1. Diffwrence between json and xml

|  |  |  |
| --- | --- | --- |
|  | JSON | XML |
| Stands for | *JSON* means JavaScript Object Notation. | *XML* means Extensible Markup Language. |
| History | Douglas Crockford and Chip Morningstar released JSON in 2001. | The XML Working Group released XML in 1998. |
| Format | JSON uses a maplike structure with key-value pairs. | XML stores data in a tree structure with namespaces for different data categories. |
| Syntax | The syntax of JSON is more compact and easier to read and write. | The syntax of XML substitutes some characters for entity references, making it more verbose. |
| Parsing | You can parse JSON with a standard JavaScript function. | You need to parse XML with an XML parser. |
| Schema documentation | JSON is simple and more flexible. | XML is complex and less flexible. |
| Data types | JSON supports numbers, objects, strings, and Boolean arrays. | XML supports all JSON data types and additional types like Boolean, dates, images, and namespaces. |
| Ease of use | JSON has smaller file sizes and faster data transmission. | XML tag structure is more complex to write and read and results in bulky files. |
| Security | JSON is safer than XML. | You should turn off DTD when working with XML to mitigate potential security risks. |

1. 3 xml and json files for department,year,student

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

<department>

<department>

<deptid>1</deptid>

<deptname>IT</deptname>

</department>

<department>

<deptid>2</deptid>

<deptname>CSE</deptname>

</department>

<department>

<deptid>3</deptid>

<deptname>ECE</deptname>

</department>

</department>

YEAR.XML

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

<year>

<year>

<id>1</id>

<name>firstyear</name>

</year>

<year>

<id>2</id>

<name>secondyear</name>

</year>

<year>

<id>3</id>

<name>thirdyear</name>

</year>

<year>

<id>4</id>

<name>fourthyear</name>

</year>

</year>

STUDENT.XML

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

<student>

<student>

<id>1</id>

<name>Preethika</name>

<dept>IT</dept>

</student>

<student>

<id>2</id>

<name>Prathyusha</name>

<dept>CSE</dept>

</student>

<student>

<id>3</id>

<name>Vennela</name>

<dept>ECE</dept>

</student>

</student> DEPARTMENT.JSON

{

"departments": [

{

"id": 1,

"name": "Computer Science",

"location": "Building A"

},

{

"id": 2,

"name": "Electrical Engineering",

"location": "Building B"

}

]

}

Year JSON:

{

"years": [

{

"id": 1,

"name": "Freshman"

},

{

"id": 2,

"name": "Sophomore"

}

]

}

STUDENT.JSON

{

"students": [

{

"id": 1,

"name": "John Doe",

"department\_id": 1,

"year\_id": 3

},

{

"id": 2,

"name": "Jane Smith",

"department\_id": 2,

"year\_id": 2

}

/\* Add more students here if needed \*/

]

}

3.a file with department as root,year as subroot,ad student as an element dept<?xml version="1.0" encoding="UTF-8"?>

<department>

<year>

<id>1</id>

<name>firstyear</name>

<student>

<student>

<id>1</id>

<name>Preethika</name>

<dept>IT</dept>

</student>

<student>

<id>2</id>

<name>Vennela</name>

<dept>ECE</dept>

</student>

</student>

</year>

<year>

<id>2</id>

<name>secondyear</name>

<student>

<student>

<id>1</id>

<name>Preethika</name>

<dept>IT</dept>

</student>

<student>

<id>2</id>

<name>Vennela</name>

<dept>ECE</dept>

</student>

</student>

</year>

<year>

<id>3</id>

<name>thirdyear</name>

<student>

<student>

<id>1</id>

<name>Preethika</name>

<dept>IT</dept>

</student>

<student>

<id>2</id>

<name>Vennela</name>

<dept>ECE</dept>

</student>

</student>

</year>

<year>

<id>4</id>

<name>fourthyear</name>

<student>

<student>

<id>1</id>

<name>Preethika</name>

<dept>IT</dept>

</student>

<student>

<id>2</id>

<name>Vennela</name>

<dept>ECE</dept>

</student>

</student>

</year>

User creation screen by using all elements

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>User Creation Form</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="container">

<h2>User Creation Form</h2>

<form action="/submit" method="POST">

<!-- Text Input -->

<label for="username">Username:</label>

<input type="text" id="username" name="username" required>

<!-- Password Input -->

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

<!-- Email Input -->

<label for="email">Email:</label>

<input type="email" id="email" name="email" required>

<!-- Drop-down Menu -->

<label for="country">Country:</label>

<select id="country" name="country">

<option value="usa">United States</option>

<option value="canada">Canada</option>

<option value="uk">United Kingdom</option>

<option value="australia">Australia</option>

</select>

<!-- Radio Buttons -->

<label>Gender:</label>

<div>

<input type="radio" id="male" name="gender" value="male">

<label for="male">Male</label>

</div>

<div>

<input type="radio" id="female" name="gender" value="female">

<label for="female">Female</label>

</div>

<div>

<input type="radio" id="other" name="gender" value="other">

<label for="other">Other</label>

</div>

<!-- Checkbox -->

<label>Interests:</label>

<div>

<input type="checkbox" id="sports" name="interests" value="sports">

<label for="sports">Sports</label>

</div>

<div>

<input type="checkbox" id="music" name="interests" value="music">

<label for="music">Music</label>

</div>

<div>

<input type="checkbox" id="tech" name="interests" value="tech">

<label for="tech">Technology</label>

</div>

<!-- Textarea -->

<label for="bio">Biography:</label>

<textarea id="bio" name="bio" rows="4" cols="50"></textarea>

<!-- List -->

<label for="skills">Skills:</label>

<select id="skills" name="skills[]" multiple>

<option value="html">HTML</option>

<option value="css">CSS</option>

<option value="js">JavaScript</option>

<option value="python">Python</option>

</select>

<!-- Submit Button -->

<button type="submit">Create User</button>

</form>

</div>

</body>

</html>

4:

| **Authentication** | **Authorization** |
| --- | --- |
| In the [authentication](https://www.geeksforgeeks.org/authentication-in-computer-network/) process, the identity of users are checked for providing the access to the system. | While in [authorization](https://www.geeksforgeeks.org/what-is-aaa-authentication-authorization-and-accounting/) process, a the person’s or user’s authorities are checked for accessing the resources. |
| In the authentication process, users or persons are verified. | While in this process, users or persons are validated. |
| It is done before the authorization process. | While this process is done after the authentication process. |
| It needs usually the user’s login details. | While it needs the user’s privilege or security levels. |
| Authentication determines whether the person is user or not. | While it determines **What permission does the user have?** |
| Generally, transmit information through an ID Token. | Generally, transmit information through an Access Token. |
| The OpenID Connect (OIDC) protocol is an authentication protocol that is generally in charge of user authentication process. | The OAuth 2.0 protocol governs the overall system of user authorization process. |
| Popular Authentication Techniques-   * Password-Based Authentication * Passwordless Authentication * 2FA/MFA (Two-Factor Authentication / Multi-Factor Authentication) * [Single sign-on (SSO)](https://www.geeksforgeeks.org/introduction-of-single-sign-on-sso/) * Social authentication | Popular  Authorization Techniques-   * Role-Based Access Controls (RBAC) * [JSON web token (JWT) Authorization](https://www.geeksforgeeks.org/json-web-token-jwt/) * SAML Authorization * OpenID Authorization * OAuth 2.0 Authorization |
| The authentication credentials can be changed in part as and when required by the user. | The authorization permissions cannot be changed by user as these are granted by the owner of the system and only he/she has the access to change it. |
| The user authentication is visible at user end. | The user authorization is not visible at the user end. |
| The user authentication is identified with username, password, face recognition, retina scan, fingerprints, etc. | The user authorization is carried out through the access rights to resources by using roles that have been pre-defined. |
| **Example**: Employees in a company are required to authenticate through the network before accessing their company email. | **Example:** After an employee successfully authenticates, the system determines what information the employees are allowed to access. |

5.Create a login screen

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login Screen</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="login-container">

<h2>Login</h2>

<form action="/login" method="post">

<div class="form-group">

<label for="username">Username:</label>

<input type="text" id="username" name="username" required>

</div>

<div class="form-group">

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

</div>

6.create a user creation screen by using all elements

<html>

<head>

<title> Registration </title>

</head>

<body>

<body style="background-color:B2D3C2">

<form>

<h3><b><center> REGISTRATION FORM </center> </b></h3>

<table border="0" align="center">

<tr>

<td>First Name : </td>

<td><input type ="text"></td>

</tr>

<tr>

<td>Last Name : </td>

<td><input type ="text"></td>

</tr>

<tr>

<td>Phone number: </td>

<td><input type ="text"></td>

</tr>

<tr>

<td>Gender : </td>

<td><input type ="radio" name="g">Female

<input type ="radio" name="g">Male

<input type ="radio" name="g">Other</td>

</tr>

<tr>

<td>

<ul>List of the subjects:</td>

<td><li>Java</li>

<li>C</li>

<li>c++</li></td>

</ul>

</tr>

<tr>

<td>Email: </td>

<td><input type ="text"></td>

</tr>

<tr>

<td>branch: </td>

<td><select name="branch">

<option>--SELECT--</option>

<option>CSE</option>

<option>EEE</option>

<option>ECE</option>

</td>

</tr>

<tr>

<td>Address: </td>

<td> <textarea rows="5" cols="5"> </textarea> </td>

</tr>

<tr>

<td>DOB: </td>

<td><input type ="date"></td>

</tr>

<tr>

<td>HOBBIES</td>

<td><input type="checkbox" name="rcb">playing cricket</input>

<input type="checkbox" name="rcb">reading books</input></td>

</tr>

<tr>

<td>Landmark: </td>

<td><input type="text" name="input" placeholder="(optional)"></td>

</tr>

</form>

</body>

</html>

<button type="submit">Login</button>

</form>

</div>

</body>

</html>

6.create a user creation screen by using all elements

7.List all users

from flask import Flask, jsonify

from flask\_sqlalchemy import SQLAlchemy

app = Flask(\_\_name\_\_)

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///users.db'

db = SQLAlchemy(app)

class User(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(80), nullable=False)

email = db.Column(db.String(120), unique=True, nullable=False)

@app.route('/users', methods=['GET'])

def list\_users():

users = User.query.all()

return jsonify([{'id': user.id, 'name': user.name, 'email': user.email} for user in users])

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

update user

from flask import request

@app.route('/user/<int:id>', methods=['PUT'])

def update\_user(id):

user = User.query.get\_or\_404(id)

data = request.get\_json()

user.name = data.get('name', user.name)

user.email = data.get('email', user.email)

db.session.commit()

return jsonify({'message': 'User updated successfully'})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

delete user: @app.route('/user/<int:id>', methods=['DELETE'])

def delete\_user(id):

user = User.query.get\_or\_404(id)

db.session.delete(user)

db.session.commit()

return jsonify({'message': 'User deleted successfully'})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

8.Html page with google map

<!DOCTYPE html>

<html>

<head>

<title>Google Map Example</title>

<style>

/\* Set the size of the div element that contains the map \*/

#map {

height: 100%;

}

/\* The height of the HTML and body elements must be set to 100% \*/

html, body {

height: 100%;

margin: 0;

padding: 0;

}

</style>

</head>

<body>

<div id="map"></div>

<script>

// Initialize and add the map

function initMap() {

// The location

const location = { lat: -34.397, lng: 150.644 };

// The map, centered at the location

const map = new google.maps.Map(document.getElementById("map"), {

zoom: 8,

center: location,

});

// The marker, positioned at the location

const marker = new google.maps.Marker({

position: location,

map: map,

});

}

</script>

<script async defer src="https://maps.googleapis.com/maps/api/js?key=YOUR\_API\_KEY&callback=initMap"></script>

</body>

</html>

9.Html page for video file

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Video Example</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

video {

max-width: 100%;

height: auto;

}

</style>

</head>

<body>

<h1>Video File Example</h1>

10.Html page for audio file

<video controls><!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Audio Example</title>

</head>

<body>

<h1>Audio File Example</h1>

<audio controls>

<source src="example.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

</body>

</html>

10.Html page to upload a file

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Audio Example</title>

</head>

<body>

<h1>Audio File Example</h1>

<audio controls>

<source src="example.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

</body>

</html>