MODULE 2

Server Side API: Part 2

What is REST?

- REpresentational State Transfer
- A group of software architecture design constraints that bring about efficient, reliable, and scalable distributed systems

- In a nutshell:
  - Specifically formatted URLs that return JSON

Uniform interfaces

- Representation and resources
  - Any information that can be named can be a resource
- Resource identification
  - Resources are identified by uniform resource identifiers, also known as URI’s
  - More definitions for the URI in your browser

RESTful URIs: Example with DVDStore

- Get all the films
  - https://localhost:8080/api/films
- Get comedies
  - https://localhost:8080/api/films/comedies
  - https://localhost:8080/api/films/4
- Get all actors in comedies
  - https://localhost:8080/api/films/comedies/actors
  - https://localhost:8080/api/films/4/actors

Old Friends

Method	Definition
POST	The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
GET	The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.
PUT	The PUT method replaces all current representations of the target resource with the request payload.
PATCH	The PATCH method is used to apply partial modifications to a resource.
DELETE	The DELETE method deletes the specified resource.

Method	CRUD	Example
POST	create	http;//www.te.com/students/
GET	read	http;//www.te.com/students/1234
PUT	update	http;//www.te.com/students/1234
DELETE	delete	http;//www.te.com/students/1234

HTTP Response Codes

- Key to REST to know if your request worked

Status Code	Description
100.x	Information Responses
200.x	Successful Responses
300.x	Redirection Messages
400.x	Client Error Responses
500.x	Server Error Responses

Successful Responses

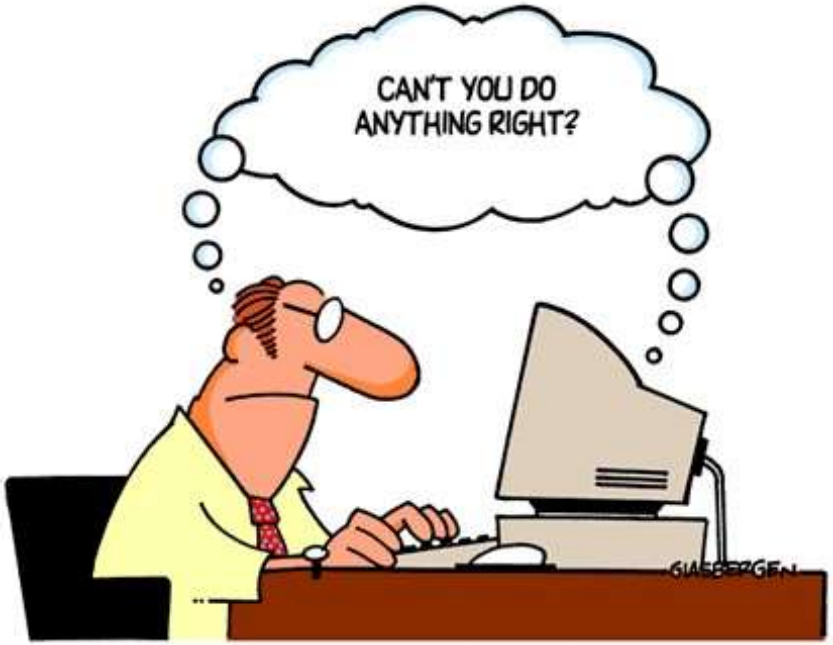
Code	Description
200 OK	The request has succeeded. The information returned with the response is dependent on the method used in the request.
201 Created	The request has succeeded and a new resource has been created as a result of it. This is typically the response sent after a POST request, or after some PUT requests.
202 Accepted	The request has been received but not yet acted upon. It is non-committal, meaning that there is no way in HTTP to later send an asynchronous response indicating the outcome of processing the request. It is intended for cases where another process or server handles the request, or for batch processing.
204 No Content	There is no content to send for this request, but the headers may be useful. The user-agent may update its cached headers for this resource with the new ones.

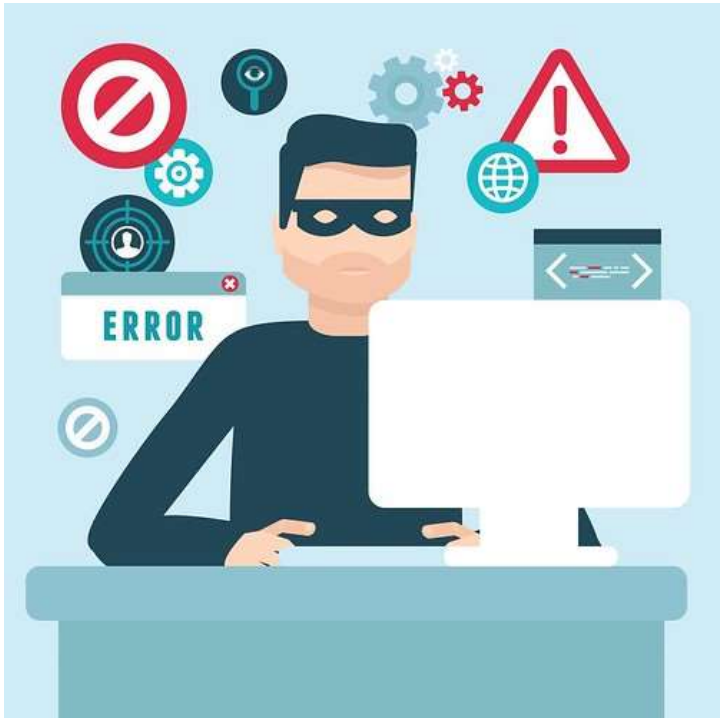
Errors

Code	Description
400 Bad Request	The request cannot be fulfilled due to bad syntax.
401 Unauthorized	Similar to 403 Forbidden, but specifically for use when authentication is possible but has failed or not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication.
403 Forbidden	The request was a legal request, but the server is refusing to respond to it. Unlike a 401 Unauthorized response, authenticating will make no difference.
404 Not Found	The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.
409 Conflict	Indicates that the request could not be processed because of conflict in the request, such as an edit conflict.

Code	Description
500 Internal Server Error	The request cannot be fulfilled due to bad syntax.
503 Service Unavailable	The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state.

The 2 Types of users





Bad Input

- We ask for a number, user types a letter.
- We ask for a birthday, user types wrong format.
- We ask for confirmation password, user types different password
- Etc.

Bad Input

- Why?
  - User's don't know any better
  - They made a mistake when typing
  - They are trying to break our application

- What could go wrong?
  - Input could be corrupted and data not standardized
  - Our app crashes
  - User is trying to hack our system

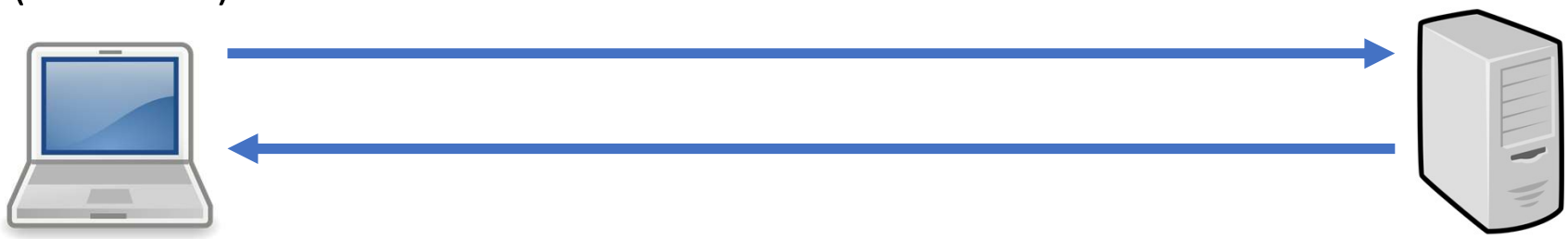
Validation

Client Validation

- Blocks the browser from sending the request, eliminating the need for a full page load
- Provides better feedback to the user
- Eliminates the need for a request/response round-trip
- Uses Javascript
- Can be bypassed
- (Module 4)

Server Validation

- Occurs when the request is received
- Protects the server from malicious input
- If a hacker bypasses the UI provided, they cannot bypass server-side validation
- Uses Java/C#



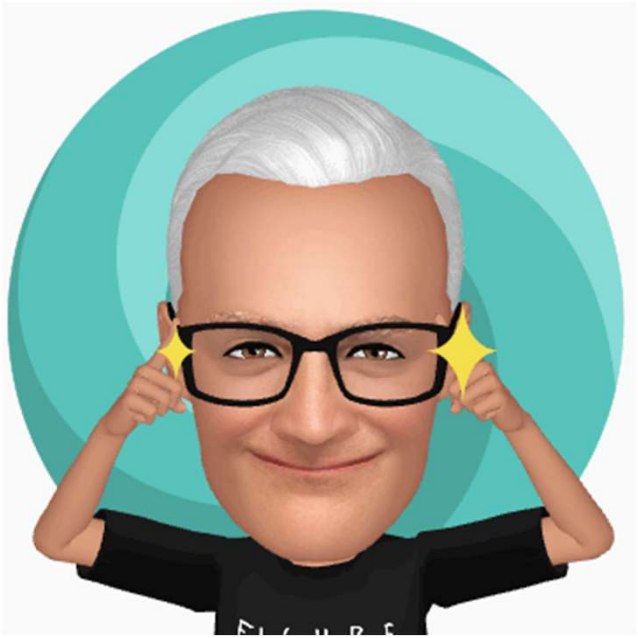
Types of Validation

- Required Values
- Minimum / Maximum Length
- Within acceptable range of values
- Password matches
- Valid type (credit card, email address, date, phone-number)
- Pattern Match (regular expression)
- etc.

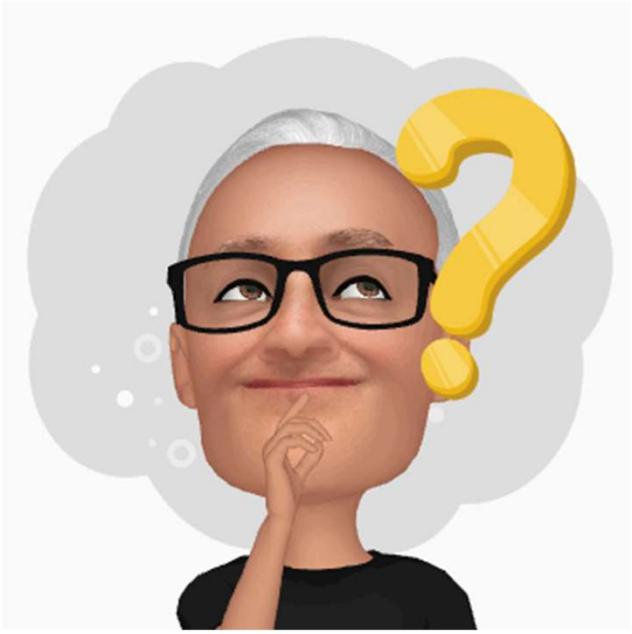
Data Annotations

- using System.ComponentModel.DataAnnotations
  - [Required] - Indicates a property is required to have a value
    - [Required(ErrorMessage = "This field must have a value")]
  - [StringLength(int)] - indicates a maximum length for the property
  - [StringLength(int, MinimumLength=int)] - indicates a maximum and a minimum length for a property
  - [Range(min, max)] - provides a minimum and maximum range for a numeric property
  - [Compare(propertyName)] - ensures two properties are equivalent
  - [EmailAddress] - validates the field as an email address

LET'S CODE!



WHAT QUESTIONS DO YOU HAVE?



Reading for tonight: Authentication