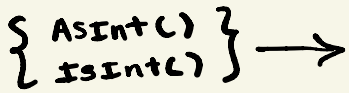
Asp.net

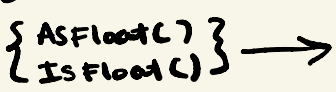
@Razor
Razor is a markup syntax that lets you embed server-based code into web pages.

Razor code block are enclosed in @{ ... }

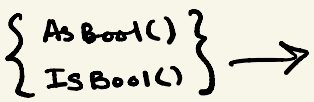
Converting data types



Converts a string to a integer



Converts a string to a floating -point number.



Converts a string to a Boolean.

.NET languages compile to CIL (common intermediate language) or CLR

.NET is , as you say, a library of code that .NET languages can take to.

.NET languages come in different flavors such as :

C#, VB.NET, Managed C++, F#.

Difference between library and framework

-Library contains many pieces of functionality that you may pick and choose from using one piece of technology doesn't mean you're locked into the rest.

- A framework, however very much sets out how you will be working it provides a workflow. This means rapid development.

ASP.NET is a web development platform which provides programming model, a comprehensive software infrastructure and various services required to build up robust web applications.

ASP.NET works on top of the HTTP protocol uses the HTTP command and policies to set a browser -to-server bilateral communication and cooperation.

ASP.NET is a technology which works on .NET framework that contains all web-related functionalities.

.NET framework is made of an object- oriented hierarchy.

An ASP.NET web application is made of pages. When a user requests an ASP.NET page, IIS delegate the processing of the page to asp.net runtime system.

App.net runtime transforms the .aspx page into an instance of a class. Which inherits from the base class page of the .net framework.

Each asp.net page is an object and all its components .

Server-side controls are also objects.

1. Common language runtime or CLR
It performs memory management, exception handling, debugging, security checking, thread execution, code execution, code safety, verification, and compilation.

- code managed by CLR managed code

Managed code is compiled - converts the source code into CPU independent intermediate language (IL) code.

A just in time (JIT) compiler compiles IL code into native code, which is CPU specific.

2. Common Type System
- it provides guidelines for declaring, using, and managing types at runtime, and cross - language communication.

3. Metadata and Assemblies

- metadata is the binary information, describing the program which is either stored in portable executable file (PEF) in the memory. Assembly is a logical unit consisting of the assembly manifest, type metadata, IL code, and a set of resources like image files.

4. ASP.NET AJAX

Asp.net Ajax contains the component that allow the develops to update the data non a website without a complete reload of the page.

ADO.net allows connection to data sources for retrieving, manipulating, updating the data.

6. Windows communication foundation (WCF)
It is the technology used for building and executing connected system.

ASP.NET Security

Authentication : it is the process of ensuring the user's identity and authenticity. ASP.NET allows four types of authentication.

- Windows Authentication
- Forms Authentication
- Passport Authentication
- Custom Authentication.

IIS Authentication: SSL

THE secure socket socket layer or SSL is the protocol used to ensure a secure connection.

With SSL enabled, the browser encrypts all data sent to the server, and decrypts all data coming from server.

Using SSL, the server authenticates itself by sending its digital certificate.

To use SSL, you need to buy a digital secure certificate from a trusted certification authority and it in web server.

WCF

It is a framework for building, configuring, and deploying networks - distributed services. It enables hosting services in any type of operating systems.

WCF application consists of three components.

WCF service
WCF service Host
WCF service client

Fundamental concepts of WCF

1. Message : communication unit includes several parts apart from the body. Body - body + other stuff.

2. Binding : it defines a way an endpoint communicates. It comprises of some binding elements that make the infrastructure for communication.

3. Contracts: it is a collection of operations that specifies what functionality the endpoint exposes to the client. It generally consists of an interface name

Attributes : WCF service is defined by ServiceContract and Operation contract attributes, whereas a web service is defined by web service and web method attributes.

Hosting mechanism various activation. Mechanisms are there for WCF hosting IIS , WAS self -hosting , but web service is hosted only by IIS.

Serializer: WCF supports data contract serializer by employing system.Runtime.Serialization whereas a web service supports xml serialization.

Contracts

The contracts layer is just next to the application layer and contains information similar to that of a real world contract that specifies the operation of a service and kind of accessible information it will make.

Service contract : this contract provides information to the client as well as to the outer world about the offering of endpoint and protocols to be used in communication layer.

Data contract: the data exchanged by a service is identified by a data contract. Both the client and the service has to be in agreement with the. Data contract.

Message contract: a data contract is controlled by a message contract. It primarily does the customization of the type formatting of the SOAP message parameters.

Policy and binding : there are certain pre-conditions for communication with a service. And such conditions are defined by policy and binding contract , a client needs to follow this contract.

Console Application C#

Encapsulation: is a mechanism of building the data, and functions that use them.

data abstraction is mechanism of exposing only the interfaces and hiding the implementation details from the user.

```
Console.WriteLine();
```

This Belongs to system class.

```
Var comp = new computer();
    Comp.GetNewType();
```

```
String Name;  // Automatically set itself to private.
```

```
Public, Private, Protected, Interned.
```

Internal : which one allow classes of the same name space or the same type to use it.

Protected: which allow any classes that are based on this particular class, to have access to that field.

- Protected String _name = "Unknown";

```
Public String Name
{ get {
        return _name;
}
Set {
        _name = value;
}
```

Value is Parameter that is provided by our setter

value is Lowercase.

```
Public Void TogglePower()   <- main class have
{                              access to this
    if (isOn)
    { TurnOff(); }
    else
    { TurnOn(); }

Protected Void TurnOn()
{   isOn = true; }
Protected Void TurnOff()
{   isOn = false; }
```

Now other classes can only control the computer power state, by hitting toggle.

They can see whether the computer is on or off, but they can no longer have access to the internal methods that make the process of toggling the power work.

Inheritance it is the mechanism in C# by which one class I allowed to inherit features (fields and methods) of another class.

Describes the ability to create new classes based on existing classes.

Abstract classes cannot be instantiated.

```
Class Desktop : Computer        // extends computer class.

    Public Desktop (String name):
                                    base (name)
    { }

    Public enum CaseType
    {
        Tower,
        MiniTower
    }
    Public CaseType CaseType {get; Private set;}
```

In order to use same variables again in the child class, in super class variable needs to have virtual.

```
Public class computer
{ public virtual string Name {get; set;} }

Public class Desktop: Computer
{
    Public override string name {get; set;}
}
```

Polymorphism: you can have multiple classes that can be used interchangeably, even tough each class implements the same properties or methods in different ways

Interfaces : is a contract that allows us to know that particular public methods and properties will always be Available on class that implements the interface.

A thing that is added into interface, is automatically assumed to be public.

Composition: it'll allow us to share the same code, in multiple places.

Inheritance it is the mechanism in C# by which one class I allowed to inherit features (fields and methods) of another class.

Describes the ability to create new classes based on existing classes.

Console Application C#

Encapsulation: is a mechanism of building the data, and functions that use them.
data abstraction is mechanism of exposing only the interfaces and hiding the implementation details from the user.

C# provides automatic garbage collection.

Laptop.cs : Computers, Isleep

```
Private Isleep sleepController;

public Display display
    {get; set;}

public bool IsSleeping {
  get { return SleepController.
        IsSleeping;}}

Public Void ToggleSleep()}
SleepController. ToggleSleep();
}

Public Laptop (String Name,
    int width, int height)
  : base (name)
{ display = new Display
        (width, height);
  SleepController = new
        SleepController();
}
```

DesktOP.cs : Computer, ISleep

```
Private Isleep sleepcontroller;

Public bool IsSleeping
{get { return sleepcontroller.
        IsSleeping;}}

Public Void ToggleSleep()
{ if (ison) return;
  SleepController. ToggleSleep();}

Public override void TogglePower()
{ if( isSleeping ) return;
  base. TogglePower();}
```

```
Public enum CaseType
{Tower, MiniTower}

Public CaseType CaseType
    {get; Private set;}
```

Display.cs

```
Public int width {get; private set;}
Public int height {get; private set;}

Public Display(int width, int height)
{
    this.width = width;
    this.height = height;
}
```

IpowerOn.cs

```
bool Ison {get;}
Void TogglePower();
```

PowerController : IPowerOn

```
public bool Ison {get; Private set;}
Public virtual void TogglePower();
```

ISleep.cs

```
bool IsSleeping
    {get;}
Void ToggleSleep();
```

SleepController : ISleep

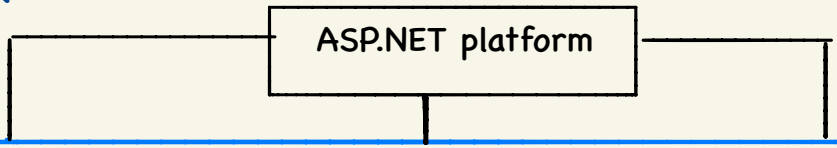
```
Public bool IsSleeping {get; Private set;}
Public ToggleSleep();
```

Using Generics c#

```
Public IList Items {get;} = new ArrayList();
                    }

Public void Dispose (params object[] dispose)
{       foreach (var item in dispose)
        {   items.add(item);
        }
}

Public class Multislot
{ public IList<Food> foods {get;} = new List<Food>();
  Public IList<Trash> waste {get;} = new List<Trash>();
}

Public void ProcessAll()
{ Process<Food>(foods, (food) => composer.GrowKale(food));
  Process<Waste>(waste, (trash) => SmartIncinerator.Burn(trash))
}
```

Generic Demonstration

```
Public void Process (IList<T> items,
                     Action<T> disposalMethod)
{
    Foreach (Var Item in items)
    {
        disposalMethod(item);
    }
}
```



Application Life cycle	Request Life cycle.
Travels the life of web app from moment it's starts to end.	Defines the path that HTTP request follows as it moves through the ASP.NET platform from the point at which request is received to response is sent.

Global.asax And Global.asax.cs

Now the term global application class is usually used to refer to the Global.asax.cs file to the context.

System.Diagnostic.Debugger.Break()

- Another way to select a debug point.

* Global Application class

```
PUblic Class MvcApplication :
                    System.web.HTTPApplication
{
    Protected Void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();
        RouteConfig.RegisterRoutes
                    (RouteTable.Routes);
    }
}
```

- Mvc class is instantiated in Asp.NET Framework.

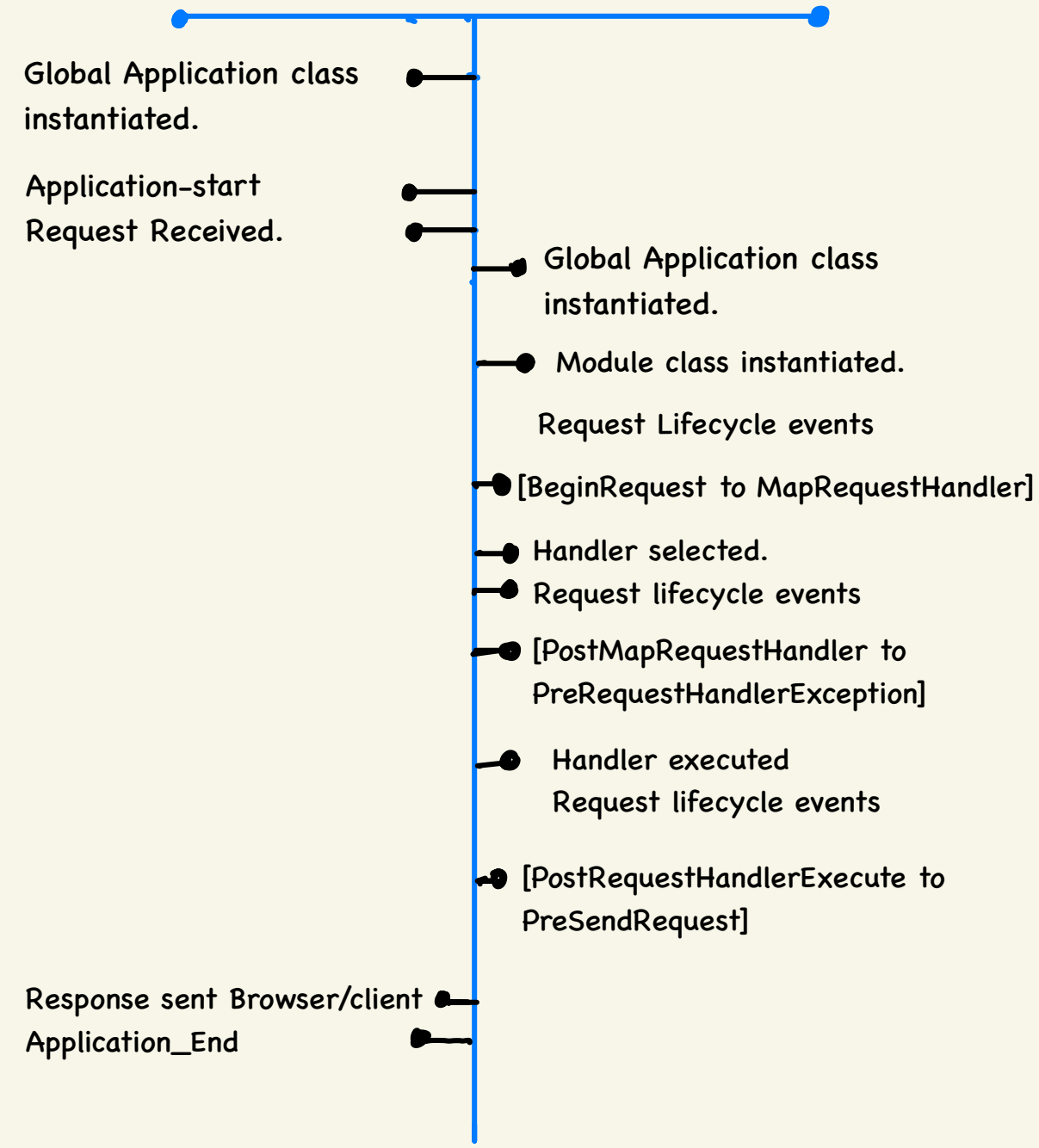
  - Application-start - called when the Application is started.
  - Application-End - called when the Application Is about to terminate.

- Application-End is called before the Application is terminated & is an opportunity to release any resources that the Application maintains.

  Modules are self-contained that receives the life-cycle events and can monitor & manipulate requests

- Handlers are responsible for generating response for request.

Adding the Request handling Process to the Life-cycle diagram



To access the data stored in the global Application class

- Use the HTTPContext.Application

- System.webHttpContext

```
Session["request_timeStamp"] = stamp;

Application["app_timestamp"] = stamp;
```

↗ global class.

- Can Retrieve it by HTTPContext.Class

```
HTTPContext.Application["app_timestamp"]

or

Session["request_timestamp"]
```

HttpRequest object

In C#, the HttpRequest class is part of the System.web namespace and represents the HTTP request that the server receives from the client. It provides properties & methods to access information about the request, such as query strings, form data, cookies, headers.

Use Request to Access this Property.

- ContentEncoding
- ContentLength
- currentExecutionFilePathExtension
- Headers
- HTTPMethod
- InputStream
- IsLocal
- MapPath
- Url
- UserAgent
- UserHostAddress
- UserHostName

1. HttpRequest.QueryString : Retrieves the collection of query string variables. (e.g. ?id=1);

```
String id = Request.QueryString("id");
```

2. HttpRequest.Form : Provides access to form variables posted by a form (e.g. POST request data).

```
String username = Request.Form["username"];
```

3. HttpRequest.Cookies : contains cookies sent by the client.

```
HttpCookie cookie = Request.Cookies["mycookie"];
```

4. HttpRequest.Headers : Retrieves the collection of HTTP headers sent by the client.

```
String userAgent = Request.Headers["User-Agent"];
```

5. HttpRequest.Url : Gets the Uri object of the requested URL.

```
Uri Url = Request.Url;
```

6. HttpRequest.HttpMethod : Returns the HTTP method used, such as GET or POST.

```
String method = Request.HttpMethod;
```

7. HttpRequest.UserHostAddress : Retrieves the IP address of the client making the request.

```
String ClientIp = Request.UserHostAddress;
```

8. HttpRequest.Files : Accesses files uploaded in a request, like those from an HTML file upload.

```
HttpPostedFile file = Request.Files["fileupload"];
```

Additional properties Defined by HTTP Request class.

- Files: Returns a collection of files sent by the browser in a form.
- Form :- provides access to the raw form data.
- params :- A collection of combined data items from query string Form Fields, & cookies.

HttpResponse

In C#, the HttpResponse class, part of the System.web namespace represents the HTTP response sent back to the client. This class allows you to set the response content, headers, cookies, and status code, among other response properties.

1. HttpResponse.StatusCode: sets or gets the HTTP status code for the response (e.g. 200 for success, 404 for not found.

```
Response.StatusCode = 404;
```

2.

HTTPResponse Class

- AppendHeader
- BufferedOutput
- Cache
- CacheControl
- Charset
- Clear
- clearContent
- Headers
- IsClientConnected
- Output
- OutputStream
- RedirectLocation
- Status
- StatusCode
- write(data)
- writeFile(path)

```
([Bind(Prefix="HomeAddress", Exclude="city")]
                                    AddressSummary
                                        summary)

    [Bind(Include="city")]
    Public Class AddressSummary
    {
        Public string city {get; set;}
        Public string country {get; set;}
    }
```

• NET Design Patterns

The 23 Gang of four (GOF) patterns are generally considered the foundation for all other Patterns.

- creational
- Structural
- Behavioral

Creational Patterns

- Abstract factory: Provides an interface for creating families of related or dependent objects w/o specifying their concrete classes.

Abstract Factory (continentFactory)
Declares an interface for operations that create abstract products