.NET Core Training

Building a RESTful API with ASP.NET Core 3

Week 3 : 9th Feb – 15th Feb

Action Items

- Inheritance
- Abstract classes
- Events
- Linked List

Action Items

Events :

- An event is a notification sent by an object to signal the occurrence of an action. Events in .NET follow the observer design pattern.
- The class who raises events is called **Publisher,** and the class who receives the notification is called **Subscriber** .
- There can be multiple subscribers of a single event. Typically, a publisher raises an event when some action occurred. The subscribers, who are interested in getting a notification when an action occurred, should register with an event and handle it.
- The delegate defines the signature for the event handler method of the subscriber class.

Event Declaration :

Syntax: ***event*** *<delegate-name> <event-name> ;*

Action Items

Abstract Classes :

Abstraction in C# is the process to hide the internal details and showing only the functionality. The abstract modifier indicates the incomplete implementation. The keyword abstract is used before the class or method to declare the class or method as abstract. Also, the abstract modifier can be used with indexers, events and properties.

*Example : public **abstract** void geek(); // this indicates the method 'geek()' is abstract*

***abstract** class gfg // this indicates the class 'gfg' is abstract*

Action Items

Inheritance :

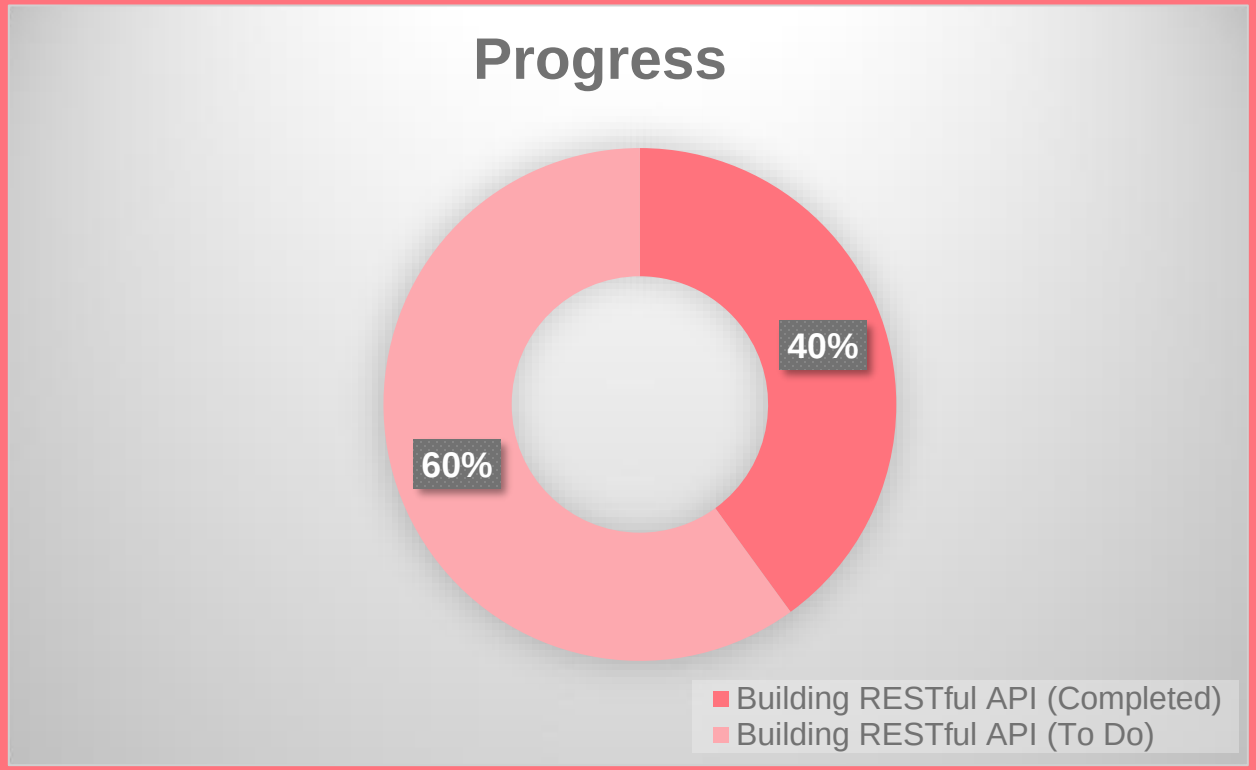
Acquiring (taking) the properties of one class into another class is called inheritance. Inheritance provides reusability by allowing us to extend an existing class.

Types of Inheritance :

- Single inheritance
- Hierarchical inheritance
- Multiple inheritance using Interfaces

AGENDA

1. Progress
2. Learnings from the course
3. Demo


Progress
40%
60%
Building RESTful API (Completed)
Building RESTful API (To Do)

Learnings from the Course

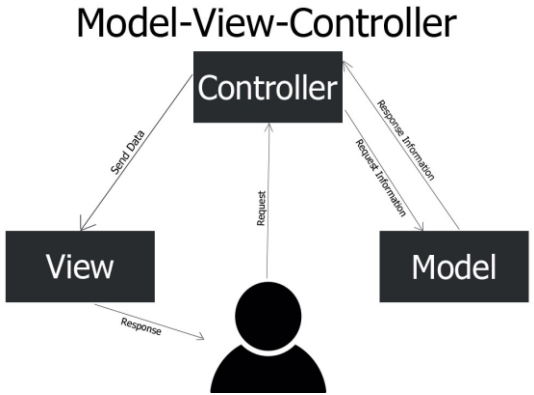
Introduction to REST

Structuring the Outer Facing Contract

Introduction to REST

MVC (Model View Controller) :

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application.



Introduction to REST

RESTful API :

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.

- REST is an Architectural Style, not a standard.
- We use standards to implement this architecture.

Introduction to REST

REST Architectural Constraints:

REST defines 6 architectural constraints which make any web service – a true RESTful API.

- Uniform Interface
- Client–server
- Stateless
- Cacheable
- Layered system
- Code on demand (optional)

Introduction to REST

Richardson Maturity Model

Leonard Richardson analyzed a hundred different web service designs and divided them into four categories based on how much they are REST compliant. This model of division of REST services to identify their maturity level is called Richardson Maturity Model.

Richardson used three factors to decide the maturity of a service i.e., URI, HTTP, Methods and HATEOAS (Hypermedia). The more a service employs these technologies – more mature it shall be considered.

Introduction to REST

In this analysis, Richardson described these maturity levels as below:

- Level Zero
- Level One
- Level Two
- Level Three

Level Zero

Level zero of maturity does not make use of any of URI, HTTP Methods, and HATEOAS capabilities.

Level One

Level one of maturity makes use of URIs out of URI, HTTP Methods, and HATEOAS.

Introduction to REST

Level Two

Level two of maturity makes use of URIs and HTTP out of URI, HTTP Methods, and HATEOAS.

Level Three

Level three of maturity makes use of all three, i.e., URIs and HTTP and HATEOAS.

Structuring the Outer Facing Contract

Resource Identifiers :

REST APIs use Uniform Resource Identifiers (URIs) to address resources. REST API designers should create URIs that convey a REST API’s resource model to its potential client developers. When resources are named well, an API is intuitive and easy to use.

Resource naming guidelines :

- Use nouns to represent resources
- Represent hierarchy while naming resources

Structuring the Outer Facing Contract

HTTP Methods :

The primary or most-commonly-used HTTP verbs (or methods) are:

- POST - CREATE
- GET - READ
- PUT - UPDATE / REPLACE
- PATCH - UPDATE / MODIFY
- DELETE - DELETE

Structuring the Outer Facing Contract

PAYLOAD :

A payload in API is the actual data pack that is sent with the GET method in HTTP. It is the crucial information that you submit to the server when you are making an API request. The payload can be sent or received in various formats, including JSON. Usually, the payload is denoted using the “{}” in a query string.

Payload = “{}”

Structuring the Outer Facing Contract

Content Negotiation in REST APIs :

Generally, resources can have multiple presentations, mostly because there may be multiple different clients expecting different representations. Asking for a suitable presentation by a client, is referred as content negotiation.

Two types of content negotiation :

- Server-driven
- Agent-driven

Structuring the Outer Facing Contract

Content Negotiation in REST APIs :

If the selection of the best representation for a response is made by an algorithm located at the server, it is called **server-driven negotiation**. If that selection is made at agent or client-side, its called **agent-driven content negotiation**.

Content negotiation using HTTP headers :

At server side, an incoming request may have an entity attached to it. To determine its type, server uses the HTTP request header Content-Type

Example : *Accept: application/json*

Structuring the Outer Facing Contract

Content negotiation using URL patterns :

Another way to pass content type information to the server, the client may use the specific extension in resource URIs

Example : http://rest.api.com/v1/employees/20423.xml

http://rest.api.com/v1/employees/20423.json

Structuring the Outer Facing Contract

HTTP Status Codes:

REST APIs use the Status-Line part of an HTTP response message to inform clients of their request's overarching result.

HTTP defines these standard status codes that can be used to convey the results of a client's request. The status codes are divided into the five categories.

- **1xx: Informational** – Communicates transfer protocol-level information.
- **2xx: Success** – Indicates that the client's request was accepted successfully.
- **3xx: Redirection** – Indicates that the client must take some additional action in order to complete their request.
- **4xx: Client Error** – This category of error status codes points the finger at clients.
- **5xx: Server Error** – The server takes responsibility for these error status codes.

DEMO

THANKYOU!