ASP.NET Web API CRUD Operations

CRUD stands for "Create, Read, Update, and Delete," which are the four basic database operations. Many HTTP services also model CRUD operations through REST or REST-like APIs. In this tutorial, you will build a very simple web API to manage a list of products. Each product will contain a name, price, and category (such as "toys" or "hardware"), plus a product ID.

The products web API will expose following methods.

|  |  |  |
| --- | --- | --- |
| **Action** | **HTTP method** | **Relative URI** |
| Get a list of all products | GET | /api/products |
| Get a product by ID | GET | /api/products/id |
| Get a product by category | GET | /api/products?category=category |
| Create a new product | POST | /api/products |
| Update a product | PUT | /api/products/id |
| Delete a product | DELETE | /api/products/id |

Notice that some of the URIs include the product ID in path. For example, to get the product whose ID is 28, the client sends a GET request for <http://hostname/api/products/28>.

### Resources

The products API defines URIs for two resource types:

|  |  |
| --- | --- |
| **Resource** | **URI** |
| The list of all the products. | /api/products |
| An individual product. | /api/products/id |

### Methods

The four main HTTP methods (GET, PUT, POST, and DELTETE) can be mapped to CRUD operations as follows:

* GET retrieves the representation of the resource at a specified URI. GET should have no side effects on the server.
* PUT updates a resource at a specified URI. PUT can also be used to create a new resource at a specified URI, if the server allows clients to specify new URIs. For this tutorial, the API will not support creation through PUT.
* POST creates a new resource. The server assigns the URI for the new object and returns this URI as part of the response message.
* DELETE deletes a resource at a specified URI.

Note: The PUT method replaces the entire product entity. That is, the client is expected to send a complete representation of the updated product. If you want to support partial updates, the PATCH method is preferred. This tutorial does not implement PATCH.

## Create a New Web API Project

Start by running Visual Studio 2010 and select **New Project** from the **Start** page. Or, from the **File** menu, select **New** and then **Project**.

In the **Templates** pane, select **Installed Templates** and expand the **Visual C#** node. Under **Visual C#**, select **Web**. In the list of project templates, select **ASP.NET MVC 4 Web Application**. Name the project "ProductStore" and click **OK**.

In the **New ASP.NET MVC 4 Project** dialog, select **Web API** and click **OK**.

## Adding a Model

A model is an object that represents the data in your application. In ASP.NET Web API, you can use strongly typed CLR objects as models, and they will automatically be serialized to XML or JSON for the client.

For the ProductStore API, our data consists of products, so we'll create a new class named Product.

If Solution Explorer is not already visible, click the **View** menu and select **Solution Explorer**. In Solution Explorer, right-click the **Models** folder. From the context meny, select **Add**, then select **Class**. Name the class "Product".