Walkthrough 11 - MVC Filtering

Setup

This walkthrough will add filtering by a dropdown list to the Hospital site.

1. Start SQL Server.
2. Open Hospital from the end of the previous walkthrough.

CurrentAdmissionsController.cs

1. Right-click the Controllers folder and select Add / Controller... .
2. Choose MVC Controller with views, using Entity Framework, click Add.
3. Set Model class to **Admissions (Hospital.Models)**.
4. Set Data context class to **CHDBContext (Hospital.Models)**.
5. Set the controller name to **CurrentAdmissionsController**, click Add.

_Layout.cshtml

1. Open Views / Shared / _Layout.cshtml and add a link for the Current Admissions controller.
2.
```
...
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
</li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="CurrentAdmissions" asp-action="Index">Current Admissions</a>
</li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Medications" asp-action="Index">Medications</a>
</li>
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
</li>
...
```
3. Save the file.
4. Run the site. Click the Current Admissions link. Over 5,000 records are returned; this isn't great for performance.

CurrentAdmissionsController.cs

1. Update the Index method to use LINQ to retrieve current admissions only (discharge date of null) and to sort by patient last name, then first name.
2.
```
public async Task<IActionResult> Index()
{
    ~~var cHDBContext = _context.Admissions.Include(a => a.AttendingPhysician).Include(a => a.NursingUnit).Include(a => a.Patient);~~
    var admissions = from a in _context.Admissions
                     .Include(a => a.Patient)
                     .Where(a => a.DischargeDate == null)
                     .OrderBy(a => a.Patient.LastName)
                     .ThenBy(a => a.Patient.FirstName)
                     select a;

    return View(await ~~chdbContext~~admissions.AsNoTracking().ToListAsync());
}
```
3. Save the file, refresh the site.

Index.cshtml

1. Open Views / CurrentAdmissions / Index.cshtml. Update the title and heading.
2.
```
@model IEnumerable<Hospital.Models.Admissions>

@{
    ViewData["Title"] = "~~Index~~Current Admissions";
}

<h1>~~Index~~@ViewBag.Title</h1>

<p>
    <a asp-action="Create">Create New</a>
...
```
3. Update the table headings.
4.
```
...
<table class="table">
    <thead>
        <tr>
            <th>
                ~~@Html.DisplayNameFor(model => model.DischargeDate)~~
                Patient
            </th>
            <th>
                @Html.DisplayNameFor(model => model.PrimaryDiagnosis)
```

```
            </th>
            <th>
                @Html.DisplayNameFor(model => model.SecondaryDiagnoses)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Room)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Bed)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.AttendingPhysician)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.NursingUnit)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Patient)
            </th>
            <th></th>
        </tr>
    </thead>
    ...
```

5. Update the table row.

6.
```
...
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.DischargeDate)
                @Html.DisplayFor(modelItem => item.Patient.LastName),
                @Html.DisplayFor(modelItem => item.Patient.FirstName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.PrimaryDiagnosis)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.SecondaryDiagnoses)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Room)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Bed)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.AttendingPhysician.PhysicianId)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.NursingUnit.NursingUnitId)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Patient.PatientId)
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |
                @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |
                @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
            </td>
        </tr>
}
...
```

7. Save the file, refresh the site.

CurrentAdmissionsController.cs

1. Update the Index method to accept a parameter for nursing unit id, populate a select list of nursing units and to apply a Where clause to the admissions data source if there is a value.

2.
```
public async Task<IActionResult> Index(string nursingUnitId)
{
    ViewBag.NursingUnitId = new SelectList(_context.NursingUnits.ToList(), "NursingUnitId", "NursingUnitId");

    var admissions = from a in _context.Admissions
                     .Include(a => a.Patient)
                     .Where(a => a.DischargeDate == null)
                     .OrderBy(a => a.Patient.LastName)
                     .ThenBy(a => a.Patient.FirstName)
                     select a;

    if (!string.IsNullOrEmpty(nursingUnitId))
    {
        admissions = admissions.Where(a => a.NursingUnitId == nursingUnitId);
    }

    return View(await admissions.AsNoTracking().ToListAsync());
}
```

3. Save the file.

Index.cshtml

1. Add a dropdown list to display the available nursing units and a link to reset. The dropdown list will cause the form to submit when its value changes.

2.
```
...
    <a asp-action="Create">Create New</a>
</p>
```

```
<form asp-action="Index" method="get">
    <p>
        Nursing Unit Id:
        <select asp-for="@Model.FirstOrDefault().NursingUnitId" asp-items="@ViewBag.NursingUnitId" onchange="submit()"></select>
        | <a asp-action="Index">All Current Admissions</a>
    </p>
</form>

<table class="table">
...
```

3. Save the file, refresh the site.
4. As an alternative, remove the reset link and add an "empty" item to the dropdown list, which will cause reset.
5.
```
...
<form asp-action="Index" method="get">
    <p>
        Nursing Unit Id:
        <select asp-for="@Model.FirstOrDefault().NursingUnitId" asp-items="@ViewBag.NursingUnitId" onchange="submit()">
            <option value="">All Current Admissions</option>
        </select>
        | <a asp-action="Index">All Current Admissions</a>
    </p>
</form>
...
```

6. Save the file, refresh the site.

CurrentAdmissionsController.cs

1. Add a data source that retrieves the nursing unit manager name and display that in the dropdown list.
2.
```
public async Task<IActionResult> Index(string nursingUnitId)
{
    var nursingUnits = from n in _context.NursingUnits
                       orderby n.ManagerLastName
                       select new { Name = n.ManagerFirstName + " " + n.ManagerLastName, n.NursingUnitId };

    ViewBag.NursingUnitId = new SelectList(_context.NursingUnits.ToList()nursingUnits, "NursingUnitId", "NursingUnitId""Name");

    var admissions = from a in _context.Admissions
                        .Include(a => a.Patient)
                        .Where(a => a.DischargeDate == null)
                        .OrderBy(a => a.Patient.LastName)
                        .ThenBy(a => a.Patient.FirstName)
                        select a;

    if (!string.IsNullOrEmpty(nursingUnitId))
    {
        admissions = admissions.Where(a => a.NursingUnitId == nursingUnitId);
    }

    return View(await admissions.AsNoTracking().ToListAsync());
}
```

3. Save the file.

Index.cshtml

1. Update the label text.
2.
```
...
<form asp-action="Index" method="get">
    <p>
        Nursing Unit IdManager:
        <select asp-for="@Model.FirstOrDefault().NursingUnitId" asp-items="@ViewBag.NursingUnitId" onchange="submit()">
            <option value="">All Current Admissions</option>
        </select>
    </p>
</form>
...
```

3. Save the file, refresh the site.

CurrentAdmissionsController.cs

1. Update the page title programmatically instead of hard-coding it in the view.
2.
```
public async Task<IActionResult> Index(string nursingUnitId)
{
    var nursingUnits = from n in _context.NursingUnits
                       orderby n.ManagerLastName
                       select new { Name = n.ManagerFirstName + " " + n.ManagerLastName, n.NursingUnitId };

    ViewBag.NursingUnitId = new SelectList(nursingUnits, "NursingUnitId", "Name");

    var admissions = from a in _context.Admissions
                       .Include(a => a.Patient)
                       .Where(a => a.DischargeDate == null)
                       .OrderBy(a => a.Patient.LastName)
                       .ThenBy(a => a.Patient.FirstName)
                       select a;

    ViewBag.Title = "All Current Admissions";

    if (!string.IsNullOrEmpty(nursingUnitId))
    {
        admissions = admissions.Where(a => a.NursingUnitId == nursingUnitId);
        ViewBag.Title = $"Current Admissions - {nursingUnitId}";
    }
```

```
    return View(await admissions.ToListAsync());
}
```

3. Save the file.

Index.cshtml

1. Delete the hard-coded title.
2.
```
@model IEnumerable<Hospital.Models.Admission>

@{
    ViewData["Title"] = "Current Admissions";
}

<h1>@ViewBag.Title</h1>
...
```

3. Save the file, refresh the site.
4. Notice that the dropdown doesn't quite behave correctly when the page is first loaded. This is due to using the FirstOrDefault method which is making Barbara Parsons, 2SOUTH the "current" nursing unit.
5. Remove the select tag helper and instead use the Html DropDownList helper.
6.
```
...
<form asp-action="Index" method="get">
    <p>
        Nursing Unit Manager:
        <select asp-for="@Model.FirstOrDefault().NursingUnitId" asp-items="@ViewBag.NursingUnitId" onchange="submit()">
            <option value="">All Current Admissions</option>
        </select>
        @Html.DropDownList("nursingUnitId", (IEnumerable<SelectListItem>)ViewBag.NursingUnitId, "All Current Admissions", new { onchange =
"submit()" })
    </p>
</form>
...
```

7. Save the file, refresh the site.

Index.cshtml

1. Examine the Edit, Details and Delete link and note the scaffolder left them commented. Uncomment them and set them to PatientId.
2.
```
...
            <td>
                @Html.ActionLink("Edit", "Edit", new { /* id = item.PrimaryKey */PatientId }) |
                @Html.ActionLink("Details", "Details", new { /* id = item.PrimaryKey */PatientId }) |
                @Html.ActionLink("Delete", "Delete", new {/* id = item.PrimaryKey */PatientId })
            </td>
        </tr>
}
    </tbody>
</table>
```

3. Save the file, refresh the site.
4. Set Nursing Unit Manager to All Current Admissions. Try the Details link of the first patient (Aching, Tiffany), it appears to work. Try the Details link of the second patient (Atwater, Mallory); notice the populated DischargeDate, that is not a "current" patient and is incorrect.
5. Click the Back to List link and try the Edit link, it will fail. Examine the error. Notice that it mentions that the Admissions entity needs a composite key.
6. Update the links to use the correct composite key.
7.
```
...
            <td>
                @Html.ActionLink("Edit", "Edit", new { id = item.PatientId, admissionDate = item.AdmissionDate }) |
                @Html.ActionLink("Details", "Details", new { id = item.PatientId, admissionDate = item.AdmissionDate }) |
                @Html.ActionLink("Delete", "Delete", new { id = item.PatientId, admissionDate = item.AdmissionDate })
            </td>
        </tr>
}
    </tbody>
</table>
```

8. Save the file.

CurrentAdmissionsController.cs

1. Update the Details method to use the admissionDate.
2.
```
public async Task<IActionResult> Details(int? id, DateTime? admissionDate)
{
    if (id == null || admissionDate == null)
    {
        return NotFound();
    }

    var admission = await _context.Admissions
        .Include(a => a.AttendingPhysician)
        .Include(a => a.NursingUnit)
        .Include(a => a.Patient)
        .FirstOrDefaultAsync(m => m.PatientId == id && m.AdmissionDate == admissionDate);
    if (admission == null)
```

```
    {
        return NotFound();
    }

    return View(admission);
}
```

3. Save the file.

Details.cshtml

1. Update the Edit link to use the correct composite key.
2.
```
<div>
    @Html.ActionLink("Edit", "Edit", new { id = Model.PatientId, admissionDate = Model.AdmissionDate }) |
    <a asp-action="Index">Back to List</a>
</div>
```

3. Save the file, refresh the site.
4. Try the Details link of Atwater, Mallory, it will work correctly now.

CurrentAdmissionsController.cs

1. Update the Get Edit method to use the admissionDate.
2.
```
public async Task<IActionResult> Edit(int? id, DateTime? admissionDate)
{
    if (id == null || admissionDate == null)
    {
        return NotFound();
    }

    var admission = await _context.Admissions.~~FindAsync(id)~~FirstOrDefaultAsync(a => a.PatientId == id && a.AdmissionDate == admissionDat
e);
    if (admission == null)
    {
        return NotFound();
    }
    ViewData["AttendingPhysicianId"] = new SelectList(_context.Physicians, "PhysicianId", "PhysicianId", admission.AttendingPhysicianId);
    ViewData["NursingUnitId"] = new SelectList(_context.NursingUnits, "NursingUnitId", "NursingUnitId", admission.NursingUnitId);
    ViewData["PatientId"] = new SelectList(_context.Patients, "PatientId", "PatientId", admission.PatientId);
    return View(admission);
}
```

3. Update AdmissionExists and the Post Edit to use admissionDate.
4.
```
private bool AdmissionExists(int id, DateTime admissionDate)
{
    return _context.Admissions.Any(e => e.PatientId == id && e.AdmissionDate == admissionDate);
}
```

5.
```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("PatientId,AdmissionDate,DischargeDate,PrimaryDiagnosis,SecondaryDiagnoses,AttendingPh
ysicianId,NursingUnitId,Room,Bed")] Admission admission)
{
    if (id != admission.PatientId)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(admission);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!AdmissionExists(admission.PatientId, admission.AdmissionDate))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["AttendingPhysicianId"] = new SelectList(_context.Physicians, "PhysicianId", "PhysicianId", admission.AttendingPhysicianId);
    ViewData["NursingUnitId"] = new SelectList(_context.NursingUnits, "NursingUnitId", "NursingUnitId", admission.NursingUnitId);
    ViewData["PatientId"] = new SelectList(_context.Patients, "PatientId", "PatientId", admission.PatientId);
    return View(admission);
}
```

6. Save the file, refresh the site.
7. Return to the list of all current patients. Note the first patient (Aching, Tiffany). Edit that patient's admission record and set the DischargeDate to today.
8. When returned to the list of all current patients, note that the patient is no longer present.
9. Update the Delete and DeleteConfirmed methods to use the admissionDate.
10.
```
public async Task<IActionResult> Delete(int? id, DateTime? admissionDate)
{
    if (id == null || admissionDate == null)
    {
```

```
        return NotFound();
    }

    var admission = await _context.Admissions
        .Include(a => a.AttendingPhysician)
        .Include(a => a.NursingUnit)
        .Include(a => a.Patient)
        .FirstOrDefaultAsync(m => m.PatientId == id && m.AdmissionDate == admissionDate);
    if (admission == null)
    {
        return NotFound();
    }

    return View(admission);
}
```

11.
```
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteConfirmed(int id, DateTime admissionDate)
{
    var admission = await _context.Admissions.~~FindAsync(id)~~FirstOrDefaultAsync(a => a.PatientId == id && a.AdmissionDate == admissionDat
e);
    _context.Admissions.Remove(admission);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

12. Save the file, refresh the site.
13. Delete an admission record.

Details.cshtml

1. Improve the Details view by enhancing the heading and removing unnecessary fields.
2.
```
...

<h1>~~Details~~Current Admission - @Model.Patient.FirstName @Model.Patient.LastName</h1>

<div>
    ~~<h4>Admission</h4>~~
    <hr />
    <dl class="row">
        ~~<dt class = "col-sm-2">~~
            ~~@Html.DisplayNameFor(model => model.DischargeDate)~~
        ~~</dt>~~
        ~~<dd class = "col-sm-10">~~
            ~~@Html.DisplayFor(model => model.DischargeDate)~~
        ~~</dd>~~
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.PrimaryDiagnosis)
        </dt>
        ...
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.NursingUnit.NursingUnitId)
        </dd>
        ~~<dt class = "col-sm-2">~~
            ~~@Html.DisplayNameFor(model => model.Patient)~~
        ~~</dt>~~
        ~~<dd class = "col-sm-10">~~
            ~~@Html.DisplayFor(model => model.Patient.PatientId)~~
        ~~</dd>~~
    </dl>
</div>
...
```

3. Save the file, view the details of a current admission.