Оглавление

[Установка и развёртывание 2](#_Toc391883090)

[Требования 2](#_Toc391883091)

[Соединение с сервером БД 2](#_Toc391883092)

[Эксплуатация 2](#_Toc391883093)

[Основные возможности 2](#_Toc391883094)

[Краткое описание модулей 3](#_Toc391883095)

[Controller 3](#_Toc391883096)

[HomeController.cs 3](#_Toc391883097)

[Models 3](#_Toc391883098)

[ShortInfoModel.cs 3](#_Toc391883099)

[DetailedInfoModel.cs 4](#_Toc391883100)

[SortOptions.cs 4](#_Toc391883101)

[Views 4](#_Toc391883102)

[Index.cshtml 4](#_Toc391883103)

[ContactInfoPartialView.cshtml 4](#_Toc391883104)

[DetailedPartialView.cshtml 4](#_Toc391883105)

[Почему MVC 4](#_Toc391883106)

[А если «на пальцах»? 5](#_Toc391883107)

# Установка и развёртывание

## Требования

Для работы задания необходимы следующие компоненты:

* .NET Framework v. 4 (http://www.microsoft.com/ru-ru/download/details.aspx?id=17718).
* MS VS 2010 с установленным пакетом обновления для MVC 4 (взять пакет можно отсюда: <http://www.microsoft.com/ru-ru/download/details.aspx?id=30683>) или минимальный установочный компонент MVC 4 (<http://www.asp.net/mvc/mvc4>) вместе с IIS.
* MS SQL Server (мною использовался 2012).
* Если используется IIS, то движок Razor (.cshtml), который должен быть выбран при конфигурации сайта.
* Браузер (тестирование на кроссбраузерность не проходил, но на IE 11, Firefox и Chrome всё работает).

В дальнейшем полагается, что компоненты установлены и настроены.

Подготовка базы данных

Для создания и заполнения таблицы БД необходимо исполнить скрипты:

* CreateTable.sql,
* GenerateData.sql,

в папке ~\SQLScripts.

## Соединение с сервером БД

Для правильной работы приложения с БД необходимо прописать корректный connectionString в файле ~\ChaplyginMVCProject\Web.config.

Строка соединения, прописанная в переданном архиве проекта предназначена для соединения с SQL сервером в локальной сети с авторизацией через проверку подлинности Windows.

# Эксплуатация

## Основные возможности

Для входа на сайт необходимо пройти по ссылке каталог\_сайта/ или каталог\_сайта/Home/Index.

Сортировка выполняется щелчком по элементу шапки таблицы, по которому надо провести сортировку. Изначально сортировка происходи по полю ID.

Щелчок по ячейке ID открывает диалоговое окно с данными контракта (конкретно: контактные данные исполнителя) и окрашивает все ячейки соответствующей строки в голубой цвет.  
Щелчок по другому элементу открывает страницу с полным описанием контракта.

Сортировка и окно с подробной информацией вызывается методов GET, поэтому можно сохранить ссылку с требуемым построением/выбранным контрактом, либо перейти на страницу, указав параметры вручную.

# Краткое описание модулей

## Controller

### HomeController.cs

Центр бизнес-логики приложения на серверной стороне. 2 метода построения моделей:

1. Index – страница по умолчанию (View - Index.cshtml), вывод всех (см. примечание в коде файла) строк таблицы Contract, управление сортировкой (вх. параметр: модель SortOptions, по умолчанию - null).
2. RequestDetails – получение дополнительной информации по одному контракту, в зависимости от флага либо вся информация (View – DetailedPartialView.cshtml), либо только контактные данные (View – ContactInfoPartialView.cshtml). 404, если запрошенный контракт не найден в базе (вх. параметры: ИД контракта, флаг на запрос краткой или полной информации, по умолчанию - полная).

## Models

*Прошу обратить внимания, что по принципам MVC считается плохим тоном организовывать логику работы с моделью (обработку и отображение) внутри самого класса модели, хотя это и расходится с некоторыми основными принципами ООП.*

### ShortInfoModel.cs

Описание всех полей для отображения в кратком виде, включает:

* ID,
* номер контракта,
* дату подписания,
* итоговую сумму,
* ФИО исполнителя.

### DetailedInfoModel.cs

Наследник модели ShortInfoModel, используется для отображения дополнительной информации, включает в себя следующие дополнительные элементы:

* предмет договора,
* подписант,
* информация об исполнителе,
* контактные данные представителя.

### SortOptions.cs

Класс отвечающий за сортировку таблицы на представлении Index.cshtml. В готовом виде можно использовать для сортировки по убыванию/возрастанию, комбинированной сортировки.

## Views

### Index.cshtml

Начальная страница сайта, отображение таблицы контрактов, использует модель IEnumerable<ShortInfoModel>.

### ContactInfoPartialView.cshtml

Частичное представление, используется для вставок в другие представления, отображает панель с контактнымы данными. Использует модель DetailedInfoModel.

### DetailedPartialView.cshtml

Представление для отображения всей инофрмации о контракте (кроме ID). Может использоваться как частичное представление и встраиваться в рабочие страницы, по умолчанию работает как отдельное представление. Использует модель DetailedInfoModel.

# Почему MVC

Потому что мы имеем те же инструменты, что и на Web Forms, но при этом достигаем большего контроля над кодом, в плане сопровождения и тестирования за счёт разделения обязанностей, хотя на начальных этапах разработки код может сохранять избыточность.

Подробнее можно почитать здесь:

<http://metanit.com/sharp/mvc/1.1.php#mvcvswebforms>.

## А если «на пальцах»?

На примере обращения к начальной странице:

1. Посылаем запрос на http://адрес\_сайта/.
2. Т.к. HomeController – контроллёр по умолчанию, и явно контроллёр не выбран (иначе был бы адрес http://адрес\_сайта/имя\_контроллёра/) – обращаемся к нему.
3. Метод не задан, автоматически вызываем метод Index.
4. Метод заполняет (см. код) связанную со страницей Index.cshtml модель IEnumerable<ShortInfoModel> и начинает генерацию страницы.
5. Генерация, в общем, похожа на ту, которая происходит в Web Forms, однако у нас есть связь с моделью, как глобальным параметром для всего представления.
6. Страница сгенерирована и доступна клиенту.

Поэтому грубая техническая разница:

* Механизм передачи и обработки данных внутри сервера.
* Обработка событий.