View, Layout, Area in MVC

Lesson Objectives

- *View in MVC*
- *Partial View*
- *Layout in MVC*
- *Area in MVC*

Lesson Objectives

- *Bundling and Minification*
- *Integrate Controller, View and Model*
- *Data Validation*

Section 1

VIEW

View in MVC

- View: what thing user see
  - ✓ Design elements: image, logo, title, …
  - ✓ Html element: header tags, link, text, …
  - ✓ Model data: name of student, price of product, …
  - ✓ User interact: to enter name of student, to select region,…

View and Model

- View often display data from one model
- View can display data from more than one model
  - ✓ Let's use ViewModel
- View can display data from a part of the model
  - ✓ Let's use ViewModel

View

- View contains
  - ✓ Html code
  - ✓ CSS, JS code or references
  - ✓ Business code to display model/viewmodel
- Stored in folder with name is name of controller
- Have file name extension: *.cshtml, *.vbhtml

Render view

```
public class HomeController : Controller
{

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

- If not specify
  - ✓ Find view with name is Index (name of the action) in folder Views/Home (name of controller)
  - ✓ If not, find in Shared folder
- If specify:
  - ✓ Specify view name only
  - ✓ Specify full path

Section 2

PARTIAL VIEW

Partial View

- Partial view in MVC is a view that is rendered within another view. The HTML output generated by executing the partial view is rendered into the calling (or parent) view.

Partial View

- Use partial view:
  - ✓ breaking up large views into smaller components
  - ✓ reduce duplication of view content
  - ✓ allow view elements to be reused

Render Partial View

- Html.RenderPartial()
- Html.Partial()
- Html.RenderAction()
- Html.Action()
- jQuery load function

Partial() vs. RenderPartial()

Html.Partial()

- Returns html string
- Injects the html string of the partial view into the main view.
- Performance is slow.
- Need not to be inside the braces.

Html.RenderPartial()

- Returns void.
- Writes html in the response stream.
- Perform is faster compared
- Must be inside braces @{ }.

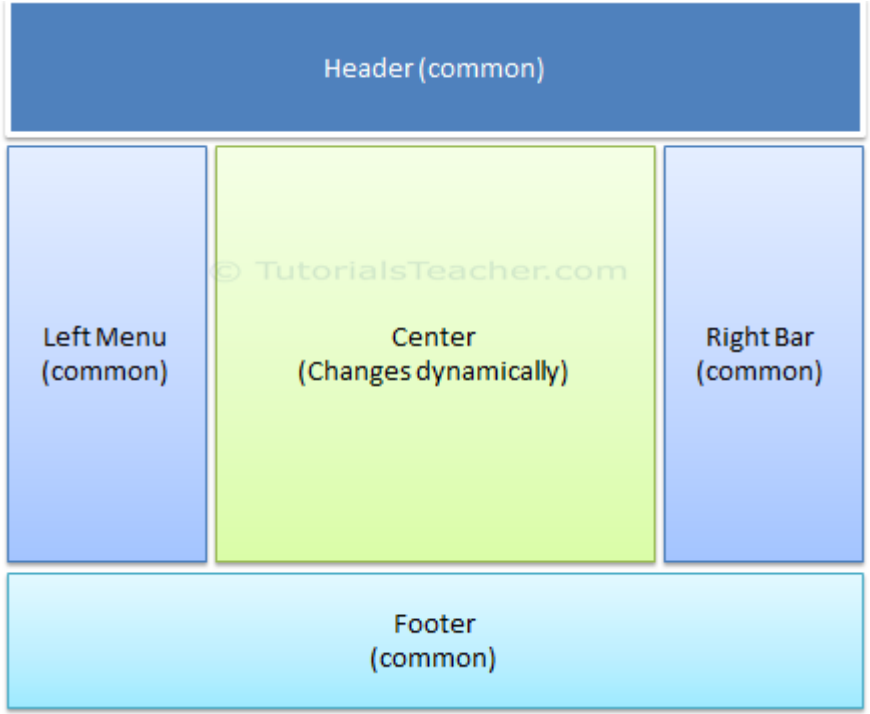
Question to discuss

- Both view and partial view have *.cshtml extension. So, what is convention for name of partial view?
- How to share partial view from difference controllers?
- Inside partial view, can we call the nested partial views?

Section 3

LAYOUT

Layout



- Application always has consistent frame
- It contains common parts in the UI
- Static/common parts can be load one time to improve performance
- Developer can reduce cost to write same code in every page

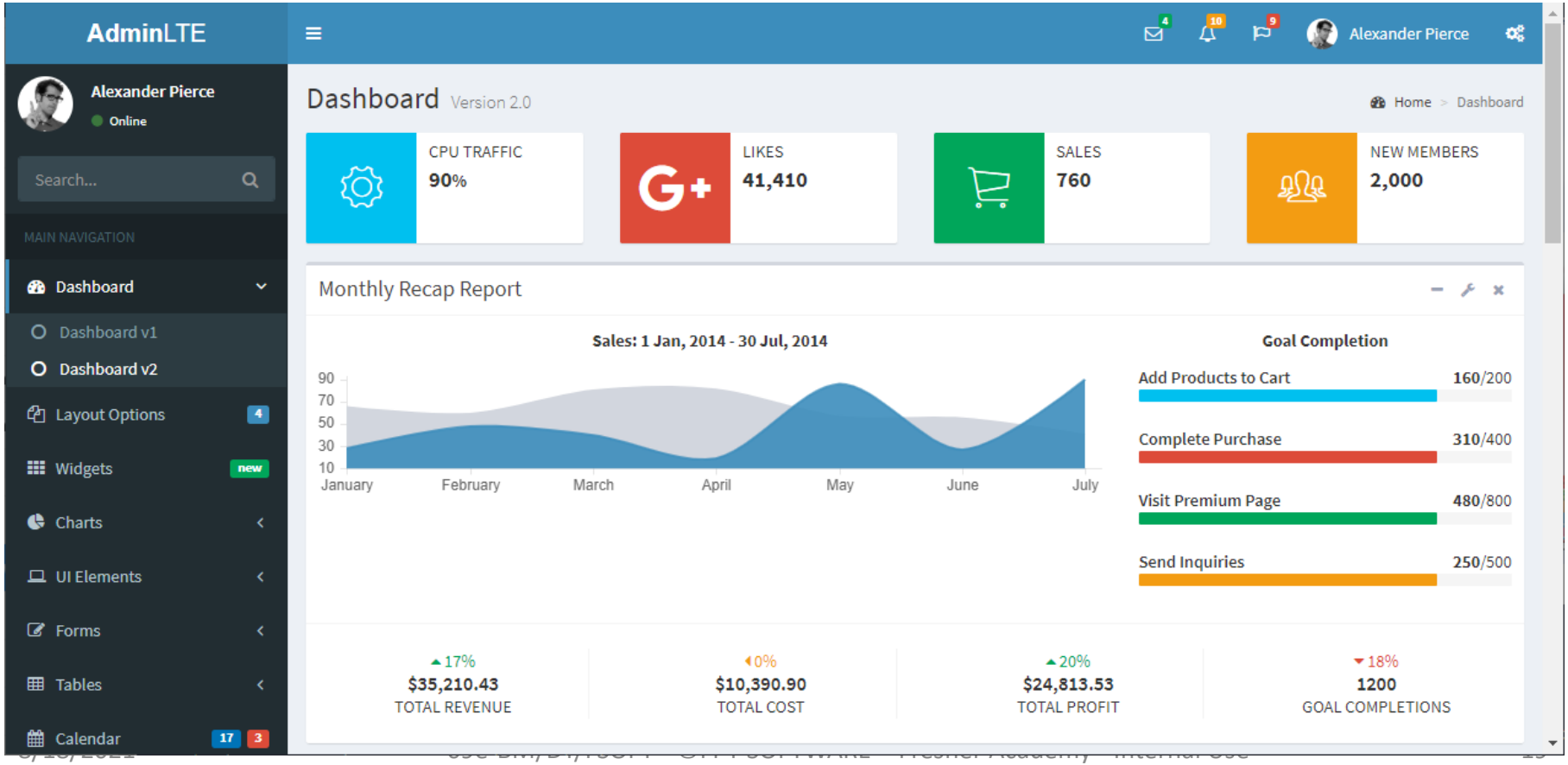
Layout view

- The view that:
  - ✓ Implement UI frame
  - ✓ Contains common UI parts
  - ✓ Define a common site template
  - ✓ Load and share resources between views
  - ✓ Do not depend on particular object

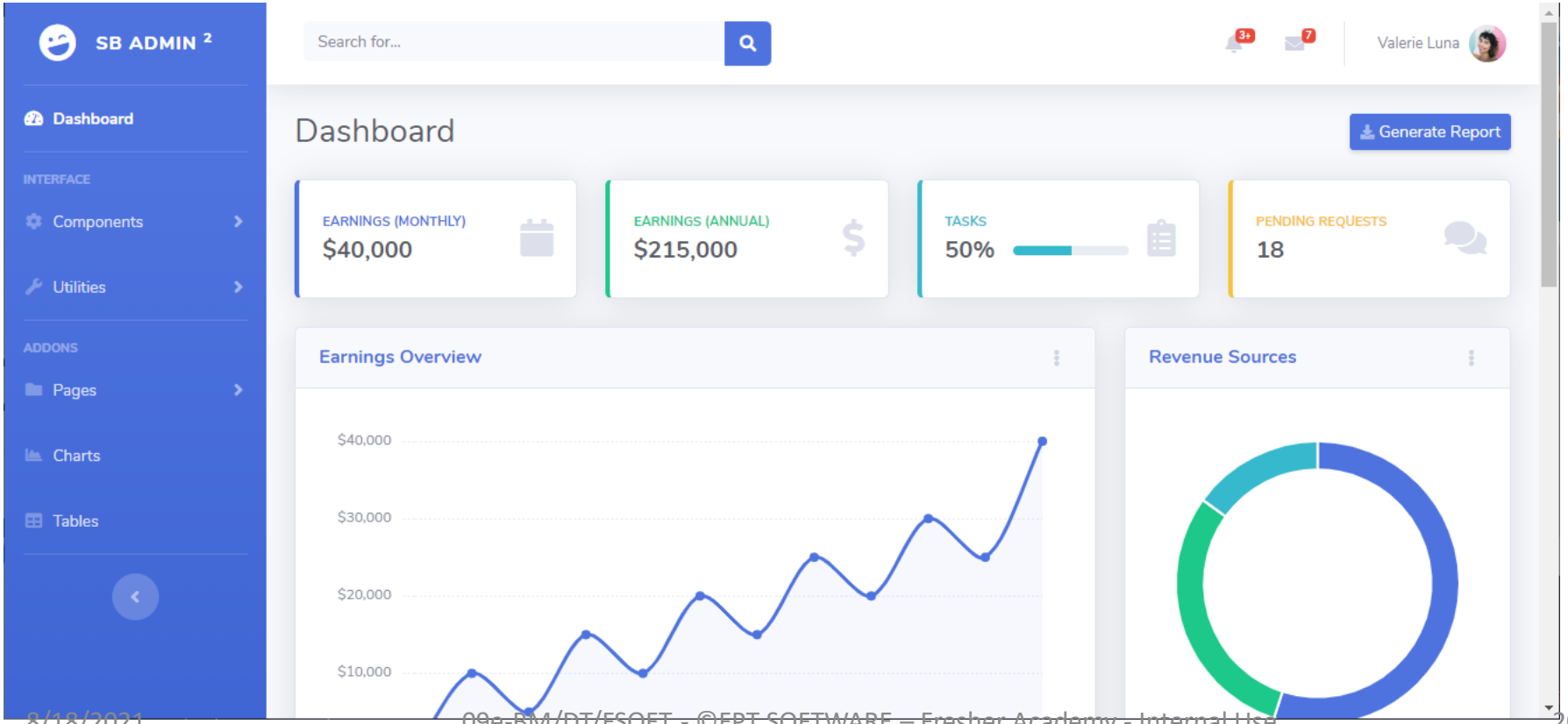
Layout view

- Layout views are shared with multiple views, so it is often stored in the Shared folder
- Layout view has same extension as other views. To specify, we use _**Layout** prefix
- Every view use one and only one layout view
- If layout view is not specified, MVC use default layout in _ViewStart.cshtml

Populate layout- adminlte.io



Populate layout- sb-admin



Additional section

AREA

AREA

- Large application can include a large number of controller, views and model classes.
- Difference groups of users have difference experience, roles and behaviours.

AREA

- Maintain a large number of views, models and controllers with the default ASP.NET MVC project structure can become unmanageable
- Area allows us to partition large application into smaller units

Area

- Each unit contains separate MVC folder structure, same as default MVC folder structure.
- For example, we can create areas for admin, finance, HR, marketing etc.
- All areas need to register in Application_Start event in Global.asax.cs as AreaRegistration.RegisterAllAreas();

Area

- Demo: create new area and configure

Section 4

BUNDLING AND MINIFICATION

Bundling and Minification

- Bundling and minification are two techniques to improve request load time.
- Bundling and minification improves load time by reducing the number of requests to the server and reducing the size of requested assets (such as CSS and JavaScript.)

Bundling

- Bundling combines or bundles multiple files into a single file.
- You can create CSS, JavaScript and other bundles.
- Fewer files means fewer HTTP requests and that can improve first page load performance.

Minification

- Minification performs a variety of different code optimizations to scripts or css, such as removing unnecessary white space and comments and shortening variable names to one character.

Use in ASP.NET MVC

- Use BundleCollection to register
- Use ScriptBundle to register scripts
- Use StyleBundle to register stylesheet
- Use the "*" wildcard character to select multiple files
- Use resource bundled by virtual path

Use in ASP.NET MVC

```
1 reference
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 with and learn from. T
    // ready for production, use the build tool at https://modernizr.com to pick
    bundles.Add(new ScriptBundle("~/bundles/modernizr").Include(
                "~/Scripts/modernizr-*"));

    bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(
              "~/Scripts/bootstrap.js"));

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

Section 6

DATA VALIDATION

Data Validation

- Any data sent to server need to validate
- Validation approach:
  - ✓ Client validation
  - ✓ Server validation
  - ✓ Combined of client and server validation

Client Validation

- Pros:
  - ✓ Fast
  - ✓ User friendly
  - ✓ Reduce server workload
- Cons:
  - ✓ Not secure enough
  - ✓ Can not access resource, data in server
  - ✓ Unmanaged code in client

Client Validation

- Uses:
  - ✓ Javascript
  - ✓ jQuery
  - ✓ jQueryValidator

Server Validation

- Pros:
  - ✓ Can implement complex business
  - ✓ Can access any resource, data
  - ✓ Secure
- Cons:
  - ✓ Slow
  - ✓ Use unfriendly
  - ✓ Increment server workload

Combined of client and server validation

- Validate both in client and server
- User friendly, fast, secure
- More code to implement

DataAnnotations

- MVC uses DataAnnotations attributes to implement validations.
- DataAnnotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class.
- MVC will automatically enforce these validation rules and display validation messages in the view.

DataAnnotations

Attribute	Description
Required	Indicates that the property is a required field
StringLength	Defines a maximum length for string field
Range	Defines a maximum and minimum value for a numeric field
RegularExpression	Specifies that the field value must match with specified Regular Expression

DataAnnotations

Attribute	Description
CreditCard	Specifies that the specified field is a credit card number
CustomValidation	Specified custom validation method to validate the field
EmailAddress	Validates with email address format

DataAnnotations

Attribute	Description
FileExtension	Validates with file extension
MaxLength	Specifies maximum length for a string field
MinLength	Specifies minimum length for a string field
Phone	Specifies that the field is a phone number using regular expression for phone numbers

ErrorMessage

- Gets or sets an error message to associate with a validation control if validation fails.
- Format: [Required(ErrorMessage = "You must specify data for this field.")]

Best Practices

- Always validate as much as possible
- Focus on the first invalid controls after validate
- Highlight all invalid controls
- Use appropriate message for each type of error

Lesson Summary

Thank you