TechLift
KICKSTART YOUR CAREER

WEB ENGINEERING

.Net

P@SHA

PAKISTAN PSEB SOFTWARE EXPORT BOARD

Week 3 - Day 14

Recap Project (Time: 1 hr)

The students will work in pairs and build a simple

Student Registration Form

They may take help from this video to complete the project.

ASP.NET Web Applications

Learning Objectives

By the end of this session, the students will have developed an understanding of:

- Using Third Party Libraries using Nuget [MVC]
- Introduction to Design Patterns
- MVC Design Pattern
- ASP.NET MVC Out of the Box Features

Third Party Libraries using Nuget [MVC]

NuGet which is a package manager for .NET and Visual Studio. NuGet can be used to find and install packages, that is, software pieces and assemblies and things that you want to use in your project.

NuGet is not a tool that is specific to ASP.NET MVC projects. This is a tool that you can use inside of Visual Studio for console applications, WPF applications, Azure applications, any types of application.

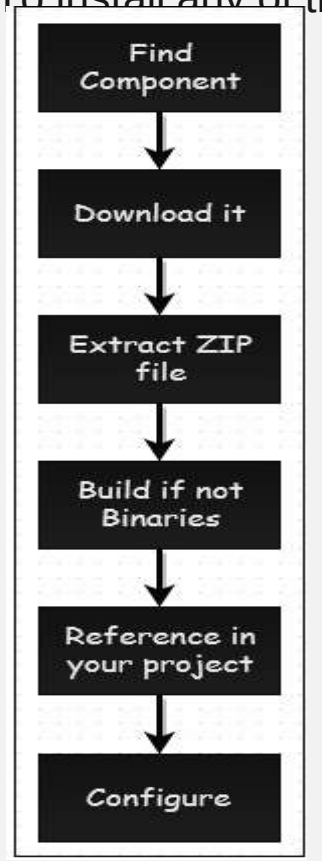
NuGet is a package manager, and is responsible for downloading, installing, updating, and configuring software in your system. From the term software we don’t mean end users software like Microsoft Word or Notepad 2, etc. but pieces of software, which you want to use in your project, assembly references.

For example, assemblies you want to use might be mock, for mock object unit testing, or NHibernate for data access, and components you use when building your application. The above-mentioned components are open source software, but some NuGet packages you find are closed source software. Some of the packages you'll find are even produced by Microsoft.

The common theme among all the packages mentioned above, like mock and NHibernate, and Microsoft packages like a preview for the Entity Framework, is that they don't come with Visual Studio by default.

When not using Nuget

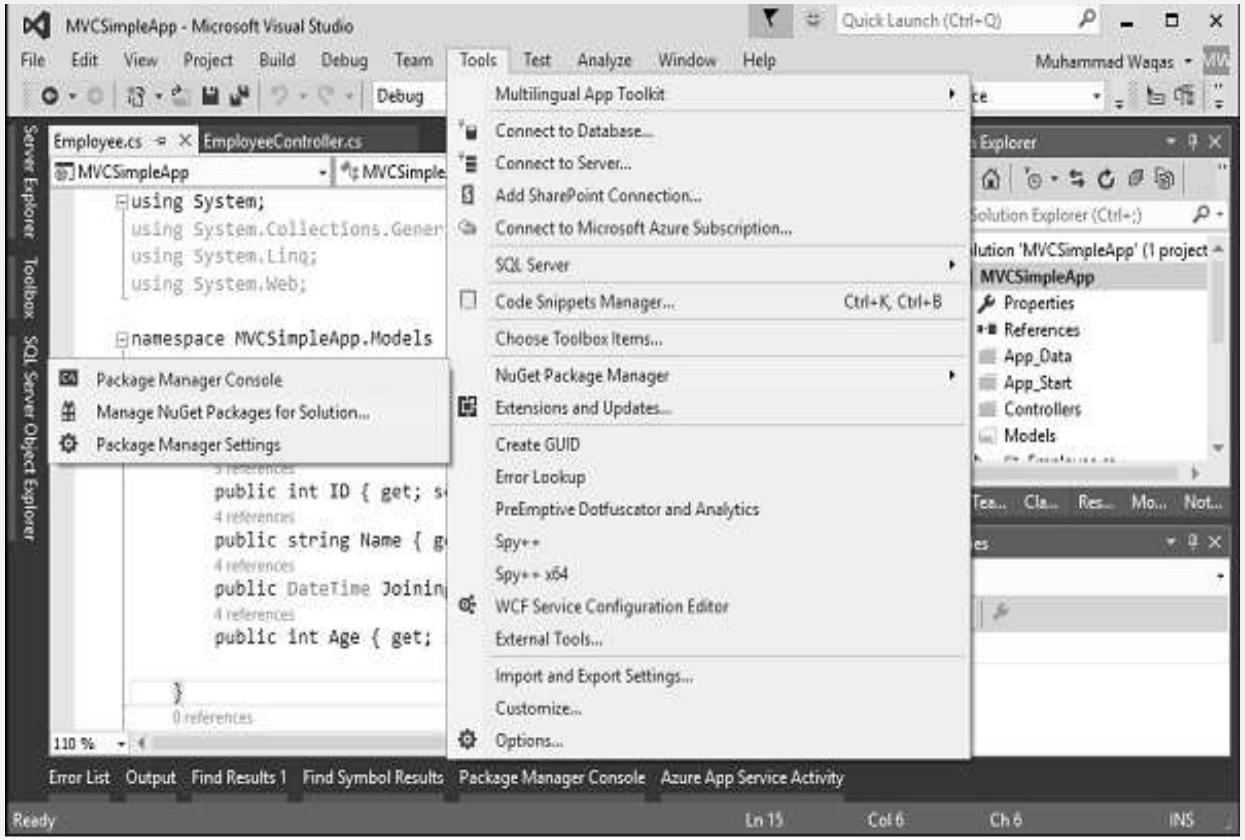
To install any of these components without NuGet, you will need the following steps.



NuGet Installation:

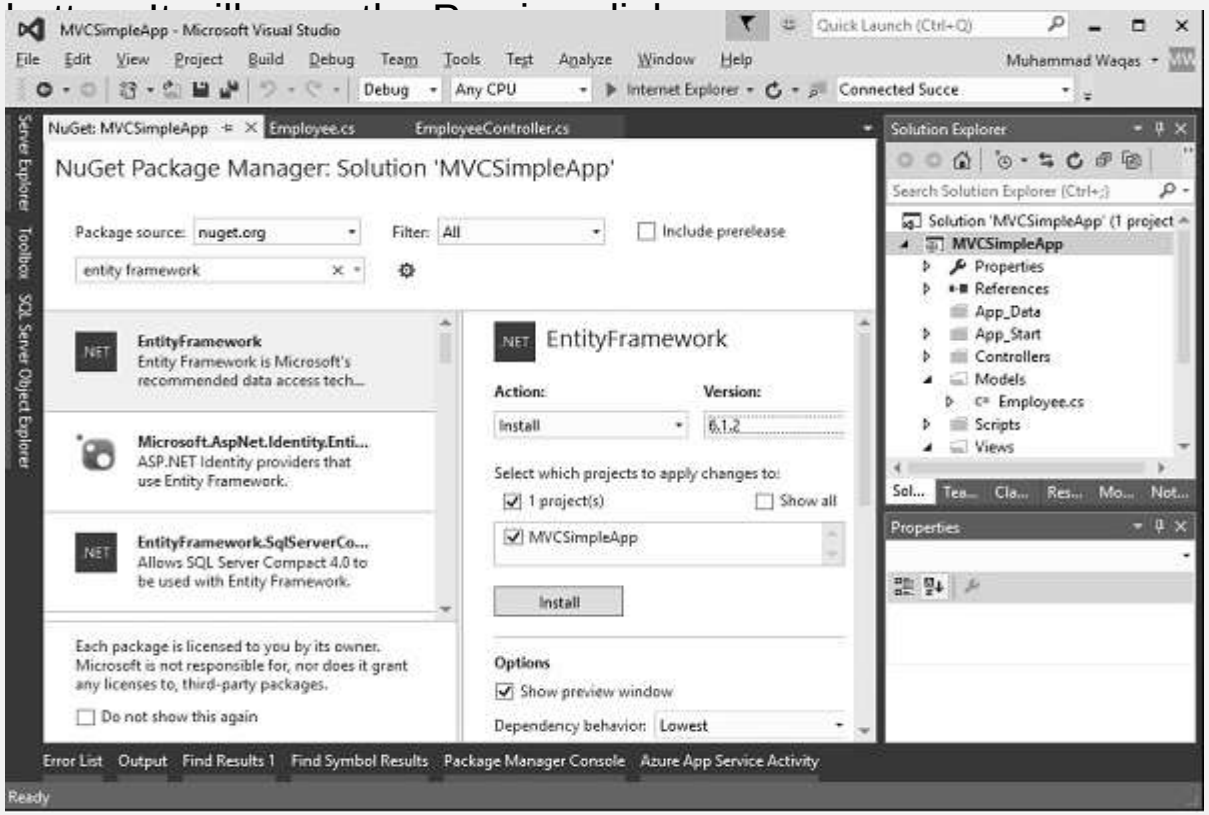
NuGet replaces all of the steps discussed earlier and you just need to say Add Package. NuGet knows where to download the latest version, it knows how to extract it, how to establish a reference to that component, and even configure it. This leaves you more time to just build the software.

Step 1 − Install the Entity Framework. Right-click on the project and select NuGet Package Manager → Manage NuGet Packages for Solution:

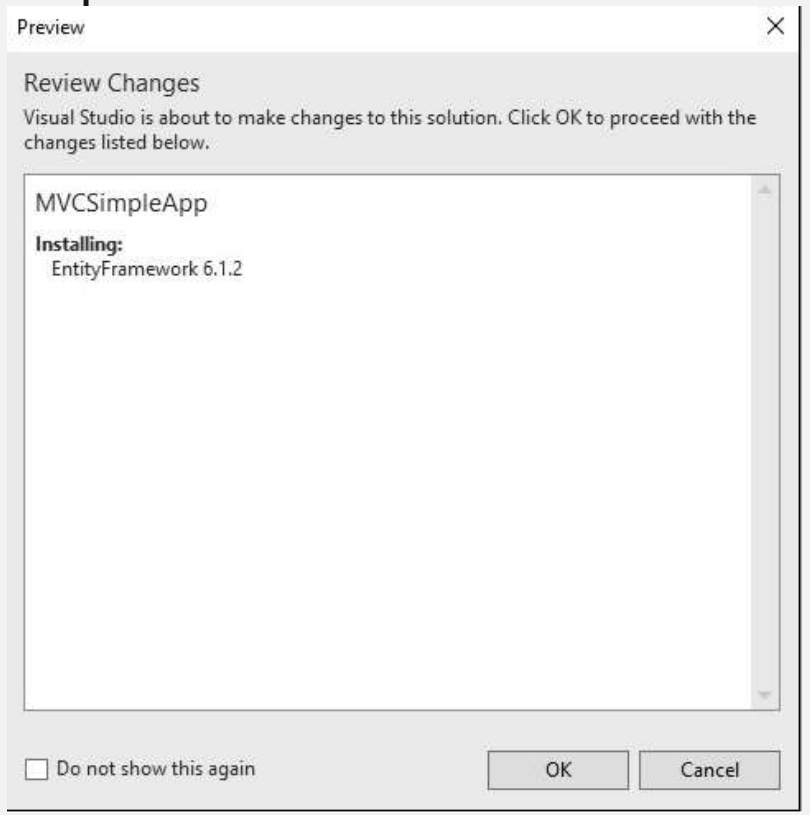


NuGet Installation(..continued)

Step 2 − Select the Entity Framework and click ‘Install’

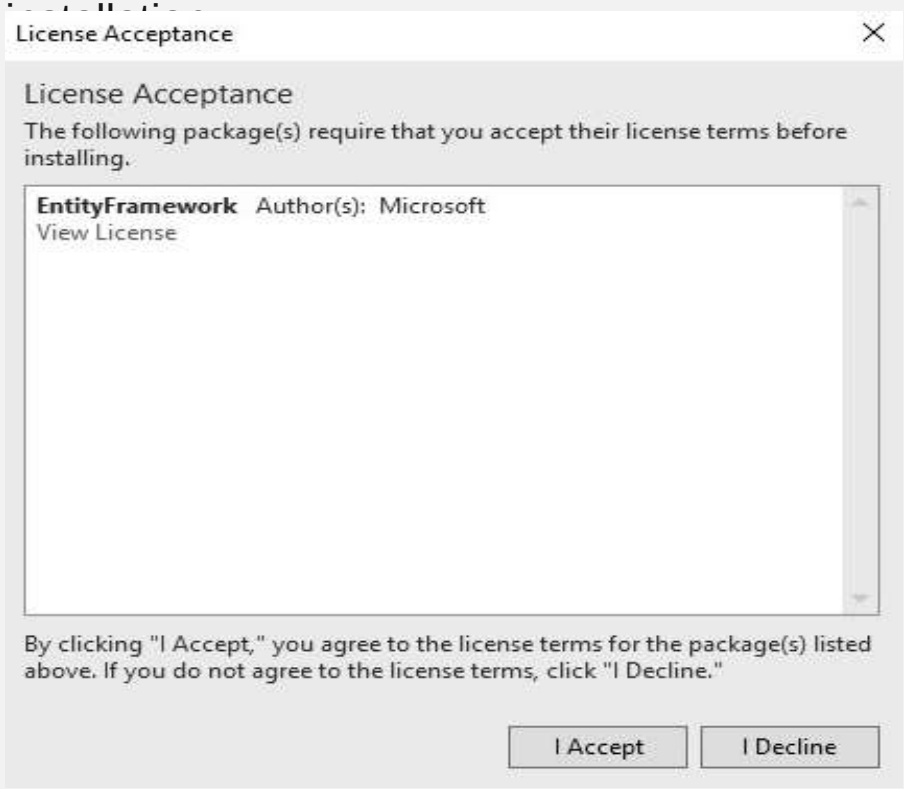


Step 3 − Click Ok to continue.



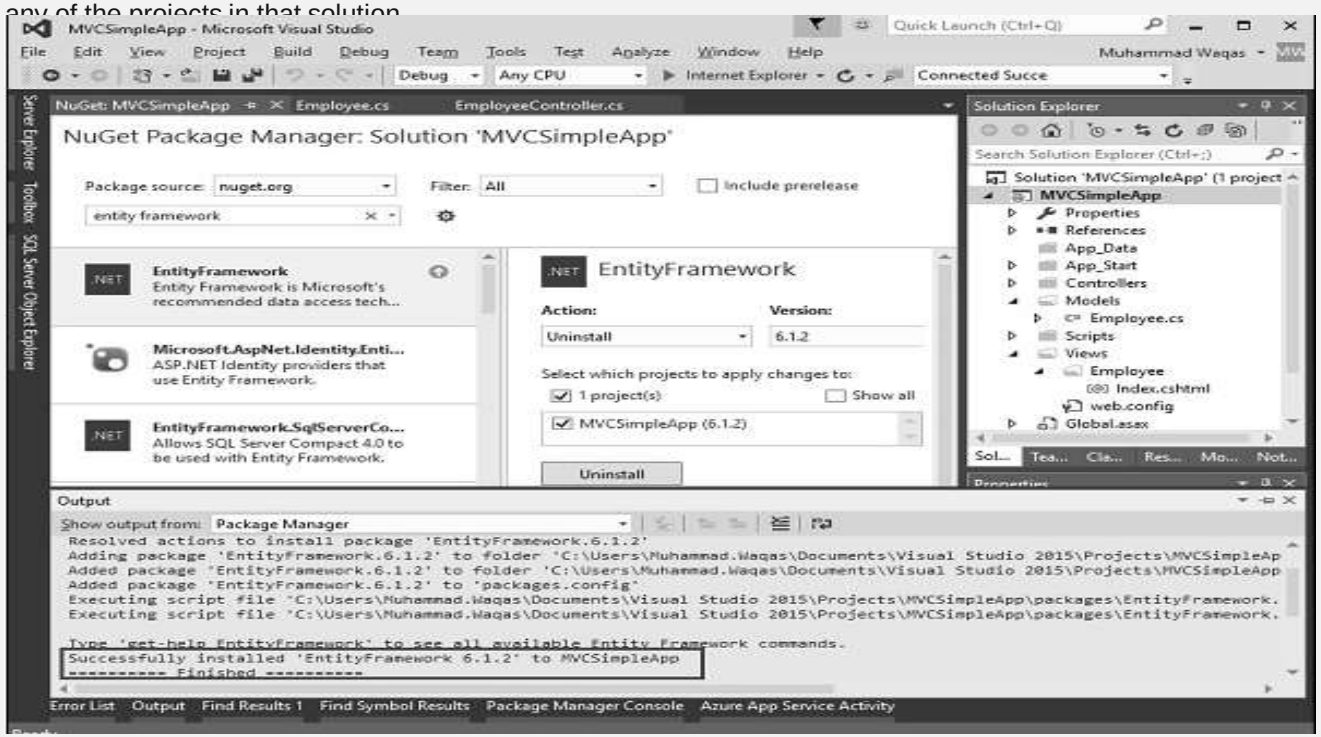
NuGet Installation(..continued)

Step 5 − Click the ‘I Accept’ button to start the installation.



Once the Entity Framework is installed you will see the message in out window as shown above.

When you install a package with NuGet, you will see a new packages directory in the same folder as the solution file hosting your project. This package directory contains all the packages that you have installed for any of the projects in that solution.



Implementation

The trainer will ask the students to follow the above mentioned steps and install a Nuget package on their systems

Once completed, click here to check the output.

Introduction to Design Pattern

Design patterns represent the best practices used by experienced object-oriented software developers. Design patterns are solutions to general problems that software developers faced during software development. These solutions were obtained by trial and error by numerous software developers over quite a substantial period of time.

Types of Design Patterns

S.N.	Pattern & Description
1	**Creational Patterns** These design patterns provide a way to create objects while hiding the creation logic, rather than instantiating objects directly using new operator. This gives program more flexibility in deciding which objects need to be created for a given use case.
2	**Structural Patterns** These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.
3	**Behavioral Patterns** These design patterns are specifically concerned with communication between objects.
4	**J2EE Patterns** These design patterns are specifically concerned with the presentation tier. These patterns are identified by Sun Java Center.

Creational Design Patterns

The following 6 design patterns belong to the Creational Design Pattern category.

1. Singleton Design Pattern
2. Factory Design Pattern
3. Abstract Factory Design Pattern
4. Builder Design Pattern
5. Fluent Interface Design Pattern
6. Prototype Design Pattern

Structural Design Patterns

The following 7 design patterns belong to the Structural Design Pattern category.

1. Adapter Design Pattern
2. Facade Design Pattern
3. Decorator Design Pattern
4. Bridge Design Pattern
5. Composite Design Pattern
6. Proxy Design Pattern
7. Flyweight Design Pattern

Behavioral Design Patterns

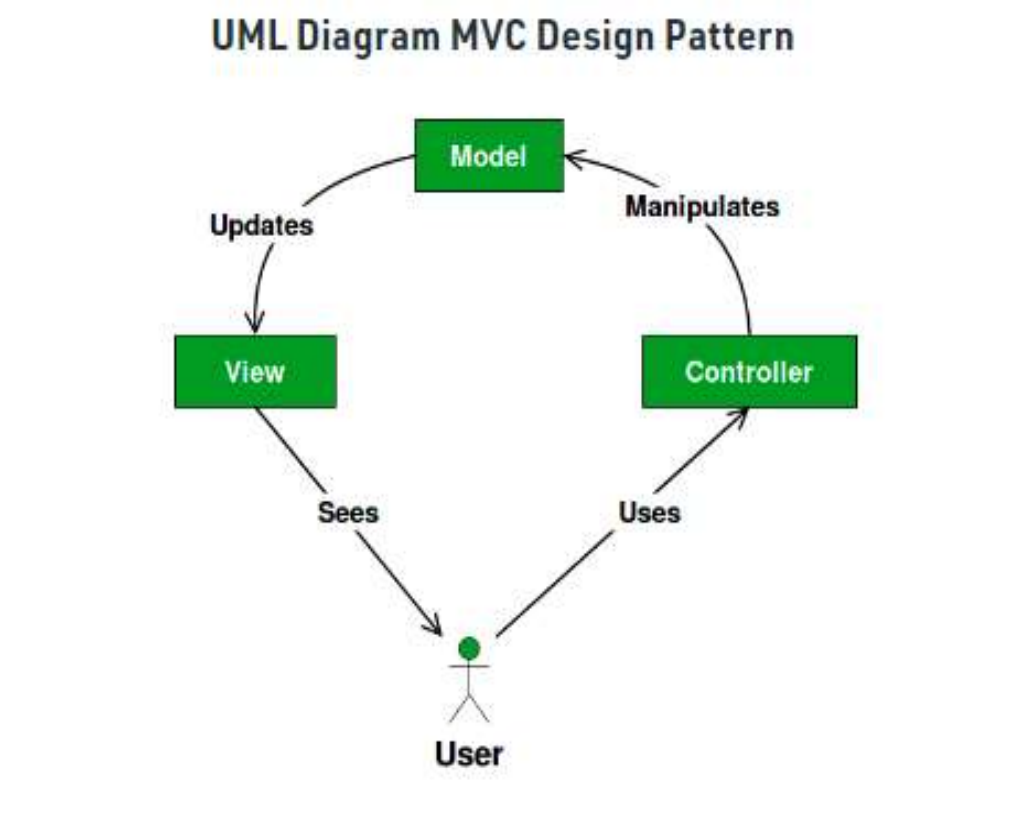
The following 7 design patterns belong to the Structural Design Pattern category.

1. Iterator Design Pattern
2. Observer Design Pattern
3. Chain of Responsibility Design Pattern
4. Interpreter Design Pattern
5. Command Design Pattern
6. Memento Design Pattern
7. Mediator Design Pattern
8. State Design Pattern
9. Template Method Design Pattern
10. Strategy Design Pattern
11. Visitor Design Pattern

MVC Design Pattern

The **Model View Controller** (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects.

MVC is more of an architectural pattern, but not for complete application. MVC mostly relates to the UI / interaction layer of an application. You're still going to need business logic layer, maybe some service layer and data access layer.



Features of ASP.NET MVC

ASP.NET is one of the most successful web application development frameworks by Microsoft. With every update, new and extended features are added that help developers deploy highly scalable and high-performance web applications.

When coupled with application monitoring and other performance tools, such as a profiler, ASP.NET becomes a powerful solution for building incredible apps.

Within the framework itself, there are myriad features to help you overcome common development challenges, do more with your apps, and boost performance.

Cross-platform & container support

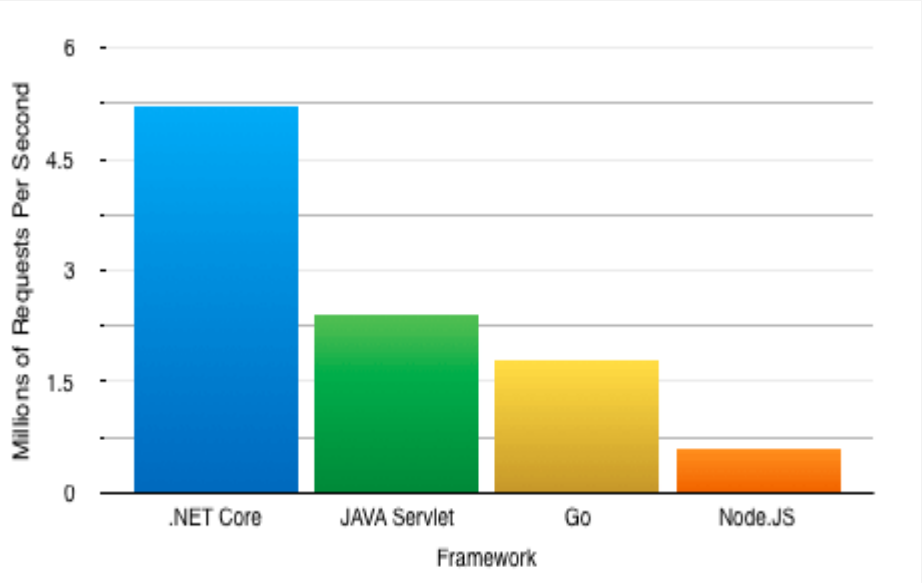
With the introduction of .NET Core, you can now create ASP.NET applications and deploy them to **Windows, Linux, and macOS**. Microsoft and the community have put a huge effort into making Linux a first-class citizen for running ASP.NET.

Containers are eating the clouds these days. Docker and other technologies are all the rage. ASP.NET Core allows developers to utilize all of these new technologies. Microsoft Azure even has support for deploying your application to containers and Kubernetes.

High performance

Some say that performance is a critical feature of your software. I tend to agree! With the introduction of ASP.NET Core and the Kestrel web server, ASP.NET is touted as one of the fastest web application frameworks available. TechEmpower has some cool benchmarks you can check out.

The technology that powered the ASP.NET integrated pipeline and IIS was roughly 15 years old. It did everything and carried a lot of baggage with it. The new Kestrel web server was redesigned from the ground up to take advantage of asynchronous programming models, be much more lightweight, and fast!



Asynchronous via async/await

ASP.NET has excellent support for utilizing asynchronous programming patterns. Async is now implemented in all common .NET Framework classes and most third-party libraries. Most modern applications spend most of their time and CPU cycles waiting for database queries, web service calls, and other I/O operations to complete.

One of the reasons ASP.NET Core is faster is its extensive use of asynchronous patterns within the new MVC and Kestrel frameworks.

```
//mark the method as async
public async Task GetGWB()
{
    HttpClient hc = new HttpClient();
    //await keyword handles all the complexity of async threading and callbacks
    await hc.GetAsync("http://geekswithblogs.net/Default.aspx");
    return true;
}
```

Multiple environments and development mode

One of my favorite features is the new environment feature. It allows you to easily differentiate parts of your code for their behavior in development, staging, production, etc. There was no standard way to do this before ASP.NET Core.

For example, it is used within your Startup.cs file to help configure your application. In this case, whether or not we want to show a more detailed exception page for development only.

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
{
    app.UseMiddleware<StackifyMiddleware.RequestTracerMiddleware>();
    app.UseSession();
    loggerFactory.AddNLog();
    app.AddNLogWeb();

    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();

    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        //app.UseBrowserLink();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }
```

HomeWork

Understand and Practice the above mentioned design patterns

Learning Objectives

By the end of this session, the students have practised

- Using Third Party Libraries using Nuget [MVC]
- Introduction to Design Patterns
- MVC Design Pattern
- ASP.NET MVC Out of the Box Features

Conclusion & Q/A

See you tomorrow!