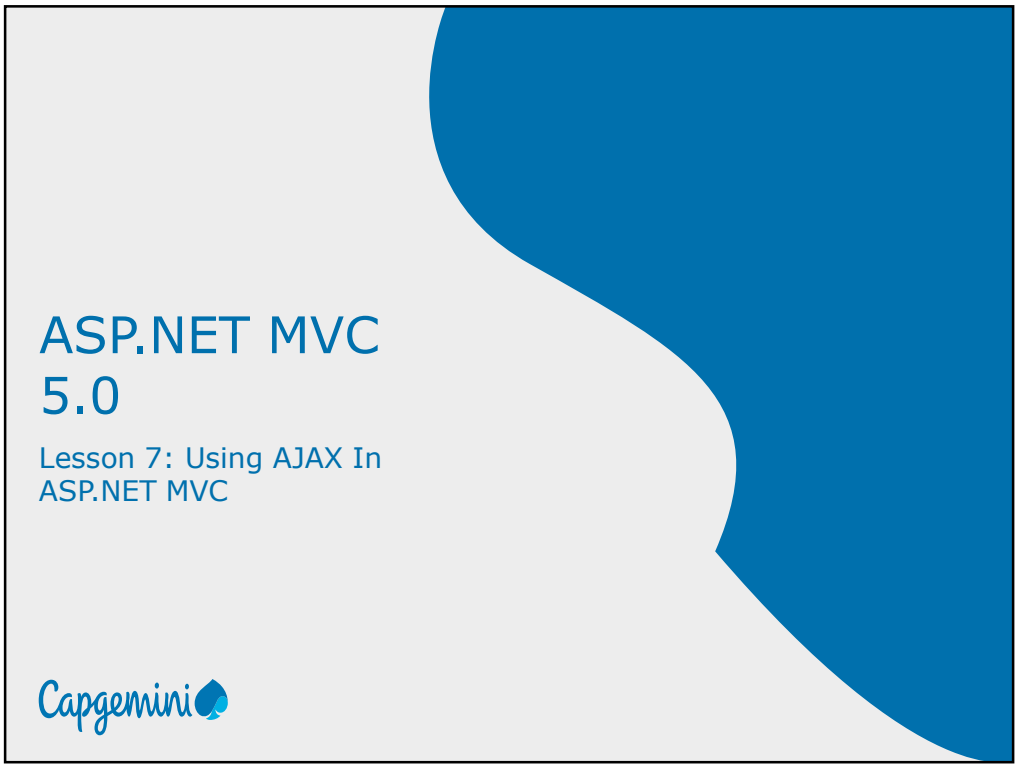
ASP.NET MVC
5.0
Lesson 7: Using AJAX In
ASP.NET MVC
Capgemini

Lesson Objectives

- AJAX Helpers
- ASP.NET AJAX Client Libraries
- AJAX & Partial View
- JSON
- jQuery

6.1 AJAX

AJAX Introduction

- AJAX stands for Asynchronous JavaScript and XML.
- AJAX uses client-side scripting to exchange data with a web server.
- AJAX allows the web page to change its content without refreshing the whole page .
- AJAX Improves the user experience and creates responsive web application.
- AJAX Improves bandwidth utilization
  - Core support of AJAX comes from the open source jQuery JavaScript library

AJAX replaces Synchronous request / Response Model. Data can be loaded on demand.

Only data which is required is retrieved from the server

Data-driven (as opposed to page-driven)

AJAX uses the XML HTTP request object is used to send asynchronous request back to the server for data. The server can send us back that data in JSON format of XML format. It could also be HTML and XHTML that the server is sending back. But those request are happening asynchronously so they're not really blocking the user or preventing them from continuing to view and interact with the webpage.

Once that data arise back to the client, we're using technologies like JavaScript and dynamic HTML and cascading style sheets to update and manipulate the display. The end result is that we're building a more responsive application and also a richer interface so users today expect web applications to look and feel almost like desktop applications. These asynchronous request allow us to reduce flicker. That's not a page refresh that has to redraw the entire screen.

6.1 AJAX

Raw AJAX Drawbacks

- To reload a small section of the page using a little bit of information we need to write lot of scripts
- Using XML HTTP request directly in your application code need to be avoided.
- In versions of Internet Explorer prior to IE7, XMLHttpRequest is implemented using ActiveX, whereas in other browsers, such as Mozilla Firefox, Safari, Opera, and Internet Explorer 7, it is a in-built JavaScript object this leads to browser inconsistencies.
- Wiring up to events requires different code on different browsers.

6.1 AJAX

AJAX in ASP.NET MVC

- There is a need to take advantage of existing libraries and plugin code to fit our need which has been already written and tested debug across multiple browsers and work more productively.
- ASP.NET MVC provides 2 client-side libraries
  - ASP.NET AJAX Library
  - jQuery
- ASP.NET MVC supports unobtrusive Ajax which is based on jQuery. The unobtrusive Ajax means that you use helper methods to define your Ajax features, rather than adding blocks of code throughout the views.

6.1 AJAX

AJAX in ASP.NET MVC

- Microsoft AJAX
  - Component Oriented
  - OOP Style
  - CLR flavor to Javascript
  - Support for WCF & JSON
- jQuery
  - Plugin oriented
  - CSS Selectors
  - DOM Manipulation

AJAX libraries are component oriented and they tend to have a very object oriented style. With Microsoft AJAX, you can create namespaces, you can create classes within those namespaces. It gives CLR type flavor to JavaScript. It's essentially a wrapper around that XML HTTP request object and it gives us a more convenient familiar programming interface to access the XMLHTTP request functionality. There's also a great support in the AJAX libraries from Microsoft for WCF Web Services and JSON serialization and deserialization.

jQuery is not implemented by Microsoft but it is shipped by Microsoft with ASP.NET and it's fully supported by Microsoft. It's an open source project. It gives us great CSS selectors which allow you to reach into the DOM and essentially query the DOM and pull back one or more elements that we need to manipulate. Once DOM elements are referenced it's very easy to change their styles, text and their values using jQuery. By downloading jQuery plugins we can perform client-side validation,widgets (jQuery UI)for free to work with jQuery.

6.2 AJAX Helpers

AJAX Helpers

- AJAX Helpers are used to create AJAX enabled elements like Ajax enabled forms and links which performs request asynchronously
- AJAX Helpers are extension methods of AJAX Helper class which exist in System. Web. Mvc. Ajax namespace.

AJAX libraries are component oriented and they tend to have a very object oriented style. With Microsoft AJAX, you can create namespaces, you can create classes within those namespaces. It gives CLR type flavor to JavaScript. It's essentially a wrapper around that XML HTTP request object and it gives us a more convenient familiar programming interface to access the XMLHTTP request functionality. There's also a great support in the AJAX libraries from Microsoft for WCF Web Services and JSON serialization and deserialization.

jQuery is not implemented by Microsoft but it is shipped by Microsoft with ASP.NET and it's fully supported by Microsoft. It's an open source project. It gives us great CSS selectors which allow you to reach into the DOM and essentially query the DOM and pull back one or more elements that we need to manipulate. Once DOM elements are referenced it's very easy to change their styles, text and their values using jQuery. By downloading jQuery plugins we can perform client-side validation,widgets (jQuery UI)for free to work with jQuery.

AJAX Helpers contd..

- AJAX Action Links
  - This helper creates an anchor tag with asynchronous behavior.

```
@Ajax.ActionLink("Click here to see the LOT options", "Search",new
AjaxOptions{UpdateTargetId = "chkoptions",
                    InsertionMode=InsertionMode.Replace,
                    HttpMethod="GET"})
```

The ActionLink method of the Ajax property creates an anchor tag with asynchronous behavior. Let says the user is about to create/edit course details. You might want the user to have a look at the various LOT options so you could provide a link which will display the list on the same page. To incorporate this behavior, you can add the code snippet shown on the slide in edit.cshtml.

The first parameter for ActionLink method specifies the link text and name of the action you want to invoke asynchronously is the second parameter. The next parameter is the AjaxOptions. This parameter specifies how to send the request and what will happen with the result the server returns. Options are also provided for handling errors, displaying a loading element, displaying confirmation dialog and so on. In our example, we use options to specify that you want to replace the element with an id “chkoptions” using whatever response comes from the server.

AJAX Helpers contd..

- AJAX Forms
  - This helper creates a form with an asynchronous behavior

```
@using (Ajax.BeginForm("Display", "Course",
          new AjaxOptions
          {
             InsertionMode = InsertionMode.Replace,
             HttpMethod = "GET",
             LoadingElementId = "ajax-loader",
             UpdateTargetId = "SearchResults",   }))
                    <input type="text" name="q" />
                      <input type="submit" value="Find" />
                      <img id="ajax-loader"
                    src="@Url.Content("~/Content/Images/find.gif")"
                         style="display:none" alt="No Image"/>
}
```

Let us now consider another example where you might want to add a search functionality in your page based on user input. For this you will need a form tag with an input box and you also want asynchronous behavior for the form.

Ajax Forms helper can be used for this purpose. The method is Ajax.BeginForm for rendering the form with asynchronous behavior. When the user clicks the submit button the browser send an asynchronous GET request for searching a name of a course. We also specified a LoadingElementId as part of the options, this will show the desired element when the asynchronous request is in progress.

You could also display an error message if the search failed by specifying OnFailure as part of the options and assign a function name to it which could be either in the same page or in a .js file separately.

6.2 AJAX Helpers

AJAX Helpers Properties

AjaxOptions class defines properties that allow you to specify callbacks for different stages in the AJAX request life cycle. There are following properties provided by AjaxOptions class for AJAX helpers:

Url

Specify the URL that will be requested from the server.

Confirm

Specify a message that will be displayed in a confirm dialog to the end user. When user clicks on OK button in the confirmation dialog, the Ajax call performs.

OnBegin

Specify a JavaScript function name which is called at the beginning of the Ajax request.

OnComplete

Specify a JavaScript function name which is called at the end of the Ajax request.

OnSuccess

Specify a JavaScript function name which is called when the Ajax request is successful.

OnFailure

Specify a JavaScript function name which is called if the Ajax request fails.

LoadingElementId

Specify progress message container's Id to display a progress message or animation to the end user while an Ajax request is being made.

LoadingElementDuration

Specify a time duration in milliseconds that controls the duration of the progress message or animation.

UpdateTargetId

Specify the target container's Id that will be populated with the HTML returned by the action method.

InsertionMode

Specify the way of populating the target container. The possible values are InsertAfter, InsertBefore and Replace (which is the default).

6.3 Partial Page Rendering

Partial Page Rendering

- AJAX supports Partial Page refresh. i.e. Portion of the page gets updated
- In ASP.NET Web forms it is done through update panel. In MVC framework it is done through Partial View.
- If request to Action Method is asynchronous request we can return a partial view which results in updating a portion of the screen by checking Request. Is Ajax Request() method inside the Controller.
- To do partial page updating ensure jquery.unobtrusive-ajax.min.js is included in the script bundle and make sure Unobtrusive Java Script Enabled is set to true in the root of Web. config file.

6.4 JSON

JSON Introduction

- JSON stands for Java Script Object Notation
- JSON is a lightweight format for exchanging data between the client and server.
- JSON is a syntax for passing around objects that contain name/value pairs, arrays and other objects.
- It is often used in AJAX applications because of its simplicity and because its format is based on JavaScript object literals.
- JSON is language independent and text based. It is easy to parse.
- Supported by most of the languages.

JSON Types

Number : integer, real or floating point
String : double-quoted Unicode with backslashes
Boolean : true and false
Array : ordered sequence of comma-separated values enclosed in square brackets
Object : collection of comma-separated “key”:value pairs enclosed in curly braces
null

JSON Object

```
var jsonObject = {
firstName:"Karthik", lastName:"Muthukrishnan",
  "office":{
                    "company":”IGATE",
                    "city":"Bangalore"
   },
   "Hobbies":["Programming","Music"]
}
```

6.4 JSON

JSON in ASP.NET MVC

- Producing JSON from a controller action is very easy. We can take an object, wrap it inside of a JSON result and the MVC framework will serialize that object into JSON.
- Similarly from controller anonymous type wrapped in a JSON result will be sent over the wire as JSON to the client

jQuery Ajax features

- Allows part of a page updated
- Cross-Browser support
- Simple API
- GET and POST supported
- Load JSON,XML, HTML …

6.5 Ajax using jQuery

Ajax using jQuery

- JQuery has the following methods that can be used to perform Ajax requests:
  - ajax() - Load a remote page using an HTTP request. This is jQuery's low-level AJAX implementation.
  - load() - Load HTML from a remote file and inject it into the DOM.
  - get() - Load a remote page using an HTTP GET request.
  - getJSON() - Load JSON data using an HTTP GET request.
  - getScript() - Loads, and executes, a local JavaScript file using an HTTP GET request.
  - post() - Loads HTML by performing an HTTP post request.

Using get(), getJSON() & post()

- $.get(url,data,callback,datatype) can retrieve data from a server.

```
$.get('GetContents.html',function(data){
        $('#targetDiv').html(data);
},'html');
```

- datatype can be html, xml, json
- $.getJSON(url,data,callback) can retrieve data from a server.

```
$.getJSON('GetContents.aspx,{id:5},function(data){
        $('#targetDiv').html(data);
});
```

- $.post(url,data,callback,datatype) can post data to a server and retrieve results.

Using ajax() function

- ajax() function is configured by assigning values to JSON properties

sample.json file contents

```
{
"people": [
{ "name": "Jack",
"age" : 20 },
{ "name": "Grant",
"age": 21 },
{ "name": "Lisa",
"age": 21 }
]
}
```

```
$(function() {
$.ajax({
"url": "/sample.json",
"type": "get",
"dataType": "json",
"success":
function(data) {
console.log(data);
} }); });
```

async

Set to false if the request should be sent synchronously. Defaults to true. Note that if you set this option to false, your request will block execution of other code until the response is received.

cache

Whether to use a cached response if available. Defaults to true for all dataTypes except "script" and "jsonp". When set to false, the URL will simply have a cachebusting parameter appended to it.

done

A callback function to run if the request succeeds. The function receives the response data (converted to a JavaScript object if the dataType was JSON), as well as the text status of the request and the raw request object.

fail

A callback function to run if the request results in an error. The function receives the raw request object and the text status of the request.

always

A callback function to run when the request is complete, regardless of success or failure. The function receives the raw request object and the text status of the request.

context

The scope in which the callback function(s) should run (i.e. what this will mean inside the callback function(s)). By default, this inside the callback function(s) refers to the object originally passed to $.ajax().

data

The data to be sent to the server. This can either be an object or a query string, such as foo=bar&baz=bim.

dataType

The type of data you expect back from the server. By default, jQuery will look at the MIME type of the response if no dataType is specified.

jsonp

The callback name to send in a query string when making a JSONP request. Defaults to "callback".

timeout

The time in milliseconds to wait before considering the request a failure.

type

The type of the request, "POST" or "GET". Defaults to "GET". Other request types, such as "PUT" and "DELETE" can be used, but they may not be supported by all browsers.

url

The URL for the request.

The url option is the only required property of the $.ajax() configuration object; all other properties are optional. This can also be passed as the first argument to $.ajax(), and the options object as the second argument.

Configuration option	Purpose
`url`	The URL of the content, for the request.
`data`	The data to be sent to the server.
`error`	This function is called in the event of the request failing - the function will be passed three arguments: the `jqXHR` object, a string describing the error, and an optional exception object, if one is generated.
`dataType`	This describes the type of data that you're expecting to see returned from the server. By default, jQuery will try to work this out automatically, but it could be one of the following: XML, JSON, script, or HTML.
`Success`	A function to be called if the request is successful.
`type`	The type of request to make, for example, `'POST'`, `'GET'` or `'PUT'` – the default is `'GET'`.

GET call to Controller's Method

- GET call to Controller's Method which will return string data
  - we have following method in the controller

```
public string TellMeDate()
{
    return DateTime.Today.ToString();
}
```

  - let's make an async call using jQuery Ajax

```
<p id="rData">
</p>
<script type="text/jscript">
    var url = "/Home/TellMeDate";
    $.get(url, null, function (data) {
        $("#rData").html(data);
    });
</script>
```

GET call to Controller's Method

- If you want to generate Ajax GET request when user clicks button, we can use following instead.

```
<script type="text/jscript">
   $('#ButtonID').click(function () {
      var url = "/Home/TellMeDate";
      $.get(url, null, function (data) {
         $("#rData").html(data);
      });
   })
</script>
```

POST call to Controller's Method

- here is the method accepting two parameters name and address

```
[HttpPost]
public string SubmitSubscription(string Name, string Address)
{
   if (!String.IsNullOrEmpty(Name) && !String.IsNullOrEmpty(Address))
      //TODO: Save the data in database
      return "Thank you " + Name + ". Record Saved.";
   else
      return "Please complete the form.";

}
```

POST call to Controller's Method

- Here is the jQuery Ajax POST function

```
<input type="button" value="Save" id="Save" />
<span id="msg" style="color:red;"/>

<script type="text/javascript">
   $('#Save').click(function () {
      var url = "/Home/SubmitSubscription";
      var name = $("#Name").val();
      var address = $("#Address").val();
      $.post(url, { Name: name, Address: address }, function (data) {
         $("#msg").html(data);
      });
   })
</script>
```

DEMO

- jQuery AJAX Demo

Summary

- MVC framework provides basic AJAX building blocks.
- ASP.NET MVC AJAX libraries are used to invoke the WCF web service and also to support the AJAX helpers.
- Numerous plugins are available for free at plugins.jQuery.com
- We can widgets like datepicker, autocomplete textbox etc in MVC application using jQuery UI

Review Question

- Question 1: Use Ajax. Begin Form instead of Html. Begin Form when you want to
  - submit JSON to the server with jQuery
  - turn a form submission into an asynchronous request
  - place JavaScript code inside a form
  - All of the above
- Question 2: The MVC framework, by default, will take the Ajax Settings you specify for the Ajax. Action Link helper and
  - Serialize the settings into JSON and inject a script tag to download the settings separately
  - Encode the settings into data- attributes and place them in the DOM
  - Serialize the settings into JSON and place them in the DOM
  - Store the settings in the user's session on the server