ЛАБА 3

Нужно было сделать ТЕЛЕФОННЫЙ СПРАВОЧНИК с возможностями:

* просматривать список справочника (отсортировано по алфавиту);
* добавлять строку в справочник;
* удалять строку из справочника;
* изменять строку в справочникe.

Модель обеспечивает, хранение и изменение строк справочника в файле в json-формате

**MVC:** **Model-View-Controller** – паттерн (шаблон) проектирования, в котором приложение состоит из трех взаимодействующих компонентов: модель, представление, контроллер. В правильно разработанном MVC-приложений компоненты *относительно* независимы. Каждый компонент имеет свою зону ответственности: модель – данные и бизнес-логика; представление – динамическое формирование разметки для отправки клиенту; контроллер – обработка запроса, формирование экземпляра модели, вызов **Razor Engine**.

НАЧНЕМ С МОДЕЛИ

Contact.cs

Нашими данными являются контакты (по заданию). Это id, name, phone. Соответственно Contact – класс, содержащий данные поля и перегруженные конструкторы.

ContactsHolder.cs

Здесь происходит вся работа с json. Все контакты хранятся в List<Contact>. Можете изменить расположение json-файла. Соответствующие функции Find, Insert, Update(Удаление старого и добавление нового), Delete(находит по id и удаляет), GetAllContacts(массив контактов), WriteAllContactsInJsonFile(запись всего в json), ReadAllContactsFromJsonFile(возвращает список контактов).

КОНТРОЛЛЕРЫ

DictController.cs

Это файл обработки запросов и создания экземпляра модели. Каждая функция имеет соответствующее представление view, которое в итоге будет «вызвано».

Index: создается объект модели ContactsHolder и получается весь список контактов.

Add: переход на представление(страницу) Add

AddSave: после того, как в представлении Add вы нажмете кнопку «ок», выполняется эта функция, которая добавляет контакт в список и вызывает метод Insert, который в свою очередь сохраняет это в json. Возвращается на Index.

Update: Переходим по ссылке Update и далее мы находим по id нужный контакт и этому контакту присваиваем новый контакт. После того, как мы нажимаем кнопку «ок» вызывается UpdateSave.

UpdateSave: создает новый контакт с новыми данными и вызывается Update из модели, который удаляет старый и вставляет новый контакт с изменениями. Возвращается на Index.

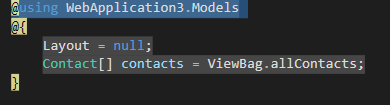
Delete: выводит все контакты и переходит на представление Delete. Когда мы нажимаем на ссылку Delete, вызывается функция DeleteSave.

DeleteSave: находит по id контакт и вызывает функцию Delete из модели, которая удаляет контакт из списка и переписывает файл.

ПРЕДСТАВЛЕНИЯ

Add: отправляет POST запрос на контроллер AddSave. Тут все просто.

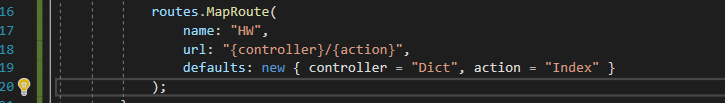
Delete: выводит список всех контактов. Ссылка Delete отправляет запрос на контрллер DeleteSave.

Index: ссылки Add и Delete, которые отправлсяют запрос на контроллеры Add и Delete соответственно. А так же список всех контактов. Чтобы этот список инициализировался при каждом переходе на эту страницу в верху кода видим это 

Собственно здесь и «достается» весь список из модели, который иниализируется в контроллере каждый раз, когда мы переходим на эту страницу.

Update: отправляет POST запрос на контроллер UpdateSave. Тут все просто.

**А теперь про то, как вся эта хуета работает.**

Файл RouteConfig выполняет маршрутизацию. Вот эта вещи указывает на то, какая страница будет стартовой при запуске проекта.

Контроллеры: При получении запроса система маршрутизации выбирает для обработки запроса нужный контроллер и передает ему данные запроса. Контроллер обрабатывает эти данные и посылает обратно результат обработки. Все запросы обрабатываются в контроллерах.

Представления нужны для формирования внешнего вида приложений. При нажатии на любую кнопку на представлении вызывается обработчик из контроллеров, который в свою очередь обрабатывает запрос с помощью модели.

ВАЖНО! Представление напоминает обычную страницу html. Здесь могут быть определены все стандартные элементы разметки html, здесь могут подключаться стили, скрипты. Но полноценной html-страницей представление все равно не является, потому что во время выполнения эти представления компилируются в сборки и уже затем используются для генерации html-страниц, которые видит пользователь в своем браузере.

ЛАБУ ТАК ЖЕ НУЖНО ОПУБЛИКОВАТЬ НА IIS УНИВЕРА, ТАК КАК МЫ ДЕЛАЛИ ЭТО ВО 2 лабе.