Using APIs breakout

in list Breakouts

Description

Add a more detailed description...

API

Delete

0%

- [ ] 1) Go to your API's website
- [ ] 2) Find a "Getting Started" guide
- [ ] 3) It should give you a URL to use
- [ ] 4) Note that URL
- [ ] 5) Make note of any indexes in the data and how they correlate with each other. IE in the weather API Temp label indexes match the rest of the lists.
- [ ] NOTE: Every API is going to be a little different: the tags will be organized different, you might be required to sign up or apply for credentials. When calling APIs be flexible. Every API is unique.

Add an item

Visual Studio

Delete

0%

- [ ] NOTE: Throughout this process there will be a few different using statements needed, use Visual Studio's auto correct feature to pull them in.
- [ ] 0) Can't find a URL? Use this one as a test: "https://swapi.co/api/people/1/"
- [ ] 1) Create an MVC App, using the MVC template
- [ ] 2) go into your homecontroller
- [ ] 3) Create a getData method with the following signature
- [ ] 4) public string GetData()
- [ ] 5) HttpWebRequest request = WebRequest.CreateHttp("URL"); Paste this into your get data method
- [ ] 6) Add in you APIs URL, this is calling the API on the remote server.
- [ ] 7) NOTE: While the example one does not., some APIs require keys, those keys will go around here,

every APIs keys will function a little different, many won't have a key, but it needs to go before we pull out info from our request. It would look a little like: request.Headers.Add("X-Mashape-Key",value);

- [ ] 8) Next we need to get the response from the server, this is the reply from the server and it gives us back allllll it's info.
- [ ] 9) Paste in this line: HttpWebResponse response = (HttpWebResponse)request.GetResponse(); This is where we're asking the remote server for data.
- [ ] 10) Next we need to read in the raw data into a streamreader. Remember from file I/O a read allows us to look at an external file.
- [ ] 11) paste in: StreamReader rd = new StreamReader(response.GetResponseStream());
- [ ] 12) Now we need to store that raw data into a string, which will be easier to parse.
- [ ] 13) string ApiText = rd.ReadToEnd();
- [ ] 14) Now return ApiText!
- [ ] 15) Let's test it to be sure it's working. Call the method in your index action and dump the output into a viewbag. Does the info show up?
- [ ] 16) It's time to parse our string and turn it into a JSON object or XML object NOTE: This is where things change based upon whether you use JSON/XML. Most APIs come back in JSON, many do both, and very few are XML only.
- [ ] 17) Before we convert let's make a model to make passing this info around smoother. Go to your models folder and create a Person model. Looking at the JSON in a viewer, pick 5 of the most useful datapoints and make them into properties.
- [ ] 18) Once the model is created, let's build in HomeController a new method: public Person ConvertAPIToPerson(string APIText)
- [ ] 19) Paste in this Line: JToken t = JToken.Parse(text);
- [ ] This line is converting our big string into JSON format so we can start digging through it
- [ ] 20) Now here's where APIs vary wildly, but we need to dig through the JSON data to find what we want. Fortunately SWAPI has a clean and lean structure
- [ ] 21) Create a person object, we'll be storing our data in here. Using the model allows us to port around more data at once.
- [ ] 22)Paste in this line: p.Name = t["name"].ToString();
- [ ] Note how the string in the [] matches with what's in the API response.
- [ ] 23) Given the previous line, figure out how to pull out both URL and homeworld and store them both in our person object
- [ ] 24) Now return the person!
- [ ] 25) Lastly, like we did with the previous method, call it in our index action and pass the person's data

along to the view using a viewbag

Add an item

Activity

Show Details