



Logols Learning

WEEKEND WEB DEVELOPMENT BOOT CAMP

TRAINING: WEB API

REST

- ▶ Representational state transfer
- ▶ REST is protocol independent
- ▶ Architectural Style
 - ▶ Client-Server Communication
 - ▶ Stateless
 - ▶ Cacheable
 - ▶ Uniform Interface
 - ▶ Addressable Resources

- REST is an architectural style that is protocol independent.
- It requires client server communication similar to how a browser interacts with a server.
- It is stateless, meaning that one request doesn't know anything about another.
- It needs to be cacheable meaning the client could cache results if they wanted to.
- And it needs to have a uniform interface.
- <http://carminedimascio.com/2013/09/restful-design-principles/>

HTTP

- ▶ Many REST services now use HTTP
- ▶ This is Hypertext Transfer Protocol
- ▶ What web pages use
- ▶ Request and Response
- ▶ Stateless
- ▶ Requests: GET, POST, PUT, DELETE

- REST is often implemented over HTTP and uses the GET, POST, PUT, DELETE structure for a uniform interface.
- HTTP is what all web requests and responses use.
- It is stateless
- So http works well with REST principles.

Controller

- ▶ The request initially goes to the controller
- ▶ Based upon the route a method is run.
- ▶ The JSON body of the request can be bound to a model.
- ▶ Whatever is returned from the method is returned in JSON.

- The request initially goes to the controller.
- Based upon the route and the request type attribute a method is run.
- The JSON body of the request can be bound to a model.
- Whatever is returned from the method is returned as JSON in the response body.

Attributes

- ▶ Allow for declarations on classes, methods, or properties.
- ▶ Give special behavior or properties.
- ▶ [HttpGet], [HttpPost], [HttpPut], [HttpDelete]
- ▶ Example:

```
[HttpPost]
public void Post([FromBody]Answer answer)
{
    _service.Insert(answer);
}
```

- Attributes allow for declarations on classes, methods, and properties.
- They give special behavior or properties to those classes, methods, or properties.
- They are specified above a class, method, or property definition with the name of the attribute between square brackets [].
- Parameters can be passed to attributes using parentheses ().
- For WebApi we are using attributes to specify the http request type.
- The example shows an HttpPost attribute used to designate a post request.

Routing

- ▶ Tells ASP.Net which controller should receive the request
- ▶ Based on the pattern of the request
- ▶ Can be customized
- ▶ Default Route: "{controller}/{action}/{id}"

- As I mentioned, every request is routed to a controller.
- You can control how the routing works.
- There is a default route which is controller/action/id.
- This is in reference to how the url looks, the controller is specified and then the action and then the id.
- You can create custom routes to change which requests go to which controller.
- Example: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/adding-a-controller>

JSON

- ▶ JavaScript Object Notation
- ▶ How an Object is represented in JavaScript.

Example:

```
{  
  'studentID': 0  
  'firstName': 'Kathy',  
  'lastName': 'Smith'  
}
```

- JSON stands for JavaScript Object Notation.
- It is how an object can be created on the file and is represented in JavaScript.
- Essentially an object is surrounded by curly braces {}
- Inside each property or variable is named followed by a colon : followed by the value.
- There can be objects within objects or arrays within objects as well.

Binding

- ▶ Return value to JSON Data
- ▶ JSON in Request Body converted to Class
- ▶ Example:


```
[HttpPost]
public void Post([FromBody]Answer answer)
{
    _service.Insert(answer);
}
```

- The return value of an action method in a controller is converted to JSON in the response body.
- JSON that exists within the request body can also be converted or bound or serialized into a C# class.
- This automatic binding makes it easier to convert between JSON and C#.

CLI Commands

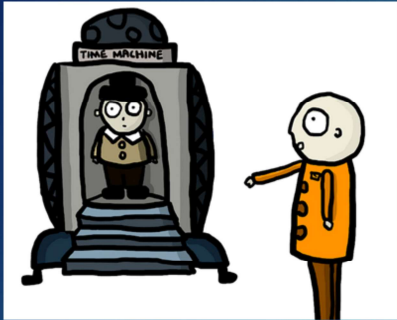
- ▶ mkdir – Create Directory
- ▶ cd – Change Directory
- ▶ Add project:
 - ▶ dotnet new classlib
 - ▶ dotnet new webapi
- ▶ Add reference:
 - ▶ dotnet add reference [path]/[name.csproj]
 - ▶ dotnet add package Dapper
 - ▶ dotnet add package MySql.Data

- In order to make projects for our main components we will need to use CLI commands.
- mkdir creates a new directory on your computer.
- cd changes the directory that you are currently in.
- dotnet new classlib – creates a new class library project. This is what we use to create a new .Net project that is code only.
- dotnet new webapi – creates a new web api project. This is what we use to create a new .Net project that will contain our web api.
- dotnet add reference – allows us to reference code from another project. One project does not know about the other until a reference is added.
- dotnet add package – allows us to add a package to our code. A package is created by someone and can be downloaded over the internet for use in your project.
- We will be using two packages from the Nuget package manager which are Dapper and MySql.Data.



Postman

- Let's take a tour of Postman
 - Open Postman
 - Using the correct URL, make a get request to the API to get data back.
 - Using the correct URL and request body, make a post request to post data.
 - Using the correct URL and request body, make a put request to update data.
 - Using the correct URL, make a delete request to the API to delete data.

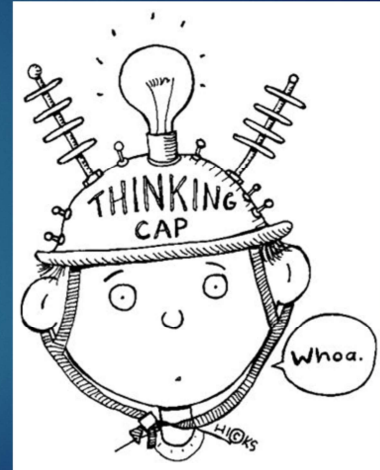


EXAMPLE

ASP.NET WEB API

ASSESSMENT

ASP.NET WEB API



- Name four http request types.
- Write on the board JSON for a pet object specifying name and pet type.
- Write on the board a controller action definition for a post to insert a pet using the JSON definition.
- Write on the board a controller action definition for a post to update a pet using the JSON definition.
- Write on the board a controller action definition for a get to one pet in the system based upon id.

Assignment

- ▶ A small internet has been brought back online. Everyone wants your status report.
- ▶ They also want to be able to insert, update, and delete data.
- ▶ Add a web api to get, insert, update, and delete person status.



QUICK REVIEW

WEB API



- Name four http request types.
- Write on the board JSON for a pet object specifying name and pet type.
- Write on the board a controller action definition for a post to insert a pet using the JSON definition.
- Write on the board a controller action definition for a post to update a pet using the JSON definition.
- Write on the board a controller action definition for a get to one pet in the system based upon id.

Additional Resources

- ▶ Microsoft Page

- ▶ <https://www.asp.net/web-api>

- ▶ Microsoft Tutorial

- ▶ <https://docs.microsoft.com/en-us/aspnet/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api>

- ▶ PluralSight

- ▶ <https://app.pluralsight.com/player?author=jon-flanders&name=aspnetwebapi-m1-introduction&mode=live&clip=0&course=aspnetwebapi>

Keep Practicing!

- ▶ Try creating different web api's.
- ▶ Use the different http actions.
- ▶ Hook it up to the database.