

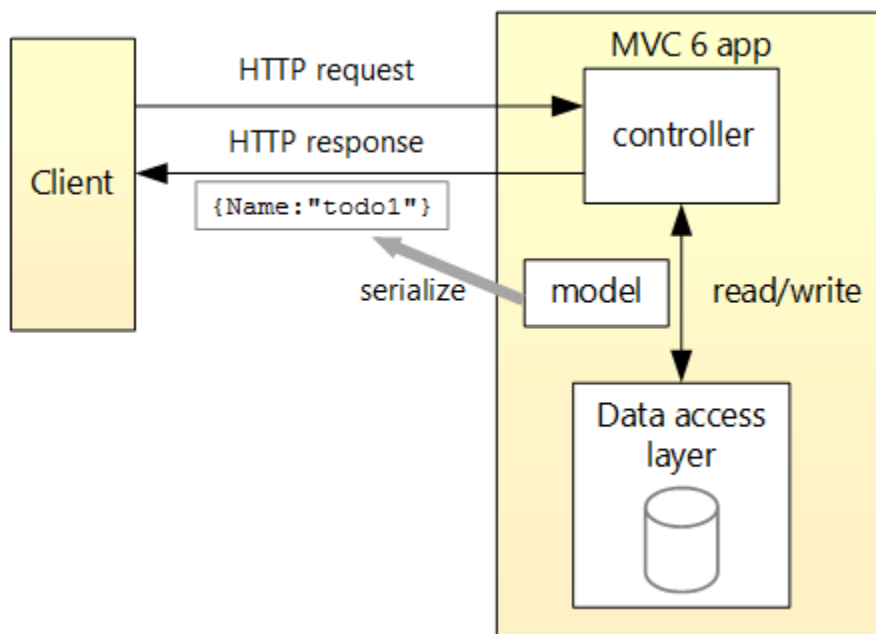
## Taller WEB API 2

Las versiones anteriores de ASP.NET incluyen el marco Web API para la creación de APIs web. En ASP.NET 5, esta funcionalidad se ha fusionado en el marco MVC 6. La unificación de los dos marcos hace que sea más fácil de construir aplicaciones que incluyen tanto la interfaz de usuario (HTML) y las API, porque ahora ellos comparten la misma base de código y pipeline.

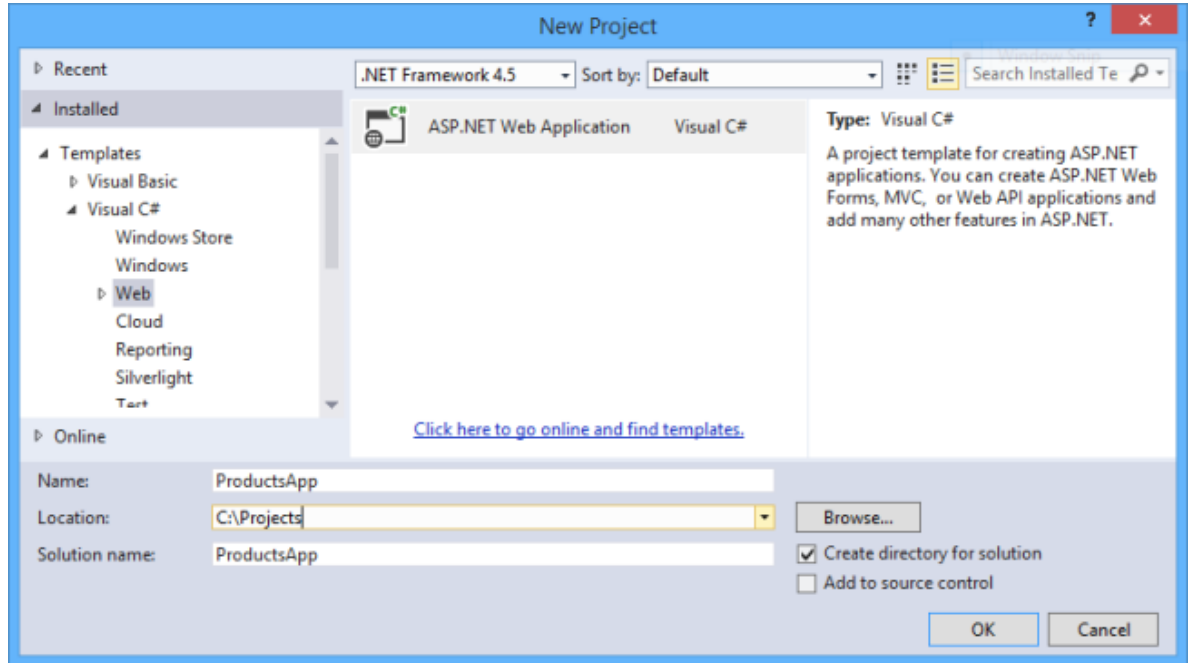
### Rutas en WEB API 2

API	Description	Request body	Response body
GET /api/todo	Get all to-do items	None	Array of to-do items
GET /api/todo/{id}	Get an item by ID	None	To-do item
POST /api/todo	Add a new item	To-do item	To-do item
PUT /api/todo/{id}	Update an existing item	To-do item	None
DELETE /api/todo/{id}	Delete an item.	None	None

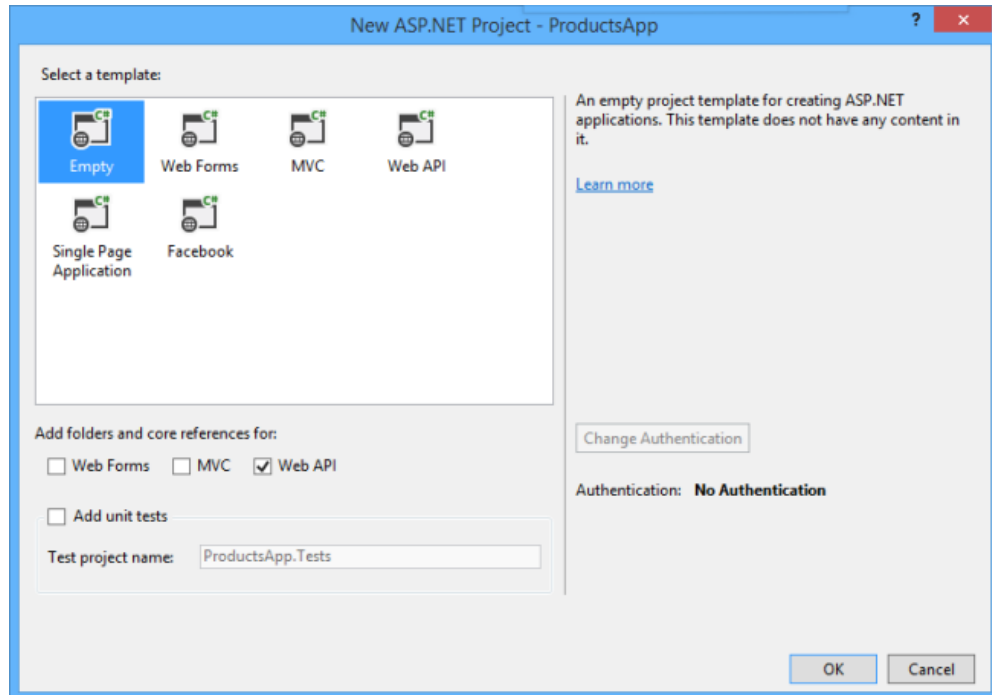
### Diagrama de WEB API 2



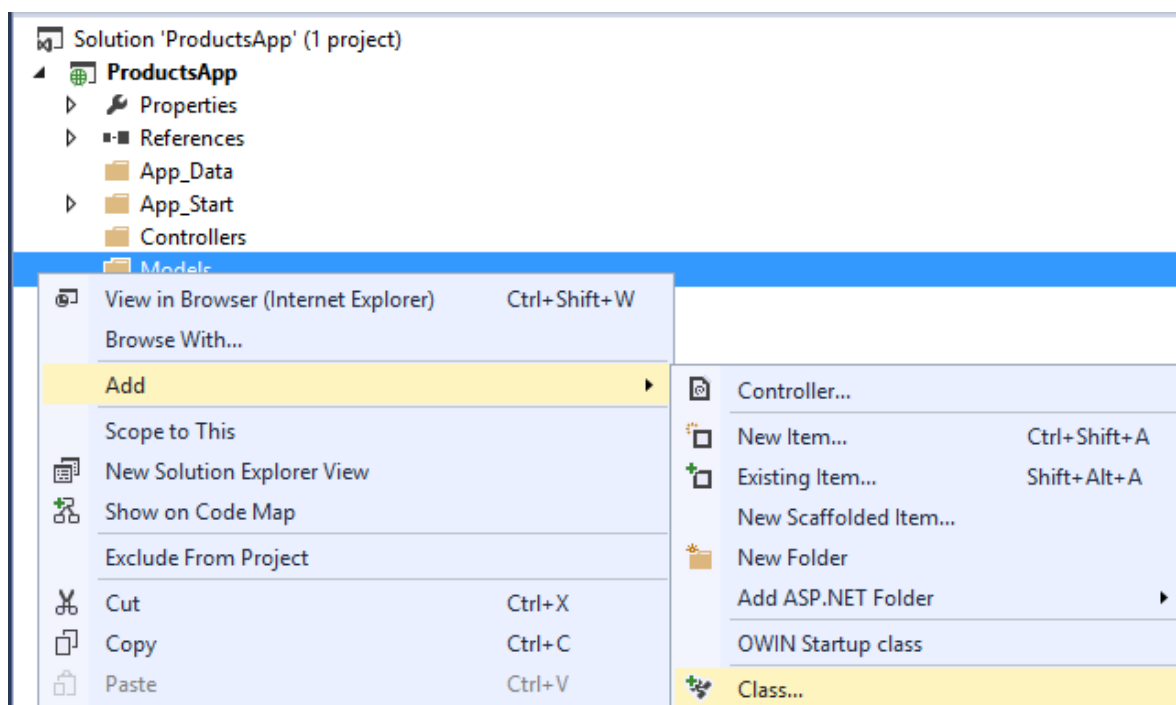
1. Se va a crear un proyecto en web api 2, con el nombre **ProductsApp**.



2. Se selecciona el proyecto Asp.net, con el tipo de proyecto **Web API**



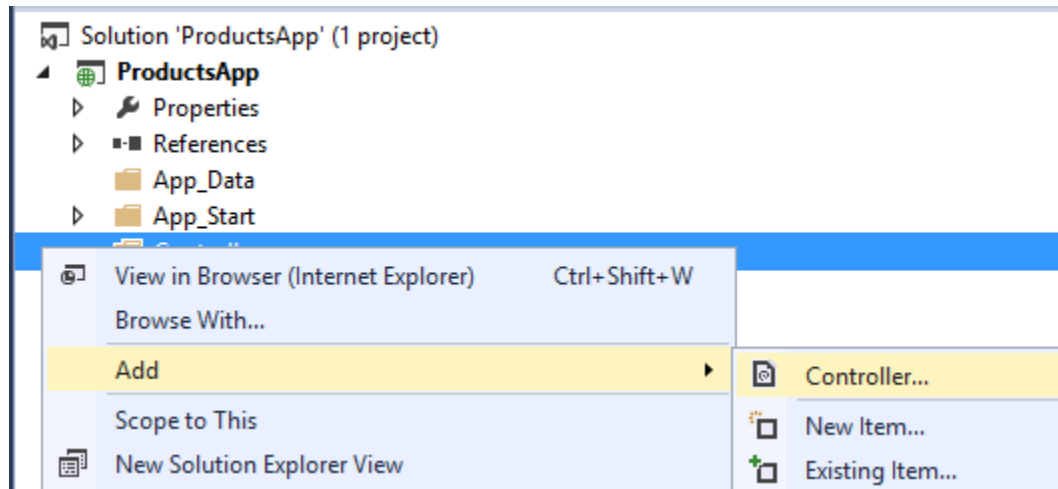
3. Ahora agregamos un modelo.



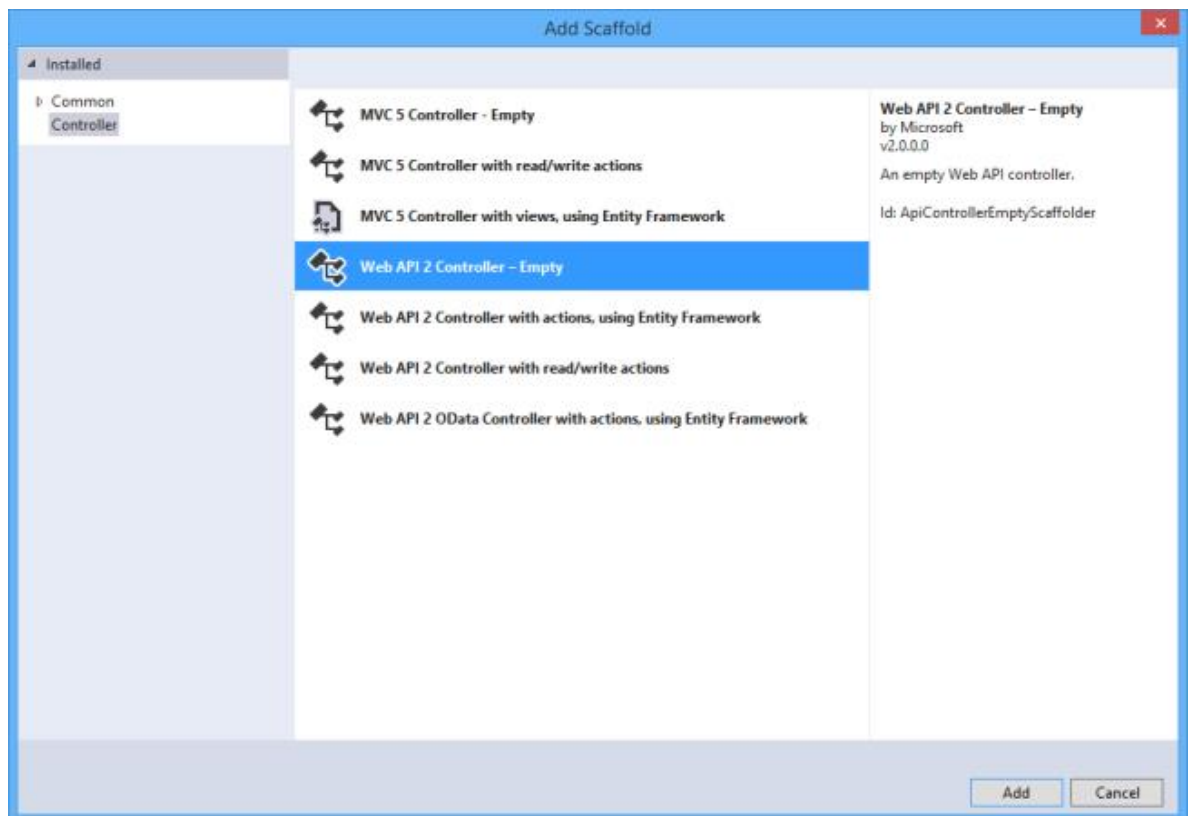
4. El contenido del modelo es el siguiente

```
namespace ProductsApp.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Category { get; set; }
        public decimal Price { get; set; }
    }
}
```

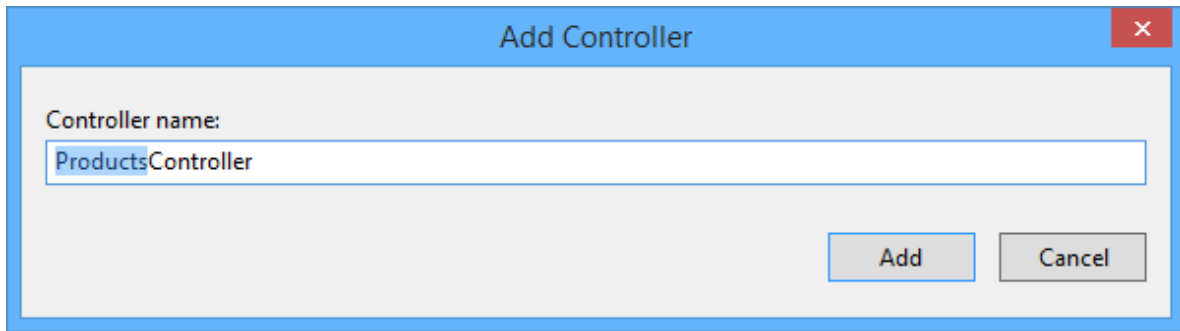
5. Agregamos un controller



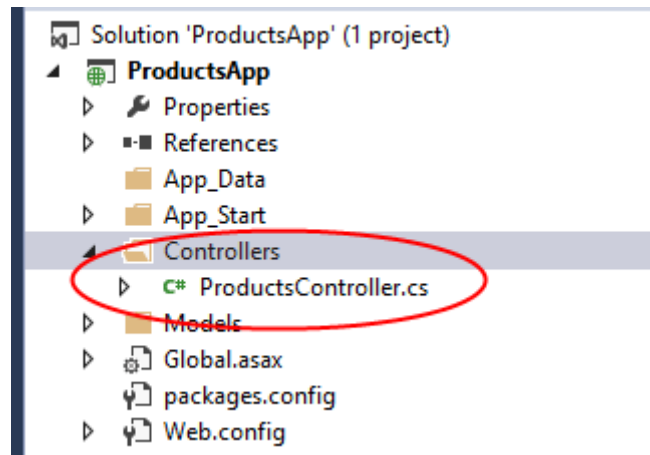
6. En este punto podemos seleccionar varios controladores, en caso de que exista un modelo creado en Entity Framework, pero vamos a seleccionar **Web API 2 Controller – Empty**



7. Agregamos un controlador con el nombre de **ProductsController**



8. Debería quedar el árbol del proyecto de la siguiente forma



9. El código del controlador debería quedar de la siguiente forma.

```
using ProductsApp.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Web.Http;

namespace ProductsApp.Controllers
{
    public class ProductsController : ApiController
    {
        Product[] products = new Product[]
        {
            new Product { Id = 1, Name = "Tomato Soup", Category = "Groceries", Price = 1 },
            new Product { Id = 2, Name = "Yo-yo", Category = "Toys", Price = 3.75M },
            new Product { Id = 3, Name = "Hammer", Category = "Hardware", Price = 16.99M }
        };

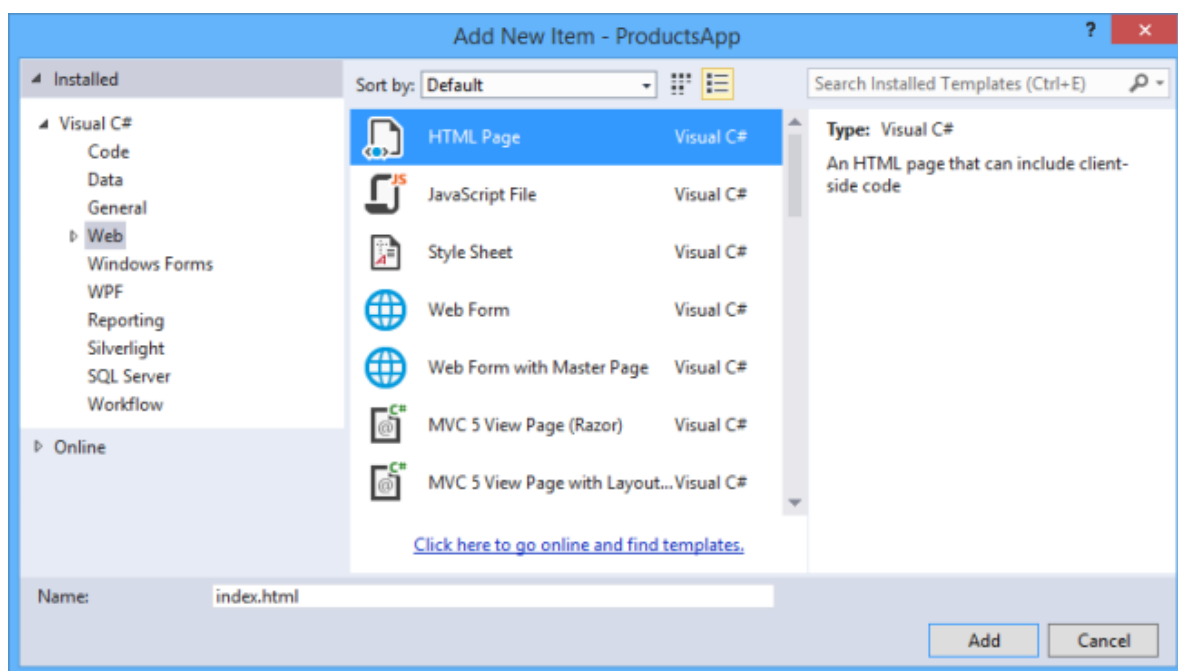
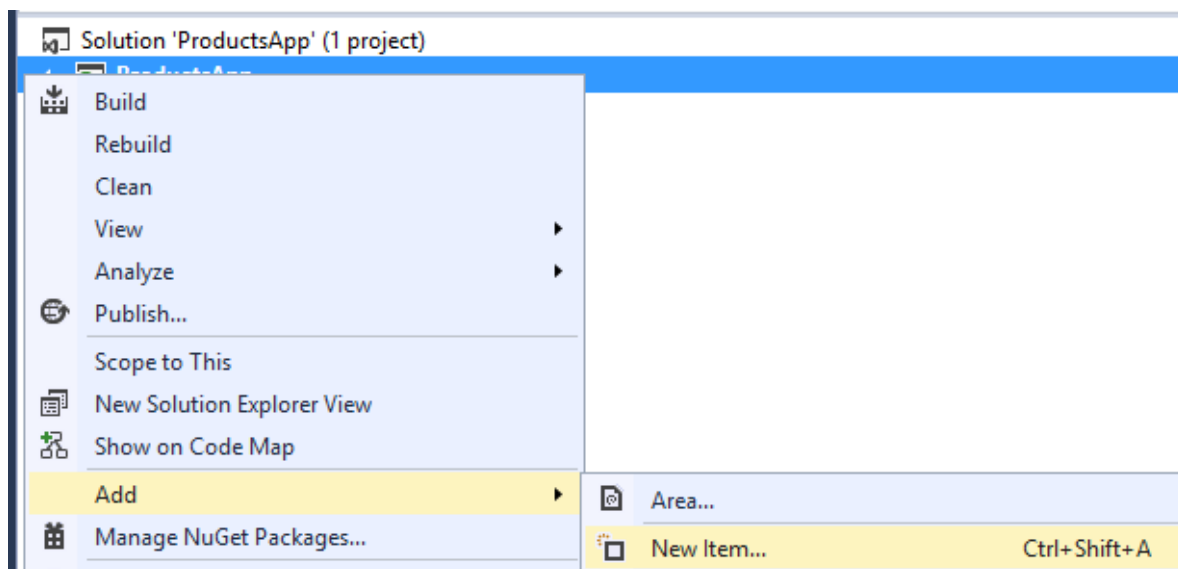
        public IEnumerable<Product> GetAllProducts()
        {
            return products;
        }

        public IHttpActionResult GetProduct(int id)
        {
            var product = products.FirstOrDefault((p) => p.Id == id);
            if (product == null)
            {
                return NotFound();
            }
            return Ok(product);
        }
    }
}
```

10. Ahora utilizaremos 2 rutas del proyecto.

Controller Method	URI
GetAllProducts	/api/products
GetProduct	/api/products/id

11. Vamos a crear una página de prueba en HTML para utilizar con jquery.



12. Así debe quedar la página de HTML.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Product App</title>
</head>
<body>

  <div>
    <h2>All Products</h2>
    <ul id="products" />
  </div>
  <div>
    <h2>Search by ID</h2>
    <input type="text" id="prodId" size="5" />
    <input type="button" value="Search" onclick="find();" />
    <p id="product" />
  </div>
```

13. Y como debe quedar el código de JQuery. N

**Nota:** Recuerde las rutas del controlador que se mostraron en el punto 10.

- Inicializamos con el document ready.
- Con la función getJSON, obtendremos por medio del controlador todos los productos.

```
$(document).ready(function () {
  // Send an AJAX request
  $.getJSON(apiUrl)
    .done(function (data) {
      // On success, 'data' contains a list of products.
      $.each(data, function (key, item) {
        // Add a list item for the product.
        $('<li>', { text: formatItem(item) }).appendTo($('#products'));
      });
    });
});
```

- El método find() vamos a realizar una búsqueda por id.

```
function find() {
  var id = $('#prodId').val();
  $.getJSON(apiUrl + '/' + id)
    .done(function (data) {
      $('#product').text(formatItem(data));
    })
    .fail(function (jqXHR, textStatus, err) {
      $('#product').text('Error: ' + err);
    });
}
```

14. Así debería quedar toda la página de HTML.



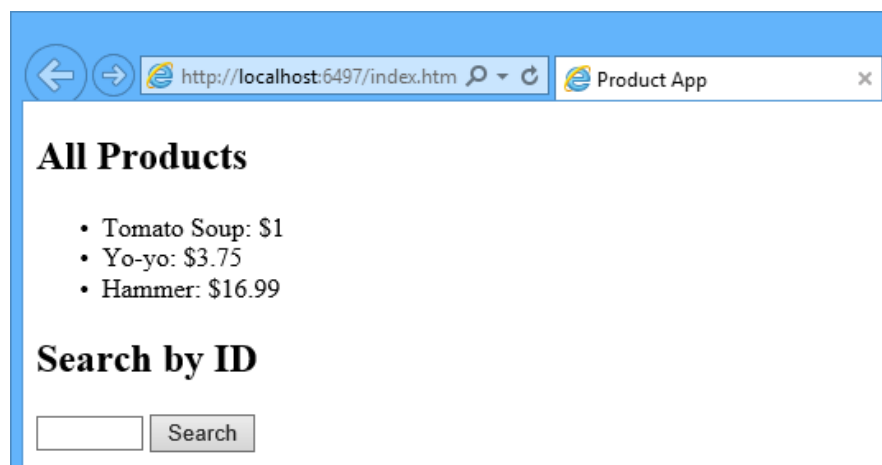
```
<script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-2.0.3.min.js"></script>
<script>
    var uri = 'api/products';

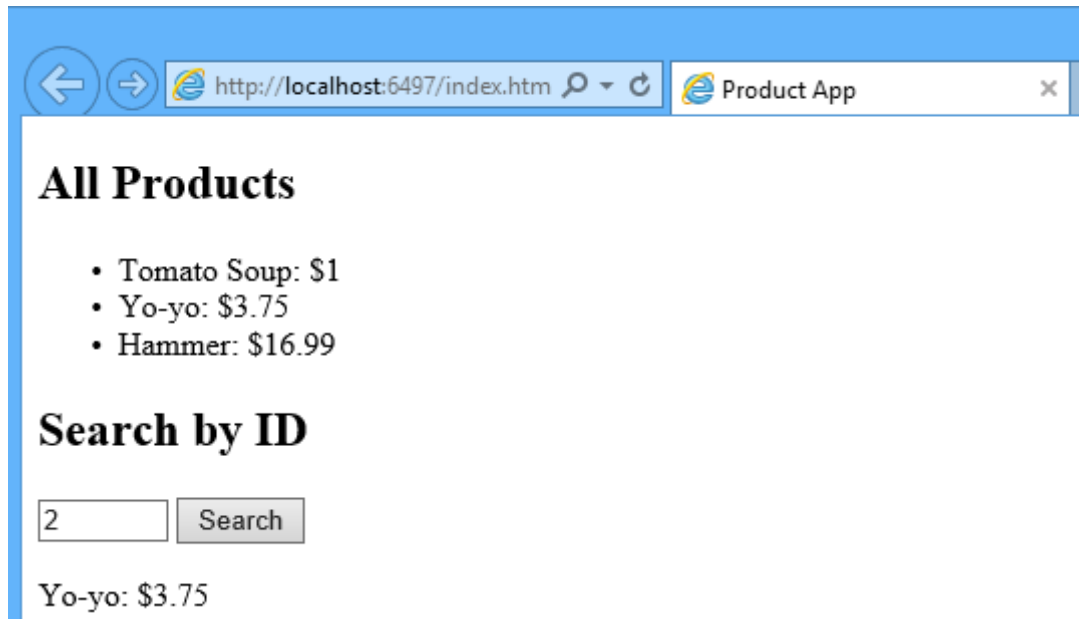
    $(document).ready(function () {
        // Send an AJAX request
        $.getJSON(uri)
            .done(function (data) {
                // On success, 'data' contains a list of products.
                $.each(data, function (key, item) {
                    // Add a list item for the product.
                    $('#li>', { text: formatItem(item) }).appendTo($('#products'));
                });
            });

        function formatItem(item) {
            return item.Name + ': $' + item.Price;
        }

        function find() {
            var id = $('#prodId').val();
            $.getJSON(uri + '/' + id)
                .done(function (data) {
                    $('#product').text(formatItem(data));
                })
                .fail(function (jqXHR, textStatus, err) {
                    $('#product').text('Error: ' + err);
                });
        }
    })
</script>
</body>
</html>
```

15. De esta forma debería quedar la aplicación.



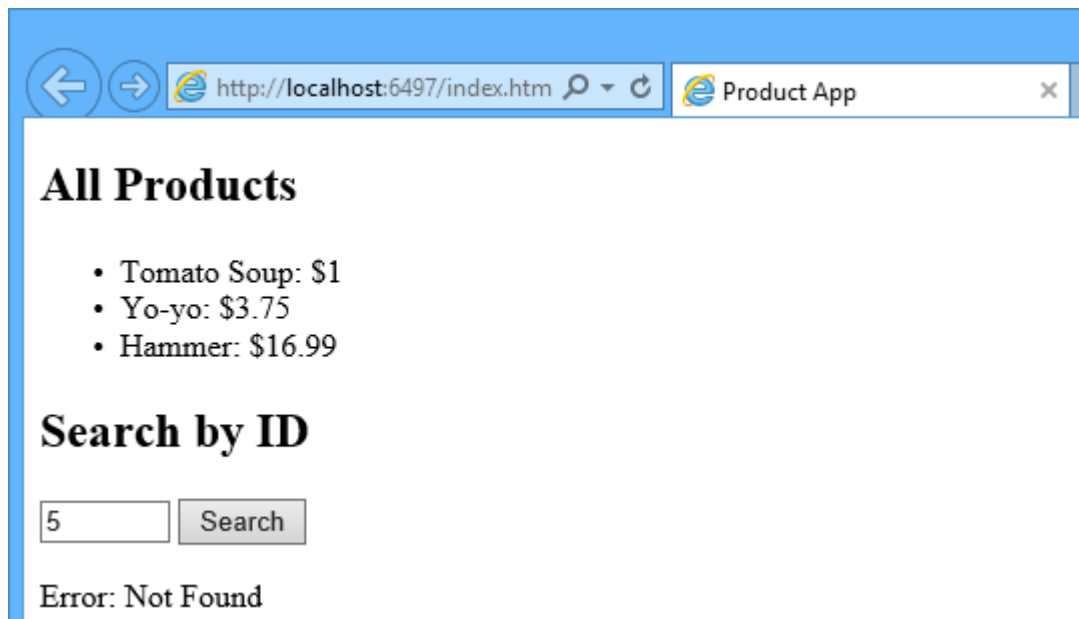


**All Products**

- Tomato Soup: \$1
- Yo-yo: \$3.75
- Hammer: \$16.99

**Search by ID**

Yo-yo: \$3.75



**All Products**

- Tomato Soup: \$1
- Yo-yo: \$3.75
- Hammer: \$16.99

**Search by ID**

Error: Not Found

## Fuente

<http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>

<http://aspnetmvc.readthedocs.io/projects/mvc/en/latest/getting-started/first-web-api.html>

<https://msdn.microsoft.com/es-es/library/dn595062.aspx>