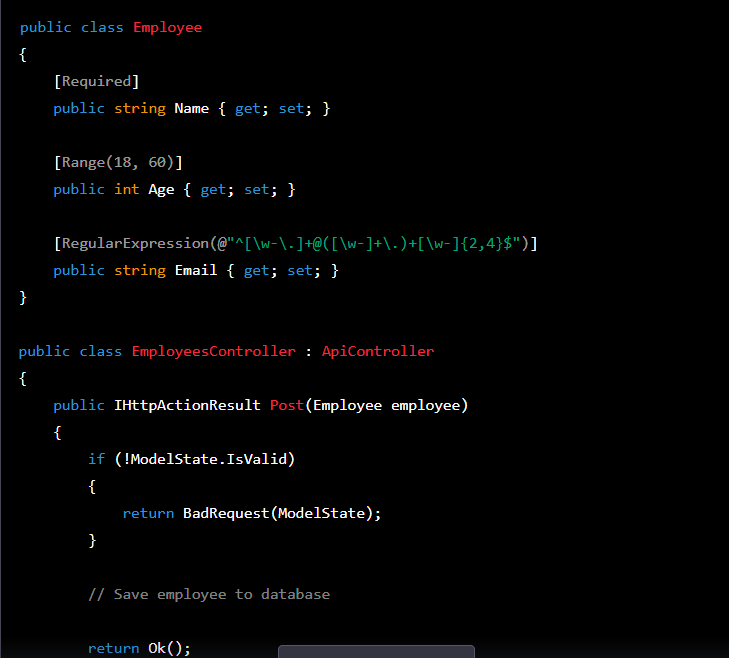
**DataAnnotations**

DataAnnotations are a set of attributes that can be used to define constraints, rules, and metadata for the properties of a model class.

These attributes can be used to perform validation on user input, format data for display, and provide additional information about the model.



**Exception Filter**

Exception filters are used to handle exceptions that occur during the execution of web API action

Exception Filter provides an ability to handle the exception for all the method, controller classes in one place.

. Exception filters execute when some of the exceptions are thrown from an action.

The exception can be anything. This is by creating a class that inherits from IExceptionFilter and FileAttribute interface.

using System;

using System.Net;

using System.Net.Http;

using System.Web.Http.Filters;

public class CustomExceptionFilter : ExceptionFilterAttribute

{

public override void OnException(HttpActionExecutedContext context)

{

if (context.Exception is DivideByZeroException)

{

var response = new HttpResponseMessage(HttpStatusCode.BadRequest)

{

Content = new StringContent("Attempted to divide by zero!")

};

context.Response = response;

}

else

{

var response = new HttpResponseMessage(HttpStatusCode.InternalServerError)

{

Content = new StringContent("An error occurred.")

};

context.Response = response;

}

}

}

[CustomExceptionFilter]

public class ValuesController : ApiController

{

public int Get()

{

int x = 1, y = 0;

return x / y; // attempting to divide by zero

}

}

**Action Filters**

Action Filters are used to perform custom processing before or after an action method is executed.

Action filters are executed before or after the WEB API actionmethod is executed.they can be used to modify the request or response,perform logging,caching,authorization,or any other custom processing.

We should use exception filters to handle exceptions and action filters to perform custom processing before or after an action method is executed.

when to use action filter?

\* You can use an action filter to check whether a user is authenticated and authorized to access a particular action method. This can help prevent unauthorized access to sensitive resources.

\* Action filters can be used to validate input data and ensure that it meets the required format and constraints

using System;

using System.Net;

using System.Net.Http;

using System.Web.Http.Controllers;

using System.Web.Http.Filters;

public class ValidateModelAttribute : ActionFilterAttribute

{

public override void OnActionExecuting(HttpActionContext actionContext)

{

if (!actionContext.ModelState.IsValid)

{

var response = new HttpResponseMessage(HttpStatusCode.BadRequest)

{

Content = new StringContent("Invalid input data.")

};

actionContext.Response = response;

}

}

}

public class ValuesController : ApiController

{

[ValidateModel]

public IHttpActionResult Post(MyModel model)

{

// code to process valid input data

}

}

**Inversion Of controller**

when one object requires the services of another object, it has to create an instance of that object and call its methods directly. This creates tight coupling between objects, making the code difficult to maintain and test.

Instead of creating objects directly, objects are requested from the container, which in turn creates and initializes them. The container manages the lifetime of objects and injects dependencies into them, making it easier to test and maintain the code.

Error Model

an error model is a structured representation of an error message that provides additional information about an error that occurred during the processing of a request.

The error model can be used to provide more meaningful error messages to clients and to help diagnose and troubleshoot errors in the application.

**Modelclass**

public class ErrorModel

{

public int ErrorCode { get; set; }

public string ErrorMessage { get; set; }

public string ErrorDetails { get; set; }

}

Code:

public IHttpActionResult GetUsers()

{

try

{

// perform some operation that might throw an exception

}

catch (Exception ex)

{

// create an error model and set its properties based on the exception

var error = new ErrorModel {

ErrorCode = 500,

ErrorMessage = "An error occurred while processing the request",

ErrorDetails = ex.Message

};

// return the error message to the client

return Content(HttpStatusCode.InternalServerError, error);

}

}