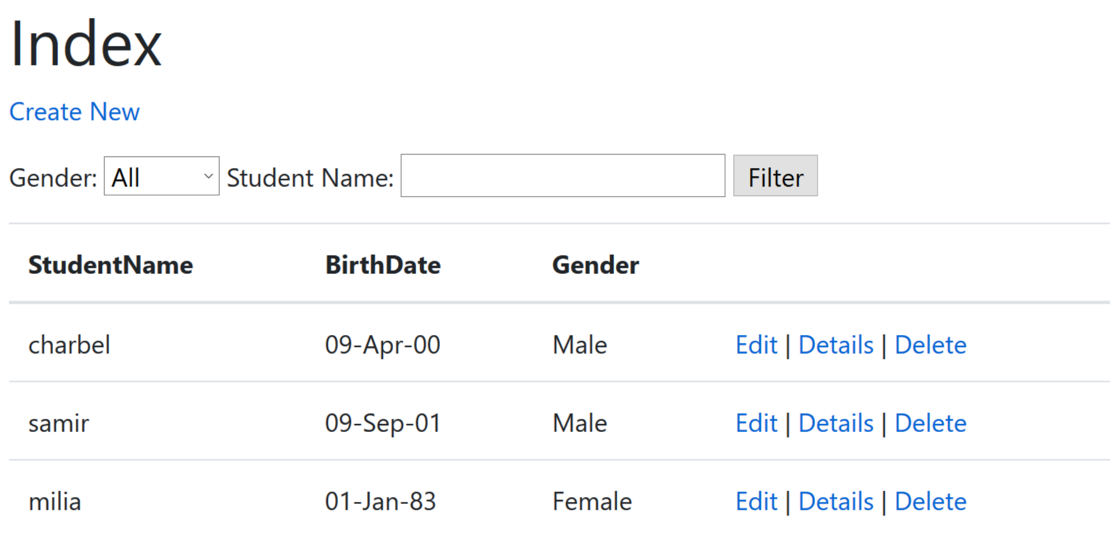
Add Search to your project

The ability to search students by name or gender.



Index

Create New

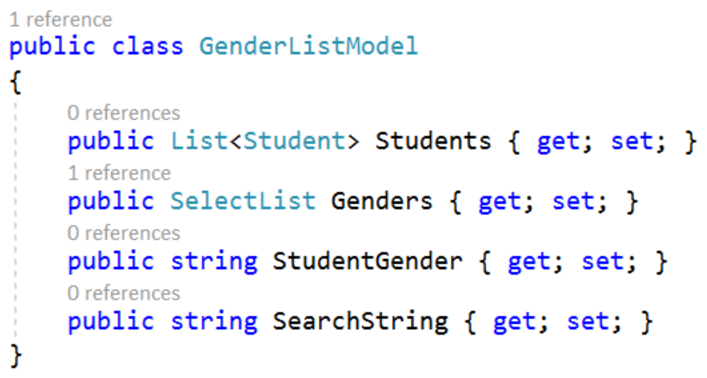
Gender: All Student Name: Filter

StudentName	BirthDate	Gender	
charbel	09-Apr-00	Male	Edit | Details | Delete
samir	09-Sep-01	Male	Edit | Details | Delete
milia	01-Jan-83	Female	Edit | Details | Delete

Steps

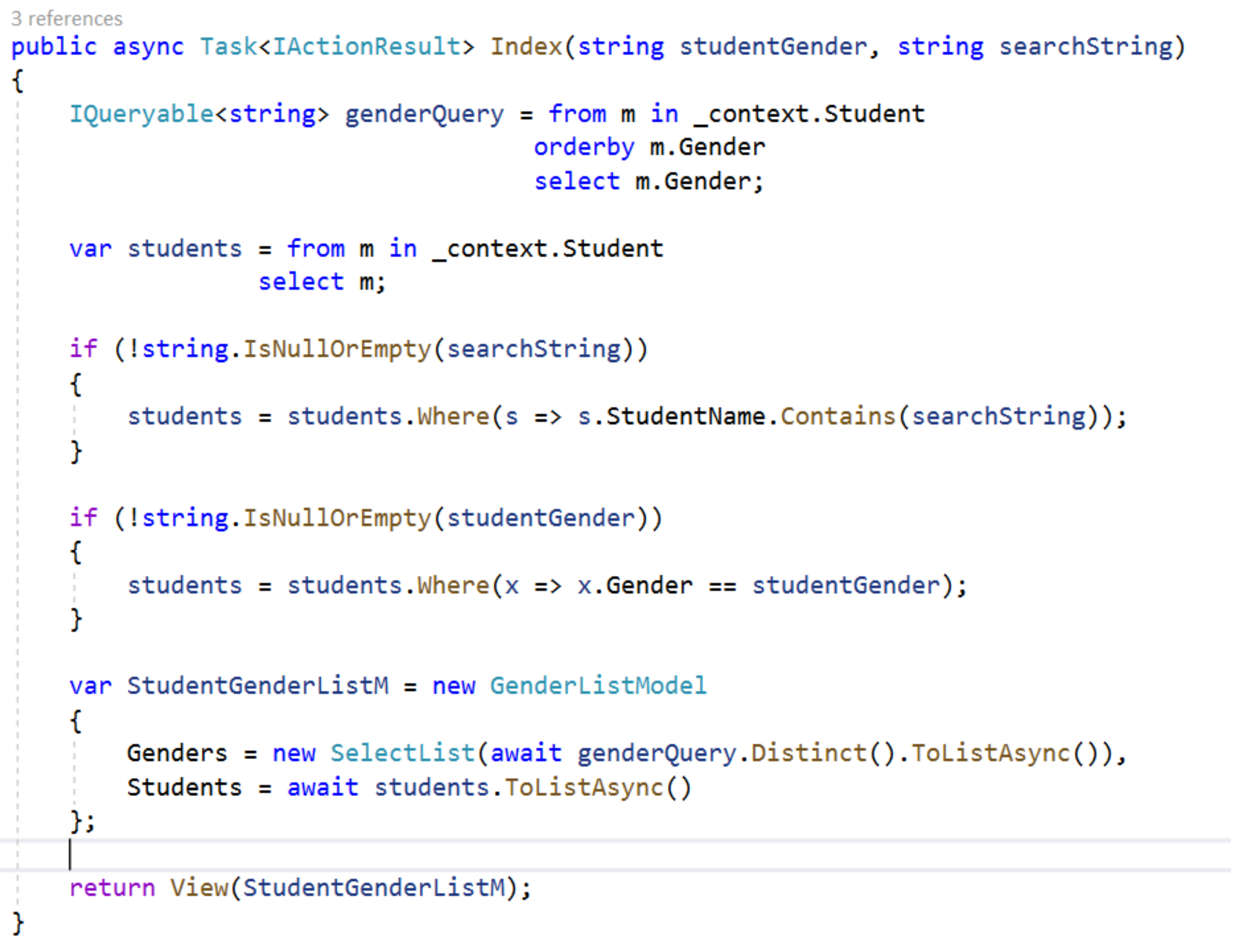
Create a class in the Models folder, this class model will contain:

- A list of Students.
- A `SelectList` containing the list of Genders. This allows the user to select a gender from the list.
- `StudentGender`, which contains the selected gender.
- `SearchString`, which contains the text users enter in the search text box.



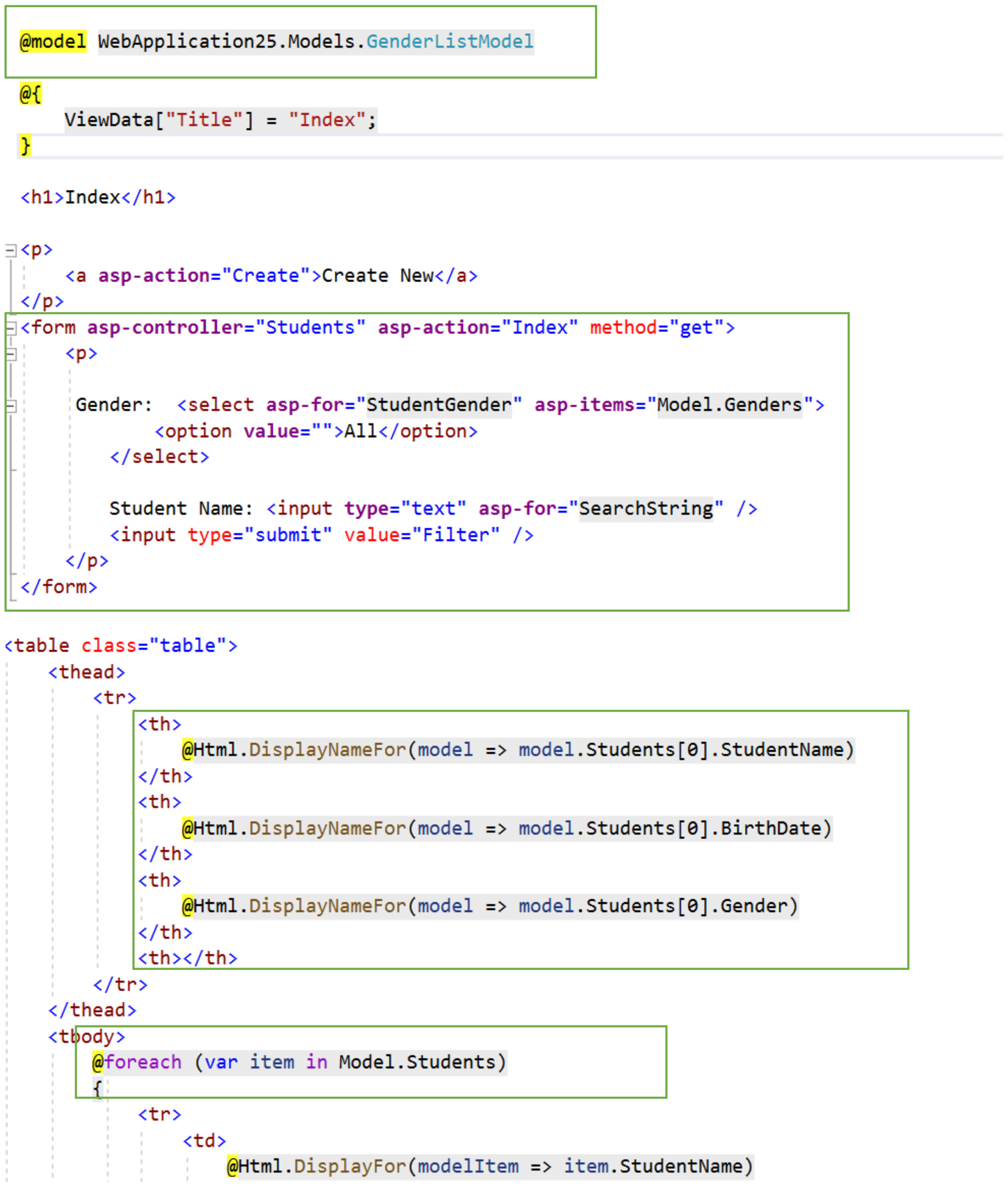
```
1 reference
public class GenderListModel
{
    0 references
    public List<Student> Students { get; set; }
    1 reference
    public SelectList Genders { get; set; }
    0 references
    public string StudentGender { get; set; }
    0 references
    public string SearchString { get; set; }
}
```

Update the `Index` method found inside *Controllers/StudentsController.cs* with the following code:



```
3 references
public async Task<IActionResult> Index(string studentGender, string searchString)
{
    IQueryable<string> genderQuery = from m in _context.Student
                                     orderby m.Gender
                                     select m.Gender;

    var students = from m in _context.Student
                   select m;

    if (!string.IsNullOrEmpty(searchString))
    {
        students = students.Where(s => s.StudentName.Contains(searchString));
    }

    if (!string.IsNullOrEmpty(studentGender))
    {
        students = students.Where(x => x.Gender == studentGender);
    }

    var StudentGenderListM = new GenderListModel
    {
        Genders = new SelectList(await genderQuery.Distinct().ToListAsync()),
        Students = await students.ToListAsync()
    };

    return View(StudentGenderListM);
}
```

Open the *Views/Students/Index.cshtml* file, and modify it



```
@model WebApplication25.Models.GenderListModel
@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<form asp-controller="Students" asp-action="Index" method="get">
    <p>

     Gender:  <select asp-for="StudentGender" asp-items="Model.Genders">
            <option value="">All</option>
        </select>

        Student Name: <input type="text" asp-for="SearchString" />
        <input type="submit" value="Filter" />
    </p>
</form>

<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.Students[0].StudentName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Students[0].BirthDate)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Students[0].Gender)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.Students)
        {
            <tr>
                <td>
                    @Html.DisplayFor(modelItem => item.StudentName)
```

- Run your project now

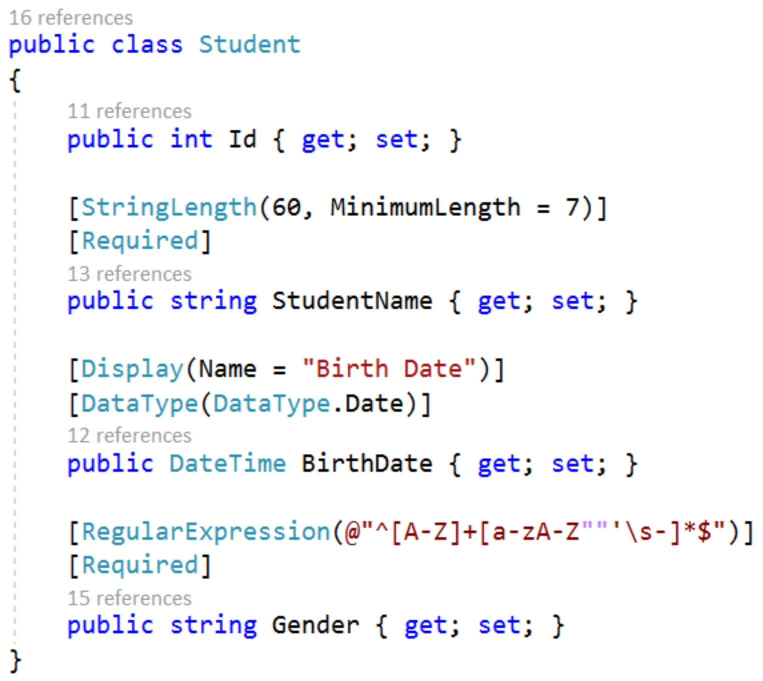
Add validation expression

The DataAnnotations namespace provides a set of built-in validation attributes that are applied declaratively to a class or property. DataAnnotations also contains formatting attributes like `DataType` that help with formatting and don't provide any validation.

The validation attributes specify behavior that you want to enforce on the model properties they're applied to:

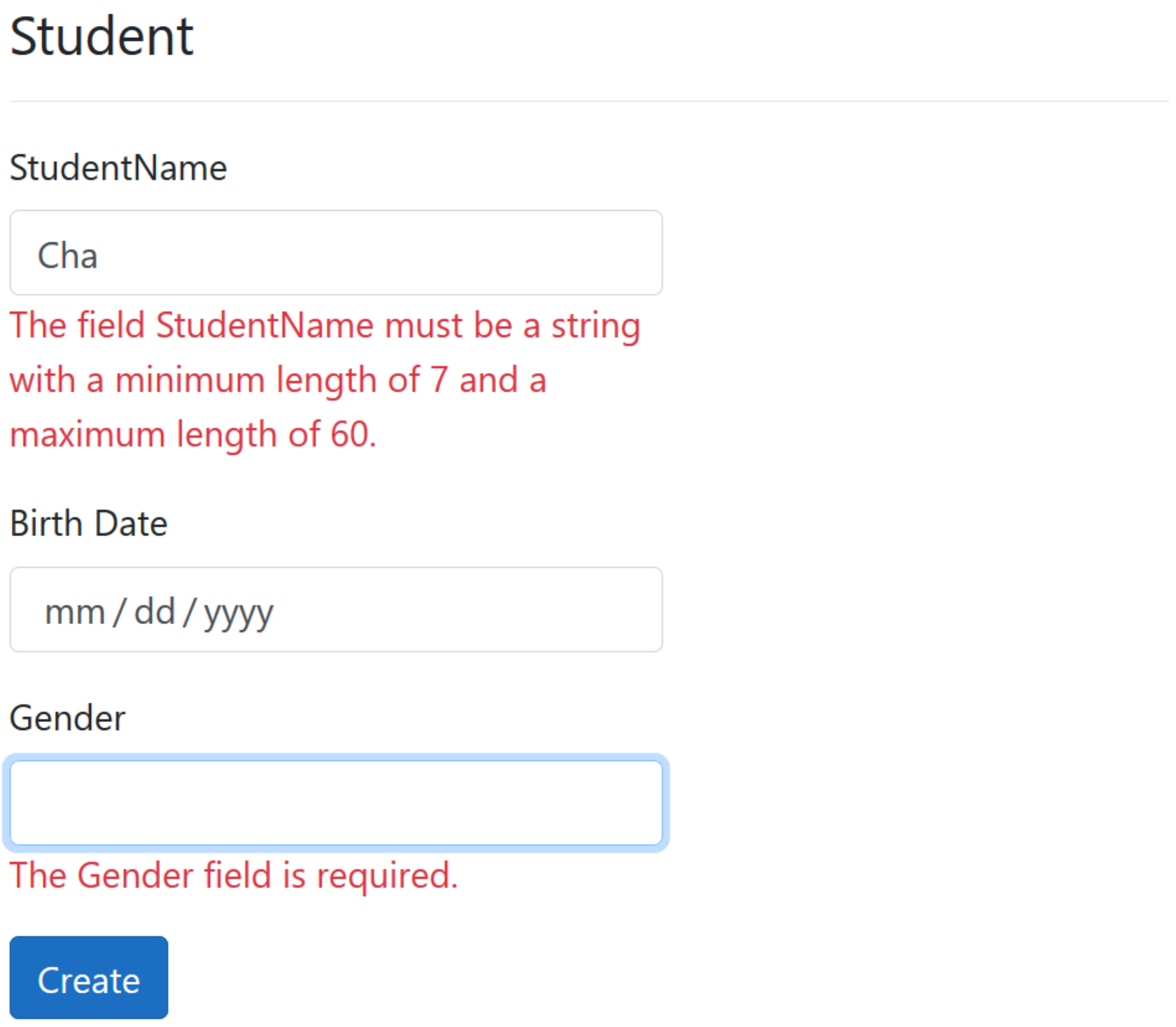
- The `Required` and `MinimumLength` attributes indicate that a property must have a value; but nothing prevents a user from entering white space to satisfy this validation.
- The `RegularExpression` attribute is used to limit what characters can be input. In the preceding code, "Gender":
  - Must only use letters.
  - The first letter is required to be uppercase. White space, numbers, and special characters are not allowed.
- The `Range` attribute constrains a value to within a specified range.
- The `StringLength` attribute lets you set the maximum length of a string property, and optionally its minimum length.
- Value types (such as `decimal`, `int`, `float`, `DateTime`) are inherently required and don't need the `[Required]` attribute.

Modify the student class



```
16 references
public class Student
{
    11 references
    public int Id { get; set; }

    [StringLength(60, MinimumLength = 7)]
    [Required]
    13 references
    public string StudentName { get; set; }

    [Display(Name = "Birth Date")]
    [DataType(DataType.Date)]
    12 references
    public DateTime BirthDate { get; set; }

    [RegularExpression(@"^[A-Z]+[a-zA-Z""'\s-]*$")]
    [Required]
    15 references
    public string Gender { get; set; }
}
```

Run your project and try to add a new student



Student

StudentName

Cha

The field StudentName must be a string with a minimum length of 7 and a maximum length of 60.

Birth Date

mm / dd / yyyy

Gender

The Gender field is required.