WEEK-1:

AIM:

- Develop static pages (using only HTML) of an online Book store.
- The pages should resemble: www.amazon.com
- The website should consist of the following pages-
  Home page, Registration and user Login, User profile page, Books catalogue, Shopping cart, Payment By credit card, order confirmation.

DESCRIPTION:

HTML is the standard markup language for Web pages.

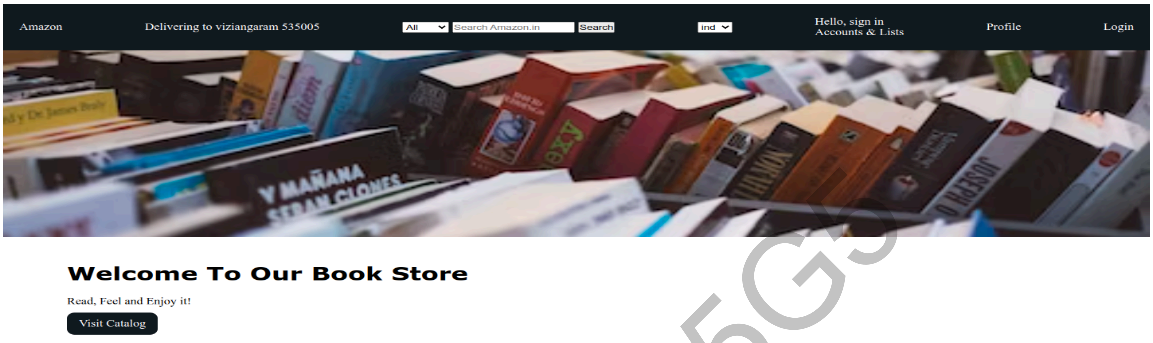
With HTML you can create your own Website.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <link rel="stylesheet" href="./styles.css">
   <title>Week-1</title>
</head>
<body>
   <div id="main">
       <header>
           <a href="./index.html">Amazon</a>
           <div class="address">
               <p>Delivering to vizianagaram 535005</p>
           </div>
           <div class="search-bar">
               <select name="category" id="category">
                   <option value="All">All</option>
                   <option value="books">books</option>
               </select>
               <input type="search" name="search-bar" placeholder="Search Amazon.in">
               <button>Search</button>
           </div>
           <select name="country" id="country">
               <option value="ind">ind</option>
           </select>
           <div>
               <p>Hello, sign in</p>
               <p>Accounts & Lists</p>
           </div>
           <div>
               <a href="./profile.html">Profile</a>
           </div>
           <div>
```

```
               <a href="./login.html">Login</a>
           </div>
       </header>
       <div id="home-head">
           <img src="./Home.png" alt="book-store">
           <div id="home-page">
               <a href="./catalog.html">Visit Catalogue</a>
           </div>
       </div>
   </div>
</body>
</html>
```

OUTPUT



Welcome To Our Book Store

Read, Feel and Enjoy it!

Register Page:

```
<div id="auth">
       <div id="register">
           <h1>Sign In</h1>
           <form>
               <input type="text" placeholder="Enter Your First Name">
               <input type="text" placeholder="Enter Your Middle Name">
               <input type="text" placeholder="Enter Your Last Name">
               <input type="email" placeholder="Enter Your Email">
               <input type="password" placeholder="Enter Your Password">
               <button>Sign In</button>
           </form>
       </div>
</div>
```

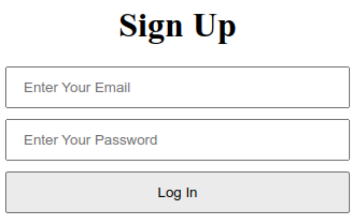
Output:

WEEK-2:

Login:

```
<div id="login">
    <h1>Sign Up</h1>
    <form>
        <input type="email" placeholder="Enter Your Email">
        <input type="password" placeholder="Enter Your Password">
         <button>Log In</button>
    </form>
</div>
```

Output:

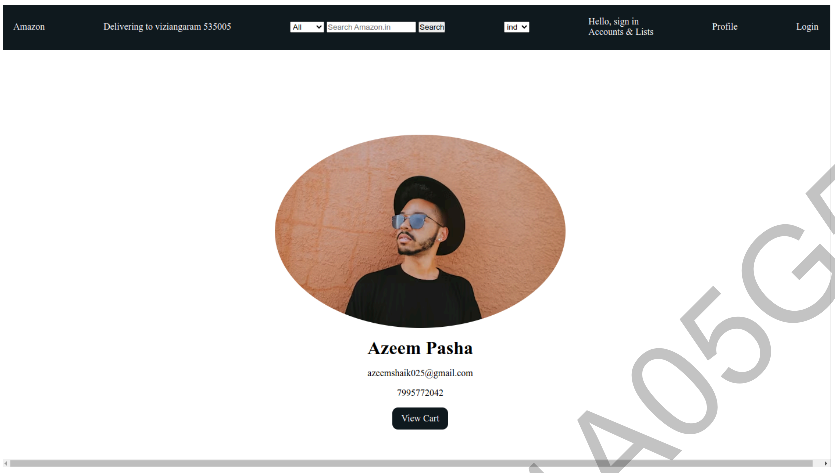


Profile :

```
<div id="main">
       <header>
           <a href="./index.html">Amazon</a>
           <div class="address">
               <p>Delivering to viziangaram 535005</p>
           </div>
           <div class="search-bar">
               <select name="category" id="category">
                   <option value="All">All</option>
               </select>
               <button>Search</button>
           </div>
           <div>
               <a href="./profile.html">Profile</a>
           </div>
           <div>
               <a href="./login.html">Login</a>
```

```
                </div>
            </header>
            <div id="profile-head">
                <img src="https://images.unsplash.com/photo" alt="Profile">
                <h1>Azeem Pasha</h1>
                <p>azeemshaik025@gmail.com</p>
                <p>7995772042</p>
                <a href="./cart.html">View Cart</a>
            </div>
        </div>
```

Output:



Azeem Pasha

azeemshaik025@gmail.com

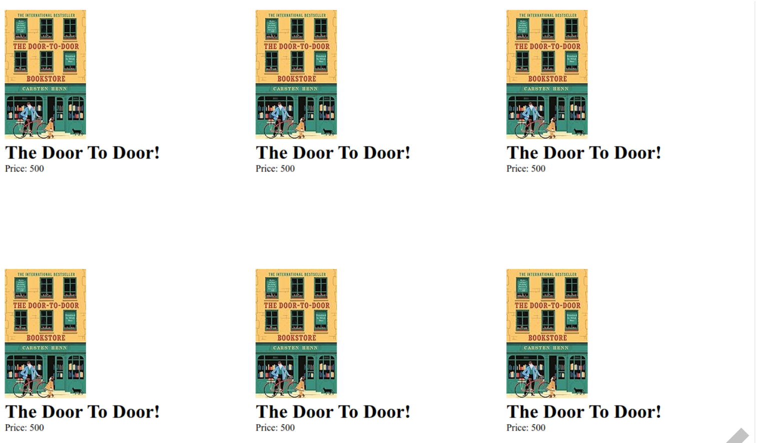
7995772042

Shopping cart :

```
<div id="cart-head">
            <h1>My Cart</h1>
            <div id="cart-body">
                <div class="cart-item">
                    <img src="https://m.media-amazon.com/images/" alt="Book">
                    <h1>The Door To Door!</h1>
                    <p>Price: 500</p>
                </div>
                    <p>Price: 500</p>
                </div>
                <div class="cart-item">
                    <img src="https://m.media-amazon.com/images/W/MEDIAX.jpg" alt="Book">
                    <h1>The Door To Door!</h1>
                    <p>Price: 500</p>
                </div>
                <div class="cart-item">
                    <img src="https://m.media-amazon.com/images/Wjpg" alt="Book">
                    <h1>The Door To Door!</h1>
                    <p>Price: 500</p>
                </div>
```

```
    </div>
</div>
```

Output:



The Door To Door!
Price: 500

The Door To Door!
Price: 500

The Door To Door!
Price: 500

The Door To Door!
Price: 500

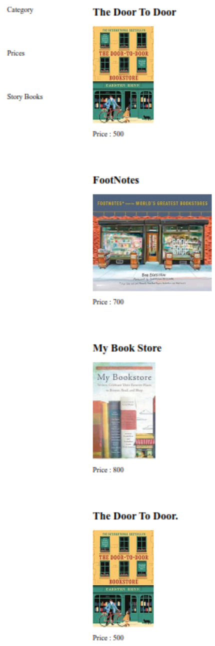
The Door To Door!
Price: 500

The Door To Door!
Price: 500

Books Catalog:

```
<main>
    <div id="side-bar">
        <p>Category</p>
        <p>Prices</p>
        <p>Story Books</p>
    </div>
    <div id="book-store">
        <section>
            <h1>My Book Store</h1>
            <img src="https://m.media-amazon.com/imag_AC_UY218_.jpg" alt="book">
            <p>Price : 800</p>
        </section>
    </div>
</main>
```

Output:



Category

Prices

Story Books

The Door To Door

Price : 500

FootNotes

Price : 700

My Book Store

Price : 800

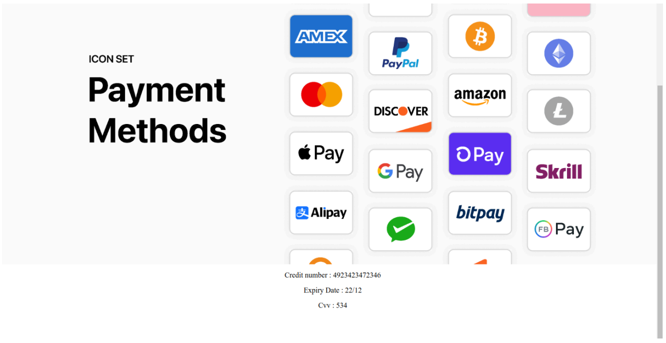
The Door To Door.

Price : 500

Payment page :

```
<div id="cred-head">
           <h1>Payment Process</h1>
           <img
src="https://s3-alpha.figma.com/hub/file/1491748397/23eff972-20c1-43ba-b27c-4bd7ed3c9916-cover.png"
alt="Payment">
           <p>Credit number : 4923423472346</p>
           <p>Expiry Date : 22/12</p>
           <p>Cvv : 534</p>
</div>
```

Output :



Credit number : 4923423472346

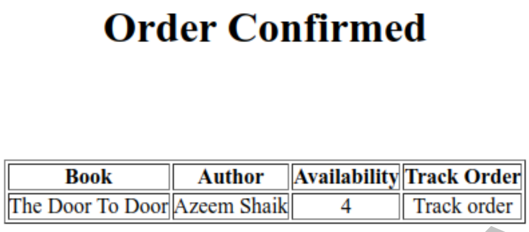
Expiry Date : 22/12

Cvv : 534

Order confirmation Page :

```
<div id="confirm-head">
    <h1>Order Confirmed</h1>
    <table border="1">
        <thead>
            <th>Book</th>
            <th>Author</th>
        </thead>
        </tbody>
    </table>
</div>
```

Output :



Order Confirmed

Book	Author	Availability	Track Order
The Door To Door	Azeem Shaik	4	Track order

WEEK-3:

AIM:
Develop and demonstrate the usage of inline, internal and external style sheet using CSS.
Design a web page using CSS which includes the following:
1) Use different font styles
2) Control the repetition of image with background-repeat and no-repeat property
3) Define style for links as a: link, a: active, a: hover, a: visited
4) Add customized cursors for links.

DESCRIPTION:

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

Internal CSS:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>INTERNAL CSS</title>
    <style>
        #home-head{
            display: flex;
            flex-direction: column;
        }
        #home-head img{
            width: 100%;
            height: 20em;
            object-fit: cover;
            z-index: -1;
        }

        #home-page{
            display: flex;
            flex-direction: column;
            padding: 2em 5em;
        }
        #home-page >*{
            padding: 10px 0em;
        }
        #home-page h1{
            font-weight: 5vw;
            font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida
Sans Unicode', Geneva, Verdana, sans-serif;
        }

        #home-page p{
            font-weight: 1vw;
        }
        #home-page a,#profile-head a{
            padding: 10px 15px;
            width: fit-content;
            background-color: #131921;
            border-radius: 10px;
            text-decoration: none;
            color: white;
        }
    </style>
</head>
<body>
    <div id="home-head">
        <img src="./Home.png" alt="book-store">
        <div id="home-page">
            <h1>Welcome To Our Book Store</h1>
            <p>Read, Feel and Enjoy it!</p>
            <a href="./catalog.html">Visit Catalog</a>
        </div>
    </div>
</body>
</html>
```

Inline CSS :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>INLINE CSS</title>
</head>
<body style="font-family:Arial, Helvetica, sans-serif">
    <div id="home-head" style="display: flex; flex-direction:column">
        <img style="width:100%; height:20em; object-fit:cover; z-index:-1; "
src="./Home.png" alt="book-store">
        <div id="home-page" style="display:flex; flex-direction:column; padding:2em
5em;">
            <h1 style="font-weight: 5vw; font-family: 'Lucida Sans', 'Lucida Sans
Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;">Welcome
To Our Book Store</h1>
            <p style="font-weight: 1vw;">Read, Feel and Enjoy it!</p>
```

```
           <a styl ="padding: 10px 15px; width: fit-content;  border-radius: 10px;
text-decoration: none; color: white; background-color: #131921;"
href="./catalog.html">Visit Catalog</a>
       </div>
   </div>
</body>
</html>
```

External CSS :

```
    #home-head{
       display: flex;
       flex-direction: column;
   }
   #home-head img{
       width: 100%;
       height: 20em;
       object-fit: cover;
       z-index: -1;
   }

   #home-page{
       display: flex;
       flex-direction: column;
       padding: 2em 5em;
   }
   #home-page p{
       font-weight: 1vw;
   }
   #home-page a,#profile-head a{
       padding: 10px 15px;
       width: fit-content;
       background-color: #131921;
       border-radius: 10px;
       text-decoration: none;
   }
```

WEEK 4:

AIM:

Develop and demonstrate JavaScript with POP-UP boxes and functions for the following problems:
a) Input: Click on Display Date button using onclick() function
Output: Display date in the textbox
b) Input: A number n obtained using prompt
Output: Factorial of n number using alert
c) Input: A number n obtained using prompt
Output: A multiplication table of numbers from 1 to 10 of n using alert
d) Input: A number n obtained using prompt and add another number using confirm
Output: Sum of the entire n numbers using alert

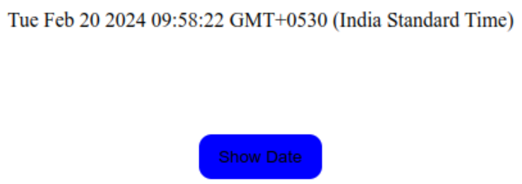
DESCRIPTION:

The **innerHTML** property sets or returns the HTML content (**inner HTML**) of an element.
The **prompt**() method displays a dialog box that **prompts** the user for input.
The **prompt**() method returns the input value if the user clicks "OK".
The **parseInt** method parses a value as a string and returns the first integer.
The **querySelector**() method returns the first element that matches a CSS selector.

CODE:

a) Show Date :

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="style.css">
   <title>Week 4</title>
</head>
<body>
   <div id="main">
       <p id="date"></p>
       <button onclick="showDate()">Show Date</button>
   </div>
   <script src="./script.js"></script>
</body>
</html>
const dateParagraph = document.querySelector("#date");
const showDate = ()=>{
   const date = new Date();
   dateParagraph.innerHTML = `${date}`;
}
```

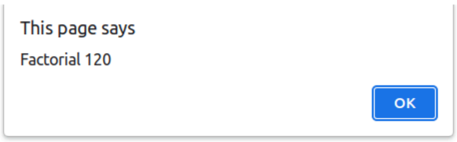


Tue Feb 20 2024 09:58:22 GMT+0530 (India Standard Time)

b) Factorial Program

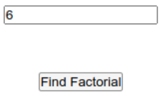
```
<body>
    <div id="factorial">
        <input type="number" placeholder="Enter the number" name="number">
        <button type="submit" onclick="factorial()">Find Factorial</button>
    </div>
    <script>
        const input = prompt("Enter Your Number : ");
        const factorial = (e) =>{
            let fac = 1;
            let n = input;
            for(var i=1;i<=n;i++){
                fac = fac * i;
            }
            alert(`Factorial ${fac}`);
        }
    </script>
</body>
```

Output :



This page says

Factorial 120

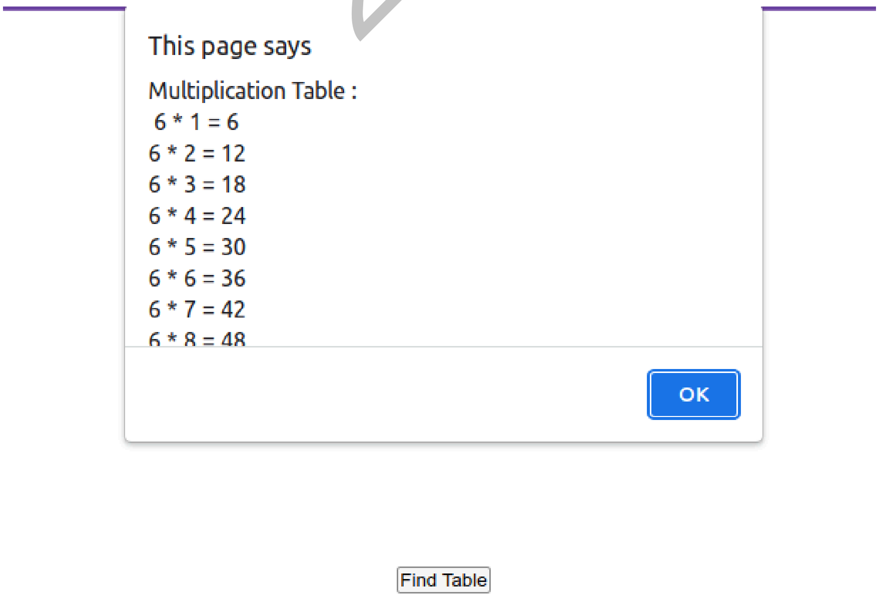


6

c) Multiplication Table

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=devic-width, initial-scale=1.0">
   <link rel="stylesheet" href="style.css">
   <title>Multiplication Table</title>
</head>
<body>
   <div id="table">
       <button type="submit" onclick="multiplication()">Find Table</button>
   </div>
   <script>
       const input = prompt("Enter Your Number : ");
       const multiplication = ()=>{
           const value = input;
           let s = '';
           for(var i=1;i<=10;i++){
               x = `${value} * ${i} = ${value * i}\n`;
               s += x;
           }
           alert(`Multiplication Table : \n ${s}`)
       }
   </script>
</body>
</html>
```

Output :

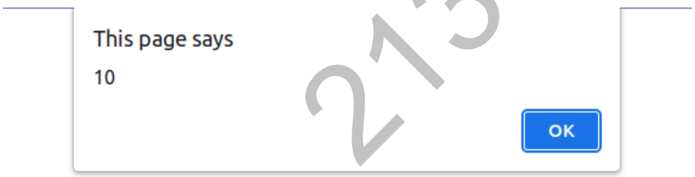


```
This page says
Multiplication Table :
 6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
```

d) Sum of two numbers

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="style.css">
   <title>Sum of two numbers</title>
</head>
<body>
   <div id="sum">
       <button onclick="add()">Add</button>
   </div>
   <script>
       const input = prompt("Enter The Number A : ")
       const inputB = prompt("Enter The Number B : ");
       const add = ()=>{
           const sum = Number.parseInt(input) + Number.parseInt(inputB);
           alert(sum);
       }
   </script>
</body>
</html>
```

Output :



This page says

10

WEEK 5:

AIM:

Write JavaScript to validate the following fields of the Registration page.
1. First Name (Name should contains alphabets and the length should not be less than 6 characters).
2. Password (Password should not be less than 6 characters length).
3. E-mail id (should not contain any invalid and must follow the standard pattern name@domain.com)
4. Mobile Number (Phone number should contain 10 digits only).
5. Last Name and Address (should not be Empty).

DESCRIPTION:

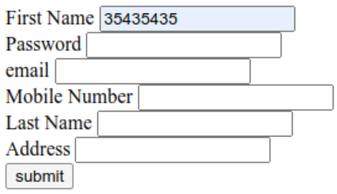
Validating fields on a registration page using JavaScript is typically done on the client side to provide immediate feedback to users before they submit the form.
The **alert**() method displays an **alert** box with a message and an OK button. The **alert**() method is used when you want information to come through to the user. The **getElementById**() method returns an element with a specified value. The **getElementById**() method returns null if the element does not exist.

CODE:

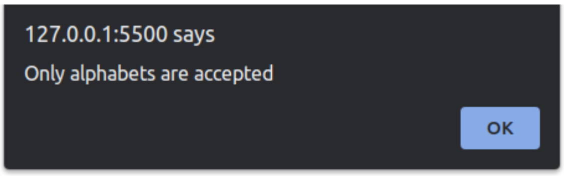
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <style>
       body{
           display: flex;
           align-items: center;
           justify-content: center;
           width: 100dvw;
           height: 100dvh;
       }
       form{
           display: flex;
           flex-direction: column;
           align-items: center;
           justify-content: space-around;
           height: 25dvh;
       }
       form div{
           width: 25dvw;
           width: 100%;
       }
   </style>
   <title>w-5 q-1</title>
</head>
<body>
   <form name="myform">
```

```
        <div>
            <label>First name : </label>
            <input type="text" name="firstname"/>
        </div>
        <div>
            <label>Password : </label>
            <input type="password" name="password"/>
        </div>
        <div>
            <label>Email : </label>
            <input type="text" name="email"/>
        </div>
        <div>
            <label>Mobile number : </label>
            <input type="text" name="mobilenumber"/>
        </div>
        <div>
            <label>Last name : </label>
            <input type="text" name="lastname"/>
        </div>
        <div>
            <label>Address : </label>
            <input type="text" name="address"/>
        </div>
        <button onclick="handleFormValidation()" type="submit">Submit</button>
    </form>
    <script>
        const handleFormValidation = (e)=>{
            const firstname = document.myform.firstname.value;
            const firstnameregex = /^[A-Za-z]*$/;
            if(firstname.length < 6 && firstnameregex.test(firstname)){
                alert("Enter valid first name!");
            }
            else{
                alert("Valid first name!");
            }
        }
    </script>
</body>
</html>
```

Output :

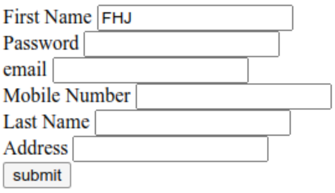


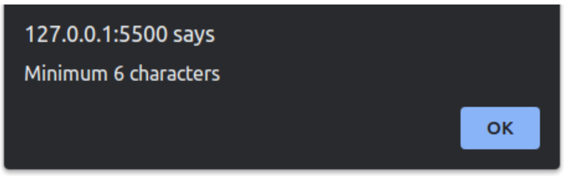
First Name 35435435
Password
email
Mobile Number
Last Name
Address

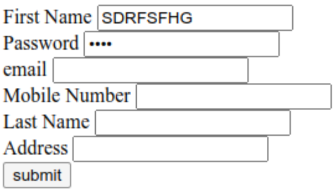


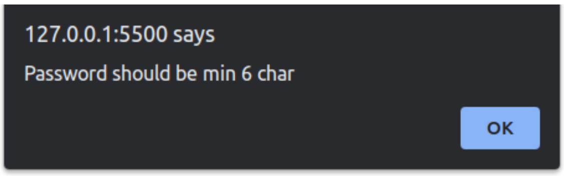
127.0.0.1:5500 says

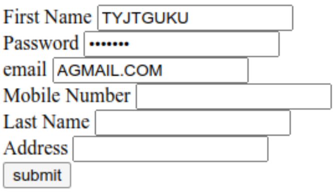
Only alphabets are accepted

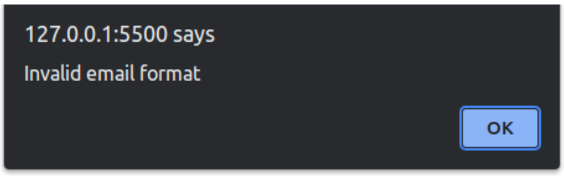

First Name FHJ
Password
email
Mobile Number
Last Name
Address

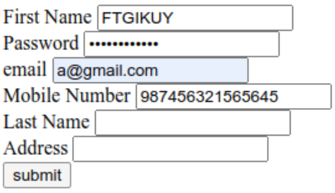

127.0.0.1:5500 says
Minimum 6 characters

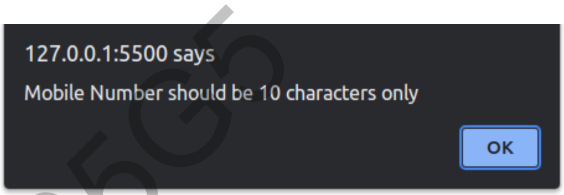

First Name SDRFSFHG
Password ••••
email
Mobile Number
Last Name
Address

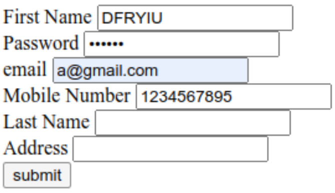

127.0.0.1:5500 says
Password should be min 6 char

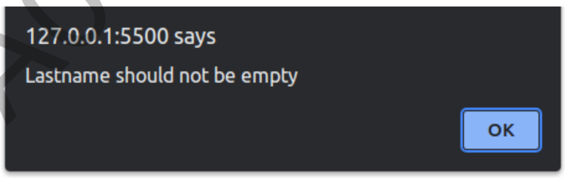

First Name TYJTGUKU
Password •••••••
email AGMAIL.COM
Mobile Number
Last Name
Address


127.0.0.1:5500 says
Invalid email format


First Name FTGIKUY
Password ••••••••••••
email a@gmail.com
Mobile Number 987456321565645
Last Name
Address


127.0.0.1:5500 says
Mobile Number should be 10 characters only


First Name DFRYIU
Password ••••••
email a@gmail.com
Mobile Number 1234567895
Last Name
Address


127.0.0.1:5500 says
Lastname should not be empty

WEEK 6:

AIM:

Validate the registration, user login, user profile and payment by creditcard pages using JavaScript.

DESCRIPTION:

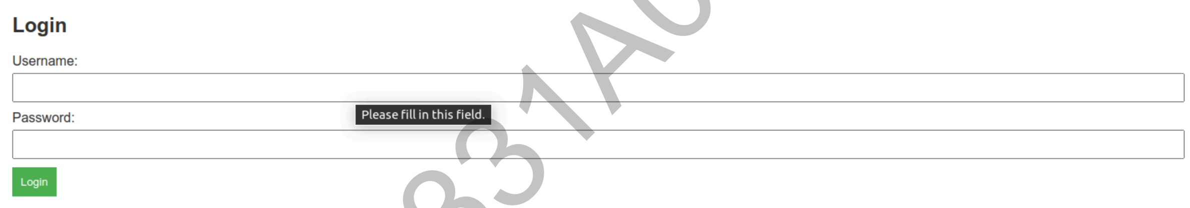
Validating registration, user login, user profile, and payment by credit card pages using JavaScript involves implementing client-side validation to ensure that users provide accurate and appropriate information.

CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>User Registration and Login</title>
</head>
<body>

<!-- Registration Page -->
<div id="registrationPage">
   <h2>Registration</h2>
   <form id="registrationForm">
       <label for="username">Username:</label>
       <input type="text" id="username" required>
       <br>
       <label for="password">Password:</label>
       <input type="password" id="password" required>
       <br>
       <button type="button" onclick="registerUser()">Register</button>
   </form>
</div>

<!-- Login Page -->
<div id="loginPage" style="display: none;">
   <h2>Login</h2>
   <form id="loginForm">
       <label for="loginUsername">Username:</label>
       <input type="text" id="loginUsername" required>
       <br>
       <label for="loginPassword">Password:</label>
       <input type="password" id="loginPassword" required>
       <br>
       <button type="button" onclick="loginUser()">Login</button>
   </form>
</div>
```

```
<!-- User Profile Page -->
<div id="userProfile" style="display: none;">
    <h2>User Profile</h2>
    <p id="loggedInUser"></p>
    <button type="button" onclick="logoutUser()">Logout</button>
    <button type="button" onclick="showPaymentPage()">Proceed to Payment</button>
</div>

<!-- Payment by Credit Card Page -->
<div id="paymentPage" style="display: none;">
    <h2>Payment by Credit Card</h2>
    <form id="paymentForm">
        <label for="creditCardNumber">Credit Card Number:</label>
        <input type="text" id="creditCardNumber" required>
        <br>
        <label for="expiryDate">Expiry Date:</label>
        <input type="text" id="expiryDate" placeholder="MM/YY" required>
        <br>
        <button type="button" onclick="processPayment()">Process Payment</button>
    </form>
</div>

<script>
    function registerUser() {
        // Perform registration logic
        alert('User registered successfully!');
        showLoginPage();
    }

    function loginUser() {
        // Perform login logic
        let username = document.getElementById('loginUsername').value;
        document.getElementById('loggedInUser').innerText = `Logged in as: ${username}`;
        showUserProfile();
    }

    function logoutUser() {
        // Perform logout logic
        document.getElementById('loggedInUser').innerText = '';
        showLoginPage();
    }

    function processPayment() {
        // Perform payment logic
        alert('Payment processed successfully!');
    }

    function showLoginPage() {
        document.getElementById('registrationPage').style.display = 'none';
        document.getElementById('loginPage').style.display = 'block';
        document.getElementById('userProfile').style.display = 'none';
        document.getElementById('paymentPage').style.display = 'none';
    }
    function showUserProfile() {
        document.getElementById('registrationPage').style.display = 'none';
        document.getElementById('loginPage').style.display = 'none';
        document.getElementById('userProfile').style.display = 'block';
        document.getElementById('paymentPage').style.display = 'none';
    }

    function showPaymentPage() {
        document.getElementById('registrationPage').style.display = 'none';
        document.getElementById('loginPage').style.display = 'none';
        document.getElementById('userProfile').style.display = 'none';
        document.getElementById('paymentPage').style.display = 'block';
    }

    // Initial display
    showLoginPage();
</script>

</body>
</html>
```

Output :

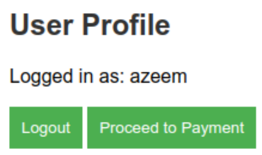


Login

Username:

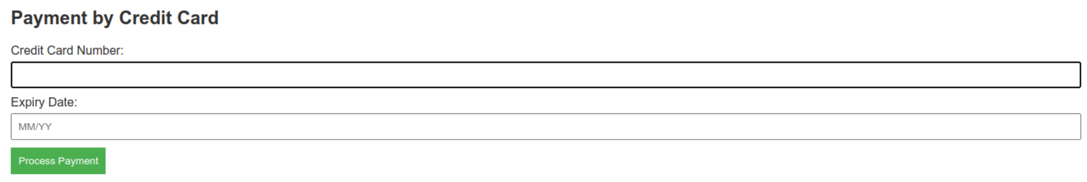
Please fill in this field.

Password:



User Profile

Logged in as: azeem

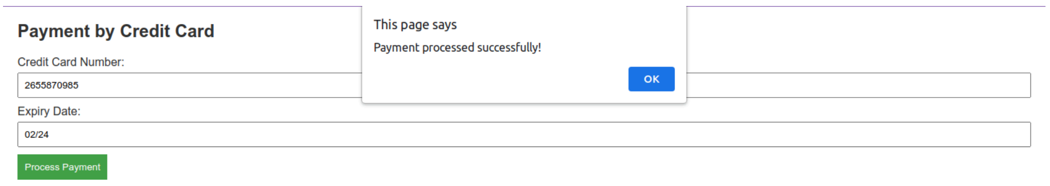


Payment by Credit Card

Credit Card Number:

Expiry Date:

MM/YY



Payment by Credit Card

Credit Card Number:

2655870985

Expiry Date:

02/24

This page says

Payment processed successfully!

WEEK 7:

AIM:

Write an XML file which will display the Book information which includes the following:
1) Title of the book
2) Author Name
3) ISBN number
4) Publisher name
5) Edition
6) Price

DESCRIPTION:

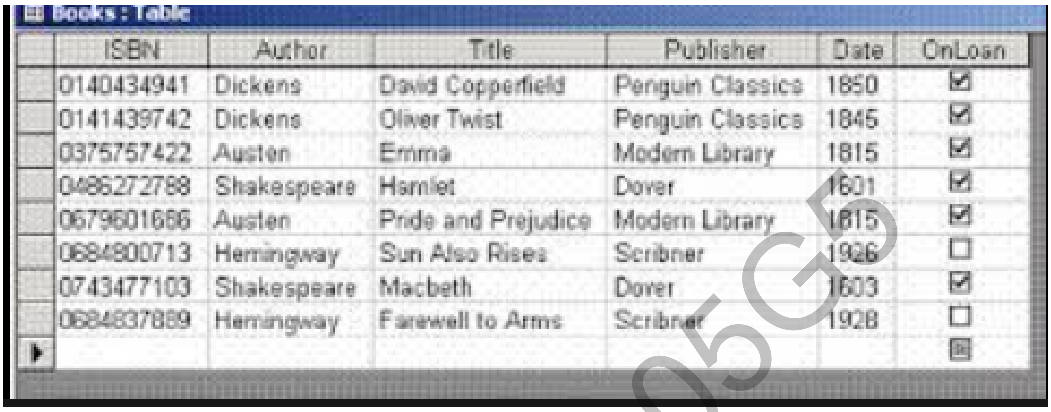
XML stands for eXtensible Markup Language.
XML was designed to store and transport data.
XML was designed to be both human- and machine-readable.

CODE:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE bookstore SYSTEM "bookstore.dtd">
<bookstore>
   <book>
       <title>Introduction to XML</title>
       <author>Jane Doe</author>
       <isbn>978-1234567890</isbn>
       <publisher>ABC Publications</publisher>
       <edition>2nd Edition</edition>
       <price>29.99</price>
   </book>
   <book>
       <title>Data Science Essentials</title>
       <author>John Smith</author>
       <isbn>978-0987654321</isbn>
       <publisher>XYZ Publishers</publisher>
       <edition>1st Edition</edition>
       <price>45.99</price>
   </book>
</bookstore>
```

```
<?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>
     <head>
       <style>
         table {
           border-collapse: collapse;
           width: 100%;
         }
         th, td {
           border: 1px solid #dddddd;
           text-align: left;
           padding: 8px;
         }
         th {
           background-color: grey;
           color: white;
         }
         .author {
           text-transform: uppercase;
           font-weight: bold;
           color: blue; /* Use your own color */
         }
         .isbn {
           color: green; /* Use your own color */
         }
         .publisher {
           color: orange; /* Use your own color */
         }
         .edition {
           color: purple; /* Use your own color */
         }
         .price {
           color: red; /* Use your own color */
         }
       </style>
     </head>
     <!-- xsltproc book_transform.xsl book_info.xml -o output.html -->
     <body>
       <table>
         <tr>
           <th>Title</th>
           <th>Author</th>
           <th>ISBN</th>
           <th>Publisher</th>
           <th>Edition</th>
           <th>Price</th>
         </tr>
         <xsl:for-each select="bookstore/book">
           <tr>
             <td><xsl:value-of select="title"/></td>
             <td class="author"><xsl:value-of select="author"/></td>
             <td class="isbn"><xsl:value-of select="isbn"/></td>
             <td class="publisher"><xsl:value-of select="publisher"/></td>
             <td class="edition"><xsl:value-of select="edition"/></td>
             <td class="price"><xsl:value-of select="price"/></td>
           </tr>
         </xsl:for-each>
       </table>
     </body>
   </html>
 </xsl:template>
</xsl:stylesheet>
```

Output :



Books : Table

ISBN	Author	Title	Publisher	Date	OnLoan
0140434941	Dickens	David Copperfield	Penguin Classics	1850	☑
0141439742	Dickens	Oliver Twist	Penguin Classics	1845	☑
0375757422	Austen	Emma	Modern Library	1815	☑
0486272788	Shakespeare	Hamlet	Dover	1601	☑
0679601686	Austen	Pride and Prejudice	Modern Library	1815	☑
0684800713	Hemingway	Sun Also Rises	Scribner	1926	□
0743477103	Shakespeare	Macbeth	Dover	1603	☑
0684837889	Hemingway	Farewell to Arms	Scribner	1928	□
					▨

WEEK 9:

Create a simple visual bean with an area filled with a colour. The shape of the area depends on the property shape. If it is set to true then the shape of the area is Square and it is Circle, is false. The colour of the area should be changed dynamically for every mouse click.

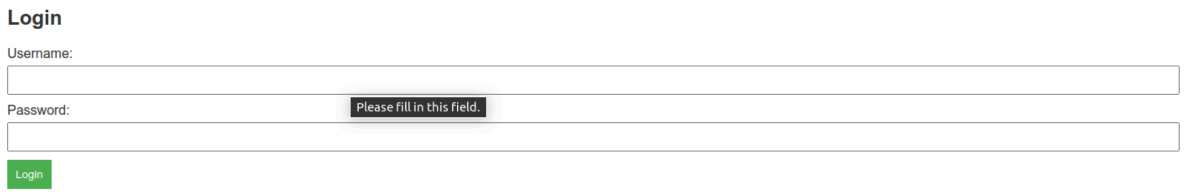
Code :

```
import java.awt.*;
import java.awt.event.*;
import java.io.Serializable;

public class C2 extends Canvas implements Serializable {
   transient private Color colour; // not persistent
   private boolean rectangular; // is persistent

   public C2() {
       addMouseListener(new MouseAdapter() {
           public void mousePressed(MouseEvent me) {
               change();
           }
       });
       rectangular = false;
       setSize(200, 100);
       change();
   }

   public boolean getRectangular() {
       return rectangular;
   }

   public void setRectangular(boolean flag) {
       this.rectangular = flag;
       repaint();
   }

   public void change() {
       colour = randomColor();
       repaint();
   }

   private Color randomColor() {
       int r = (int) (255 * Math.random());
       int g = (int) (255 * Math.random());
       int b = (int) (255 * Math.random());
       return new Colour(r, g, b);
   }

   public void paint(Graphics g) {
       Dimension d = getSize();
```

```
        int h = d.height;
        int w = d.width;
        g.setColor(colour);
        if (rectangular) {
            g.fillRect(0, 0, w - 1, h - 1);
        } else {
            g.fillOval(0, 0, w - 1, h - 1);
        }
    }
}
```

```
    <form id="loginForm">
        <label for="loginUsername">Username:</label>
        <input type="text" id="loginUsername" required>
        <br>
        <label for="loginPassword">Password:</label>
        <input type="password" id="loginPassword" required>
        <br>
        <button type="button" onclick="loginUser()">Login</button>
    </form>
</div>

<!-- User Profile Page -->
<div id="userProfile" style="display: none;">
    <h2>User Profile</h2>
    <p id="loggedInUser"></p>
    <button type="button" onclick="logoutUser()">Logout</button>
    <button type="button" onclick="showPaymentPage()">Proceed to Payment</button>
</div>

<!-- Payment by Credit Card Page -->
<div id="paymentPage" style="display: none;">
    <h2>Payment by Credit Card</h2>
    <form id="paymentForm">
        <label for="creditCardNumber">Credit Card Number:</label>
        <input type="text" id="creditCardNumber" required>
        <br>
        <label for="expiryDate">Expiry Date:</label>
        <input type="text" id="expiryDate" placeholder="MM/YY" required>
        <br>
        <button type="button" onclick="processPayment()">Process Payment</button>
    </form>
</div>

<script>
    function registerUser() {
        // Perform registration logic
        alert('User registered successfully!');
        showLoginPage();
    }

    function loginUser() {
        // Perform login logic
        let username = document.getElementById('loginUsername').value;
        document.getElementById('loggedInUser').innerText = `Logged in as: ${username}`;
        showUserProfile();
    }

    function logoutUser() {
        // Perform logout logic
        document.getElementById('loggedInUser').innerText = '';
        showLoginPage();
    }

    function processPayment() {
        // Perform payment logic
        alert('Payment processed successfully!');
    }

    function showLoginPage() {
        document.getElementById('registrationPage').style.display = 'none';
        document.getElementById('loginPage').style.display = 'block';
        document.getElementById('userProfile').style.display = 'none';
        document.getElementById('paymentPage').style.display = 'none';
    }

    function showUserProfile() {
        document.getElementById('registrationPage').style.display = 'none';
        document.getElementById('loginPage').style.display = 'none';
        document.getElementById('userProfile').style.display = 'block';
        document.getElementById('paymentPage').style.display = 'none';
    }

    function showPaymentPage() {
        document.getElementById('registrationPage').style.display = 'none';
        document.getElementById('loginPage').style.display = 'none';
        document.getElementById('userProfile').style.display = 'none';
        document.getElementById('paymentPage').style.display = 'block';
    }

    // Initial display
    showLoginPage();
 </script>

 </body>
 </html>
```

Output :

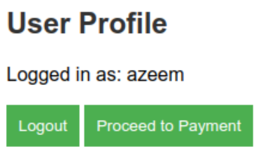


Login

Username:

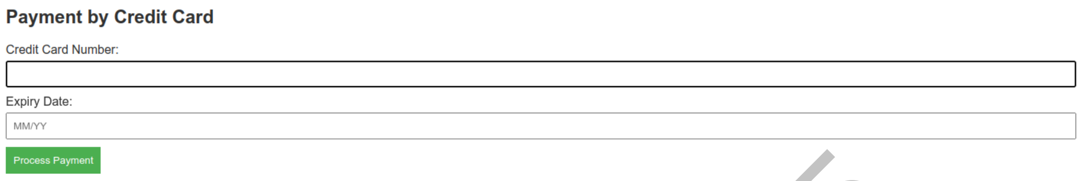
Password:

Please fill in this field.



User Profile

Logged in as: azeem

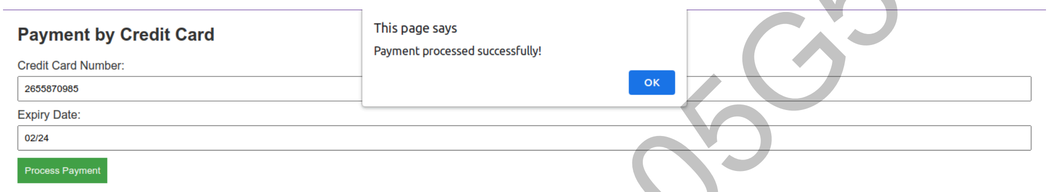


Payment by Credit Card

Credit Card Number:

Expiry Date:

MM/YY



Payment by Credit Card

Credit Card Number: 2655870985

Expiry Date: 02/24

This page says

Payment processed successfully!

Week-10

Aim:

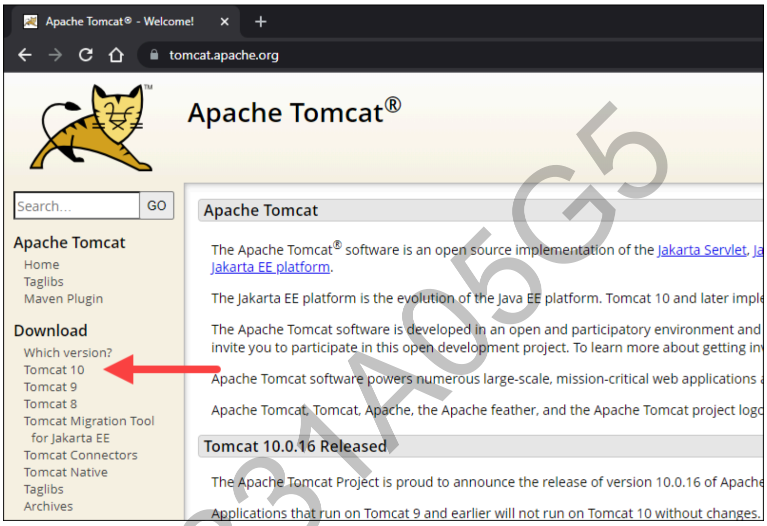
Install TOMCAT web server. While Installation assigns port number 8080. Make sure that all these ports are available i.e, no other process is using this port.

Description:

Apache TOMCAT is a free open source implementation of the Jakarta Servlet, Jakarta Expression Language, and web socket technologies. It provides a pure ‘JAVA’ HTTP web server environment in which java code runs.

Procedure:

- Browse to the official tomcat website. Locate the download section and click on the latest tomcat version.



Apache Tomcat

The Apache Tomcat® software is an open source implementation of the Jakarta Servlet, Ja
Jakarta EE platform.

The Jakarta EE platform is the evolution of the Java EE platform. Tomcat 10 and later imple

The Apache Tomcat software is developed in an open and participatory environment and
invite you to participate in this open development project. To learn more about getting in

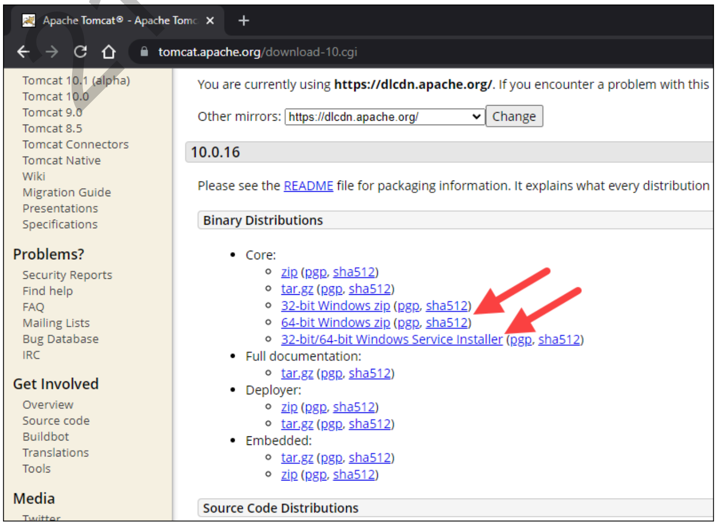
Apache Tomcat software powers numerous large-scale, mission-critical web applications

Apache Tomcat, Tomcat, Apache, the Apache feather, and the Apache Tomcat project logo

Tomcat 10.0.16 Released

The Apache Tomcat Project is proud to announce the release of version 10.0.16 of Apache

Applications that run on Tomcat 9 and earlier will not run on Tomcat 10 without changes.



You are currently using **https://dlcdn.apache.org/**. If you encounter a problem with this

Other mirrors: https://dlcdn.apache.org/

10.0.16

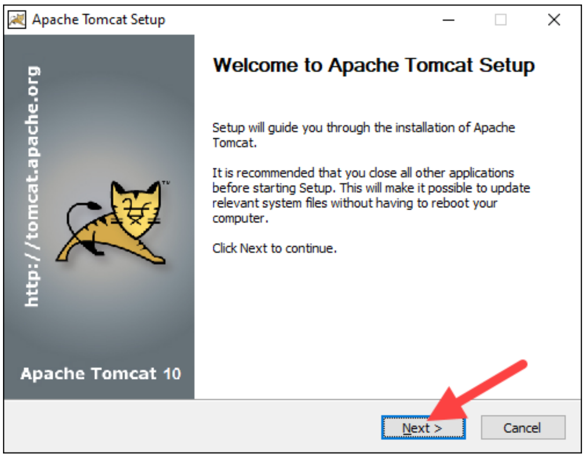
Please see the README file for packaging information. It explains what every distribution

Binary Distributions

- Core:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
  - 32-bit Windows zip (pgp, sha512)
  - 64-bit Windows zip (pgp, sha512)
  - 32-bit/64-bit Windows Service Installer (pgp, sha512)
- Full documentation:
  - tar.gz (pgp, sha512)
- Deployer:
  - zip (pgp, sha512)
  - tar.gz (pgp, sha512)
- Embedded:
  - tar.gz (pgp, sha512)
  - zip (pgp, sha512)

Source Code Distributions

- On the downloads page locate the binary distributions area.
- Install the TOMCAT using Windows Service Installer.
- Open the downloaded Windows Service Installer file to start the installation process.
- In the TOMCAT setup welcome screen. Click on the ‘Next’.

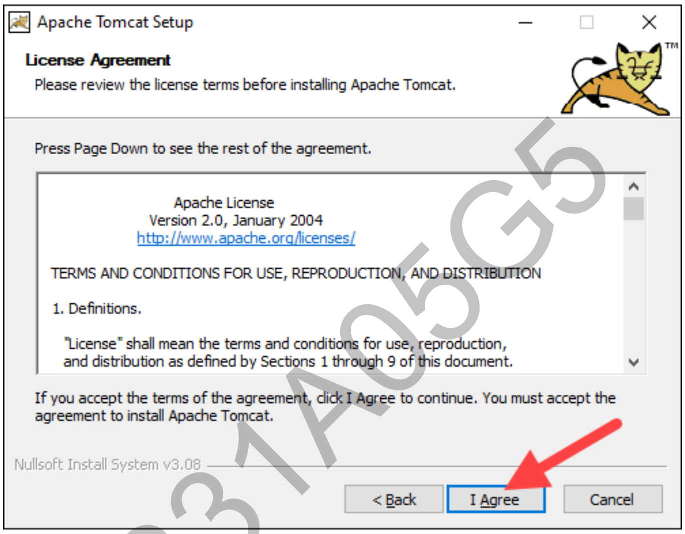

Apache Tomcat Setup

Welcome to Apache Tomcat Setup

Setup will guide you through the installation of Apache Tomcat.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.


Apache Tomcat Setup

License Agreement

Please review the license terms before installing Apache Tomcat.

Press Page Down to see the rest of the agreement.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

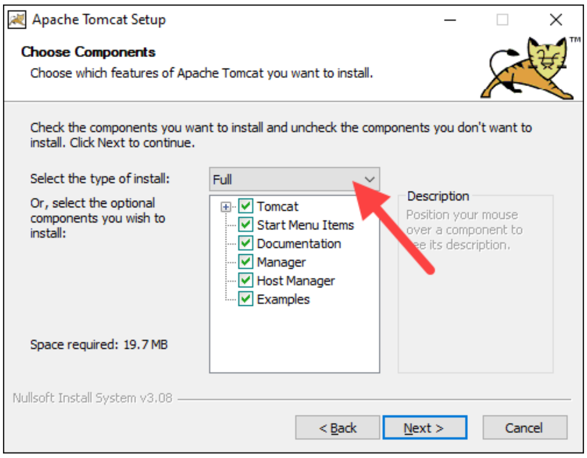
1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Apache Tomcat.

Nullsoft Install System v3.08

- Read the license agreement and click on 'I Agree'.
- In the Tomcat component selection screen, choose Full in the dropdown menu to ensure the wizard installs the Tomcat Host Manager and Servlet and JSP examples web applications. Alternatively, keep the default Normal installation type and click Next.
- The next step configures the Tomcat server. For instance, enter the Administrator login credentials or choose a different connection port. When finished, click Next to proceed to the next step.


Apache Tomcat Setup

Choose Components

Choose which features of Apache Tomcat you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Full

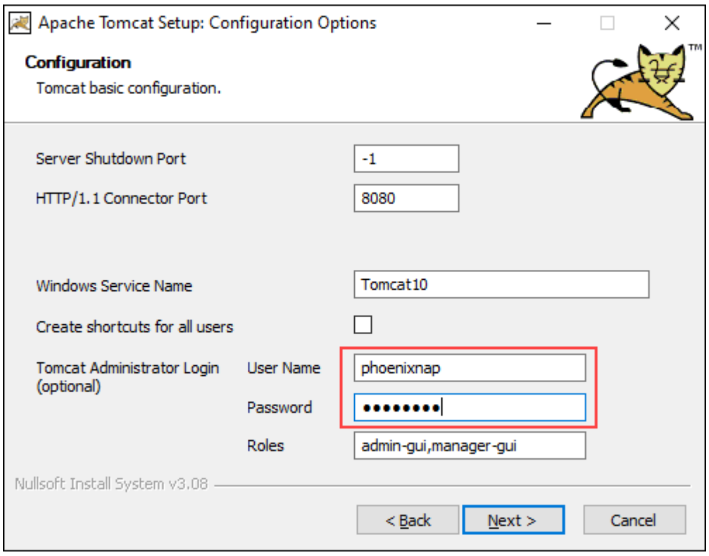
Or, select the optional components you wish to install:

- ☑ Tomcat
- ☑ Start Menu Items
- ☑ Documentation
- ☑ Manager
- ☑ Host Manager
- ☑ Examples

Description
Position your mouse over a component to see its description.

Space required: 19.7 MB

Nullsoft Install System v3.08


Apache Tomcat Setup: Configuration Options

Configuration

Tomcat basic configuration.

Server Shutdown Port: -1

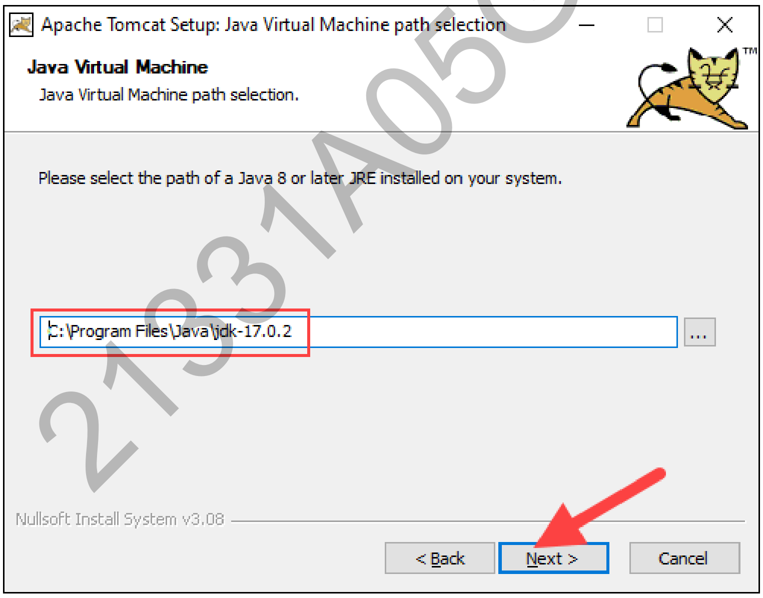
HTTP/1.1 Connector Port: 8080

Windows Service Name: Tomcat10

Create shortcuts for all users: □

Tomcat Administrator Login (optional)
User Name: phoenixnap
Password: ••••••••
Roles: admin-gui,manager-gui

- The next step requires you to enter the full path to the JRE directory on your system. The wizard auto-completes this if you have previously set up the Java environment variables. Click Next to proceed to the next step.


Apache Tomcat Setup: Java Virtual Machine path selection

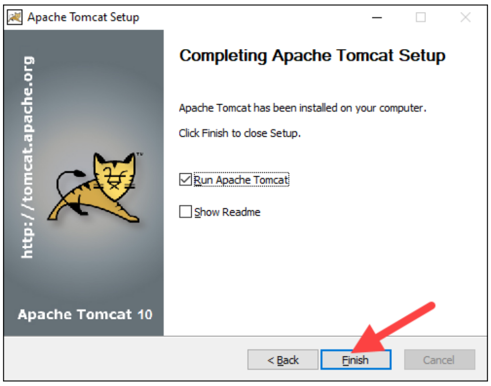
Java Virtual Machine

Java Virtual Machine path selection.

Please select the path of a Java 8 or later JRE installed on your system.

C:\Program Files\Java\jdk-17.0.2

- Choose the Tomcat server install location or keep the default one and click Install.


Apache Tomcat Setup

Completing Apache Tomcat Setup

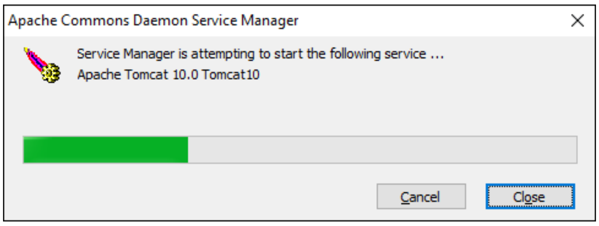
Apache Tomcat has been installed on your computer.

Click Finish to close Setup.

☑ Run Apache Tomcat

□ Show Readme

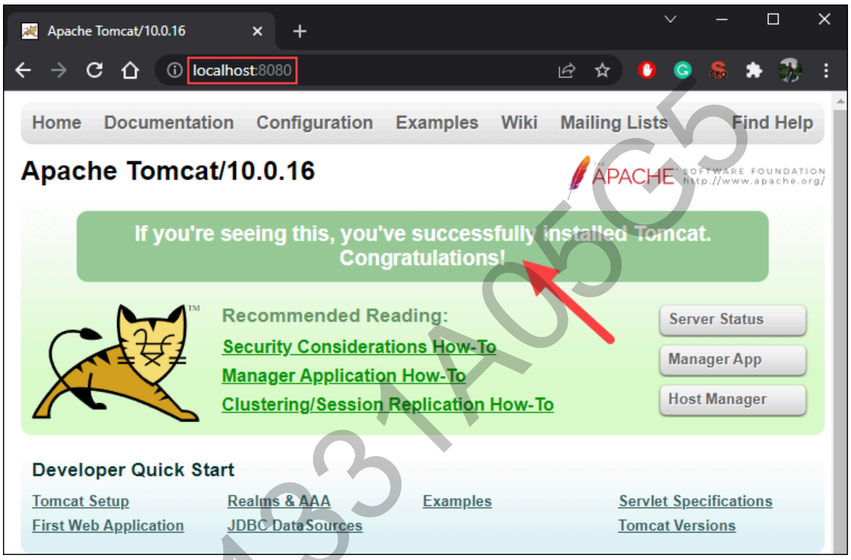
- A popup window appears that starts the Tomcat service. After the process completes, the window closes automatically. The Apache Tomcat web server is now successfully installed.



Apache Commons Daemon Service Manager

Service Manager is attempting to start the following service ...
Apache Tomcat 10.0 Tomcat10

Output:



Apache Tomcat/10.0.16

If you're seeing this, you've successfully installed Tomcat. Congratulations!

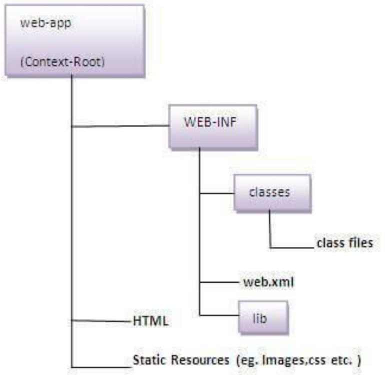
Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To

Developer Quick Start

Tomcat Setup
First Web Application
Realms & AAA
JDBC DataSources
Examples
Servlet Specifications
Tomcat Versions

Aim:

Access the above developed static web pages for books website, using these servers by Putting the web pages developed in week-1 andweek-2 in the document root. Access the pages by using the URLs : http://localhost:8080/rama/books.html



Description:

Procedure:

- First install the tomcat into the system.
- Then make a sub directory(eg., books) in the \tomcat\webapps.
- Under books create WEB-INF directory and also place week1 programs in this books directory only.
- After this start tomcat by giving the following command at the instll_dir>tomcat>bin Catalina.bat run
- At the I.E(web browser) give the url as http://localhost:8080/ books /main.html
- Port no 8080 is assigned for the tomcat.

Aim:

Write a servlet program that displays “MVGR AUTONOMOUS” message on webpage using Generic Servlet class.

Description:

- GenericServlet class implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all the methods of these interfaces except the service method.
- GenericServlet can implement any type of request as it is a protocol independent.

Program:

HTML:

```
<html>
   <head>
       <title>
           Generic Servlet
       </title>
   </head>
   <body bgcolor = "yellow" text = "red">
       <center>
           <h1>
               <br><br><br><br><br><br><br><br>Hello World!<br>
               <a href = "mvgr">Generic Servlet</a>
           </h1>
       </center>
   </body>
</html>
```

XML:

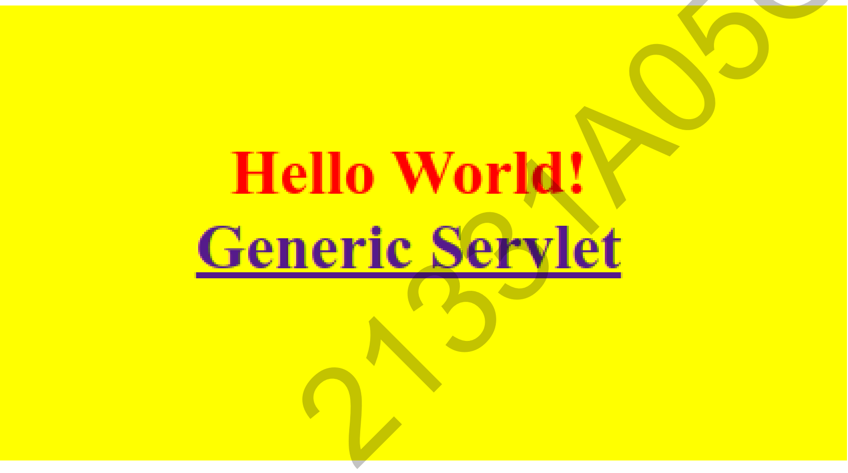
```
<web-app>
   <servlet>
       <servlet-name>GenSer</servlet-name>
       <servlet-class>Genser</servlet-class>
   </servlet>
   <servlet-mapping>
```

```
       <servlet-name>GenSer</servlet-name>
       <url-pattern>/mvgr</url-pattern>
   </servlet-mapping>
</web-app>
```

JAVA:

```
import java.io.*;
import jakarta.servlet.*;
public class Genser extends GenericServlet {
   public void service(ServletRequest req, ServletResponse res) throws
IOException, ServletException {
       res.setContentType("text/html");
       PrintWriter out = res.getWriter();
       out.print("<html><body>");
       out.print("<b>MVGR (AUTONOMOUS)</b>");
       out.print("</body></html>");
   }
}
```

Output:



Hello World!

Generic Servlet

Aim:

Implementation of a servlet to add two numbers.

Description:

- GenericServlet class implements Servlet, ServletConfig and Serializable interfaces. It provides the implementation of all the methods of these interfaces except the service method.
- GenericServlet can implement any type of request as it is a protocol independent.
- The HttpServlet class extends the GenericServlet class and implements Serializable interface. It provides http specific methods such as doGet, doPost, doHead, doTrace etc.

Program:

HTML:

```
<html>
   <head>
       <title>
           Addition in Server
       </title>
   </head>
   <body bgcolor = "green" text = "black">
       <font face = "timesnewroman" size = "4px">
           <center>
               <form action = "./add" method = "get">
                   <br><br><br><br><br><br><br><br>Number1:<input
type="text" name="n1"><br><br>
                   Number2:<input type="text" name="n2"><br><br>
                   <input type="submit" value="Calculate Sum">
               </form>
           </center>
       </font>
   </body>
</html>
```

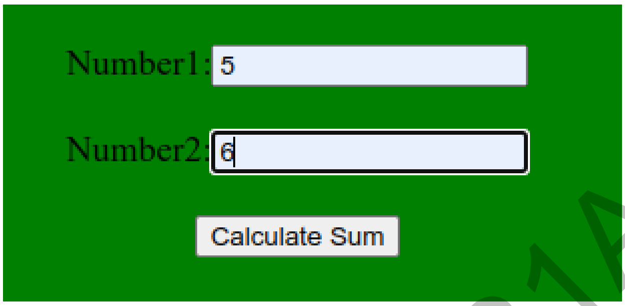
XML:

```
<web-app>
   <servlet>
       <servlet-name>HttSer</servlet-name>
       <servlet-class>Addition</servlet-class>
   </servlet>
   <servlet-mapping>
       <servlet-name>HttSer</servlet-name>
       <url-pattern>/add</url-pattern>
   </servlet-mapping>
</web-app>
```

JAVA:

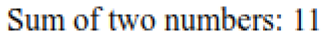
```
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
public class Addition extends HttpServlet {
   protected void doGet(HttpServletRequest req, HttpServletResponse
res) throws ServletException, IOException {
       PrintWriter pw = res.getWriter();
       res.setContentType("text/html");
       String n1 = req.getParameter("n1");
       String n2 = req.getParameter("n2");
       int result = Integer.parseInt(n1) + Integer.parseInt(n2);
       pw.println("Sum of two numbers: " +result);
       pw.close();
   }
}
```

Output:



Number1: 5

Number2: 6



Sum of two numbers: 11

Week-11

Aim:

To read Initilization parameters

Description:

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

There are various advantages of JSP:

- Extension to Servlet
- Easy to maintain
- No need to recompile and redeploy
- Less code than servlet

Program:

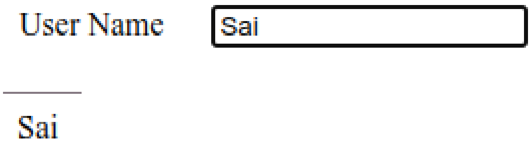
HTML:

```
<html>
      <body>
            <form action=”welcome.jsp”>
                  User Name:
                  &nbsp;&nbsp;
                  <input type=”text” name=”uname”>
            </form>
      </body>
</html>
```

Welcome.jsp:

```
<html>
      <body>
            <%
                  import java.servlet.*;
                  public class initparam(ServletRequest req,
ServletResponse res)
                  {
                        String n=req.getParameter(‶uname”);
                        out.println(n);
                  }
            %>
      </body>
</html>
```

Output:



User Name Sai

Sai

Aim:

To read Context parameters

Description:

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

There are various advantages of JSP:

- Extension to Servlet
- Easy to maintain
- No need to recompile and redeploy
- Less code than servlet

Program:

JSP:

```
<%
      import javax.servlet.*;
      import java.io.*;
      ServletContext c=getServletContext();
      String p=c.getParameter(“mycontextparam”);
      out.print(p);
%>
```

XML:

```
<servlet>
      <servlet-name>Context</servlet-name>
      <servlet-class>com.example.ServletClass</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>Context</servlet-name>
      <url-pattern>/servlet/path.jsp</url-pattern>
</servlet-mapping>
<context-param>
      <param-name>mycontextparam</param-name>
      <param-value>Saicharan</param-value>
</context-param>
```

Output:

Aim:

Implement a JSP program to implement sessions using HTTP Sessions interface.

Description:

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

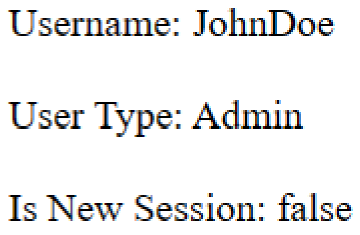
There are various advantages of JSP:

- Extension to Servlet
- Easy to maintain
- No need to recompile and redeploy
- Less code than servlet

Program:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
      <head>
            <meta charset="UTF-8">
      </head>
      <body>
            <%
                // Obtaining the session object
                HttpSession session = request.getSession(true);
                // Checking if session is new or not
                boolean isNewSession = session.isNew();
                // Setting session attributes
                session.setAttribute("username", "JohnDoe");
                session.setAttribute("userType", "Admin");
                // Getting session attributes
                String username = (String)
            session.getAttribute("username");
                String userType = (String)
            session.getAttribute("userType");
            %>
            <h2>Session Example</h2>
            <p>Username: <%= username %></p>
            <p>User Type: <%= userType %></p>
            <p>Is New Session: <%= isNewSession %></p>

      </body>
</html>
```

Output:



```
Username: JohnDoe

User Type: Admin

Is New Session: false
```

Aim:

Implement a Java Servlet Program to implement sessions using Cookies.

Description:

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

There are various advantages of JSP:

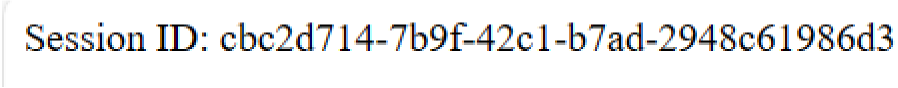
- Extension to Servlet
- Easy to maintain
- No need to recompile and redeploy
- Less code than servlet

Program:

```
<%@ page language="java" %>
<%@ page import="java.io.*,java.util.*" %>
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<html>
      <head><title>Session Management using Cookies</title></head>
      <body>
            <%
               Cookie[] cookies = request.getCookies();
               String sessionId = null;

               // Checking if the session cookie already exists
               if (cookies != null) {
                   for (int i = 0; i < cookies.length; i++) {
                       if (cookies[i].getName().equals("sessionId")) {
                           sessionId = cookies[i].getValue();
                           break;
                       }
                   }
               }

               // If session cookie doesn't exist, create a new session
            ID
               if (sessionId == null) {
                   sessionId = UUID.randomUUID().toString();
                   Cookie sessionCookie = new Cookie("sessionId",
            sessionId);
```

```
                   sessionCookie.setMaxAge(60 * 60 * 24); // Session
            cookie valid for 1 day
                   response.addCookie(sessionCookie);
               }
            %>

            <h1>Session ID: <%=sessionId%></h1>
      </body>
</html>
```

Output:



```
Session ID: cbc2d714-7b9f-42c1-b7ad-2948c61986d3
```

Week-12

Aim:

Write a JSP program to generate multiplication of a given number

Description:

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

There are various advantages of JSP:

- Extension to Servlet
- Easy to maintain
- No need to recompile and redeploy
- Less code than servlet

Program:

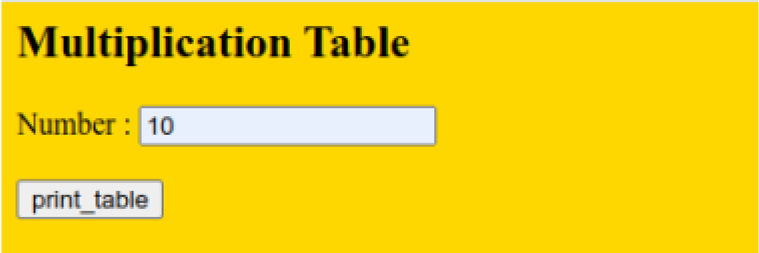
HTML:

```
<html>
      <head>
            <title> Print Multiplication Table </title>
      </head>
            <body bgcolor="gold">
                  <h2>Multiplication Table </h2>
                  <form action="table.jsp" method="post">
                        Number : <input type="text" name="num"> <br>
                        <br>
                        <input type="submit" value="print_table">
                  </form>
            </body>
</html>
```

JSP:

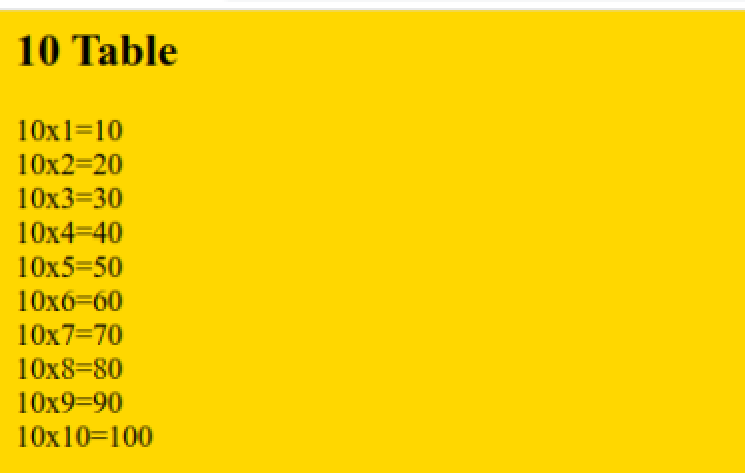
```
<body bgcolor="gold">
      <%
            int n = Integer.parseInt(request.getParameter("num"));
            out.println("<h2>" + n + " Table" + "<br>" + "</h2>");
            for(int i=1;i<=10;i++) {
                  out.println(n + "x" + i + "=" + (n*i) + "<br>");
            }
      %>
</body>
```

Output:



Multiplication Table

Number : 10



```
10 Table

10x1=10
10x2=20
10x3=30
10x4=40
10x5=50
10x6=60
10x7=70
10x8=80
10x9=90
10x10=100
```

Aim:

Write a JSP program to check if a number is an armstrong number or not

Description:

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

There are various advantages of JSP:

- Extension to Servlet
- Easy to maintain
- No need to recompile and redeploy
- Less code than servlet

Program:

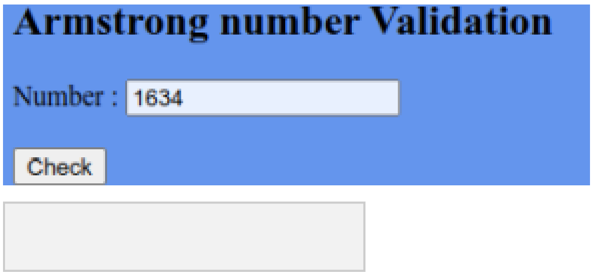
HTML:

```
<html>
      <head>
      <title> Armstrong </title>
      </head>
      <body bgcolor="CornflowerBlue">
            <h2>Armstrong number Validation </h2>
            <form action="armstrong.jsp" method="post">
                  Number : <input type="text" name="num"> <br> <br>
                  <input type="submit" value="Check">
            </form>
      </body>
</html>
```

JSP:

```
<%@ page import="java.lang.Math.*" %>
<%
      String a = request.getParameter("num");
      int digits = a.length();
      int n =Integer.parseInt(a);
      int temp=n;
      int rem,sum=0;
      while(n>0) {
      rem = n%10;
      sum=sum+(int)(Math.pow(rem,digits));
      n=n/10;
      }
      if (temp == sum) {
      out.println(temp + " is Armstrong number");
      } else {
      out.println(temp + " is not Armstrong number");
      }
%>
```

Output:



Armstrong number Validation

Number : 1634

Aim:

Write a JSP Program to find the salary of an employee whose basic salary has to be taken as an input from user. Use the following rules to compute the gross salary

- DA-DMS allowances= 90% of basic
- HRA=10% of basic
- Gross salary= basic +DA+HRA

Description:

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

There are various advantages of JSP:

- Extension to Servlet
- Easy to maintain
- No need to recompile and redeploy
- Less code than servlet

Program:

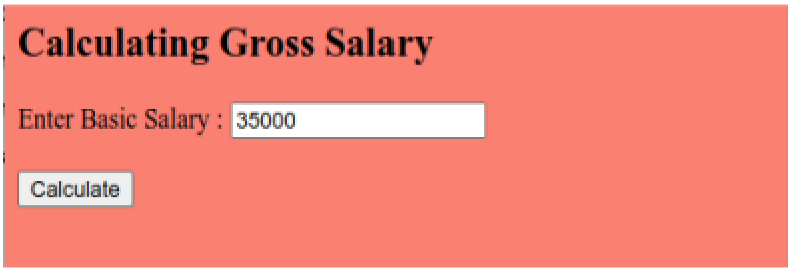
HTML:

```
<html>
      <head>
            <title> Gross Salary </title>
      </head>
      <body bgcolor="Salmon">
            <h2> Calculating Gross Salary </h2>
            <form action="gross.jsp" method="post">
                  Enter Basic Salary : <input type="number" name="sal">
                  <br> <br>
                  <input type="submit" value="Calculate">
            </form>
      </body>
</html>
```

JSP:

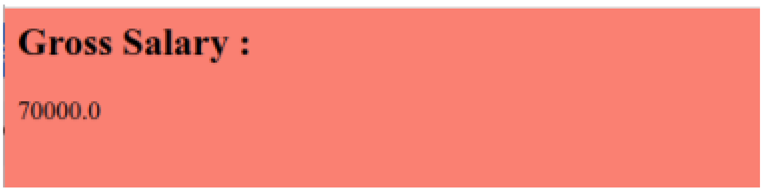
```
<%
      int basic=Integer.parseInt(request.getParameter("sal"));
      double DA = 0.90 * basic;
      double HRA = 0.1 * basic;
      double gross = basic +DA + HRA;
      out.println("<h2>" + "Gross Salary : " + "</h2>" + gross);
%>
```

Output:



Calculating Gross Salary

Enter Basic Salary : 35000



Gross Salary :

70000.0

Week-13

Aim:

Write a JSP which does the following job:

Insert the details of the 3 or 4 users who register with the web site (week 12) by using registration form. Authenticate the user when he submits the login form using the user name and password from the database.

Description:

Utilizing Java Database Connectivity (JDBC) with JSP enables seamless interaction between web applications and databases. By establishing a connection through JDBC, JSP pages can execute SQL queries, retrieve data, and perform database operations dynamically. This integration facilitates dynamic content generation and data-driven web applications. JDBC provides a robust framework for managing database connectivity within JSP, empowering developers to create efficient and scalable web solutions. Leveraging JDBC within JSP enhances the interactivity and functionality of web applications, enabling seamless integration with backend databases.

Program:

Creation of table:

```
<%@ page import="java.sql.Connection" %>

<%@ page import="java.sql.DriverManager" %>

<%@ page import="java.sql.Statement" %>

<%@ page import="java.sql.SQLException" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

      <head>

            <title>Create a table using JSP</title>

      </head>

      <body>

            <h1>Create a table using JSP</h1>

            <%

                  Connection con = null;

                  try {

                        Class.forName("com.mysql.jdbc.Driver");

                        con                                      =
                        DriverManager.getConnection("jdbc:mysql://localh
                        ost:3306/college", "root", "password");

                        Statement st = con.createStatement();
```

```
                String query = "CREATE TABLE Persons (PersonID INT, LastName VARCHAR(255), FirstName VARCHAR(255), City VARCHAR(255))";

                st.executeUpdate(query);

                out.println("Table Persons created successfully");

                st.close();

                con.close();

            }

            catch (ClassNotFoundException | SQLException e) {

                out.println("An error occurred: " + e.getMessage());

            }

            finally {

                try {

                    if (con != null) {

                        con.close();

                }

                }

                catch (SQLException e) {

                    out.println("An error occurred while closing the connection: " + e.getMessage());

                }

            }

        %>

    </body>

</html>
```

Creation of table:

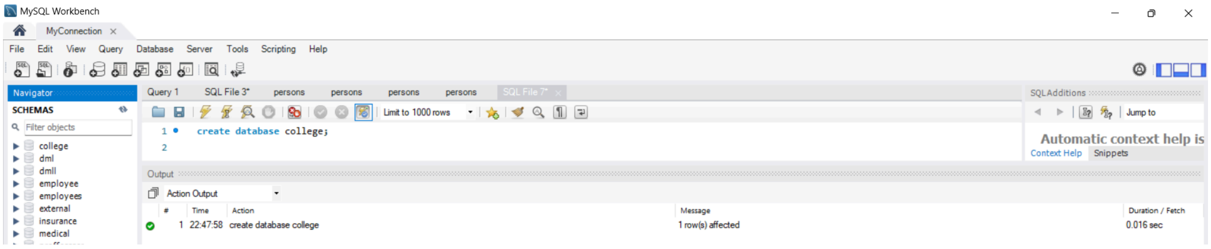
```
<%@ page import="java.sql.Connection" %>

<%@ page import="java.sql.DriverManager" %>

<%@ page import="java.sql.PreparedStatement" %>

<%@ page import="java.sql.SQLException" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>
```

```
<html>
    <head>
        <title>Insert Data using JSP</title>
    </head>
    <body>
        <h1>Insert Data using JSP</h1>
        <%
            Connection con = null;
            PreparedStatement pst = null;
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                con = DriverManager.getConnection("jdbc:mysql://localhost:3306/college", "root", "password");
                // Insert data into the Persons table
                String insertQuery = "INSERT INTO Persons (PersonID, LastName, FirstName, City) VALUES (?, ?, ?, ?)";
                pst = con.prepareStatement(insertQuery);
                // Set values for the placeholders
                pst.setInt(1, 1); // Replace with actual values
                pst.setString(2, "Doe");
                pst.setString(3, "John");
                pst.setString(4, "New York");
                //Execute the insert query
                int rowsAffected = pst.executeUpdate();


                if (rowsAffected > 0) {
                out.println("Data inserted successfully");
                } else {
                        out.println("No rows affected. Data insertion
                        failed.");

                  }

                  }

                  catch (ClassNotFoundException | SQLException e) {

                        out.println("An error occurred: " +
                  e.getMessage());

                  }

                  finally {

                        try {

                              if (pst != null) {

                              pst.close();

                        }

                        if (con != null) {

                              con.close();

                        }

                  }

                  catch (SQLException e) {

                        out.println("An error occurred while closing the
                  connection: " + e.getMessage());

                  }

                  }

            %>

      </body>

</html>
```

Retrieving data from table:

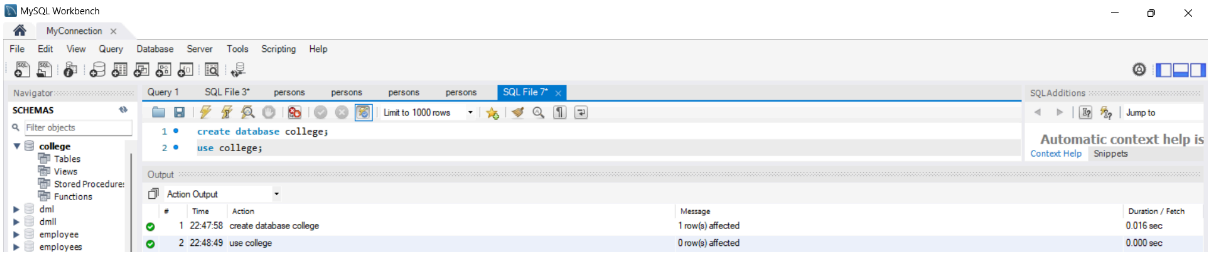
```
<%@ page import="java.sql.Connection" %>

<%@ page import="java.sql.DriverManager" %>

<%@ page import="java.sql.ResultSet" %>

<%@ page import="java.sql.SQLException" %>

<%@ page import="java.sql.Statement" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Retrieve Data using JSP</title>

</head>

<body>

<h1>Retrieve Data using JSP</h1>

<%

Connection con = null;

Statement st = null;

ResultSet rs = null;

try {

Class.forName("com.mysql.jdbc.Driver");

con                                                                   =
DriverManager.getConnection("jdbc:mysql://localhost:3306/college",
"root", "password");

st = con.createStatement();

String query = "SELECT * FROM Persons";

rs = st.executeQuery(query);

out.println("<table border='1'>");

out.println("<tr><th>PersonID</th><th>LastName</th><th>FirstName</th><t
h>City</th></tr>");

while (rs.next()) {

out.println("<tr>");

out.println("<td>" + rs.getInt("PersonID") + "</td>");

out.println("<td>" + rs.getString("LastName") + "</td>");

out.println("<td>" + rs.getString("FirstName") + "</td>");
out.println("<td>" + rs.getString("City") + "</td>");
out.println("</tr>");
}
out.println("</table>");
} catch (ClassNotFoundException | SQLException e) {
out.println("An error occurred: " + e.getMessage());
} finally {
try {
if (rs != null) {
rs.close();
}
if (st != null) {
st.close();
}
if (con != null) {
con.close();
}
} catch (SQLException e) {
out.println("An error occurred while closing the connection: " + e.getMessage());
}
}
%>
</body>
</html>
```

Output:



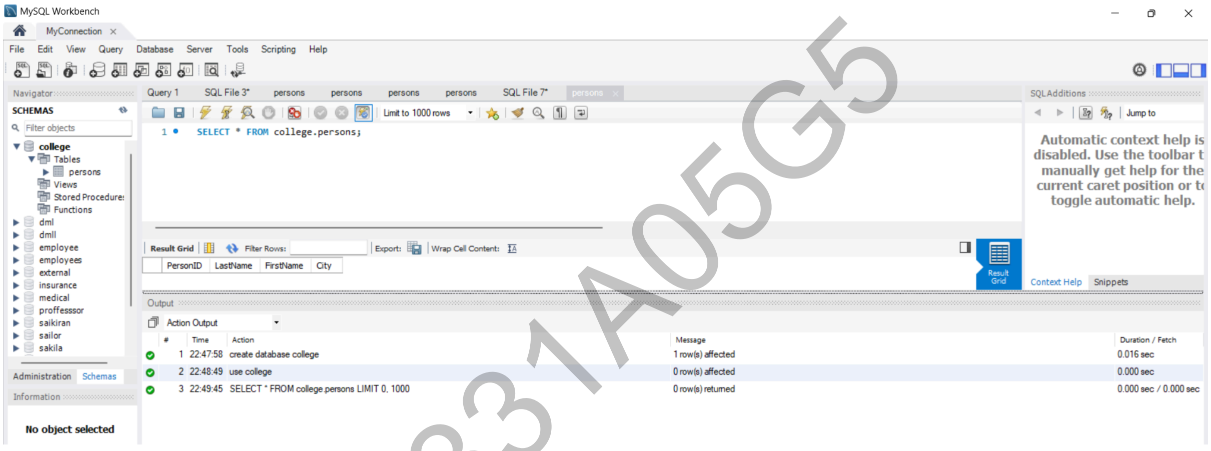
```
create database college;
```

```
#  Time      Action                   Message            Duration / Fetch
1  22:47:58  create database college  1 row(s) affected  0.016 sec
```



```
create database college;
use college;
```

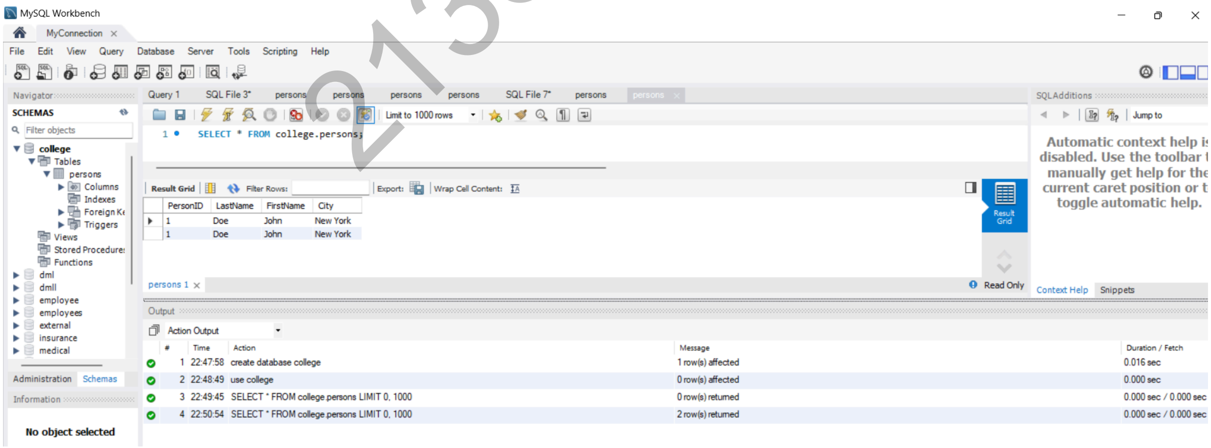
```
#  Time      Action                   Message            Duration / Fetch
1  22:47:58  create database college  1 row(s) affected  0.016 sec
2  22:48:49  use college              0 row(s) affected  0.000 sec
```



```
SELECT * FROM college.persons;
```

PersonID	LastName	FirstName	City

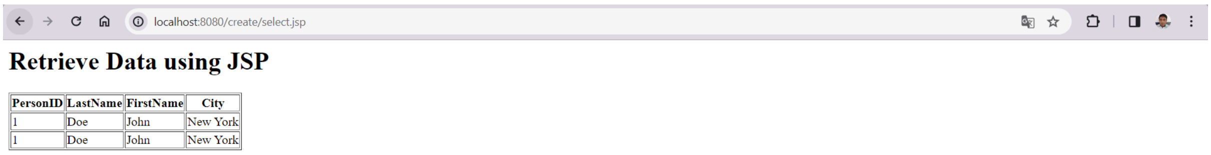
```
#  Time      Action                                          Message            Duration / Fetch
1  22:47:58  create database college                         1 row(s) affected  0.016 sec
2  22:48:49  use college                                     0 row(s) affected  0.000 sec
3  22:49:45  SELECT * FROM college.persons LIMIT 0, 1000     0 row(s) returned  0.000 sec / 0.000 sec
```



```
SELECT * FROM college.persons;
```

PersonID	LastName	FirstName	City
1	Doe	John	New York
1	Doe	John	New York

```
#  Time      Action                                          Message            Duration / Fetch
1  22:47:58  create database college                         1 row(s) affected  0.016 sec
2  22:48:49  use college                                     0 row(s) affected  0.000 sec
3  22:49:45  SELECT * FROM college.persons LIMIT 0, 1000     0 row(s) returned  0.000 sec / 0.000 sec
4  22:50:54  SELECT * FROM college.persons LIMIT 0, 1000     2 row(s) returned  0.000 sec / 0.000 sec
```



Retrieve Data using JSP

PersonID	LastName	FirstName	City
1	Doe	John	New York
1	Doe	John	New York