1. **Платформа –** набор библиотек + набор инструментов для разработки ПО (Web-приложений) на основе этой платформы.

Конкретно про ASP.NET, платформа 2-го уровня. Она находится **над** платформой .Net

1. **.Net framework** – реализация CLI спецификации.

В .Net входит библиотека (FCL) и CLR.

***FCL (Framework Class Library)*** – реализация CLI BCL компанией Microsoft. Можно рассматривать, как API CRL.

***BCL (Base Class Library)***. Описание библиотеки поставляется в виде xml-файла ***CLILibraryTypes.xml***

1. **CLR** – (слова Смелова, помимо лекции) по большому счёту, это интерпретатор, который интерпретирует результат компиляции любого .net языка. 2-й варик: DLL, которая написана по спецификации COM, она устанавливается в Windows и его основная задача – интерпретировать MSIL(IL).

MSIL – промежуточный язык (высокоуровневый ассемблер). Реализация CLI CIL компанией Microsoft.

1. **Http-handler** – простейшее ASP.NET приложение. Для получения запроса, обработки, формирования и отправки ответа. **HttpContext** – контейнер, хранящий запрос и ответ, и с его помощью мы формируем ответ, извлекаем из него запрос и т.д.

В конфиге прописывается метод и path, по которому будет происходить доступ.

1. **Как идёт запрос:** нужно знать ip сервера, к которому мы обращаемся. Это символическое имя должно быть разрешено какого-то механизма (один из механизмов: DNS + файлик hosts). Есть функция **gethostbyname** – получаем ip по имени.

Как понять, к какому dns серверу мы обращаемся? В сетевых настройках указывается ip dns. Если всё стоит автоматически, то DHCP сам раздаёт все сетевые настройки.

1. **IIS-server** – набор серверов (с каждым узлом свои сетевые настройки (протокол, порт, ip) т.е. каждый узел ­­- это отдельный сервер) => наше приложение – это сервер, который работает в рамках iis-server. В рамках iis, для каждого сервера можно указывать версию CLR.NET, и поэтому в рамках iis могут работать несколько приложений на разных версиях .Net.

В рамках платформы есть набор стандартных обработчиков, которые предназначены для обработки запросов приложения определённого типа (Forms, MVC, …).

1. **Как контроллер взаимодействует с моделью (MVC)**: в рамках контроллера создаётся объект модели(точнее, DBContext (EntityFramework)). Модель создаётся для каждого запроса ЗАНОВО. Для каждого запроса создаётся НОВЫЙ экземпляр контролера. Контроллер живёт ровно 1 запрос. Результат работы контроллера – объект View (C# объект, который создаётся с помощью ViewEngine (механизм, который преобразует файлик cshtml в объект C#, и это происходит только ОДИН раз. Дальше работа идёт с созданным cs-файлом. Основная задача этого объекта: сгенерировать response, который потом уходит на клиент)).
2. **Global**.**asax**: содержит обработчики жизненного цикла приложений.
3. **Динамический объект ViewBag - коллекция**: способ передачи данных между controller и view. Это объект, которому можно динамически добавлять свойства и эти свойства будут доступны во view. (Единственный механизм передачи данных между view and controller). Добавление свойства **через точку**.
4. В MVC **3 модели**:

* набор параметров, который пришёл на Action (Action Model);
* которая вынеcена в MVC (которую м создаём при помощи EF). Модель данных
* Модель View( return View(“some data”));

1. Для маршрутизации можем также использовать атрибуты **[HttpGet]**…

Также, для указания нескольких методов **[AcceptVerbs(“post”, “get”)]**

Также отдельно можно указать маршрут: **[Route(“{x:int}/{name}”)]**

**[RoutePrefix(“bstu”)] –** указание префикса: <http://localhost:8081/bstu/>...

**Все атрибуты указываются ПЕРЕД методом (Action). Prefix – Перед КЛАССОМ.**

1. **SOA – Архитектура ориентированная на сервисы**
2. **Все веб сервисы можно разбить на 2 группы:**

**RPC – сервисы** – сервисы, обеспечивающие интерфейс удалённых процедур (предполагается, что интерфейс представляется в виде удалённого объекта с доступными методами).

**REST – сервисы** – представляет набор URI. 4-метода запросов (GET, POST, PUT, DELETE).

1. **ASMX – сервисы (способ взаимодействия RPC):**

**4 технологии:**

* **SOAP** – протокол обмена информацией. Позволяет организовать обмен информацией между клиентом и сервером. Сразу был по типу RPC, но после обновлений начал работать и по другим сервисам. Может перемещаться в разных протоколах (TCP, NamedPipe). Но если WEB, то HTTP в качестве транспорта. Заголовки для SOAP-сообщений: **application/xml+soap**
* **WSDL (Web service definition language)** – язык, формат xml, для досконального описания web-сервисов. Позволяет специальные xml файлики, которые описывают сервис. Там много комментариев для всего. **ОБЯЗАТЕЛЬНО ПОСМОТРЕТЬ ЛК Web-сервисы**
* **XML** – для разметки формата запроса и ответа
* **UDDI** – технология подразумевает, создание стандартных хранилищ WSDL описаний. Каталоги, в которых будут описания сервисов. Смотреть, какой сервис нам нужен.

**ASMX-сервис – упрощённый SOAP web-сервис. Сервис, который основывается на SOAP-протоколе. 3 признака:**

* SOAP протокол
* HTTP – транспорт
* исполняться с IIS сервером

**WCF – отдельная платформа, для разработки приложений, имеющих сервисно-ориентированную архитектуру.**

**Сервис – dll, которая имеет свой web.config**

1. **.NET Framework** – это старая платформа, работающая только под Windows.

**.NET Core** – это новая платформа, которая может работать как под Windows, так и под Linux. Но она еще довольно "сырая" - множество инструментов еще не перенесены на нее.

Подключить одно к другому не получится, потому что у них разные системные библиотеки (а также разные системы сборки если вы используете Visual Studio версии 2015 или ниже).

**Используйте .NET Core для своего серверного приложения, когда:**

* У вас есть кроссплатформенные потребности.
* Вы нацелены на микросервисы.
* Вы используете контейнеры Docker.
* Вам нужны высокопроизводительные и масштабируемые системы.
* Вам необходимы параллельные версии .NET для каждого приложения.

**Используйте .NET Framework для своего серверного приложения, когда:**

* В настоящее время ваше приложение использует .NET Framework (рекомендуется расширять, а не мигрировать).
* Ваше приложение использует сторонние библиотеки .NET или пакеты NuGet, недоступные для .NET Core.
* Ваше приложение использует технологии .NET, которые недоступны для .NET Core.
* Ваше приложение использует платформу, которая не поддерживает .NET Core. Windows, macOS и Linux поддерживают .NET Core.