterValues() - If the parameter appears more than once, and returns multiple values getParameterValues() is used to fetch the data. ↓ (for checkboxes)

(3) getParameterNames() - This method is used if you want a complete list of all the parameters in the current request.) ↙ (how much data came in request with what name).

(input field value)

NOTE:- All 3 methods work with object of request.

Practical ↓

for Checkboxes { request.getParameter("math"); request.getParameter("Phy"); }

↓

If we want to fetch values at one go.

→ abc.com/?name= + password= + ____ + - - - - -

max. limit = 1024
(no limit on POST)

→ more secure.

request.getParameterValues("a1");

↓

captures values stored in a1.

10 Sept '18

CLASSTIME | Page No.
Date / /

JavaScript

→ User defined Objects :-

To create Objects in JS, we have functions to create user defined objects.

```
function Student() {
```
obj. for whose attributes need to be defined.

eg - name
age
marks.

Now we need to get values ∴ we make parameterised func.

```
function Student( name, age, marks)
{
this. name = name;
this. age = age;
this. marks = marks;
}
```
this. → if name of parameter in func. is same as attr. we use this.

To create object

```
Student1 = new Student ("abc", 10, 50);
```
With obj, we must use 'var' keyword

- Composite objects

```
function student()
    {

    }

function marks(sub1, sub2)
    {

    }
```

Now student1 = new student("abc", 10, new marks(50,60);

In JS, we have an operator 'new' to create objects.

Here we are able to get multiple values.

CLASSTIME | Page No.
Date / /

Servelets

Name: [ ] → request.getParameter ("name");

For a field having multiple values ↓ returns only 1 value.

Address:
Line1: [ ]
Line2: [ ]
City: [ ] Postal code: [ ]

To access these, we fetch one value at a time, we need to keep concatenating them until address is complete.

↓

For data relating to same block, we use

Line1 < input type = "text" name = "address"/>
Line2 < input type = " " name = "address"/>
city < name = "address"/>
Postal code < name = "address"/>

making a group.

↓

use getParameterValues()

works only when things are made into a group with same name.

CLASSTIME | Page No.
Date / /

String[] val = request.getParameterValues("address")

returns multiple values
∴ store result in array.

at every index, we get value of one textbox.

To display values ↴

```
for(var i=0; i<val.length; i++)
    {
        val[i].value;
    }
```

NOTE: For using getParameter() and getParameterValues() we need to know name attribute for these defined textboxes.

- getParameterNames() ↴

used to fetch attribute names for textboxes.

request.getParameterNames( );

This method doesn't have a parameter as we need to fetch all values from request

We have one obj
∴ To st
use an enu

methods in enume

Enumeration e

(1) has more El
anything
returned

(2) next() —

while (
{

Object is superclass of every class.

Obje

To f

St

(Store

We have one object in 1 request
∴ To store returned values, we use an enumeration
(collection of obj.)

Methods in enumeration.

Enumeration e = request.getParameterNames().

(1) hasMoreElements() – checks if there is anything in e. Boolean value is returned – True / False.
→ no element i.e no execution takes place

(2) next() → to iterate through obj.

while (e.hasMoreElements())
{ boolean value.

Object is superclass of every class.
Object o = e.nextElement()
returns an object value.

To fetch its value, we use.

String a = (String) o;
(Stores name of Textbox 1) → Explicit Typecast.

```
while(e.hasMoreElements())
{
    Object o = e.NextElement();
    String a = (String) o;
    cout<<a;
}
```

First time loop runs, we get name value of textbox 1.
Second time, we go to while loop and check if e is empty or not.
And then e.NextElement() auto increments the index & next name value is displayed. (+e)

```
out.println(a + " " + request.getParameter(a));
```

{(1) produces O/P (2) does processing also}

(fetches value in "abc")

O/P the value of name attr. of TEXTBOX 1.

```
<input type = " " name = "abc"
```

NOTE: Id parameter doesn't work on multiple pages. Thus we use name attr.

Through out.println method, we have producing O/P as well as processing simultaneously.

Thus, this is the disadvantage of servlet

CLASSTIME | Page No.
Date / /

∴ We have concept of JSP (MVC model)

NOTE:- setContentType ("text/html");
↓

MVC
- Model: basic operations that are to be identified & preparing a structure upon which program works → structure
- View: presentation (how must the program must look)
- Controller: processing part that combines model & view

in servlets, these 2 things were done together.

NOTE: JSP defined MVC concept of doing everything individually.

(Java Server Page)

<html> + Java code (classes, loops, import package, etc.) </html>

extension → .jsp

provided class definition (i.e ctor) is not given.

CLASSTIME | Page No.
Date / /

After saving in .jsp, it works on server side using Java code.

Java code is to be written inside JSP tags

<html> (tag) int i=10 (tag) </html>

JSP tags.
(After this, i gets value = 10)

- HTTP request & response headers

Special info. to identify a specific node.
Headers never contain value/data in them.

→ Request header parameters -

(1) Get /xyz.html HTTP 1.1 → Version of HTTP
or POST host: www.abc.com // where page has been
User agent: mozilla (windows) forwarded
Accept: image/gif, image/jpg

browser used (used to send request)

mozilla works on which window version.

HTTP can accept only images defined by extension.

Accept language : en-US
Accept Charset : off
+
value → abc

eg :- www.abc.com/xyz.html?name=abc.
(request)

→ Response header parameters-

HTTP/1.1 200 OK
Symbol.
server status codes.
(eg :- Resource not found (Error 404))

last modified :
{changes incorporate}.
Date : (current Date)
Status : 200
Content_Type : text/html
Servlet engine : Tomcat → (name of server)
Content length : byte
{explicitly defined in servelets}

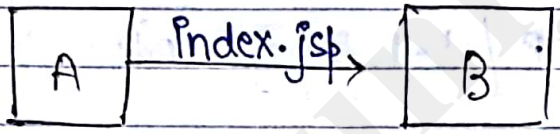
last modified → most imp. (decides whether to compile or not)

JSP Syntax

<html> (JSP) <html>

Java code
↳ http Get/Post

Container → Interface b/w application & server which helps to execute application on browser, depending on servers we have diff. containers.



container.

- Tags in JSP :-

Scriplet -

(1) ~~Scriplet~~ - <%. %> → base tag

(2) Expression - <%. = %> to define java code

(3) Declaration - <%. ! %>

Scriplet.

- ~~Scriplet~~ - can contain any no. of java statements, variables or methods declarations.

* Index.jsp → {we can write any Java code}

```
<html>
<%. out.println("Hello World"); %>
</html>
```

scriplet tag in JSP.

CLASSTIME | Page No.
Date / /

Apache - Webapps → root Folder → index.jsp.
On Browser → localhost / index.jsp.

- Expression Tag - <%= %>
It contains an expression that is evaluated & converted to a string for sending on displaying the response.

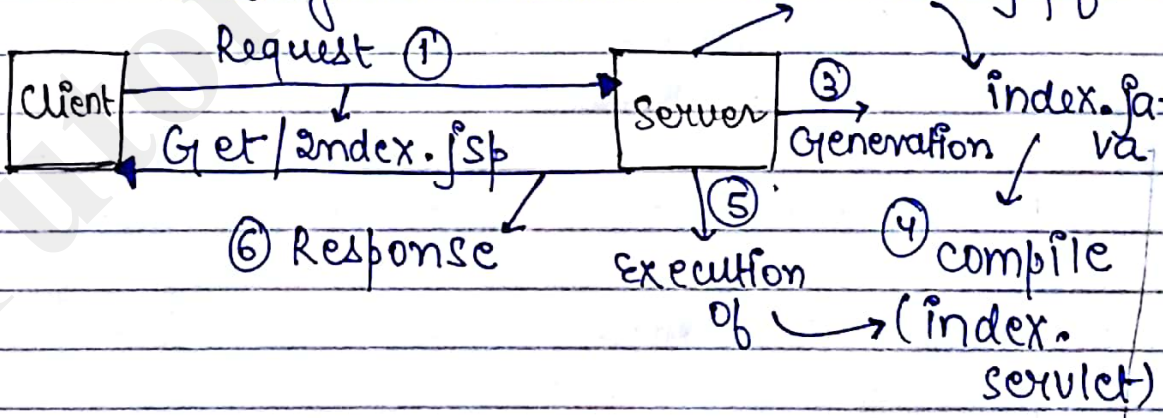
<%= (new java.util.Date()) %>

To print an expression directly after evaluation. — evaluates this expression & then displays in string format.

- Declaration Tag :- Initialises & define variables
<%! int i = 10; %>

* JSP Compilation { How a JSP page is compiled & executed on server }

- JSP processing (8 marks)
Diagram in book.



NOTE: Step (2) and (3) are together known as "Translation phase".

not a normal .java file, so change to .java

NOTE: Here we are dealing with 2 containers i.e

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

CLASSTIME | Page No.
Date / /

JSP and Servlet

8 marks

Step ① :- First step is Browser sends an HTTP request to the Web Server.

Step ② :- The WebServer recognizes that the request is for a .jsp page and forwards it to a JSP engine → (container)

Step ③ :- The JSP engine loads the jsp page from the disk & converts it into a servlet content.

Step ④ :- The servlet file will be compiled & executed by Servlet engine. → (Servlet container)

Step ⑤ :- Http response is generated & send back to the browser.

JVM ← dependent with OS, platform dependent.

Java byte code → Platform independent

sends an HTTP request
er.
nires that the
and forwards it to
)
he jsp page from the
rvlet content.
compiled & executed
t container)
& send back to the

Step 4 The servlet file will be compiled & executed by the servelette engine.

Step 5 HTTP response is generated & sent back to the Browser.

* Converted Servulet Page of a JSP page :-
(Conversion of JSP to Servlet)

- JSP page ↓

```
<html>
<head>
<title> First program </title>
</head>
<%. int v = 0; %.>
<body>
count : <%. out. println( v++); %.>
</body>
</html>
```

- Corresponding changes in servlet ↓

```
import... } import stmts
                as it is

class _jsp extends HttpServlet {

    public void _jspservice (

response. setContentType ("text/html");
    out.println ("<html><title>....</html>");
      int v = 0;
      out. println (v++);
      out.println (" ");
```

CLASSTIME | Page No.
Date / /

NOTE: Life cycle of JSP → translation()
jspinit()
jspservice()
jspdestroy().

Life cycle of Servlet & JSP works in the same manner except the keyword jsp before methods.

NOTE: As the jsp page is converted, the normal compiler runs. (as it is not .class file but .java file).

NOTE: A Web container has 2 diff. engines called

(1) Catalina
↓
It is a servlet engine

(2) Jasper
↓
It is a JSP engine.

→ Once a JSP page is identified with its extension, translation phase is done with the help of Jasper and execution will be done by Catalina.
↓
When we define path for server, we have to use class path variable as "Catalina".

• Directive Elements-

→ These are the elem
to process itself.

→ 3 types of Directi

(1) page directives
supports multiple
processing, lang.
encoding for <%@
that page.

(2) include dir
<%@

(3) tag library
<%@

pr
i.e th

• page directi

(1) Buffer - It
tics for the

<%@

(

→ translation()
ce()
oy().

erulet & JSP works in
manner except the
fore methods.

nverted, the normal
. (as it is not .class
a file).

s 2 diff. engines called

(2) Jasper
↓
It is a JSP
engine.

ntified with its
phase is done with
execution will be

h forever, we
variable as

These are the elements/tags that helps a page to process itself.

→ 3 types of Directive elements are :-

(1) page directives @ page → { We want to handle errors at some other page & after that we get redirected to main page }
supports multiple processing lang. ←
encoding for that page.
<%@ page %>

(2) include directives
<%@ include %>

(3) tag library
<%@ taglib %> → Used to define custom tags (i.e user defined tags).

provides page direction.
i.e they are directive elements.

• page directives - has 10 attributes/properties.

(1) Buffer - It defines the buffer characteristics for the O/P response object.

<%@ page buffer = none / KB (not more than that)
(to set buffer size)

If size is none, it means that we have nothing in buffer and if it has some value (in KB), it gets delivered to user i.e it is the size of packet at user's end.

(2) Autoflush - When buffer gets full, we use attribute autoflush that automatically flushes out buffer size.
It returns a boolean value

True False.

autoFlush = "true";

It fetches value of buffer, sends to client & then automatically flushes out buffer.

NOTE:- AutoFlush has a permanent effect i.e value doesn't persists.

The size of buffer defined by user is the max. size of data to send.

NOTE:- By default, autoflush has "false" value. Buffer has no size by default X.

(3) contentTyp (Identifies typ
ca
This is written in JSP also.
eg :-

(4) extends
eg :-

(5) import -

(6) Session -

<%@ page se
i.e HTT
automatic
ents s
(persiste
variable o
multip

(Identifies type of responce)

(3) contentType - Same as response.setContentType
↓
can be ("text/html")/ "microsoft word/doc"

This is written in JSP also.

eg :- <%@ page buffer = 100kB contentType = "text/html"/>

(4) extends -

eg :- @page extends = "packagename.classname"%>

(5) import - @page import="java.util.*"%>

{everything in (=) as it is a value}.

imp.
(6) session - returns a boolean value

True False

<%@ page session = "true"

↓

i.e HTTP protocol automatically implements session

↓

(persistence of a variable over multiple values)

CLASSTIME | Page No.
Date / /

18 Sept, 18.

→ HttpServlet. Session } to access session in HTML

→ In JSP, session is an inbuilt obj.

session.getCreationTime()

Inbuilt object

to access session iff session is true

If session value is false, it throws an error

"Can't access session as value is false"

Practical

→ To install Java Web App.

Method-1 Tools → Plugins → Available plugins

Java EE Base (Install)

Page Directives

(7) errorPage-

(1) handle same

<%

(8) isErrorP page

checks for an e

* Directive

(1) page

(2) includ

<

h
m
p

JSP

Sept 18

Page Directives :-

(7) errorPage - An error occurs & we need to handle it.

① handle at same page

② handle error by redirecting to a diff. page

<%@ page errorPage = "abc.jsp" %>

page that handles errors

(8) isErrorPage - Checks whether the redirected page is capable of handling errors. (checks for an error page)

↓ returns a boolean value True if any error page is present or not.

* Directive elements.

(1) page directives

(2) include directives

<%@ include %>

header
main program

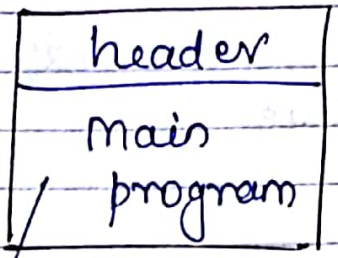
Through this, diff files can be embedded in our program.

CLASSTIME | Page No.
Date / /

NOTE: extends / imports is diff. from merging/embedding.

used with page directives.

Merging 2 files



<html>
Header
</html>
} file 1

Now we use @ include to merge file 2.

xyz.jsp {
<html> (plain text)
Header
<%@ include file = "abc.jsp" %>
</html>

abc.jsp {
<html>
main program
</html>

(When we execute xyz.jsp, contents of abc.jsp gets merged into xyz.jsp appearing as if it is only a single file)

NOTE:-

(3) t

• T

(1) C
(2) f
(3)
(4) fo
(5)

NOTE:-
- <%@include--> is used only with jsp files.
- No attribute is available in include directive.

(3) taglibrary - (tag/lib) {used to access inbuilt tags}.

<%@ taglib.

For all the tags defined in JSP, we have library.

<c:forEach>.

- Tags in JSP (2nbuilt) {defined in JSTL}.

5 super category.

(1) core "c"
(2) functions "fn"
(3) sql "sql"
(4) formatting "fmt"
(5) Standard. "INBL"

prefix.

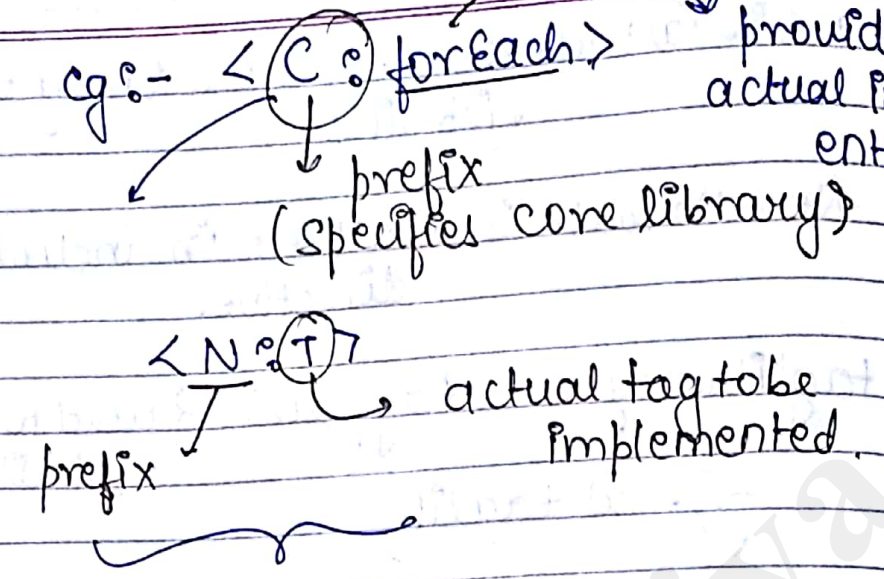
Now to check which tag belongs to which library, we have prefix & url

<%@ taglib prefix="" "" url= %>

specify the super category

location where code is to be executed

CLASSTIME | Page No.
Date / /



custom tags.
(Here we don't have a predifined prefix tag and URL)

We can provide any implementation to <N:T> by defining our own definition.

URL	URN	URI
(Uniform Resource Locator)	(Uniform Resource name)	(Uniform Resource Identifier)
↓	↓	↓
complete path from where we fetch things (URN + URI)	index.html (particular resource to be accessed)	Name of machine from where we get URN.

NOTE:-

CLASSTIME | Page No.
Date / /

NOTE:- Execution of all these tags is done in .war file.
Every core tag has only one URL as there is one .war file.

- Inbuilt / Implicit Objects :-

(1) Request
(2) Response
(3) Out
(4) Session
(5) Application → (Folder created).
(6) Page Context
(7) Exception.

These objects need not be defined in JSP.

(works as a reference)

pageContext("abc.jsp");

The page whose value is to be used on to another page.

Objects that handles exception.
{ we can directly throw obj. of Exception without defining it }

- Actions defined in JSP :- (JSP actions).

XML provides mapping

<mapping>
{ , defines Str. to be executed.

→ JSP actions are always declared in XML format.

<jsp : useBean>

always JSP as we work on JSP.

action to be performed. (can be anything)

(1) <jsp : useBean>
(2) <jsp : getProperty>
(3) <jsp : setProperty>

} JSP actions.

- JavaBean → (Java class) (class)

smallest component of software that a user can change / modify its implementation.

Button } creation of a Button is common (input type = button) but implementation can vary acc. to user.

Generally, we define GUI in JavaBean but we can also define classes

<jsp:useBean class = "abc" >

to use a JavaBean class.

(used to define reusable components)

to use a JavaBean class.

Reusable components means properties can be reused.

Now, we can either get the value of property or set properties

<jsp:getProperty>

<jsp:setProperty>

These can't work independently. For them, we first need to use <jsp:useBean>.

(4) <jsp:include>. — same as @include. Additional files are included in XML format

(5) <jsp:forward> → Where the request has to be redirected { same as form action = "" }

(6) <jsp:plugin>

Requirement by a program gets automatically download & is used.

<jsp:plugin = "Microsoft Office/ Word">

MS-Word is req. for the page.

NOTE:- All of these elements provide some action i.e output thus they are JSP actions.

NOTE:- And JSP directives provide us a direction i.e which to execute first, thus they are directives.

- JSTL

Java Standard Tag Libraries.
(root category to define custom tags)

The user defined tags are registered using JSTL and then only tag will execute.

Specifically JSP standard tag library.

as all basic things are in form of tags.

CLASSTIME | Page No.
Date / /

* onBlur - When a page gets loaded, we see that text of textbox is always not clear then any data with blur data can be called using onBlur which can register onBlur event.

* onChange - used with text, textarea and file upload.

used to change content.

* onClick -
for links, text and images.

* onDoubleClick -
All things of onClick work if we double click.

NOTE: All things will work with help of source & event listeners.

Imp. onFocus - We need to statically authenticate a user & password.

Name [ ]
Password [ ]
[Submit]

```
<html>
<script>
var a = "  ";
var b = "  ";
```

after we check with user entered values using getElementById and if matched we show alert.

Date / /

<jsp:plugin = "Microsoft Office/ Word">

ms-Word is req. for the page.

NOTE: All of these elements provide some action i.e output thus they are JSP actions.

NOTE: And JSP directives provide us a direction i.e which to execute first, thus they are directives.

- JSTL

Java Standard Tag Libraries.
(root category to define custom tags)

The user defined tags are registered using JSTL and then only tag will execute.

Specifically JSP standard tag library.

as all basic things are in form of tags.

Sept 4, 18

JSTL

<% ↓ %>

Scriplet tag to define everything in JSP.

But in JSTL, these scriplet & expression tags have been replaced by some other tags.

<% %>
<%! %>
<%= %>

} JSTL removes this & declares an explicit broad category.

These require longer compilation time. Thus, to prevent this

↓

<html>
<JSTL>
</html>

} Interpolating & making tags dynamic

- 5 categories-

(1) JSTL core tags-
(2) Sql tags-
(3) formatting tags-
(4) XML
(5) JSTL functions or functions.

→ core tags ↓
basic functionality in terms of loops, controls.

→ sql tags ↓
When we use DB.

- Core tags - It defines the basic functionality in terms of expressions, functions and variables & with exceptions & iterations.

* How to identify JSTL tag ↓

<% @ taglib
use a specific library for defined tag.

* To diff. tags for these 5 categories ↓

<% @ taglib prefix = "c" uri = %>
interpreter is ready to handle core tags.
actual implementation

- Attributes ↓

(1) out → <c:out>
to get output.
(implementation is same) { in JSP { <% out.println() %> }

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

If used in JSP, we additionally req. all servlet files but this is not needed in JSTL as these are only JSP tags.

* attributes →

<c:out value
default
escape XML >

If any XML tag is passed in c:out & escape XML value is set true, then XML won't be executed but printed as it is.

NOTE:- all these tags work on expression language.

<c:out value (1+2); ✗
won't work

We use symbols to denote a particular attribute.

Expression is passed in ${ } so that it evaluates.

To identify whether an expression belongs to a special language.

(expression)

${ } → Now if we write ${1+2}; then it would

be evaluated & will be stored in value attribute.

eg:- <c:out value = "${1+2}" />

("") are req. otherwise it throws a syntax error.

<%@taglib prefix = "c" uri = " " %>
<html>
<c:out value = "${1+2}" />
</html>

{same as import stmt} Inbuilt object in JSP

O/P

3

matches whether the defined prefix matches with prefix used. If not matched, it returns a syntax error {symbol not found}.

→ value - The attribute value is used to output any info. and this is a must value/field that we need to define for a <c:out> tag. There is no default value for value attribute.

(doesn't initialize to a junk value)

→ default - This attribute is used to display ~~fallback~~ fallback info. (this is not a req. field)

escapeXml - This attribute value can be set as true if the tag needs to escape the XML characters. {by default value is set to true}.

Exp. is blank i.e no value is evaluated.

<%@ taglib - - - - .
<html>
<c:out value = "${ }" >
default = >

in this case, default gets executed.

If value doesn't get evaluated, then just for a program, default value will get printed i.e either value will be as O/P or default

Both values can't be executed simultaneously.

2. Default, only value gets passed & not an expression.

(no compulsory attr.)

(2) <c:set> —{to set some property}
This JSTL tag is the corresponding version of setProperty which is available as a tag here. This tag evaluates an expression & uses the result to set a value of a corresponding Bean class or variable.

attributes - (1) value - This attribute is used to save the current info.

(2) property - It is used to modify any defined characteristic of a Bean class.

Imp. (3) var - This attribute defines a variable with a particular name.

(4) scope - This defines the scope of a variable i.e stored earlier by the var attribute

(5) target - It defines the name of the variable whose property needs to be modified.

eg:- <c:set value=" "

name

can be any value/expression.

Now the value will persist till how much time is done through scope.

To initialise value of name, define in value=" "

var is used to store a duplicate value of value attr.

Property - value of chrs. specified in target.

Target field denotes a class name.

<c:

To

<

<c:se

2 values in scope.

2 tags

<c:set var = "salary" value = "${1000}">

(trying to update a variable salary)

To print values.

<c:out = "${salary}">

as value of 1000 is in salary var.

<c:set var = " " value = " " scope = "application">

var & value are accessible throughout program. (throughout document)

values in scope. { scope = "application" / OR = "session" → till session is maintained.

2 tags req. to set properties of Java Bean class.

<c:set>

<JSP:setProperty>

(3) <c:if> - defines if cond. { corresponding JSTL tag of if }.

- attributes ↓.

→ test - (req. field) This attribute evaluates a condition.

→ var - This defines name of the variable to store condition result.

→ scope - defines scope cond. of variable (not testing cond.)

<c:if test = "${salary}"

either true / false

since salary = 1000

i.e if "1000" (true)

then, ↓

<c:if test = "${salary}" var = "v"

if we define var, then we need to define scope as to what is lifetime of var.

to store cond. (optional)

4 Sept, 18

JSTL

- Core tags :-

<%@ taglib prefix = " uri = " %>

→ <c:choose> {works on a test condition}

works only with <c:choose>

<c:when> provides test condn </c:when>

+

<c:otherwise> (else value) </c:otherwise>

</c:choose>

eg :- Tags can be closed in diff. lines containing lot of stmts.

<c:if>

≡≡≡

</c:if>

{<c:if> and <c:when> are used in combination}

```
<c:choose>
    <c:set var = "salary" value = "
                                1000"/>
    <c:when test = "${ salary > 5000}"
```

Implementation of otherwise

works same as if.

```
    </c:when>
    <c:otherwise>
        <c:out value = "In otherwise"/>
    </c:otherwise>
</c:choose>
```

For printing Strings, use " " directly & not ${}.

The when & otherwise won't work till choose tag is closed.

- Implementation of when ↓

```
<c:choose>
    :
    :
    <c:when test = "${ salary > 500}"
```

or we can use <c:out value = "${ salary }"/>.

```
//<c:out value = "In when"/>
    :
    :
```

(same code)

used for evaluating expressions.

CLASSTIME | Page No.
Date / /

eg:- <c:set var = "salary" value = "1000"/>

<c:if test = { } value = " ">

In terms of test problem discussed earlier, we have an attribute scope which works in terms of ~~scope~~, session, application, test, page, context.

In a tag, we can use multiple tags

↓

<c:set var = " " value = " ">
<c:when test = "{ salary > 500}"
<c:out value = "{ salary }"/>

↓

evaluates a test cond & returns boolean.

<c:set var = "salary" value = "9000"/>
</c:when>.

not in EL ??

* <c:choose> ↓

It doesn't have any attribute. It will work in combination with <c:when> that has only 1 attribute i.e test which is used to evaluate a condition. (compulsory attr.)

→ <c:otherwise> doesn't have any attribute.

(works as if-else)

* <c:remove> ↓

This tag removes a variable from a specified scope or the first scope where the variable is available.

— Attributes :-

(1) var (2) scope

```
<c:set var="salary" value="1000"
       scope="  "/>
Sal:<c:out value="${sal}"/>
<c:remove                    in terms of session,
    var="salary"/>           application, page,
                             context, test.
Salary:<c:out value="${salary}"/>
```

↓ can't directly use " " as we need to fetch value.

Sal : 1000

Salary : → displays a blank value (value gets deleted)

CLASSTIME | Page No.
Date / /

→ We can only initialize new value of variable after <c:remove> using <c:set>.only.
Reference holds but value becomes '0'.

NOTE:- In case we define a default value for the variable, then if we use <c:remove> value gets deleted & default value gets printed.

* <c:import> →
(tag to be implemented)

→ <c:import> provides string functionality that <jsp:include> provides but with the inclusion of absolute URL's
There are 5 attributes for import tag.

(1) url - It is used to retrieve & import the webpage page.
(2) context - It is used to give the name of a local web application.
(3) var - to define a variable
(4) scope - scope upto which var will persist.
(5) charencoding - It is used to import the

character set from the URL

NOTE:- <c:import> → used to import any URL which can be a JSP page / remote appn.

<c:import url="/WebApplication/webpage/abc.jsp" />

Here we are trying to include our JSP page into our document.
{ If files are in same directory, use abc.jsp }

var="x" />

Stores values of JSP page.

NOTE:- import is used to display along with chr. encoding i.e w/o processing we are fetching raw data

(diff. from include)

- <c:forEach> - This tag gives an alternative to embed a for loop, do while loop with the attributes as :-

(1) Items - This is the information on which the loop will work.
(works only under collection package)

(2) begin- Starting index.

3) end- It is the stop index

4) step- To define increment.

5) var- It holds the description of the current item.

* To print things from 1 to 10 ↓

```
<c:forEach begin="1" end="10"
        var="i" />
          holds value of current item.
<c:out value="${i}" />
</c:forEach>
```

NOTE:- If we don't define step, then by default it increments by 1.

eg:- 1, 3, 5, 7, 9 } set step = "2".

* <c:catch> - It is used to capture any throwable } super class of exception Handling } statement, inside the body and

(JSP By Heinsberg)

CLASSTIME | Page No.
Date / /

display the corresponding error with the help of var attribute.

Here we don't have any explicit try tag The things which cause an exception are to be included in <c: catch>

<c:catch var="e"> {exception e}
(holds a name only)
(We try to handle divide by zero error)
<%. int x = 5/0 %> (Should it be used inside declaration tag.)
</c:catch>
<c:if test="${e != Null}">
<c:out value="${e.message}"/>
automatically print corresponding error msg using var.message

o/p ques in ppr: (Arithmetic Exception: divide by zero)

Working of functions ↓

→ <%. taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions %>.

(1) contains - <fn:contains> {implemented as methods}

(2) <f

(3) <fn

(4) <fn

(5) <

(6) <

(7) <f

(8) <fn

(9) <

(6) <

(7) <

(8) <

Scanned by CamScanner

It is used to test whether a given input string contains a substring or not.

(3) <fn: containsIgnoreCase> - The checking is case insensitive

(3) <fn: endsWith> - It is used to test if a given input string ends with a suffix or not. specified

(4) <fn: startsWith> - It is used to test if a given input string ends with a specified prefix or not.

(5) <fn: indexOf> -

(6) <fn: split> - This functions splits the string into an array of substrings.

(7) <fn: toLowerCase> -

(8) <fn: toUpperCase> -

(9) <fn: substring> - It returns the subset of a string after a specific string.

(6) <fn: substringAfter> -

(7) <fn: substringBefore> -

(8) <fn: replace> - It replaces all the occurences of a string with another given string.

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<c:set var="x" value="xyz" />

<fn:contains(x, "xy")

OR

contains("xyz", "xy");

(whether xyz contains xy)
It returns a value & thus needs to be evaluated. Since we can't store it anywhere thus we use

<c:out value="${fn:contains("xyz", "xy")}" />

to use this include

or we can also use

<c:set var="${ }" />

* Connect database to JSP :-

using SQL tag.

→ The attributes available for SQL tag are :-

(1) setDataSource - <sql:setDataSource>

from where conn^n has to be established, ie from where

(2) sql :

Here, the evaluation goes into rs. <

data has to be picked.

In this, we have 3 things

<sql:setDataSource
var = "db" driver = "mysql"

reference of DB from where we fetch data

url = " "
user = " "
password = " " />

(complete setDataSource tag)

{ Combination of class.forName (driver name) and .getConnection (url, user, pass); }

12) sql:query - used to design query.

Reqm is ref. of DB i.e db.

<sql:query dataSource = "${db}" var = "rs">

select * from students;

authenticating all things of DataSource & storing in db.

Here, the evaluation goes into rs.

</sql:query>

Ch-7
Custom tags.

(3) sql: update - This sql tag is used to execute update defined on the attributes inside the <body>

<sql: query>
} body.
</sql: query>

```
<sql: update datasource = "${db}"
  var = "count">
  insert into students ( );
</sql: update>
```

NOTE:- Select statements are used only with <sql: query>
These stmts. can't be used with <sql: update>.

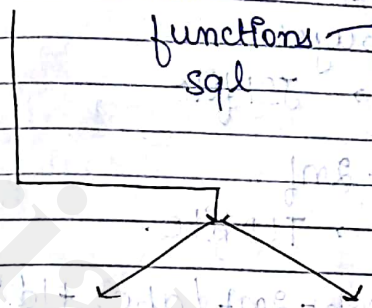
(4) sql: param - It is used for the set of (parameter) parameters that are passed with a specific value in any sql statement
{ can work like prepared stmts }

Sept 18

CLASSTIME | Page No.
Date / /

Ch-7
Custom Tags

- JSTL :- core functions → returns a value
  sql → doesn't directly execute values in tags. We need to store in variables.



* Req. to define Custom tag :-

(1) Java class (goes to source package)

(2) JSP page (where we define custom tags)

(3) TLD (Tag library descriptor)

<%@ taglib prefix="x" uri="/web-inf/" %>

eg :- <html> abc.tld.
<x:xyz>
</html>

user defined tag.

(we need to tell this prefix will fetch which uri)

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well

CLASSTIME | Page No.
Date / /

- Web Application
  ↓ (2 folders)
  Web pages
  ↳ JSP file
  Web-Inf
  ↳ TLD file.

∴ url = "/web-inf/abc.tld %>

tag has been embedded for its implementation, Javaclass & TLD runs.

↓

- TLD file - gives implementation of <x:xyz> custom tag.
  (in xml version)

```
<tag_lib>
<tlib_version> 1.0 </tlib_version>
<jsp_version> 2.0 </jsp_version>
<tag>                    (execution starts from here)
<name> xyz </name>
<tag-class> PQR </tag-class>   (filename.java)
</tag>
</tag_lib>
```

define name of tag.

{ extension not to be provided of file }.

Hex
defi
exec

- PQR

Java file

impo

clas

pub

J

implementat
that is to b
shown or

CLASSTIME | Page No.
Date / /

Here, if our tag name matches with defined tag, then this tag class PQR gets executed. PQR class has implementation of our tag.

- PQR class :- for this, we require a class. SimpleTagSupport

We extend this class and we have method doTag() which is overriden in our own class.

Java file

(package)
(tag extension)

```
import javax.servlet.jsp.tagext.*;
import java.io.*;

class PQR extends SimpleTagSupport {

public void doTag() throws JSPException,
                           IOException {

JSPWriter out = getJSPContext().getOut();

out.println("Good morning");
}}
```

out.println — (gets executed on browser)

getJSPContext() returns context of xyz (i.e this obj. of xyz) (reference of xyz)

implementation that is to be shown on browser. i.e on our JSP page.

CLASSTIME | Page No.
Date / /

NOTE:- In doTag(), we have all the implementation, thus we need some source of O/P and thus we use JSP Writer object.

{ Thus this <x:xyz> prints Good Morning }

NOTE:- If we define file PQR using a package, we use

packagename . filename.

WebApplication
- Webpages
- WebInf
- Source package
  - def
    - PQR.

eg:- def.PQR. → Source package.

NOTE:- For 3 custom tags, we will have only 1 TLD files only as TLD only maps corresponding tags to corresponding java classes.

NOTE:- getJSPContext().getOut()

get is used to out object of referrence of JSP page fetched in getJSPContext

CLASSTIME | Page No.
Date / /

Practical → (Star printing pattern).

<%. int n = request.getParameter()%>
< c:foreach
begin =
end =
variable =

Now when we use c:foreach 'n' is not accessible as 'n' is local to the scriplet tag

↓

Soln →

<c:set var="n" value=${param.text}"/>

var → by default, session/ scope of variable is throughout the page.

value → acc. to type of data. We are picking value & assinging it.

<c:forEach
begin =
end = "n" X "${n}"

as it will treat it as a string.

var = "x" >

(to be used in next loop)

10 Oct, 18

- Practical →

When we use
<%@ taglib prefix = "c" uri = "http://java.sun.com/jsp/jstl/core" %>

In practical, we need to include a file in libraries

Libraries →
Add library
↓
JSTL 1.2.1
↓
And if we expand it, we see all our files .core - - -

→ Custom Tags with Attributes -

① <c:out value
default
escapeXml
↓
For custom tags.
(core tags)

<x:xyz>
(prefix)

<x:xyz ab =
cd =
ef =
attr. of custom tags.
Implementation by user

CLASSTIME | Page No.
Date / /

eg:- <x:xyz ab = number (no. input)
cd = string/>(string input)

execution of page is never interpolated by JSP page. TLD and java file is responsible for execution.

- TLD ↓

<tag>
<name> xyz </name>
<tagclass> PQR </tagclass>

<attribute>
<name> ab </name>
</attribute> → If this is closed after </tag> then attribute doesn't consize into <tag> scope.
</tag>

multiple attributes ↓

<attribute>
<name> ab </name>
<name> cd </name>
</attribute> ↓

How to define DT of attribute field

<rtexprvalue> → specifies type of field works with req. tag and not alone.

req. text field value (only for attribute)

CLASS/TUT/E | Page No.
Date / /

<attribute>
{ <rtexprvalue> </rtexprvalue>,
<required> true </required>.
</attribute>
corresponds to particular name given

eg:-

<attribute>
<name> abc </name>
works for only abc. { <rtexprvalue>
<req> ____ </req> }
</attribute>

First use <req> and then <rtexpr value>.

- To define DT of attr fields (Java file)

class PGR extends SimpleTagSupport {
int ab ; String cd ; ⟶ mapping of attr. fields with same name as defined in TLD file.

public void doTag () {
}

Setter method for ab attribute

public void setab (int ab)
{
this.ab = ab;
}

(mapping through names)

public void setcd (String cd)
{
this.cd = cd;
}

CLASSTIME | Page No.
Date / /

NOTE:-
- carefully write names of setter methods same as TLD file names.
  check for case-sensitivity.

- When defining custom tags, we get error

Body content

<x:xyz/> / <x:xyz></x:xyz>

(empty body) — no implementation. } used for empty tags.

<body-content> scriptless </body-content>

{38 marks – JSP}

CLASSTIME | Page No.
Date / /

10 Oct, 18

Ch-8

Form Processing

imp

Name : [ ] (N)

Lucky number : [ ] (L) (1-100)

DOB : [ ] (D)

Gender : ○ M ○ F (G)

Food Items : [ ] Indian [ ] Chinese

[Submit]

} to be done in practical.

abc.jsp

I/c

Implementation using core tags. (JSTL)

We need to o/p data on same page

∴ form action = " " refer to same page.

<form action = "abc.jsp" >

</form>

} Designing of form.

If we execute some script in <form> it doesn't get executed when we load our ~~doc~~ page.

You have entered :

Name : <c:out value = "${param.N}"/>

fetch a particular value

name

prints null value until submit is clicked.

CLASSTIME | Page No.
Date / /

Lucky no: <c:out value = "${param.L}"/>

(it doesn't make an array until submit is clicked)

DOB: <c:out value = "${param.D}"/>

Gender: <c:out value = "${param.G}"/>

textfields ✓✓

(param deals with a single value)

But for checkboxes

↓

In jsp; we had

request.getParameterValues()

But for JSTL, we have

↓

(paramValues.)

↓

<c:forEach items = "${paramValues.}"
var = "x" >

no begin & end

<c:out value = "${x}"/>

</c:forEach>

can be accessed throughout page.

NOTE:-

== "" } on web, both
== "Null" } serve the same purpose of showing a null value.

- Form Validation-

Name : [ ] * Please enter Name } error msg. should get displayed.

degenerated
form has to be ~~disintegrated~~ with selected values.

To create submitted value. <input type = "hidden" name = "submitted" value = "true" />.

textbox & data are not visible

to check if data is submitted or not.

<c:if test = "${param.N == "" || param.N == "Null" && param.submitted}" />

If data is submitted.

0 && 1 = 0

If data hasn't been submitted.

Please enter name </c:if>

body executes if both are true.

combines into empty

param.N == "" || param.N == "Null" } In JS both are diff.

(empty)

empty param.N &&

For lucky no.:

<c:if test = "${empty param.L && param.submitted && param.L <= 100}" />

Please ---

</c:if>

Name: <input type = "text" value = <c:out value = "${ }" />

Cond. bl used w/o tag ??

Here expression language can be used without tag.

Name: <input type = "text" value = "${param.N}" />

value is an attr. of textbox

${ } can be used without <c: tag>.

{ same cond. for lucky no, DOB, name }

CLASSTIME | Page No.
Date

For radiobuttons → can have value/attribute

Radio button has no null value.

if test = "${param.G == 'm'}" checked (put in single quotes as double quotes are interpolated)

JSTL can use same properties as HTML

Checking only if any one of these radiobutton is selected. (can't decide about male/female property).

used when multiple items are needed.

<c:choose>
<c:when test = "${ == 'm' }"> (value of radio button)
<input type = "radio" value = "m" checked name = "G">

<input type = "radio" value = "f" name = "G">

<c:otherwise>
<input ------>

to display both radio buttons while replication
○ m ○ f.

- <c:if> → main str. of testing in JSP.

30 Oct, 18

CLASSTEAM Page No. Date / /

Form Processing And Validation

Parameters available in Request Header.

data that goes from app^n layer to physical layer.

(1) content length - This is of integer type that gives the body length of the request or -1 if the value is unknown.

Body content → no. of parameters (gender, name, lucky no.)

(2) content Type - (response.setContentType) (text/html)

(things available over the net) comes under category of MIME.

multipurpose internet multimedia extension

(all things on web like images, jpg, mp3, mp4, .gif)

MIME → string type.

(3) context path - (String DT) folder of server where we upload our project

container → server root → local pages.

Deployment descriptor → context Path. → reference path through which we access pages.

mapping of pages & corresponding files.

provides same structure as TLD.

web.xml

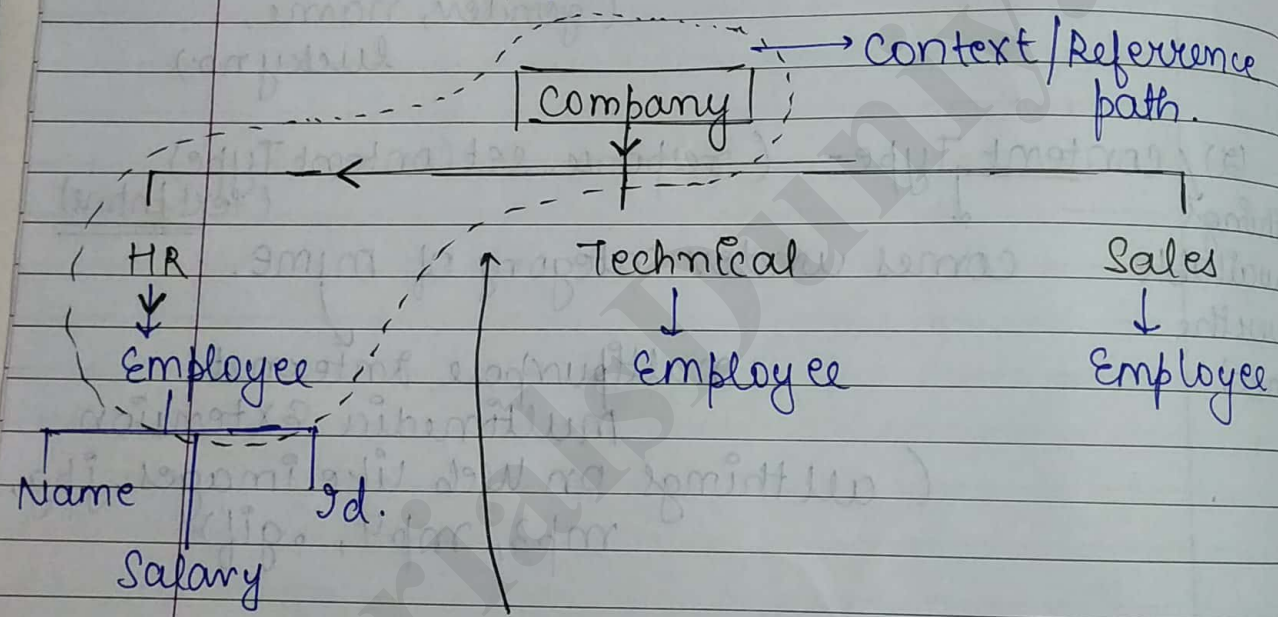
deployment descriptor for every file.

(tells where file is deployed)

For a single file, we don't need web.xml as no mapping is req.

WebApplication1

localhost/WebApplication1/abc.jsp.



www.company.com/HR?name="abc"

path followed to locate this will be our context.

/company/HR/Employee/name=abc

context path

machine.

machine.

(8) Remote Host - (String DT) it gives the client's host name or I/P address if name is unknown.

(9) Remote uri -

(10) Remote url -

(11) Remote User - This returns the user name used to make the request otherwise a null value is returned.

(12) Server Name -

(13) Server Port - port of service accessed.
(telnet → 21, SMTP → 25) eg HTTP (80)

(14) Secure- returns boolean value whether page is of secured server or not.

"${pagecontext.request.method}"

Putting in expression language tag as the content comes from a diff. page & so it needs to be evaluated.

"${pagecontext.request.remoteAddr}"

for access to current page (current page reference).

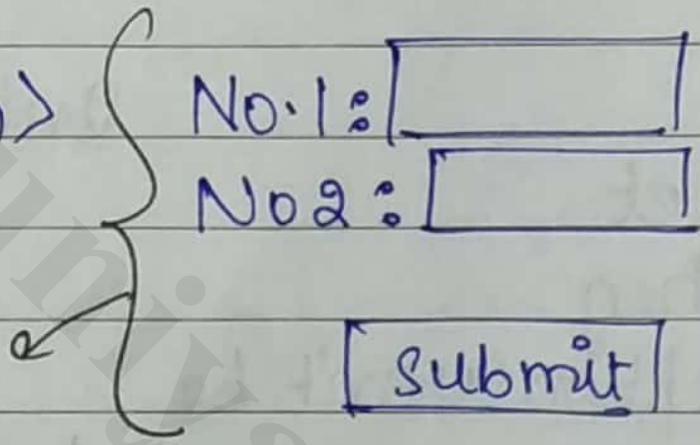
(10.52.2.1)

getJSPcontext → get context of JSP page.

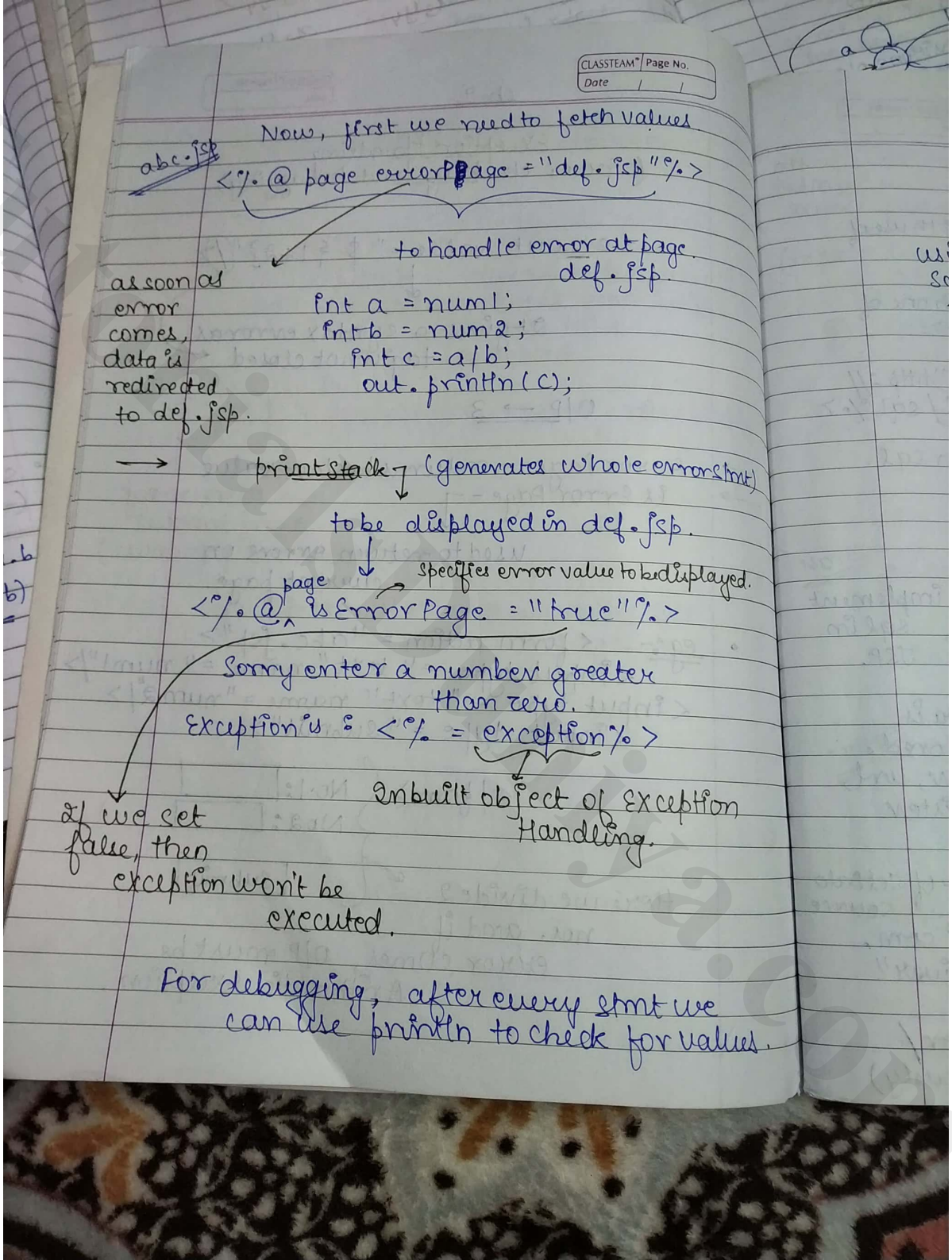
Used to return errors on current page.

- eg:-

```
<form action = "abc.jsp">
<input type = "text" name = "num1"/>
<input type = "text" name = "num2"/>.
<input type = "submit">
</form>
```



Here we divide 2 nos. and if error comes, O/P must be Arithmetic Exception.

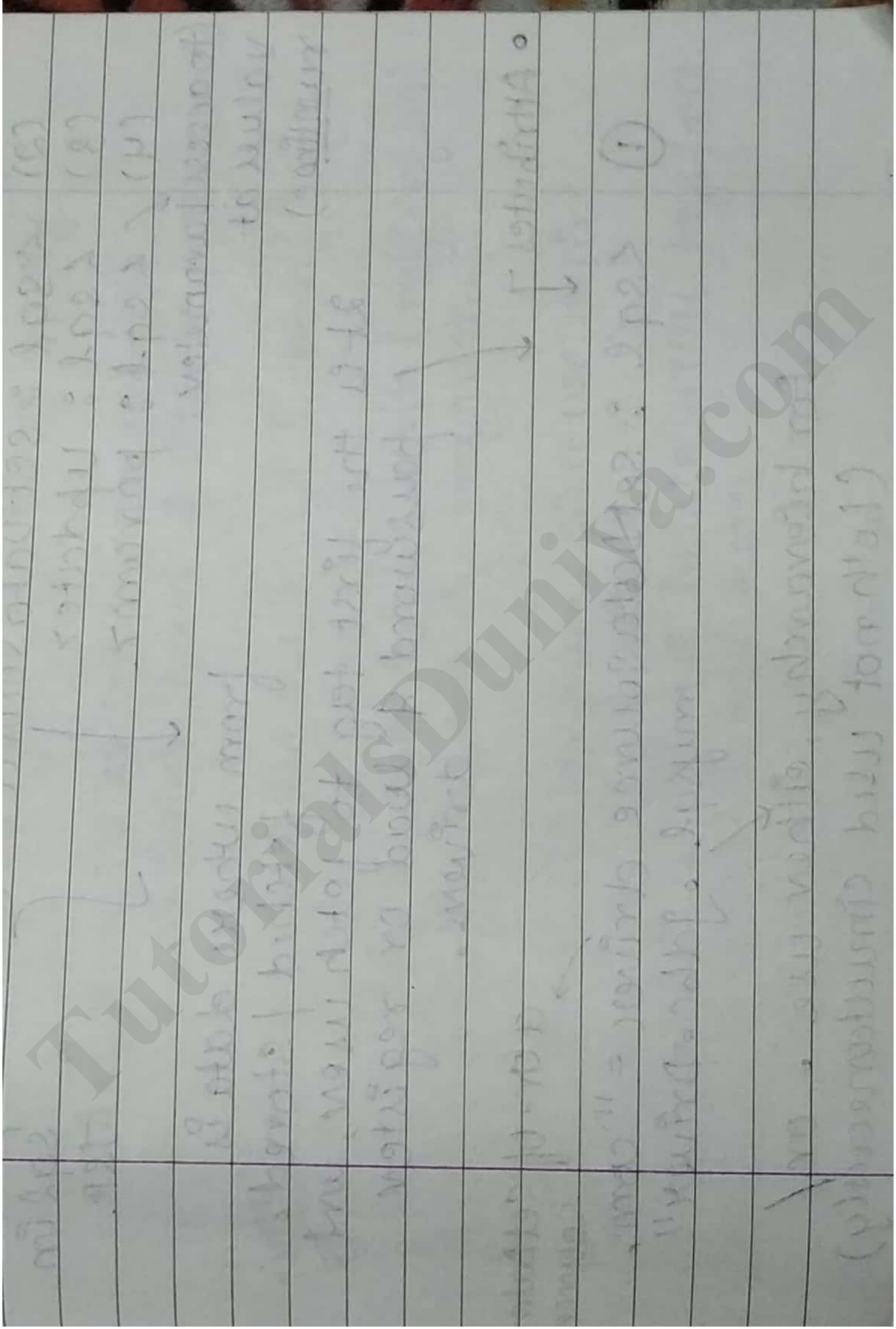


TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well



CLASSTEAM | Page No. | Date

10 oct, 18

Sql in JSP

Sql is embedded in JSP with use of JSTL tags

Includes 5 tags in which one of them is <sql>.

<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>

Includes tags for sql.

* main tags :-

(1) <sql:query>
(2) <sql:setDataSource>
(3) <sql:update>
(4) <sql:param>.

to implement sql in JSP.

(to access parameter values at runtime)

from where data is fetched / stored.

It is the first tag to feteh user, url, password & load or register drivers.

• Attributes

(1) <sql:setDataSource driver="com.mysql.jdbc.Driver"

attr. of setDataSource

For hierarchy, either use . or / (both not used simultaneously)

attr. of setDataSource { url = user = password

(to store current of DB re

(2) Creati

<sql

NOTE:

• Only

• In

NOTE:

NOTE:

</ sql : query >

NOTE:-

- Only select stmts come under < sql : query > & no other query.
- In < sql : update > we can use update, insert & delete commands

NOTE:- dataSource = " ${conn} ".

↙

This is done to use conn (reff. of DB) throughout the program.

NOTE:- For a JSP program, first we define <%@ taglib - L ----- >
and it is optional to use HTML tags / not as it is not req.

- Include
  - can be used throughout the program
  - If file not found while using include, it doesn't give error & program runs.
- import
  - used only once at top of program
  - If package not found, it gives an error.
- sql program using JSP :-

```
<%@ page import = "java.lo.*;
                   java.util.*" %>
                   uri = " "
<%@ taglib prefix = "sql" %>
<%@ taglib prefix="c" uri = " " %>

<sql:setDataSource var = "conn"
driver = "com.mysql.jdbc.Driver"
url = "jdbc:mysql://localhost/Student"
user = "root" password = " " />

<sql:query datasource = "${conn}"
var = "rs" />

select * from stud; </sql:
query>

<c:forEach var = "row" items="
```

${rs.rows}"> that has no. of rows

<c:out value = "${row.id}"/>

</c:forEach>.

works as rs.getInt("id")

row.id → defined column in DB.

current variable

NOTE:- Instead of rows, if we need to use colums then we use ${rs.cols}

NOTE:- Here, we don't need to close conn as everything is deployed on server & when we close tags, everything gets flushed out automatically.

- Update ↓

<sql:update dataSource = " "
var = " ">
Insert into stud values( )
</sql:update>.

To o/p these values
include tag <sql:query>
select * .... </sql:query>

and use forEach.

CLASSTEAM Page No.
Date / /

- **Delete**

```
<sql:update dataSource = " "
            var = " ">
    Delete from Stud where
            id = "100";
</sql:update>.
```

Q Design a form with 2 fields : name & roll no. and when we click submit data should be displayed on next page & also data must be inserted into DB.

el can't be used w/o out tag (i.e w/o c: tag)

```
<c:set var = "name" value = "${param.name}" />
<c:set var1 = "rno" value = "${param.rollno}" />
<sql:update dataSource = " "
    Insert into Stud values (var, var1);
</sql:update>.
```

Ch-2 and Ch-7

CLASSTEAM | Page No.
Date / /

- To display values at runtime ↓

```
<sql:update dataSource = " "
  Insert into Stud values (?,?,?);
<sql:param value = "${param.Name}"/>
```

first value replaces first ?
(Here we don't have concept of ordinal values. like in JDBC for ?).

```
<sql:param value = "${param.roll}"/>
<sql:param value = "${param.marks}"/>
</sql:param>
</sql:update>.
```

Generally used for GUI purposes.

Java Beans in JSP

normal Java class which is used to define/design s/w components i.e some reusable components that user can modify.

Button (generic class)

But implementation can differ acc. to user

→ Java Bean class is a specially constructed Java class which has the following characteristics :-

(1) It provides a default no argument constructor.
(2) It can have a no. of properties
(3) Corresponding getter and setter methods should be defined for the properties
(4) It should be serializable.

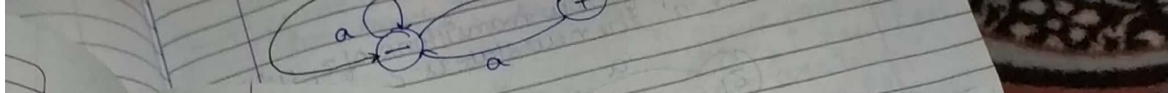
till the time the object is not flushed out from memory, changes must persist and this is done with help of serializable interface.

Program :- public class Student implements Java.io.Serializable {

Student()
{ }

setter methods.

getter methods



String name;
int age;

// now we require getter & setter methods since our constructor is empty. These methods are used to set initial values of data variables.

setter methods.

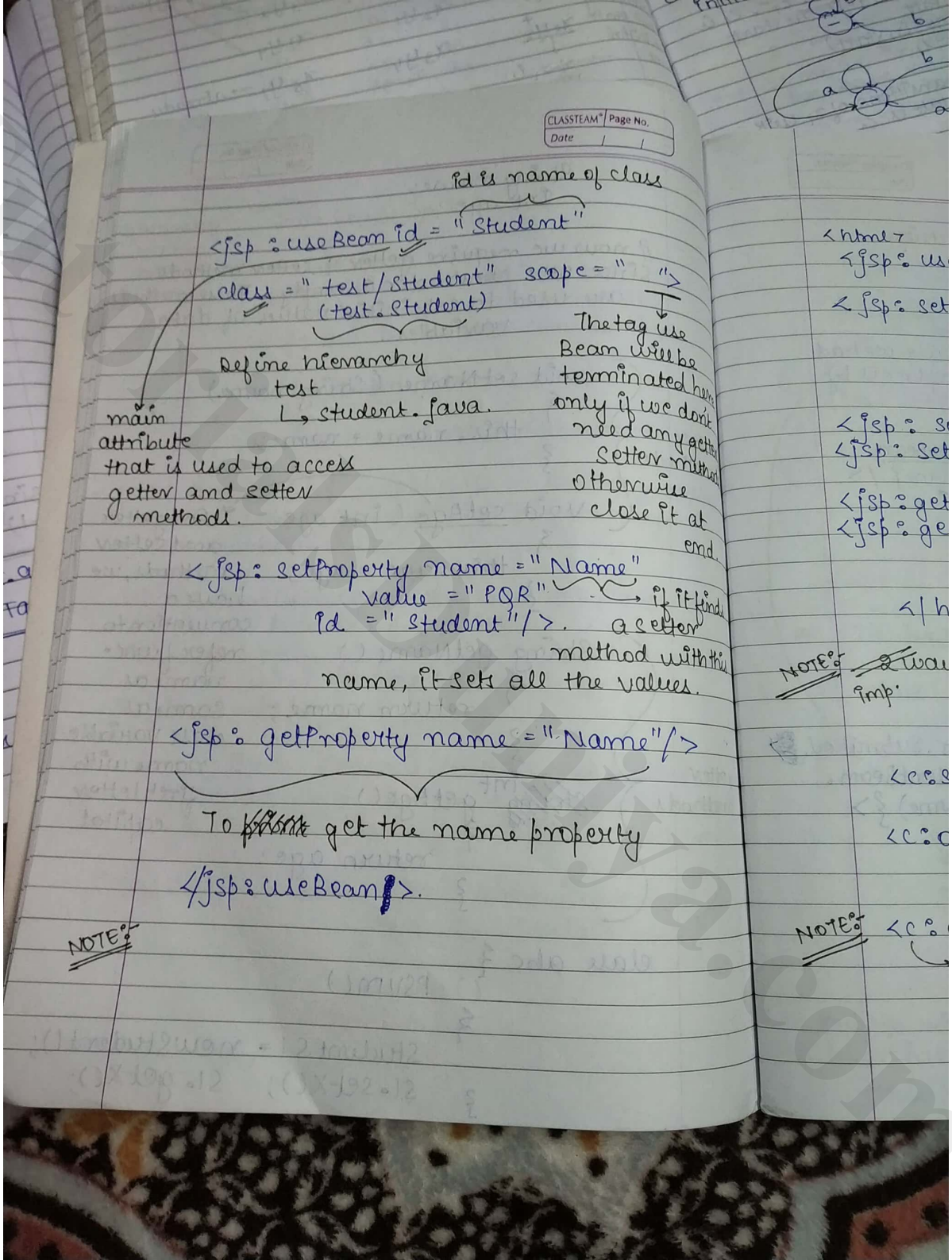
```
void setName (String name)
{
    this.name = name;
}

void setAge (int age)
{
    this.age = age;
}
```

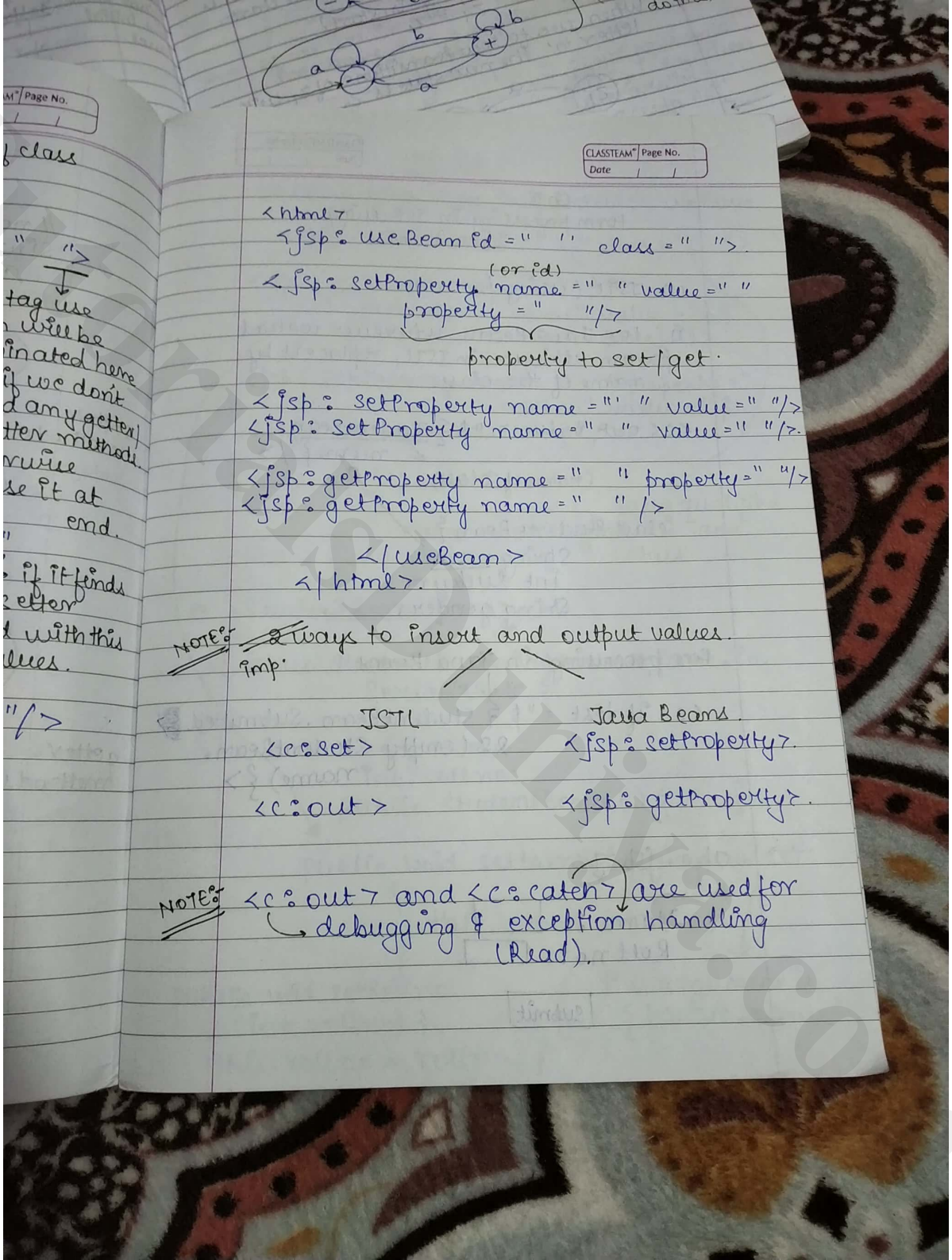
For getter and setter methods, we have a convention to refer func. name as same as class variable name with first letter capital.

getter methods

```
String getName()
{
    return name;
}

int getAge()
{
    return age;
}
```

```
class abc {
    PSVM()
    {
        Student S1 = new Student();
        S1.setX();  S1.getX();
    }
```





Ch-8

Form processing in JSP using Java Beans

(1) JSTL (using param).

(2) For Java Beans, wherever we had param attribute in JSTL, replace it by name of the class.

```
<c:out value = "${StudentBean.name}" />
```

(name of class).

```
class StudentBean {
    String name;
    int luckyno;
    String gender;
```

For processing in Java Beans

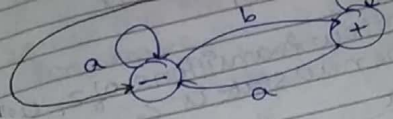
```
<c:if test = "${StudentBean.submitted
    && (empty StudentBean.name)}>
```

→ abc.jsp

Name : [ ]

Roll no : [ ]

[Submit]



def. jsp

CLASSTEAM Page No. Date

To send form data in javaclass, we have

<jsp:useBean id = " " class = " ">.

<jsp:setProperty name = " " property = "name" value = "${param.name}" />.

<jsp:setProperty name = " " property = "rollno" value = "${param.rollno}" />.

</jsp:useBean>.

fully qualified name used to find the java class.

* StudentBean.java

```
class StudentBean implements java.io.Serializable {

    String name;
    int rollno;
    public StudentBean() { }

    public void setName(String name)
    {
        this.name = name;
    }
    public void setRollno
        (int rollno) {
    this.rollno = rollno; }
```

this argument is ${param.name}

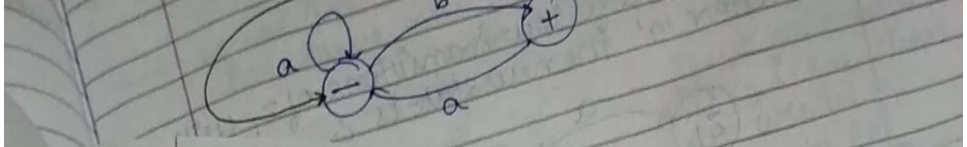
CLASSTEAM | Page No.
Date / /

NOTE:- In this class, we have only defined setter methods. ∴ If we try to use getProperty in the JSP file, then it will throw an error as we have no corresponding getter method.

NOTE:- We can use multiple <jsp:useBean>
eg:-

```
<jsp:useBean>
<jsp:getProperty name="  "
    property="  " />
</jsp:useBean>.
```

NOTE:- To validate for form values, we can provide implementation / methods in java file in setter method. This is usage of validation in Java Beans (controlling part is hidden, thus follows MVC model).



CLASSTEAM | Page No.
Date / /

Transaction processing (JDBC)

→ The statements that cause change in our schema are called transaction processing (Insert, Delete, Update), not select stmts. X

* 3 methods-

(1) setAutoCommit()

available in every class. has a default value "true".

To set it to false ↓

```
conn.setAutoCommit(false);
String sql = "Insert into Emp
              Values (1, "Enna");
Stmt.executeUpdate();
conn.commit();
```

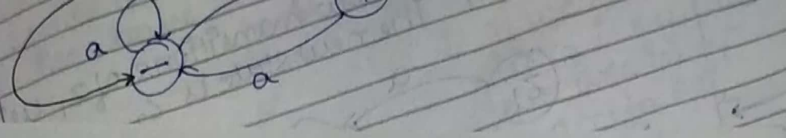
(2) to set the transaction to true.

(3) To rollback ↓ {random name. can be empty also}

use rollback().

```
conn.setAutocommit(false);
String sql = "          "
Savepoint sp = new setSavepoint("ab");
conn.rollback(sp); // or
St.rollback();
```

ref. of Savepoint.



provides reusability.

CLASSTEAM | Page No.
Date / /

Java Beans (ch-28)

brief description

* Introspection - getting class specifications (getters and setter) methods of a defined Java Bean class. 2 ways of introspection:-

- properties
  - getter property
  - setter property
- Bean Info Interface.
  - defines full description of the class.

There are 4 types of properties in Java Beans

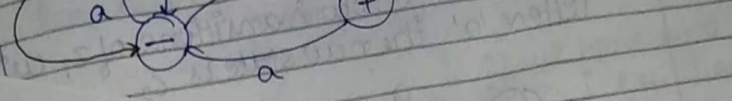
(1) simple { can have only 1 value }
eg:- getName()

(2) Indexed property { here we have multiple values & we need to get/set value of a particular index. }
(used for multivalued attributes)

eg:- Name - [First (0) | middle (1) | last (2)] { 3 values for same property }

(3) Bound properties.

(4) Constraint



5 Nov, 18

CLASSTEAM Page No. Date / /

Design Patterns

→ Java Beans - supports reusability using s/w components.

Has 3 things → properties
events
methods (always called on event)

For event handling, we have a model
Java Event Delegation model.

To use any event, register the corresponding listeners with the source.

On method_name
↳ using this on makes it an event.

* Introspection - class/process that fetches info. of properties, events or methods of a class

2 ways of introspection

Design patterns (properties).

Bean Info Interface.

① Simple properties.
② Indexed property.

CLASSTEAM Page No.
Date

② Bound properties
④ Constraints.

→ Introspection is a process of analysing a Beanclass to determine its capabilities (properties, methods & events).

This can be done in 2 ways

(1) with the help of properties
(2) By implementing the Bean Info Interface.

* Design pattern for properties

how we design our property
(has 4 patterns)

(1) Simple - A simple property has a single value and can be generalised with the help of foll. syntax

public T getN()
public void setN(T arg)
where N is the name of the property & T is its type.

(2) Indexed property - An indexed property consists of multiple values & can be identified with the syntax.

for fetching a single value { public T getN(int index)
public void setN(int index, T value)

for fetching all values

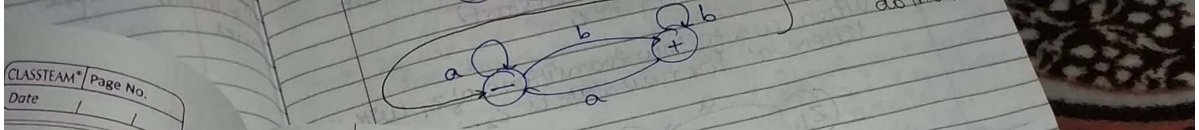
(3) Bound

changes property
i.e changi
boundary
condition

* B

(1) pr

(2)



CLASSTEAM Page No. Date

for fetching all values in terms of array.

public T[] getN()
public void setN(T values[])

(3) Bound properties - A Bean that has a bound property generates an event when we will try to change the property whereas if we attempt to change the value associated with that property, then corresponding property comes under the constraint property. The event name that will be ~~included~~ is reported by bound properties is ↓

changes property i.e changing boundary condition.

PropertyChangeListener.

(4) Constraint - used to change values of a property.

* Bean ~~Info~~ Interface :- (has 3 methods)

(1) property descriptor-

getName()

property name.

(2) method descriptor-

getName()

method name.

(3) Event Descriptor-

getListenerType.

For creating a BeanInfo class, it is compulsory to use Bean/ BeanInfo with classname

class StudentBeanInfo.

used so as to implement getListenerType.

→ When we have complex structures i.e multiple classes access our java class then serializibility comes into picture.

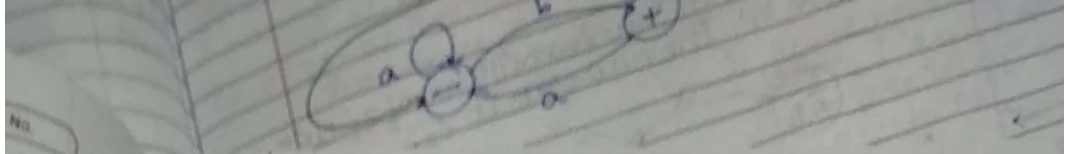
→ For Class and Obj., writing serializable is optional.

NOTE:- For design patterns, we only have property access & not method access

{ eg:- of student class }.

- Serializibility ↓

* Persistence - It is the ability to save the current state of a Bean also including the values & instance variables. This can be implemented with the help of Java.io



Serializable Interface.

If you don't want to save the current state, then the same can be done with the help of transient keyword.

(modifier)

If we add transient, then the thing becomes non-volatile.

→ For a Bean class, we have a package java.beans.*.

→ Serializable is implemented in a package java.io as it is related to I/P & O/P.

* Customizers - It is an interface through which a Bean developer can provide the implementation to change or to describe the state of the property.

(Table 28.1 and 28.2)

Q Design a XYZ Bean class (with given properties - name, marks, rno).

Q How to implement introspection.

TutorialsDuniya.com

Download FREE Computer Science Notes, Programs, Projects, Books PDF for any university student of BCA, MCA, B.Sc, B.Tech CSE, M.Sc, M.Tech at https://www.tutorialsduniya.com

- Algorithms Notes
- Artificial Intelligence
- Android Programming
- C & C++ Programming
- Combinatorial Optimization
- Computer Graphics
- Computer Networks
- Computer System Architecture
- DBMS & SQL Notes
- Data Analysis & Visualization
- Data Mining
- Data Science
- Data Structures
- Deep Learning
- Digital Image Processing
- Discrete Mathematics
- Information Security
- Internet Technologies
- Java Programming
- JavaScript & jQuery
- Machine Learning
- Microprocessor
- Operating System
- Operational Research
- PHP Notes
- Python Programming
- R Programming
- Software Engineering
- System Programming
- Theory of Computation
- Unix Network Programming
- Web Design & Development

Please Share these Notes with your Friends as well