CODES

Capacitación Web – AJAX

ÍNDICE

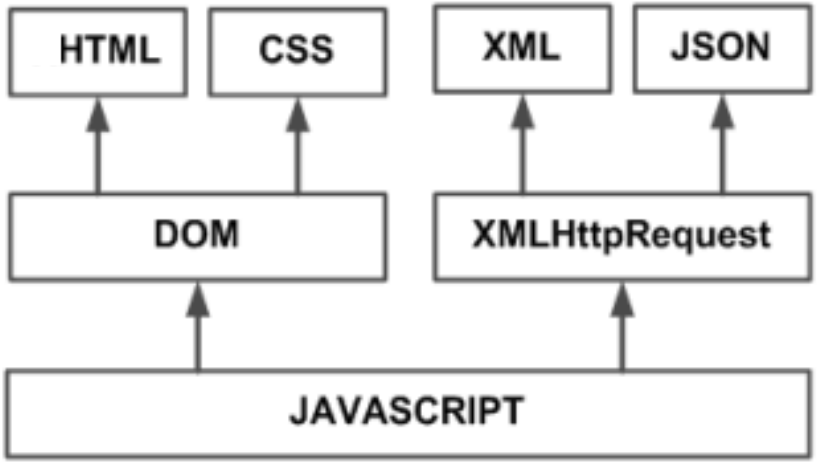
CONCEPTOS

- **Ajax**: -acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML (Extensible Markup Language)
  - es una técnica de desarrollo web que permite la comunicación entre el cliente y el servidor de manera asincrónica, no es un lenguaje de programación, sino que utiliza conjuntamente varias tecnologías existentes para crear aplicaciones web.
  -Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Es posible lograr aplicaciones web capaces de actualizarse continuamente sin tener que volver a cargar la página completa. mejorando la interactividad, velocidad y usabilidad.

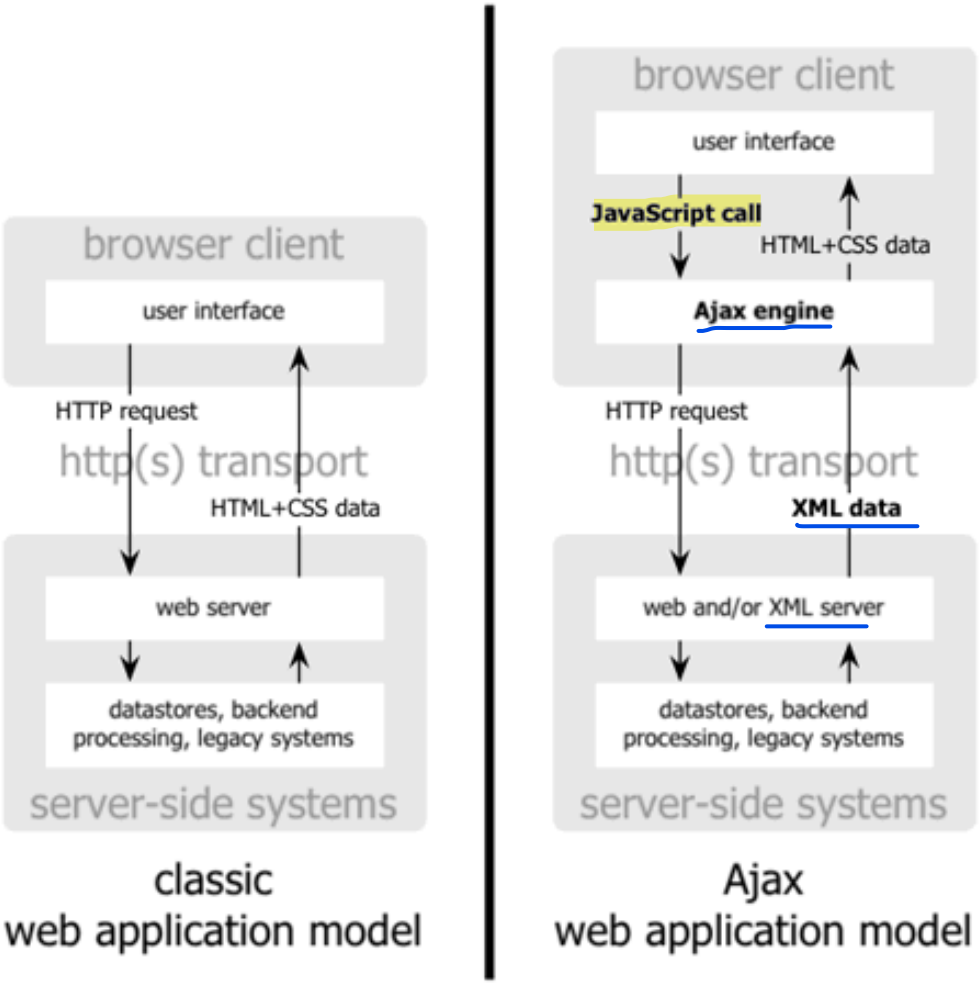
- **HttpHandler**: Un ASP.NET HTTP handler es el proceso que se ejecuta en respuesta a una solicitud realizada a una aplicación web ASP.NET. El manejador (handler) más común en ASP.NET es el que procesa los archivos .aspx (las páginas) y también el .ashx que es el que permite procesar peticiones.

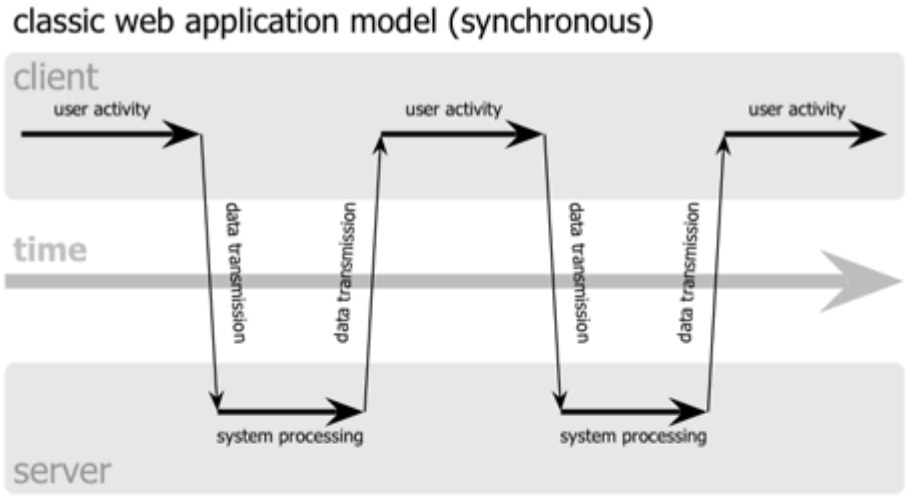
Las tecnologías que forman AJAX son:

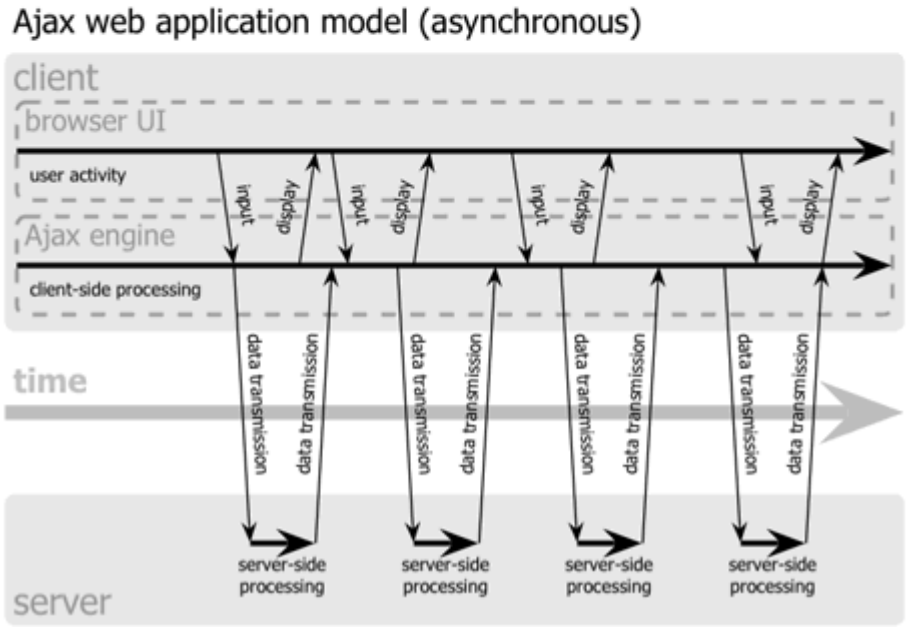
- HTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.



En el siguiente esquema, la imagen de la izquierda muestra el modelo tradicional de las aplicaciones web. La imagen de la derecha muestra el nuevo modelo propuesto por AJAX:







El esquema de la izquierda muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX. La imagen superior muestra la interación síncrona propia de las aplicaciones web tradicionales. La imagen inferior muestra la comunicación asíncrona de las aplicaciones creadas con AJAX.

1. AJAX

Antes de entrar en un ejemplo práctico de Ajax, introduciremos el concepto de JSON:

JSON (JavaScript Object Notation) es la forma de denotar objetos en javascript. Es un formato sencillo que define una estructura de datos para para el intercambio de información- La notación de objetos mediante JSON es una de las características principales de JavaScript y es un mecanismo definido en los fundamentos básicos del lenguaje.

En los últimos años, JSON se ha convertido en una alternativa al formato XML, ya que es más fácil de leer y escribir, además de ser mucho más conciso. No obstante, XML es superior técnicamente porque es un lenguaje de marcado, mientras que JSON es simplemente un formato para intercambiar datos.

La especificación completa de JSON se puede consultar en RFC 4627 y su tipo MIME oficial es application/json.

EJEMPLO JSON:

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

Ejemplo Ajax:

Vamos a usar, a modo de servidor, para hacer la petición, una API de chistes random.

URL: https://official-joke-api.appspot.com/jokes/programming/random

Y vamos a ver, en la práctica, como se realiza con el método tradicional y también utilizando AJAX para ver alguna de las diferencias de las que charlamos.

- Modo sin AJAX:

```html
<html>
    <head>
        <title>Ejemplo Sin AJAX</title>
    </head>
    <body>
       <form type="GET" action="https://official-joke-api.appspot.com/jokes/programming/random">
        <input type="submit" value="Contar Chiste"></input>
       </form>
    </body>
</html>
```

- Modo con AJAX:

```html
<html>
    <head>
        <script type="text/javascript" src="main.js"></script>
        <title>Ejemplo Con AJAX</title>
    </head>
    <body>
        <input type="button" value="Contar Chiste" onclick="onClickEnviar()"></input><br/>
        <label>Setup:</label>
        <input type="text" value="" style='width:800px' id="txtSetup"></input><br/>
        <label>Punchline:</label>
        <input type="text" value="" style='width:800px' id="txtPunchline"></input>
    </body>
</html>
```

- Modo con AJAX: .js

```
function onClickEnviar(){
    //Objeto para realizar la peticion AJAX
    var peticion = new XMLHttpRequest();

    //Indico que se va a ejecutar una vez que la misma finalice
    peticion.onreadystatechange = function(){
        if(this.readyState == 4 && this.status==200){
            handlerResultado(this);
        }
    }
    //Indico a donde voy a hacer la peticion
    peticion.open("GET", "https://official-joke-api.appspot.com/jokes/programming/random");

    //Disparo la peticion
    peticion.send();
}

function handlerResultado (This) {

    var res = JSON.parse(This.responseText)[0];
    document.getElementById("txtSetup").value = res.setup;
    document.getElementById("txtPunchline").value = res.punchline;
}
```

- Modo con AJAX y JQuery:

```html
<html>
    <head>
        <script type="text/javascript" src="https://code.jquery.com/jquery-3.6.0.js"></script>
        <script type="text/javascript" src="mainJQ.js"></script>
        <title>Ejemplo Con AJAX y JQuery</title>
    </head>
    <body>
        <input type="button" id="ajaxBtn" value="Contar Chiste" onclick="onClickEnviar()"></input><br/>
        <label>Setup:</label>
        <input type="text" value="" style='width:800px' id="txtSetup"></input><br/>
        <label>Punchline:</label>
        <input type="text" value="" style='width:800px' id="txtPunchline"></input>
    </body>
</html>
```

- Modo con AJAX y JQuery: .js

```
function onClickEnviar(){
    $.ajax({
        method: "GET",
        url: "https://official-joke-api.appspot.com/jokes/programming/random",
        dataType: 'json',
        success: function (data){
            handlerResultado(data);
        },
        error: function (jqXhr, textStatus, errorMessage) { // error callback
            console.log('Error: ' + errorMessage);
        }
      });
}

function handlerResultado (data) {
        var res = data[0];
       $("#txtSetup").val(res.setup);
       $("#txtPunchline").val(res.punchline);
}

```

FIN

Gracias

codes
CONSULTORIA
Y DESARROLLO DE SISTEMAS

CMMI

AJAX Básico

Material obligatorio: ¿Qué es AJAX?

Con AJAX (Asynchronous JavaScript And Xml) podremos realizar lo siguiente:

- Actualizar una página web sin recargar completamente la página.
- Solicitar datos al servidor luego de haber cargado la página.
- Recibir datos del servidor luego de haber cargado la página.
- Enviar datos al servidor como una tarea de fondo, es decir, que no ocupe el procesamiento central.

AJAX no es un lenguaje de programación, sino que es una combinación de 2 elementos:

- Un objeto embebido en el navegador llamado "XMLHttpRequest" (que utilizaremos para solicitar datos al servidor).
- Objetos HTML y JavaScript para mostrar o usar los datos que solicitamos.

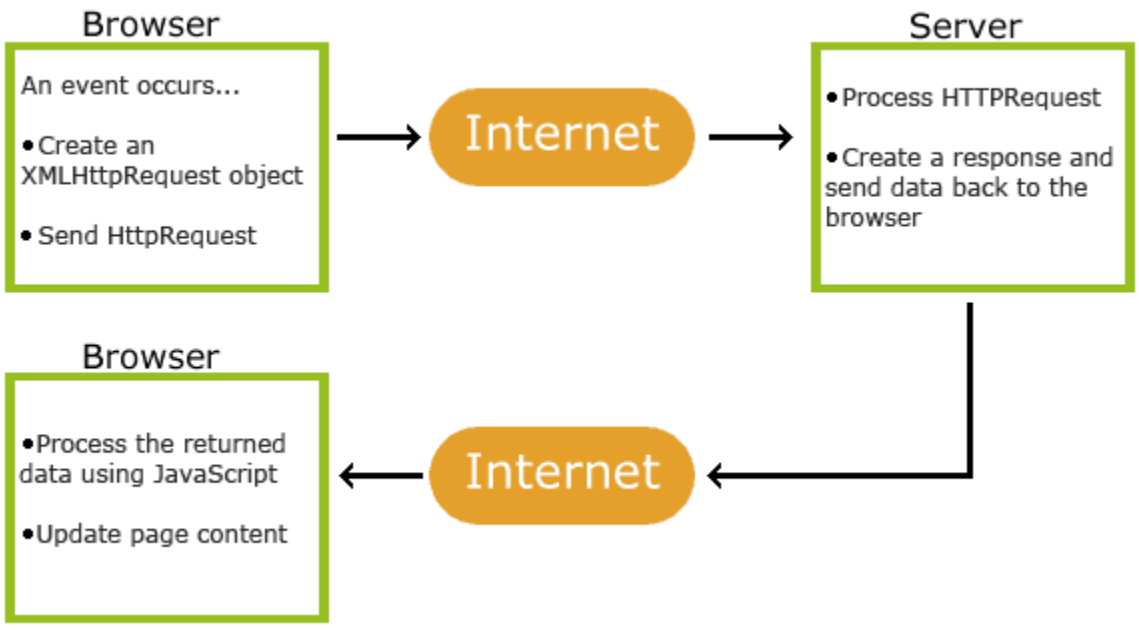
Last modified: Thursday, 17 January 2019, 9:39 AM

◄ Material obligatorio: curso de JSON | Jump to... | Material obligatorio: ¿Cómo funciona AJAX? ►

AJAX Básico

Material obligatorio: ¿Cómo funciona AJAX?

A continuación mostramos gráficamente como es la interacción entre los distintos actores (cliente - servidor) al realizar una petición AJAX:



1. Ocurre un evento en la página web (carga de la página, el usuario hace click en un botón, escribe en un campo de texto, etc).

2. Un objeto XMLHttpRequest es creado por medio de JavaScript (lenguaje del cliente).

3. El objeto XMLHttpRequest envía una petición al servidor.

4. El servidor procesa la petición.

5. El servidor envía la respuesta al cliente.

6. La respuesta es procesada por medio de JavaScript (lenguaje del cliente).

7. Se actualiza parte de la página mediante JavaScript.

Last modified: Thursday, 17 January 2019, 9:39 AM

◄ Material obligatorio: ¿Qué es AJAX? Jump to... Material complementario: ReadyState y códigos de respuesta ►

AJAX Básico

Material complementario: ReadyState y códigos de respuesta

Valores posibles de XMLHttpRequest.ReadyState:

0	UNSENT	Se creó el cliente pero `open()` no fue invocado.
1	OPENED	`open()` fue invocado
2	HEADERS_RECEIVED	`send()` fue invocado. Encabezado y status disponibles.
3	LOADING	Descargando la respuesta; `responseText` contiene datos parciales.
4	DONE	Se completó la operación

Códigos de estado de respuesta HTTP más utilizados:

200 OK
400 Bad Request
401 Unauthorized
402 Forbidden
404 Not Found
500 Internal Server Error

Last modified: Thursday, 17 January 2019, 9:39 AM

◄ Material obligatorio: ¿Cómo funciona AJAX? Jump to... Demo simple AJAX ►

JSON - Introduction

❮ Previous Next ❯

JSON

JSON stands for **J**ava**S**cript **O**bject **N**otation

JSON is a **text format** for storing and transporting data

JSON is "self-describing" and easy to understand

JSON Example

This example is a JSON string:

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

It defines an object with 3 properties:

- name
- age
- car

Each property has a value.

If you parse the JSON string with a JavaScript program, you can access the data as an object:

What is JSON?

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight data-interchange format
- JSON is plain text written in JavaScript object notation
- JSON is used to send data between computers
- JSON is language independent *

*
The JSON syntax is derived from JavaScript object notation, but the JSON format is text only.

Code for reading and generating JSON exists in many programming languages.

The JSON format was originally specified by Douglas Crockford.

Why Use JSON?

The JSON format is syntactically similar to the code for creating JavaScript objects. Because of this, a JavaScript program can easily convert JSON data into JavaScript objects.

Since the format is text only, JSON data can easily be sent between computers, and used by any programming language.

JavaScript has a built in function for converting JSON strings into JavaScript objects:

`JSON.parse()` JSON -> JavaScript

JavaScript also has a built in function for converting an object into a JSON string:

`JSON.stringify()` JavaScript -> JSON

You can receive pure text from a server and use it as a JavaScript object.

You can send a JavaScript object to a server in pure text format.

Storing Data

When storing data, the data has to be a certain format, and regardless of where you choose to store it, *text* is always one of the legal formats.

JSON makes it possible to store JavaScript objects as text.

❮ Previous Log in to track progress Next ❯

JSON Syntax

‹ Previous Next ›

The JSON syntax is a subset of the JavaScript syntax.

JSON Syntax Rules

JSON syntax is derived from JavaScript object notation syntax:

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

JSON Data - A Name and a Value

JSON data is written as name/value pairs (aka key/value pairs).

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

Example

```
"name":"John"
```

JSON names require double quotes.

In JSON, *keys* must be strings, written with double quotes:

JSON

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

In JavaScript, keys can be strings, numbers, or identifier names:

JavaScript

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

JSON Values

In **JSON**, *values* must be one of the following data types:

- a string
- a number
- an object
- an array
- a boolean
- null

In **JavaScript** values can be all of the above, plus any other valid JavaScript expression, including:

- a function
- a date
- undefined

In JSON, *string values* must be written with double quotes:

JSON

In JavaScript, you can write string values with double *or* single quotes:

JavaScript

```
{name:'John'}
```

JavaScript Objects

Because JSON syntax is derived from JavaScript object notation, very little extra software is needed to work with JSON within JavaScript.

With JavaScript you can create an object and assign data to it, like this:

Example

```
person = {name:"John", age:31, city:"New York"};
```

You can access a JavaScript object like this:

Example

```
// returns John
person.name;
```

Try it Yourself »

It can also be accessed like this:

Example

```
// returns John
person["name"];
```

Data can be modified like this:

Example

```
person.name = "Gilbert";
```

Try it Yourself »

It can also be modified like this:

Example

```
person["name"] = "Gilbert";
```

Try it Yourself »

You will learn how to convert JavaScript objects into JSON later in this tutorial.

JavaScript Arrays as JSON

The same way JavaScript objects can be written as JSON, JavaScript arrays can also be written as JSON.

You will learn more about objects and arrays later in this tutorial.

JSON Files

- The file type for JSON files is ".json"
- The MIME type for JSON text is "application/json"

JSON Data Types

❮ Previous Next ❯

Valid Data Types

In JSON, values must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- *null*

JSON values **cannot** be one of the following data types:

- a function
- a date
- *undefined*

JSON Strings

Strings in JSON must be written in double quotes.

Example

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

Example

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

JSON Objects

Values in JSON can be objects.

Example

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

Objects as values in JSON must follow the JSON syntax.

JSON Arrays

Values in JSON can be arrays.

Example

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

Example

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

JSON null

Values in JSON can be null.

Example

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

Previous Log in to track progress Next

JSON Object Literals

❮ Previous Next ❯

This is a JSON string:

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

Inside the JSON string there is a JSON object literal:

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

JSON object literals are surrounded by curly braces {}.

JSON object literals contains key/value pairs.

Keys and values are separated by a colon.

Keys must be strings, and values must be a valid JSON data type:

- string
- number
- object
- array
- boolean
- null

Each key/value pair is separated by a comma.

It is a common mistake to call a JSON object literal "a JSON object".

JavaScript Objects

You can create a JavaScript object from a JSON object literal:

Example

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

Try it Yourself »

Normally, you create a JavaScript object by parsing a JSON string:

Example

```
myJSON = '{"name":"John", "age":30, "car":null}';
myObj = JSON.parse(myJSON);
```

Try it Yourself »

Accessing Object Values

You can access object values by using dot (.) notation:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
x = myObj.name;
```

You can also access object values by using bracket ([]) notation:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
x = myObj["name"];
```

Try it Yourself »

Looping an Object

You can loop through object properties with a for-in loop:

Example

```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);

let text = "";
for (const x in myObj) {
  text += x + ", ";
}
```

Try it Yourself »

In a for-in loop, use the bracket notation to access the property *values*:

Example

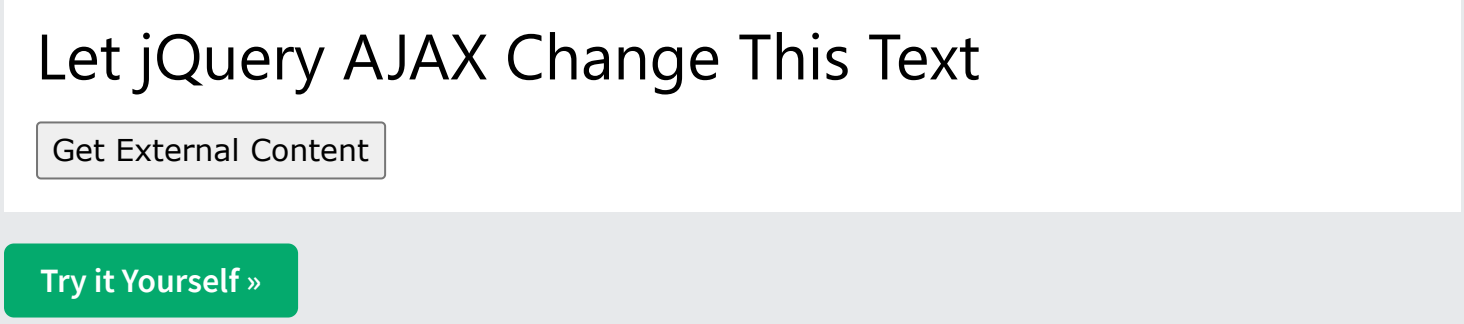
```
const myJSON = '{"name":"John", "age":30, "car":null}';
const myObj = JSON.parse(myJSON);
```

jQuery - AJAX Introduction

❮ Previous Next ❯

AJAX is the art of exchanging data with a server, and updating parts of a web page - without reloading the whole page.

jQuery AJAX Example



Try it Yourself »

What is AJAX?

AJAX = Asynchronous JavaScript and XML.

In short; AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.

Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.

You can learn more about AJAX in our AJAX tutorial.

What About jQuery and AJAX?

jQuery provides several methods for AJAX functionality.

Without jQuery, AJAX coding can be a bit tricky!

Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implementation. This means that you will have to write extra code to test for different browsers. However, the jQuery team has taken care of this for us, so that we can write AJAX functionality with only one single line of code.

jQuery AJAX Methods

In the next chapters we will look at the most important jQuery AJAX methods.

❮ Previous Log in to track progress Next ❯

jQuery - AJAX load() Method

‹ Previous Next ›

jQuery load() Method

The jQuery `load()` method is a simple, but powerful AJAX method.

The `load()` method loads data from a server and puts the returned data into the selected element.

Syntax:

```
$(selector).load(URL,data,callback);
```

The required URL parameter specifies the URL you wish to load.

The optional data parameter specifies a set of querystring key/value pairs to send along with the request.

The optional callback parameter is the name of a function to be executed after the `load()` method is completed.

Here is the content of our example file: "demo_test.txt":

```
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a paragraph.</p>
```

The following example loads the content of the file "demo_test.txt" into a specific `<div>` element:

Example

```
$("#div1").load("demo_test.txt");
```

It is also possible to add a jQuery selector to the URL parameter.

The following example loads the content of the element with id="p1", inside the file "demo_test.txt", into a specific `<div>` element:

Example

```
$("#div1").load("demo_test.txt #p1");
```

Try it Yourself »

The optional callback parameter specifies a callback function to run when the `load()` method is completed. The callback function can have different parameters:

- `responseTxt` - contains the resulting content if the call succeeds
- `statusTxt` - contains the status of the call
- `xhr` - contains the XMLHttpRequest object

The following example displays an alert box after the load() method completes. If the `load()` method has succeeded, it displays "External content loaded successfully!", and if it fails it displays an error message:

Example

```
$("button").click(function(){
  $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
    if(statusTxt == "success")
      alert("External content loaded successfully!");
    if(statusTxt == "error")
      alert("Error: " + xhr.status + ": " + xhr.statusText);
  });
});
```

Try it Yourself »

jQuery AJAX Reference

jQuery - AJAX get() and post() Methods

❮ Previous Next ❯

The jQuery get() and post() methods are used to request data from the server with an HTTP GET or POST request.

HTTP Request: GET vs. POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

GET is basically used for just getting (retrieving) some data from the server. **Note:** The GET method may return cached data.

POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

To learn more about GET and POST, and the differences between the two methods, please read our HTTP Methods GET vs POST chapter.

jQuery $.get() Method

The `$.get()` method requests data from the server with an HTTP GET request.

Syntax:

```
$.get(URL,callback);
```

The following example uses the `$.get()` method to retrieve data from a file on the server:

Example

```
$("button").click(function(){
  $.get("demo_test.asp", function(data, status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

Try it Yourself »

The first parameter of `$.get()` is the URL we wish to request ("demo_test.asp").

The second parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

Tip: Here is how the ASP file looks like ("demo_test.asp"):

```
<%
response.write("This is some text from an external ASP file.")
%>
```

jQuery $.post() Method

The `$.post()` method requests data from the server using an HTTP POST request.

Syntax:

```
$.post(URL,data,callback);
```

The required URL parameter specifies the URL you wish to request.

The optional data parameter specifies some data to send along with the request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

Example

```
$("button").click(function(){
  $.post("demo_test_post.asp",
  {
    name: "Donald Duck",
    city: "Duckburg"
  },
  function(data, status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

Try it Yourself »

The first parameter of `$.post()` is the URL we wish to request ("demo_test_post.asp").

Then we pass in some data to send along with the request (name and city).

The ASP script in "demo_test_post.asp" reads the parameters, processes them, and returns a result.

The third parameter is a callback function. The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

Tip: Here is how the ASP file looks like ("demo_test_post.asp"):

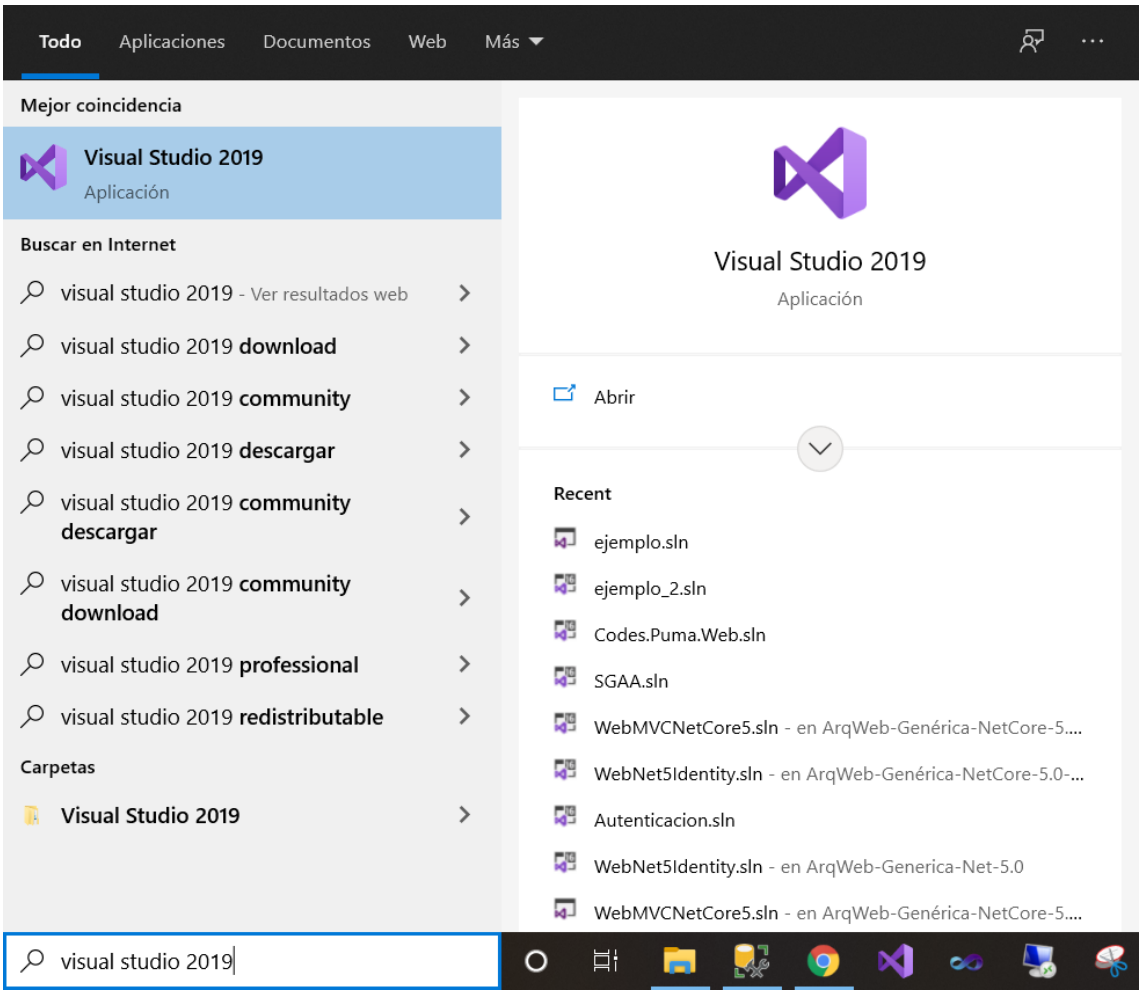
```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " & city & ".")
%>
```

jQuery AJAX Reference

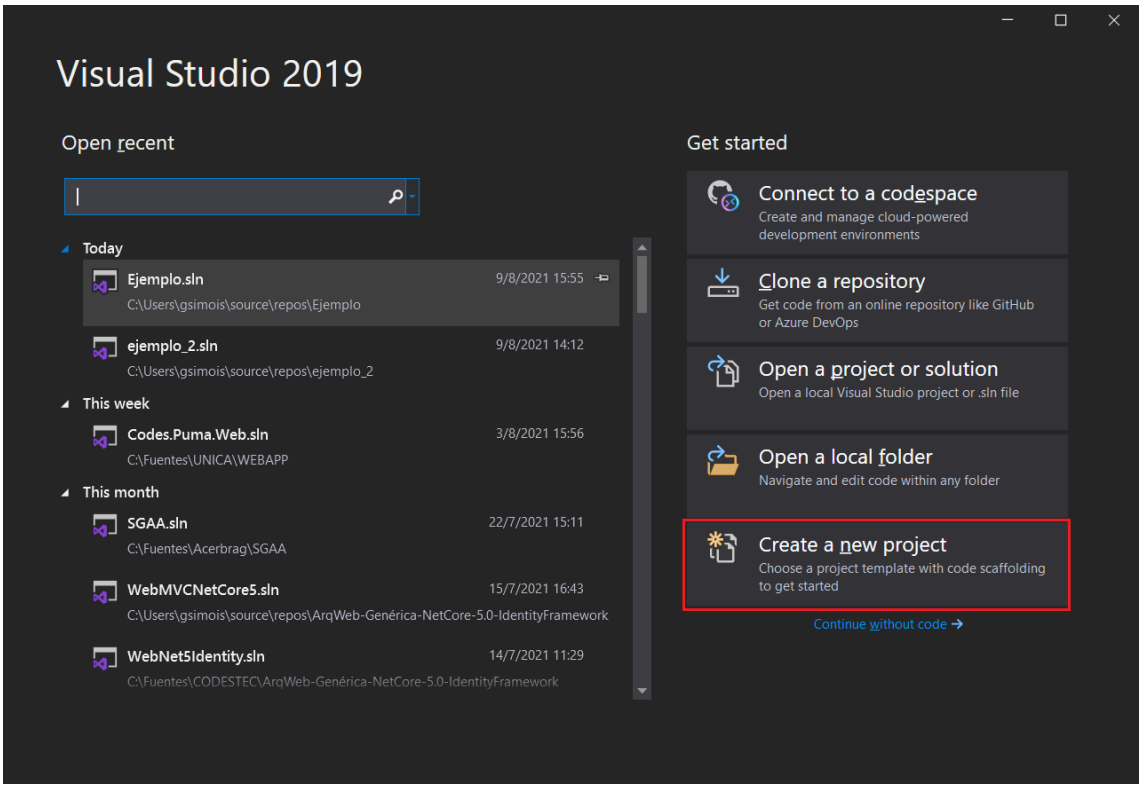
For a complete overview of all jQuery AJAX methods, please go to our jQuery AJAX Reference.

Creación de Proyecto Web de ASP.NET utilizando Visual Studio 2019

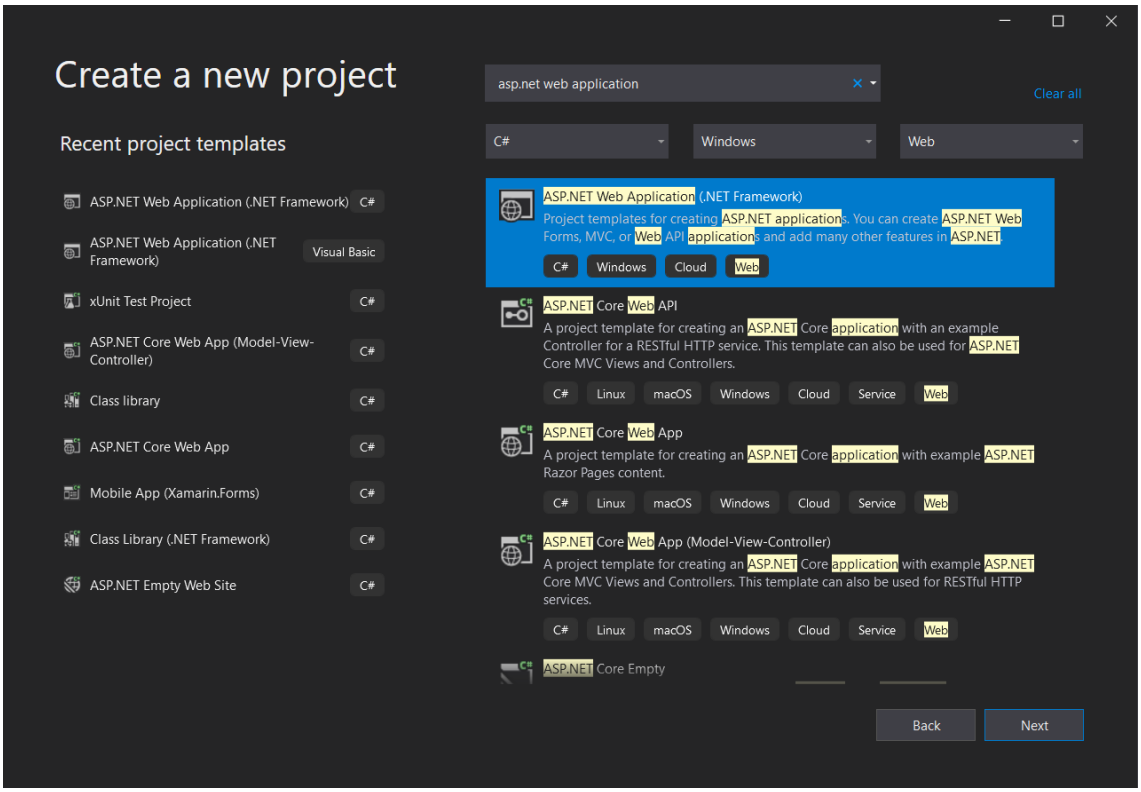
1) Desde la máquina virtual, Ejecutar Visual Studio 2019



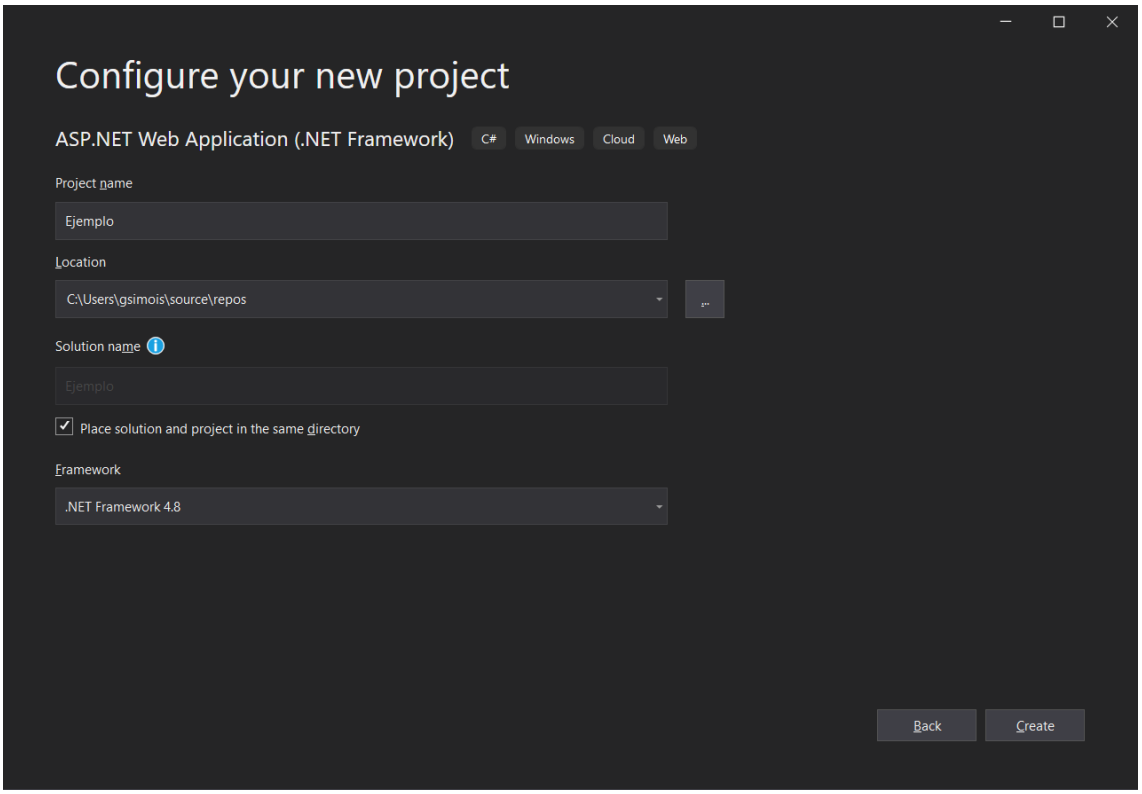
2) Luego, seleccionar la opción **Crear nuevo proyecto**



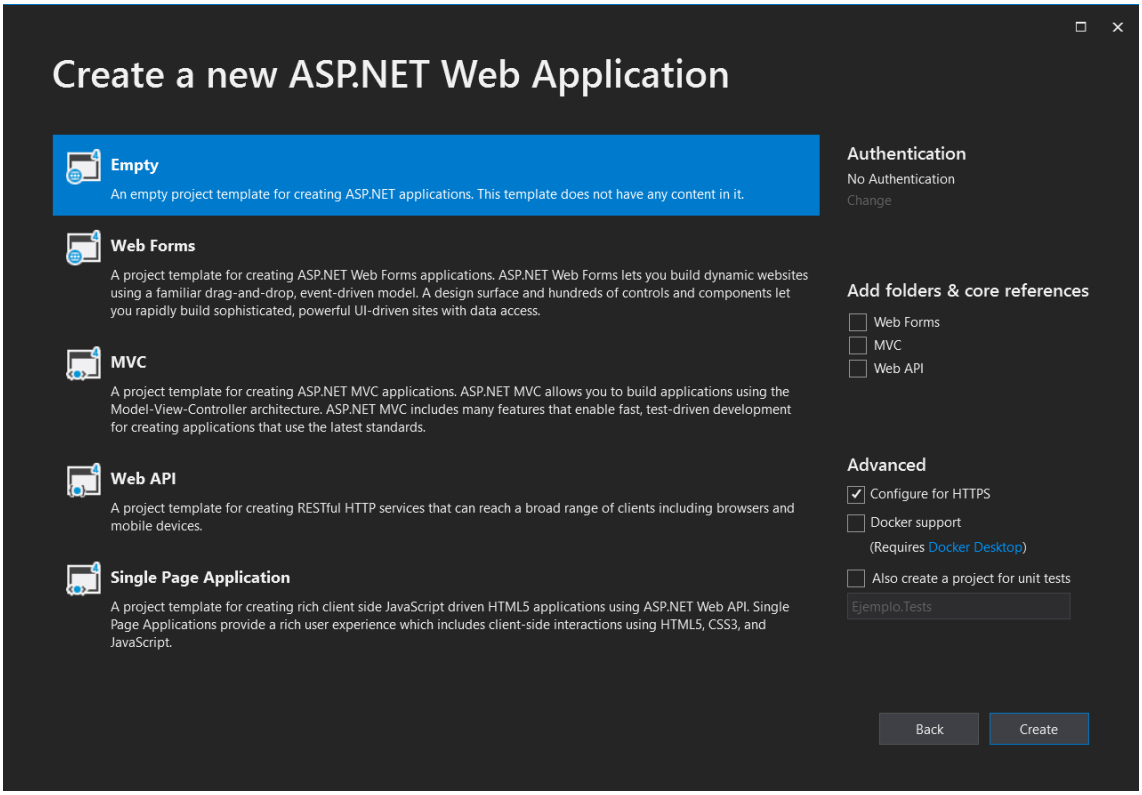
3) En el buscador de templates, escribir “Asp.Net web application”, seleccionar la opción **Asp.Net Web Application (.Net Framework)**, y luego hacer click en **Siguiente**.



4) Ingresar luego el nombre del proyecto y la ubicación de los archivos del proyecto:

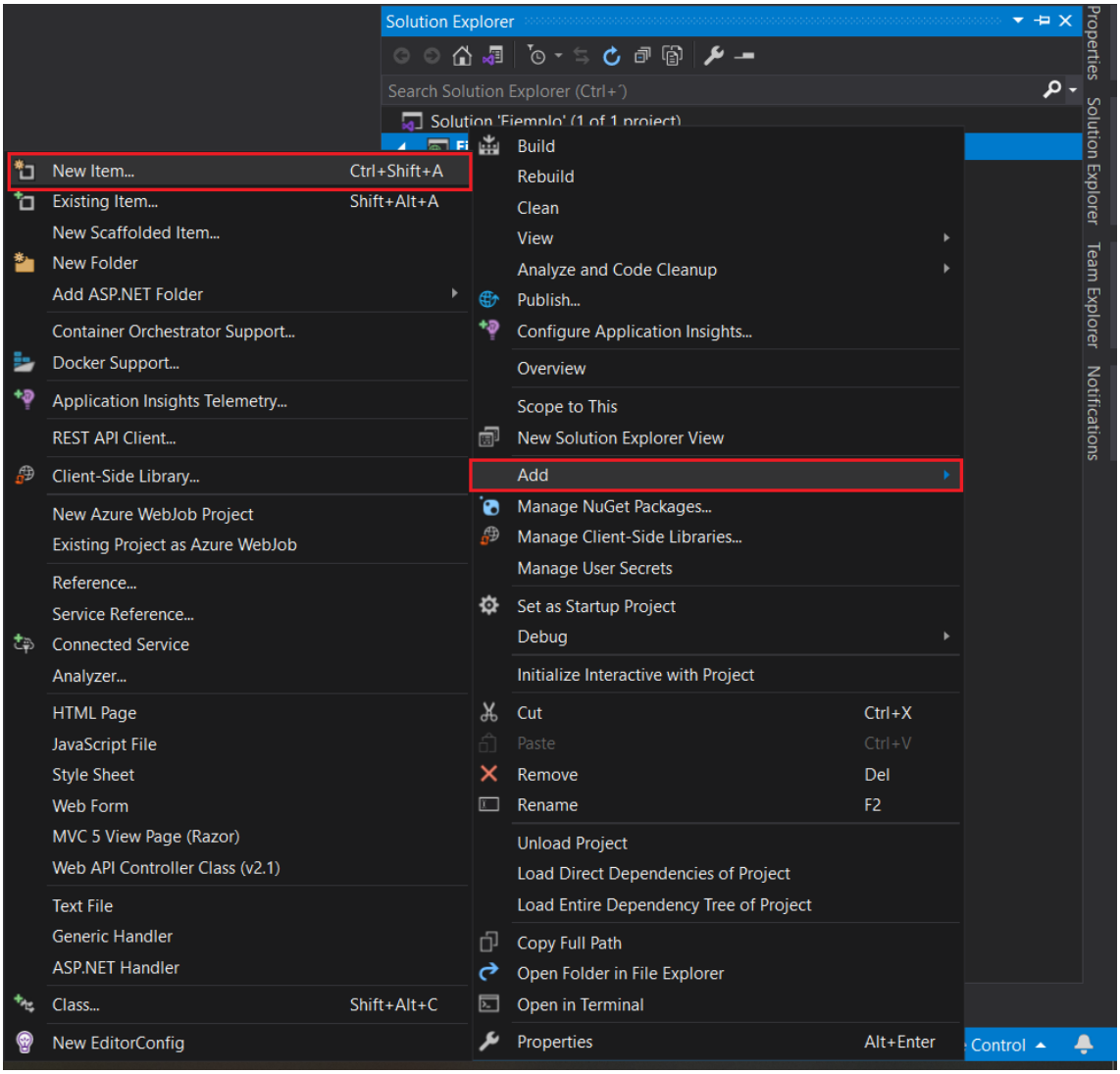


5) Seleccionar el template **Vacío**

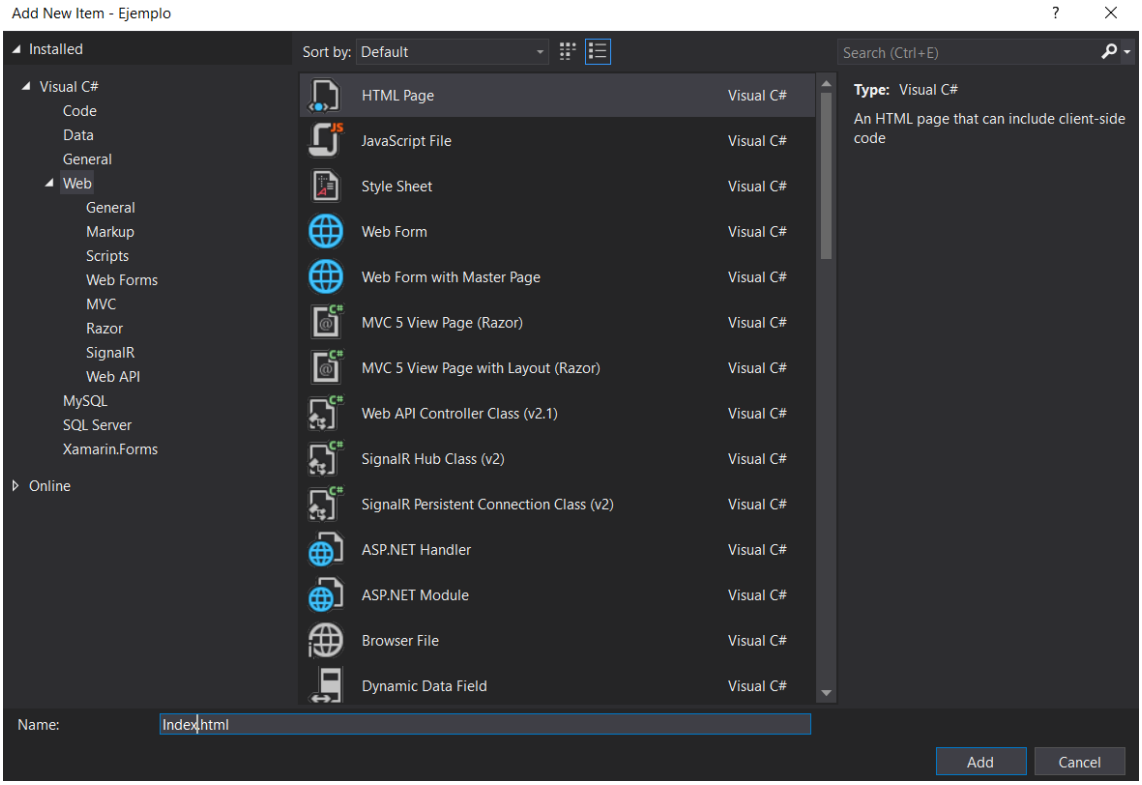


6) Para agregar un nuevo archivo al proyecto web (.txt, .html, .css, .js, .ashx), en el **Explorador de Soluciones** seleccionar el proyecto (en este caso "**Ejemplo"**) y luego

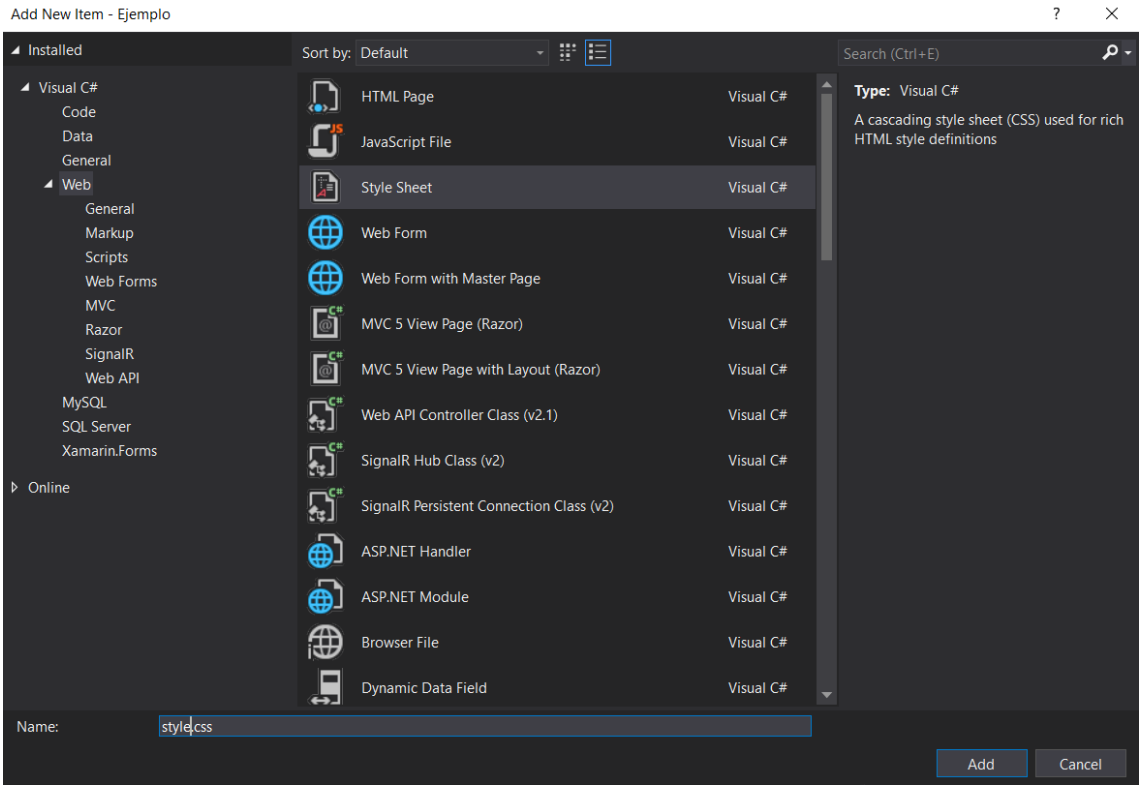
hacer **click derecho** sobre el mismo. Luego seleccionar la opción **Agregar**, **Nuevo Elemento**.



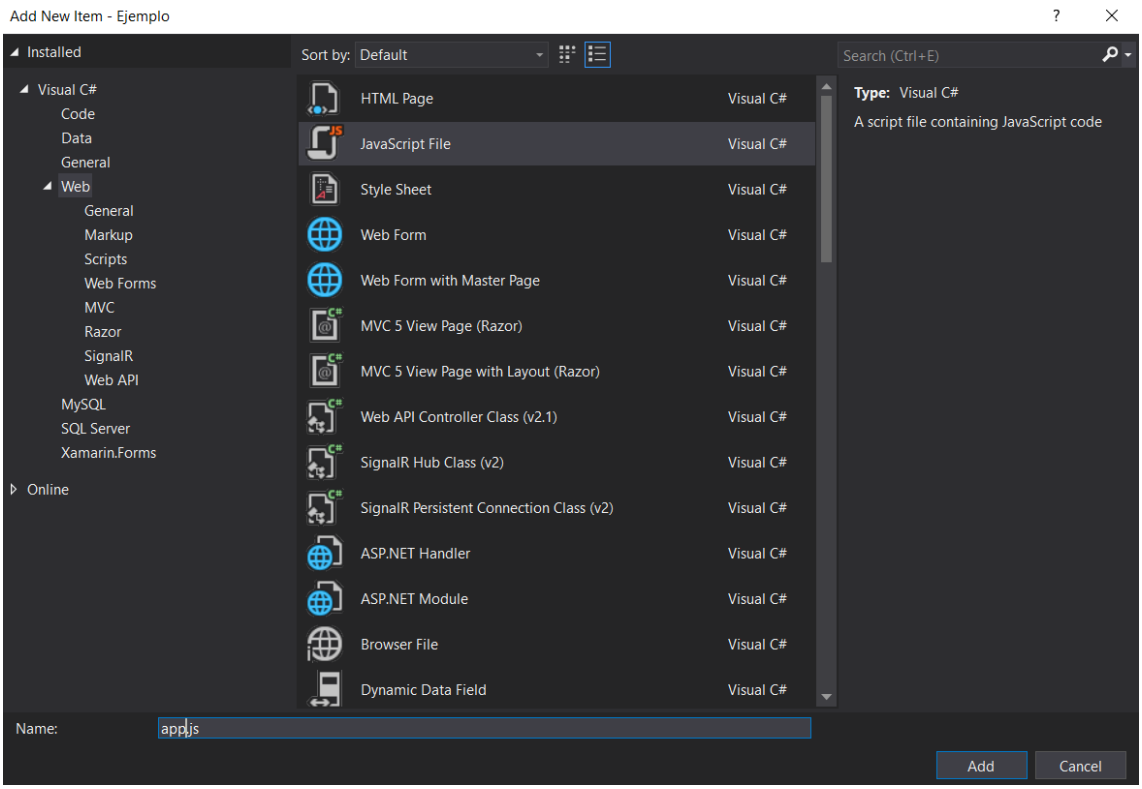
7) Agregar un nuevo HTML:



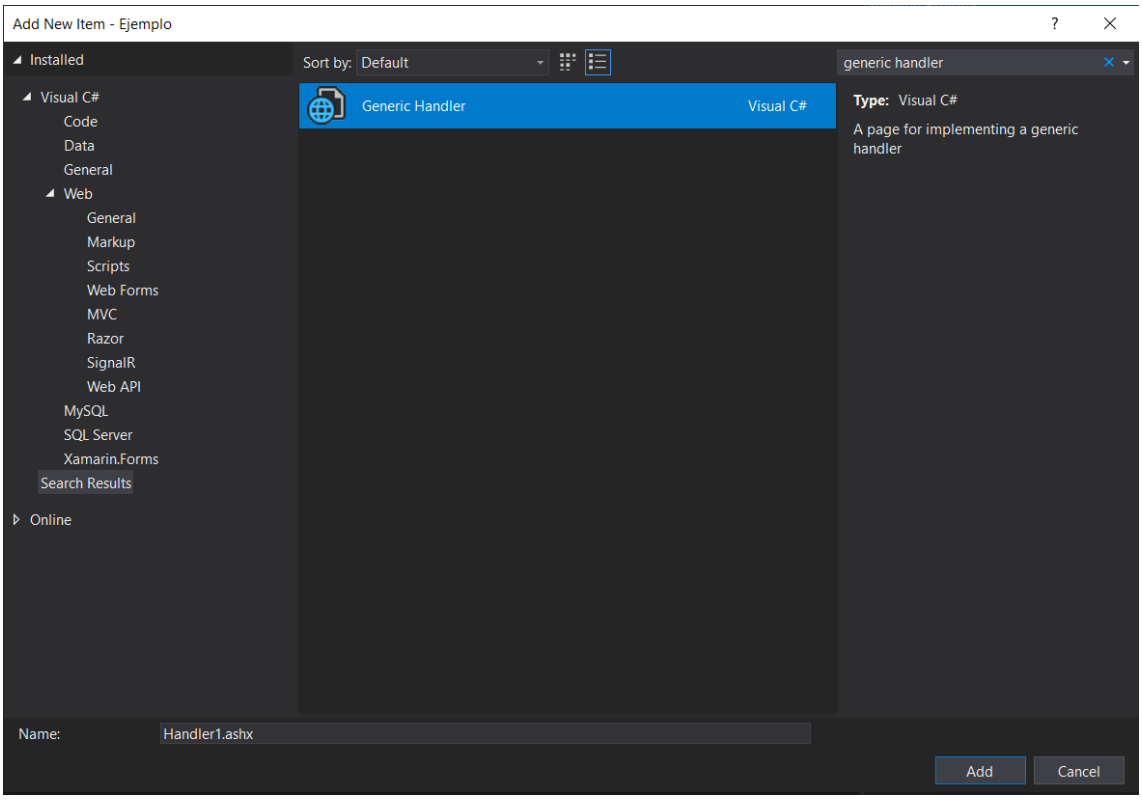
8) Agregar un nuevo archivo CSS:



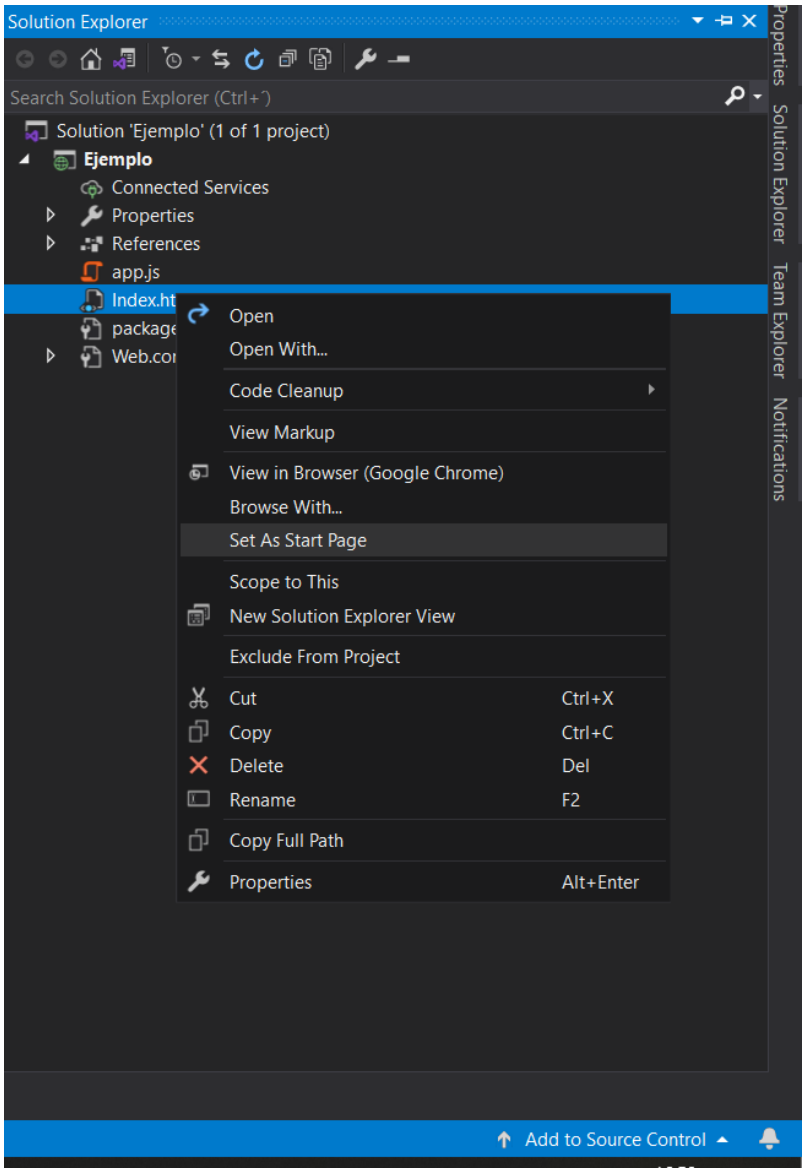
9) Agregar un nuevo archivo JS:



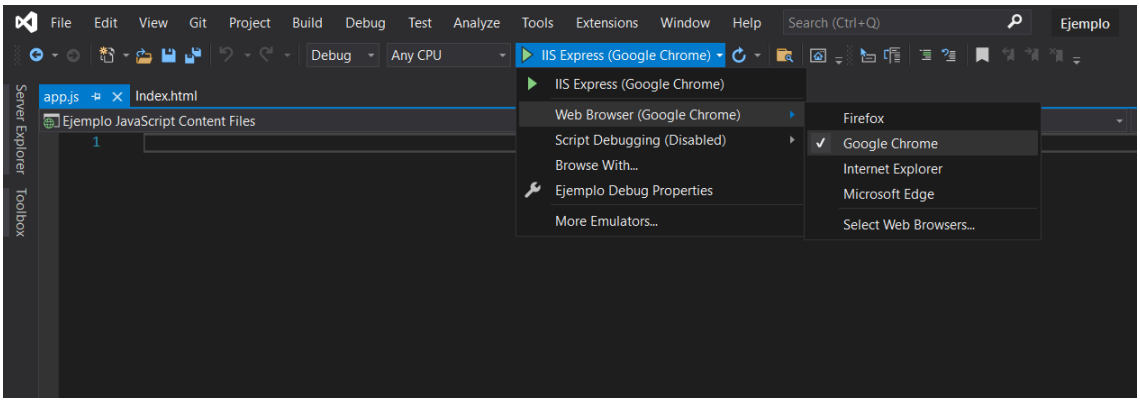
10) Agregar un nuevo **Handler ASP.NET (lenguaje C#)**:

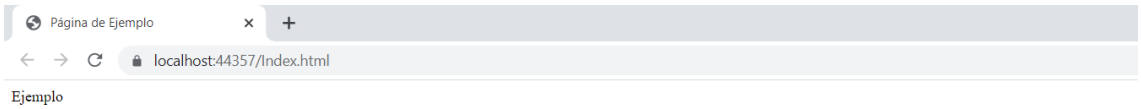


11) Establecer Index.html como página de Inicio: Seleccionar el archivo **index.html** (o el .html que corresponda), y seleccionar la opción **Establecer como Página de Inicio.**



12) Ejecutar la aplicación Web





Ejemplo