**Программирование интернет-серверов**

**1ый вопрос**

1. Протокол HTTP: клиент-сервер; типы сообщений, структура запроса, структура ответа, статус (серии значений), методы, заголовки, параметры. Понятие stateless-протокола.
2. Протокол HTTPS: TLS, шифронаборы, сертификаты, процедура рукопожатия.
3. HTML. Структура HTML-страницы. Каскадные таблицы стилей (CSS). Модель DOM. Пример.
4. Протокол WebSockets: принципы работы и применения. Пример.
5. JavaScript. Основные стандарты. Типы данных. Программные структуры. Принцип применения. Пример.
6. Методология Ajax. Структура Ajax-приложения, принципы разработки и применения. Пример.

AJAX (Асинхронный JavaScript и XML) представляет собой технологию гибкого взаимодействия между клиентом и сервером. Благодаря ее использованию мы можем осуществлять асинхронные запросы к серверу без перезагрузки всей страницы. Правда, в настоящее время все больше вместо формата XML используется формат JSON для взаимодействия между клиентом и сервером.

Применительно к ASP.NET MVC использование AJAX вылилось в целую концепцию под названием "ненавязчивого AJAX" и ненавязчивого JavaScript (unobtrusive Ajax/JavaScript). Смысл этой концепции заключается в том, что весь необходимый код JavaScript используется не на самой веб-странице, а помещается в отдельные файлы с расширением \*.js. А затем с помощью тега <script> мы в веб-станице ссылаемся на данный файл кода.

Таким образом мы отделяем визуализацию от логики приложения. Что имеет свои плюсы. Так, выделение скрипта в отдельный загружаемый файл увеличивает производительность сайта, поскольку файл сохраняется в кэше и затем от туда подгружается.

Ajax – методология построения динамических приложений, при которых не осуществляется полная перезагрузка html-страниц.

Синхронный запрос – запрос, при котором поток, выдавший http-запрос, блокируется до поступления запроса.

Асинхронный запрос – запроса, при котором поток выдавший http-запрос, не блокируется до поступления запроса. Обрабатывается callback

Callback – функция, переданная в другую функцию в качестве аргумента, которая затем вызывается по завершению какого-либо действия

1. Web-приложение. Архитектура web-приложения. Особенности реализации web-приложения. Web-сервер и web-клиент. Пример.
2. ASP.NET: публикация ASP.NET-приложения, структура и параметры узла IIS, реальный и виртуальный каталоги, процедура настройки web-узла.
3. ASP.NET: http-обработчики, порядок разработки, http-обработчик для взаимодействия с клиентом по протоколу WebSockets. Пример.

HTTP-обработчики (HTTP Handlers) используются для генерации содержимого ответа на HTTP-запрос. Мы можем использовать HTTP-обработчики для настройки обработчик запросов. При обработке одного запроса мы можем задействовать несколько различных модулей, но только один http-обработчик может быть сопоставлен с запросом.

HTTP-обработчик представляет класс, реализующий интерфейс System.Web.IHttpHandler. Данный интерфейс определяет один метод и одно свойство:

* ProcessRequest(context): данный метод в качестве параметра принимает объект контекста запроса HttpContext и генерирует ответ клиенту
* IsReusable: это свойство указывает, будет ли данный обработчик использоваться другими запросами

HTTP-обработчик выбирается системой после возникновения события MapRequestHandler. И сразу после выбора обработчика срабатывает событие PostMapRequestHandler

Непосредственная генерация ответа обработчиком происходит после события PreRequestHandlerExecute - после этого события происходит вызов метода ProcessRequest и генерация ответа. А сразу после генерации ответа возникает событие PostRequestHandlerExecute

Зарегистрировать обработчики в web.config!

1. ASP.NET: ASMX-сервисы, WSDL, SOAP, прокси, порядок разработки, принципы применения. Пример.
2. ASP.NET: MVC-приложение, структура MVC-приложения, назначение основных компонентов приложения, маршрутизация. Пример.

MVC – архитектурный паттерн, 4 компоненты: модель, представление, контроллер и маршрутизатор.

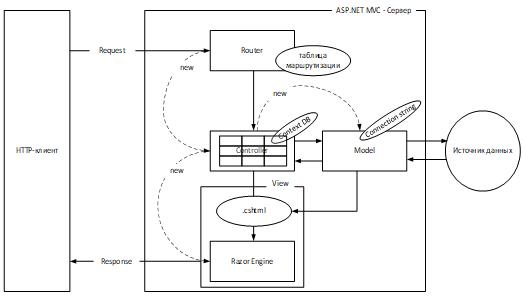
1) RouteTable – создан. Это первый шаг, выполняется первый раз при первом запуске приложения. RouteTable отображает url-адреса для обработчиков.

2) UrlRoutingModule – перехватывает запрос. Это второй шаг, происходит всякий раз, когда вы делаете запрос. UrlRoutingModule перехватывает каждый запрос, создаёт и выполняет правильный обработчик.

3) MvcHandler – выполняет. Создаёт контроллер, передаёт ему ControllerContext и выполняет контроллер.

4) Контроллер выполняет – определяет, какой метод контроллера выполнять, создаёт список параметров и выполняет метод.

5) Метод RenderView вызывается – метод контроллера вызывает RenderView() для рендеринга контента обратно в браузер. Метод делегирует свою работу определённому ViewEngine.



Структура MVC:

App\_Data – содержит файлы данных приложения, такие как LocalDB, файлы .mdf, xml. IIS никогда не будет обслуживать файлы из этой папки.

App\_Start – содержит файлы классов, которые будут выполняться при запуске приложения. Обычно это файлы конфигурации, такие как BundleConfig, FilterConfig, RouteConfig.

Content – содержит статические файлы, такие как Css, images, файлы значков.

Controllers – файлы классов для контроллеров.

Fonts – содержит файлы пользовательских шрифтов, для приложения.

Models – файлы классов моделей. Обычно класс модели включается в себя открытые свойства, которые будут использоваться приложения для хранения и манипулирования данными приложения.

Scripts – файлы JS, VBScript для приложения.

Views – html-файлы для приложения. Как правило файл .cshtml, в котором пишем HTML и C# код, или VB.NET.

Содержит отдельную папку для каждого контроллера.

Файлы конфигураций:

- Global.asax – позволяет писать код, который запускается в ответ на события уровня приложения, такие как Application\_BeginRequest, application\_start, application\_error, session\_start, session\_end.

- Packages.config – управляется nugget, чтобы отслеживать, какие пакеты и версии установлены в приложения.

- Web.config – конфигурации уровня приложения.

Shared – содержит представления, которые будут совместно использоваться различными контроллерами, например, файлами макетов.

**Маршрутизация:**

В приложении WebForms каждый url-адрес должен соответствовать определённому .aspx. То есть сопоставление каждого url с физическим файлом.

Маршрутизация позволяет определить шаблон url, который сопоставляется с обработчиком запросов. Обработчик может быть файлов или классом.

В WebForms обработчик – файл aspx, в MVC – класс Controller и метод Action.

Маршруты – шаблоны URL, которые используются для обработки запросов и которые могут использоваться для динамического создания URL.

Таблица маршрутов создаётся при первом запуске вашего приложения. Настраивается в global.asax.

RegisterRoutes() в Application\_Start, создаёт таблицу маршрутов по умолчанию.

Когда приложения обрабатывает запрос, оно просматривает коллекцию маршрутов в свойстве Routes, чтобы найти маршрут, соответствующий формату url-запроса. Порядок маршрутов в свойство Routes имеет большое значение, поскольку приложения использует первый найденный в коллекции маршрут, соответствует url-адресу

Все настроенные маршруты приложения хранятся в RouteTable и будут использоваться механизмом маршрутизации для определения соответсвующего класса или файла обработчика для входящего запроса.

Зарегистрировать маршрут можем в классе RouteConfig, который находится в RouteConfig.cs папке App\_Start.

Настроить маршрут с помощью метода расширения MapRoute() объекта RouteCollection. Минимум 2 параметра (имя, шаблон URL). Можно зарегистрировать несколько маршрутов с разными именами.

Можно поставить ограничение к маршруту, в свойстве constraints

После настройки в RouteConfig всех маршрутов, его надо зарегистрировать в событии Application\_Start, Global.asax

1. ASP.NET: MVC-приложение, маршрутизатор, принципы устройства и работы. Пример.

Процесс маршрутизации состоит из следующих этапов:

Модуль UrlRoutingModule пытается сопоставить текущий запрос с маршрутами в таблице RouteTable.

Если сопоставление завершилось удачно, то модуль маршрутизации выбирает обработчик маршрутов сопоставленного маршрута - объект IRouteHandler.

Затем у объекта IRouteHandler вызывается метод GetHandler, который возвращает объект IHttpHandler, используемый для обработки запроса.

У обработчика IHttpHandler вызывается метод ProcessRequest для обработки запросов.

По умолчанию обработчик запросов или объект IRouteHandler представляет экземпляр класса MvcRouteHandler, который возвращает объект MvcHandler, применяющий интерфейс IHttpHandler. Этот объект MvcHandler отвечает за инициализацию контроллера, который потом уже вызывает одно из своих действий.

1. ASP.NET: MVC-приложение, маршрутизация с помощью атрибутов, констрейны маршрутизации, принципы работы. Пример.

Здесь вместо определения маршрута мы используем метод routes.MapMvcAttributeRoutes();, который подключает в систему маршрутизации приложения функциональность атрибутов маршрутизации.

Мы также можем явно указать имя контроллера или метода или параметра в атрибуте: [Route("Home/{id:int}/{name}")].

1. ASP.NET: MVC-приложение, котроллер, жизненный цикл контроллера, взаимодействие с моделью и представлениями. Пример.
2. ASP.NET: MVC-приложение, представление, Razor Engine, жизненный цикл представления. Пример.
3. ASP.NET: MVC-приложение, модель, жизненный цикл модели, репозиторий. Пример.
4. ASP.NET: MVC-приложение, внедрение зависимостей. Пример.
5. ASP.NET: MVC Web API, структура Web API-приложения; назначение основных компонентов приложения, маршрутизация. Пример.
6. WCF-сервисы: WSDL, хост, прокси, модели взаимодействия клиента и сервера, порядок разработки, принципы применения. Пример.

**2ой вопрос**

1. ASP.NET CORE: программная платформа, принципы работы, архитектура. Пример.
2. ASP.NET CORE: работа со статическими файлами, добавление заголовков, стартовые страницы, файлы для скачивания, вывод в журнал. Пример.
3. ASP.NET CORE: MVC, настройка MVC и маршрутизатора, применение атрибута Route для маршрутизации. Пример.
4. ASP.NET CORE: MVC-котроллер, действия (action) контроллера, контекст контроллера, поддержка сессии, результат работы действия, внедрение зависимостей. Пример.
5. ASP.NET CORE: события OnAction, атрибуты HttpGet, HttpPost, …, AcceptVerb, принцип передачи параметров в метод действия.
6. ASP.NET CORE: Фильтры: Action Filter, Result Filter, Authorization Filter, Resource Filter, Exception Filetr, пользовательские фильтры действий.
7. ASP.NET CORE: MVC-представление, обнаружение представления, жизненный цикл представления, методы редеринга представления в web-страницу (методы View контроллера), способы передачи данных из котроллера в представление, строготипизированные представления, директива @model. Пример.
8. ASP.NET CORE: MVC-представление, директивы @using, @function, @inherits, #inject. Пример.
9. ASP.NET CORE: MVC-представление, директивы @addTagHelper, @removeTagHelper. Пример.
10. ASP.NET CORE: MVC-представление, применение компоновки (Layout) представления, компоновка по умолчанию (\_ViewStart), применение секций @RenderSection, @RenderBody. Пример.
11. ASP.NET CORE: MVC-представление, частичные представления (partial view). Встроенные хелперы. Пример.
12. ASP.NET CORE: MVC-представление, вспомогательные методы представления (хелперы). Пример.
13. SP.NET CORE: MVC-модель, DB-модель и View-модель. Модель Entity Framework,принцип Code разработки DB-модели. Объект ModelState, назначение и принципы применения. Атрибуты валидации: Required, RegularExpression, пользовательский атрибут валидации. Пример.

**3ой вопрос**

Лабораторная работа из курса ПИС.