

Unobtrusive Ajax

Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET MVC vid Linnéuniversitetet.

Du får använda detta verk så här:

Allt innehåll i detta verk av Mats Look, förutom Linnéuniversitetets logotyp och symbol, är licensierad under:



Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens.

<http://creativecommons.org/licenses/by-nc-sa/2.5/se/>

Det betyder att du i icke-kommersiella syften får:

- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp och symbol i din nya version!

Vid all användning måste du ange källan: "Linnéuniversitetet – ASP.NET MVC" och en länk till

<https://coursepress.lnu.se/kurs/aspnet-mvc> och till Creative Common-licensen här ovan.

Ett traditionellt formulär

- ✓ Används `Html.BeginForm` renderas ett traditionellt formulär som postas tillbaka till servern och sidan renderas om fullständigt.

```
@model System.String
@{
    ViewBag.Title = "Hälsning utan Ajax";
}
<h1>Hälsning utan Ajax</h1>
@using(Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-group">
        @Html.Label("greeting", "Hälsning")
        @Html.TextBox("greeting")
    </div>

    <div class="form-group">
        <input type="submit" value="Posta hälsning" />
    </div>
}
@if (!String.IsNullOrEmpty(Model))
{
    <div id="result">
        Din hälsning (utan Ajax): @Model
    </div>
}
```

```
<h1>Hälsning utan Ajax</h1>
<form action="/" method="post">
    <input name="__RequestVerificationToken" type="hidden" value="..." />
    <div class="form-group">
        <label for="greeting">Hälsning</label>
        <input id="greeting" name="greeting" type="text" value="" />
    </div>
    <div class="form-group">
        <input type="submit" value="Posta hälsning" />
    </div>
</form>
```

ASP.NET MVC och ”*Ajax Helpers*”

- ✓ MVC har stöd för Ajax-anrop. Från och med version 3 finns även stöd för ”unobtrusive Ajax” baserad på jQuery.
- ✓ För att ”unobtrusive Ajax” ska fungera måste...
 - ...UnobtrusiveJavaScriptEnabled vara satt till true, vilken den är som standard i Web.config.
 - ...jquery-2.1.1.min.js , eller annan lämplig version, länkas in.
 - ...jquery.unobtrusive-ajax.min.js länkas in.
- ✓ Med hjälp av ”*Ajax Helpers*” är det enkelt att skapa förutsättningar för asynkrona förfrågningar.
 - `Ajax.ActionLink`
 - `Ajax.BeginForm`
 - `Ajax.RouteLink`
 - `Ajax.BeginRouteForm`

Formulär som använder Ajax

- ✓ Genom att använda `Ajax.BeginForm` renderas ett formulär som använder Ajax och endast innehållet i specificerat element ersätts. I övrigt är allt precis som då ett traditionellt formulär renderas.

```
public class BundleConfig
{
    // For more information on bundling, visit http://go.microsoft.com/fwlink/?LinkId=301862
    public static void RegisterBundles(BundleCollection bundles)
    {
        bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
            "~/Scripts/jquery-{version}.js"));

        bundles.Add(new ScriptBundle("~/bundles/jqueryval").Include(
            "~/Scripts/jquery.validate*"));

        // Use the development version of Modernizr to develop on and learn the
        // usage patterns, then in production use the build tool at
        // http://modernizr.com/docs/latest/#version to build a
        // custom Modernizr.min file for the production version
        bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
            "~/Scripts/modernizr-*"));

        bundles.Add(new StyleBundle("~/Content/css").Include(
            "~/Content/site.css"));
    }
}
```

```
@model System.String
@{
    var ajaxOptions = new AjaxOptions
    {
        UpdateTargetId = "result"
    };
}
@using (Ajax.BeginForm(ajaxOptions))
{
    @Html.AntiForgeryToken()

    <div class="form-group">
        @Html.Label("greeting", "Hälsning")
        @Html.TextBox("greeting")
    </div>

    <div class="form-group">
        <input type="submit" value="Posta hälsning" />
    </div>

    <div id="result">
        @Model
    </div>
}
@section Scripts {
    @Scripts.Render("~/bundles/ajax")
}
```

Microsoft jQuery Unobtrusive Ajax
jQuery plugin that unobtrusively sets
up jQuery Ajax.

Microsoft jQuery Unobtrusive Validation
jQuery plugin that unobtrusively sets up
jQuery.Validation.

jQuery
jQuery Ajax

Tags:
Dep

```
<form action="/" data-ajax="true" data-ajax-mode="replace" data-ajax-update="#result">
    <input name="greeting" type="text" value="Hälsning" />
    <div class="form-group">
        <input type="submit" value="Posta hälsning" />
    </div>
    <div id="result">
        @Model
    </div>
</form>
<script src="~/Scripts/jquery-2.1.1.js"></script>
<script src="~/Scripts/jquery.validate.js"></script>
<script src="~/Scripts/jquery.unobtrusive-ajax.js"></script>
```

Elementär hantering av Ajax-förfrågan

- ✓ Precis som en vanlig förfrågan hanteras en Ajax-förfrågan av en publik metod i en controllerklass. I sin enklaste form returnerar en sådan metod en sträng som ersätter innehållet i specificerat element.

Hälsning genom Ajax

Hälsning
Tjabba, tjena, hallå!

Posta hälsning

Din Ajax-hälsning: Tjabba, tjena, hallå!

```
[HttpPost]
[ValidateAntiForgeryToken]
public string Index(string greeting)
{
    return "Din Ajax-hälsning: " + greeting;
}
```

```
<div id="result">
    Tjabba, tjena, hallå!
</div>
```

```
<script src="/Scripts/jquery-2.1.1.js"></script>
<script src="/Scripts/
```

Men vad händer om JavaScript är avstängt?

- ✓ Om JavaScript är avstängt...
 - ...anropas aldrig `onsubmit`, varför...
 - ...formuläret istället postas som vanligt, och...
 - ...allt fungerar utan att några fel inträffar, men webbläsaren visar endast det som controllermetoden returnerade.



En controllermetod – olika typer av förfrågningar

- ✓ Controllermetoden måste kunna hantera både en Ajax-förfrågan och en vanlig förfrågan. Genom att använda metoden `Request.IsAjaxRequest()` kan du ta reda på vilket sätt förfrågan görs.
 - Är det fråga om en Ajax-förfrågan returneras en sträng med hjälp av `Content`.
 - Är det en "vanlig" förfrågan returneras en vy som vanligt.

```
public ActionResult Index()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Index(string greeting)
{
    if (Request.IsAjaxRequest())
    {
        return Content("Din Ajax-hälsning: " + greeting);
    }

    return View((object)greeting);
}
```

```
@model System.String

@{
    ViewBag.Title = "Hälsning genom Ajax";
    var ajaxOptions = new AjaxOptions
    {
        UpdateTargetId = "result"
    };
}

<h1>Hälsning genom Ajax</h1>

@using (Ajax.BeginForm(ajaxOptions))
{
    @Html.AntiForgeryToken()

    <div class="form-group">
        @Html.TextBoxFor(m => m)
```


Ajax och "partial views"

- ✓ Istället för att "bara" returnera en sträng kan en partiell vy returneras, som även används då det inte är en Ajax-förfrågan.

```
namespace HelloAjax.Models
{
    public class Greeting
    {
        [Required(ErrorMessage="Du måste skriva en hälsning")]
        [Display(Name="Hälsning", Prompt="Skriv en hälsning")]
        public string Message { get; set; }
    }
}
```

```
public ActionResult Index()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Index(Greeting greeting)
{
    if (Request.IsAjaxRequest())
    {
        return PartialView("_Greeting", greeting);
    }

    return View(greeting);
}
```

```
@model HelloAjax.Models.Greeting

@if (Model != null &&
    !String.IsNullOrEmpty(Model.Message))
{
    <p>
        Din hälsning: <strong>@Model.Message</strong>
    </p>
}
```

```
@model HelloAjax.Models.Greeting
@{
    var ajaxOptions = new AjaxOptions
    {
        UpdateTargetId = "result"
    };
}

@using (Ajax.BeginForm(ajaxOptions))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)

    <div class="form-group">
        @Html.EditorFor(model => model.Message)
        @Html.ValidationMessageFor(model => model.Message)
    </div>

    <div class="form-group">
        <input type="submit" value="Posta hälsning" />
    </div>

    <div id="result">
        @Html.Partial("_Greeting")
    </div>

    @section Scripts{
        @Scripts.Render("~/bundles/jqueryval")
        @Scripts.Render("~/bundles/ajax")
    }
}
```

Tja!

Posta hälsning

Din hälsning: Tja!

Enkel hantering av fel vid Ajax-förfrågan

- ✓ Det är viktigt att `Response.StatusCode` sätts till lämplig felkod så att JavaScript-funktionen som definieras av `OnFailure` körs på klienten.

```
public class Greeting
{
    [Required(ErrorMessage="Du måste skriva en hälsning",
    [MustBeEvenNumberOfCharacters(ErrorMessage="Hälsning  
må ha ett jämt antal tecken", Prompt="Skriv en hälsning")
    public string Message { get; set; }
}
```

```
public ActionResult Index()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Index(Greeting greeting)
{
    if (Request.IsAjaxRequest())
    {
        if (ModelState.IsValid)
        {
            return PartialView("_Greeting", greeting);
        }
        else
        {
            Response.StatusCode = 400;
            return Content("Ett fel inträffade.");
        }
    }
    return View(greeting);
}
```

```
@model HelloAjax.Models.Greeting
@{
    var ajaxOptions = new AjaxOptions
    {
        UpdateTargetId = "result",
        OnFailure = "handleAjaxFailure"
    };
}

@using (Ajax.BeginForm(ajaxOptions))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)

    <div class="form-group">
        @Html.EditorFor(model => model.Message)
        @Html.ValidationMessageFor(model => model.Message)
    </div>

    <div class="form-group">
        <input type="submit" value="Posta hälsning" />
    </div>

    <div id="result">
        @Html.Partial("_Greeting")
    </div>
}

@section Scripts{
    @Scripts.Render("~/bundles/jqueryval")
    @Scripts.Render("~/bundles/ajax")

    <script>
        function handleAjaxFailure(ajaxContext) {
            $("#result").html("<span class='field-validation-error'>" + ajaxContext.responseText + "</span>");
        }
    </script>
}
```

En udda hälsning!

Posta hälsning

Ett fel inträffade.

Hantering av fel vid Ajax-förfrågan med JSON

- ✓ För att skicka samtliga felmeddelanden gällande modellens egenskaper till klienten kan JSON användas.
- ✓ På klienten används `jQuery.validate().showErrors()` för att visa felen där de ska visas.

```
public ActionResult Index()
{
    return View();
}

[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Index(Greeting greeting)
{
    if (Request.IsAjaxRequest())
    {
        if (ModelState.IsValid)
        {
            return PartialView("_Greeting", greeting);
        }

        Response.StatusCode = 400;

        // Transform the modelstate errors to a dictionary where property names is
        // associated with there errors.
        var errors = ModelState
            .Where(kvp => kvp.Value.Errors.Any())
            .ToDictionary(kvp => kvp.Key,
                kvp => kvp.Value.Errors.Select(e => e.ErrorMessage).ToArray());

        return Json(new { Errors = errors });
    }

    return View(greeting);
}
```

```
@model HelloAjax.Models.Greeting
@{
    var ajaxOptions = new AjaxOptions
    {
        UpdateTargetId = "result",
        OnFailure = "handleAjaxFailure"
    };
}

@using (Ajax.BeginForm(ajaxOptions))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)

    <div class="form-group">
        @Html.EditorFor(model => model.Message)
        @Html.ValidationMessageFor(model => model.Message)
    </div>

    <div class="form-group">
        <input type="submit" value="Posta hälsning" />
    </div>

    <div id="result">
        @Html.Partial("_Greeting")
    </div>
}

@section Scripts{
    @Scripts.Render("~/bundles/jqueryval")
    @Scripts.Render("~/bundles/ajax")
}

<script>
    function handleAjaxFailure(ajaxContext) {
        // Empty the ajax target element.
        $('#result').empty();

        // Convert the response text, a JSON string, into a dictionary.
        var errors = JSON.parse(ajaxContext.responseText)["Errors"];

        // Show property errors.
        $('#form').validate().showErrors(errors);
    }
</script>
```

En udda hälsning!
Hälsningen måste innehålla ett jämt a

Posta hälsning

...används ValidationSummary så blir det komplexare

- ✓ Används ValidationSummary för att visa fel som inte är knutna till någon egenskap i modellen, medför det att skriptet som visar fel blir mer omfattande.

```
@model HelloAjax.Models.Greeting
@{
    var ajaxOptions = new AjaxOptions
    {
        UpdateTargetId = "result",
        OnSuccess = "handleAjaxSuccess",
        OnFailure = "handleAjaxFailure"
    };
}

@section Scripts{
    @Scripts.Render("~/bundles/jqueryval")
    @Scripts.Render("~/bundles/ajax")
}

<script>
function handleAjaxSuccess(ajaxContext) {
    // Remove all none property errors.
    if ($('#div[data-valmsg-summary="true"]').length != 0) {
        if ($('#div[data-valmsg-summary="true"]').hasClass("validation-summary-errors")) {
            $('#div[data-valmsg-summary="true"]').removeClass("validation-summary-errors");
        }
        $('#div[data-valmsg-summary="true"]').addClass("validation-summary-valid");
    }
    $('#div[data-valmsg-summary="true"] ul').empty();
}

function handleAjaxFailure(ajaxContext) {
    // Empty the ajax target element.
    $('#result').empty();

    // Convert the response text, a JSON string, into a dictionary.
    var errors = JSON.parse(ajaxContext.responseText)["Errors"];

    // Show none property errors. Insert validation summary elements, if we need to,
    // and append modelstate errors to the validation summary.
    if (errors[""] != undefined) {
        if ($('#div[data-valmsg-summary="true"]').length == 0) {
            $('#form').prepend('<div class="validation-summary-errors" data-valmsg-summary="true"><ul></ul></div>');
        }
        else if ($('#div[data-valmsg-summary="true"]').hasClass("validation-summary-valid")) {
            $('#div[data-valmsg-summary="true"]').removeClass("validation-summary-valid");
        }
        $('#div[data-valmsg-summary="true"]').addClass("validation-summary-errors");
        for (var i in errors[""]) {
            $('#div[data-valmsg-summary="true"] ul').append("<li>" + errors[""][i] + "</li>");
        }
        delete errors[""];
    }

    // Show property errors.
    $('#form').validate().showErrors(errors);
}
</script>
```