Session 10

JSON in Real World

Objectives

- Describe the support of different browser and programming languages for JSON
- Describe JSON content types
- Explain how to use JSON for Web and Data Storage
- Compare JSON with relational databases
- Identify security and data portability issues with respect to JSON

Support for JSON

- Each major programming language can incorporate JSON, natively or through libraries.
- This is possible by incorporating two functionalities namely:
  - Parsing
  - Formatting

JSON with C# and Java

- These languages support statically typed classes and not objects of HashMap or Dictionary type.
- The solution is to use a library along with a custom code for converting these structures into static type instances.
- The Gson library from Google is one such library that resolves the issue.

What is Gson

- Is an open-source Java library for transforming a Java object to JSON data and vice-versa.
- Offers easy mechanisms such as constructor (factory method) and `toString()`.
- Functions well with arbitrary Java objects, involving the pre-existing ones.
- Serializes and deserializes huge data without any issues.
- Is convenient to learn and use by only using `toJson()` and `fromJson()`.

JSON with PHP

- From 5.2.0 version, the extension for JSON is packaged into PHP.
- Following table shows the functions for encoding and decoding JSON structures:

Function	Description
`json_encode`	Serializes the stated array or object and returns it in the JSON format if successful or FALSE.
`json_decode`	Deserializes JSON data and returns the suitable PHP type.
`json_last_error`	Returns the error that happened last.

MIME Type of JSON

- Is also called media type or content type.
- Is a two-part identifier composed of a type and subtype isolated by a slash.
- Aids in identifying the type of formatted content being sent over the Web.
- Is ‘application/json’ for JSON.
- Is unofficially ‘text/json’ or ‘text/Javascript’.

Applications of JSON

APIs	• For exchanging data to and from APIs • Popular in social networking sites implementing API
NoSQL	• For storing data easily in NoSQL databases, as JSON is easily convertible into JavaScript
Asynchronous JavaScript and JSON (AJAJ)	• For replacing XML when new data is fetched by a loaded Web page in a browser
JSON-RPC (Remote Procedure Call)	• For replacing Simple Object Access Protocol (SOAP) or XML-RPC and for sending several calls and notifications to a server
Package Management	• For storing metadata

JSON HTTP and Files

- JSON displays the data fetched from a Web server efficiently through the `XMLHttpRequest` object.
- The object exchanges data in the background, which prevents reloading the full page for updates.
- Following is the syntax of initializing the object:

```
variable = new XMLHttpRequest();
```

JSON for Data Storage

JSON helps in storing data fetched from services and applications.

- **Reasons**: Simplicity and 'just adequate structure'

RDBMS and Structured Query Language (SQL) are now replaced by NoSQL databases for developers who prefer JSON.

- **Trends**: More implementation of JSON-centric document databases, JSON in traditional RDBMS'
- **Reasons**: Developer-friendly and agility

Document versus Relational Databases

	Document-oriented Databases	Relational Databases
Coupling	Tighter due to the need to navigate the document while querying	Loose
Portability and Standardization	No, due to new query language setup for each document store	Yes, due to well-defined SQL query core
Optimization	Restricted, due to no abstraction between the logical data structure and its physical storage	Fully optimized, due to abstraction ensured by highly queryable stored data definition
Consistency and Normalization	No, due to no support for constraints	Yes

Benefits of Using JSON in RDBMS

Decision Power for:

- The data portions to be abstracted
- Areas where flexibility is essential
- Locations where strict schema and constraints are essential

JSON Query as Native SQL:

- Allows querying randomly across JSON records in tables
- Allows expanding systems to use JSON

JSON versus RDBMS Data Models

	JSON	RDBMS
Storage Structure	Is array or object.	Is a table.
Metadata	Can be in a schema, but is not pre-created.	Is stored in a schema generated while creating a table.
Data Retrieval	Uses evolving languages namely, JSON Query Language (JAQL) and JSONiq	Uses SQL.
Sorting	Is only for arrays.	Is for tables.
Learning Curve	Is smoother.	Is time consuming.
Application	Is used in several programming languages.	Is in the form of many commercial and open-source databases.

Security Issues in JSON

- JSON, as a flexible subset of JavaScript, has some security issues.
- In the following figure the root cause increasing the risk of malicious script, a major issue:

```
var data = eval('(' + JSONresponse + ')');
```

Issues with Eval()

- Most JSON text is syntactically JavaScript.
- JavaScript interpreter converts JSON into a JavaScript object without validation.
- This increases the risk of authentication forgery, identity and data theft, and misuse of resources.
- **Solution**: `JSON.parse()` and `JSON.stringify()` functions processing text only in JSON format

XSS and CSRF Issues

Cross Site Scripting (XSS)

Request: http://vulnerablesite.local/index?name=<script>alert("xss")</script>
Response:

```
<html>
    <body>
        <div>
            Hello <script>alert("xss")</script>
        </div>
    </body>
</html>
```

Cross Site Request Forgery (CSRF)

Request:
http://vulnerablesite.local/changepassword?newpwd=MyS3cr3tPa$$word

Attack:

```
<img src="http://vulnerablesite.local/changepassword?newpwd=owned">
```

XSS and CSRF Solution

By using the `jsonify()` function

Implementation Issues

Denial of Service (DoS) Attack	• Renders a resource or a machine on a network unavailable to its targeted users
Mass Assignment Vulnerability	• Maltreats the functioning pattern of record in a Web application for illegitimately changing confidential data items

Portability Issues

Unicode Line Terminators	Null Character	UTF Encoding	JSON Numbers	Unsupported Data Types
• Are allowed in JSON without being escaped and that they need to be backslash escaped for portability.	• Is allowed in JSON string if escaped as "\u0000", which creates issues with C strings.	• Involves having a few escaped characters using UTF-16 surrogate pairs, which a few JSON parsers do not recognize.	• Involve numbers and floating integers, which are distinguished by some languages only, not by all. • Have no specifications for rounding, overflow, and precision loss.	• Involve Error, Date, Undefined, Function, and Regular Expression, which JSON does not accept.

Handling JSON Securely

- Avoiding `eval()` by using a JavaScript library available at www.json.org, such as JSON sans `eval()`
- Ensuring data integrity via XMLHttpRequests

Summary 1-2

- Modern programming languages incorporate JSON via libraries or native parsing support.
- Gson is an open-source Java library for converting a Java object into JSON and vice-versa.
- Formal MIME type for JSON text is 'application/json'.
- JSON is used in several applications, such as in APIs, NoSQL databases, RDBMS', Web development, and application packages.

Summary 2-2

- Unlike relational databases, JSON does not have tables, pre-created metadata, and support for SQL.
- Using `eval()` for parsing JSON data can lead to security attacks such as XSS and CSRF.
- JSON is prone to DoS attack and vulnerability of mass assignment.
- Developers can secure JSON structures by replacing `eval()` with a JavaScript library and/or using XMLHttpRequests.