Form

HTML form

- ជាការបង្កើត form ដោយប្រើប្រាស់ន្ធវធាតុ html។
- Attributes
  - action attribute
  - method attribute(get,post)

ឧទាហរណ៍

```
<h2>Create</h2>
<div class="row">
    <div class="col col-md-5 ml-2">
        <form action="/product/create" method="post">
            <div class="form-group">
                <label for="Name">Name</label>
                <input type="text" name="Name" class="form-control" />
            </div>
            <div class="form-group">
                <label for="price">Price</label>
                <input type="text" name="price" class="form-control" />
            </div>
            <div class="form-group">
                <label for="description">description</label>
                <input type="text" name="description" class="form-control" />
            </div>
            <input type="submit" name="submit" class="btn btn-danger" />
        </form>
    </div>
</div>
```

ឧទាហរណ៍

```
public class ProductController : Controller
{

    [HttpGet]
    0 references
    public IActionResult Create()
    {
        return View();
    }
    [HttpPost]
    0 references
    public IActionResult Create(string name, decimal price, string description)
    {
        return Content($"{name}-{price}-{description}");
    }
```

បង្កើតFormដោយប្រើ Tags Helper

- ជាការបង្កើត form ដោយពឹងផ្អែកលើ class object។
- មុននិងអាចបង្កើតformតាមរយៈ tag helper បានដាច់ខាតត្រូវប្រើប្រាស់នូវ @model ជាមុនសិនដែលវាត្រូវស្ថិតនៅផ្នែកខាងលើបំផុតនៃ view។
- និងត្រូវ import namespaceជាមុនសិន
  - @addTagHelper *,Microsoft.AspNetCore.Mvc.TagHelpers

ឧទាហរណ៍

```
@addTagHelper *,Microsoft.AspNetCore.Mvc.TagHelpers
@model Product

<h2>Create</h2>
<div class="row">
    <div class="col col-md-5 ml-2">
        <form action="/product/create" method="post">
            <div class="form-group">
                <label asp-for="ProductName">Name</label>
                <input type="text" asp-for="ProductName" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="UnitPrice">Price</label>
                <input type="text" asp-for="UnitPrice" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="Description">description</label>
                <input type="text" asp-for="Description" class="form-control" />
            </div>
            <input type="submit" name="submit" class="btn btn-danger" />
        </form>
    </div>
</div>
```

tag helper

use @model directive

Tags Helper

- Tag helper វាផ្តល់ភាពងាយស្រួលដល់យើងសរសេរធាតុ html ជាមួយ razor syntaxដែលវាមានលក្ខណៈដូចគ្នានិង html standard code ប៉ុន្តែត្រូវបានដំណើរការដោយrazor engineនៅលើ server។

Tags Helper

- Form tag helper
- Input tags helper
- Label tag helper
- Anchor tag helper

Form Tag Helper

- Form tag helper គឺវាត្រូវភ្ជាប់ជាមួយ <form> element
- Form tag helper វាផ្តល់នូវ server-side attributes មួយចំនួនដែលវាជួយអោយហើយក្នុងបង្កើតនូវ html។

Form Tag Helper

attribute	មុខងារ
asp-action	តំលៃរបស់វាគឺជា action method របស់ controller ណាមួយ
asp-controller	តំលៃជាឈ្មោះរបស់ controller ណាមួយដែលត្រូវប្រើ
asp-area	តំលៃរបស់ជាឈ្មោះ area នៃcontroller ណាមួយដែលត្រូវប្រើ
asp-route-{value}	កំណត់តំលៃអោយ route parameterតែមួយ(ឧ. asp-route-id="23")
asp-route	កំណត់ឈ្មោះrouteដែលបានកំណត់ក្នុង config method នៃ startup class
asp-all-route-data	ចំពោះroute មាន parameters ច្រើន

ឧទាហរណ៍ៈ

```
@addTagHelper *,Microsoft.AspNetCore.Mvc.TagHelpers
@model Product
<h2>Create</h2>
<div class="row">
    <div class="col col-md-5 ml-2">
        <form asp-action="create" asp-controller="product" asp-area="" method="post" >
            <div class="form-group">
                <label asp-for="ProductName">Name</label>
                <input type="text" asp-for="ProductName" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="UnitPrice">Price</label>
                <input type="text" asp-for="UnitPrice" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="Description">description</label>
                <input type="text" asp-for="Description" class="form-control" />
            </div>
            <input type="submit" name="submit" class="btn btn-danger" />
        </form>
    </div>
</div>
```

ឧទាហរណ៍៖

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseStaticFiles();
    app.UseRouting();
    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute("home",
                                     "{controller}/{action}/{id?}",
                                     new { controller = "Product", action = "Index" });

    });
}
```

Anchor Tag Helper

attribute	មុខងារ
asp-action	តំលៃរបស់វាគឺជា action method របស់ controller ណាមួយ
asp-controller	តំលៃជាឈ្មោះរបស់ controller ណាមួយដែលត្រូវប្រើ
asp-area	តំលៃរបស់ជាឈ្មោះ area នៃcontroller ណាមួយដែលត្រូវប្រើ
asp-route-{value}	កំណត់តំលៃអោយ route parameterតែមួយ(ឧ. asp-route-id="23")
asp-route	កំណត់ឈ្មោះrouteដែលបានកំណត់ក្នុង config method នៃ startup class
asp-all-route-data	ចំពោះroute មាន parameters ច្រើន

ឧទាហរណ៍៖

```
<a asp-action="detail" asp-controller="employee" asp-route-id="12">Detail</a>
```

```
<a href="/employee/detail/12">Detail</a>
```

Label tag helper

- Label tag helper វានិងបង្ហាញជា label ក្នុង html វាមានតែមួយ attribute server-side តែប៉ុន្នោះគឺ asp-for។

ឧទាហរណ៍

```
<form asp-action="create" asp-controller="product" asp-area="" method="post" >
    <div class="form-group">
        <label asp-for="ProductName">Name</label>
        <input type="text" asp-for="ProductName" class="form-control" />
    </div>
    <div class="form-group">
        <label asp-for="UnitPrice">Price</label>
        <input type="text" asp-for="UnitPrice" class="form-control" />
    </div>
    <div class="form-group">
        <label asp-for="Description">description</label>
        <input type="text" asp-for="Description" class="form-control" />
    </div>
    <input type="submit" name="submit" class="btn btn-danger" />
</form>
```

Input tags helper

- Input tags helper ត្រូវបានគេប្រើប្រាស់ជាមួយ<input /> របស់ html
- វាមាន server-side attribute ចំនួនពីរគឺ.ៈ
  - asp-for
    - វានិងភ្ជាប់(bind)ជាមួយនិង model propertyហើយវានិងបង្ហាញ html អាស្រ័យ type, name និង data annotation នៃ model property
  - asp-format

Input tags helper

- Asp-for វានិងបង្កើត html type attribute ដោយស្វ័យប្រវត្តិដោយបឹងផ្អែកលើលក្ខ័ណនិងលំដាប់ដូចខាងក្រោម(The asp-for automatically generates the HTML type attribute based on the following criteria and in the order specified)ៈ
  - Type Specified in the HTML
  - The Data annotation attribute applied to the model property.
  - The .NET data type model type is used

Data Annotation attribute

ATTRIBUTE	INPUT TYPE
[EmailAddress]	type=”email”
[Url]	type=”url”
[HiddenInput]	type=”hidden”
[Phone]	type=”tel”
[DataType(DataType.Password)]	type=”password”
[DataType(DataType.Date)]	type=”date”
[DataType(DataType.Time)]	type=”time”

Based on the .NET data type

NET TYPE	INPUT TYPE
Bool	type=”checkbox”
String	type=”text”
DateTime	type=”datetime-local”
Byte, int, Single, Double	type=”number”
decimal, double, float	type=”text”

ឧទាហរណ៍

```
<form asp-action="create" method="post" asp-controller="employee">
    <div asp-validation-summary="All"></div>
    <div>
        <label asp-for="Name">Name</label>
        <input type="text" asp-for="Name" />
        <span asp-validation-for="Name"></span>
    </div>
    <div>
        <label asp-for="Age">Age</label>
        <input asp-for="Age" value="" />
        <span asp-validation-for="Age"></span>
    </div>
    <div>
        <label asp-for="Sex"></label>
        <input type="radio" asp-for="Sex" value="male" />Male
        <input type="radio" asp-for="Sex" value="female" />female
        <span asp-validation-for="Sex"></span>
    </div>
    <input type="submit" value="Submit" />
</form>
```

Select Tag Helper

attribute	មុខងារ
asp-for	សម្រាប់bind ជាមួយ model object propperty
asp-items	ជាបណ្តុំនៃ selectlist item

ឧទាហរណ៍៖

create.cshtml

```
@model Employee
@{
    ViewBag.Title = "Create";
}
<form asp-action="create" method="post" asp-controller="employee">
    <div asp-validation-summary="All"></div>
    <div>
        <label asp-for="Sex"></label>
        <select asp-for="Sex" asp-items="@((List<SelectListItem>)ViewBag.Sex)">
            <option value="">Select your gender</option>
        </select>
        <span asp-validation-for="Sex"></span>
    </div>
    <input type="submit" value="Submit" />
</form>
```

EmployeeController.cs

```
public ViewResult Create()
{
    var sex = new List<SelectListItem>
    {
        new SelectListItem {Text="Female",Value="Female"},
        new SelectListItem {Text="Male",Value="Male"},
    };
    ViewBag.Sex = sex;
    return View();
}
```

Model Binding

- វាជាតំណើរការនៃការផ្គូរផ្គងទិន្នយដែលបានបញ្ជូនទៅកាន់ server តាមរយៈ Http request ជាមួយនិង parameters នៃ action method របស់controller ណាមួយ
  (The **Model binding** is the process of mapping the data posted over an HTTP request to the parameters of the action method in the Controller)។
- Asp.net mvc និងធ្វើការស្វែងនូវតំលៃ argument របស់ action method តាមបីរបៀបគឺ៖
  - Form data
  - Routing variable
  - Query string

Simple Type Binding

- Simple type binding ជាការ bind រវាងprimitive type(string, integer, Boolean ….)ជាមួយនិង arguments នៃaction method របស់ controller។
- គ្រប់requests(keys)ទាំងអស់ត្រូវតែមានឈ្មោះដូចគ្នានិង action method parameters' name
- ដំណើរការនៃ model binding វាស្វែងរកតំលៃនៃ arguements តាមរយៈដូចជា៖
  - Form data values
  - Routing variables
  - Query strings

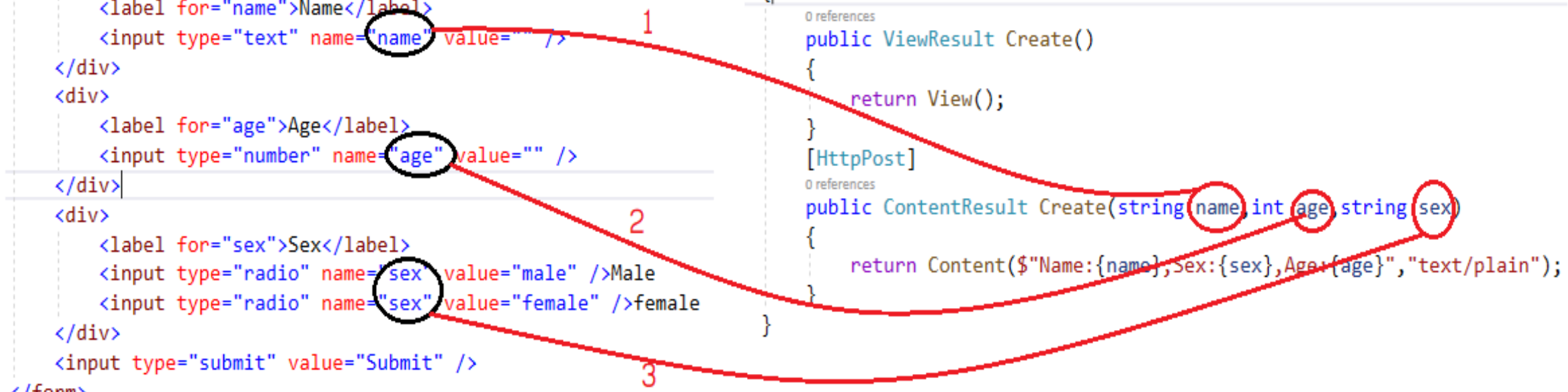
ឧទាហរណ៍

create.cshtml

```
<form action="/employee/create" method="post">
    <div>
        <label for="name">Name</label>
        <input type="text" name="name" value="" />
    </div>
    <div>
        <label for="age">Age</label>
        <input type="number" name="age" value="" />
    </div>
    <div>
        <label for="sex">Sex</label>
        <input type="radio" name="sex" value="male" />Male
        <input type="radio" name="sex" value="female" />female
    </div>
    <input type="submit" value="Submit" />
</form>
```

EmployeeController

```
0 rererences
public class EmployeeController : Controller
{
    0 references
    public ViewResult Create()
    {
        return View();
    }
    [HttpPost]
    0 references
    public ContentResult Create(string name,int age,string sex)
    {
        return Content($"Name:{name},Sex:{sex},Age:{age}","text/plain");
    }
}
```



Complex type binding

- នៅពេលដែល argumentនៃaction method មានparameterជាcomplex type ដូចជា class object ជាដើមនោះដំណើរការនៃ model binding ដោយវាចាប់យកតាល់public properties នៃcomplex type ហើយវាធ្វើការភ្ជាប់ម្តងមួយៗ(When the argument of the action method is a complex type like a class object then Model Binding process gets all the public properties of the complex type and performs the binding of each of them)។
- ដំណើរការនៃ model binding វាស្វែងរកតំលៃនៃpublic properties តាមរយៈដូចជា៖
  - Form data values
  - Routing variables
  - Query strings

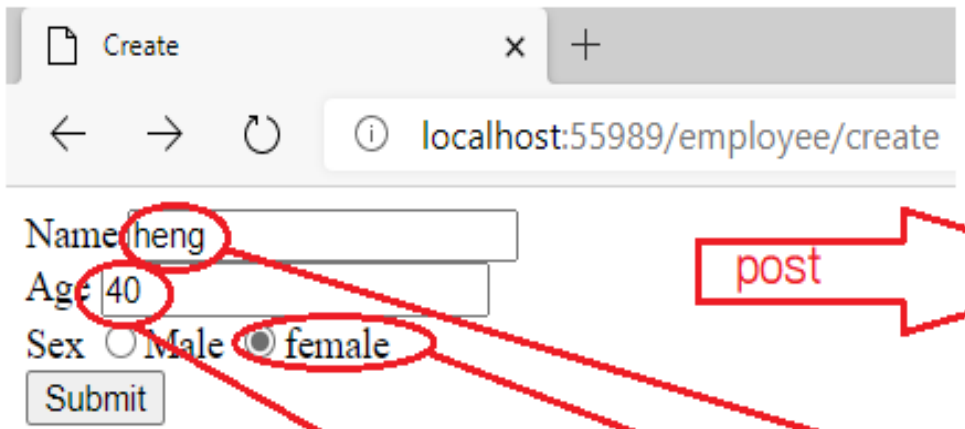
Complex type binding

- Property name ត្រូវតែផ្គូរផ្គងជាមួយនិង request data
- Property ត្រូវតែជាប្រភេទ public និង read-write

ឧទាហរណ៍

```
namespace Middleware.Entities
{
    4 references
    public class Employee
    {
        6 references
        public string Name { get; set; }
        4 references
        public int Age { get; set; }
        7 references
        public string Sex { get; set; }
    }
}
```

ឧទាហរណ៍



post

```
public class EmployeeController : Controller
{
    0 references
    public ViewResult Create()
    {
        return View();
    }
    [HttpPost]
    0 references
    public ContentResult Create(Employee emp)
    {
        return Content($"Name:{emp.Name},Sex:{emp.Sex},Age:{emp.Age}","text/plain");
    }
}
```

emp.Name="heng"

emp.Sex="female"

emp.Age=40

Model Validation

- វាជាយន្តការបំពេញតម្រូវការនៃការត្រួតពិនិត្យភាពត្រឹមត្រូវនូវ user's request ឬក៏ទិន្នន័យស្របតាមគោលការណ៍ឬកមានសុពលភាព(valid)ដែលគេចង់បានឬក៏រអត់ដើម្បីbind the model។
- គេមានវិធីសាស្ត្រចំនួនពាដើម្បីពិនិត្យភាពត្រឹមទិន្នន័យ៖
  - Server-side validation
  - Client-side validation
  - និង remote validation

Server-side validation

- Manually model validation
- Data Annotation model validation
  - Attribute validation

Manually Validation

- ModelState.AddModelError(string key, string Error Message)
- ModelState.IsValid

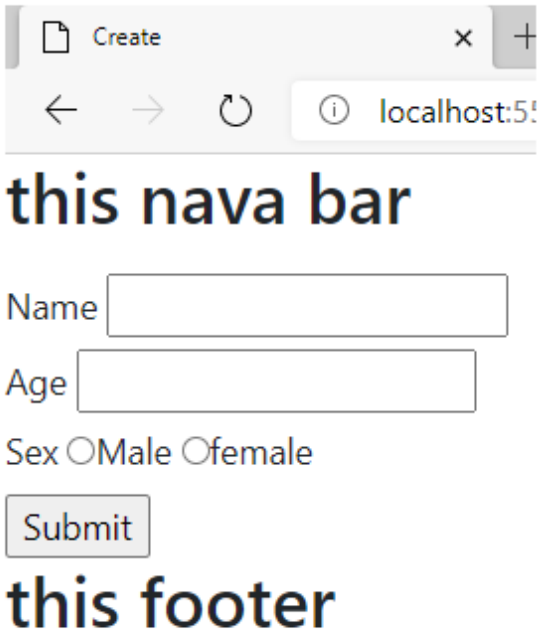
ឧទាហរណ៍

```
namespace Middleware.Entities
{
    4 references
    public class Employee
    {
        6 references
        public string Name { get; set; }
        4 references
        public int Age { get; set; }
        7 references
        public string Sex { get; set; }
    }
}
```

ឧទាហរណ៍

create.cshtml

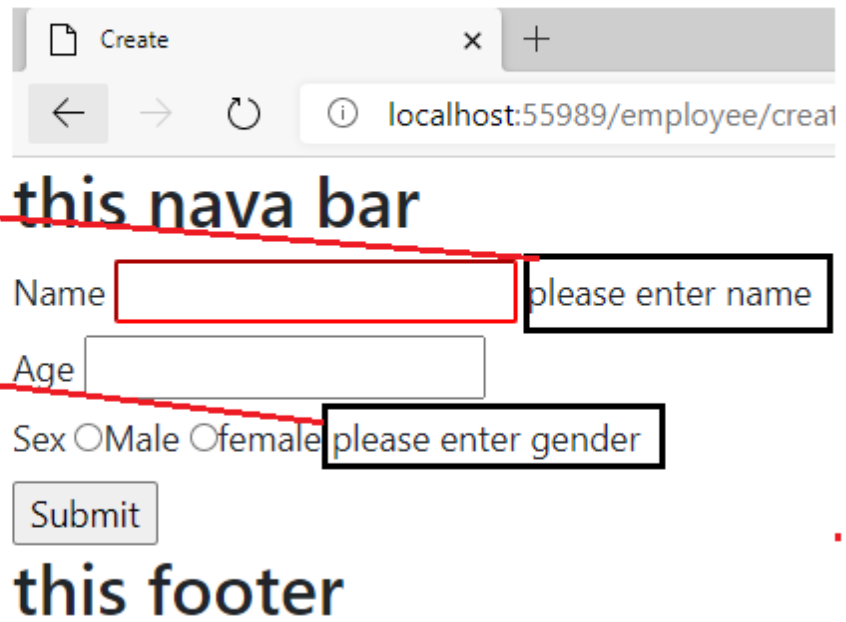
```
@model Employee
@{
    ViewBag.Title = "Create";
}
<form asp-action="create" method="post" asp-controller="employee">
    <div asp-validation-summary="All">

    </div>
    <div>
        <label asp-for="Name">Name</label>
        <input type="text" asp-for="Name" />
        <span asp-validation-for="Name"></span>
    </div>
    <div>
        <label asp-for="Age">Age</label>
        <input asp-for="Age" value="" />
        <span asp-validation-for="Age"></span>
    </div>
    <div>
        <label asp-for="Sex"></label>
        <input type="radio" asp-for="Sex" value="male" />Male
        <input type="radio" asp-for="Sex" value="female" />female
        <span asp-validation-for="Sex"></span>
    </div>
    <input type="submit" value="Submit" />
</form>
```



ឧទាហរណ៍

EmployeeController.cs

```
public ViewResult Create()
{
    return View();
}
[HttpPost]
0 references
public IActionResult Create(Employee emp)
{
    if (string.IsNullOrWhiteSpace(emp.Name))
    {
        ModelState.AddModelError(nameof(emp.Name), "please enter name");
    }
    if (string.IsNullOrWhiteSpace(emp.Sex))
    {
        ModelState.AddModelError(nameof(emp.Sex), "please enter gender");
    }
    if (!ModelState.IsValid)
    {
        return View(emp);
    }
    return Content($"Name:{emp.Name},Sex:{emp.Sex},Age:{emp.Age}","text/plain");
}
```



Display Error

- ដើម្បីបង្ហាញ errors message ត្រូវប្រើនូវ៖
  - **asp-validation-summary**
    - **All**
    - **ModelOnly**
    - **None**
  - **asp-validation-for**
    - Property name of model object

```
@model Employee
@{
    ViewBag.Title = "Create";
}
<form asp-action="create" method="post" asp-controller="employee">
    <div asp-validation-summary="All"></div>
    <div>
        <label asp-for="Name">Name</label>
        <input type="text" asp-for="Name" />
        <span asp-validation-for="Name"></span>
    </div>
    <div>
        <label asp-for="Age">Age</label>
        <input asp-for="Age" value="" />
        <span asp-validation-for="Age"></span>
    </div>
    <div>
        <label asp-for="Sex"></label>
        <input type="radio" asp-for="Sex" value="male" />Male
        <input type="radio" asp-for="Sex" value="female" />female
        <span asp-validation-for="Sex"></span>
    </div>
    <input type="submit" value="Submit" />
</form>
```

Data Annotation Attributes

- Data Annotation attribute គឺវាជួយយើងដើម្បីកំណត់នូវគោលការណ៏ឬក្បួនច្បាប់(rule)ទៅកាន់ model class ឬ propertiesដើម្បីdata validation និងបង្ហាញ message សមរម្យណាមួយទៅដល់end users(Data Annotations help us to define the rules to the model classes or properties for data validation and displaying suitable messages to end users)។
- *System.ComponentModel.DataAnnotations* namespace

Data Annotation Attributes

Attribute	Description
CreditCard	This validates that the property has a credit card format.
Compare	This attribute validates that two property in model class match like password and compare password.
EmailAddress	This validates the property has email address format.
Phone	This validates that the property has a telephone number format.
Range	This validates that the property value within a specified range.
RegularExpression	This validates that the property value matches a specified regular expression.
Required	This validates that the field is not null
StringLength	This validates that a string property value doesn't exceed a specified length limit.
Url	This validates that the property has a URL format.
Remote	This validates input on the client by calling an action method on the server

ឧទាហរណ៍ៈ

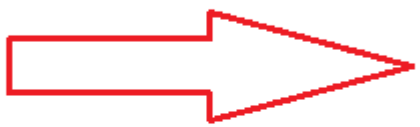
```
4 rererences
public class Employee
{
    [Required(ErrorMessage ="{0} is needed")]
    4 references
    public string Name { get; set; }
    [Required(ErrorMessage = "{0} is needed")]
    [Range(1,100)]
    4 references
    public int? Age { get; set; }
    [Required(ErrorMessage = "{0} is needed")]
    5 references
    public string Sex { get; set; }
}
```

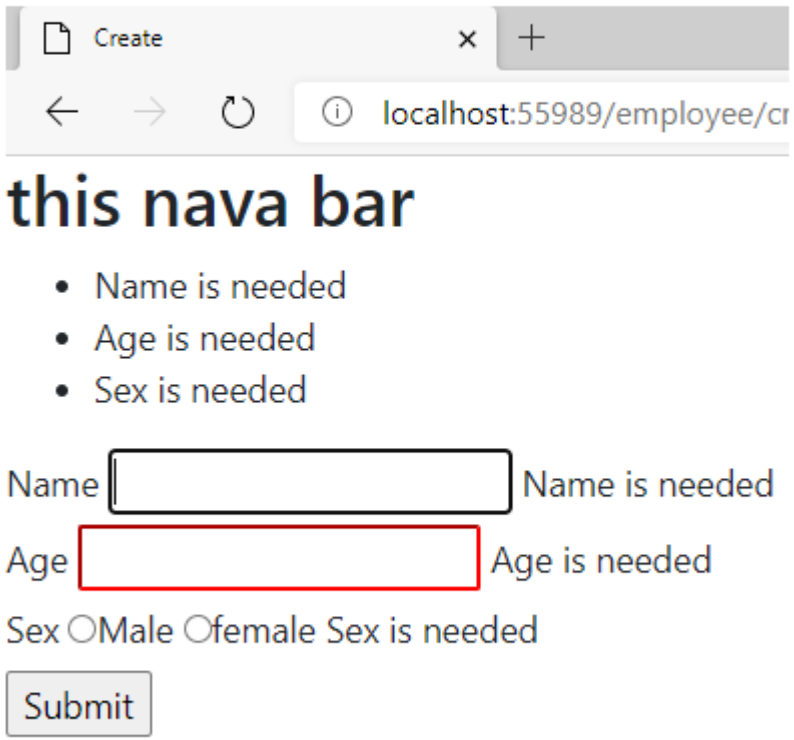
EmployeeController.cs

```
public ViewResult Create()
{
    return View();
}
[HttpPost]
0 references
public IActionResult Create(Employee emp)
{
    if (!ModelState.IsValid)
    {
        return View(emp);
    }
    return Content($"Name:{emp.Name},Sex:{emp.Sex},Age:{emp.Age}","text/plain");
}
```

ឧទាហរណ៍

```razor
@model Employee
@{
    ViewBag.Title = "Create";
}
<form asp-action="create" method="post" asp-controller="employee">
    <div asp-validation-summary="All"></div>
    <div>
        <label asp-for="Name">Name</label>
        <input type="text" asp-for="Name" />
        <span asp-validation-for="Name"></span>
    </div>
    <div>
        <label asp-for="Age">Age</label>
        <input asp-for="Age" value="" />
        <span asp-validation-for="Age"></span>
    </div>
    <div>
        <label asp-for="Sex"></label>
        <input type="radio" asp-for="Sex" value="male" />Male
        <input type="radio" asp-for="Sex" value="female" />female
        <span asp-validation-for="Sex"></span>
    </div>
    <input type="submit" value="Submit" />
</form>
```





Client-side validation

- វាជាផ្នែងផ្ទាត់ភាពត្រូវនៃទិន្នន័សៅផ្នែកខាង browserតែម្តង។
- ដូច្នេះដើម្បីពិនិត្យភាពត្រឹមត្រូវសៅខាងbrowserគេត្រូវការ JavaScript library ដូចខាងក្រោម៖

  1. jquery.js
  2. jquery.validate.js
  3. jquery.validate.unobtrusive.js

ឧទាហរណ៍ៈ

```
4 references
public class Employee
{
    [Required(ErrorMessage ="{0} is needed")]
    4 references
    public string Name { get; set; }
    [Required(ErrorMessage = "{0} is needed")]
    [Range(1,100)]
    4 references
    public int? Age { get; set; }
    [Required(ErrorMessage = "{0} is needed")]
    5 references
    public string Sex { get; set; }
}
```

EmployeeController.cs

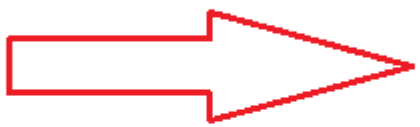
```
public ViewResult Create()
{
    return View();
}
[HttpPost]
0 references
public IActionResult Create(Employee emp)
{
    if (!ModelState.IsValid)
    {
        return View(emp);
    }
    return Content($"Name:{emp.Name},Sex:{emp.Sex},Age:{emp.Age}","text/plain");
}
```

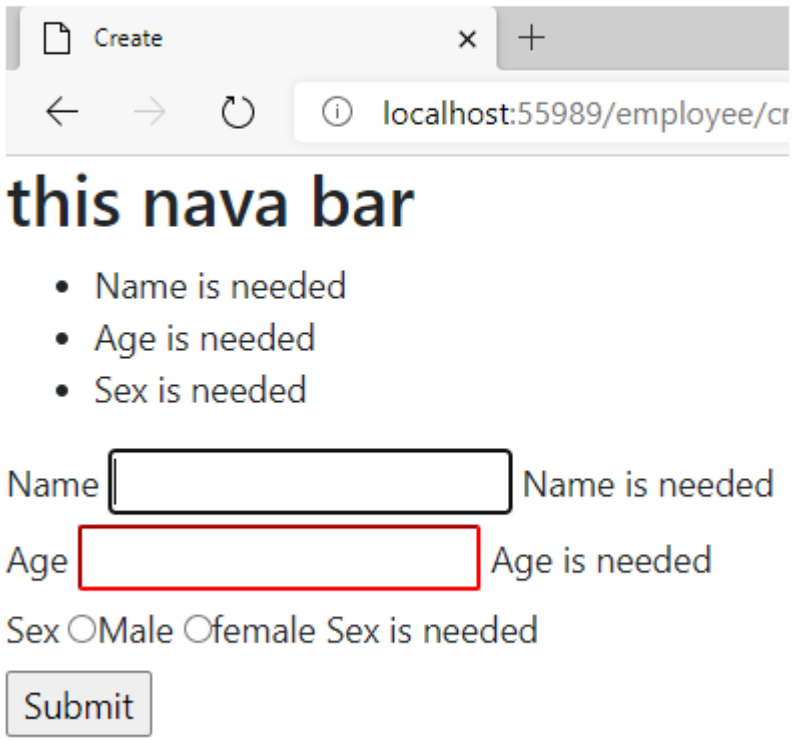
ឧទាហរណ៍

```
<environment include="Development">
    <script src="~/lib/jquery/jquery.js"></script>
    <script src="~/lib/jquery-validate/jquery.validate.js"></script>
    <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>
</environment>
```

ឧទាហរណ៍

```razor
@model Employee
@{
    ViewBag.Title = "Create";
}
<form asp-action="create" method="post" asp-controller="employee">
    <div asp-validation-summary="All"></div>
    <div>
        <label asp-for="Name">Name</label>
        <input type="text" asp-for="Name" />
        <span asp-validation-for="Name"></span>
    </div>
    <div>
        <label asp-for="Age">Age</label>
        <input asp-for="Age" value="" />
        <span asp-validation-for="Age"></span>
    </div>
    <div>
        <label asp-for="Sex"></label>
        <input type="radio" asp-for="Sex" value="male" />Male
        <input type="radio" asp-for="Sex" value="female" />female
        <span asp-validation-for="Sex"></span>
    </div>
    <input type="submit" value="Submit" />
</form>
```





Remote validation

- Remote validation ជាលក្ខណៈពិសេសមួយដែលគេប្រើដើម្បីពិនិត្យភាពត្រឹមត្រូវនៃ request នៅខាង client ជាពត៌មានលើ server។

ឧទាហរណ៍

```
public class Employee
{
    [Required(ErrorMessage ="{0} is needed")]

    [Remote(action: "NameExist", controller:"Employee")]
    4 references
    public string Name { get; set; }
    [Required(ErrorMessage = "{0} is needed")]
    [Range(1,100)]
    4 references
    public int? Age { get; set; }
    [Required(ErrorMessage = "{0} is needed")]
    5 references
    public string Sex { get; set; }
}
```

ឧទាហរណ៍

EmployeeController.cs

```
public JsonResult NameExist(string name)
{
    var names = new List<string>() {"lyty","sokkha","po ey","phally"};

    var n = names.Where(s => s == name.ToLower()).FirstOrDefault();

    if (!string.IsNullOrEmpty(n))
    {
        return Json(data:$"{name} is already exist");
    }
    return Json(data: true);
}
```