



Mentoring: ASP.NET MVC

ASP.NET MVC: ViewModel

- Модели представлений
- Пакет AutoMapper
- Связывание коллекций
- Атрибуты Data annotations
- Display/Editor templates, UIHint.
- Частичное связывание
- Произвольное связывание моделей
- Самовалидирующиеся модели
- Удалённая валидация

Какими бывают Модели?

- Концепция ASP.NET MVC предполагает наличие двух ролей, реализуемых компонентом Модель (Model):
 - Доменная модель (domain model) — посредник при взаимодействии контроллеров и представлений с бизнес-логикой приложения;
 - Модель представления (view model) — предоставление и связывание данных в представлениях.
- Данные роли могут быть реализованы как в одном классе, так и в отдельных.

Модели представления

- Источником данных являются доменные модели и связывание с представлением;
- Набор полей отражает потребности представления и контроллера;
- Моделей представления, построенных на базе одной доменной модели, может быть много.

Пример — модель представления регистрации

```
public class RegisterViewModel
{
    /// <summary>
    /// E-mail пользователя
    /// </summary>
    public string Email { get; set; }

    /// <summary>
    /// Пароль
    /// </summary>
    public string Password { get; set; }

    /// <summary>
    /// Подтверждение пароля
    /// </summary>
    public string ConfirmPassword { get; set; }
}
```

Работа с моделями представления

```
public ActionResult Register(RegisterViewModel model)
{
    var user = new User
    {
        Email = model.Email,
        Password = model.Password,
    };

    var result = user.Register();
    if (result.Success)
    {
        return RedirectToAction("RegisterSuccess");
    }

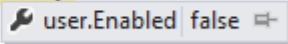
    ViewBag.ErrorMessage = "Ошибка: " + result.Description;
    return View(model);
}
```

Проблемы с конвертацией моделей

- (Слишком) много полей:

```
var user = new User
{
    Email           = model.Email,
    Password        = model.Password,
    Birthdate       = model.Birthdate,
    Country         = model.Country,
    State           = model.State,
    City            = model.City,
    Street          = model.Street,
    House           = model.House,
    Apartments      = model.Apartments,
    SubscribeForNews = model.SubscribeForNews,
    SubscribeForJobs = model.SubscribeForJobs,
};
```

- Новые поля:

```
if (user.Enabled)
{
    
}
```

NPM пакет AutoMapper

- Унифицирует механизм преобразования типов (доменная модель ↔ модель представления);
- Упрощает преобразование классов со сходным набором полей;
- Проверяет «забытые» поля.

```
Mapper.CreateMap<RegisterViewModel, User>()  
    .ForMember(dest => dest.Enabled, opt => opt.UseValue(true));  
  
var user = Mapper.Map<User>(model);
```

<http://automapper.org/>

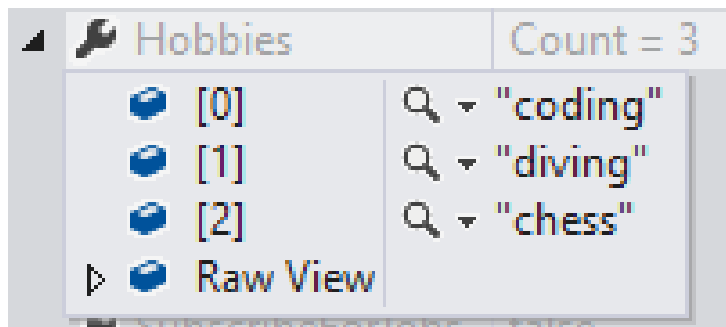
- Со стороны представления:

```
<input type="text" name="hobbies[0]" />  
<input type="text" name="hobbies[1]" />  
<input type="text" name="hobbies[2]" />
```

- Со стороны модели:

```
public List<string> Hobbies { get; set; }
```

- Со стороны контроллера:



- В качестве типа свойства модели может выступать любой класс `ICollection<T>`;
- Допускается сложное связывание (`user[0].Name`);
- Нумерация начинается с нуля и строго последовательна. Пропуск индекса обрывает связывание коллекции.

Атрибуты Data Annotations

- Предназначены для кастомизации представления и связывания модели, а также для её валидации;
- Сами по себе ничего не реализуют, а лишь описывают требования к свойству/классу;
- Поддерживаются рядом технологий:
 - ASP.NET MVC
 - Entity Framework
 - ...

Основные атрибуты

- Required
- Compare
- Range
- StringLength
- DataType
- Display
- DisplayFormat
- RegularExpression

[http://msdn.microsoft.com/ru-ru/library/system.componentmodel.dataannotations\(v=vs.110\).aspx](http://msdn.microsoft.com/ru-ru/library/system.componentmodel.dataannotations(v=vs.110).aspx)

Использование DisplayFor/EditorFor

- Вспомогательный метод DisplayFor позволяет отобразить поле модели, используя т.н. шаблон отображения;
- Для создания собственного частичного представления в качестве шаблона отображения его необходимо поместить в каталог Views\Shared\DisplayTemplates;
- Имя и модель представления должны совпадать с именем модели.
- Для метода EditorFor используется каталог Views\Shares\EditorTemplates.

- [UHint] — позволяет указать произвольный шаблон отображения/редактирования для указанного свойства модели.
- [Bind] — частичное связывание.
- ModelBinders — произвольное связывание.
- [Remote] — удалённая валидация.
- IValidateableObject — самовалидация модели.

Спасибо за внимание!

Контактная информация:

Дмитрий Верескун

Инструктор

EPAM Systems, Inc.

Адрес: Саратов, Рахова, 181

Email: Dmitry_Vereskun@epam.com

<http://www.epam.com>