

Retailer WCF SP

by Sam Phoenix

FILE	2644_SAM_PHOENIX_RETAILER_WCF_SP_113620_526813057.DOCX		
TIME SUBMITTED	08-MAY-2018 02:57 AM (UTC+0100)	WORD COUNT	1483
SUBMISSION ID	88572999	CHARACTER COUNT	9377



RETAILER WCF

Distributed Enterprise Applications



MAY 7, 2018

Sam Phoenix (P151028)

Lecturer: Tim Bashford

Contents

Introduction	1
The service	1
The client.....	1
Problems encountered	3
How it could be improved.....	10
How this could be deployed	10
Conclusion.....	10
References	11

Introduction

This report discusses the development of a WCF service that provides ecommerce data to a client. The site in question is based on an existing side-business which sells handmade gloves; all imagery used for this exercise is also used on an existing site illustrating the products for sale in a live environment.

The service

The service is self-hosted and supplies the entity details to the client. For the purpose of this exercise, it uses lists to manage the data; if this was live, it would use the Entity Framework to manage a database.

It also stores the product images; however, as described later, there were problems getting them to transfer to the client. It does handle the image paths, although the images themselves are stored on the client end rather than the service.

The client

The client was developed using Web Forms, which includes user management such as registration and login information out of the box, as well as 3rd party login APIs – demonstrated in this assignment is the Google API, which is shown below:

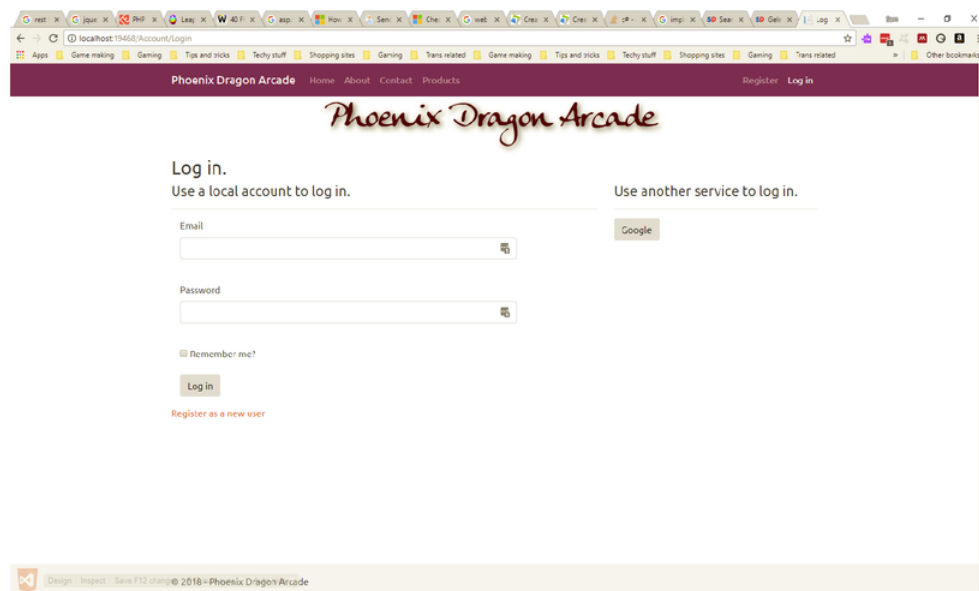


Figure 1: Login with Google enabled

Google permits the use of localhost, so the below figures demonstrate a functional login system using Google API:

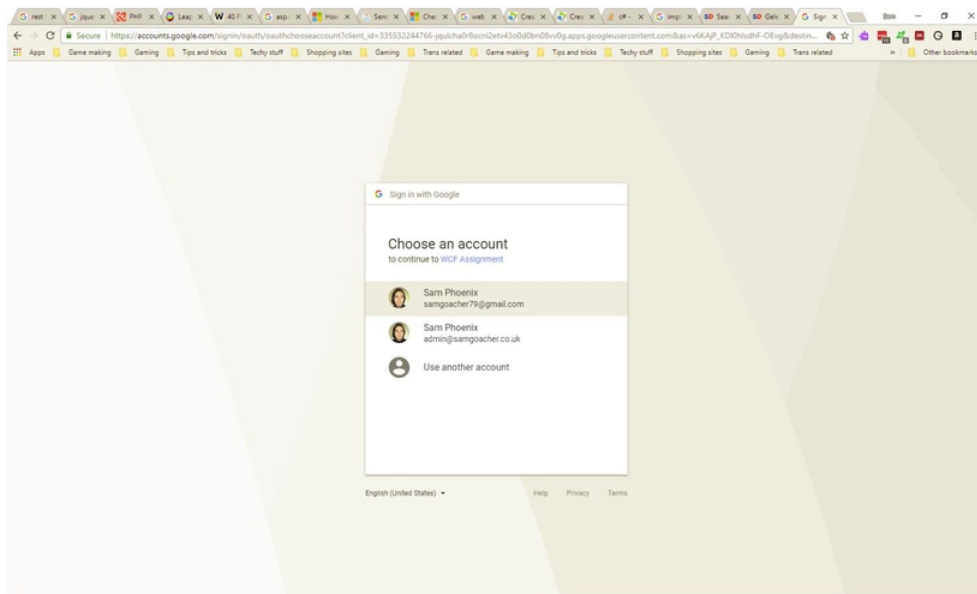


Figure 2: Logging in via Google

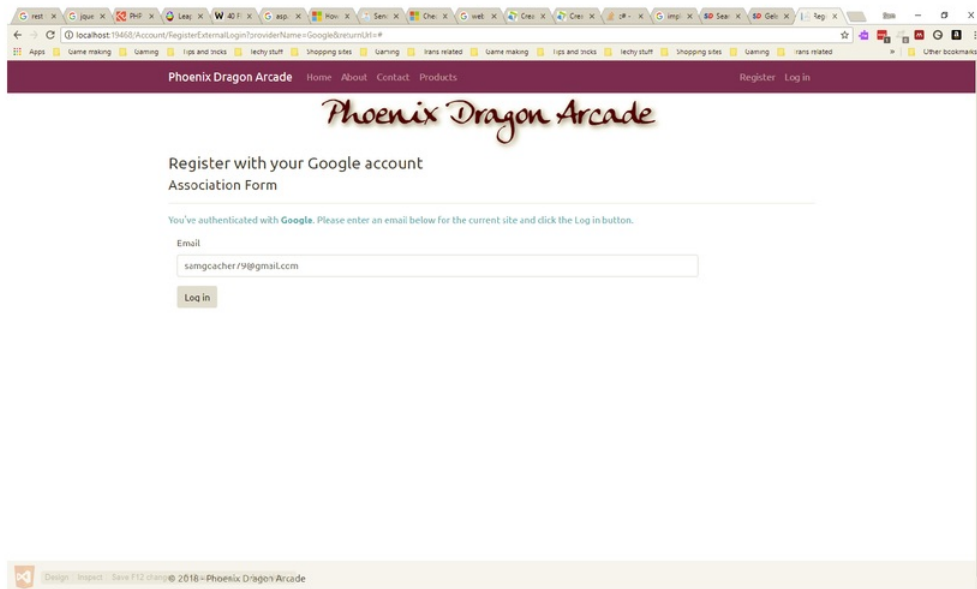


Figure 3: Register at site with Google account

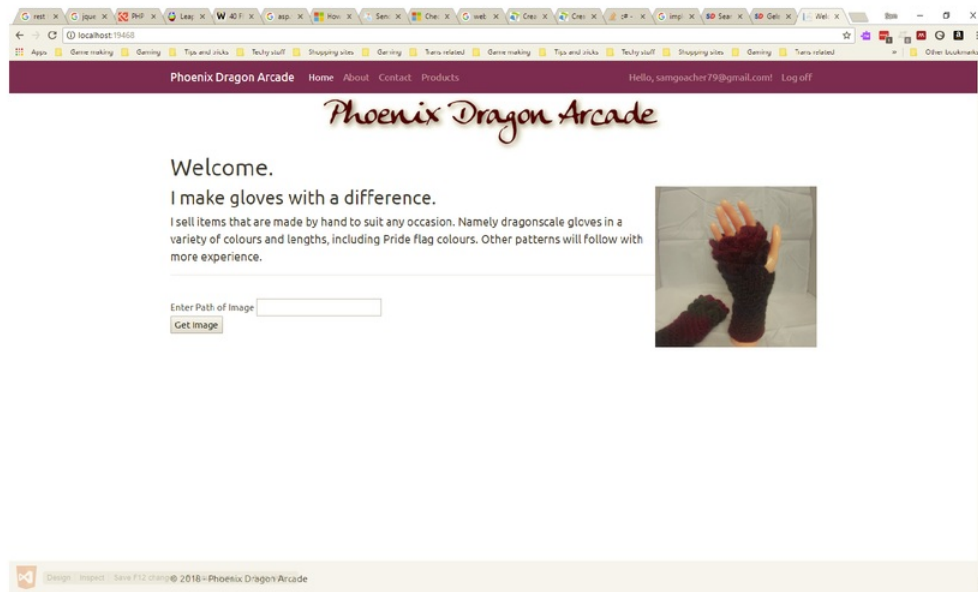


Figure 4: Back to home page, now logged in. The lower half of the page will be explained later

The 3rd party APIs are disabled by default; in order to enable them, the API credentials need to be added into the StartupAuth.cs file as shown below:

```
// Uncomment the following lines to enable logging in with third party
login providers
//app.UseMicrosoftAccountAuthentication(
//    clientId: "0000000048123D66",
//    clientSecret: "7GyCViGCGFVbPoH6FM62AI7kV0Fd6NWu");

//app.UseTwitterAuthentication(
//    consumerKey: "",
//    consumerSecret: "");

//app.UseFacebookAuthentication(
//    appId: "",
//    appSecret: "");

app.UseGoogleAuthentication(new GoogleOAuth2AuthenticationOptions()
{
    ClientId = "335532244766-
jqulcha0r8ocn12etv43o0d0bn08vv0g.apps.googleusercontent.com",
    ClientSecret = "kBLnnwmCPTw2S1jyn69Nj0vq"
});
```

This is the preferred way to handle login information, as services such as Google handle the majority of the security related to user information management. When storing these details to a database, as would be the case in a live system, sensitive details need to be encrypted using a secure means such as Blowfish.

Problems encountered

There was a steep learning curve involved, and much time was lost on not only (re)learning how to develop the client, but also on how to get it to communicate effectively with the service, as well as

finding the most effective way to host this service. It was eventually decided that self-hosting would be easier to work with, after one or two iterations tried using IIS.

While there are many resources around to help with building a website using Web Forms, most of them assume the developer will access the database in the same application, rather than via a service. This was the case with a tutorial discovered on the Microsoft site[1], although on discovering it was part of a series to build an ecommerce site, it has proved invaluable in other ways.

The initial test involved a simple page with a table, and a link to retrieve product details from the service. A demonstration of this working is shown in Figure 5 and Figure 6 below:

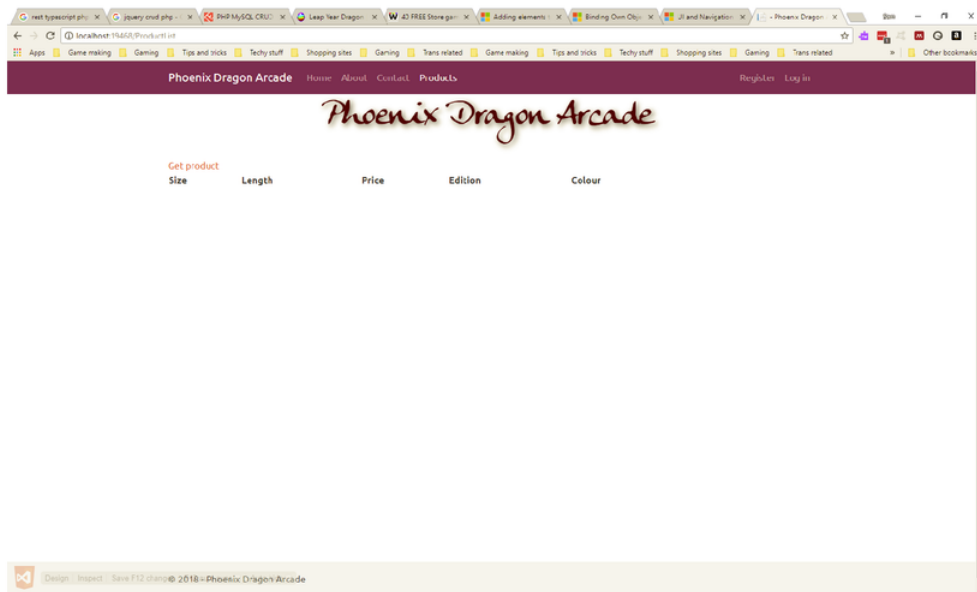


Figure 5: Test page to retrieve product details

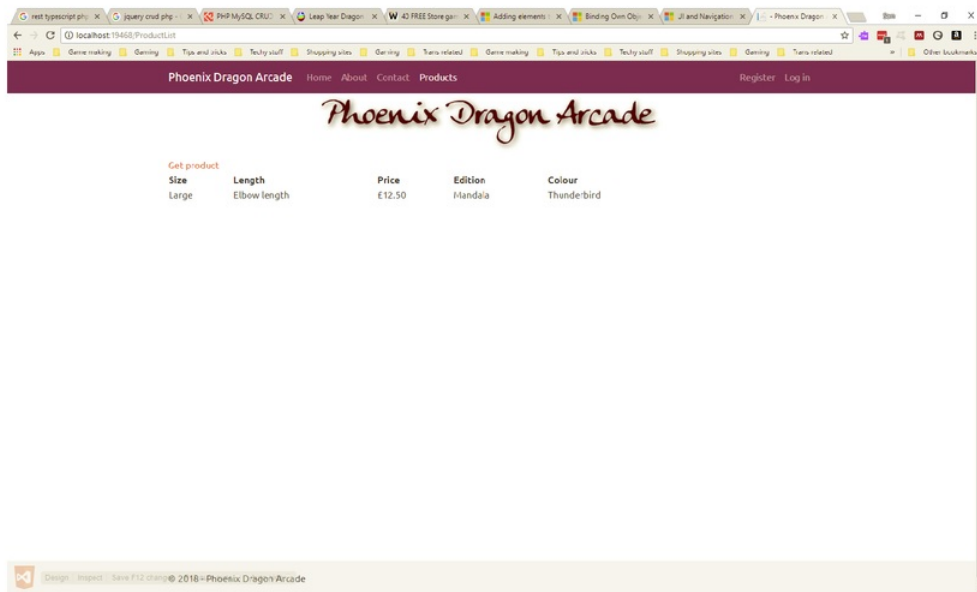


Figure 6: Clicking the link successfully retrieves data

Finding a way to list all products on a page proved to be more difficult. This was a common error encountered when trying to use the ListView tool:

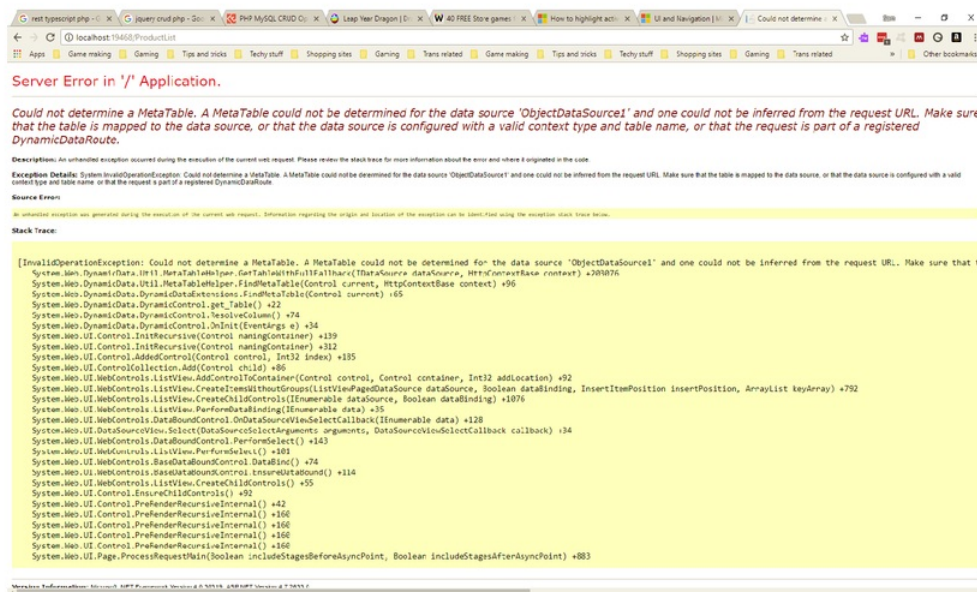


Figure 7: Error message encountered while trying to list all products

It was eventually discovered that, when not working with a database, it is best to leave the control “unbound” and set it up manually.[2] Thus adjustments to the code had to be made while following the tutorial.

Below is shown how it looks once it works as intended:

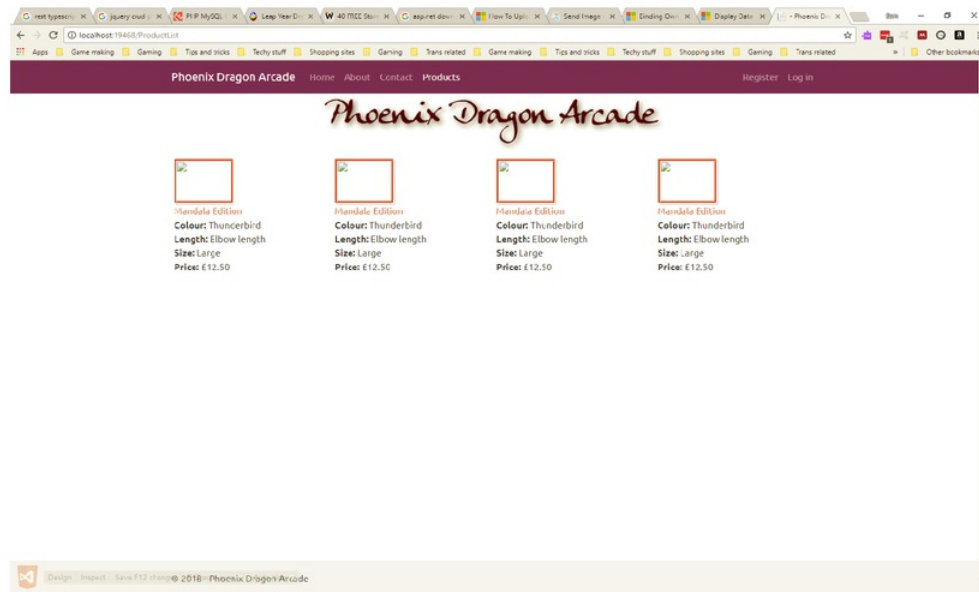


Figure 8: Working product page without images

It was found that there was a problem with how items were being added to the list – this was an error with the way lists were being populated and was an easy problem to fix. Trying to get the images to download from the service was a different story, which also cost time to solve. One solution[3] was tried, and the default page was altered to test this:

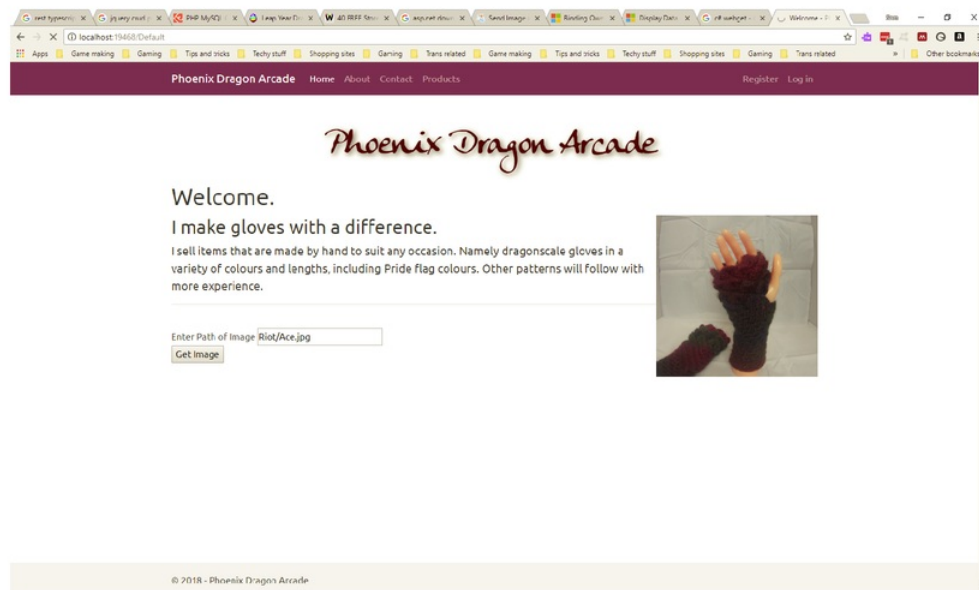


Figure 9: Testing the remote image function

After several instances of “file not found” exceptions, and struggling to work out how to alter the code to provide a valid URL to download from, the final result – the page being refreshed with no image displayed or any other kind of feedback – an improvisation had to be made:

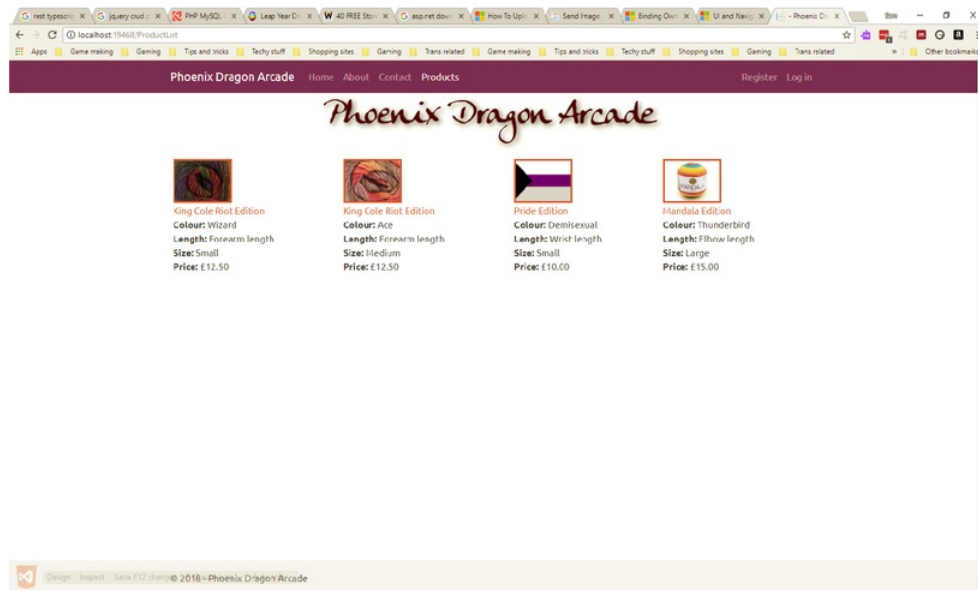


Figure 10: Product page with images

This is when it was decided to store images on the client side, rather than the service as would have been preferred.

The list also successfully links to product details, after some adjustment to the tutorial code:

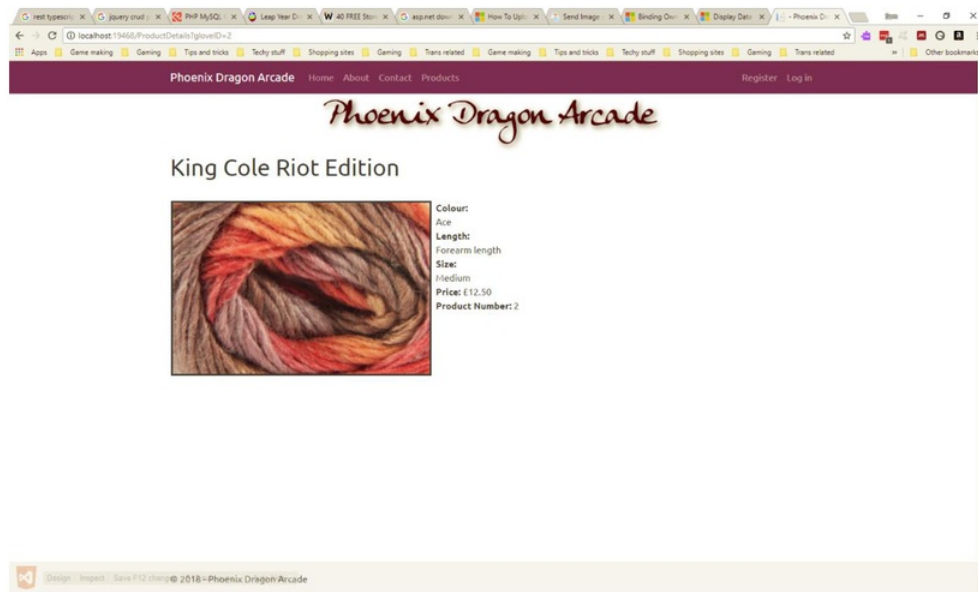


Figure 11: Product details

Ensuring the checkout can only be accessed by logged in users turned out to be a simple task; add a folder, then add a Web.config to that folder with the following setting:

```
<?xml version="1.0"?>
<configuration>
  <system.web>
    <authorization>
      <deny users="?" />
    </authorization>
  </system.web>
</configuration>
```

According to the tutorial[1], the above settings will ensure that any pages stored in this folder will be restricted to logged in users.

The shopping cart, on the other hand, caused more problems. Adding the button was the easy part:

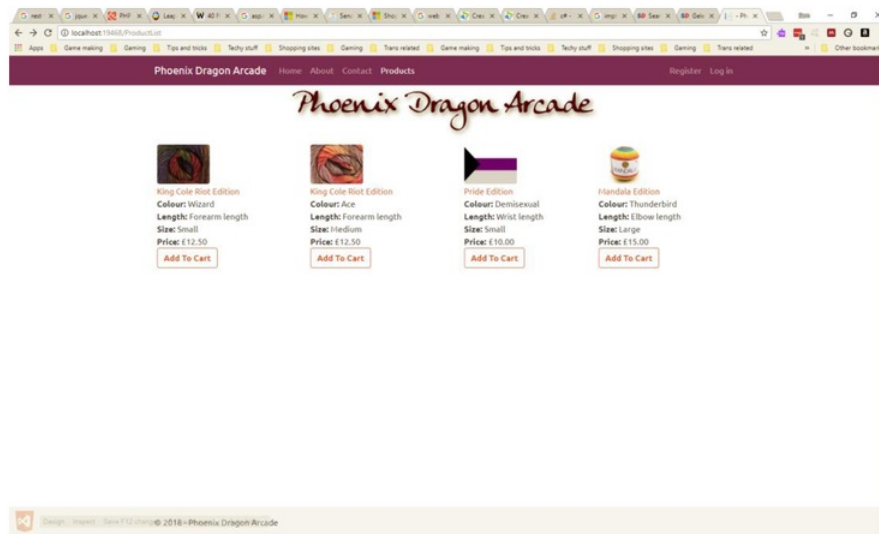


Figure 12: Updated product list with "add to cart" buttons

However, clicking the button produced this:

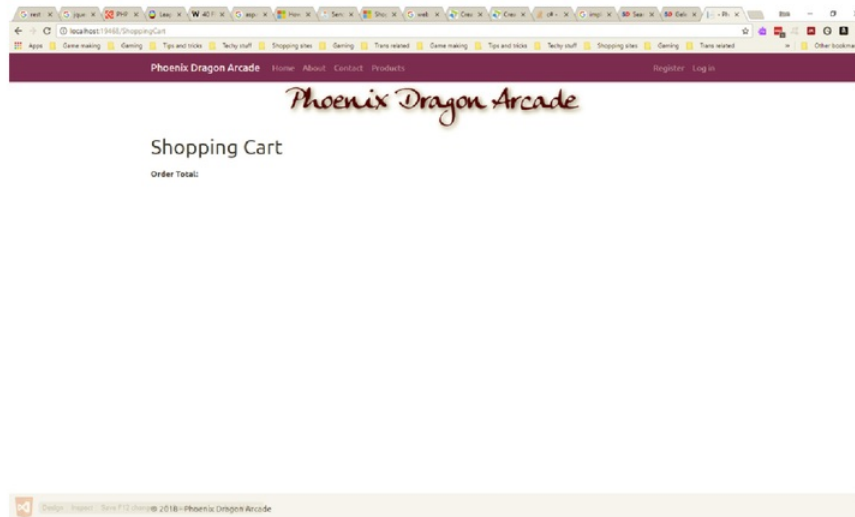


Figure 13: Empty shopping cart page

After more time spent trying and failing to debug the problem, it was decided to accept that the site as submitted is not currently fully functional. If the database was handled within the client, it would not be a problem, but accessing any data from the service makes the system more complicated.

Another problem experienced was an attempt to set up the client to use SSL – there was an error upon trying to do so, as shown below:

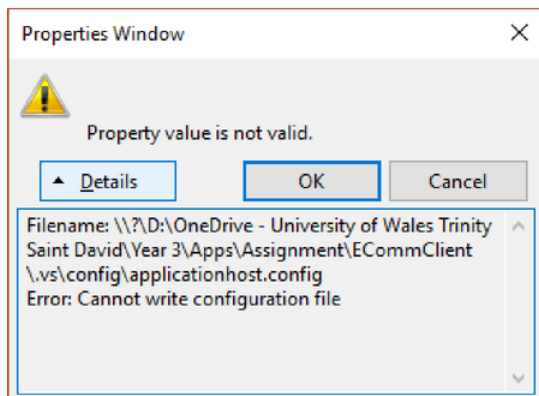


Figure 14: Trying to set up client to use SSL

Using SSL is seen as standard these days, especially when handling sensitive data such as user details. In a test environment it is not such a problem, but in a live environment it would be.

The service has been set up to allow for accessing its database rather than using lists, but that part does not work at this time.

How it could be improved

With more time to improve this system, the service would be modified to supply access to the database[4] rather than lists, with the client in turn being improved to accept data in this way[5]. It would likely then be possible to debug the shopping cart and make it functional with this improvement.

Once the core functionality has been fixed, work could continue on the other required and suggested features. Also, the secure http issue would be solved, along with image transfer across the network, rather than being stored by the client.

How this could be deployed

With a fully functioning service that provides access to the database, all the models that would normally reside on the client side would be handled instead by the service, making it a self-contained data provider. It would be hosted on a Windows Server, and accessed by either the Web Forms client, or by another web application such as one using PHP[6], which could be hosted on a different server.

Other possibilities include being accessed not just by standalone web apps, but by CMS widgets – plugins for WordPress exist that integrate shopping cart systems into a self-hosted WordPress site. It could also be accessed by mobile apps.

Conclusion

There is good potential for a web service that can supply multiple websites, and ASP.Net offers a way to simplify services, such as user management and database access, by offering tools within its frameworks. In practice, more time would be needed to get used to working with it. There were many problems encountered that may have been easier to fix were it not for making sure the service and client were decoupled.

References

- [1] 'Display Data Items and Details | Microsoft Docs'. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/web-forms/overview/getting-started/getting-started-with-aspnet-45-web-forms/display_data_items_and_details. [Accessed: 03-May-2018].
- [2] 'Binding Own Object to ListView | The ASP.NET Forums'. [Online]. Available: <https://forums.asp.net/t/1549691.aspx>. [Accessed: 06-May-2018].
- [3] 'Send Image as System.IO.Stream thru WCF REST Service and Consume it in ASP.NET Website'. [Online]. Available: <https://www.c-sharpcorner.com/blogs/send-image-as-systemiostream-thru-wcf-rest-service-and-consume-it-in-asp-net-website1>. [Accessed: 06-May-2018].
- [4] 'Create a Simple WCF service against Pubs Database using EF, LINQ – Part I | Rami Vemula'. [Online]. Available: <http://www.intstrings.com/ramivemula/articles/create-a-simple-wcf-service-against-pubs-database-using-ef-linq-part-i/>. [Accessed: 08-May-2018].
- [5] 'Create a Simple WCF Service Client against Pubs Database using EF, LINQ – Part II | Rami Vemula'. [Online]. Available: <http://www.intstrings.com/ramivemula/articles/create-a-simple-wcf-service-client-against-pubs-database-using-ef-linq-part-ii/>. [Accessed: 08-May-2018].
- [6] 'How to consume WCF service from PHP client ?' [Online]. Available: <https://social.msdn.microsoft.com/Forums/vstudio/en-US/daf37585-fcdb-4a6e-8d7b-5f990c1c49bd/how-to-consume-wcf-service-from-php-client-?forum=wcf>. [Accessed: 08-May-2018].

Retailer WCF SP

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

73/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

GRADING FORM: SMU GENERIC_GJ

SAM PHOENIX

73

RESEARCH-INFORMED LITERATURE

	An acceptable number of sources used, though disappointingly, all from the web.	2.5
---	---	-----

KNOWLEDGE AND UNDERSTANDING

	Good evidence of knowledge and understanding. The security aspects of this (using auth through google, for example) are particularly good. The report, however, is a tad brief on concepts - you've done a very good job of talking about your implementation, but could've more robustly considered the principles of best practice.	20
---	---	----

ANALYSIS

	Some reasonable synthesis - valid conclusions derived and a good effort overall.	15
---	--	----

PRACTICAL APPLICATION AND DEP.

	A very good implementation. A few things (such as products) are hard coded, and I suspect that polymorphism might've saved you quite a bit of development time in repeating boilerplate, but nonetheless, a well planned and implemented piece of work.	31
---	---	----

SKILLS FOR PROFESSIONAL PRAC.

	A well-written, nicely presented assignment.	4.5
---	--	-----