.NET Framework: Developing Modern Web Apps with ASP.NET MVC – Workshop*PLUS*

Wael Kdouh

Senior Consultant

v1.0

Module 6: Client-side Development

Module Overview

Module 6: Client-side Development

Section 1: ASP.NET MVC and JavaScript

Lesson: Project Templates

Why Use JavaScript in ASP.NET MVC?

- Combines server-side and client-side processing
- More responsive web applications
- JavaScript comes in many forms:
  - AJAX – Partial page update and refresh
  - Jquery – Elegantly and Efficiently find and manipulate HTML DOM elements
  - MooTools – Modular JavaScript and code reuse
  - Prototype – Simplify development of dynamic web application
  - Node.js – Developing high performance JavaScript using multithreading model
  - Industry standard for modern web development, etc.

Project Template: JavaScript Libraries Part 1

- Bootstrap.js and bootstrap.min.js
  - HTML, CSS, and JavaScript-based design templates for creating responsive websites

Note: Always use the .min version of a JavaScript file to improve load time

Project Template: JavaScript Libraries Part 2

- Jquery-version.intellisense.js
  - Extending IntelliSense for jQuery library
- Jquery-version.js and jquery-version.min.js
  - Main JQuery version
- Jquery-version.min.map
  - Allows to map to the un-minified version of JQuery for troubleshooting purposes
- Jquery.validate.js (Jquery.validate.min.js)
  - Provide client side validation using jQuery
  - Multilanguage support
- Jquery.validate.unobtrusive.js and jquery.validate.unobtrusive.min.js

Client-side Development Configuration Files

- gruntfile.js
  - JavaScript configuration of Grunt tasks
- gulpfile.js
  - JavaScript configuration of Gulp tasks
- packages.config
  - Lists NuGet packages
- Package.json
  - Lists npm packages
- Bower.json
  - Lists Bower packages

Demo: Default JavaScript Libraries

Module 6: Client-side Development

Section 2: VS Client-side Dev Tooling

Lesson: Bower and Gulp

Why Use Gulp (or Grunt) and Bower?

- Modern web applications incorporate various and rich client-side libraries, such as jQuery, TypeScript, Bootstrap etc.
- Easy management of client-side packages
- Automating build tasks such as scripts compilation, bundling, minification or unit testing
- Use the existing tools from the web development community

What Is Bower?

- “A package manager for the web” (http://bower.io)
- Installs and restores client-side libraries
- Keeps track of all the packages in a manifest file, bower.json
- Improves page load

What is Gulp?

- “The JavaScript Task Runner” (http://gulpjs.com)
- An application to automate routine client-side development tasks (compilation, bundling, minification, unit testing, etc.)
- gulpfile.js contains Gulp tasks with JavaScript-like configuration
- Grunt is another task runner
    - Gulpfile.js uses JSON-like syntax for configuration

Bundling and Minification

- Bundling reduces the number of requests to the server
  - Combining multiple files into a single file
  - Create CSS, JavaScript and other bundles
- Minification reduces the size of the requested assets (CSS and JavaScript)
- To achieve bundling and minification in ASP.NET MVC
  - Use the “Include” or “IncludeDirectory” of the Bundle Class
  - Use Gulp or Grunt tasks

Demo: Bower and Gulp

Module 6: Client-side Development

Section 3: Development Techniques

Lesson: JavaScript and jQuery

Using JavaScript in MVC

- JavaScript scripts can be defined inside a View using the html script tag like in a html page
- Use the MVC @section tag to organize JavaScript scripts
  - The @RenderSection is used to inject JavaScript at a desired location inside the View
- For best practices, declare JavaScript scripts inside a .js file
- Use a minification tool in Visual Studio for optimization
- IntelliSense support for JavaScript in Visual Studio

jQuery and Microsoft

- Lightweight open source JavaScript library
- Deprecated Microsoft.Ajax libraries in favor of jQuery
- Distributed jQuery library with Visual Studio projects since 2008
- Extended Microsoft product support for jQuery
  - Enterprises can open jQuery support cases 24x7 with Microsoft Support
- Integrated Client template support
- Default templates use jQuery

jQuery

- Reduces client-side coding
- CSS 3-based syntax for traversing and manipulating DOM
- Concise wrappers for Ajax calls
- Abstracted to eliminate cross-browser differences
- Unobtrusive client validation
- XPath selectors to access elements in the DOM
- Elements are retrieved in the form of jQuery objects
- Start with jquery(), jquery., $() or $. to use jQuery

jQuery: Selectors

- Execute commands on a single or multiple selected DOM elements
- Basic types of selectors:
  - Based on HTML elements IDs. For example, $("#main")
  - Based on cascading style sheets (CSS). For example, $(".header")
  - Based on element tags. For example, $("div")
  - Based on element attributes. For example, $("[type = 'button']")
- Build more complex selectors through combination
  - $("#main p.quote") ⇔ Select paragraphs with a "quote" CSS class located inside elements with IDs equal to "main"
- $(this) operator

jQuery: Selectors (continued)

- Specific operators are used to expand selection options
  - A white space selects all elements that are descendants of the given ancestor. For example, $(“div p”) ⇔ $(“div”).find(“p”)
  - The > operator selects direct child elements of the given ancestor. For example, $(“div > p”) ⇔ $(“div”).children(“p”)
  - The + operator selects adjacent elements. For example, $(“div + p”) ⇔ $(“div”).next(“p”)
  - The ~ operator selects all siblings elements. For example, $(“div ~ p”) ⇔ $(“div”).nextall(“p”)
  - The comma operator selects all the specified elements. For example, $(div, p, a)

jQuery: Filters

- Used with Selectors, or alone
  - **Positional filters** :first, :even, :eq(index), :gt(index), :not (selector) etc.
  - **Child filters** :nth-child(expression), :first-child, :only-child
  - **Content filters** :contains(text), has(selector), :parent, :empty
  - **Form filter** :visible, :hidden, :button, :input, :selected

- Can be chained by appending with colon (:) *

* Examples in the notes section:

jQuery: Methods

- Class/Style Methods
  - Used to apply CSS styles to the result of a selector
  - .addClass(), .css(), .height(), .position()
- DOM Methods
  - .before(), .insertBefore(), .append(), .empty(), .attr()

jQuery: Events

- jQuery simplifies events implementation
- Events are triggered by the page or end user's interaction
- Events are often used to attach a callback function
- To bind an event:
  - Use of function to bind a event directly. For example, .click()
  - Use .on() For example, .on("click", …)
  - Use .bind() For example, .bind("click", …)
  - Use .live() For example, .live("click, …)

Unobtrusive JavaScript

- Traditional use of JavaScript
  - <input type="button" value="Click me" onclick="handleClick()" />
- For cleaner HTML page, remove inline JavaScript references
- Use jQuery to attach handlers to DOM elements

```
<script type="text/javascript">
    $(document).ready(function () {
            $(“:button").bind("click", function () {
                    alert(“Hello World!”);
            });
    });
</script>
<div>
    <input type="button" value="Click me" />
<div>
```

Demo: jQuery selectors and events

Module 6: Client-side Development

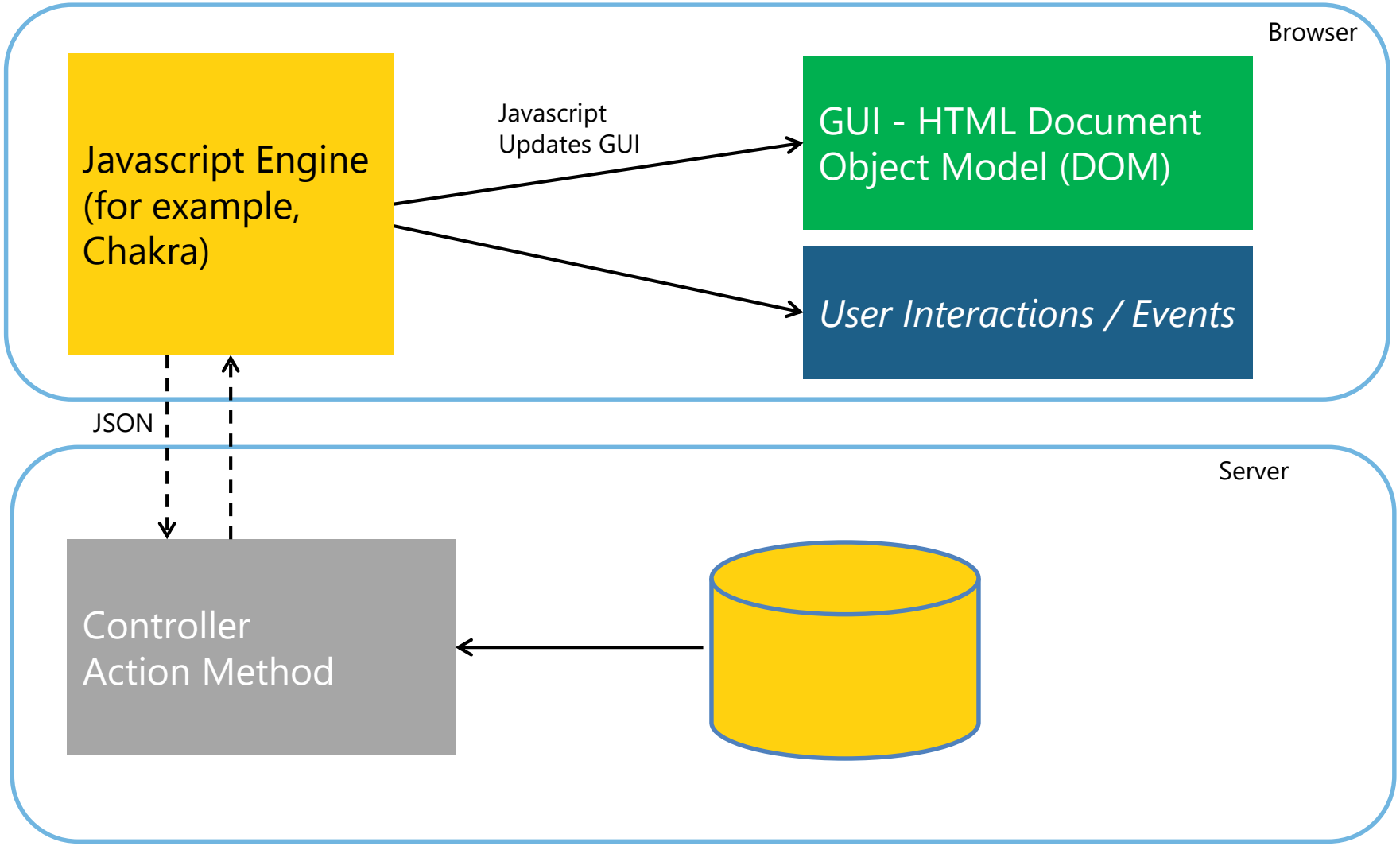
Section 3: Development Techniques

Lesson: AJAX

AJAX: (Asynchronous JavaScript and XML)

- Send and receive data from a server asynchronously (in the background) – better performance
- Uses XMLHttpRequest object
- JSON typically used instead of XML
- Back button and bookmarking challenges
- Downgrade challenges for mobile devices that do not support JavaScript
- Webcrawlers do not index pages created by Ajax
- Does not work across domains by default

AJAX Engine


Browser
Javascript Engine (for example, Chakra)
Javascript Updates GUI
GUI - HTML Document Object Model (DOM)
User Interactions / Events
JSON
Server
Controller Action Method

AJAX in Jquery

- Simplifies Ajax implementation in JavaScript
- The full-feature .ajax() method performs an asynchronous HTTP (Ajax) request
- Shortcuts: .get(), .getScript(), .getJSON(), .post(), .load()
- $.ajaxSetup() method specifies settings for an Ajax call

Ajax in Actions

- Create a new ActionMethodSelectorAttribute

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method)]
public class AcceptAjaxAttribute : ActionMethodSelectorAttribute
{
    public override bool IsValidForRequest(
    ControllerContext controllerContext, MethodInfo methodInfo)
    {
        return controllerContext.HttpContext
        .Request.IsAjaxRequest();
    }
}
```

```
    [AcceptAjax]
    public ActionResult Details(int id)
    {
        var employee = _repository.FindEmployee(id);
        return View(employee);
    }
```

Demo: AJAX

Module Summary

- In this module, you learnt about:
  - ASP.NET MVC support for Client-side Technologies
  - Visual Studio Client-side Dev Tooling
  - JavaScript and jQuery
  - AJAX

Lab: Client-side Development


Microsoft