Web Application Development

Web APIs, JSON, AJAX

Web APIs

- A Web API (Application Programming Interface) is a set of rules and protocols that allows one application to communicate with another over the web.
- You can consume Web API Services using any front-end technology like JavaScript, Jquery, Angular or React.
- We will study Jquery to consume APIs, and Laravel to create our own API.

Web API

- Web APIs are created using server side lanaguages/technogies, and front end consumes it.
- WebAPIs are just a set of link that provides data in JSON formate.
  - For example https://jsonplaceholder.typicode.com/users
  - The response message contains a JSON object.
  - Some APIs may return data in XML format.

JSON vs XML

- Comparing JSON with XML
  - The simple excerpt of JSON and XML are as following

```json
{"employees":[
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
]}
```

JSON

```xml
<employees>
    <employee>
        <firstName>John</firstName>
        <lastName>Doe</lastName>
    </employee>
    <employee>
        <firstName>Anna</firstName>
        <lastName>Smith</lastName>
    </employee>
    <employee>
        <firstName>Peter</firstName>
        <lastName>Jones</lastName>
    </employee>
</employees>
```

XML

JSON

- JSON
  - **JSON** (JavaScript Object Notation) is a lightweight data-interchange format.
  - It is easy for humans to read and write.
  - It is easy for machines to parse and generate.
  - JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

JSON

- JSON object Syntax
  - { "name":"John", "age":30, "car":null }

  - JSON objects are surrounded by curly braces {}.
  - JSON objects are written in key/value pairs.
  - Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
  - Keys and values are separated by a colon.
  - Each key/value pair is separated by a comma.

JSON VS XML

- JSON

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

- XML

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

JSON VS XML

- JSON

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

JSON VS XML

- XML

```
<widget>
    <debug>on</debug>
    <window title="Sample Konfabulator Widget">
        <name>main_window</name>
        <width>500</width>
        <height>500</height>
    </window>
    <image src="Images/Sun.png" name="sun1">
        <hOffset>250</hOffset>
        <vOffset>250</vOffset>
        <alignment>center</alignment>
    </image>
    <text data="Click Here" size="36" style="bold">
        <name>text1</name>
        <hOffset>250</hOffset>
        <vOffset>100</vOffset>
        <alignment>center</alignment>
        <onMouseUp>
            sun1.opacity = (sun1.opacity / 100) * 90;
        </onMouseUp>
    </text>
</widget>
```

JSON

- Comparing JSON with XML
  - Both are self descriptive
  - Both are hierarchical
  - Both can be parsed by programming languages
  - JSON is shorter and therefore quicker
  - JSON doesn’t uses tags as XML

AJAX

- What is AJAX
  - Stands for Asynchronous JavaScript and XML
  - Ajax isn’t a language.
    - Dynamic display and interaction using the Document Object Model
    - Asynchronous data retrieval using XMLHttpRequest and Javascript

AJAX

- Ajax is not a programming language or a tool, but a feature/concept. Ajax is a client-side script that communicates to and from a server/database without the need for a postback or a complete page refresh.

- Ajax provides the method of exchanging data with a server, and updating parts of a web page – without reloading the entire page

A Simple Jquery AJAX Call

```
$(document).ready(function () {
  // AJAX call to a API, Replace with your Web API URL
  const url = "https://jsonplaceholder.typicode.com/users/1";

$.ajax({
        url: url,
        type: "GET",
        dataType: "json",
        success: function (data) {
          // Process the data varaible

        },

        error: function (xhr, status, error) {

          console.error("Error fetching data:", error);
          $('#output').html("<p>An error occurred.</p>");
        }
  });
});
```

On successful call, a data (as json) will be returned

On failure, a code of failure message will be returned

AJAX

- In the code (next slide) an ajax call is made to https://jsonplaceholder.typicode.com/users/1
- The response message contains a json object as described in following


```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
}
```

A Simple Jquery AJAX Call

```
<body>
  <div id="output"></div>

  <script>
    $(document).ready(function () {
      // AJAX call to a API
      const url = "https://jsonplaceholder.typicode.com/users/1";
// Replace with your URL
      $.ajax({
        url: url,
        type: "GET",
        dataType: "json",
        success: function (data) {
          // Dynamically create HTML from JSON response
          const output =
            <p><strong>ID:</strong> ${data.id}</p>
            <p><strong>Name:</strong> ${data.name}</p>
            <p><strong>Username:</strong>${data.username}</p>
    `;
          // Append the output to the div
          $('#output').html(output);
        },
        error: function (xhr, status, error) {
          console.error("Error fetching data:", error);
          $('#output').html("<p>An error occurred while fetching
data.</p>");
        }
      });
    });
  </script>
</body>
```

AJAX Call

- Response

Id: 1

Name: Leanne Graham

email: Sincere@april.biz

AJAX Call

- The HTTP request contains following important parameters
  - Method
    - Get, Post, Put, Delete, Patch
  - URL
    - url of target web site
  - Data
    - An optional parameter having JSON object. It is required when you want to send data through HTTP request

AJAX

- In the code (next slide)
  - an ajax call is made to https://jsonplaceholder.typicode.com/users
  - The response message contains a json object as described in following (see next)

```
[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2,
    "name": "Ervin Howell",
    "username": "Antonette",
    "email": "Shanna@melissa.tv",
    "address": {
      "street": "Victor Plains",
      "suite": "Suite 879",
      "city": "Wisokyburgh",
      "zipcode": "90566-7771",
      "geo": {
        "lat": "-43.9509",
        "lng": "-34.4618"
      }
    },
    "phone": "010-692-6593 x09125",
    "website": "anastasia.net",
    "company": {
      "name": "Deckow-Crist",
      "catchPhrase": "Proactive didactic contingency",
      "bs": "synergize scalable supply-chains"
    }
  },
  {
    "id": 3,
    "name": "Clementine Bauch",
    "username": "Samantha",
    "email": "Nathan@yesenia.net",
    "address": {
      "street": "Douglas Extension",
      "suite": "Suite 847",
      "city": "McKenziehaven",
      "zipcode": "59590-4157",
      "geo": {
        "lat": "-68.6102",
        "lng": "-47.0653"
      }
    },
    "phone": "1-463-123-4447",
    "website": "ramiro.info",
    "company": {
      "name": "Romaguera-Jacobson",
      "catchPhrase": "Face to face bifurcated interface",
      "bs": "e-enable strategic applications"
    }
  },
  {
    "id": 4,
    "name": "Patricia Lebsack",
    "username": "Karianne",
    "email": "Julianne.OConner@kory.org",
    "address": {
      "street": "Hoeger Mall",
      "suite": "Apt. 692",
      "city": "South Elvis",
      "zipcode": "53919-4257",
      "geo": {
        "lat": "29.4572",
        "lng": "-164.2990"
      }
    },
    "phone": "493-170-9623 x156",
    "website": "kale.biz",
    "company": {
      "name": "Robel-Corkery",
      "catchPhrase": "Multi-tiered zero tolerance productivity",
      "bs": "transition cutting-edge web services"
    }
  },
  {
    "id": 5,
    "name": "Chelsey Dietrich",
    "username": "Kamren",
    "email": "Lucio_Hettinger@annie.ca",
    "address": {
      "street": "Skiles Walks",
```

AJAX call

Id	Name	email
1	Leanne Graham	Sincere@april.biz
2	Ervin Howell	Shanna@melissa.tv
3	Clementine Bauch	Nathan@yesenia.net
4	Patricia Lebsack	Julianne.OConner@kory.org
5	Chelsey Dietrich	Lucio_Hettinger@annie.ca
6	Mrs. Dennis Schulist	Karley_Dach@jasper.info
7	Kurtis Weissnat	Telly.Hoeger@billy.biz
8	Nicholas Runolfsdottir V	Sherwood@rosamond.me
9	Glenna Reichert	Chaim_McDermott@dana.io
10	Clementina DuBuque	Rey.Padberg@karina.biz

Task

- Once you have web APIs in hand, now it is possible to consume them from any type of application irrespective of their technology.
- We can use AJAX to send some new data to the server. And get response from server in JSON format, Process or display the data using DOM.

Task

- Create a dashboard where you can execute
  - Add
  - Update
  - Get
  - Get all
  - Delete
- Operation on
  - http://exampleapi.somee.com/api/person

Dashboard

Users Record

Add New Record

Search person by Name/Email: Search

USER ID	NAME	EMAIL	PHONE	CITY	OPERATIONS
1	Leanne Graham	Sincere@april.biz	1-770-736-8031 x56442	Gwenborough	View | Edit | Delete
2	Ervin Howell	Shanna@melissa.tv	010-692-6593 x09125	Wisokyburgh	View | Edit | Delete
3	Clementine Bauch	Nathan@yesenia.net	1-463-123-4447	McKenziehaven	View | Edit | Delete
4	Patricia Lebsack	Julianne.OConner@kory.org	493-170-9623 x156	South Elvis	View | Edit | Delete
5	Chelsey Dietrich	Lucio_Hettinger@annie.ca	(254)954-1289	Roscoeview	View | Edit | Delete

Web API: Get All records

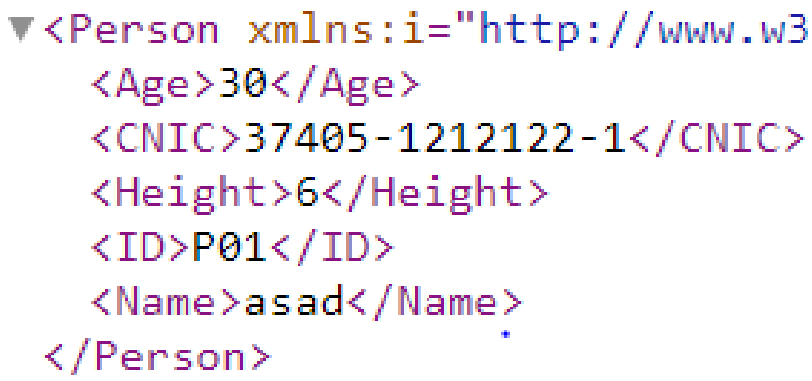
– So far you have used Web APIs to

- Get all records

Web API: Get single recored

– So far you have created following web APIs

- Get specific record



Reference contents

- Some of HTTP status code

HTTP Status Codes

Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Unauthorized
202	Accepted	403	Forbidden
301	Moved Permanently	404	Not Found
303	See Other	410	Gone
304	Not Modified	500	Internal Server Error
307	Temporary Redirect	503	Service Unavailable

  - https://developer.mozilla.org/en-US/docs/Web/HTTP/Status