

CO550 – Web Applications

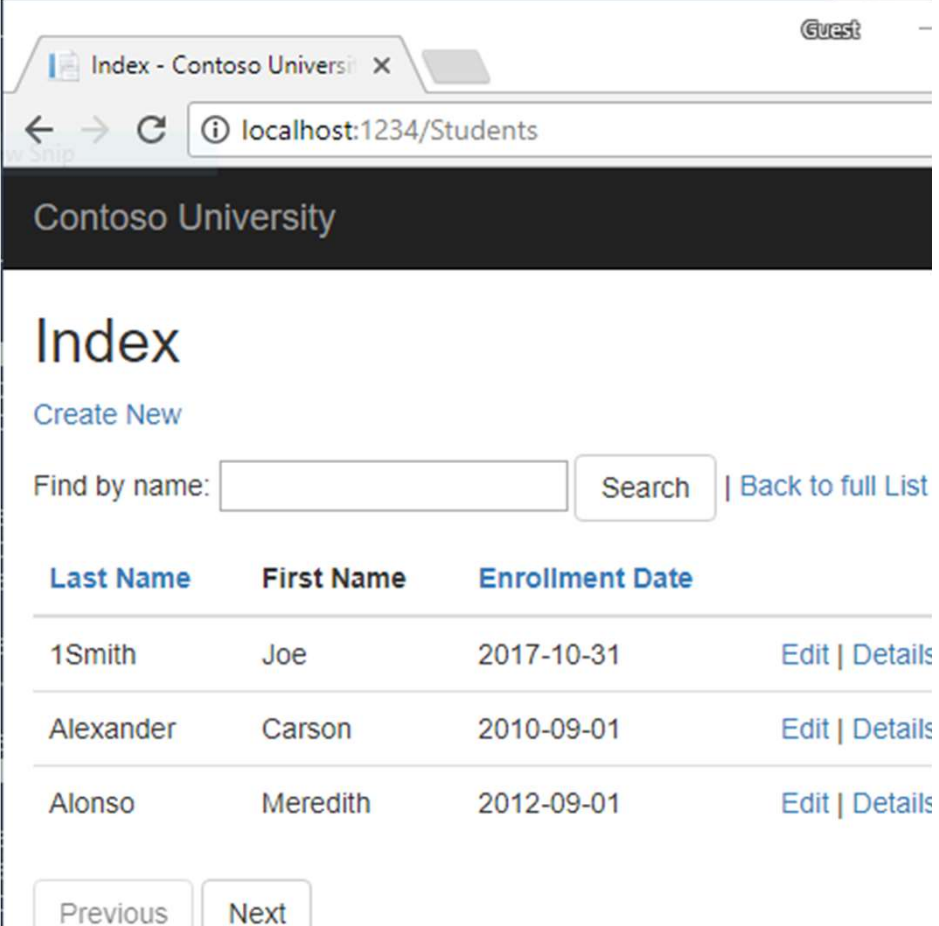
UNIT 7 – CRUD, Asynchronous programming, Customising Views, Overposting & Security

Recap

Step 1 of our ASP.NET Core Razor Pages Tutorial covered...

<https://docs.microsoft.com/en-us/aspnet/core/data/ef-rp/intro?view=aspnetcore-2.1&tabs=visual-studio>

- Manipulating template files
- Setting up a data model
- Scaffolding the data model
- Create the database with `EnsureCreated`
- Initialising the database with seed data



The screenshot shows a web browser window with the URL `localhost:1234/Students`. The page title is "Index - Contoso University". The page content includes a search bar with the text "Find by name:" and a "Search" button. Below the search bar is a table with the following data:

| Last Name | First Name | Enrollment Date | |
|-----------|------------|-----------------|----------------|
| 1Smith | Joe | 2017-10-31 | Edit Details |
| Alexander | Carson | 2010-09-01 | Edit Details |
| Alonso | Meredith | 2012-09-01 | Edit Details |

At the bottom of the page, there are "Previous" and "Next" navigation buttons.

Step 2 Covers...

<https://docs.microsoft.com/en-us/aspnet/core/data/ef-rp/crud?view=aspnetcore-2.1>

- Understanding the the CRUD pages created
- Asynchronous calls
- Using the "{id:int}" route template
- Adding related data to a Student Details page (get enrolled courses of a student)
- Updating the Create page
- Overposting and basic security

Handler Methods

- Razor Page Handlers or Handler Methods are a way of connecting user requests to our methods. Requests come from the **.cshtml** file.
- Razor Pages are following particular naming convention. As you could notice from the last post that there are quite a few Handler Methods that .NET Core tooling generates for us, some of them are:
 - OnGet
 - OnPost
 - OnGetAsync
 - OnPostAsync
 - OnPostRemoveLoginAsync
 - OnGetLinkLoginCallbackAsync

Source: <https://codingblast.com/asp-net-core-razor-pages-handlers/>

Understanding the CRUD pages

The scaffolded code uses the following pattern for Create, Edit, and Delete pages:

- Get and display the requested data with the HTTP GET method **OnGetAsync**.
- Save changes to the data with the HTTP POST method **OnPostAsync**.

As noted on previous slide, these are called “Handler Methods”

The Index and Details pages get and display the requested data with the HTTP GET method **OnGetAsync**

Understanding Asynchronous Calls

Asynchronous programming is writing code that allows several things to happen at the same time without "blocking", or waiting for other things to complete.

This is different from **synchronous** programming, in which everything happens in the order it is written.

<https://exceptionnotfound.net/using-async-and-await-in-asp-net-what-do-these-keywords-mean/>

Understanding Asynchronous Calls

- A web server has a limited number of threads available
- In high load situations all of the available threads might be in use (and server can't process new requests until the threads are freed up)
- **Synchronous** code: many threads may be tied up while they aren't actually doing any work because they're waiting for I/O to complete.
- **Asynchronous** code: when a process is waiting for I/O to complete, its thread is freed up.
- Asynchronous code enables server resources to be used more efficiently = ability to handle more traffic without delays.

Understanding Asynchronous Calls

In the following code, the `async` keyword, `Task<T>` return value, `await` keyword, and `ToListAsync` method make the code execute asynchronously.

C#

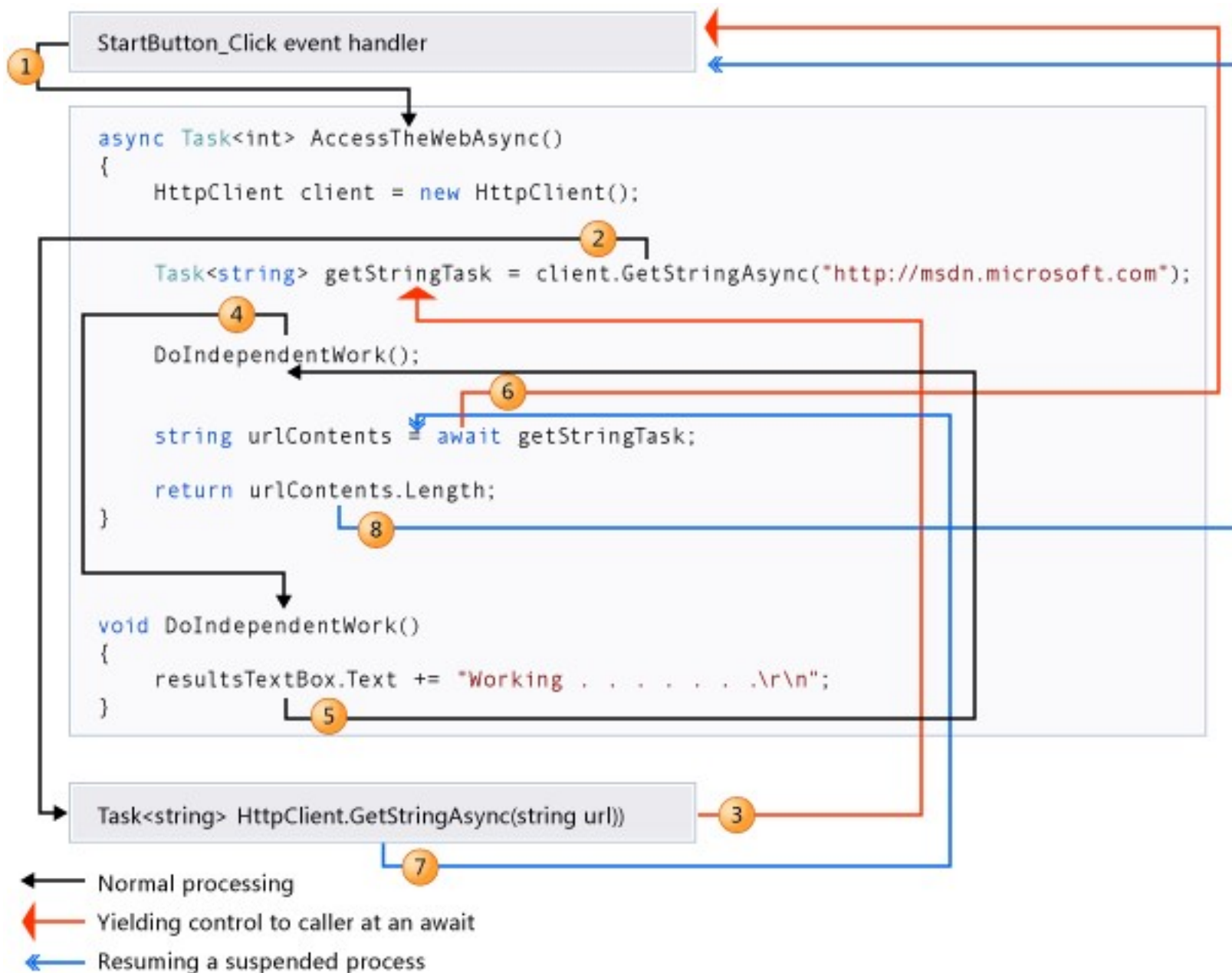
```
public async Task OnGetAsync()  
{  
    Student = await _context.Student.ToListAsync();  
}
```


Understanding Asynchronous Calls

Only statements that cause queries or commands to be sent to the DB are executed asynchronously.

That includes:

- `ToListAsync`
- `SingleOrDefaultAsync`
- `FirstOrDefaultAsync`
- `SaveChangesAsync`



Understanding Asynchronous Calls

Further reading and watching:

- <https://exceptionnotfound.net/using-async-and-await-in-asp-net-what-do-these-keywords-mean/>
- <https://docs.microsoft.com/en-us/dotnet/csharp/async>
- Video: <https://channel9.msdn.com/Series/Three-Essential-Tips-for-Async>
- <http://www.entityframeworktutorial.net/entityframework6/async-query-and-save.aspx>

Page Directive Routing

When viewing our web application:

- select a “Details” link for a Student
- The URL is of the form <http://localhost:5000/Students/Details?id=2>.
- The Student ID is passed using a query string (?id=2).
- A request to the page with the "{id:int}" route template that does not include a integer route value returns an HTTP 404 (not found) error
- For example, <http://localhost:5000/Students/Details> returns a 404 error.
- To make the ID optional, append ? to the route constraint:
`@page "{id:int?}"`

Further reading:

<https://www.learnrazorpages.com/razor-pages/routing>

Customising the Details Page

The **Include** and **ThenInclude** methods cause the context to load the **Student.Enrollments** navigation property, and within each enrollment the **Enrollment.Course** navigation property.

C#

```
public async Task<IActionResult> OnGetAsync(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    Student = await _context.Student
        .Include(s => s.Enrollments)
        .ThenInclude(e => e.Course)
        .AsNoTracking()
        .FirstOrDefaultAsync(m => m.ID == id);

    if (Student == null)
    {
        return NotFound();
    }
    return Page();
}
```

Customising the Details Page

Then we need to update Pages/Students/Details.cshtml

```
    </dd>
    <dt>
        @Html.DisplayNameFor(model => model.Student.Enrollments)
    </dt>
    <dd>
        <table class="table">
            <tr>
                <th>Course Title</th>
                <th>Grade</th>
            </tr>
            @foreach (var item in Model.Student.Enrollments)
            {
                <tr>
                    <td>
                        @Html.DisplayFor(modelItem => item.Course.Title)
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.Grade)
                    </td>
                </tr>
            }
        </table>
    </dd>
</dl>
```

Overposting and Security

Source: <https://andrewlock.net/preventing-mass-assignment-or-over-posting-in-asp-net-core/>

Explained through an example...

Let's take this example model of a user:

```
public class UserModel
{
    public string Name { get; set; }
    public bool IsAdmin { get; set; }
}
```

We only want users to be able to edit “Name”

Overposting and Security

Here's the razor page form:

```
@model UserModel

<form asp-action="Vulnerable" asp-controller="Home">
  <div class="form-group">
    <label asp-for="Name"></label>
    <input class="form-control" type="TextBox" asp-for="Name" />
  </div>
  <div class="form-group">
    @if (Model.IsAdmin)
    {
      <i>You are an admin</i>
    }
    else
    {
      <i>You are a standard user</i>
    }
  </div>
  <button class="btn btn-sm" type="submit">Submit</button>
</form>
```


Overposting and Security

And here's a vulnerable method:

```
[HttpPost]
public IActionResult Vulnerable(UserModel model)
{
    return View("Index", model);
}
```

We'll use
the posted
"model"

here for

something... like... saving the updated data to the database


Why is this vulnerable?

Overposting and Security

Here's why:

with a simple bit of HTML manipulation, or by using Postman/Fiddler, a malicious user can set the IsAdmin field to true.

```
[HttpPost]
public IActionResult Vulnerable(UserModel model)
{
    return View("Index", model);
}
```



The screenshot shows a debugger window with a table of model properties:

| Property | Value |
|----------|--------|
| IsAdmin | true |
| Name | "Sock" |

Overposting – One Solution

- Using **TryUpdateModelAsync** or similar methods
- Only updates the attributes specified

C#

```
var emptyStudent = new Student();

if (await TryUpdateModelAsync<Student>(
    emptyStudent,
    "student", // Prefix for form value.
    s => s.FirstMidName, s => s.LastName, s => s.EnrollmentDate))
{
```

Overposting – Another Solution (ViewModel)

- Using a “ViewModel”
- A view model typically contains a subset of the properties included in the model used by the application. (The **application model** is often called the **domain model**.)
- The view model contains only the properties needed for the UI layer (for example, the Create page)

C#

```
using System;

namespace ContosoUniversity.Models
{
    public class StudentVM
    {
        public int ID { get; set; }
        public string LastName { get; set; }
        public string FirstMidName { get; set; }
        public DateTime EnrollmentDate { get; set; }
    }
}
```

Overposting – ViewModel

The following code uses the `StudentVM` view model to create a new student:

C#

```
[BindProperty]
public StudentVM StudentVM { get; set; }

public async Task<IActionResult> OnPostAsync()
{
    if (!ModelState.IsValid)
    {
        return Page();
    }

    var entry = _context.Add(new Student());
    entry.CurrentValues.SetValues(StudentVM);
    await _context.SaveChangesAsync();
    return RedirectToPage("./Index");
}
```

Bootstrap

Bootstrap Video Tutorials

We are not focusing on front-end development in this module, but ASP.NET projects come with Bootstrap baked in so it is useful for you to know a bit about it...

Recommendations for your self-directed learning:

- <https://app.pluralsight.com/player?author=scott-allen&name=aspdotnet-mvc5-fundamentals-m4-bootstrap&mode=live&clip=0&course=aspdotnet-mvc5-fundamentals>
- https://mva.microsoft.com/en-US/training-courses/introduction-to-asp-net-mvc-8322?l=rdZAhAay_7004984382

Bootstrap grid examples

Basic grid layouts to get you familiar with building within the Bootstrap grid system.

Three equal columns

Get three equal-width columns **starting at desktops and scaling to large desktops**. On mobile devices, tablets and below, the columns will automatically stack.

| | | |
|-----------|-----------|-----------|
| .col-md-4 | .col-md-4 | .col-md-4 |
|-----------|-----------|-----------|

Three unequal columns

Get three columns **starting at desktops and scaling to large desktops** of various widths. Remember, grid columns should add up to twelve for a single horizontal block. More than that, and columns start stacking no matter the viewport.

| | | |
|-----------|-----------|-----------|
| .col-md-3 | .col-md-6 | .col-md-3 |
|-----------|-----------|-----------|

Two columns

Get two columns **starting at desktops and scaling to large desktops**.

| | |
|-----------|-----------|
| .col-md-8 | .col-md-4 |
|-----------|-----------|

Full width, single column

No grid classes are necessary for full-width elements.

Two columns with two nested columns

Per the documentation, nesting is easy—just put a row of columns within an existing row. This gives you two columns **starting at desktops and scaling to large desktops**, with another two (equal widths) within the larger column.

At mobile device sizes, tablets and down, these columns and their nested columns will stack.

| | |
|-----------|-----------|
| .col-md-8 | .col-md-4 |
| .col-md-6 | .col-md-6 |

Some example layouts...

SIDEBAR

[Link](#)[Link](#)[Link](#)[Link](#)

SIDEBAR

[Link](#)[Link](#)[Link](#)[Link](#)[Link](#)[Link](#)

SIDEBAR

[Link](#)[Link](#)[Link](#)

Hello, world!

This is a template for a simple marketing or informational website. It includes a large callout called the hero unit and three supporting pieces of content. Use it as a starting point to create something more unique.

[Learn more »](#)

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio dui.

[View details »](#)

Heading

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus. Etiam porta sem malesuada magna mollis euismod. Donec sed odio

Login to site.com



Username

example@gmail.com

Password

Remember login

(if this is a private computer)

Login

Help to login

Register now for **FREE**

- ✓ See all your orders
- ✓ Fast re-order
- ✓ Save your favorites
- ✓ Fast checkout
- ✓ Get a gift (only new customers)

[Read more](#)


Yes please, register now!


Digging Deeper into the theory


VIDEOS For Discussion


https://www.youtube.com/playlist?list=PLDmvslp_VR0x2CmC6c4AZhZfYX7G2nBlo


First 5 videos covering: Environment setup, Application structure, Lifecycle of an app, Middleware

- 

1 Introduction to ASP.NET Core 2 | Environment Setup | Part 1 | Eduonix
Eduonix Learning Solutions
- 

2 Introduction to ASP.NET Core 2 | Overview | Part 2 | Eduonix
Eduonix Learning Solutions
- 

3 Introduction to ASP.NET Core 2 | Application Structure | Part 3 | Eduonix
Eduonix Learning Solutions
- 

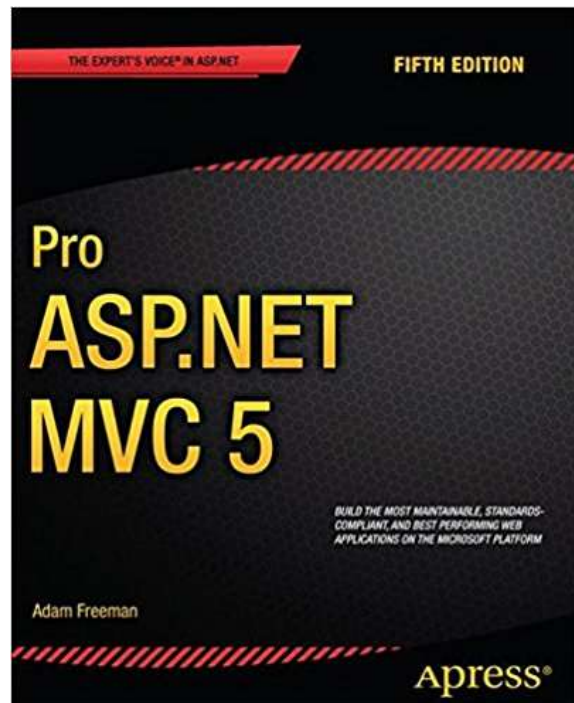
4 Introduction to ASP.NET Core 2 | Life Cycle Of An App | Part 4 | Eduonix
Eduonix Learning Solutions
- 

5 ASP.NET Core 2 Tutorial | Middleware | Part 5 | Eduonix
Eduonix Learning Solutions

Further Reading

Pro ASP.NET MVC 5 (Fifth Edition)

Available online (and in print) via BNU Library



Chapter 1: Putting ASP.NET MVC in Context (p. 1-10)

Chapter 3: The MVC Pattern (p. 51-66)

- The history of MVC
- Understanding the MVC pattern
- Loose coupling
- Automated testing

NOW: TUTORIAL Workshop

ASP.NET Core Razor Pages tutorial – work through the steps

<https://docs.microsoft.com/en-us/aspnet/core/data/ef-rp/sort-filter-page?view=aspnetcore-2.1>

Logbooks 4 and 5 now supplied