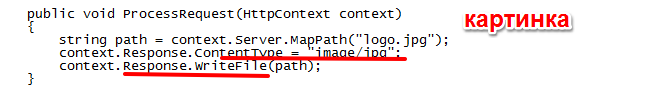
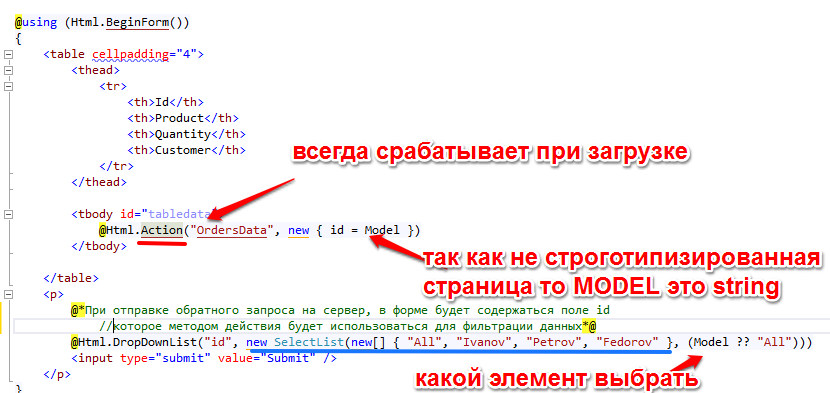


ContentТype ТЕКСТ

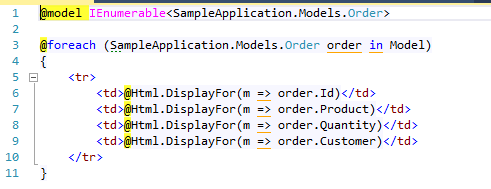


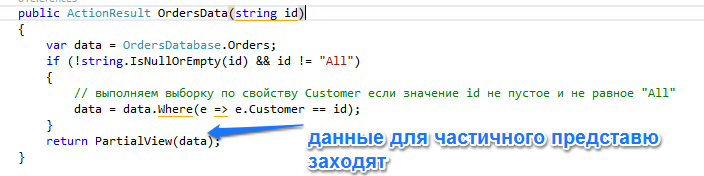
Картинка

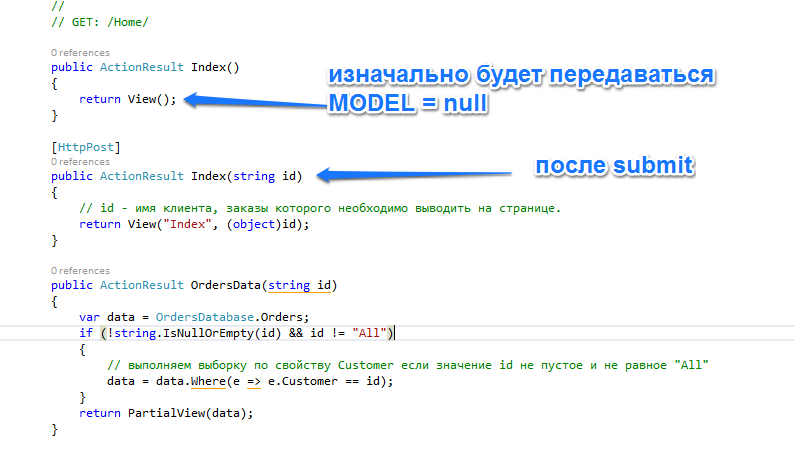




Частичное представление OrdersData

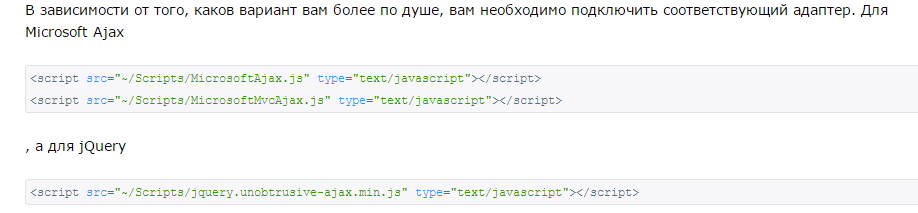


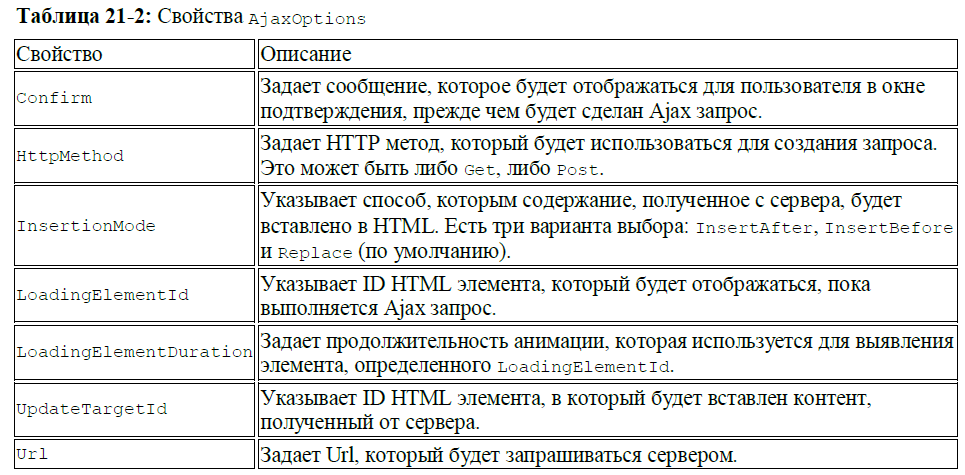


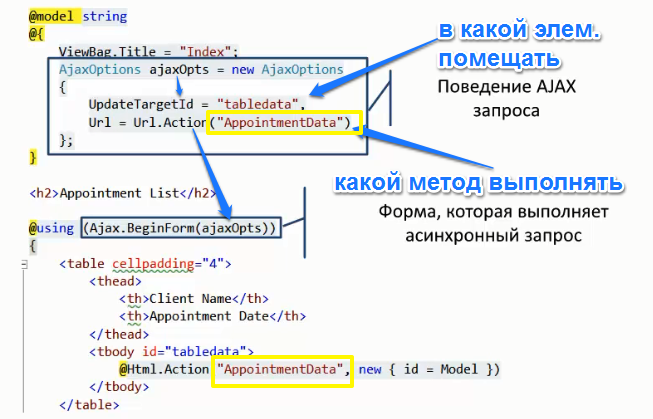


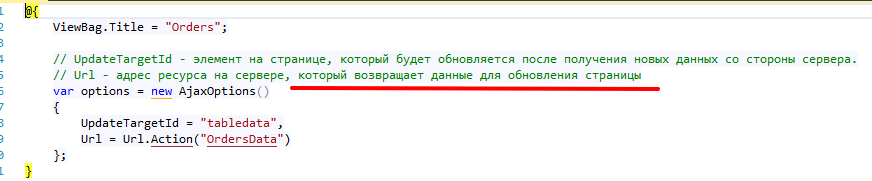


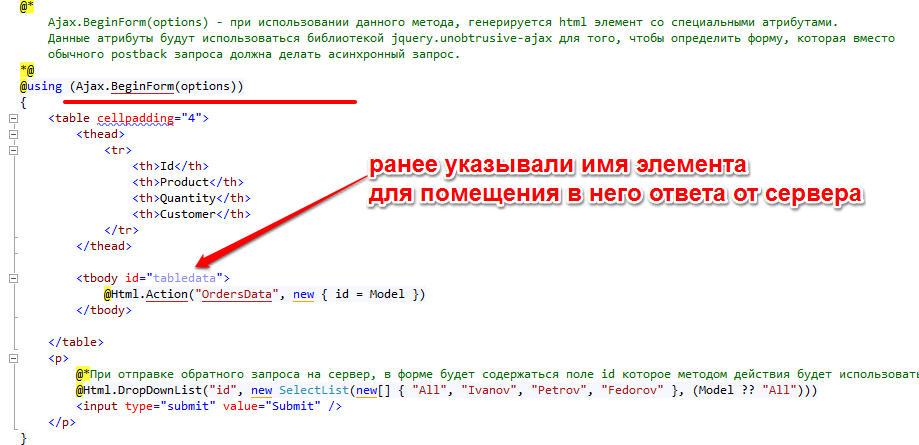
Начну с того, что MVC может работать с двумя вариантами Ajax библиотек (конечно я же имею ввиду с коробки, не более того) — jQuery и Microsoft Ajax. Чтобы знать, для какого адаптера создавать разметку, существует настройка в web.config UnobtrusiveJavaScriptEnabled и соответствующее значение true (для работы с jQuery) и false (для работы с Microsoft Ajax). Если же нам необходимо поменять значение только для одного представления, можно воспользоваться методом —@{Html.EnableUnobtrusiveJavaScript(bool);}. Хочу обратить внимание, что данная настройка влияет и на формирование валидационных данных на стороне клиент

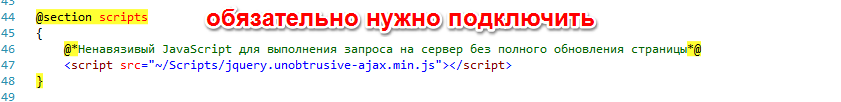




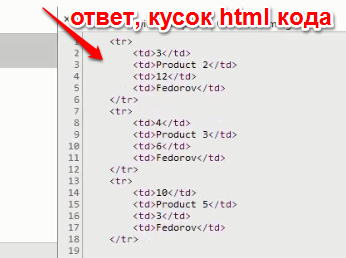








Ответ от сервера





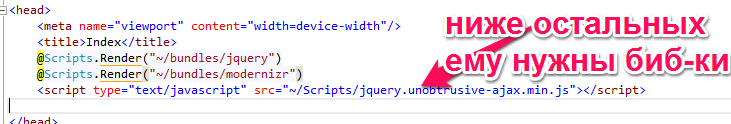
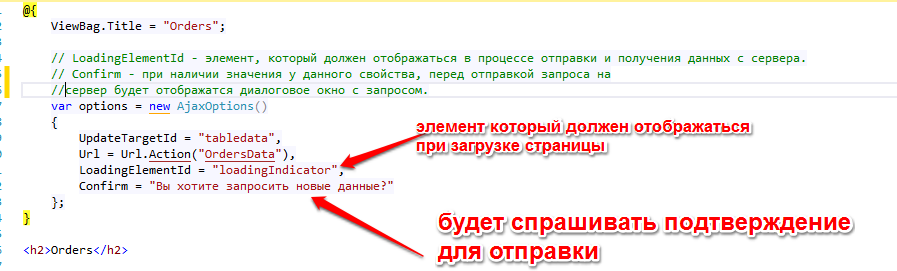
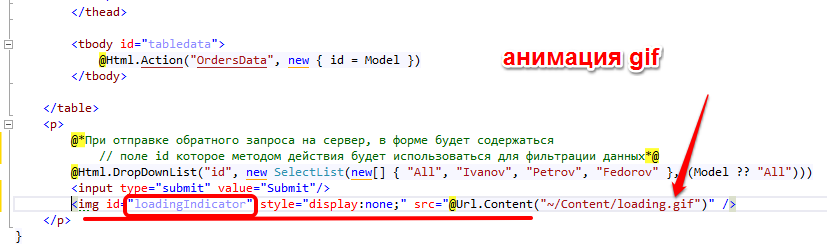


Иллюстрация индикатора запроса

Подтверждение запроса как правило нужно когда удаление из бд



Картинка в форме

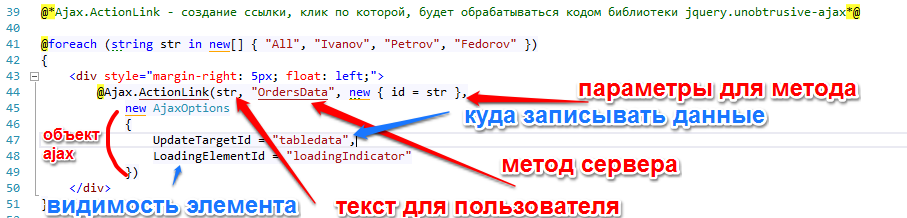


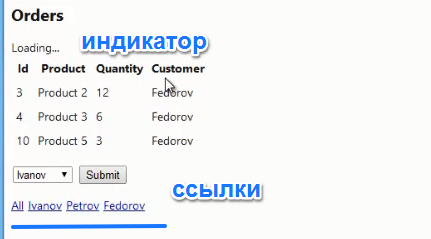


Ссылки для асинхронных запросов

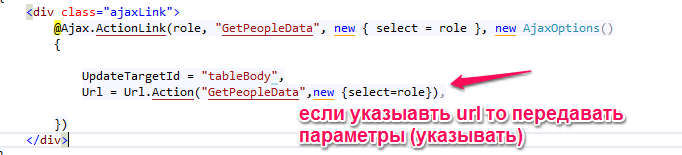
@Ajax.ActionLink







Внимание



Если отключен javascript EnableUnobtrusiveJavaScript



CallBack методы

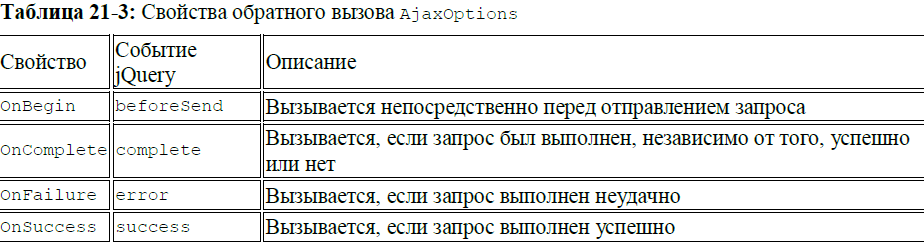
С помощью следующих свойств можно указать JavaScript функции, которые будут запускаться по мере выполнения запросов.

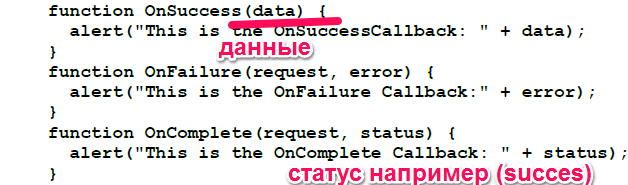
OnBegin - начало асинхронного запроса на сервер.

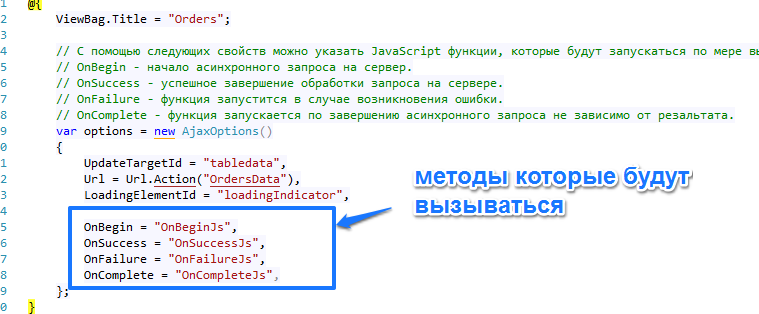
OnSuccess - успешное завершение обработки запроса на сервере.

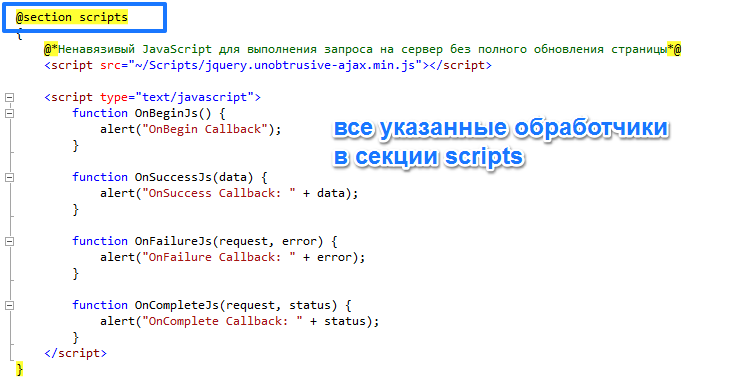
OnFailure - функция запустится в случае возникновения ошибки.

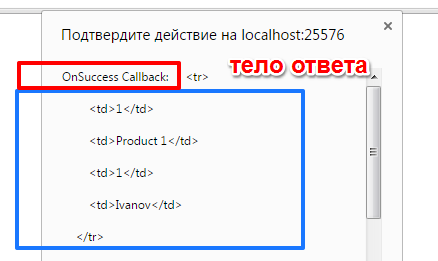
OnComplete - функция запускается по завершению асинхронного запроса не зависимо от резальтата.



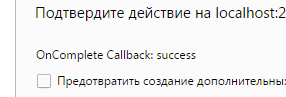






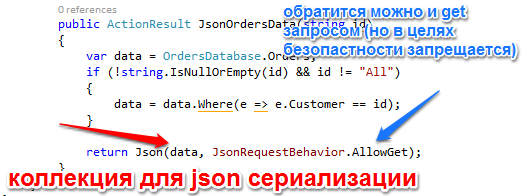


А потом

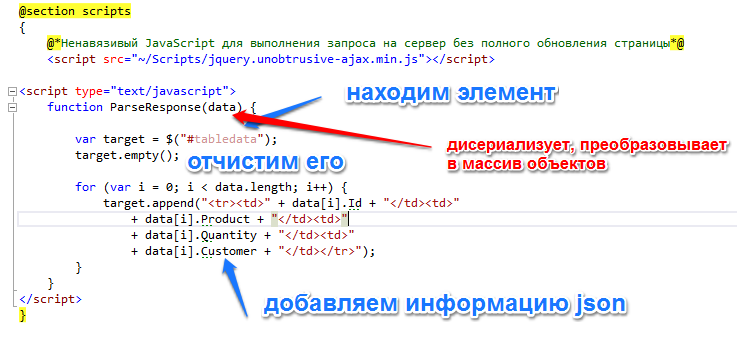


Получение JSON кода вместо HTML

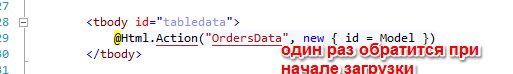
Добавили метод





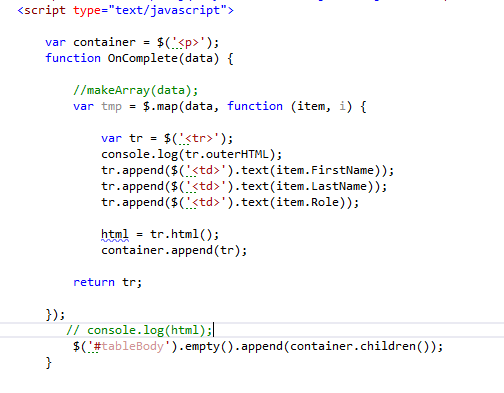


Обратится в начале потом будет подгружать в элемент этот методом JsonOrdersData

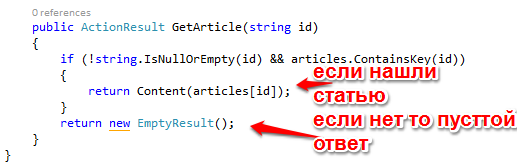


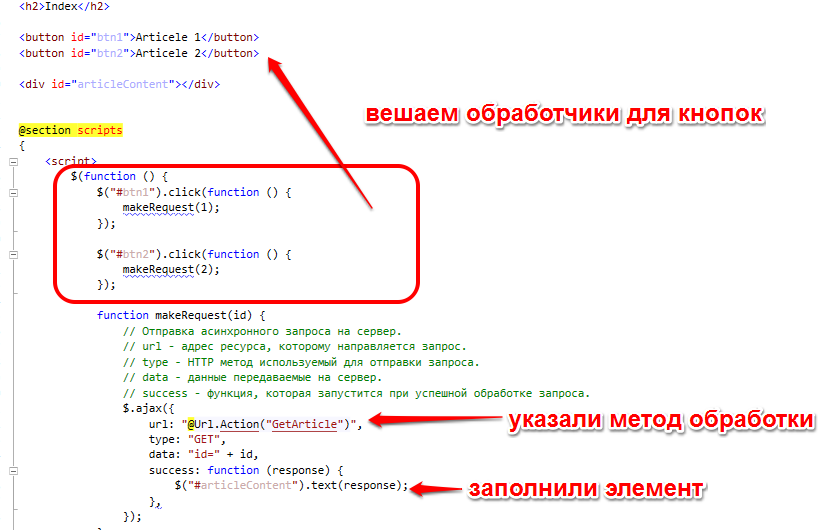


Реализация парсера json и добавления данных в тег

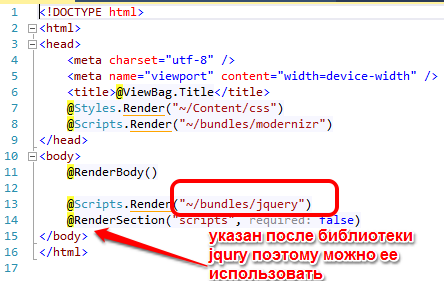


Ajax запрос без применения MVC с обычным JavaScript

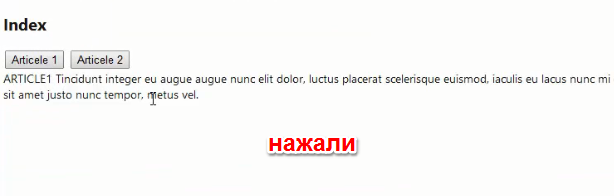




LAYOUT







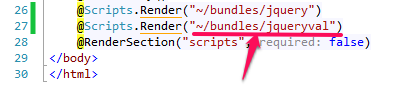
ОБЯЗАТЕЛЬНО ОБРАТИТЬ ВНИМАНИЕ

Если UnobtrusiveJavaScriptEnabled равен false

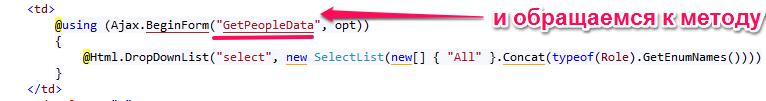
Html.EnableUnobtrusiveJavaScript(false);

Или библиотека Unobtrusive не подключена

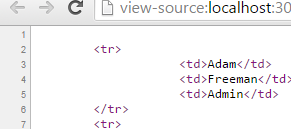
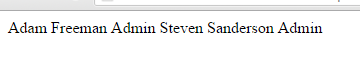




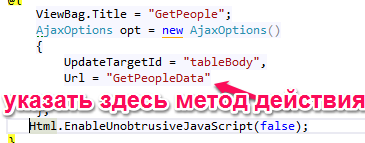
И обращаемся, указываем метод действия в форме



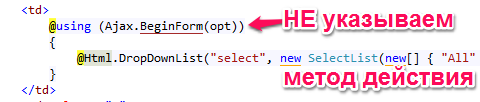
То результат метода действия заменит всю разметку так как вызовется дочерний метод "GetPeopleData" http://localhost:30450/Home/GetPeopleData



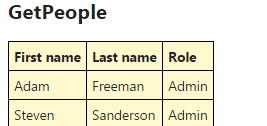
Исправить, нужно тогда указывать метод действия в



И в форме

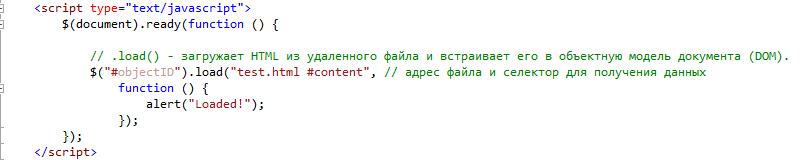


ТОГДА ОТОБРАЖАТЬСЯ БУДЕТ КОРРЕКТНО - <http://localhost:30450/Home/GetPeople>



НО РАБОТАТЬ БУДЕТ СИНХРОНО ПЕРЕГРУЖАЯ ВСЮ СТРАНИЦУ

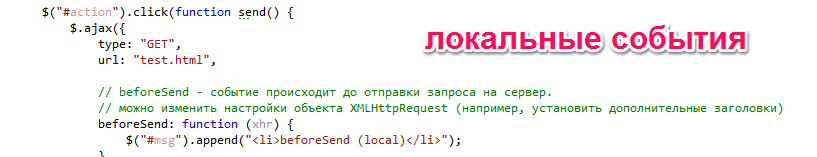
загружает HTML из удаленного файла



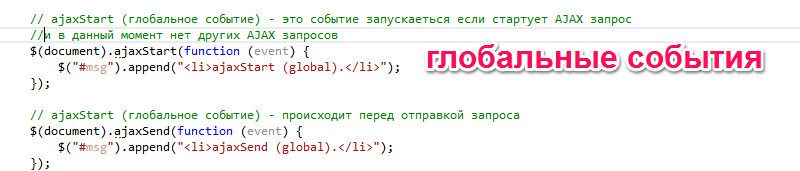
Загрузит вот сюда



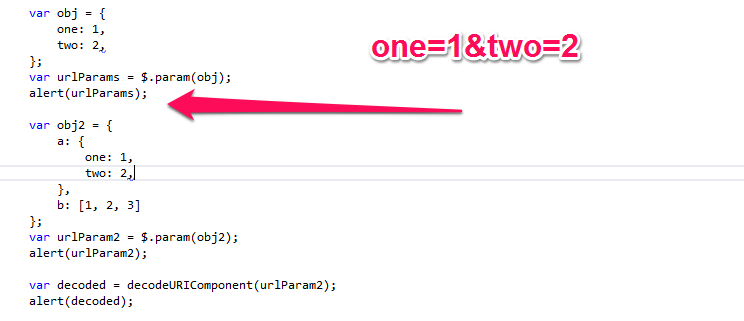
События AJAX



События после любого запроса



param() - Преобразует объект, массив или массив объектов в строку, которая соответствует формату передачи данных через url

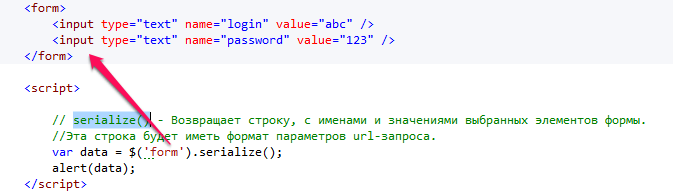


Второй alert()



serialize()

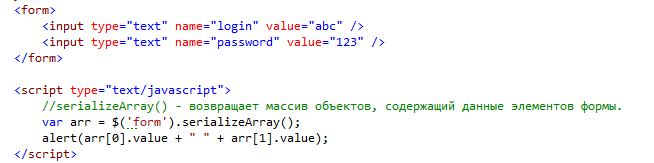
вытаскивает данные из формы и преобразует в строку





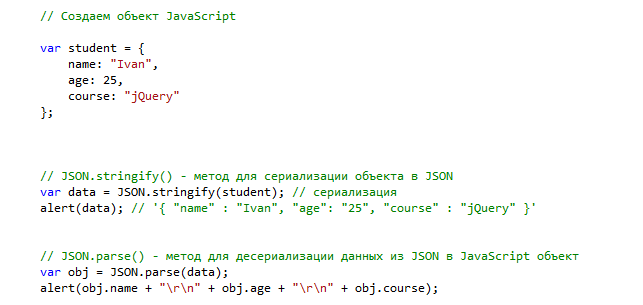
serializeArray()

возвращает массив объектов, содержащий данные элементов формы.

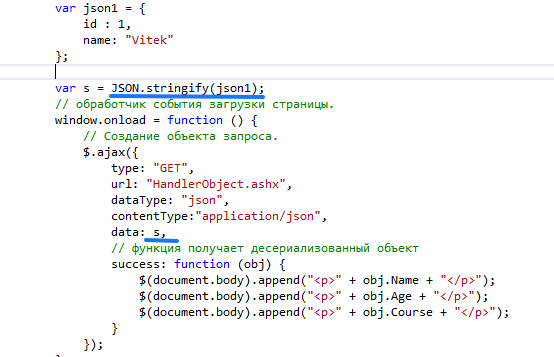


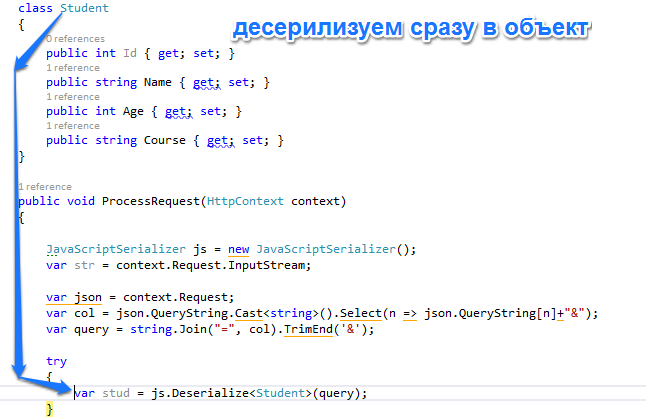


JSON.stringify() JSON.parse()

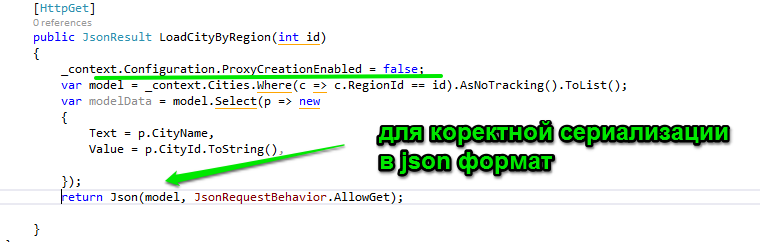


Отправляем json приходит в QueryString и его дисериализуем в класс

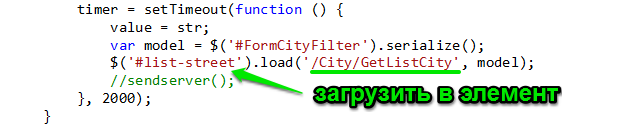




Для коректной сериализации Entity Framework objects to JSON



Загрузить в элемент с использованием ActionResult



PartialView to string

public static string RenderViewToString(string viewName, object model, ControllerContext controllerContext, ViewDataDictionary viewData = null, TempDataDictionary tempData = null)

{

if (viewData == null)

{

viewData = new ViewDataDictionary();

}

if (tempData == null)

{

tempData = new TempDataDictionary();

}

// assing model to the viewdata

viewData.Model = model;

using (var sw = new StringWriter())

{

// try to find the specified view

ViewEngineResult viewResult = ViewEngines.Engines.FindPartialView(controllerContext, viewName);

// create the associated context

ViewContext viewContext = new ViewContext(controllerContext, viewResult.View, viewData, tempData, sw);

// write the render view with the given context to the stringwriter

viewResult.View.Render(viewContext, sw);

viewResult.ViewEngine.ReleaseView(controllerContext, viewResult.View);

return sw.GetStringBuilder().ToString();

}

}

