## Creating the Products List Component

In the Pages folder, we are going to create two files. First, we're going to create Products.razor… which will be our template, and then we'll create a partial class in the Products.razor.cs file... Let's paste in some HTML in the Products.razor file first… There is nothing special about this HTML, it's just a skeleton and some placeholders. You can see that we've reserved places for searching and sorting, create product link, products list, and pagination… In this video, we're going to focus on the products list component, and we'll add other features later on in the course. Now let's add some logic to our Products class. Let's create a property of the type List<Product> and call it ProductList. We're going to initialize it too… Then, we're going to inject the ProductHttpRepository we've implemented earlier… Don't forget to add the [Inject] Attribute… And then, we're going to override the OnInitializedAsync method in order to retrieve the products when the component is initialized. To do that, we'll populate the Product list by calling the async ProductRepo.GetProducts() method. Once we get the result, we're just going to create a foreach loop… and list the products from the ProductList with Console.Writeline... Now, let's start both of our applications... and press F12, navigate to the Products page, and inspect the logs... Excellent. Our data is here. Now let's create a proper component for our Data. Let's create a new component template ProductTable.razor in the Components folder... and the ProductTable.razor.cs file too. Now let's add a parameter of the List<Product> type to the ProductTable class... and name it Products… This parameter will accept the products from the parent component... All we have to do now is to create a beautiful table that shows our products… First, let's check if there are any products on our list with @if(Products.Any())... If there is at least one product, we're going to show the table… in the header of that table we're going to add… an empty field that will contain the image… name field... supplier… price… update… delete… And in the body of the table we're going to loop through our products… and create a new row for each entry in the Products list… First one for the image… And then one for the name… And we can copy that one for the supplier… and price… And then we're going to add the update button… with the class btn btn-info, and then the delete button… with the class btn btn-danger… Now that's all for the table body… and then in the case that no products were loaded… we're going to write "Loading Products…". What we're doing here is conditional rendering. In case we got our data, we're going to display it in the table, and if we didn't, we don't need to render the entire table, but just the message. Finally, to connect these components we need to add this component

to the Products component… we're going to import it first at the top… and then replace the placeholder with our ProductTable component… and add one paramete Products… and Populate it with the ProductList... Great! Let's see what we've done so far! If we run the application and navigate to the Products… "we'll see Loading Products…" message briefly, and then our products will populate the table… As you can see, we've retrieved all the products from our database, and while that's not a lot of products, you can imagine how this can turn ugly if there were a few hundred, or a few thousand of them. Great, now this is starting to look like a real application. But we're far from over. Ideally, we want to have a details page for each item in the list.