**Памятка по использованию .NET-шаблона**

**Уровень доступа к данным**

1. Скрипт базы, будучи запущенным в любой момент времени, должен исполняться без ошибок и создавать базу данных, имеющую актуальную структуру и содержащую минимально необходимые для корректной работы приложения данные (не тестовые!).
2. Не забывайте включать в начало скрипта строчку вида

alter database <dbname> set recovery simple,

чтобы не засорять наш сервер огромными журналами транзакций.

1. Перед первичным развертыванием базы нужно задать дефолтным юзерам admin и dev длинные уникальные пароли и прописать их хеши в скрипт базы данных. Брать пароли из ранее разработанных проектов недопустимо.
2. Всегда именуйте все ограничения, которые вы добавляете в базу (включая, но не ограничиваясь, первичные ключи, внешние ключи, ограничения уникальности, значения по умолчанию).
3. Не добавляйте лишние ограничения, которые не требуются для корректного функционирования приложения (например, принудительные ограничения длины строки вида nvarchar(200), значения по умолчанию и т.д.).
4. Название любой таблицы базы данных должно заканчиваться на английское существительное во множественном числе. Соответствующие названия сущностей в приложении должны записываться в единственном числе.
5. Не используйте в именовании таблиц или полей транслитерацию с русского, а также аббревиатуры (за исключением общеупотребительных). Не допускайте орфографических ошибок в именах.
6. Все целочисленные поля таблицы, которые в приложении представляются перечислениями, должны иметь суффикс Id. Соответствующие им свойства классов в приложении должны записываться без этого суффикса.
7. Для первичной генерации модели EF из существующей базы данных нужно исполнить в консоли менеджера пакетов (View > Other Windows > Package Manager Console) команду вида

Scaffold-DbContext 'Data Source=<ServerName>;Initial Catalog=<DbName>;Integrated security=True' Microsoft.EntityFrameworkCore.SqlServer -Context DefaultDbContext -OutputDir DbModels –StartupProject Dal -Project Dal –Force

где <ServerName> – адрес локального экземпляра SQL Server (например, localhost\SQLEXPRESS), <DbName> – название базы данных.

После генерации модели следует проверить корректность перевода названий таблиц во множественном числе в названия классов модели в единственном числе и при необходимости исправить именование классов модели. Кроме того, в файле Dal/DbModels/DefaultDbContext.cs следует в методе OnConfiguring заменить локальную строку подключения на Common.Configuration.SharedConfiguration.DbConnectionString.

1. В случае необходимости возврата вместе с основной сущностью связанных с ней данных из других таблиц, всегда возвращайте их в виде соответствующих сущностей или их сокращенных версий. Не допускайте дублирования свойств одной сущности в другой, если это не имеет под собой иной веской причины.
2. Не создавайте для разных наборов параметров фильтрации индивидуальные методы запроса данных из базы, если только это не обусловлено хитрой логикой работы конкретного запроса и/или необходимостью вызова хранимых процедур.
3. Избегайте материализации больших множеств объектов и/или их зависимостей. В частности, в большинстве случаев необходимая фильтрация наборов данных должна выполняться на стороне базы, а не на стороне контроллера или представления.

**UI**

1. Всегда добавляйте favicon на сайт, предназначенный для размещения в публичном доступе. Если эта картинка отсутствует, следует повесить задачу на её изготовление.
2. При выполнении окончательной публикации сайта на основном домене либо в иной момент, когда сайт можно считать готовым и доступным публично, удаляйте дефолтный файл robots.txt, запрещающий индексацию всего сайта (как с сервера, так и из решения Visual Studio).
3. При написании в явном виде ссылок на страницы или статические файлы всегда используйте пути, начинающиеся с тильды. Скрипты также должны корректно обрабатывать такие пути (с использованием функции pageUtils.resolvePath). Все пути к ресурсам в стилевых файлах должны быть записаны относительно местонахождения стилевого файла, а не корня домена.
4. Инструкции подключения стилей и скриптов не должны быть разбросаны по всей странице. Такие инструкции должны группироваться в общую секцию (секции) и рендериться один раз в начале или конце макета страницы.
5. Не следует использовать html-тег style, а также злоупотреблять вынесением стилей в атрибут style. Практически все стили, используемые в приложении, должны находиться в стилевых файлах, а не внутри html-разметки.
6. При подключении файлов стилей и скриптов следует добавлять тег-хелпер asp-append-version="true".
7. Не следует подключать сторонние библиотеки с использованием CDN, поскольку такие сервисы в любой момент могут быть заблокированы Роскомнадзором.
8. Не забывайте выставлять атрибут Authorize тем действиям и контроллерам, которые не должны быть доступны неавторизованным пользователям.
9. Не добавляйте атрибуты, связанные с MVC, напрямую в классы сущностей. Для этой цели предназначены модели в проекте UI.
10. Избегайте использования ViewBag и ViewData для передачи данных из контроллера в представление. В подавляющем большинстве случаев все необходимые представлению данные следует объединить в единый класс, экземпляр которого будет передаваться представлению в виде модели.
11. Поддерживайте настройки в файле appsettings.json в актуальном состоянии. Версия этого файла, находящаяся в репозитории, в любой момент времени должна обеспечивать возможность корректного запуска проекта на сервере, где он развернут.

**API**

1. Для каждого контроллера указывайте маршрут (RouteAttribute), начинающийся с сегмента v{version:apiVersion}.
2. Используйте ApiVersionRangeAttribute для указания множества версий API, поддерживаемых конкретным контроллером или действием. При этом все существующие версии API должны быть перечислены в порядке возрастания в статическом свойстве AllApiVersions класса ApiVersionRangeAttribute.
3. При построении API старайтесь использовать следующую типовую схему для структурирования контроллеров (на примере объекта типа Meeting – встреча):

|  |  |  |
| --- | --- | --- |
| **Маршрут** | **HTTP-метод** | **Описание** |
| Meetings | GET | Получение списка встреч |
| PUT | Создание новой встречи |
| Meetings/{id} | GET | Получение встречи с идентификатором id. |
| DELETE | Удаление встречи с идентификатором id. |
| PATCH | Изменение встречи с идентификатором id. |
| Meetings/{id}/Cancel | POST | Отменить встречу с идентификатором id. |

1. В составе решения находится проект ApiClientsCodeGenerator, предназначенный для генерации кода для взаимодействия с API. Для указания необходимой генератору информации о параметрах действий и свойствах моделей применяйте соответствующие атрибуты:

|  |  |
| --- | --- |
| **Атрибут** | **Описание** |
| RequiredAttribute | Указание обязательности параметра метода или свойства модели, имеющего ссылочный тип данных. |
| FromQueryAttribute | Указывает, что параметр действия должен передаваться в составе строки запроса. Обязателен для указания, если запрос GET или DELETE принимает на вход сложные типы данных. |
| FromBodyAttribute | Указывает, что параметр действия передается в качестве тела запроса. |
| DisableCodeGenerationAttribute | Указывает, что тип, свойство или действие не должны включаться в сгенерированный код. В отношении свойств класса также может использоваться JsonIgnoreAttribute. |
| GenericArgumentsNotNullAttribute | Указывает, что аргументы обобщенного типа или свойства, имеющего обобщенный тип, не могут принимать значение null. Часто используется вместе со свойствами, описывающими списки объектов. |
| SerializableInAndroidAttribute | Указывает, что при генерации кода для ОС Android данный класс должен рассматриваться как сериализуемый. Обычно используется для классов, экземпляры которых должны передаваться между различными экранами приложения. Если некоторый класс помечен данным атрибутом, им также должны быть помечены все типы, используемые внутри данного класса. |

1. Каждое действие контроллера должно возвращать либо непосредственно тип результата T, либо типы IActionResult<T>, Task<T>, Task<IActionResult<T>>. Действия, возвращающие иные типы, должны быть декорированы атрибутом DisableCodeGenerationAttribute.
2. В качестве типа возврата действия в большинстве случаев целесообразно использовать общий тип-обертку (ResponseWrapper или иной подобный), содержащий информацию о результате исполнения операции. Объекты этого типа желательно возвращать в том числе и при возникновении ошибок. Помните, что возврат пустого тела ответа при коде возврата меньше 300 приведет к ошибке десериализации на клиенте.
3. Давайте действиям контроллеров осмысленные имена, которые будет удобно использовать при работе с API со стороны клиента. Помните, что название метода не обязательно должно использоваться в маршруте.
4. Все классы, используемые в API, должны иметь различные имена, даже если они находятся в различных пространствах имен. Нарушение этого требования приведет к генерации некомпилируемого кода на языке Swift.

**Генератор кода серверной части**

1. Поддерживайте в актуальном состоянии конфигурационный файл генератора. Ранее добавленные описания сущностей должны объединяться с вносимыми правками для обеспечения простого использования генератора в будущем.

**Общее**

1. Корректный способ использования шаблона заключается в копировании всего решения как единого целого, а не в сборке нового решения из кусков шаблона.
2. Если требуется предоставить исходники проекта заказчику, следует предварительно удалить из решения и с диска папку CodeGeneration, а также данную памятку.