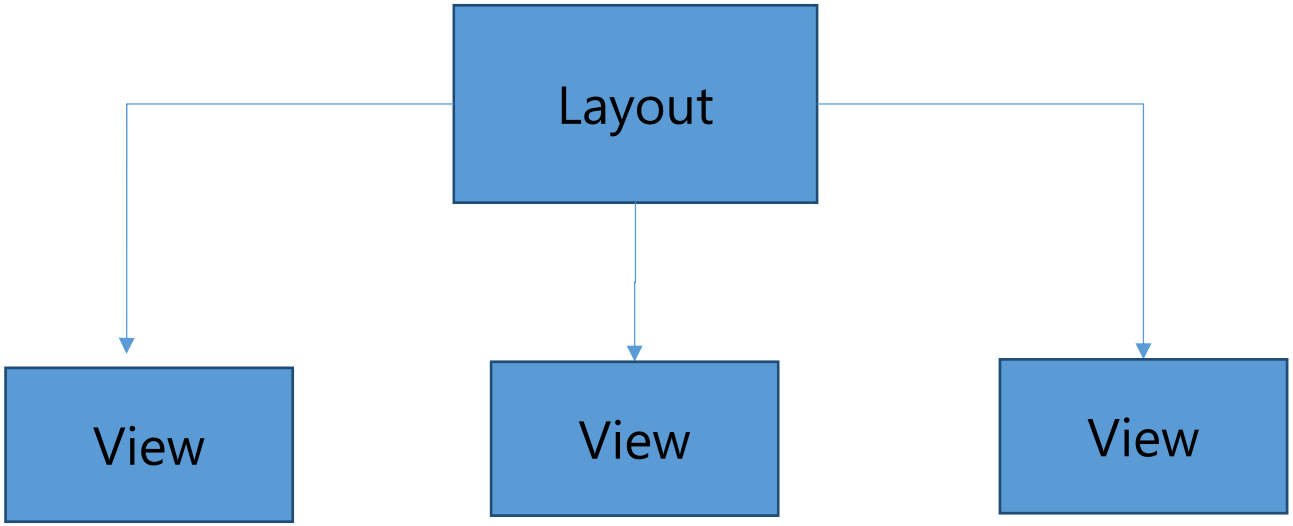
Layouts

Layout

- Un layout (ou page de disposition) définit un template pour les vues.
- Views/shared est le dossier par défaut pour les layouts.
- Le contenu des vues est inséré dans le layout à l'aide de la méthode @RenderBody().
- L'objet dynamique ViewBag est utilisé pour passer des données entre les vues et les layouts.



Exemple

la méthode @RenderSection permet de générer une section, si l'attribut required est égal à true alors une exception sera déclenchée dans le cas où la section n'est pas définie dans la vue.

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
</head>
<body>
    <div
id="section1">@RenderSection("section1",required:false)</div>
    <div>
        @RenderBody()
    </div>
</body>
</html>
```

Lier une vue à un layout

- La directive Layout permet d'associer un layout à une vue.
- Le code de la vue s'exécute avant le code du layout

```
@{
    ViewBag.Title = "Titre de la page";
    Layout = "~/Views/Shared/_template1.cshtml";

}
```

- La directive @section définit une section dans la vue.

```
@section section1{
    <p> C'est une section</p>
}
```

- Le code du fichier _ViewStart.cshtml s'exécute avant le code des vues de l'application

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Les Bundles css et javascript

- Installez la bibliothèque Microsoft.AspNet.Web.Optimization.
- Ajoutez dans le dossier App_Start une classe (BundleConfig.cs)
- Ajoutez dans la classe la méthode statique suivante:

```
public static void
RegisterBundles(System.Web.Optimization.BundleCollection bundles){
 bundles.Add(new
 System.Web.Optimization.ScriptBundle("~/bundles/jquery")
 .Include("~/Scripts/jquery-{version}.js","~/Scripts/script1.js"));
}
```

- La classe ScriptBundle permet de combiner plusieurs fichiers javascript dans un fichier unique, si la version minifiée existe dans le dossier Scripts elle sera utilisée, la présence du paramètre {version} permet de sélectionner la version la plus récente dans le dossier.

Les Bundles css et javascript

- Dans l'événement Application_Start du fichier global.asax, enregistrez les paquets en appelant la méthode statique BundleConfig.RegisterBundles

```
BundleConfig.RegisterBundles(System.Web.Optimization.BundleTable.Bun
dles);
```

- Dans le fichier web.config, ajoutez une référence du namespace System.Web.Optiomization, ce qui permet le support de la fonctionnalité @Script.Render dans les vues

```
<configuration>
   <system.web>
       <pages>
         <namespaces>
                        <add
namespace="System.Web.Optimization"/>
```

@Ajax.ActionLink

- `@Ajax.ActionLink("Légende ", "action", "contrôleur", optionsAjax)`
  - `Le dernier paramètre est un objet de type` System.Web.Mvc.Ajax.AjaxOptions.
- Propriétés de l'objet AjaxOptions
  - Confirm: contient le message à afficher à l'utilisateur pour confirmer l'action avant son exécution
  - HttpMethod
  - UpdateTargetId: id de l'élément html dans lequel le résultat de la requête Ajax sera affiché.
  - InsertionMode: définit comment le résultat sera affiché dans l'élément cible, quatre valeurs possible : InsertionMode.InsertBefore, InsertionMode.InsertAfter, InsertionMode.Replace, Insertion.ReplaceWith.
  - LoadingElementId et LoadingElementDuration
  - Evénements: OnBegin, OnComplete, OnSuccess, OnFailure