

## **WEB PROGRAMMING LAB**

### **Course Objectives:**

- Learn to create WebPages using HTML5
- Learn to process XML documents using SAX/DOM API
- Learn to create dynamic web pages using server side scripting

### **Course Outcomes:**

- Design a Web site using HTML/DHTML and style sheets
- Create dynamic web pages using server side scripting
- Develop a web application with backend database connectivity

### **List of Experiments:**

1. Develop College Website using HTML5 and CSS\
2. Develop HTML5 form with client validations using Java Script
3. Publishing XML document using XSLT
4. XML document processing using SAX and DOM
5. Write a program to encrypt the given number to display the encrypted data using JavaScript
6. Write a Python program which generates an output file based on one-line instructions in an input file
7. Develop a simple Java Servlet application
8. Develop a Java Servlet application with session tracking
9. Develop a simple JSP application
10. Creation of an application to have access from a database using JDBC
11. Develop a full-fledged web application with database access spreading over to 3 session
12. Write a web application using Ajax to do the following:
  - A. check to make sure that the credit card number is composed of exactly 16 numerical digits
  - A check to make sure that a Visa card number starts with a "4" and a MasterCard Number starts with a "5" You can check for these things using regular expressions in combination with the PHP function preg\_match. A really good regex will allow for an optional“- “between every grouping of 4numbers. For example, 4111111111111111 and 4111-1111-1111-1111 would both be valid credit card numbers. If the user has not supplied a card number with the correct number of digits, show an error message.

## HTML5

1. There is a huge buzz surrounding HTML5. HTML5 is a major upgrade to HTML specification, last such upgrade happened some 11 years ago (XHTML 2000)
2. The update introduces new features like interactivity, smart forms, improved multimedia support, better semantics, offline support. But all this is in addition to the earlier features and the slate hasn't been wiped clean or everything done from the scratch.
3. Infact HTML5 is an umbrella term used to describe all the related set of technologies that are used to develop modern and rich in feature web contents. The most important ones are Cascading style sheets (CSS) and JavaScript

## Cascading Style Sheets :( CSS)

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

### 1. Develop College Website using HTML5 and CSS?

**AIM:** Design static webpage with the help of HTML5 and Style sheets (CSS)

#### Website.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>MCET</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1 id="logo">Methodist College of Engineering and Technology</h1>
    <nav>
      <ul>
        <li><a href="#home">Home</a></li>
        <li><a href="#about">About</a></li>
        <li><a href="#courses">Courses</a></li>
        <li><a href="#contact">Contact</a></li>
      </ul>
    </nav>
  </header>
  <main>
    <section id="home">
    </section>
    <section id="about">
      <h2>About</h2>
      <p>Methodist College of Engineering & Technology is a Non-Minority Educational institution, established in the year 2008, over 6.53 acre sprawling campus, situated at Abids, in the heart of the city of
```

pearls, Hyderabad, Telangana. The college is well connected by public transport from every corner of the city. MCET is affiliated to Osmania University at the state level and with AICTE in the Central level. </p>

```
</section>
<section id="courses">
  <h2>Courses</h2>
  <h3>Undergraduate Courses</h3>
  <ul>
    <li>Computer Science & Engineering</li>
    <li>Electronics & Communications Engineering</li>
    <li>Electrical & Electronics Engineering  </li>
    <li>Civil Engineering</li>
    <li>Mechanical Engineering</li>
  </ul>
  <h3>Post Graduate Courses</h3>
  <ul>
    <li>CAD/CAM  </li>
    <li>MBA</li>
  </ul>
</section>
<section id="contact">
  <form action="/">
    <h2>Contact Us</h2>
    <input placeholder="Your name" type="text" required><br><br>
    <input placeholder="Your Email Address" type="email" required><br><br>
    <textarea placeholder="Type your Message Here...." required></textarea><br><br>
    <button type="submit">Submit</button>
  </form>
</section>
<br>
<br>
</main>
</body>
</html>
```

### style.css

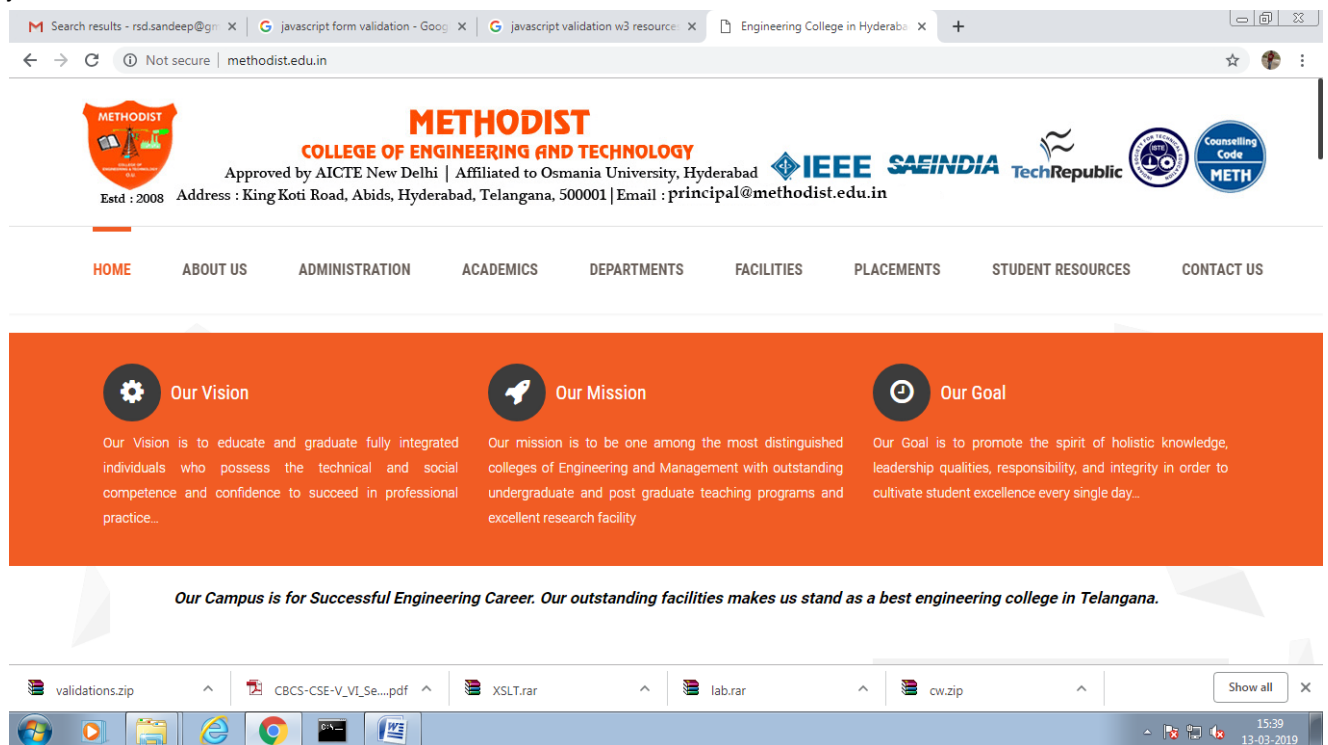
```
body {
  font-family: sans-serif;
  margin: 0;
}
main {
  max-width: 900px;
  margin: 0 auto;
}
header {
  position: fixed;
  top: 0;
  width: 100%;
  background-color: darkorange;
}
#logo {
  text-align: center;
}
nav {
  display: block;
```

```

margin: 0 auto;
max-width: 600px;
text-align: center;

}
nav ul {
list-style-type: none;
display: flex;
}
nav ul li {
flex: 1;
}
nav ul li a {
text-decoration: none;
text-transform: uppercase;
color: black;
}
nav ul li a:hover {
text-decoration: underline;
}
#home {
margin-top: 150px;
}

```



## 2. Develop HTML5 form with client validations using Java Script?

### JavaScript:

JavaScript is a scripting language most often used for *client-side web development*. Client-side refers to operations that are performed by the client (in our case the client is the browser) in a client-server relationship.

- Despite the name, JavaScript is essentially unrelated to the Java programming language.
- JavaScript was designed to add interactivity to HTML pages
- A scripting language is a lightweight programming language
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

### Index.html

```
<!DOCTYPE html>
<html lang="en"><head>
<meta charset="utf-8">
<title>JavaScript Field Label Form Validation using a registration form</title>
<meta name="keywords" content="example, JavaScript Form Validation, Sample registration form" />
<meta name="description" content="This document is an example of JavaScript Field label form validation using a sample registration form. " />
<link rel='stylesheet' href='js-field-level-form-validation.css' type='text/css' />
<script src="field-level-sample-registration-form-validation.js"></script>
</head>
<body onload="firstfocus();">
<h1>JavaScript Field Label Form Validation using a registration form</h1>
<h2>Use tab key or mouse to go next field</h2>
<form name='registration">
<ul>
<li><label for="userid">User id [5 to 12 characters] :</label></li>
<li><input type="text" name="userid" size="12" onblur="userid_validation(5,12)"/></li>
<li><label for="passid">Password [7 to 12 characters] :</label></li>
<li><input type="password" name="passid" size="12" onblur="passid_validation(7,12)"/></li>
<li><label for="username">Name [Alphabates characters] :</label></li>
<li><input type="text" name="username" size="50" onblur="allLetter()"/></li>
<li><label for="address">Address [alphanumeric characters] :</label></li>
<li><input type="text" name="address" size="50" onblur="alphanumeric()"/></li>
<li><label for="country">Country [Must select a country] :</label></li>
<li><select name="country" onblur="countryselect()">
<option selected="" value="Default">(Please select a country)</option>
<option value="AF">Australia</option>
<option value="AL">Canada</option>
<option value="DZ">India</option>
```

```

<option value="AS">Russia</option>
<option value="AD">USA</option>
</select></li>
<li><label for="zip">ZIP Code [Numeric characters only] :</label></li>
<li><input type="text" name="zip" onblur="allnumeric()" /></li>
<li><label for="email">Email [Valid email] :</label></li>
<li><input type="text" name="email" size="50" onblur="ValidateEmail()" /></li>
<li><label id="gender">Sex [Select Male or Female] :</label></li>
<li><input type="radio" name="sex" value="Male" checked /><span>Male</span></li>
<li><input type="radio" name="sex" value="Female" /><span>Female</span></li>
<li><label>Language [Optional] :</label></li>
<li><input type="checkbox" name="en" value="en" checked /><span>English</span></li>
<li><input type="checkbox" name="nonen" value="noen" /><span>Non English</span></li>
<li><label for="desc">About [Optional] :</label></li>
<li><textarea name="desc" id="desc"></textarea></li>
<li><input type="submit" name="submit" value="Submit" onclick="alert('Form submitted successfully')"
/></li>
</ul>
</form>
</body>
</html>

```

#### Field-level-sample-registration-form-validation.js

```

// After form loads focus will go to User id field.
function firstfocus()
{
var uid = document.registration.userid.focus();
return true;
}
// This function will validate User id.
function userid_validation(mx,my)
{
var uid = document.registration.userid;
var uid_len = uid.value.length;
if (uid_len == 0 || uid_len >= my || uid_len < mx)
{
alert("User Id should not be empty / length be between "+mx+" to "+my);
uid.focus();
return false;
}
// Focus goes to next field i.e. Password.
document.registration.passid.focus();
return true;
}
// This function will validate Password.
function passid_validation(mx,my)
{
var passid = document.registration.passid;

```

```

var passid_len = passid.value.length;
if (passid_len == 0 || passid_len >= my || passid_len < mx)
{
alert("Password should not be empty / length be between "+mx+" to "+my);
passid.focus();
return false;
}
// Focus goes to next field i.e. Name.
document.registration.username.focus();
return true;
}
// This function will validate Name.
function allLetter()
{
var uname = document.registration.username;
var letters = /^[A-Za-z]+$/;
if(uname.value.match(letters))
{
// Focus goes to next field i.e. Address.
document.registration.address.focus();
return true;
}
else
{
alert('Username must have alphabet characters only');
uname.focus();
return false;
}
}

// This function will validate Address.
function alphanumeric()
{
var uadd = document.registration.address;
var letters = /^[0-9a-zA-Z]+$/;
if(uadd.value.match(letters))
{
// Focus goes to next field i.e. Country.
document.registration.country.focus();
return true;
}
else
{
alert('User address must have alphanumeric characters only');
uadd.focus();
return false;
}
}

```

```

}
// This function will select country name.
function countrysselect()
{
var ucountry = document.registration.country;
if(ucountry.value == "Default")
{
alert('Select your country from the list');
ucountry.focus();
return false;
}
else
{
// Focus goes to next field i.e. ZIP Code.
document.registration.zip.focus();
return true;
}
}
// This function will validate ZIP Code.
function allnumeric()
{
var uzip = document.registration.zip;
var numbers = /^[0-9]+$/;
if(uzip.value.match(numbers))
{
// Focus goes to next field i.e. email.
document.registration.email.focus();
return true;
}
else
{
alert('ZIP code must have numeric characters only');
uzip.focus();
return false;
}
}
// This function will validate Email.
function ValidateEmail()
{
var uemail = document.registration.email;
var mailformat = /^[^w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
if(uemail.value.match(mailformat))
{
document.registration.desc.focus();
return true;
}
else

```



```
{
alert("You have entered an invalid email address!");
uemail.focus();
return false;
}
}
```

### Style.css

```
li {list-style-type: none;
font-size: 16pt;
}
.mail {
margin: auto;
padding-top: 10px;
padding-bottom: 10px;
width: 400px;
background : #D8F1F8;
border: 1px solid silver;
}
.mail h2 {
margin-left: 38px;
}
input {
font-size: 20pt;
}
input:focus, textarea:focus{
background-color: lightyellow;
}
input submit {
font-size: 12pt;
}
.rq {
color: #FF0000;
font-size: 10pt;
}
```

### 3. Publishing XML document using XSLT?

#### XML:

1. XML stands for Extensible **Markup Language**
2. XML is a **markup language** much like HTML.
3. XML was designed to **describe data**.
4. XML tags are not predefined in XML. You must **define your own tags**.
5. XML is **self describing**.
6. XML uses a DTD (**Document Type Definition**) to formally describe the data.

## What is XSLT?

- XSLT stands for XSL Transformations
- XSLT is the most important part of XSL
- XSLT transforms an XML document into another XML document
- XSLT uses XPath to navigate in XML documents
- XSLT is a W3C Recommendation
- XSLT is the most important part of XSL.
- XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML and XHTML. Normally XSLT does this by transforming each XML element into an (X)HTML element.
- With XSLT you can add/remove elements and attributes to or from the output file. You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.
- A common way to describe the transformation process is to say that **XSLT transforms an XML source-tree into an XML result-tree**.

#### cd.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type = "text/xsl" href = "cd.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
```

```

    <price>9.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <country>USA</country>
    <company>RCA</company>
    <price>9.90</price>
    <year>1982</year>
  </cd>
  <cd>
    <title>Stop</title>
    <artist>Sam Brown</artist>
    <country>UK</country>
    <company>A and M</company>
    <price>8.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Bridge of Spies</title>
    <artist>T`Pau</artist>
    <country>UK</country>
    <company>Siren</company>
    <price>7.90</price>
    <year>1987</year>
  </cd>
  <cd>
    <title>Private Dancer</title>
    <artist>Tina Turner</artist>
    <country>UK</country>
    <company>Capitol</company>
    <price>8.90</price>
    <year>1983</year>
  </cd>
</catalog>

```

### cd.xsl

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th style="text-align:left">Title</th>
      <th style="text-align:left">Artist</th>

```

```

</tr>
<xsl:for-each select="catalog/cd">
<tr>
<td><xsl:value-of select="title"/></td>
<td><xsl:value-of select="artist"/></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

## OUTPUT:

**My CD Collection**

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr. Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T' Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti
The dock of the bay	Otis Redding
Picture book	Simply Red

## 4. XML document processing using SAX and DOM?

### XML Parsers

An XML parser is a software library or package that provides interfaces for client applications to work with an XML document. The XML Parser is designed to read the XML and create a way for programs to use XML.

XML parser validates the document and check that the document is well formatted.

## Types of XML Parsers

These are the two main types of XML Parsers:

1. DOM
2. SAX

## DOM (Document Object Model)

A DOM document is an object which contains all the information of an XML document. It is composed like a tree structure. The DOM Parser implements a DOM API. This API is very simple to use.

## SAX (Simple API for XML)

A SAX Parser implements SAX API. This API is an event based API and less intuitive.

### **employee.xml**

```
<employees>
  <employee id="111">
    <firstName>amit</firstName>
    <lastName>abc</lastName>
    <location>hyd</location>
  </employee>
  <employee id="112">
    <firstName>sandeep</firstName>
    <lastName>rams</lastName>
    <location>Chennai</location>
  </employee>
  <employee id="113">
    <firstName>Rajesh</firstName>
    <lastName>Sharmain</lastName>
    <location>Pune</location>
  </employee>
</employees>
```

### **DOMParserDemo.java**

```
import java.util.ArrayList;
import java.util.List;
import org.w3c.dom.Document;
import org.w3c.dom.DocumentType;
import org.w3c.dom.Entity;
import org.w3c.dom.Element;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class DOMParserDemo {

    public static void main(String[] args) throws Exception {
        //Get the DOM Builder Factory
        DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();

        //Get the DOM Builder
        DocumentBuilder builder = factory.newDocumentBuilder();

        //Load and Parse the XML document
        //document contains the complete XML as a Tree.
```

```
Document document =  
    builder.parse(  
        ClassLoader.getResourceAsStream("employee.xml"));
```

```
List<Employee> empList = new ArrayList<>();
```

```
//Iterating through the nodes and extracting the data.
```

```
NodeList nodeList = document.getDocumentElement().getChildNodes();
```

```
for (int i = 0; i < nodeList.getLength(); i++) {
```

```
    //We have encountered an <employee> tag.
```

```
    Node node = nodeList.item(i);
```

```
    if (node instanceof Element) {
```

```
        Employee emp = new Employee();
```

```
        emp.id = node.getAttributes().
```

```
            getNamedItem("id").getNodeValue();
```

```
        NodeList childNodes = node.getChildNodes();
```

```
        for (int j = 0; j < childNodes.getLength(); j++) {
```

```
            Node cNode = childNodes.item(j);
```

```
            //Identifying the child tag of employee encountered.
```

```
            if (cNode instanceof Element) {
```

```
                String content = cNode.getLastChild().
```

```
                    getTextContent().trim();
```

```
                switch (cNode.getNodeName()) {
```

```
                    case "firstName":
```

```
                        emp.firstName = content;
```

```
                        break;
```

```
                    case "lastName":
```

```
                        emp.lastName = content;
```

```
                        break;
```

```
                    case "location":
```

```
                        emp.location = content;
```

```
                        break;
```

```
                }
```

```
            }
```

```
        }
```

```
        empList.add(emp);
```

```
    }
```

```
}
```

```
//Printing the Employee list populated.
```

```
for (Employee emp : empList) {
```

```
    System.out.println(emp);
```

```
}
```

```
}
```

```
class Employee{
```

```
    String id;
```

```
    String firstName;
```

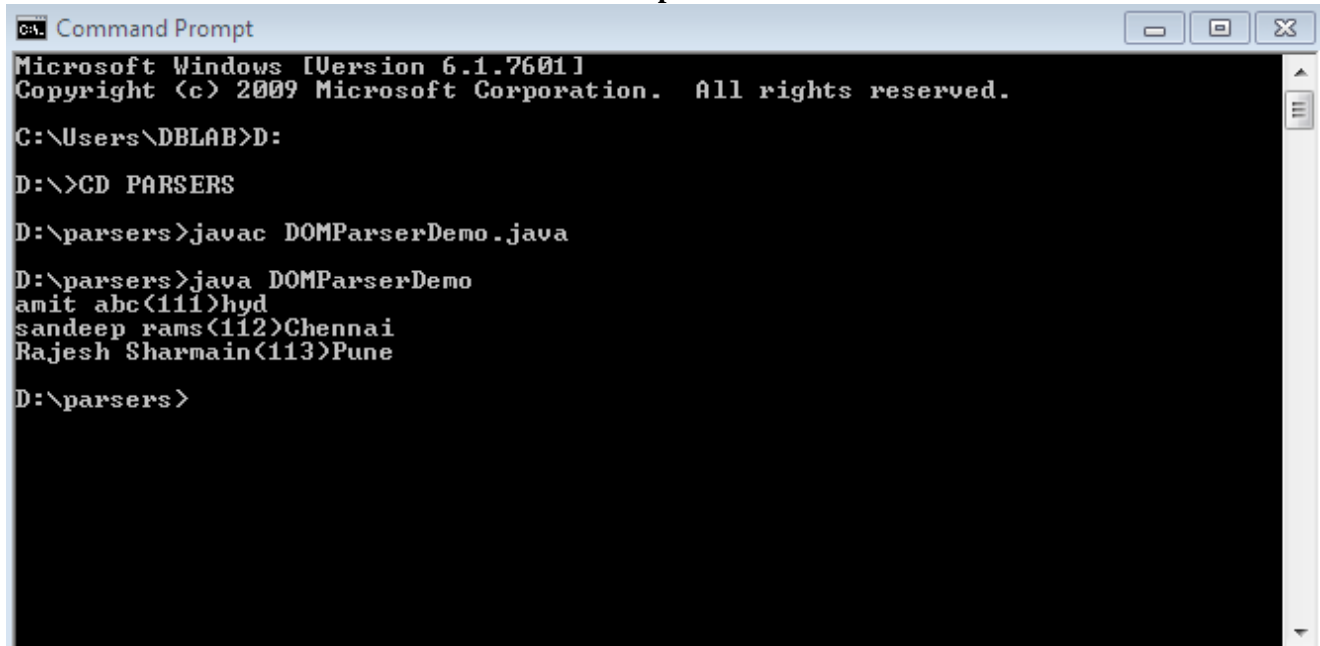
```
    String lastName;
```

String location;

@Override

```
public String toString() {  
    return firstName+" "+lastName+"("+id+")"+location;  
}  
}
```

### Output:



```
Command Prompt  
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.  
  
C:\Users\DBLAB>D:  
  
D:\>CD PARSERS  
  
D:\parsers>javac DOMParserDemo.java  
  
D:\parsers>java DOMParserDemo  
amit abc(111)hyd  
sandeep rams(112)Chennai  
Rajesh Sharmain(113)Pune  
  
D:\parsers>
```

### SAXParserDemo.java

```
import java.util.ArrayList;  
import java.util.List;  
import javax.xml.parsers.SAXParser;  
import javax.xml.parsers.SAXParserFactory;  
import org.xml.sax.Attributes;  
import org.xml.sax.SAXException;  
import org.xml.sax.helpers.DefaultHandler;  
  
public class SAXParserDemo {  
  
    public static void main(String[] args) throws Exception {  
        SAXParserFactory parserFactor = SAXParserFactory.newInstance();  
        SAXParser parser = parserFactor.newSAXParser();  
        SAXHandler handler = new SAXHandler();  
        parser.parse(ClassLoader.getResourceAsStream("employee.xml"),  
                    handler);  
  
        //Printing the list of employees obtained from XML  
        for ( Employee emp : handler.empList){  
            System.out.println(emp);  
        }  
    }  
}
```

```

/**
 * The Handler for SAX Events.
 */
class SAXHandler extends DefaultHandler {

    List<Employee> empList = new ArrayList<>();
    Employee emp = null;
    String content = null;
    @Override
    //Triggered when the start of tag is found.
    Public void startElement(String uri, String localName,
        String qName, Attributes attributes)
        throws SAXException {

        switch(qName){
            //Create a new Employee object when the start tag is found
            case "employee":
                emp = new Employee();
                emp.id = attributes.getValue("id");
                break;
        }
    }

    @Override
    public void endElement(String uri, String localName,
        String qName) throws SAXException {
        switch(qName){
            //Add the employee to list once end tag is found
            case "employee":
                empList.add(emp);
                break;
            //For all other end tags the employee has to be updated.
            Case "firstName":
                emp.firstName = content;
                break;
            case "lastName":
                emp.lastName = content;
                break;
            case "location":
                emp.location = content;
                break;
        }
    }

    @Override
    public void characters(char[] ch, int start, int length)
        throws SAXException {

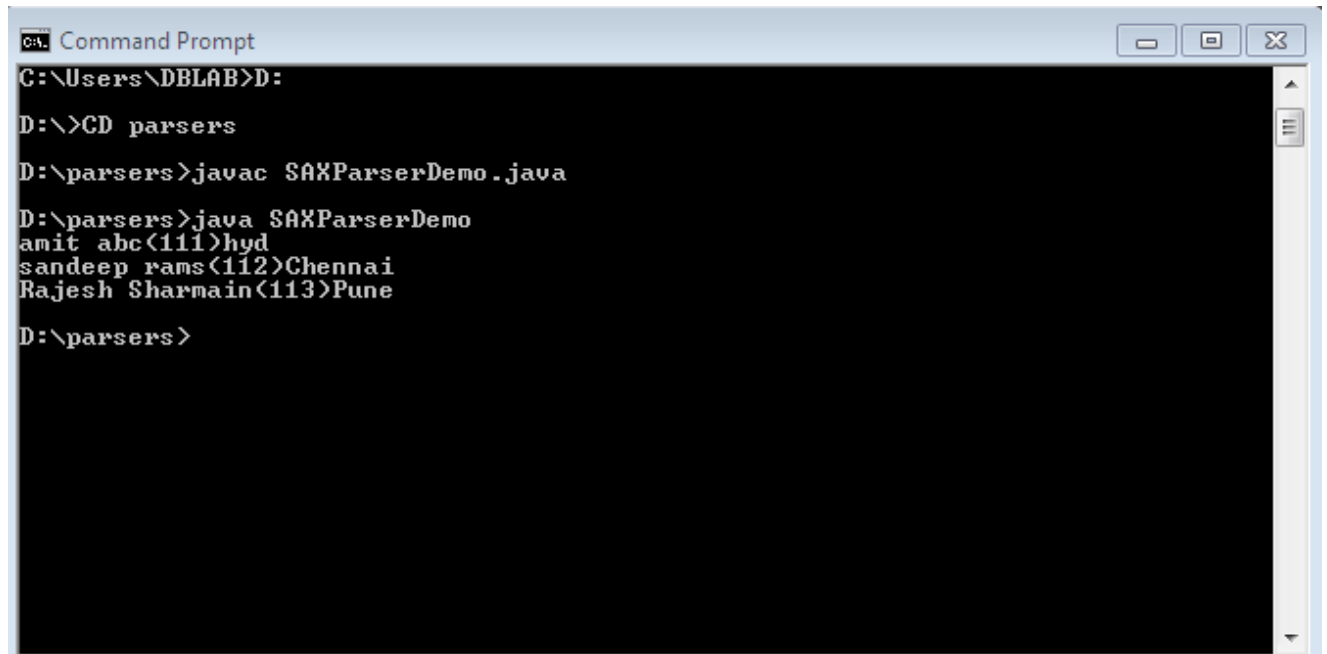
```



```
    content = String.valueOf(ch, start, length).trim();  
}  
}
```

```
class Employee {  
    String id;  
    String firstName;  
    String lastName;  
    String location;  
    @Override  
    public String toString() {  
        return firstName + " " + lastName + "(" + id + ")" + location;  
    }  
}
```

Output:



```
C:\Users\DBLAB>D:  
D:\>CD parsers  
D:\parsers>javac SAXParserDemo.java  
D:\parsers>java SAXParserDemo  
amit abc(111)hyd  
sandeep rams(112)Chennai  
Rajesh Sharmain(113)Pune  
D:\parsers>
```

##### 5. Write a program to encrypt the given number to display the encrypted data using JavaScript?

If you want to convert the information available with you into a code which is not accessible at everyone's disposal, then this is called encryption.

In JavaScript, the developer is allowed to use asymmetric type of encryption.

If you want to encrypt data in such a way that no one else gets the key to decrypt it.

Then you need to use asymmetric encryption.

In this type of encryption, you need to have separate keys for both encryption and decryption.

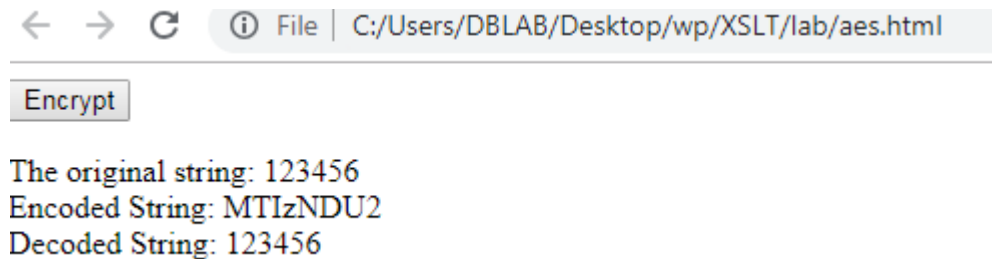
The key which is used for the decryption of code is available on the server.

On the other hand, if the developer has a lot of data, then it might take a lot of time.

## Encrypt.html

```
<!DOCTYPE html>
<html>
<body>
<button onclick="myFunction()">Encrypt</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "123456";
  var enc = window.btoa(str);
  var dec = window.atob(enc);
  var res = "The original string: " + str + "<br>" + "Encoded String: " + enc + "<br>" + "Decoded String: " + dec;
  document.getElementById("demo").innerHTML = res;
}
</script>
</body>
</html>
```

OUTPUT:



6. Write a Python program which generates an output file based on one-line instructions in an input file?

## File Operations Using Python – CRUD Operation in Python:

What are the various file operations that you can generally perform?

We call it **CRUD**. **CRUD** stands for:

- **Create**
- **Read**
- **Update**
- **Delete**

```

PRG:1
with open("in.txt") as f:
    lines = f.readlines()
    lines = [l for l in lines if "ROW" in l]
    with open("out.txt", "w") as f1:
        f1.writelines(lines)

```

```

PRG:2
f = open('list1.txt')
f1 = open('output.txt', 'a')

doIHaveToCopyTheLine=False

for line in f.readlines():

    if 'tests/file/myword' in line:
        doIHaveToCopyTheLine=True
    if doIHaveToCopyTheLine:
        f1.write(line)
f1.close()
f.close()

```

```

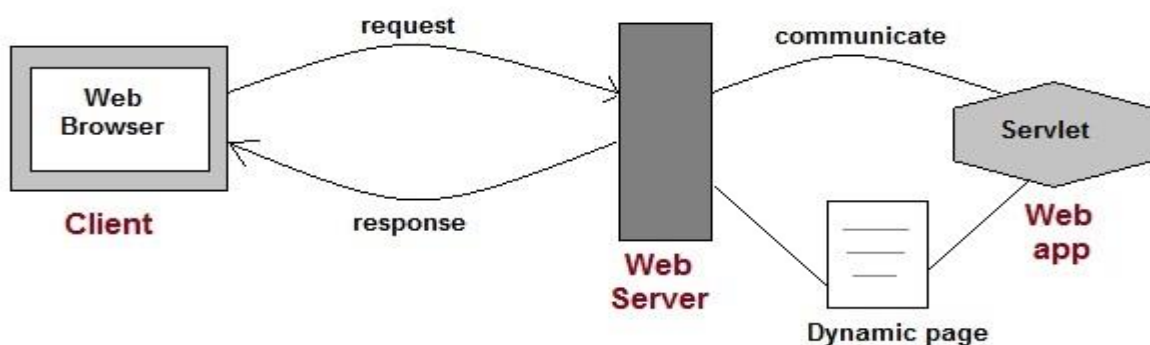
PRG3:
with open("in.txt") as f:
    with open("out.txt", "w") as f1:
        for line in f:
            if "ROW" in line:
                f1.write(line)

```

7. Develop a simple Java Servlet application?

**Servlet** Technology is used to create web applications. **Servlet** technology uses Java language to create web applications.

Web applications are helper applications that reside at web server and build dynamic web pages. A dynamic page could be anything like a page that randomly chooses picture to display or even a page that displays the current time.



## ExampleHttpServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
// Creating Http Servlet by Extending HttpServlet class
public class ExampleHttpServlet extends HttpServlet
{
    private String mymsg;
    public void init() throws ServletException
    {
        mymsg = "Http Servlet Demo";
    }
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException
    {
        // Setting up the content type of web page
        response.setContentType("text/html");
        // Writing the message on the web page
        PrintWriter out = response.getWriter();
        out.println("<h1>" + mymsg + "</h1>");
        out.println("<p>" + "Welcome to Servlet Programming!" + "</p>");
    }
    public void destroy()
    {
        // Leaving empty. Use this if you want to perform
        //something at the end of Servlet life cycle.
    }
}
```

## Web.xml

```
<web-app>
<display-name>BeginnersBookServlet</display-name>
<welcome-file-list>
<welcome-file>index.html</welcome-file>
<!--
<welcome-file>index.htm</welcome-file>
<welcome-file>index.jsp</welcome-file>
<welcome-file>default.html</welcome-file>
<welcome-file>default.htm</welcome-file>
<welcome-file>default.jsp</welcome-file>
-->
</welcome-file-list>

<servlet>
<servlet-name>MyHttpServlet</servlet-name>
<servlet-class>ExampleHttpServlet</servlet-class>
</servlet>

<servlet-mapping>
```

```
<servlet-name>MyHttpServlet</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

</web-app>
```

### Index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Http Servlet Demo</title>
</head>
<body>
<a href="/welcome">Click to call Servlet</a>
</body>
</html>
```

8) Develop a Java Servlet application with session tracking

- Session is basically a time frame and tracking means maintaining user data for certain period of time frame.
- **HTTP protocol** and **web servers** are stateless.
- All requests and responses are independent.
- Each request to the web server is treated as a new request.
- **Session Tracking** is a mechanism used by the web container to store session information for a particular user. It is used to recognize a particular user.

## Methods of Session Tracking

**There are four techniques used in Session Tracking:**

- 1) Cookies
- 2) Hidden Form Field
- 3) URL Rewriting
- 4) HttpSession

### 1) Cookies

Cookies are small piece of information sent by web server in response header and gets stored in browser side. A web server can assign a unique session ID to each web client. The cookies are used maintain the session. The client can disable the cookies.

### 2) Hidden Form Field

The hidden form field is used to insert the information in the webpages and this information is sent to the server. These fields are not viewable to the user directly.

**For example:**

```
<input type = hidden' name = 'session' value = '12345' >
```

### 3) URL Rewriting

Append some extra data through URL as request parameters with every request and response. URL rewriting is a better way to maintain session's management and work for the browsers.

**For example:**

```
http://tutorialride.com/servlet.htm;sessionid=54321
```

### 4) HttpSession Object

The HttpSession object represents a user session. The HttpSession interface creates a session between an HTTP client and HTTP server. A user session contains information about the user across multiple HTTP requests.

**For example:**

```
HttpSession session = request.getSession( );  
Session.setAttribute("username", "password");
```

Login.html

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="US-ASCII">  
<title>Login Page</title>  
</head>  
<body>  
  
<form action="LoginServlet" method="post">  
  
Username: <input type="text" name="user">  
<br>  
Password: <input type="password" name="pwd">  
<br>  
<input type="submit" value="Login">  
</form>  
</body>  
</html>
```

## LoginSuccess.jsp

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Success Page</title>
</head>
<body>
<%
String userName = null;
Cookie[] cookies = request.getCookies();
if(cookies !=null){
for(Cookie cookie : cookies){
    if(cookie.getName().equals("user")) userName = cookie.getValue();
}
}
if(userName == null) response.sendRedirect("login.html");
%>
<h3>Hi <%=userName %>, Login successful.</h3>
<br>
<form action="LogoutServlet" method="post">
<input type="submit" value="Logout" >
</form>
</body>
</html>
```

## Web.xml

```
<!--<?xml version="1.0" encoding="UTF-8"?> -->
<web-app>
<!-- xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd" id="WebApp_ID" version="3.0">
-->

<display-name>ServletCookieExample</display-name>
<welcome-file-list>
    <welcome-file>login.html</welcome-file>
</welcome-file-list>

    <servlet>
        <servlet-name>LoginServlet</servlet-name>
        <servlet-class>LoginServlet</servlet-class>
```

```

</servlet>
<servlet>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>LogoutServlet</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/LoginServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/LogoutServlet</url-pattern>
</servlet-mapping>

</web-app>

```

LoginServlet.java

```

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class LoginServlet
 */

public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final String userID = "cse";
    private final String password = "Methodist";

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {

        // get request parameters for userID and password
        String user = request.getParameter("user");
        String pwd = request.getParameter("pwd");
        if(userID.equals(user) && password.equals(pwd)){
            Cookie loginCookie = new Cookie("user",user);

```



```

        //setting cookie to expiry in 30 mins
        loginCookie.setMaxAge(30*60);
        response.addCookie(loginCookie);
        response.sendRedirect("LoginSuccess.jsp");
    }
    else{
        RequestDispatcher rd = getServletContext().getRequestDispatcher("/login.html");
        PrintWriter out= response.getWriter();
        out.println("<font color=red>Either user name or password is wrong.</font>");
        rd.include(request, response);
    }
}
}
}

```

#### LogoutServlet.java

```

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class LogoutServlet
 */
public class LogoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        response.setContentType("text/html");
        Cookie loginCookie = null;
        Cookie[] cookies = request.getCookies();
        if(cookies != null){
            for(Cookie cookie : cookies){
                if(cookie.getName().equals("user")){
                    loginCookie = cookie;
                    break;
                }
            }
        }
    }
}

```

```

        if(loginCookie != null){
            loginCookie.setMaxAge(0);
            response.addCookie(loginCookie);
        }
        response.sendRedirect("login.html");
    }
}

```

## 9) Develop a simple JSP application

### FirstJsp.jsp

```

<% @ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>First JSP</title>
</head>
<% @ page import="java.util.Date" %>
<body>
<h3>Hello ! Welcome to JSP Programming</h3><br>
<strong>Current Time is</strong>: <%=new Date() %>

</body>
</html>

```

## 10. Creation of an application to have access from a database using JDBC

To connect java application with the oracle database, we need to follow 5 following steps. In this example, we are using Oracle 10g as the database. So we need to know following information for the oracle database:

1. **Driver class:** The driver class for the oracle database is **oracle.jdbc.driver.OracleDriver**.
2. **Connection URL:** The connection URL for the oracle10G database is **jdbc:oracle:thin:@localhost:1521:xe** where jdbc is the API, oracle is the database, thin is the driver, localhost is the server name on which oracle is running, we may also use IP address, 1521 is the port number and XE is the Oracle service name. You may get all these information from the tnsnames.ora file.
3. **Username:** The default username for the oracle database is **system**.
4. **Password:** It is the password given by the user at the time of installing the oracle database.

## Create a Table

Before establishing connection, let's first create a table in oracle database. Following is the SQL query to create a table.

**Create table emp(id number(10),name varchar2(40),age number(3));**

We are connecting to an Oracle database and getting data from **emp** table.

Here, **system** and **oracle** are the username and password of the Oracle database.

### JDBCDemo.java

```
import java.sql.*;
class OracleCon{
public static void main(String args[]){
try{
//step1 load the driver class
Class.forName("oracle.jdbc.driver.OracleDriver");

//step2 create the connection object
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

//step3 create the statement object
Statement stmt=con.createStatement();

//step4 execute query
ResultSet rs=stmt.executeQuery("select * from emp");
while(rs.next())
System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));

//step5 close the connection object
con.close();
}catch(Exception e){ System.out.println(e);}
}
}
```

To connect java application with the Oracle database ojdbc14.jar file is required to be loaded.

<http://tinyurl.com/y39p7pyb>

## Two ways to load the jar file:

1. paste the ojdbc14.jar file in jre/lib/ext folder
2. set classpath

## 1) paste the ojdbc14.jar file in JRE/lib/ext folder:

Firstly, search the ojdbc14.jar file then go to JRE/lib/ext folder and paste the jar file here.

## 2) Set classpath:

There are two ways to set the classpath:

- temporary
- permanent

## How to set the temporary class path:

Firstly, search the ojdbc14.jar file then open command prompt and write:

1. C:>set classpath=c:\folder\ojdbc14.jar;.

## How to set the permanent classpath:

Go to environment variable then click on new tab. In variable name write **classpath** and in variable value paste the path to ojdbc14.jar by appending ojdbc14.jar; as  
C:\oracle\app\oracle\product\10.2.0\server\jdbc\lib\ojdbc14.jar;.

11. Develop a full-fledged web application with database access spreading over to 3 sessions

### Register\_1.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Guru Registration Form</title>
</head>
<body>
<h1>Register Form</h1>
<form action="guru_register" method="post">
<table style="width: 50%">
<tr>
<td>First Name</td>
<td><input type="text" name="first_name" /></td>
</tr>
<tr>
<td>Last Name</td>
<td><input type="text" name="last_name" /></td>
</tr>
<tr>
<td>UserName</td>
```

```

<td><input type="text" name="username" /></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="password" /></td>
</tr>
<tr>
<td>Address</td>
<td><input type="text" name="address" /></td>
</tr>
<tr>
<td>Contact No</td>
<td><input type="text" name="contact" /></td>
</tr></table>
<input type="submit" value="Submit" /></form>
</body>
</html>

```

← → ↻ ⓘ File | C:/Users/DBLAB/Desktop/login.htm

## Register Form

First Name	<input type="text"/>
Last Name	<input type="text"/>
UserName	<input type="text"/>
Password	<input type="password"/>
Address	<input type="text"/>
Contact No	<input type="text"/>
<input type="submit" value="Submit"/>	

register.java

```

package demotest;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class guru_register
 */
public class guru_register extends HttpServlet {
    private static final long serialVersionUID = 1L;

```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
// TODO Auto-generated method stub
String first_name = request.getParameter("first_name");
String last_name = request.getParameter("last_name");
String username = request.getParameter("username");
String password = request.getParameter("password");
String address = request.getParameter("address");
String contact = request.getParameter("contact");

if(first_name.isEmpty() || last_name.isEmpty() || username.isEmpty() ||
password.isEmpty() || address.isEmpty() || contact.isEmpty())
{
RequestDispatcher req = request.getRequestDispatcher("register_1.jsp");
req.include(request, response);
}
else
{
RequestDispatcher req = request.getRequestDispatcher("register_2.jsp");
req.forward(request, response);
}
}
}

```

## Register\_2.jsp

```

1. <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2. pageEncoding="ISO-8859-1"%>
3. <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
4. <html>
5. <head>
6. <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7. <title>Guru Success Page</title>
8. </head>
9. <body>
10. <a><b>Welcome User!!!!</b></a>
11. </body>
12. </html>

```

## Login and logout form

## Register\_3.jsp

```

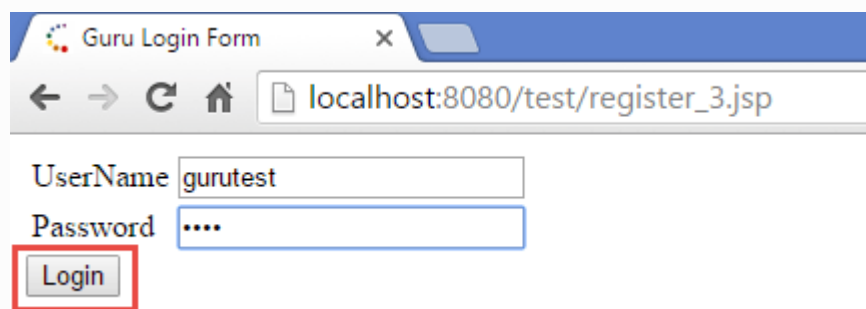
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

```

```

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Guru Login Form</title>
</head>
<body>
<form action="guru_login" method="post">
<table style="width: 50%">
  <tr>
<td>UserName</td>
<td><input type="text" name="username" /></td>
</tr>
<tr>
<td>Password</td>
<td><input type="password" name="password" /></td>
</tr>
</table>
<input type="submit" value="Login" /></form>
</body>
</html>

```



login.java(servlet)

```

package demotest;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class guru_login
 */
public class guru_login extends HttpServlet {

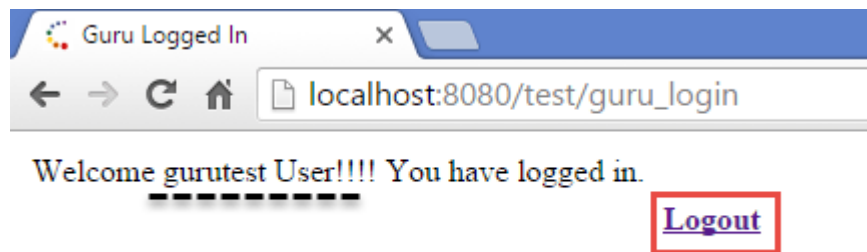
    public guru_login() {
        super();
        // TODO Auto-generated constructor stub
    }

```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
// TODO Auto-generated method stub
String username = request.getParameter("username");
String password = request.getParameter("password");
if(username.isEmpty() || password.isEmpty() )
{
RequestDispatcher req = request.getRequestDispatcher("register_3.jsp");
req.include(request, response);
}
else
{
RequestDispatcher req = request.getRequestDispatcher("register_4.jsp");
req.forward(request, response);
}
}
}
}

```



### Register\_4.jsp

```

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Guru Logged In</title>
</head>
<body>
<table style="width: 50%">
<tr><td>
<% String username = request.getParameter("username"); %>
<a>Welcome <% out.println(username); %> User!!!! You have logged in.</a></td></tr>
<tr></tr><tr><td></td><td></td><td><a href="register_3.jsp"><b>Logout</b></a></td></tr>
</table>
</body>
</html>

```

The screenshot shows a web browser window with the title 'Guru Login Form'. The address bar shows 'localhost:8080/test/register\_3.jsp'. The main content area displays a login form with two input fields: 'UserName' and 'Password'. Below the 'Password' field is a button labeled 'Login'.



12. Write a web application using Ajax to do the following:

A. check to make sure that the credit card number is composed of exactly 16 numerical digits

A check to make sure that a Visa card number starts with a "4" and a MasterCard Number starts with a "5" You can check for these things using regular expressions in combination with the PHP function preg\_match. A really good regex will allow for an optional "-" between every grouping of 4 numbers. For example, 4111111111111111 and 4111-1111-1111-1111 would both be valid credit card numbers. If the user has not supplied a card number with the correct number of digits, show an error message.

### Index.html

```
<table>
<tr>
<td>
Credit Card Number
</td>
<td>
<input type= "text" id="number_entered" onblur="validatecreditcard();" /><label id="validate"></label>
</td>
</tr>
<tr>
<td>
Expiration Date (MM/Year)
</td>
<td>
<input type= "text" id="expirationdate_entered" />
</td>
</tr>
<tr>
<td>
Security Code
</td>
<td>
<input type= "text" id="securitycode_entered" />
</td>
</tr>
<tr>
<td>
</td>
<td>
<input type="submit" value="Submit" />
</td>
</tr>
</table>
```

```

<script>
function validatecreditcard() {
var request;
try {
request= new XMLHttpRequest();

}
catch (tryMicrosoft) {
try {
request= new ActiveXObject("Msxml2.XMLHTTP");
}
catch (otherMicrosoft)
{
try {
request= new ActiveXObject("Microsoft.XMLHTTP");
}
catch (failed) {
request= null;
}
}
}
var url= "cardvalidation.php";
var cardnumber= document.getElementById("number_entered").value;
var expirationdate= document.getElementById("expirationdate_entered").value;
var securitycode= document.getElementById("securitycode_entered").value;
var vars= "creditcardnumber="+cardnumber+"&expiration="+expirationdate+"&security="+securitycode;
request.open("POST", url, true);
request.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
request.onreadystatechange= function() {
if (request.readyState == 4 && request.status == 200) {
var return_data= request.responseText;
document.getElementById("validate").innerHTML= return_data;
}
}
request.send(vars);
}
</script>

```

## Index.php

```
<?php
$number= $_POST['creditcardnumber'];
$expirationdate= $_POST['expiration'];
$securitycode= $_POST['security'];
function validatecard($number)
{
    global $type;
    $cardtype = array(
        "visa"    => "/^4[0-9]{12}(?:[0-9]{3})?$/",
        "mastercard" => "/^5[1-5][0-9]{14}$/",
        "amex"    => "/^3[47][0-9]{13}$/",
        "discover" => "/^6(?:011|5[0-9]{2})[0-9]{12}$/",
    );
    if (preg_match($cardtype['visa'],$number))
    {
        $type= "visa";
        return 'visa';
    }
    else if (preg_match($cardtype['mastercard'],$number))
    {
        $type= "mastercard";
        return 'mastercard';
    }
    else if (preg_match($cardtype['amex'],$number))
    {
        $type= "amex";
        return 'amex';
    }
    else if (preg_match($cardtype['discover'],$number))
    {
        $type= "discover";
        return 'discover';
    }
    else
    {
        return false;
    }
}
validatecard($number);
if (validatecard($number) !== false)
{
    echo "<green> $type detected. credit card number is valid</green>";
}
else
```

```
{  
echo " <red>This credit card number is invalid</red>";  
}
```

```
?>
```