Highly maintainable UI with ASP.NET MVC

Cezary Piątek

ailleron
your digital wings

Leasing Front Office

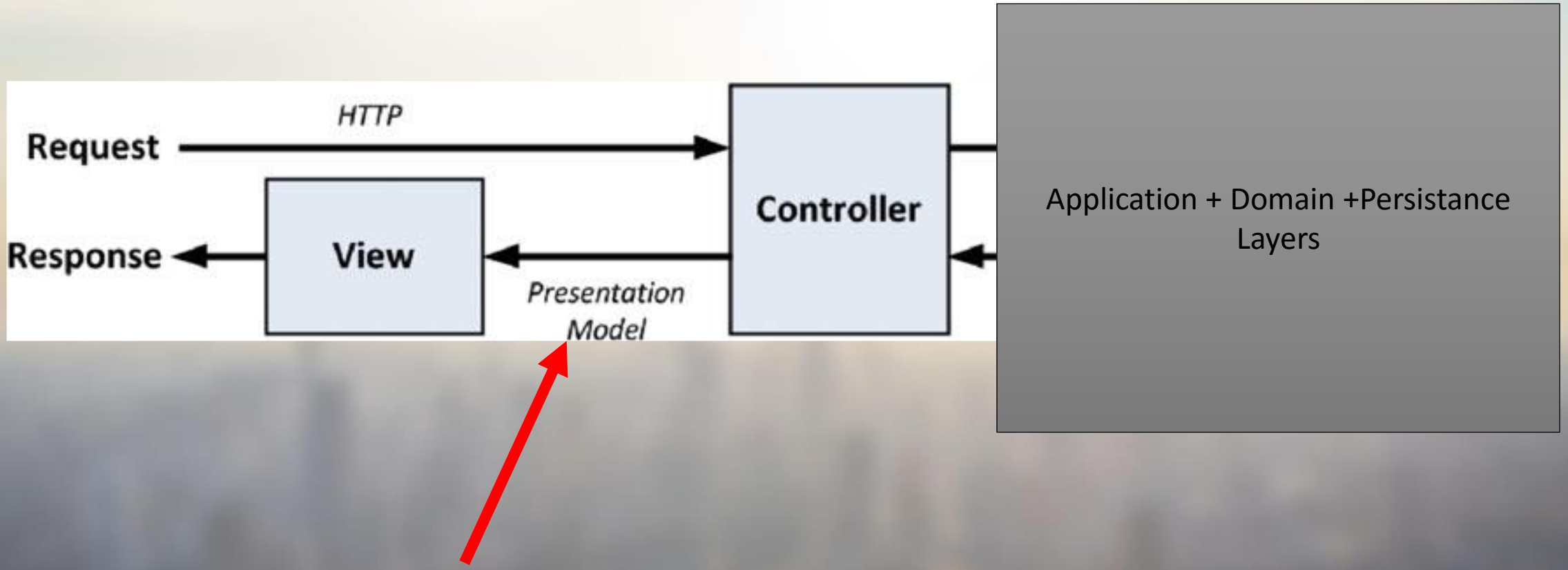
Controllers	**60**
Actions	1056
Views	956
Custom Jquery Widget	65

Agenda

1. Model
2. View
3. Controller
4. Directory Structure
5. Best practices

Model

Where is the model?



Data Clumps

```
public ActionResult NewOrder()
{
    var viewModel = this.orderService.NewOrder();
    ViewData["AvailableDiscounts"] =  new List<string> {"10%", "20%","25%"};
    ViewData["AvailableShippingCountries"] =  new List<string> {"Poland", "France","Germany"};
    return View("NewOrderForm", viewModel);
}
```

```
@Html.DropDownListFor(x=>x.Discount,
    new List<SelectListItem>((ViewData["AvailableDiscounts"] as List<string>).Select(x=> new SelectListItem() {Text = x})))

@Html.DropDownListFor(x=>x.ShippingCountry,
    new List<SelectListItem>((ViewData["AvailableShippingCountries"] as List<string>).Select(x=> new SelectListItem() {Text = x})))
```

Data Clumps

```
public class DropdownVM
{
    public string Selected { get; set; }
    public List<string> AvailableOptions { get; set; }
}

public class NewOrderViewModel
{
    public DropdownVM Discount { get; set; }
    public DropdownVM ShippingCountry { get; set; }
}
```

```
@Html.EditorFor(x => x.Discount)
@Html.EditorFor(x=>x.ShippingCountry)
```

View

ALWAYS STRONGLY TYPED

```
11  @using (Html.BeginForm("UpdateMember", "User"))
12  {
13
14      @Html.Editor("FullName")
15
16      @Html.ActionLink("Click Me", "GetDetails", "User")
17  }
```

```
10
11  @using (Html.BeginForm<UserController>(c=>c.UpdateMember(null)))
12  {
13
14      @Html.EditorFor(x=>x.FullName)
15
16      @(Html.ActionLink<UserController>(c=>c.GetDetails(), "ClickMe"))
17  }
18
```

ALWAYS STRONGLY TYPED

Mvc4Futures 4.0.20710

ASP.NET MVC Futures includes unsupported prototype features for ASP.NET MVC, from the MVC team.

To install Mvc4Futures, run the following command in the Package Manager Console

```
PM> Install-Package Mvc4Futures
```

Simple Form

Ankieta wstępna

* - pola wymagane

Podaj dane firmy

NIP: **PL7964973363**

Nazwa firmy *:

Kod pocztowy *: 00-000

Branża:

Przedział obrotów:

Podaj dane osoby kontaktowej

Imię:

Nazwisko:

Telefon komórkowy: 48

E-mail:

Informacje dodatkowe

Notatka:

Simple Form

```
@using (Html.BeginForm())
{
    <div class="form-horizontal">
        <h4>Podaj dane firmy</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)
        @Html.HiddenFor(model => model.Version)

        <div class="form-group">
            @Html.LabelFor(model => model.Nip, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Nip, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Nip, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.FullName, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.FullName, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.FullName, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ZipCode, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ZipCode, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ZipCode, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ContactFirstName, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.ContactFirstName, new { htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.ContactFirstName, "", new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.ContactLastName, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
```

```
9   @using (Html.BeginForm())
10  {
11      <div class="form-horizontal">
12          <h4>Podaj dane firmy</h4>
13          <hr />
14          @Html.ValidationSummary(true, "", new { @class = "text-danger" })
15          @Html.HiddenFor(model => model.Id)
16          @Html.HiddenFor(model => model.Version)
17
18          <div class="form-group">
19              @Html.LabelFor(model => model.Nip, htmlAttributes: new { @class = "control-label col-md-2" })
20              <div class="col-md-10">
21                  @Html.EditorFor(model => model.Nip, new { htmlAttributes = new { @class = "form-control" } })
22                  @Html.ValidationMessageFor(model => model.Nip, "", new { @class = "text-danger" })
23              </div>
24          </div>
25
26          <div class="form-group">
27              @Html.LabelFor(model => model.FullName, htmlAttributes: new { @class = "control-label col-md-2" })
28              <div class="col-md-10">
29                  @Html.EditorFor(model => model.FullName, new { htmlAttributes = new { @class = "form-control" } })
30                  @Html.ValidationMessageFor(model => model.FullName, "", new { @class = "text-danger" })
31              </div>
32          </div>
33
34          <div class="form-group">
35              @Html.LabelFor(model => model.ZipCode, htmlAttributes: new { @class = "control-label col-md-2" })
36              <div class="col-md-10">
37                  @Html.EditorFor(model => model.ZipCode, new { htmlAttributes = new { @class = "form-control" } })
38                  @Html.ValidationMessageFor(model => model.ZipCode, "", new { @class = "text-danger" })
39              </div>
40          </div>
41
42          <div class="form-group">
43              @Html.LabelFor(model => model.ContactFirstName, htmlAttributes: new { @class = "control-label col-md-2" })
44              <div class="col-md-10">
45                  @Html.EditorFor(model => model.ContactFirstName, new { htmlAttributes = new { @class = "form-control" } })
46                  @Html.ValidationMessageFor(model => model.ContactFirstName, "", new { @class = "text-danger" })
47              </div>
48          </div>
49
50          <div class="form-group">
51              @Html.LabelFor(model => model.ContactLastName, htmlAttributes: new { @class = "control-label col-md-2" })
52              <div class="col-md-10">
```

Simple Form

```
9   @helper EditorLineTemplate(IHtmlString label, IHtmlString editor, IHtmlString validationMessage)
10  {
11      <div class="form-group">
12          @label
13          <div class="col-md-10">
14              @editor
15              @validationMessage
16          </div>
17      </div>
18  }
```

```
public static IHtmlString EditorLineFor<TModel, TField>(this HtmlHelper<TModel> htmlHelper, Expression<Func<TModel, TField>> field)
{
    var label = htmlHelper.LabelFor(field);
    var editor = htmlHelper.EditorFor(field);
    var validationMessage = htmlHelper.ValidationMessageFor(field);
    return Templates.EditorLineTemplate(label, editor, validationMessage);
}
```

Simple Form

```
15  @using (Ajax.BeginForm<Case1Controller>(c => c.Save(null), Case1UIElementsIds.CreateCaseForm, "PageBody"))
16  {
17      using (Html.CreateFormSection("Podaj dane firmy"))
18      {
19          @Html.DisplayLineFor(m => m.Nip)
20          @Html.EditorLineFor(m => m.FullName)
21          @Html.EditorLineFor(m => m.ZipCode)
22          @Html.EditorLineFor(m => m.Industry)
23          @Html.EditorLineFor(m => m.TurnoverLevel)
24      }
25
26      using (Html.CreateFormSection("Podaj dane osoby kontaktowej"))
27      {
28          @Html.EditorLineFor(m => m.ContactFirstName)
29          @Html.EditorLineFor(m => m.ContactLastName)
30          @Html.EditorLineFor(m => m.ContactPhoneNumber)
31          @Html.EditorLineFor(m => m.ContactEmail)
32      }
33
34      using (Html.CreateFormSection("Informacje dodatkowe"))
35      {
36          @Html.EditorLineFor(m => m.Note)
37      }
38  }
```

Simple Form – TagHelper (ASP.NET Core)

```
8  <form asp-controller="Case1" asp-action="Update">
9      <formsection title="Podaj dane firmy">
10         <displayline for="Nip" />
11         <editorline for="FullName" />
12         <editorline for="ZipCode" />
13         <editorline for="Industry" />
14         <editorline for="TurnoverLevel" />
15     </formsection>
16     <formsection title="Podaj dane osoby kontaktowej">
17         <editorline for="ContactFirstName" />
18         <editorline for="ContactLastName" />
19         <editorline for="ContactPhoneNumber" />
20         <editorline for="ContactEmail" />
21     </formsection>
22     <formsection title="Informacje dodatkowe">
23         <editorline for="Note" />
24     </formsection>
25 </form>
```

Html Form

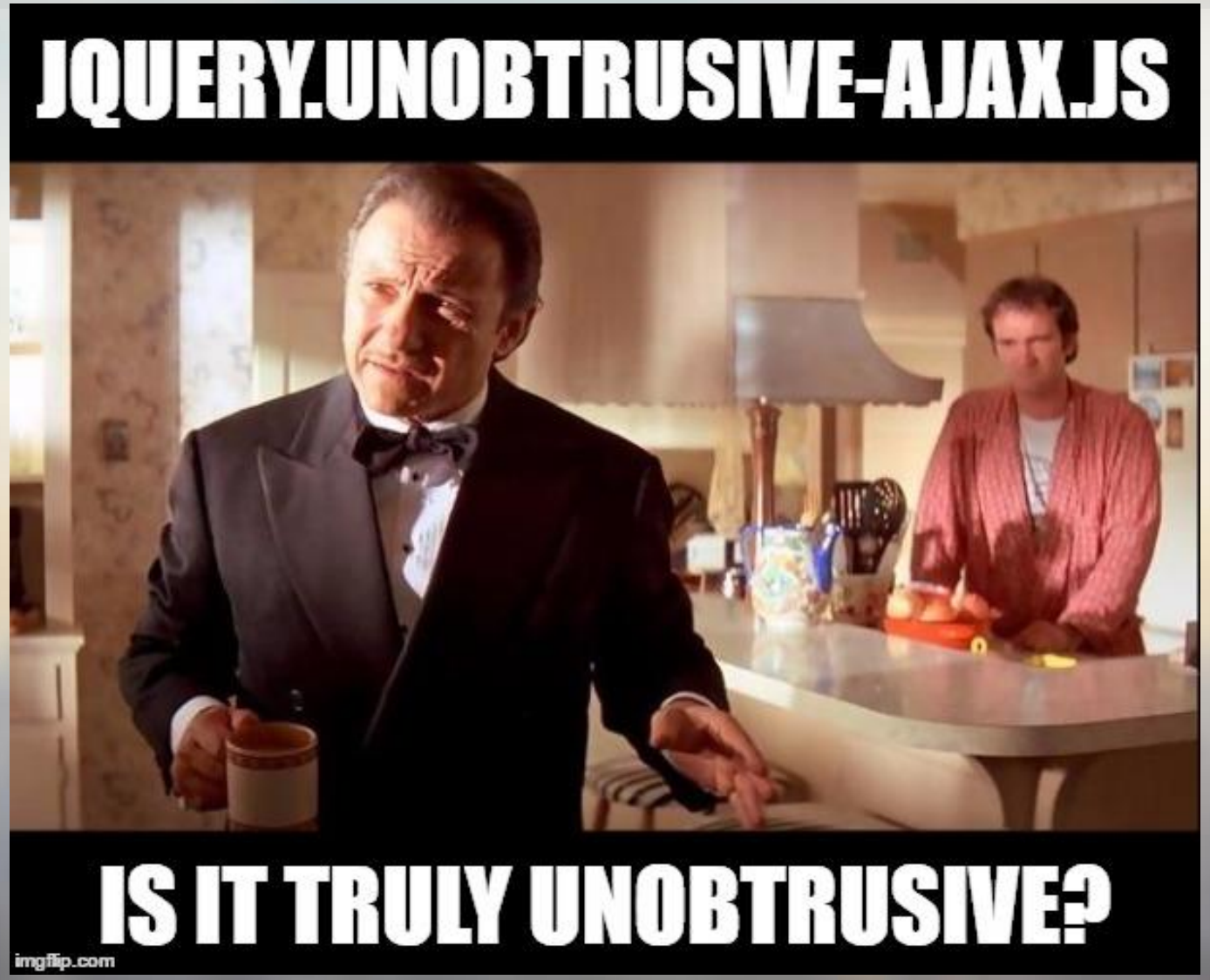
```
7
8   @using (Html.BeginForm("Register", "Account", FormMethod.Post, new { @class = "form-horizontal", role = "form", id="RegisterForm" }))
9   {
10      @Html.AntiForgeryToken()
11      <h4>Create a new account.</h4>
12      @*Form fields*@
13
```

Ajax Form

```
15  @using (Ajax.BeginForm("Save", new AjaxOptions() {
16      OnSuccess = "HandleBasicForm(data)",
17      OnFailure = "showErrors(data)"
18  }))
19  {
20      @*Form inputs*@
21  }
```

```
Uncaught EvalError: Refused to evaluate a string as JavaScript    jquery.unobtrusive-ajax.js:
because 'unsafe-eval' is not an allowed source of script in the following Content Security
Policy directive: "default-src 'self'".
```

Ajax Form


JQUERY.UNOBTRUSIVE-AJAX.JS
IS IT TRULY UNOBTRUSIVE?
imgflip.com

Ajax Form

```
@using (Ajax.BeginForm<ContractorController>(c => c.Save(null), "ContactForm"))
{

}
```

```
(function () {

    Ajax.OnSuccess("ContractorForm", function () {

    });

    Ajax.OnFailure("ContractorForm", function () {

    });

})();
```

Passing data between Razor and JavaScript

```
8   <script>
9       var message = "Hello @Model.UserName";
10      alert(message);
11  </script>
12
13  <style>
14      .page-header-layout {
15          background-color: "@Model.UserFavouriteColor"
16      }
17  </style>
18  <div class="page-header-layout">
19      <div class="header-content">
20          <h1>User Settings</h1>
21      </div>
22  </div>
```

Passing data between Razor and JavaScript

```
Refused to apply inline style because it violates the following Content Security    CaseList:7
Policy directive: "default-src 'self'". Either the 'unsafe-inline' keyword, a hash ('sha256-
RMH/evq07aWq01fNZhOR1hs8nKsiBNojRr3Qc4J6uBU='), or a nonce ('nonce-...') is required to enable
inline execution. Note also that 'style-src' was not explicitly set, so 'default-src' is used
as a fallback.
Refused to execute inline script because it violates the following Content Security  CaseList:8
Policy directive: "default-src 'self'". Either the 'unsafe-inline' keyword, a hash
('sha256-/zkCoQ+dl6aIgc/Bf8Yw3GSJ0QFJEVXubWRe4lckmzk='), or a nonce ('nonce-...') is required
to enable inline execution. Note also that 'script-src' was not explicitly set, so 'default-
src' is used as a fallback.
```

Contextual Encoding

Test

Context	Correctly encoded string
HTML	<i>Test</i>
JavaScript	\x3ci\x3eTest\x3c\x2fi\x3e
CSS	\00003Ci\00003ETest\00003C\00002F1\00003E

Passing data between Razor and JavaScript

```
16 <div id="UserSettingsPage" data-user-name="@Model.UserName" data-model="@Model.ToJson()">
17     <div class="page-header-layout">
18         <div class="header-content">
19             <h1>User Settings</h1>
20         </div>
21     </div>
22 </div>
```

```
var message = $("#UserSettingsPage").data("userName");

var model = $("#UserSettingsPage").data("model");
```

Razor + JQuery UI

```
1
2  $(function () {
3
4      $("#User").autocomplete();
5      $("#DateFrom").datepicker();
6      $("#DateTo").datepicker();
7
8  });
9
```

Razor + JQuery UI

DamianEdwards / **Unobtrusive-jQuery-UI**

Watch 4 | Star 19 | Fork 10

Code | Issues 3 | Pull requests 3 | Projects 0 | Wiki | Pulse | Graphs

Unobtrusive wire-up of jQuery UI features using data-ui-* attributes on target elements.

DateTime.cshtml

```
1  @model DateTime
2  @Html.TextBoxFor(m => m, new { data_ui_fn = "datepicker" })
3
4
5
```

Widget Factory

```
1  $.widget("ui.sequencelist", {
2      version: "1.0.0",
3      options: ...,
10     _create: function () {
11         var that = this;
12         this.element.on("click", this.options.selectorUp, function (...));
19         this.element.on("click", this.options.selectorDown, function () {
20             var $item = $(this).closest(that.options.selectorItem);
21             $item.insertAfter($item.next(that.options.selectorItem));
22             that._setArrowVisibility();
23             that.renumerateSequenceFields();
24             that._trigger("orderchanged");
25         });
26
27         this._setInitialOrder();
28         this._setArrowVisibility();
29         this.isVisible = true;
30     },
31     showWidget: function (...),
38     hideWidget: function (...),
48     setVisibility: function(isVisible)...,
55     renumerateSequenceFields: function (...),
63     _setInitialOrder: function (...),
77     _setArrowVisibility: function (...),
85     _getSequenceFields:function()...
88 });
```

Controller

```
[HttpPost]
[ValidateAntiForgeryToken]
[ValidateInput(false)]
public ActionResult DeliveryDetails(DeliveryDetailsViewModel model)
{
    // Check the selected shipping option
    if (!mCheckoutService.IsShippingOptionValid(model.ShippingOption.ShippingOptionID))
    {
        ModelState.AddModelError("ShippingOption.ShippingOptionID", ResHelper.GetString("DancingGoatMvc.Shipping.ShippingO
    }

    // Check if the billing address's country and state are valid
    var countryStateViewModel = model.BillingAddress.BillingAddressCountryStateSelector;
    if (!mCheckoutService.IsCountryValid(countryStateViewModel.CountryID))
    {
        countryStateViewModel.CountryID = 0;
        ModelState.AddModelError("BillingAddress.BillingAddressCountryStateSelector.CountryID", ResHelper.GetString("Danci
    }
    else if (!mCheckoutService.IsStateValid(countryStateViewModel.CountryID, countryStateViewModel.StateID))
    {
        countryStateViewModel.StateID = 0;
        ModelState.AddModelError("BillingAddress.BillingAddressCountryStateSelector.StateID", ResHelper.GetString("Dancing
    }

    if (!ModelState.IsValid)
    {
        var viewModel = mCheckoutService.PrepareDeliveryDetailsViewModel(model.Customer, model.BillingAddress, model.Shipp
        return View(viewModel);
    }

    var cart = mShoppingService.GetCurrentShoppingCart();
    var customer = cart.Customer ?? new Customer();

    model.Customer.ApplyToCustomer(customer);
    cart.Customer = customer;

    var modelAddressID = model.BillingAddress.BillingAddressSelector?.AddressID ?? 0;
    var billingAddress =  mCheckoutService.GetAddress(modelAddressID) ?? new CustomerAddress();

    model.BillingAddress.ApplyTo(billingAddress);
    billingAddress.PersonalName = $"{customer.FirstName} {customer.LastName}";

    cart.BillingAddress = billingAddress;
    cart.ShippingOption = mCheckoutService.GetShippingOption(model.ShippingOption.ShippingOptionID);

    cart.Save();

    return RedirectToAction("PreviewAndPay");
}
```

Perfect Controller

Patterns [edit]

Controller [edit]

The **controller** pattern assigns the responsibility of dealing with system events to a non-UI class that represents the overall system or a use case scenario. A controller object is a non-user interface object responsible for receiving or handling a system event.

A use case controller should be used to deal with *all* system events of a use case, and may be used for more than one use case (for instance, for use cases *Create User* and *Delete User*, one can have a single *UserController*, instead of two separate use case controllers).

It is defined as the first object beyond the UI layer that receives and coordinates ("controls") a system operation. The controller should delegate the work that needs to be done to other objects; it coordinates or controls the activity. It should not do much work itself. The GRASP Controller can be thought of as being a part of the application/service layer [2] (assuming that the application has made an explicit distinction between the application/service layer and the domain layer) in an object-oriented system with common layers in an information system logical architecture.

Perfect Controller

```
[ActionLinkArea(AreaNames.Admin)]
public class UserController : Controller
{
    private readonly IUserService userService;

    public UserController(IUserService userService)
    {
        this.userService = userService;
    }

    public ViewResult GetMemberForEdit(EntityDTO dto)
    {
        var memberEditViewModel = this.userService.GetMemberForEdit(dto);
        return this.View("Forms/Members/EditUserMember", memberEditViewModel);
    }

    [HttpPost]
    public JsonResult UpdateMember(UnitMemberEditDTO dto)
    {
        this.userService.UpdateMember(dto);
        return this.JsonSuccess();
    }
```

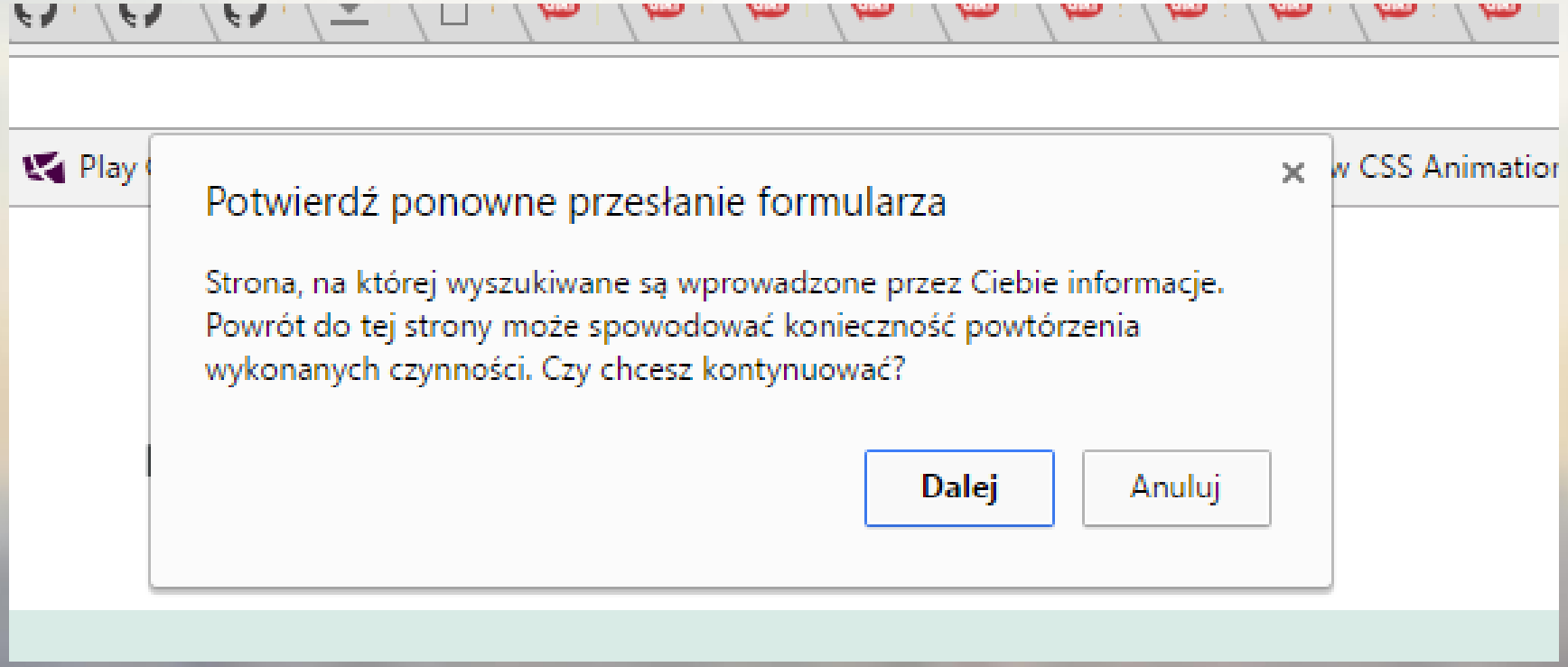
Perfect Controller

```
public ActionResult GetAdditionalProductDetails(AdditionalProductDetailsInputDTO input)
{
    var viewModelWrapper = this.eflCase3AdditionalService.GetAdditionalProductForEdit(input);
    switch (viewModelWrapper.WrappedContentType)
    {
        case AdditionalProductDetailsType.Gap:
            return this.View("Forms/AdditionalProductGap", viewModelWrapper.GapViewModel);
        case AdditionalProductDetailsType.Assistance:
            return this.View("Forms/AdditionalProductAssistance", viewModelWrapper.AssistanceViewModel);
        case AdditionalProductDetailsType.AssistanceTruck:
            return this.View("Forms/AdditionalProductAssistanceTruck", viewModelWrapper.AssistanceTruckViewModel);
        default:
            throw Fail.Because("Not supported additional product type");
    }
}
```

POST-REDIRECT-GET

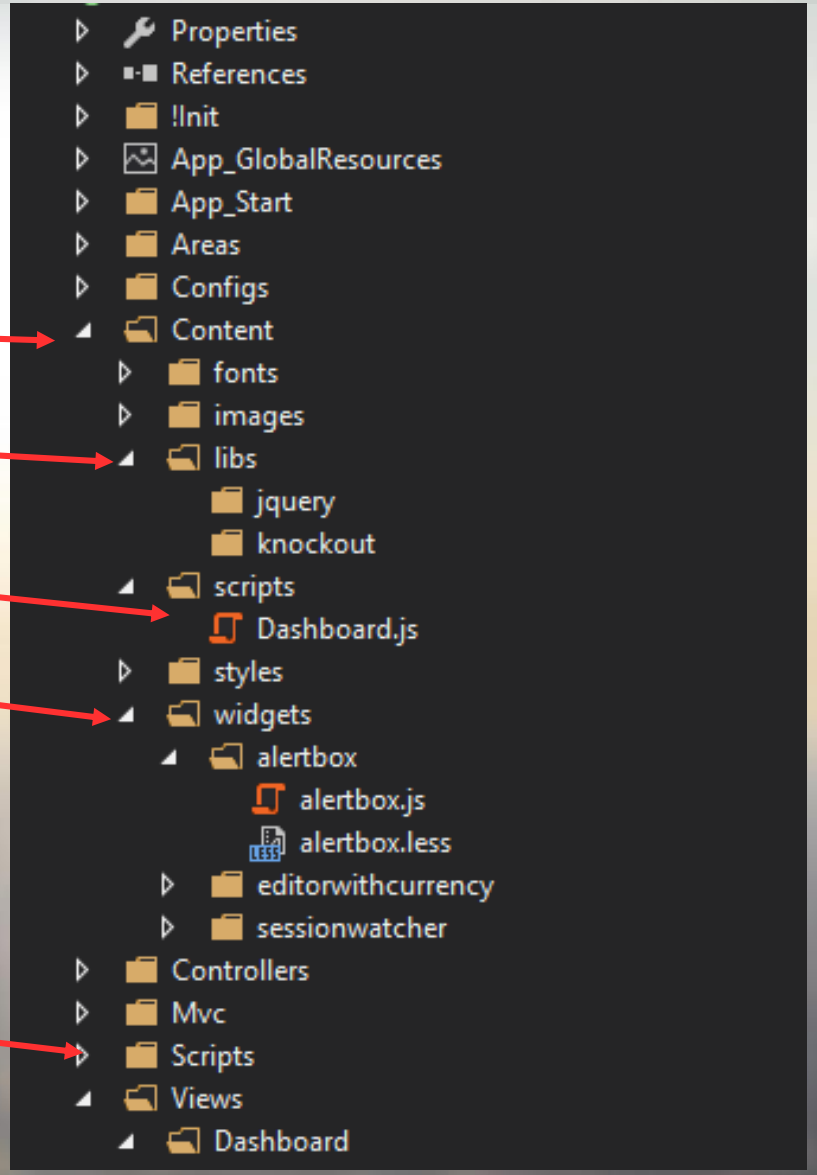
```
[HttpPost]
public ActionResult UpdateRelatedPerson(CreateUpdateRelationWithPersonDTO dto)
{
    this.eflCase4RelationService.UpdateRelationWithPerson(dto);
    return this.RedirectToAction(c => c.GetRelations(dto.CaseId));
}
```

POST-REDIRECT-GET

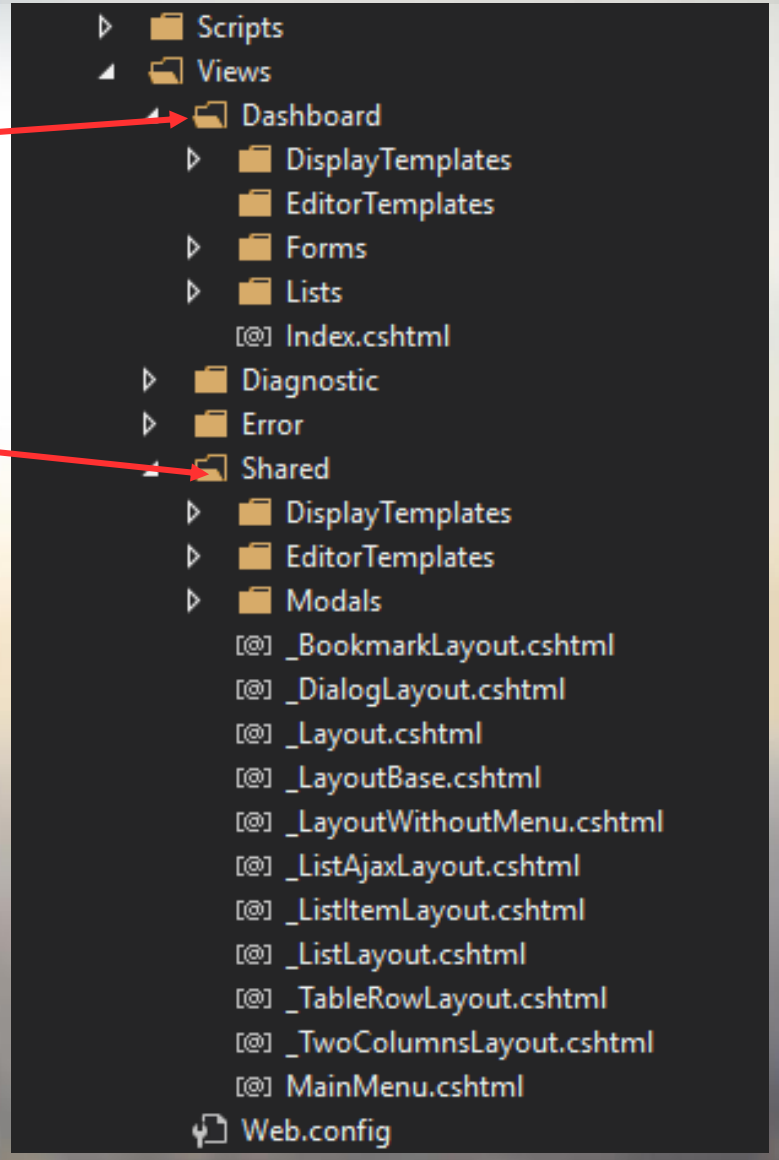


Directory Structure

Directory Structure



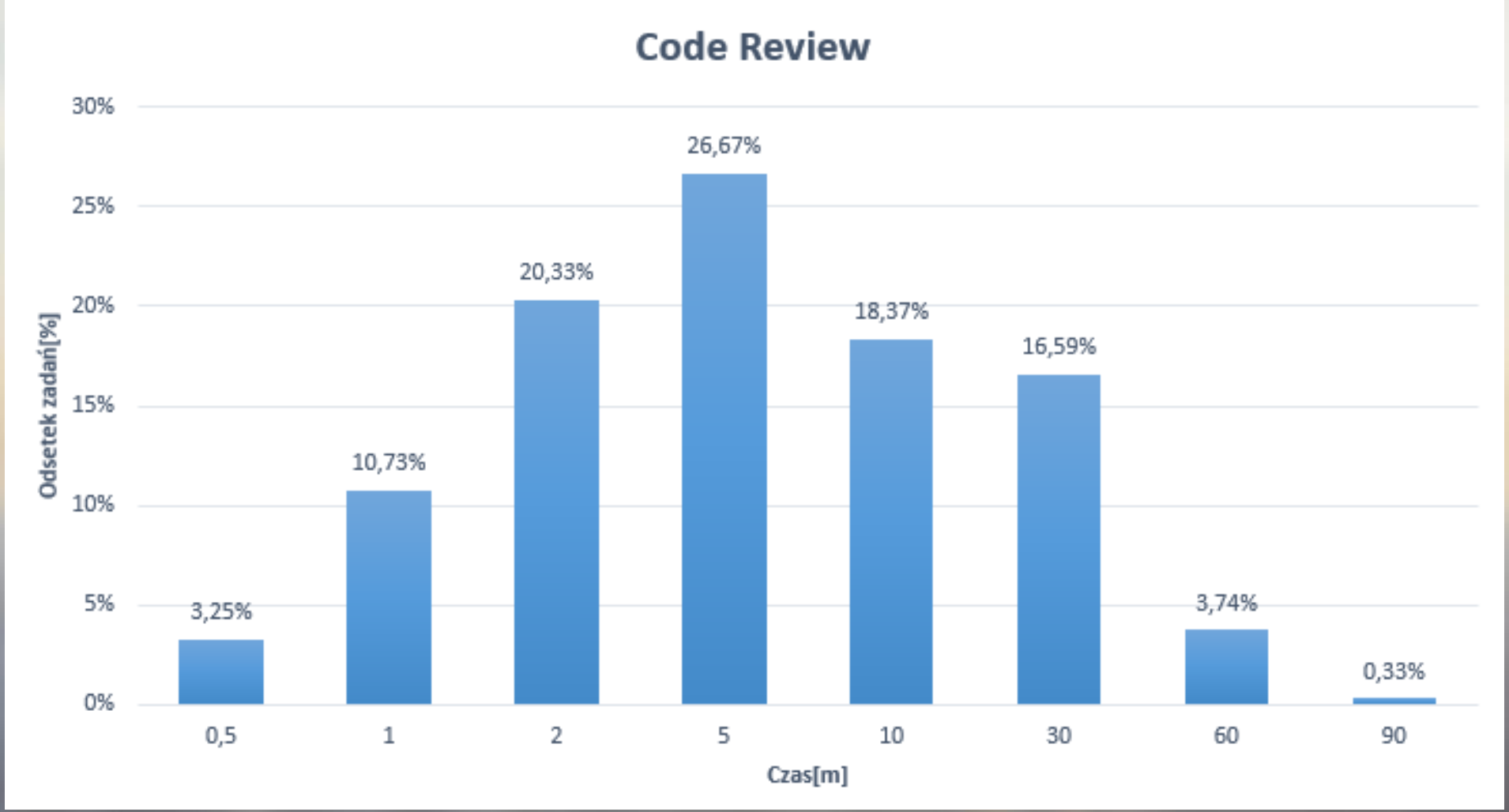
Directory Structure



Good practices

- Coding standards
- Knowledge transfer
- Code Review
- Code snippets/scaffolding
- Solution Wide Analisys

Code Review



Dziękuję za uwagę

Branches Global
USA / Boise, Dallas
Australia / Sydney, Brisbane
United Kingdom / London

Global Headquarters
ul. Życzkowskiego 20,
31-864 Krakow

Branches Poland
Warszawa, Rzeszow

tel.: + 48 12 252 34 00
office@ailleron.pl
www.ailleron.pl


ailleron
your digital wings