An Alternative to HTML Tables

I have long had a problem with using HTML tables to display data to the user. I have an even bigger problem with editing on a table, but that is a different discussion. An HTML table is easy to implement for a developer, and this is normally why developers use them. However, a table is not always the best method for conveying data to a user, especially when that data is most likely viewed on a mobile device. Of course, there are always exceptions to this rule, but these should be “the exception” and not the rule. There are many reasons why a table is not suitable for user consumption.

- A table presents too much data on the screen so the user’s eye has too much to try to concentrate on.
- A user will get much more tired at the end of the day using a table as opposed to other display types.
- The user cannot distinguish between the data in each column since each column is very uniform and nothing stands out.

The above are just some of the reasons why a table might not be the appropriate choice for displaying a list of data to the user. Look at Figures 1 and 3 for an example of a normal HTML table displayed in both a normal browser and on a mobile device. Then look at Figures 2 and 4 which should the alternative display of the same data on a desktop browser and a mobile device. As you can see the alternative design displays the data in a much easier-to-read format.

Using a HTML Table

A list of product data in a normal HTML table renders fine on a desktop browser, as illustrated in Figure 1. However, if you display that same product data on a mobile browser, you might only see the left most columns as illustrated in Figure 2. On some mobile browsers, they may render the table so small that it is hard to read. Either way, the user is forced to interact with their phone to view the data. They will either have to scroll left to right, or maybe pinch or spread with their fingers.

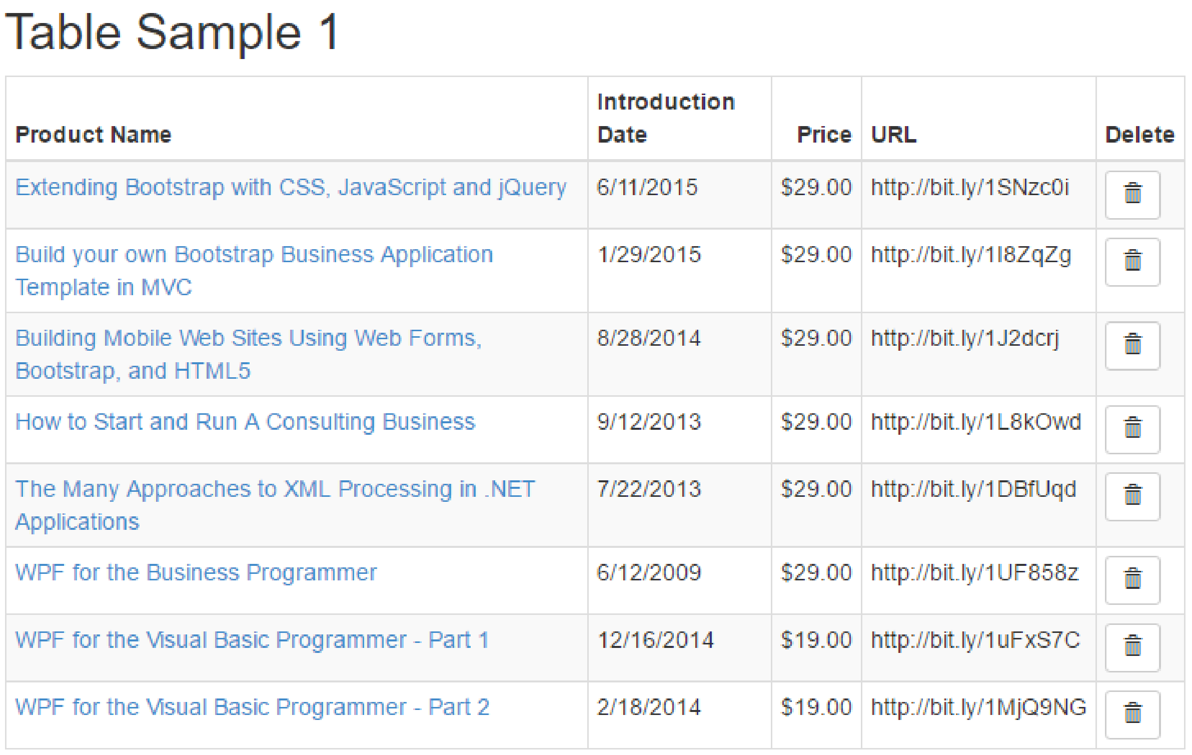
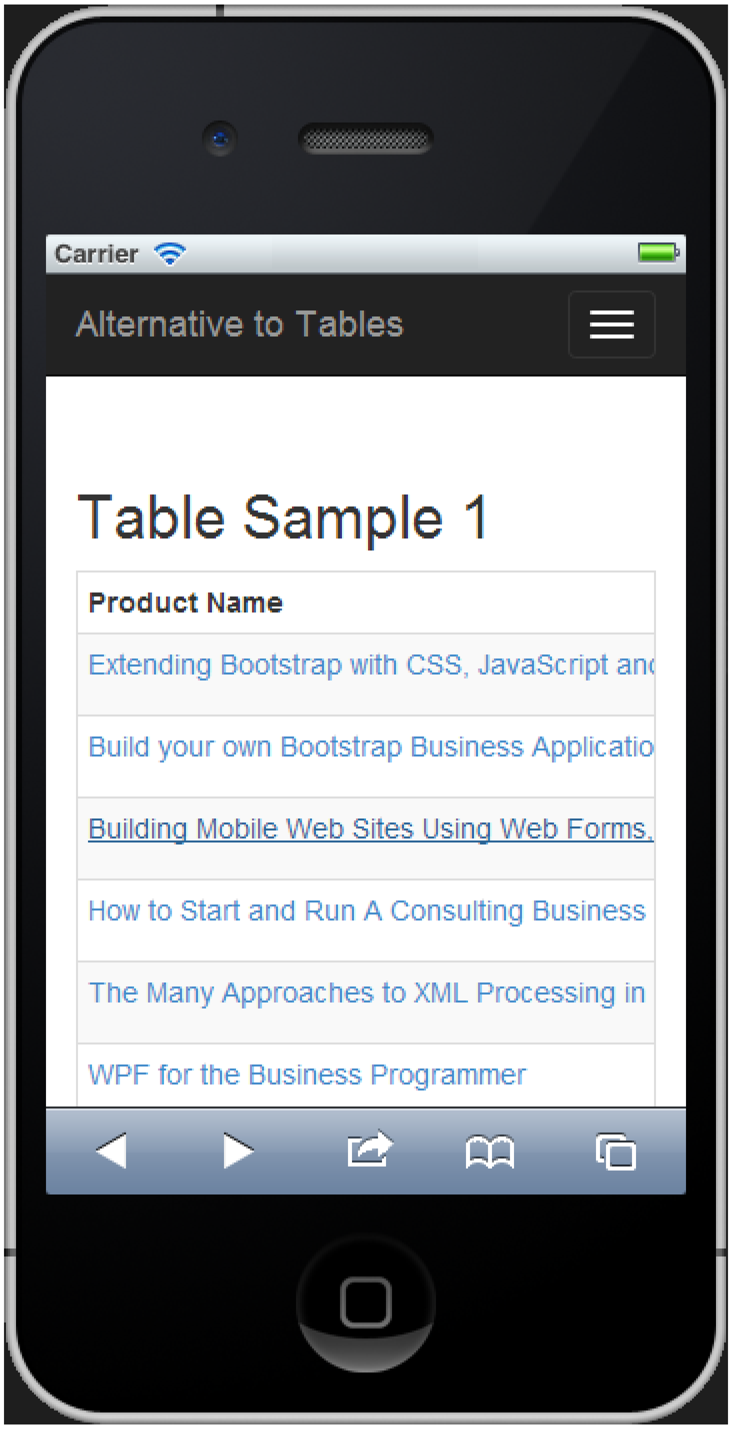


Table Sample 1

Product Name	Introduction Date	Price	URL	Delete
Extending Bootstrap with CSS, JavaScript and jQuery	6/11/2015	$29.00	http://bit.ly/1SNzc0i	🗑
Build your own Bootstrap Business Application Template in MVC	1/29/2015	$29.00	http://bit.ly/1I8ZqZg	🗑
Building Mobile Web Sites Using Web Forms, Bootstrap, and HTML5	8/28/2014	$29.00	http://bit.ly/1J2dcrj	🗑
How to Start and Run A Consulting Business	9/12/2013	$29.00	http://bit.ly/1L8kOwd	🗑
The Many Approaches to XML Processing in .NET Applications	7/22/2013	$29.00	http://bit.ly/1DBfUqd	🗑
WPF for the Business Programmer	6/12/2009	$29.00	http://bit.ly/1UF858z	🗑
WPF for the Visual Basic Programmer - Part 1	12/16/2014	$19.00	http://bit.ly/1uFxS7C	🗑
WPF for the Visual Basic Programmer - Part 2	2/18/2014	$19.00	http://bit.ly/1MjQ9NG	🗑

Figure 1: An HTML table rendered on a desktop browser.



Alternative to Tables

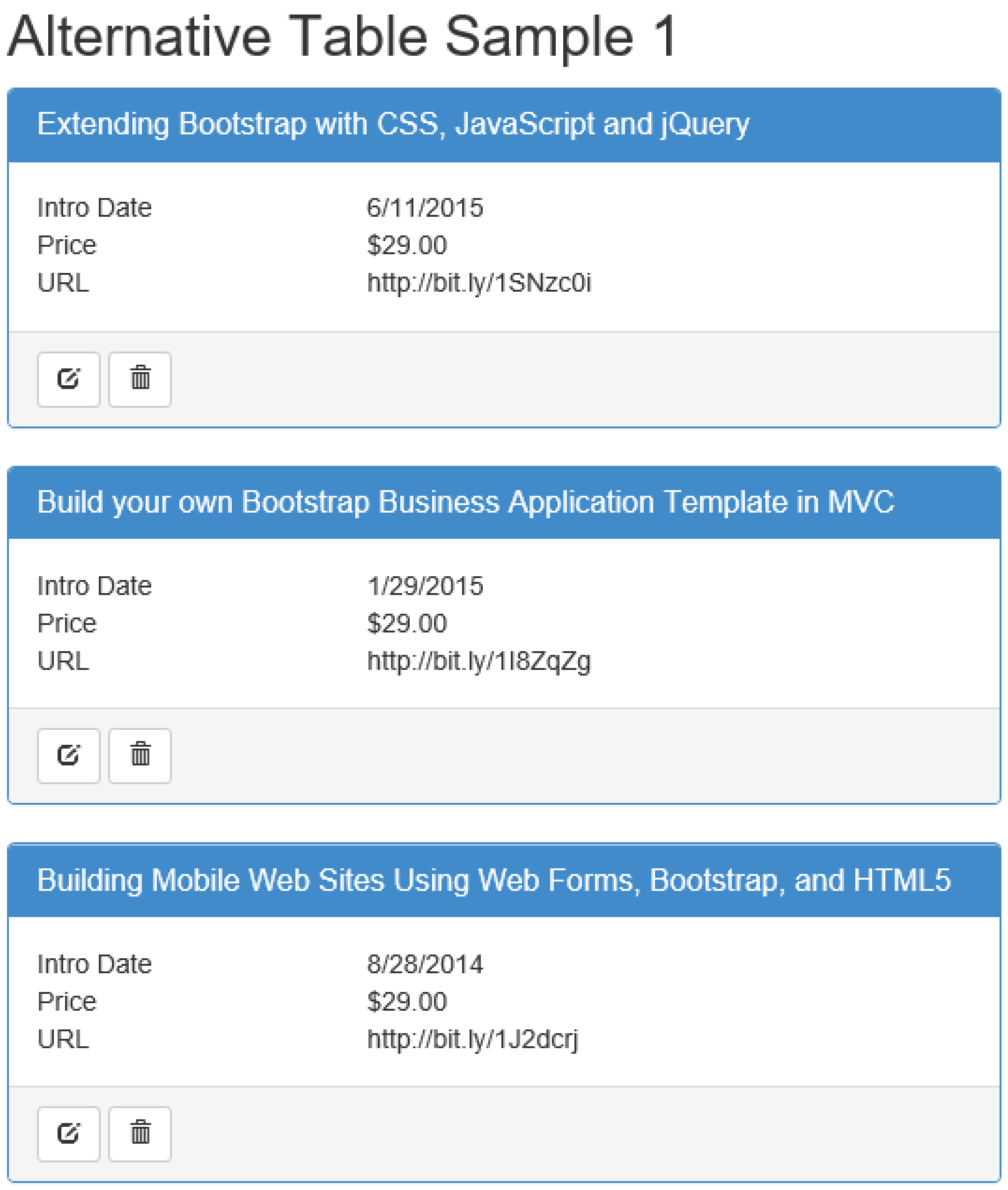
Table Sample 1

Product Name
Extending Bootstrap with CSS, JavaScript anc
Build your own Bootstrap Business Applicatio
Building Mobile Web Sites Using Web Forms,
How to Start and Run A Consulting Business
The Many Approaches to XML Processing in
WPF for the Business Programmer

Figure 2: An HTML table rendered on a mobile browser.

Using a Bootstrap Panel

Instead of using an HTML table, you might display a list of Bootstrap panel controls that contain your product data. In the sample shown in Figure 3 you see the same data, but the most important data, the product name, is displayed in the panel header area. The other information about the product is displayed within the body of the panel. The actions you can take are displayed within the panel footer area. As you can see this list of data looks just as good on a normal desktop browser (Figure 3) as on a mobile browser (Figure 4).



Alternative Table Sample 1

Extending Bootstrap with CSS, JavaScript and jQuery

Intro Date	6/11/2015
Price	$29.00
URL	http://bit.ly/1SNzc0i

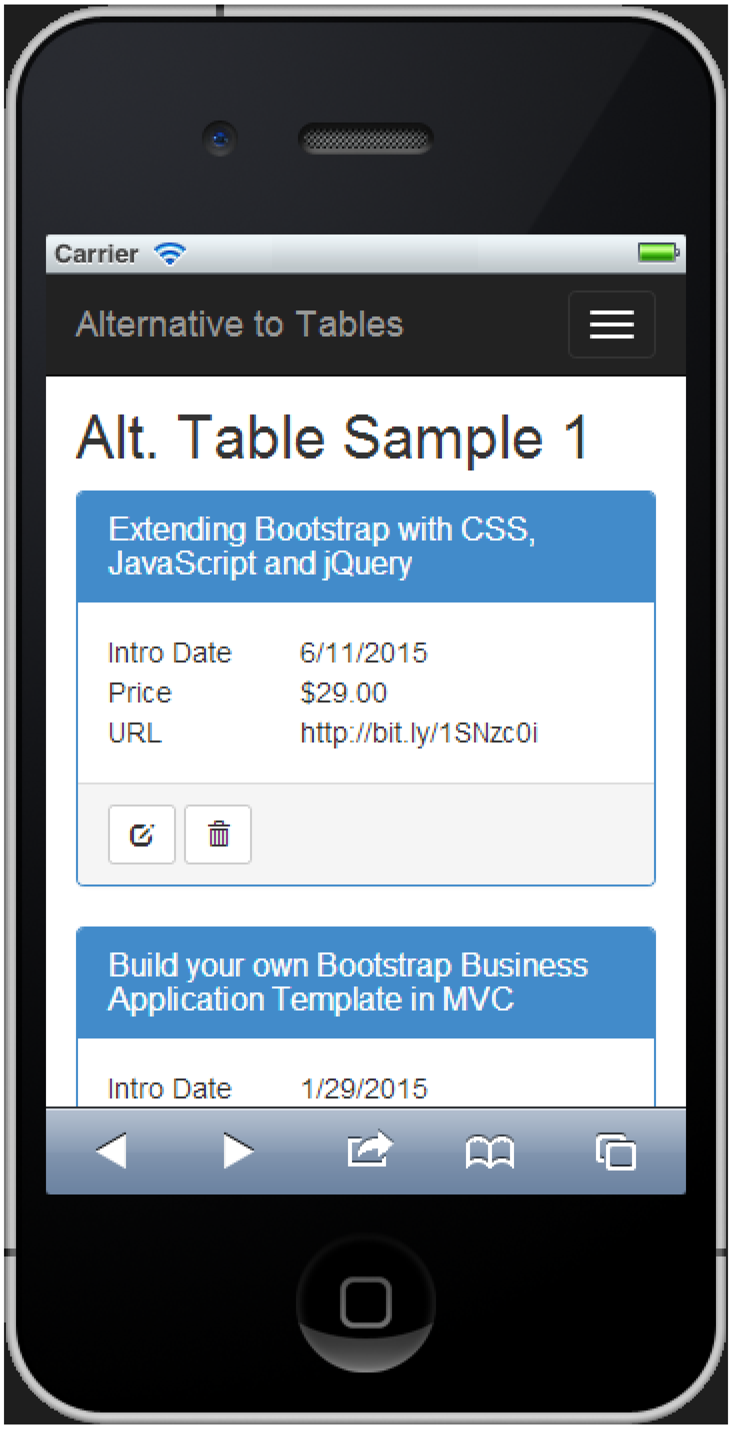
Build your own Bootstrap Business Application Template in MVC

Intro Date	1/29/2015
Price	$29.00
URL	http://bit.ly/1I8ZqZg

Building Mobile Web Sites Using Web Forms, Bootstrap, and HTML5

Intro Date	8/28/2014
Price	$29.00
URL	http://bit.ly/1J2dcrj

Figure 3: An alternative approach to a list of data rendered on a desktop browser.



Alt. Table Sample 1

Extending Bootstrap with CSS, JavaScript and jQuery

Intro Date	6/11/2015
Price	$29.00
URL	http://bit.ly/1SNzc0i

Build your own Bootstrap Business Application Template in MVC

Intro Date	1/29/2015

Figure 4: An alternative approach to a list of data rendered on a mobile browser.

Create Mock Data Classes

Instead of messing with a database, create three simple classes to build a collection of Product objects that can be rendered on an MVC page. Figure 5 illustrates the properties and methods of each of the 3 classes. To view the code for these classes, see the instructions at the end of this blog post on how to download the sample.

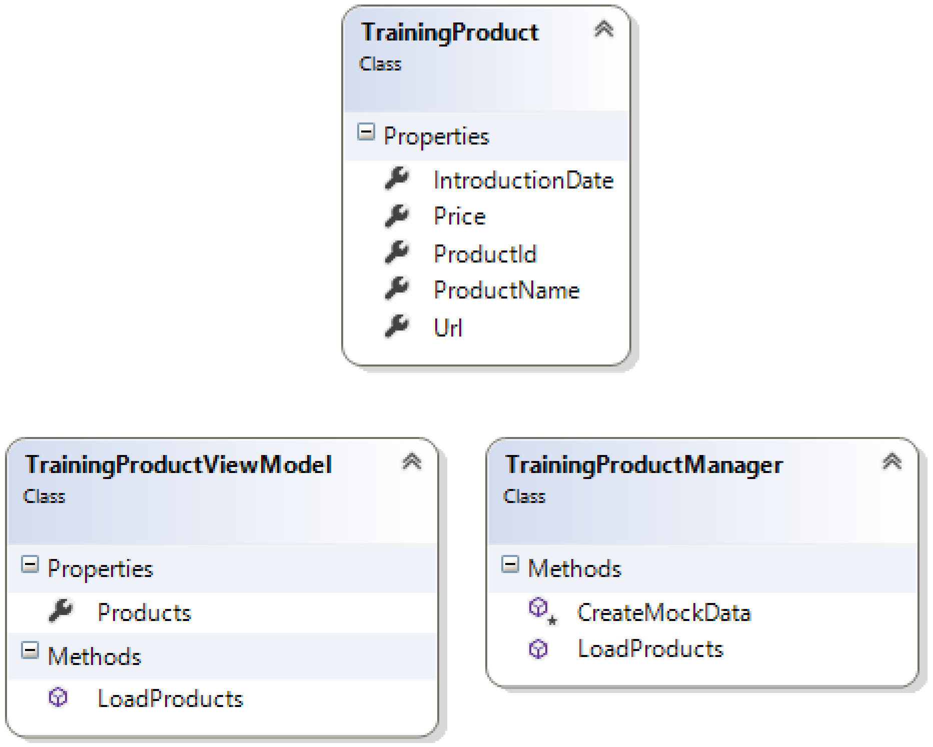


Figure 5: A set of 3 product classes help us load mock product data for this sample

The MVC Controller

You need a MVC controller to load the data and feed that data to the CSHTML pages used to render our two pages of product data. Below is the code from the MVC controller. The TableSample and AlternateTableSample methods create an instance of the TrainingProductViewModel class and call the LoadProducts method to build the Products property in the view model. It is the collection property that is used to display the list of data in both pages.

```
public class HomeController : Controller
{
  public ActionResult Index() {
    return View();
  }

  public ActionResult TableSample() {
    TrainingProductViewModel vm = new TrainingProductViewModel();

    vm.LoadProducts();

    return View(vm);
  }

  public ActionResult AlternateTableSample() {
    TrainingProductViewModel vm = new TrainingProductViewModel();

    vm.LoadProducts();

    return View(vm);
  }
}
```

Create the HTML Table

The TableSample page is the one that renders the standard HTML table. Below is the code from the TableSample.cshtml page.

```
@model AlternativeTable.TrainingProductViewModel

@{
  ViewBag.Title = "Table Sample 1";
}

<h2>Table Sample 1</h2>

@using (Html.BeginForm()) {
  <div class="table-responsive">
    <table class="table table-bordered
                  table-condensed table-striped">
      <thead>
        <tr>
          <th>Product Name</th>
          <th>Introduction Date</th>
          <th class="text-right">Price</th>
          <th>URL</th>
          <th>Delete</th>
        </tr>
      </thead>
      <tbody>
        @foreach (TrainingProduct item in Model.Products) {
          <tr>
            <td>
              <a href="#"
                 title="Edit Product">
                @item.ProductName
              </a>
            </td>
            <td>@item.IntroductionDate.ToShortDateString()</td>
            <td class="text-right">@item.Price.ToString("c")</td>
            <td>@item.Url</td>
            <td>
              <a href="#"
                 title="Delete Product"
                 class="btn btn-sm btn-default">
                <i class="glyphicon glyphicon-trash"></i>
              </a>
            </td>
          </tr>
        }
      </tbody>
    </table>
  </div>
}
```

As you can see from the above code, there is nothing out of the ordinary for this table. You use the Bootstrap table classes to help with styling the table. You loop through the collection of product data in the Products property of the TrainingProductViewModel class. Each time through the loop, you display the appropriate data from the Product class in each <td> of the table.

Create the Alternative Design

The AlternateTableSample page uses the appropriate Bootstrap panel CSS classes to create the alternative design shown in Figures 2 and 4. Below is the code in the AlternateTableSample.cshtml page.

```
@model AlternativeTable.TrainingProductViewModel

@{
  ViewBag.Title = "Alt. Table Sample 1";
}

<h2>Alt. Table Sample 1</h2>

@foreach (TrainingProduct item in Model.Products) {
  <div class="panel panel-primary">
    <div class="panel-heading">
      <h1 class="panel-title">@item.ProductName</h1>
    </div>
    <div class="panel-body">
      <div class="row">
        <div class="col-xs-4 hidden-sm hidden-md hidden-lg">
          Intro Date
        </div>
        <div class="col-xs-4 col-md-3 hidden-xs">
          Introduction Date
        </div>
        <div class="col-xs-8 col-md-9">
          @item.IntroductionDate.ToShortDateString()
        </div>
      </div>
      <div class="row">
        <div class="col-xs-4 col-md-3">
          Price
        </div>
        <div class="col-xs-8 col-md-9">
          @item.Price.ToString("c")
        </div>
      </div>
      <div class="row">
        <div class="col-xs-4 col-md-3">
          URL
        </div>
        <div class="col-xs-8 col-md-9">
          @item.Url
        </div>
      </div>
    </div>
    <div class="panel-footer">
      <div class="row">
        <div class="col-xs-12">
          <a href="#"
             title="Edit Product"
             class="btn btn-sm btn-default">
            <i class="glyphicon glyphicon-edit"></i>
          </a>
          <a href="#"
             title="Delete Product"
             class="btn btn-sm btn-default">
            <i class="glyphicon glyphicon-trash"></i>
          </a>
        </div>
      </div>
    </div>
  </div>
}
```

Let's look at each section of the above code to see how it was put together. The first thing you see is the loop through the Model.Products collection. Within this loop is where you build a complete Bootstrap panel. In the heading area of the panel is where you place the ProductName property. Within the panel body is where you create a row and two columns for each of the other properties you wish to display from the Product object.

Notice that I am changing the column widths I use depending on the size of the browser. For anything that is a medium resolution and above, (according to Bootstrap) I am using a col-md-3 for the first column and col-md-9 for the second column. However as soon as a mobile device is detected, use the size of col-xs-4. If you look at the Introduction Date field you also notice that the words used are changed also depending on the size of the browser. This is one of the great things about using Bootstrap, the ability to hide and display things using simple CSS classes.

The last difference in this code compared to the normal HTML table is I use two glyphs for the actions that the user can take. It can be sometimes hard to click on a hyperlink with your finger on a mobile device. They can even be hard to see sometimes on a mobile device. I find using large buttons with a graphic gives the user a nice big target to hit with their finger.

Summary

In this blog post, you were presented with a method to present data to the user that is different from a traditional HTML table. While you may not like the exact user interface presented here, hopefully this article will spur you to question your user interfaces a little more and come up with some alternate methods of displaying lists of data.

Sample Code

You can download the complete sample code at my website. http://www.pdsa.com/downloads. Choose “PDSA Blog”, then "An Alternative to HTML Tables" from the drop-down.