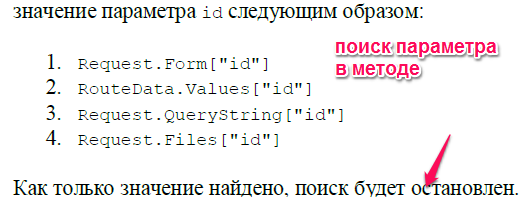
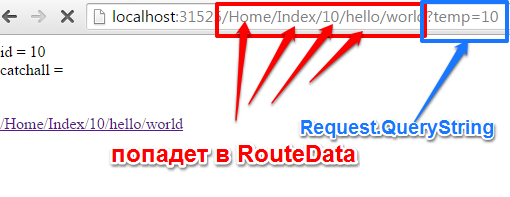


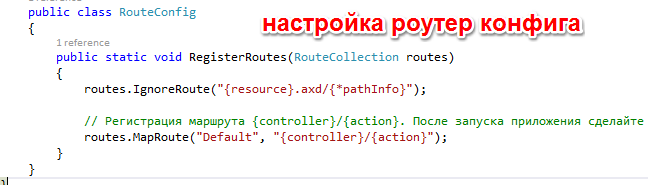
Пример использования

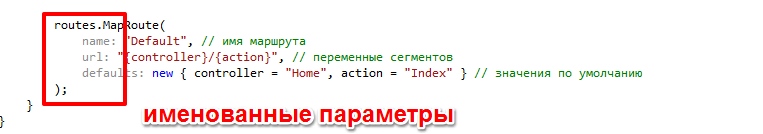


Поиск дополнительного параметра для метода (очередность)

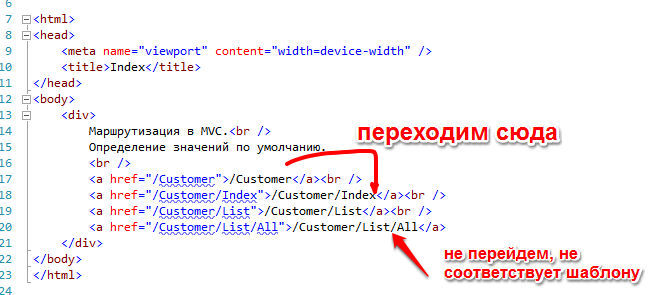




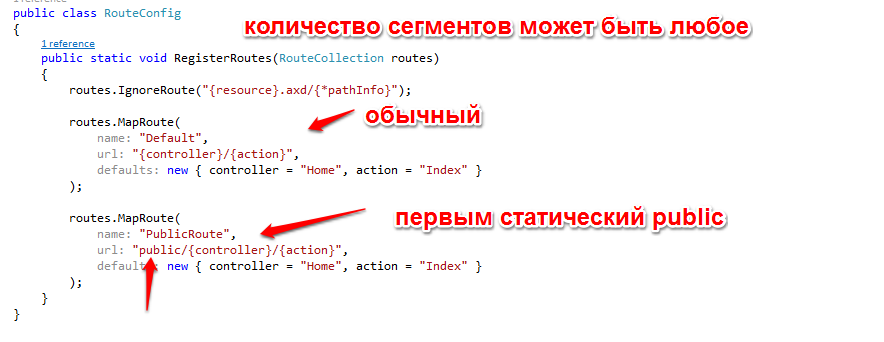


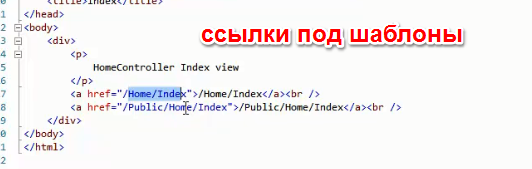


Сегменты, последний не соответствует так как содержит три сегмента

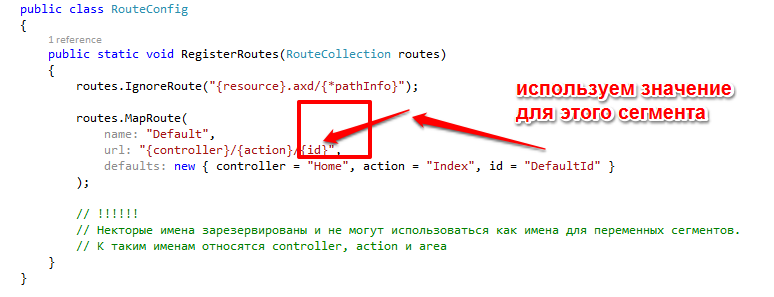


Статический сегмент





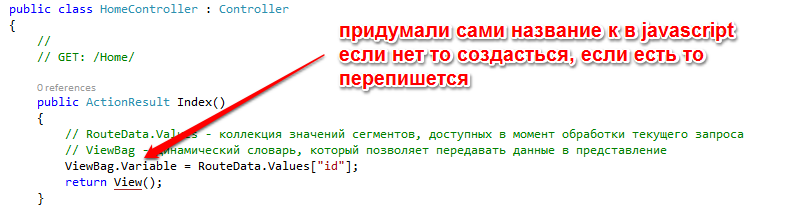
Получение значений от пользователя



ViewBag

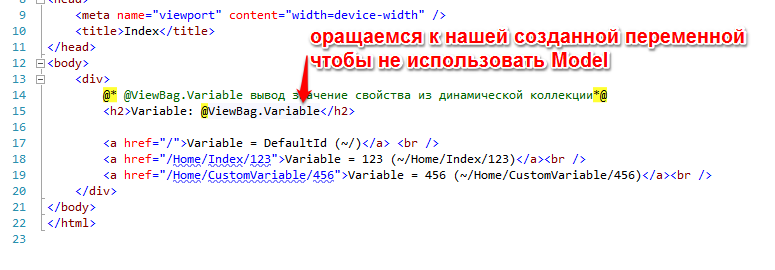
- динамический словарь, который позволяет передавать данные из конторолера в представление

RouteData.Values

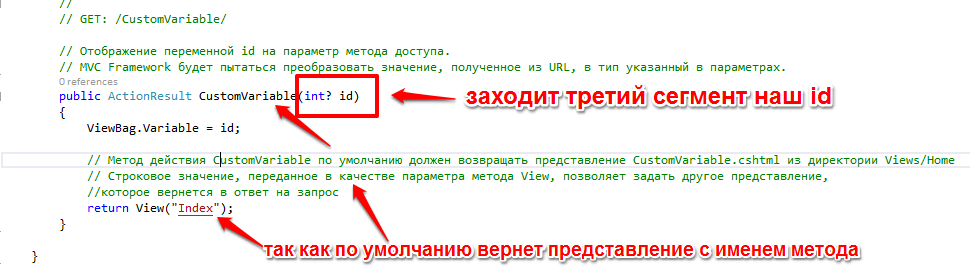
- коллекция значений сегментов, доступных в момент обработки текущего запроса

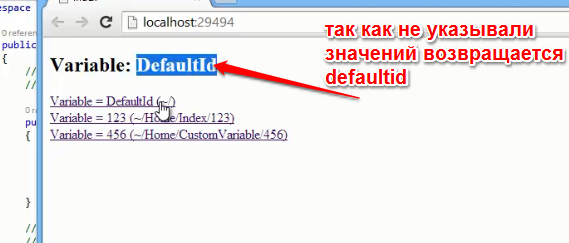
Или перегрузить этот же метод public ActionResult Index(int id) и не использовать RouteData

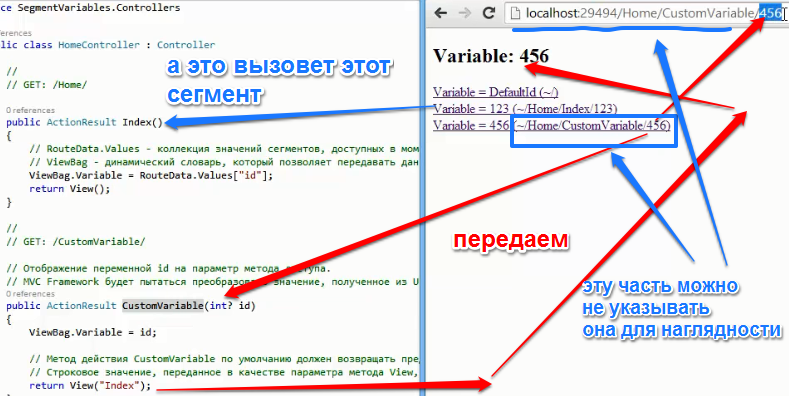
Считали данные



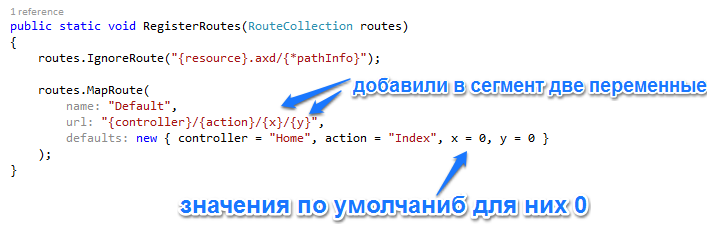
Второй способ

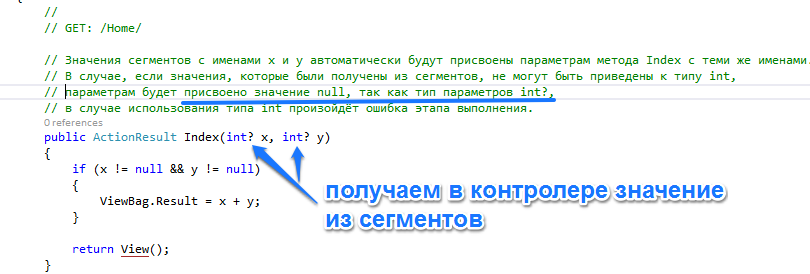




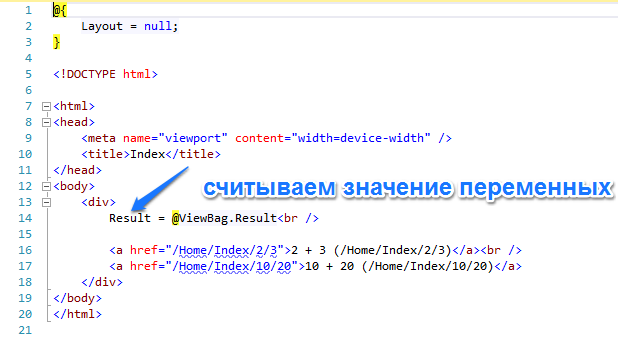


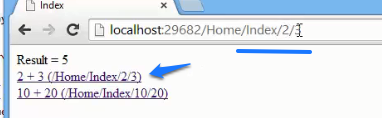
Увеличение сигментов





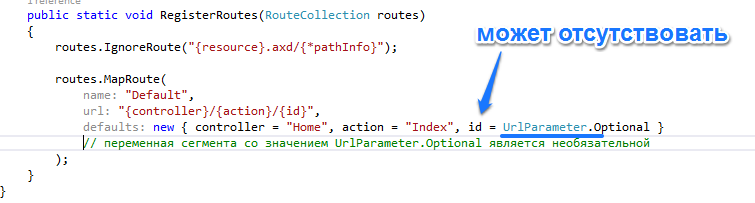
Не в переменную считывается а подставляется значение на это место (переменных ни каких нет)

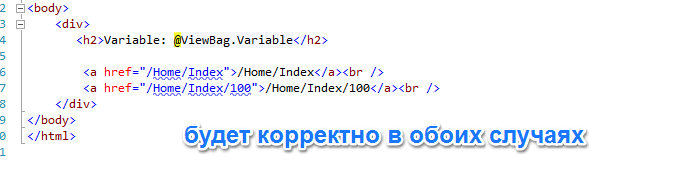




Опциональные сегменты, те которые могут отстутствовать

UrlParameter.Optional



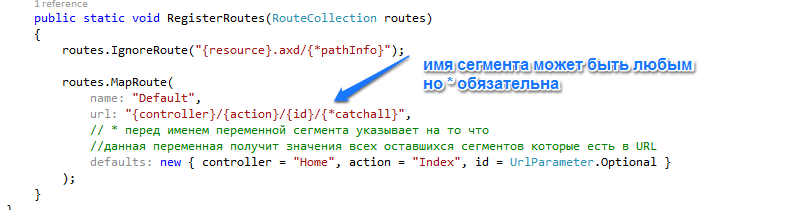


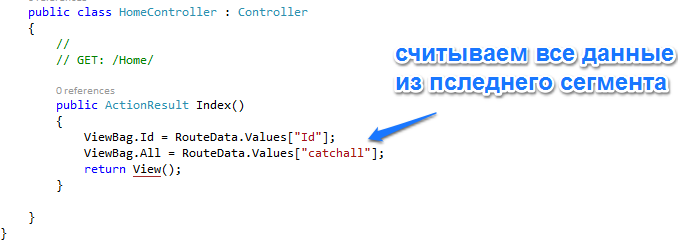
В первом отобразит null Во втором 100

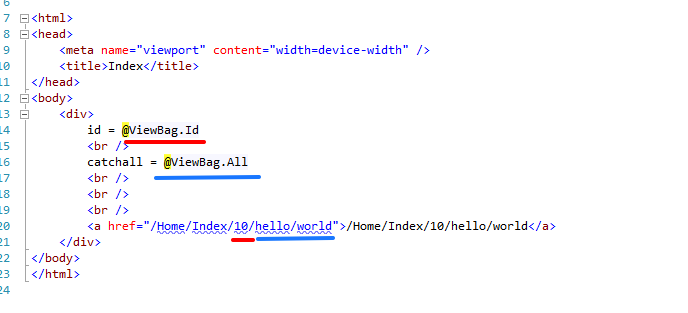
получит значения всех оставшихся сегментов которые есть в URL

\* перед именем переменной сегмента указывает на то что

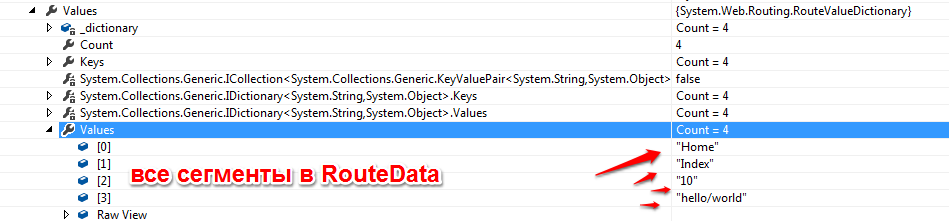
данная переменная получит значения всех оставшихся сегментов которые есть в URL





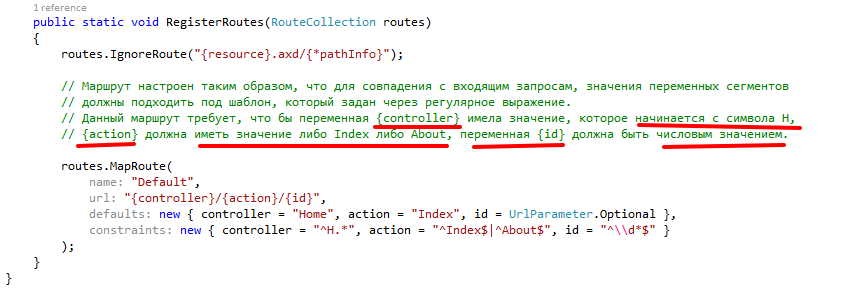


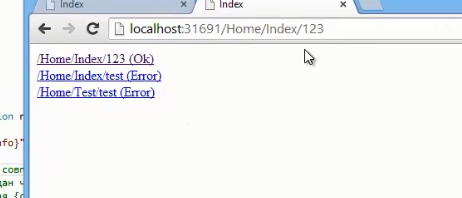


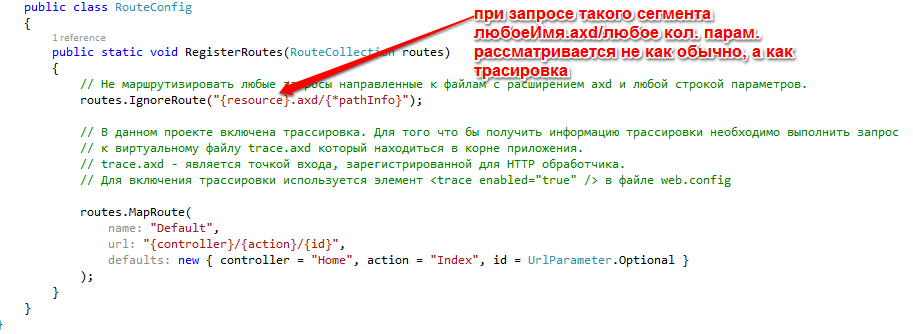


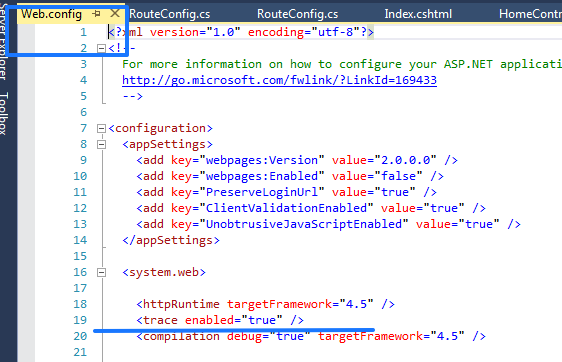
Ограничения на сигменты

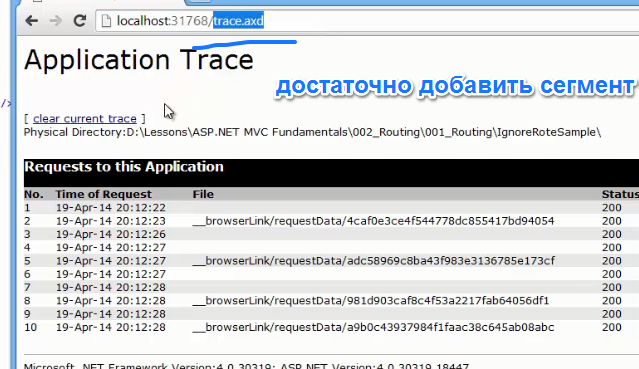
constraints

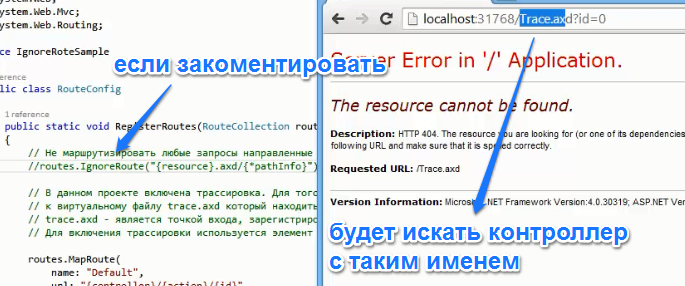




ТРАСИРОВКА  




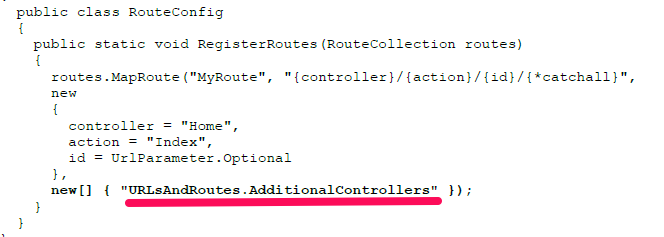




Пространство имен роутера

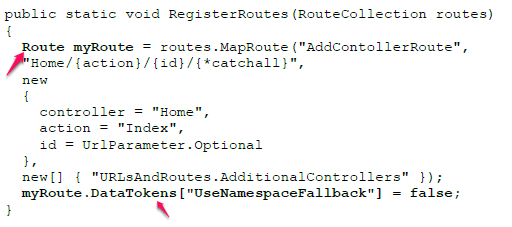
Мы выражаем пространства имен как массив строк, и в листинге мы сказали MVC смотреть сначала на пространство имен URLsAndRoutes.AdditionalControllers, прежде чем искать в другом месте.

Если подходящий контроллер не может быть найден в этом пространстве имен, тогда MVC повернется к своему поведению по умолчанию и будет искать совпадение в доступных пространствах имен.



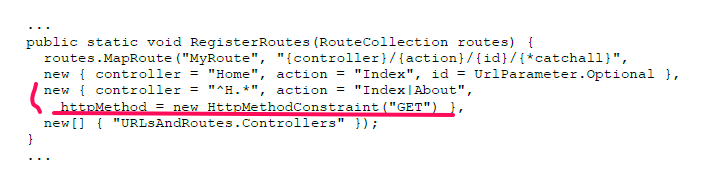
Мы можем сказать MVC фреймворку искать только в пространствах имен, которые мы указали.

Чтобы отключить поиск контроллеров в других пространствах имен, мы берем объект Route и устанавливаем ключ UseNamespaceFallback в свойстве коллекции DataTokens на false.

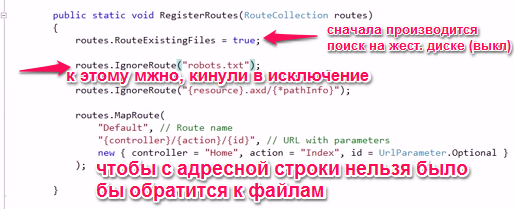


HttpMethodConstraint

Мы можем ограничить роуты, чтобы они соответствовали URL только тогда, когда он запрашивается с помощью конкретного HTTP метода

 Не имеет никакого значения, какое имя мы дали свойству, поскольку мы присвоили его экземпляру класса HttpMethodConstraint. В листинге мы назвали наше свойство для ограничения HttpMethod (как хотим так и называем главное в класс HttpMethodConstraint или только new HttpMethodConstraint("GET", "POST"), чтобы его можно было отличить от других видов ограничений.

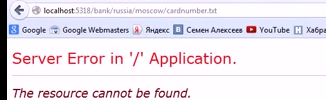
Чтобы с адресной строки нельзя было бы обратиться к файлу на диске





**Но не предоставляет доступ к файлам диска если они по сегментам подпадают по оределнные нами роутеры, если не один роут не подпадает то будет искать на диске, поэтому следует определить новый роут**

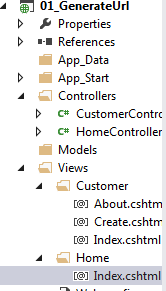


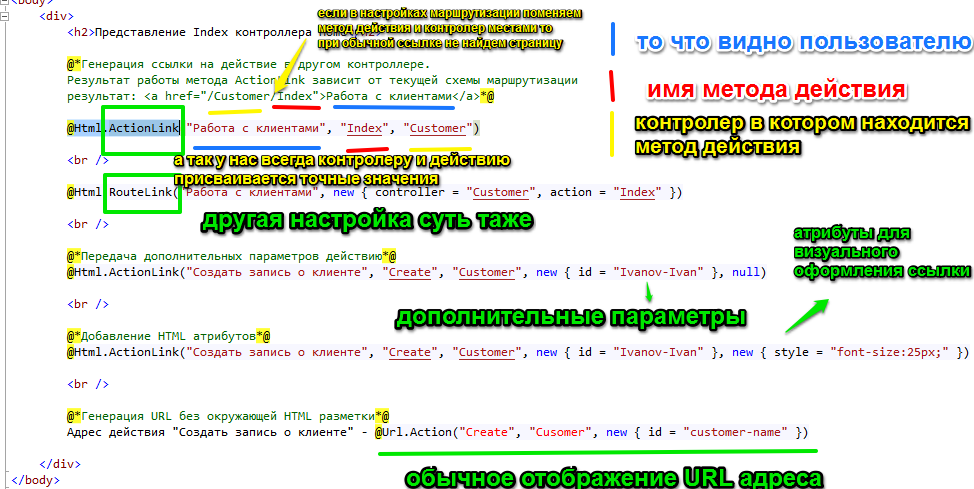


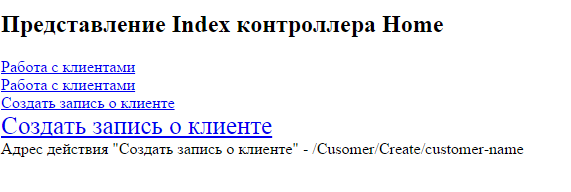
**Или помещать в** "App\_Data"

Генерация ссылок

@Html.ActionLink

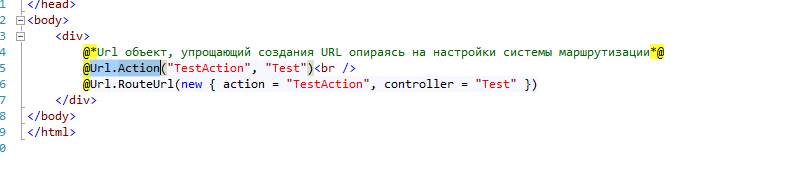




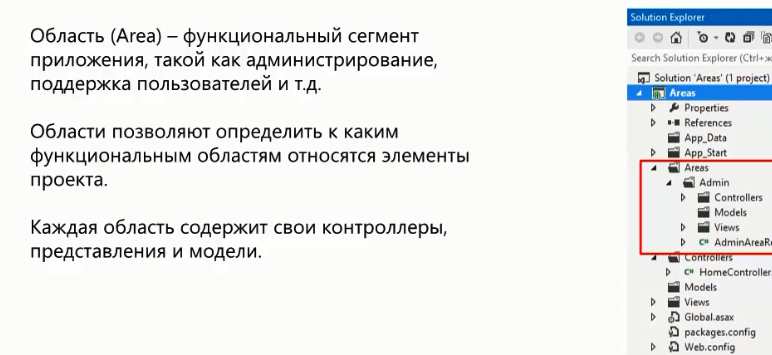


Создание URL адресов

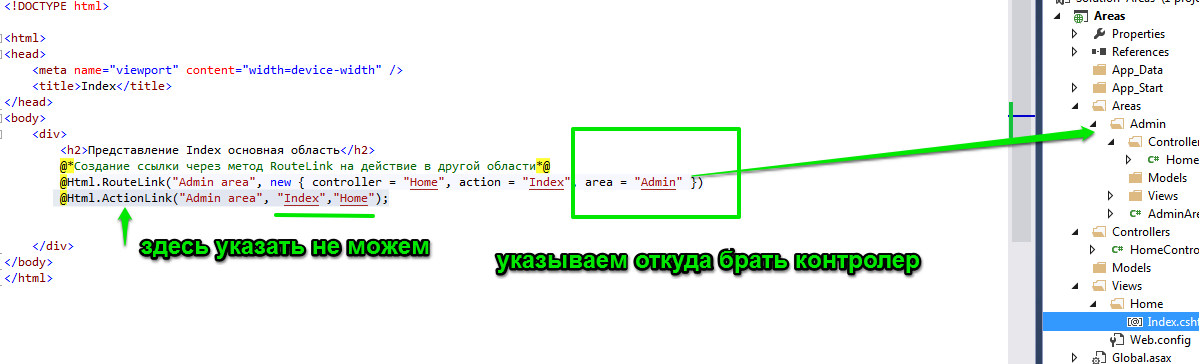
@Url объект, упрощающий создания URL опираясь на настройки системы маршрутизации







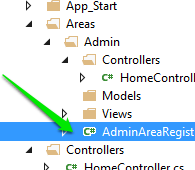
@Html.RouteLink("Admin area", new { controller = "Home", action = "Index", area = "Admin" })



Для @Html.ActionLink

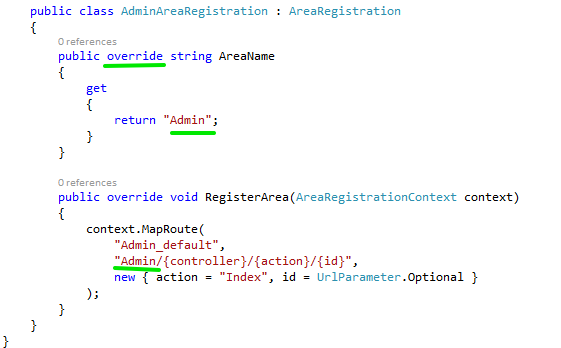
@Html.ActionLink("Admin area", "Index","Home",new {Area="Admin"},null);

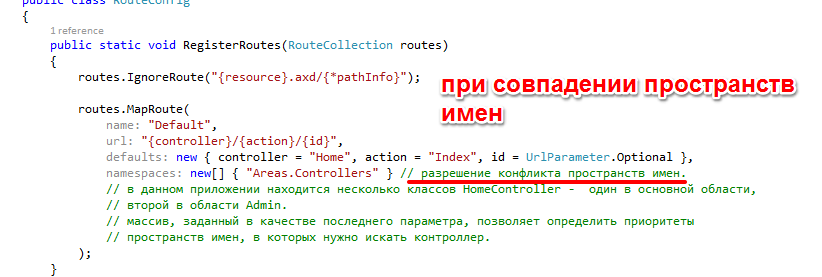
В каждой области

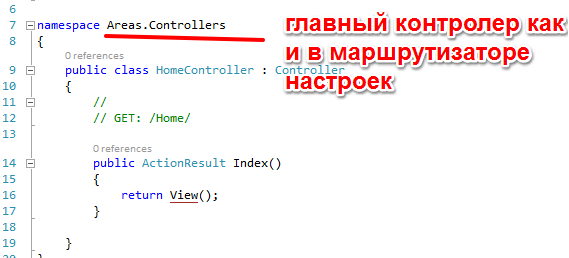


Файл маршрутизации со статическим сегментом

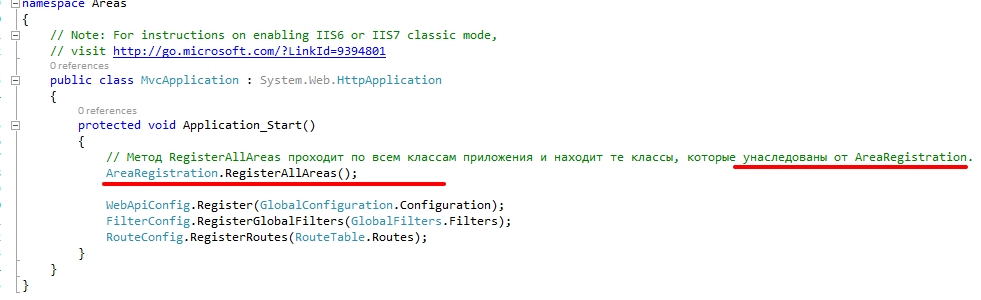
AreaRegistration



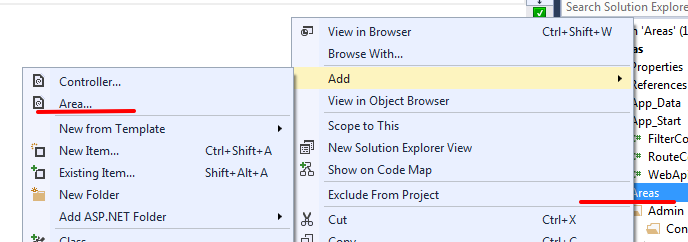




При начальном запуске страницы ищет все классы производные от AreaRegistration

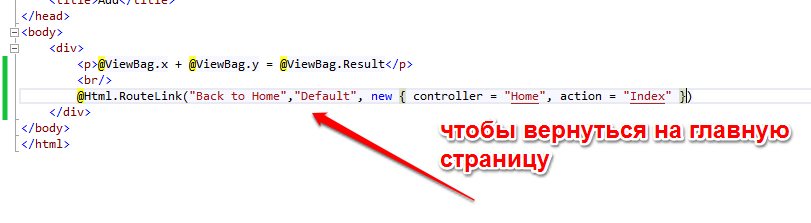


Добавить область



Вернуться с области на главную страницу

Указываем имя маршрутизатора



Есть несколько перегруженных версий метода View. Они соответствуют настройкам различных свойств объекта ViewResult, который создается. Например, вы можете переопределить макет, который используется представлением, явно указав альтернативу:

...

public ViewResult Index() {

return View("Index", "\_AlternateLayoutPage");

}

Если вы хотите отобразить конкретное представление, вы можете сделать это, если явно укажете путь к нему и обойдете фазу поиска. Вот пример:

public class ExampleController : Controller

{

public ViewResult Index()

{

return View("~/Views/Other/Index.cshtml");

}

public RedirectResult Redirect()

{

return Redirect("/Example/Index");

}

URL, на который вы хотите сделать перенаправление, выражается в виде строки и передается в

качестве параметра методу Redirect. Метод Redirect реализует временное перенаправление. Вы

можете создать постоянное перенаправление, используя метод RedirectPermanent, как показано в

листинге 15-21.

Листинг 15-21: Реализация постоянного перенаправления на литеральный URL

...

public RedirectResult Redirect() {

return RedirectPermanent("/Example/Index");

}

...

Совет

Если вы хотите, вы можете использовать перегруженную версию метода Redirect.

Она принимает параметр bool, который определяет, является ли перенаправление

постоянным или нет.