

JSON - Introduzione

JSON sta **per Java** Script Object **Notation**

JSON è un **formato di testo** per l'archiviazione e il trasporto di dati

JSON è "autodescrittivo" e facile da capire

Esempio JSON

```
{ "name": "John",  
  "age": 30,  
  "car": null }
```

Definisce un oggetto con 3 proprietà:

- nome
- età
- auto

Ogni proprietà ha un valore.

Per ricevere dati da un server web è possibile utilizzare sia JSON che XML.

I seguenti esempi JSON e XML definiscono entrambi un oggetto dipendenti, con un array di 3 dipendenti:

I seguenti esempi JSON e XML definiscono entrambi un oggetto dipendenti, con un array di 3 dipendenti:

Esempio JSON

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

Esempio 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>
```

JSON è come XML perché

- Sia JSON che XML sono "autodescrittivi" (leggibili dall'uomo)
- Sia JSON che XML sono gerarchici (valori all'interno di valori)
- Sia JSON che XML possono essere analizzati e utilizzati da molti linguaggi di programmazione
- Sia JSON che XML possono essere recuperati con un XMLHttpRequest

JSON è diverso da XML perché

- JSON non utilizza il tag di fine
- JSON è più corto
- JSON è più veloce da leggere e scrivere
- JSON può utilizzare array

La differenza più grande è:

XML deve essere analizzato con un parser XML. JSON può essere analizzato da una funzione JavaScript standard.

Tipi di dati validi

In JSON, i valori devono essere uno dei seguenti tipi di dati:

- una stringa
- un numero
- un oggetto (oggetto JSON)
- un array
- un booleano
- *null*

Stringhe JSON

Le stringhe in JSON devono essere scritte tra virgolette doppie.

Esempio

```
{"name": "John"}
```

Numeri JSON

I numeri in JSON devono essere numeri interi o in virgola mobile.

Esempio

```
{"age": 30}
```

Oggetti JSON

I valori in JSON possono essere oggetti.

Esempio

```
{  
  "employee": {"name": "John", "age": 30, "city": "New York"}  
}
```

Gli oggetti come valori in JSON devono seguire la sintassi JSON.

Array JSON

I valori in JSON possono essere array.

Esempio

```
{  
  "employees": ["John", "Anna", "Peter"]  
}
```

Booleani JSON

I valori in JSON possono essere true/false.

Esempio

```
{"sale": true}
```

JSON null

I valori in JSON possono essere nulli.

Esempio

```
{"middlename":null}
```

JSON .parse()

Un utilizzo comune di JSON è lo scambio di dati da/verso un server web.

Quando si ricevono dati da un server web, i dati sono sempre una stringa.

Analizza i dati con `JSON.parse()` e i dati diventano un oggetto JavaScript.

Esempio - Analisi JSON

Immaginiamo di ricevere questo testo da un server web:

```
'{"name":"John", "age":30, "city":"New York"}'
```

Utilizzare la funzione JavaScript `JSON.parse()` per convertire il testo in un oggetto JavaScript:

```
const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
```

JSON .stringify()

Un utilizzo comune di JSON è lo scambio di dati da/verso un server web.

Quando si inviano dati a un server web, i dati devono essere una stringa.

È possibile convertire qualsiasi tipo di dati JavaScript in una stringa con `JSON.stringify()`.

Immaginiamo di avere questo oggetto in JavaScript:

```
const obj = {name: "John", age: 30, city: "New York"};
```

Utilizzare la funzione JavaScript `JSON.stringify()` per convertirlo in una stringa.

```
const myJSON = JSON.stringify(obj);
```

Il risultato sarà una stringa che segue la notazione JSON.

`myJSON` è ora una stringa, pronta per essere inviata a un server

JSON - PHP

Un utilizzo comune di JSON è la lettura di dati da un server web e la loro visualizzazione in una pagina web.

In questo capitolo verrà illustrato come scambiare dati JSON tra il client e un server PHP.

Il file PHP(demo_file.php)

PHP ha alcune funzioni integrate per gestire JSON.

Gli oggetti in PHP possono essere convertiti in JSON utilizzando la funzione PHP `json_encode()` :

```
<?php

$myObj = new stdClass();
$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";

$myJSON = json_encode($myObj);

echo $myJSON;
?>
```

(stdClass è una comoda funzionalità fornita da PHP per creare una classe regolare. È una classe 'vuota' predefinita usata come classe di utilità per eseguire il cast di oggetti di altri tipi. Non ha genitori, proprietà o metodi)

Il client JavaScript(demo.html)

Ecco un JavaScript sul client, che utilizza una chiamata AJAX per richiedere il file PHP dall'esempio precedente:

Esempio

Utilizzare `JSON.parse()` per convertire il risultato in un oggetto JavaScript:

```
<!DOCTYPE html>
<html>
```

```

<body>
<h2>Get JSON Data from a PHP Server</h2>
<p id="demo"></p>
<script>
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
    const myObj = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myObj.name;
}
xmlhttp.open("GET", "demo_file.php");
xmlhttp.send();

</script>
</body>
</html>

```

Array PHP

Gli array in PHP verranno convertiti anche in JSON quando si utilizza la funzione PHP `json_encode()` :

File PHP(demo_file_array.php)

```

<?php
$myArr = array("John", "Mary", "Peter", "Sally");

$myJSON = json_encode($myArr);

echo $myJSON;
?>

```

Il client JavaScript(demoarray.html)

Ecco un JavaScript sul client, che utilizza una chiamata AJAX per richiedere il file PHP dall'esempio di array sopra:

Esempio

Utilizzare `JSON.parse()` per convertire il risultato in un array JavaScript:

```

<!DOCTYPE html>
<html>
<body>
<h2>Get JSON Data from a PHP Server</h2>
<p>Convert the data into a JavaScript array:</p>
<p id="demo"></p>
<script>

```

```

var xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
    const myObj = JSON.parse(this.responseText);
    document.getElementById("demo").innerHTML = myObj[2];
}
xmlhttp.open("GET", "demo_file_array.php", true);
xmlhttp.send();

</script>
</body>
</html>

```

DATABASE E PHP

PHP è un linguaggio di programmazione lato server e può essere utilizzato per accedere a un database.

Immagina di avere un database sul tuo server e di voler inviare una richiesta dal client in cui chiedi le righe di una tabella chiamata "clienti" con parametri di selezione (città,età) per estrarre solo i maggiorenni di una determinata città

Sul client, crea un oggetto JSON (*Utilizzare `JSON.stringify()` per convertire l'oggetto JavaScript in JSON*) che descriva il numero di righe che vuoi restituire.

I dati inviati al server devono essere un argomento del `send()` metodo

Esempio (json_demo_db_post.html)

```

<!DOCTYPE html>
<html>
<body>

<h2>Use HTTP POST to Get JSON Data from a PHP Server</h2>

<p id="demo"></p>

<script>
const dbParam = JSON.stringify({città:"Salerno",età:18});
const xmlhttp = new XMLHttpRequest();
xmlhttp.onload = function() {
    myObj = JSON.parse(this.responseText);
    let text = "";
    for (let x in myObj) {
        text += myObj[x].nome + "<br>";
    }
    document.getElementById("demo").innerHTML = text;
}
xmlhttp.open("POST", "json_demo_db_post.php");

```

```
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("x=" + dbParam);
</script>
</body>
</html>
```

Esempio spiegato:

- Definire un oggetto contenente due campi chiave – valore(città ed età).
- Converti l'oggetto in una stringa JSON.
- Invia una richiesta al file PHP, con la stringa JSON come parametro del metodo send().
- Attendi che la richiesta ritorni con il risultato (come JSON)
- Visualizza il risultato ricevuto dal file PHP.

File PHP(json_demo_db_post.php)

```
<?php
header("Content-Type: application/json; charset=UTF-8");
$array_parametri = json_decode($_POST["x"], true);
$citta=$array_parametri["citta"];
$eta=$array_parametri["eta"];
$hostname = "localhost";
$username = "root";
$password = "";
$database = "xml";
$conn = new mysqli($hostname, $username, $password, $database);
$query="SELECT id,nome,eta,citta FROM clienti where  citta='".$citta.'" and eta
<='".$eta;

$result = $conn->query($query);
$outp = $result->fetch_all(MYSQLI_ASSOC);
echo json_encode($outp);
?>
```

Spiegazione del file PHP:

- Convertire la richiesta in un oggetto, utilizzando la funzione PHP `json_decode()` (il parametro true converte in array associativo).
- Accedere al database e compilare un array con i dati richiesti.
- Aggiungere l'array a un oggetto e restituire l'oggetto come JSON utilizzando la funzione `json_encode()` .

INTERAZIONE CON I CAMPI DI UN FORM(*Con array associativo*)

In questo esempio i dati della stringa JSON inviati al server sono presi dai campi di input di un form e diventano argomento del send() metodo.

I dati inviati al server devono essere un argomento del send() metodo

Esempio (json_demo_db_post_form.html)

```
<!DOCTYPE html>
<html>
<body>
<h2>Make a table based on values of a form.</h2>
<form name="Nuovo_Componente" id="Nuovo_Componente" method="post">
<table>
<tr>
<td><p><b>Citta':</b></p></td>
<td><input type="text" name="citta" id="citta" size="30"></td>
</tr>
<tr>
<td><p><b>Eta':</b></p></td>
<td><input type="text" name="eta" id="eta" size="3"></td>
</tr>
<tr>
<td><input type="button" value="Invia" onClick="Invia()" /></td>
<td><input type="reset" value="Annulla" /></td>
</tr>
</table>
</form>
<p id="demo"></p>
<script>
function Invia() {
  var formData = {
    citta: document.getElementById("citta").value,
    eta: Number(document.getElementById("eta").value)
  };
  const dbParam = JSON.stringify(formData);
  const xmlhttp = new XMLHttpRequest();
  xmlhttp.onload = function() {
    myObj = JSON.parse(this.responseText);
    text = "<table border='1'>"
    text += "<tr><th colspan='3'>" + "ELENCO MAGGIORENNI</th></tr>";
    text += "<tr><th>" + "Nominativo" + "</th>";
    text += "<th>" + "Eta" + "</th>";
    text += "<th>" + "Citta" + "</th></tr>";
    for (x in myObj) {
      text += "<tr><td>" + myObj[x].nome + "</td>";
```

```

        text += "<td>" + myObj[x].eta + "</td>";
        text += "<td>" + myObj[x].citta + "</td></tr>";
    }
    text += "</table>"
    document.getElementById("demo").innerHTML = text;
}
xmlhttp.open("POST", "json_demo_db_post_form.php",true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
xmlhttp.send("x=" + dbParam);
}
</script>
</body>
</html>

```

Esempio spiegato:

- Al click sul button Invia:
 - Definire un oggetto contenente due campi chiave – valore(città ed età) i cui valori sono presi dai campi input di un form.
 - Converti l'oggetto in una stringa JSON.
 - Invia una richiesta al file PHP, con la stringa JSON come parametro del metodo send().
 - Attendi che la richiesta ritorni con il risultato (come JSON)
 - Visualizza il risultato ricevuto dal file PHP in una tabella html.

File PHP(json_demo_db_post_form.php)

```

<?php
header("Content-Type: application/json; charset=UTF-8");
$array_parametri = json_decode($_POST["x"], true);
$citta=$array_parametri["citta"];
$eta=$array_parametri["eta"];
$hostname = "localhost";
$username = "root";
$password = "";
$databse = "xml";
$conn = new mysqli($hostname, $username, $password, $databse);
$query="SELECT id,nome,eta,citta FROM clienti where  citta='".$citta.'" and eta
>='".$eta;

$result = $conn->query($query);
$outp = $result->fetch_all(MYSQLI_ASSOC);
echo json_encode($outp);
?>

```

Spiegazione del file PHP:

- Convertire la richiesta in un oggetto, utilizzando la funzione PHP `json_decode()` (il parametro true converte in array associativo).
- Accedere al database e compilare un array con i dati richiesti.
- Aggiungere l'array a un oggetto e restituire l'oggetto come JSON utilizzando la funzione `json_encode()` .

INTERAZIONE CON I CAMPI DI UN FORM(*Senza array associativo*)

In questo esempio i dati della stringa JSON inviati al server sono presi dai campi di input di un form e diventano argomento del `send()` metodo.

I dati inviati al server devono essere un argomento del `send()` metodo

Esempio (json_demo_db_post_form_senza_array.html)

```
<!DOCTYPE html>
<html>
<body>
<h2>Make a table based on values of a form.</h2>
<form name="Nuovo_Componente" id="Nuovo_Componente" method="post">
<table>
<tr>
<td><p><b>Citta':</b></p></td>
<td><input type="text" name="citta" id="citta" size="30"></td>
</tr>
<tr>
<td><p><b>Eta':</b></p></td>
<td><input type="text" name="eta" id="eta" size="3"></td>
</tr>
<tr>
<td><input type="button" value="Invia" onClick="Invia()" /></td>
<td><input type="reset" value="Annulla" /></td>
</tr>
</table>
</form>
<p id="demo"></p>
<script>
function Invia() {
  var formData = {
    citta: document.getElementById("citta").value,
    eta: Number(document.getElementById("eta").value)
  };
  const dbParam = JSON.stringify(formData);
  const xmlhttp = new XMLHttpRequest();
  xmlhttp.onload = function() {
    myObj = JSON.parse(this.responseText);
    text = "<table border='1'>"
```

```

text += "<tr><th colspan=\"3\">" + "ELENCO MAGGIORENNI</th></tr>";
text += "<tr><th>" + "Nominativo" + "</th>";
text += "<th>" + "Eta" + "</th>";
text += "<th>" + "Citta" + "</th></tr>";
for (x in myObj) {
text += "<tr><td>" + myObj[x].nome + "</td>";
text += "<td>" + myObj[x].eta + "</td>";
text += "<td>" + myObj[x].citta + "</td></tr>";
}
text += "</table>";
document.getElementById("demo").innerHTML = text;
}
xmlhttp.open("POST", "json_demo_db_post_form_senza_array.php",true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
xmlhttp.send("x=" + dbParam);
}
</script>
</body>
</html>

```

Esempio spiegato:

- Al click sul button Invia:
 - Definire un oggetto contenente due campi chiave – valore(città ed età) i cui valori sono presi dai campi input di un form.
 - Converti l'oggetto in una stringa JSON.
 - Invia una richiesta al file PHP, con la stringa JSON come parametro del metodo send().
 - Attendi che la richiesta ritorni con il risultato (come JSON)
 - Visualizza il risultato ricevuto dal file PHP in una tabella html.

File PHP(json_demo_db_post_form_senza_array.php)

```

<?php
header("Content-Type: application/json; charset=UTF-8");
$array_parametri = json_decode($_POST["x"]);
$citta=$array_parametri->citta;
$eta=$array_parametri->eta;
$hostname = "localhost";
$username = "root";
$password = "";
$databse = "xml";
$conn = new mysqli($hostname, $username, $password, $databse);
$query="SELECT id,nome,eta,citta FROM clienti where  citta='".$citta."' and eta
>=".$eta;

$result = $conn->query($query);
$outp = $result->fetch_all(MYSQLI_ASSOC);

```

```
echo json_encode($outp);  
?>
```

Spiegazione del file PHP:

- Convertire la richiesta in un oggetto, utilizzando la funzione PHP `json_decode()` (il parametro `false` sottinteso converte in oggetto php, i valori sono ricavati dalle chiavi usando l' operatore `->`).
- Accedere al database e compilare un array con i dati richiesti.
- Aggiungere l'array a un oggetto e restituire l'oggetto come JSON utilizzando la funzione `json_encode()` .