Реализация динамического выпадающего списка

По мотивам статьи «ASP.NET MVC 3: реализация динамического выпадающего списка» для ASP.NET 5.

# ASP.NET MVC 3: реализация динамического выпадающего списка

Опубликовано: 29.07.2011 23:26

Обновлено: 27.08.2011 14:47

Источник: [Implementing Dynamic DropDownList in ASP.NET MVC 3 Framework](http://www.highoncoding.com/Articles/770_Implementing_Dynamic_DropDownList_in_ASP_NET_MVC_3_Framework.aspx)

Перевод: [ASP.NET MVC 3: реализация динамического выпадающего списка | Addicted to .Net](http://blog.vkuznetsov.ru/posts/2011/07/29/aspnet-mvc-3-realizaciya-dinamicheskogo-vypadayushhego-spiska)

## Аннотация

Динамические выпадающие списки (DropDownList) позволяют пользователю выбрать элемент из первого выпадающего списка, который заполняет второй выпадающий список на основе выбора в первом. В этой статье мы намерены продемонстрировать, как реализовать динамические выпадающие списки с использованием фреймворка ASP.NET MVC 3.

## Создание модели представления

Первым шагом является создание модели представления (ViewModel), которая представляет элементы для представления. Наша модель представления "вращается" вокруг отношений между категориями и продуктами. Пользователь будет выбирать категорию из выпадающего списка, который заполнит все соответствующие продукты в другом выпадающем списке. Реализация ниже показывает модель ArticleEditViewModel:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Web;

using System.Web.Mvc;

namespace MvcApplication1.ViewModels

{

public class ArticleEditViewModel

{

private List<SelectListItem> \_categories = new List<SelectListItem>();

private List<SelectListItem> \_products = new List<SelectListItem>();

[Required(ErrorMessage = "Please select a category")]

public string SelectedCategory { get; set; }

[Required(ErrorMessage = "Please select a product")]

public string SelectedProduct { get; set; }

public List<SelectListItem> Products

{

get { return \_products; }

}

public List<SelectListItem> Categories

{

get

{

\_categories.Add(new SelectListItem(){ Text = "Ruby", Value = "1"});

\_categories.Add(new SelectListItem(){ Text = "JavaScript",Value="2"});

return \_categories;

}

}

}

}

Как вы можете видеть, чтобы сохранить наш пример простым, мы жестко закодировали категории. В следующем разделе мы покажем, как заполнить выпадающий список категорий используя модель ArticleEditViewModel.

## Заполнение выпадающего списка категорий:

Действие (action) Index контроллера HomeController возвращает новую модель ArticleEditViewModel для представления. Представление использует свойство Categories модели ArticleEditViewModel для заполнения выпадающего списка. Действие Index приведено ниже:

public ActionResult Index()

{

return View(new ArticleEditViewModel());

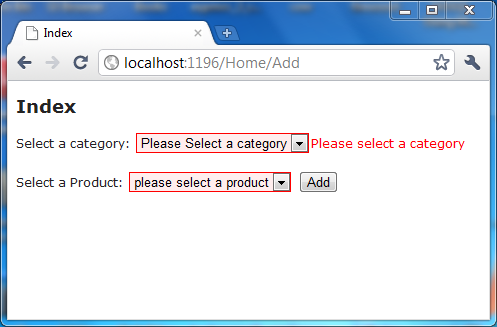
}

Выпадающий список заполняется кодом, представленным ниже:

<text>Select a category: </text> @Html.DropDownListFor(x => x.SelectedCategory, new SelectList(Model.Categories, "Value", "Text"), "Please Select a category")

@Html.ValidationMessageFor(x => x.SelectedCategory)

Выпадающий список привязан к свойству SelectedCategory (выбранная катерория) модели представления. Свойство SelectedCategory также декорировано атрибутом [Required] (обязательное), который гарантирует, что SelectedCategory не останется пустым или со значением по по умолчанию. Скриншот ниже показывает результат проверки, которая срабатывает, когда выпадающий список категорий остается со значением по умолчанию при завершении заполнения формы.



Проверка срабатывает из-за того, что опция по умолчанию "Select a category" ("выберите категорию") не имеет никакого значения, связанного с ней. ASP.NET MVC пытается связать пустую строку со свойством SelectedCategory, но не может, так как SelectedCategory отмечен атриюутом [Required].

## Получение списка продуктов и заполнение зависимого выпадающего списка

Нашей следующей задачей является получение выбранной категории, а затем и получение списка связанных продуктов с использованием categoryId. Это будет осуществляться Ajax-вызовом с помощью нашей любимой библиотеки JQuery. Реализация ниже показывает детали.

<script language="javascript" type="text/javascript">

    $(document).ready(function () {

        $("#SelectedCategory").change(function () {

            $.get('/Home/GetProductsForCategory/' + $(this).val(), function (response) {

                var products = $.evalJSON(response);

                var ddlSelectedProduct = $("#SelectedProduct");

                // clear all previous options

                $("#SelectedProduct > option").remove();

                // populate the products

                for (i = 0; i < products.length; i++) {

                    ddlSelectedProduct.append($("<option />").val(products[i].Value).text(products[i].Text));

                }

            });

        });

    });

</script>

Код выше вызывает действие **GetProductsForCategory** с categoryId в качестве входного параметра. GetProductsForCategory возвращает ответ в формате JSON, который представляет список продуктов. Перед заполнением выпадающего списка продуктами, мы очищаем выпадающий список, удаляя все существующие элементы. Действие GetProductsForCategory осуществляется следующим образом:

public string GetProductsForCategory(string id)

{

// get the products from the repository

var products = new List<SelectListItem>();

if (id == "1")

{

products.Add(new SelectListItem() {Text = "Introduction to Ruby", Value = "1"});

products.Add(new SelectListItem() {Text = "Ruby Unit Testing", Value = "2"});

}

else if(id == "2")

{

products.Add(new SelectListItem(){ Text = "JavaScript testing",Value = "1"});

products.Add(new SelectListItem() { Text = "JavaScript Ninja", Value = "2"});

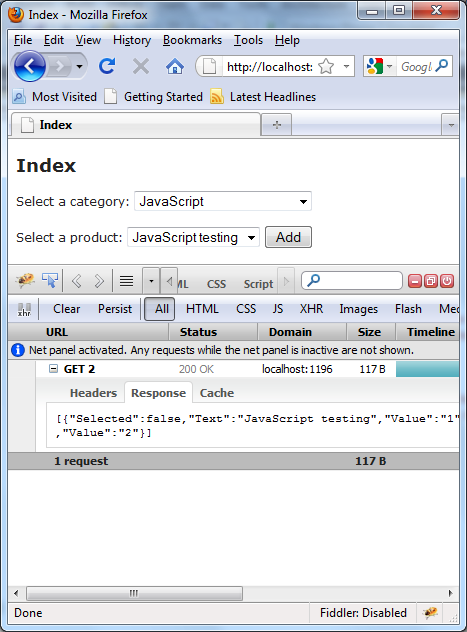
}

return new JavaScriptSerializer().Serialize(products);

}

Действие выше просто возвращает жестко закодированные значения, но в реальном приложении вы будет получать продукты из системы хранения данных.

Скриншот ниже показывает конечный результат:



## Вывод:

В этой статье мы научились создавать динамический выпадающий список в ASP.NET MVC. В будующем мы покажем, как упростить этот подход, создав HtmlHelper для динамических выпадающий списков.