FINAL REPORT

BTECH 450

JUN HEE KANG

4277795

Abstract

The project is to develop websites for fast and precise data retrieval from databases of records. Two different organisations are involved in the project, NZInstitute and ISSG. Although the project is similar in the terms that they require a web interface, the details of the pages differ greatly. NZInstitute deals with a very small database, dedicated to provide users with interesting data representation, while ISSG deals with a much larger database, with the aim focused on faster queries and flexibility in search. ASP.NET in conjunction with SQLServer will be used, and for NZInstitute, to provide interactive interfaces, Silverlight applications will be embedded into their website. The final goal is to compare the two websites, and come up with ideas and implementations of how to combine the common factors, and create methods that could be used in dynamic websites as a whole. Ideas from all tiers of the system are to be included in this combination.

Contents

About NZInstitute

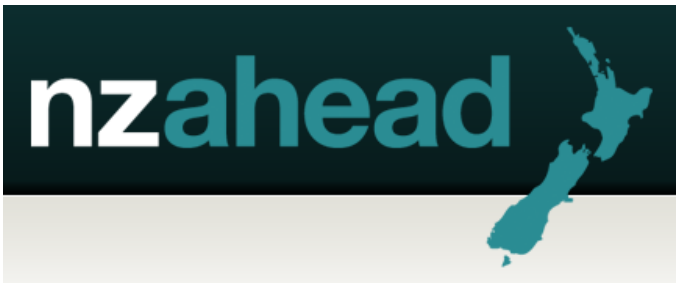
The New Zealand institute is a privately funded think tank, committed to generating ideas, solutions and debate that will improve economic prosperity, social well-being, environmental quality and environmental productivity. The Institute produces creative, provocative and independent thinking, focusing on key issues that have a major impact on New Zealand's economic and social and environmental future, and engages with New Zealanders in order to develop solutions to address these issues.

They provide suggestions to overcome the problems New Zealand faces to reach its full potential. To do this, they conduct research on various measures to compare New Zealand's performance relative to other countries, usually the OECD countries.

Criticism and research of current government policies are carried out via

The solutions suggested are made sure that they are practical and relevant to the real world. They deliver their messages to the public through various forms of media such as radio and online websites.

http://www.nzinstitute.org



http://www.nzahead.org

About ISSG

The ISSG promotes and facilitates the exchange of invasive species information and knowledge across the globe and ensures the linkage between knowledge, practice and policy so that decision making is informed.

They work with various organisations around the globe to transfer information and share databases not only with the public but with other organisations.

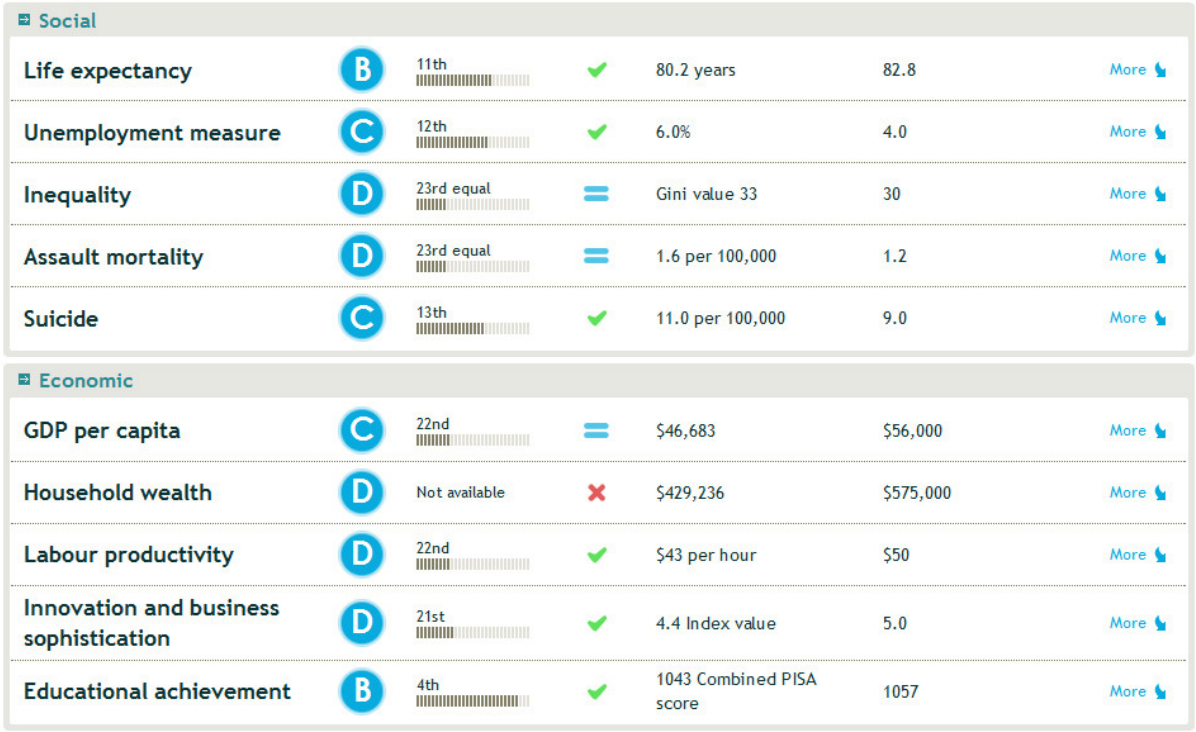
Website

http://www.issg.org/

Requirements of NZInstitute

A major project in the NZInstitute that has been worked on for a long period of time is a website launched by the institute early 2010 to suggest a standard measure of the living quality of NZ is a website called NZahead.

It is a website to display information about sixteen of the most crucial measures of living in New Zealand. Each of the measures is given a grade which depends on the current statistical value of the measure, which is compared to other OECD countries (for relative performance). A goal for the future (2015 currently) is also suggested for each measure. On the individual measure pages, recent relevant media and opinion pieces are uploaded for further information. Each of the measure is divided into three categories; social, economic and environmental.



Social						
Life expectancy	B	11th	✔	80.2 years	82.8	More
Unemployment measure	C	12th	✔	6.0%	4.0	More
Inequality	D	23rd equal	=	Gini value 33	30	More
Assault mortality	D	23rd equal	=	1.6 per 100,000	1.2	More
Suicide	C	13th	✔	11.0 per 100,000	9.0	More
Economic						
GDP per capita	C	22nd	=	$46,683	$56,000	More
Household wealth	D	Not available	✘	$429,236	$575,000	More
Labour productivity	D	22nd	✔	$43 per hour	$50	More
Innovation and business sophistication	D	21st	✔	4.4 Index value	5.0	More
Educational achievement	B	4th	✔	1043 Combined PISA score	1057	More

Above is a sample from the NZahead website. The values are updated every quarter (3 months).

As it focuses on past data, there is no ‘digital’ means to save and view past data. When there is an update on the page, the old data is deleted and there is no means to track changes in the data.

As my project at NZInstitute, a web interface will be developed to track changes in these values. This will involve web implementation, which will access the database of past and current data. Also to make the data more user friendly way to see the changes more clearly, gadgets such as graphs will also be implemented.

Requirements of ISSG

All over the globe, there are various websites to represent data from their database of species. The ISSG have also got a database they wish to share information with.

After the formation of the project proposal for NZInstitute, a recommendation was made to me to join ISSG in developing websites for them to retrieve data from their databases and provide information to others over the web.

Because the database contains lots information with lots of records, the database will have to be optimised for the queries that will be used the most.

The database that will be dealt with is a database that stores information about the islands and the species that live on the pacific islands. The data is centric on each of the islands, the invasive species and the threatened species.

Some examples of how the website will be implemented can be viewed at other organisation's websites.

Red List especially the 'other search options'

http://www.iucnredlist.org/

Ramsar Sites Database

http://ramsar.wetlands.org/

Similarities and Differences

The 450 project is divided into two parts, each part for each company and both running simultaneously. The projects have differences and similarities in terms of implementation and user requirements.

Similarities

The main focus of the organisations is similar in the sense that they are requiring a web interface to retrieve data from their databases and to do this remotely (following from the point that they need a 'web' interface). The architecture of the projects will be similar; user interface and server with an underlying database.

Differences

However they are two very distinct projects. Their specific requirements and usages, and also their implementation behaviour will be different.

NZInstitute's database will only contain 16 entries per three months, meaning it will take a long time for the database to grow big. Also, the project's main focus is on delivering the data in an interesting fashion, not quick and efficiently.

ISSG's database is much larger than NZInstitute's, and the main focus is to deliver data to the user as the data is stored in the database. It does not store information over time, so data manipulation at the program layer is irrelevant.

In simpler terms, NZInstitute's website is user centric and makes the data more interesting to viewing, and ISSG's website will focus on the data and its query speeds.

Overlap

Although the two projects have many differences, they will be implemented with the same technologies. After the two prototypes have been implemented, the projects will be investigated to see if any tiers of the architecture have components that could be factored out (common components) so they can be modified and put into one superclass, so they can be later used on other websites and projects to avoid the same coding or configuration being implemented again; i.e. save workload later on.

Technologies used

Both organisations require a dynamic website that will return different datasets for different databases. Due to this reason, ASP.NET is a suitable framework to use as a user interface platform. For both organisations, ASP.NET will be used for the user interface tier. With the release of ASP version 4 in 2010, the use of the new API will allow more functionality to be embedded into the websites. Another well known tool that could be used is PHP. Before the decision was made to use ASP.net, a comparison was made with the two tools.

	PHP	ASP
Producer	Open Source	Microsoft
Server Platform	Unix, Linux, Windows	Windows only
Speed of writing	PHP has a much wider range of built in functions, and a less verbose syntax, making it a much quicker language for development. Other features of PHP also greatly simplify and speed up project development: • type conversion - PHP converts types automatically • Form, session and cookie variables are available to use straight away on the page	ASP has a much tighter error checker than PHP's default. This means that error reports are more common in ASP. This is also a positive when you are learning as, unlike PHP, ASP will not allow you to get away with sloppy code.
Flexibility	PHP seems to be the more flexibile language. Its open source nature means that anyone can add features as they come across the need	ASP is a very rigid language - what might be efficient methodologies in PHP are completely unworkable in ASP. (ie the absence of associative arrays)
Reliability	In themselves, both languages could be said to be equally reliable. However, as PHP will run on Linux and Unix, and ASP is bound to windows, the reliability of linux and unix count in the favour of PHP.	
Speed of execution	PHP *seems* faster	
Ease of Learning	PHP is a much friendlier. Additionaly, there are more high quality online resources to help you.	ASPs more rigid structure and longhanded ways of doing things may be intimidating to the beginner.
Derived from	PHP syntax is derived from C++	ASP syntax is derived from VB

From: http://www.me-u.com/php-asp/phpvsasp.htm

The information is taken from a blog of a personal developer, and other blogs mostly contain the same information.

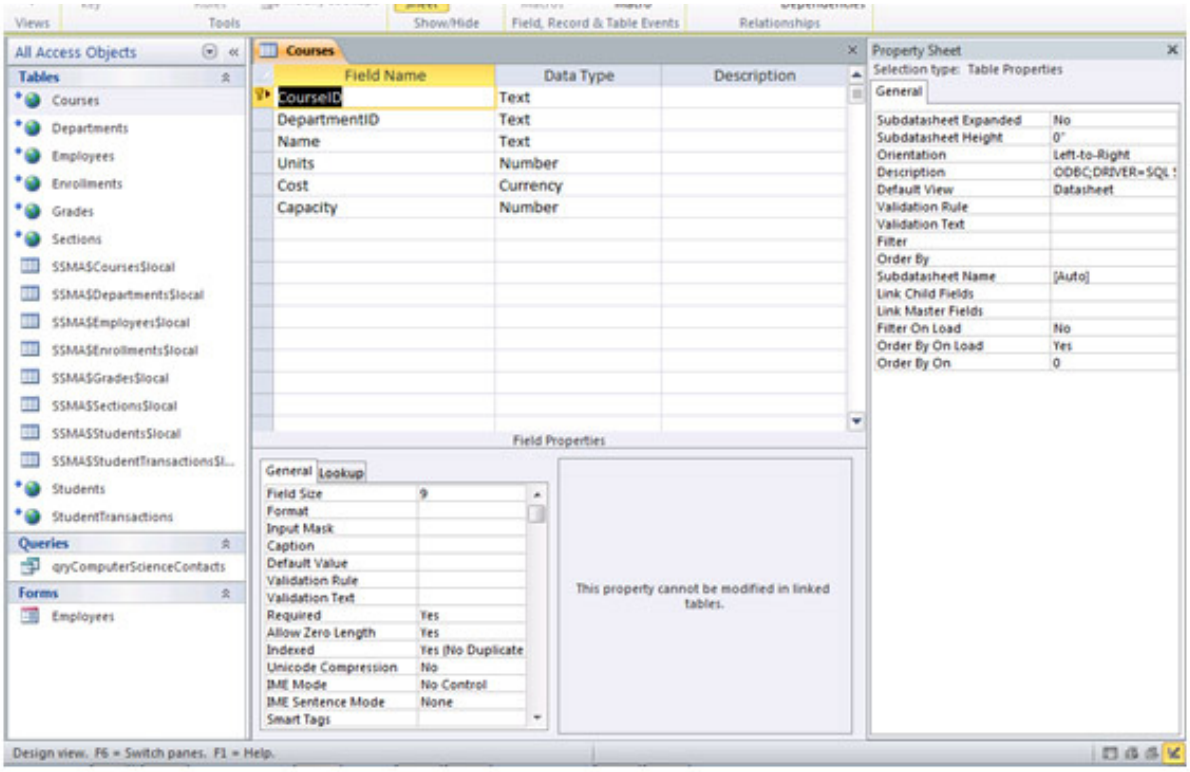
We can see that the 'usual' developer favors PHP over ASP. This is mostly to do with the fact that PHP is open source, and it allows flexibility in terms of other tools used and platforms. But due to the circumstances given for the project, ASP.net is a much more suitable choice the reasons are:

- Development language is either c# or VB for ASP. With prior C# experience, a new language did not need to be learnt; the point of the project was to look at the fundamentals and look at similarities, not
- IIS servers will host the websites, which means that ASP is much more stable in the given servers.
- Database is given in access. Conversion to MS SQL is already given in access, but to other tools can be a time consuming job.
- Other tools for development. The websites also need to take in account the possibilities for expansion later on, which will also likely to be .NET projects.

The prototypes also need a database to form their query results from. Currently NZInstitute does not have a database, so the database has to be made. As for ISSG, their databases are all in access format, which is not suitable for large databases. Access also has problems with security. Access is more vulnerable to sql injection attacks, and user level security must be set up manually.

Due to this reason, and for fast and reliable queries to be allowed, a new database will be formed for NZInstitute, and the existing database will be re-made for faster queries, and then migrated. Both organisations will use MS SQL Server, as it suits the web interface model better than access and is reliable with the use of .NET framework.

However MS SQLServer possesses the problem that it, on its own, does not provide an interface for inserting the data to the database. A separate interface must be included, which is beyond the scope of the project. To resolve this, Access can be run concurrently with SQLServer, using auto updates. Especially for ISSG where they upload data from a set format spreadsheet (access allows mass uploads from this, and the whole organisation is familiar with access), keeping the old interface and having more security is a huge plussage to the group.



Two options exist when converting the database to SQLserver. One is 'download' where the database is copied to a new copy of the SQLserver format. This case is good when there is a complete transition to the new format i.e. the access version is not to be used in later purposes. For this project, the access database is needed for the upload of data (no separate upload interface required), so downloading the database is not a very good option. The other option available is to 'link' to a new database. This will have the two versions of the database, access and SQLserver to run together on the same dataset. Synchronisation is done automatically, but refreshing the tables on update is a safer option.

Above is an example of a converted database. We can see that there are two copies of the tables, one backed up as an access format table, and the other as a SQLserver format (earth logo).

Since NZInstitute requires a website that is highly interactive with the user, additional capabilities other than just data retrieval, there is also the need to implement and embed gadgets to represent the data in various ways other than tables. For fluent communication between other aspects of the project, Silverlight has been chosen. A reason for this choice includes the fact that all other choices of technology are MS tools, and so is silverlight. Due to this reason, implementation and insertion of the silverlight application is easier to do onto the project, and communicates better with the other elements of the project.

An example silverlight application is given in the link below.

http://joestegman.members.winisp.net/Jelly/Pie.htm

Database design NZInstitute

Only containing information about the sixteen measures on the NZahead website, the database structure is not really important, but it must be easy to maintain. The two possible implementation possibilities are:

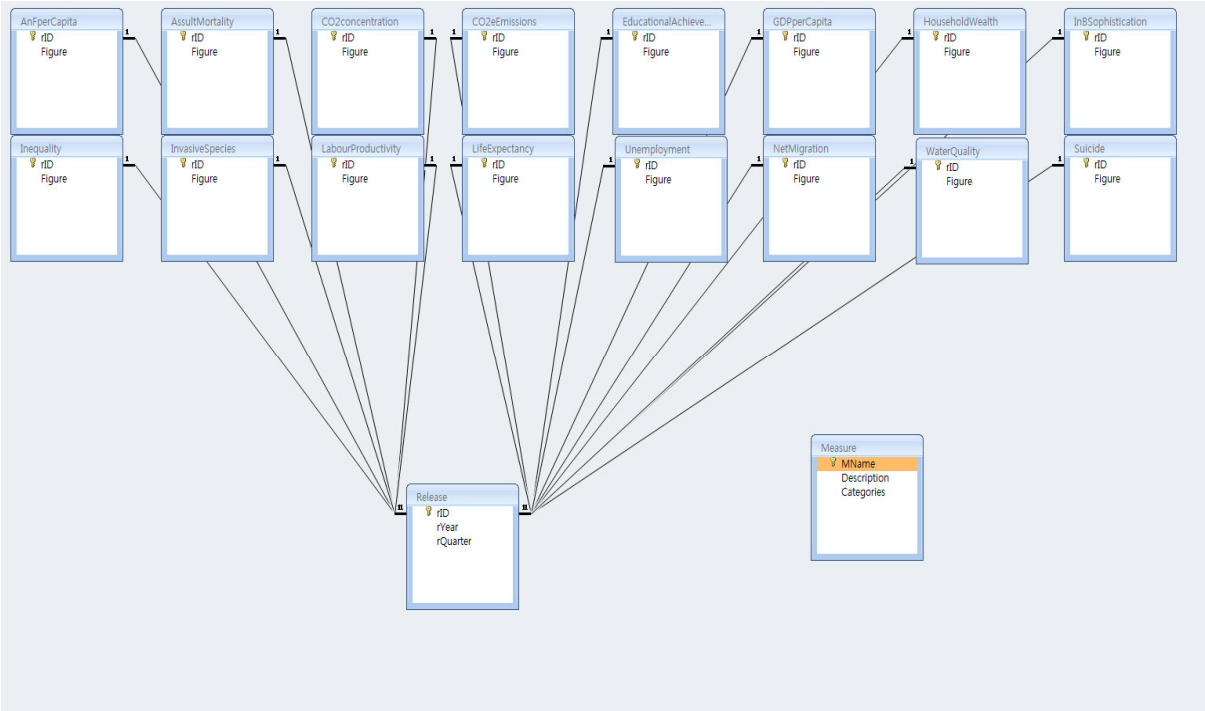
1. To have a single table database that stores the measure name as a attribute
2. Or to have a 16 table database that stores the measure name as the table name.

The two possible implementations have their advantages and disadvantages. To summarise them,

	Design 1 (single table)	Design 2 (multiple tables)
Advantages	• Supports grouping (category of measure)	• Faster queries (not too many records in this case, relevancy?) • Capability to retrieve multiple measures at once • Units are different for each measure- data type is hard to choose.
Disadvantages	• May take longer in queries • Harder to make changes to certain measures	• Hard to understand structure on first glance

On the outline, advantages and disadvantages of the database design is outlined. The main focus of the database is easy management; and from the table above, the second design is easier for many to understand as the division between the measures is made clear. The fact that each measure has its own units, this means that problems can be caused in storing the data if only one table for the record is used. Due to this reason, having only one table for the records is not a very good option.

More information than the values itself must be recorded, such as the dates of the updates and the OECD ranks etc. The final diagram of the database structure is shown below.



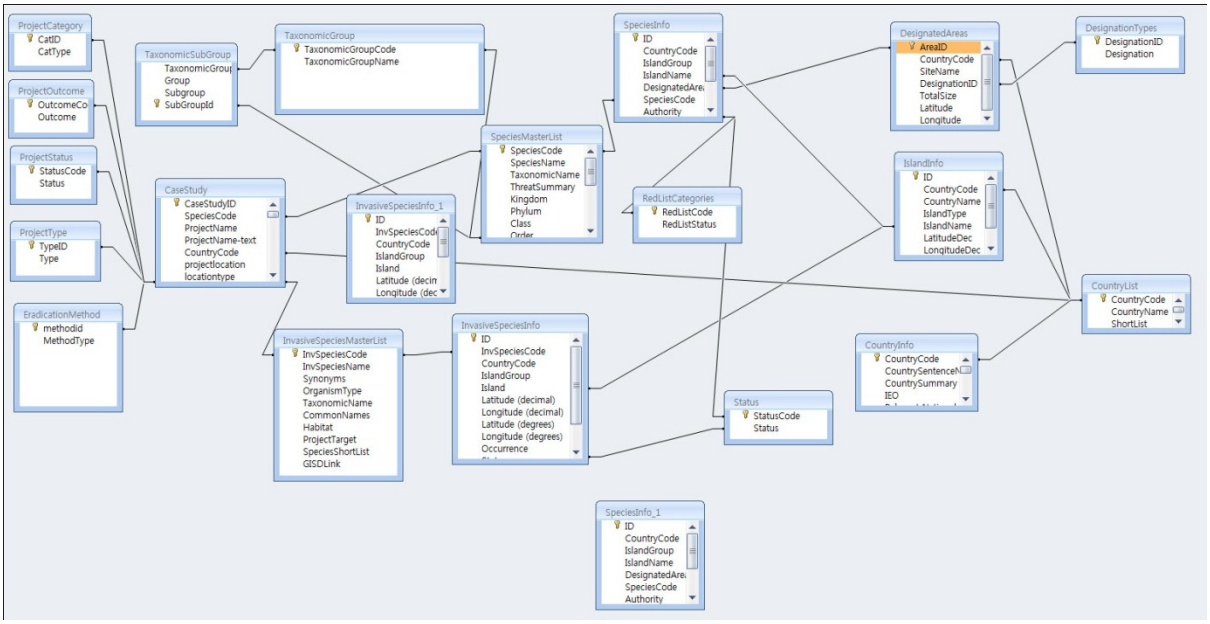
Deletion can be done by removing the whole table. Also, it is possible to remake the table and input data- which is much easier to do than going through the database and taking out individual records.

The database can be viewed as three different entities although there are 18 tables that are inside the database. First are the measures, which is the upper half of the picture. Each measure of living is given a separate table inside the database for storage due to management issues and unit of measure issues. The second is the timestamp, connected to the measure tables. The updates are done on a quarterly basis, and the point of this website is to show the changes over time, so a table was reserved for the purpose of storing when the updates were done. The last table is the measure table, which has the 16 measures in them. It stores the measures, their categories, and its description. This is included to allow for future development.

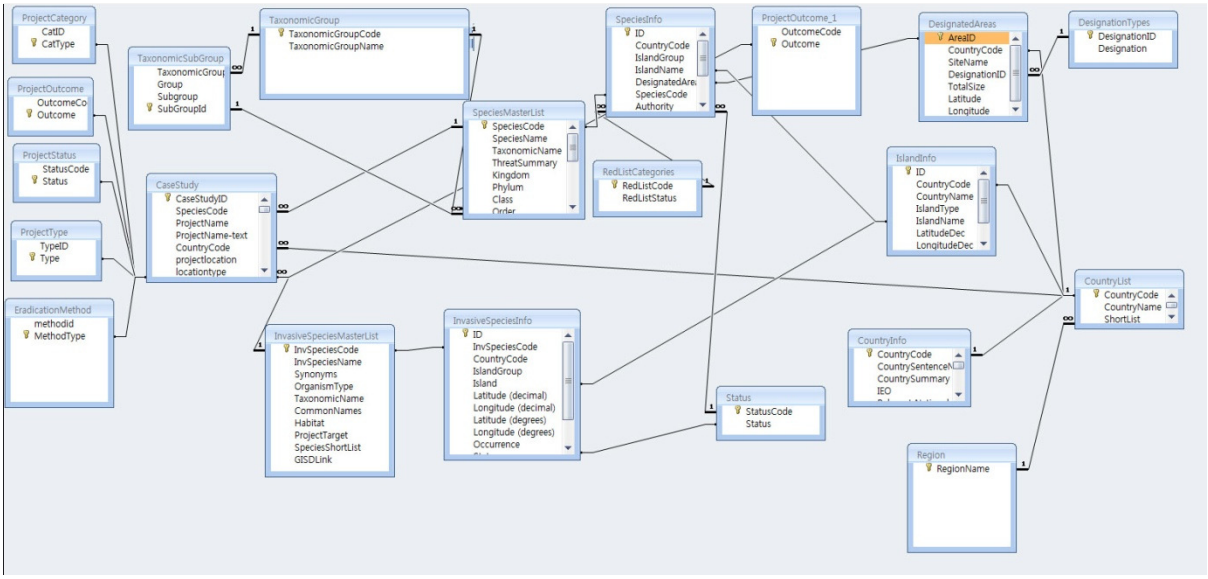
Database design ISSG

Database structure

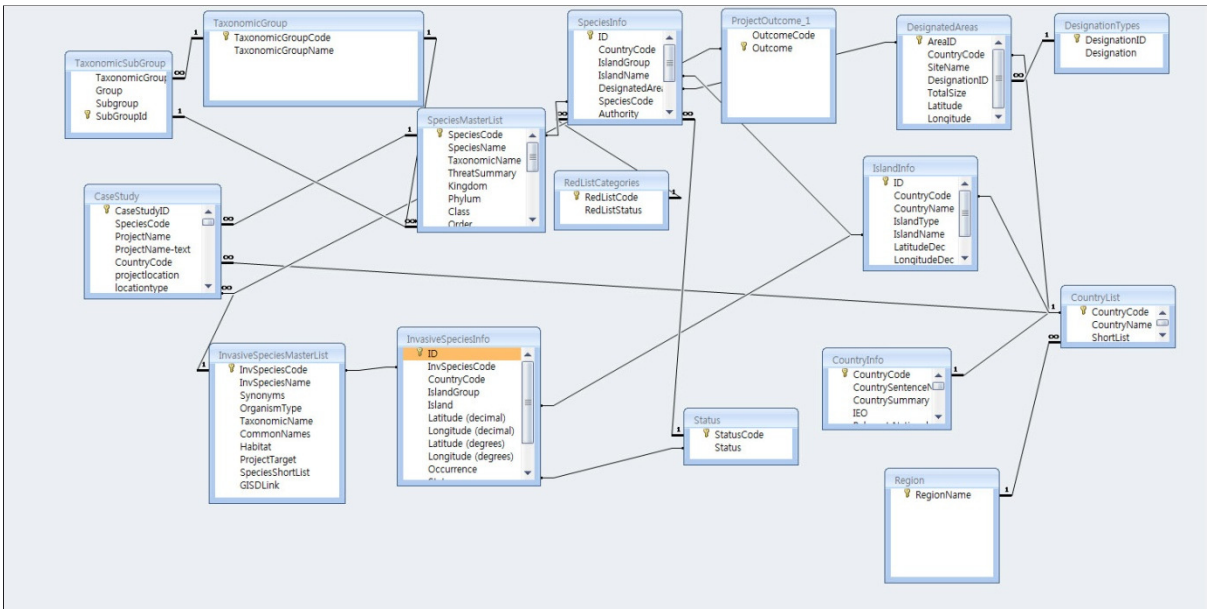
The IBIS database which will be the first database that will be worked on has all its data into the database already. However, the queries are not optimised for queries yet; the structure of the tables must be optimised to suit queries that will be of use most frequently. Looking at other databases and talking with ISSG, it was realised that the need to optimise the database for fast retrieval of data of islands, invasive species and threatened species is the number one priority. The original structure is shown below.



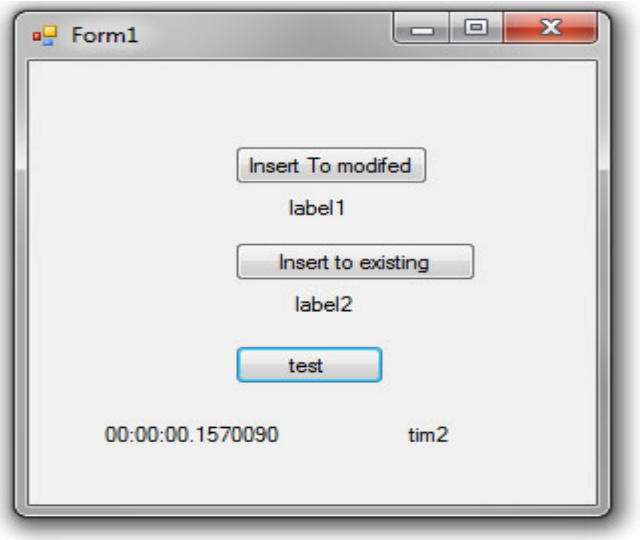
As seen above, there are tables not needed in the database, and duplicate tables, which likely happened due to mistakes in the implementation stages. The relations are not dependant on where they should be. First, a tidy up of the database was done. Duplicate tables removed, and the relationships were tidied up so the database is more ‘understandable’.



The tidied up version of the database was used for most of the implementation phase. After a working copy of the website was implemented, the database structure was looked at again to optimise performance on the queries. The project table has 8 joins in them for a query to be executed, and 5 of them could be replaced with an enum value. So another modification has been made.



The five tables that were originally placed on the left hand side are now removed, so the queries now do not need as much joins. Theoretically, the query to retrieve data from this table should now be faster. To test this, another small application was developed.



The application consists of three buttons. First and second buttons insert the dummy data into the two databases, the modified version of it and the original version (100,000 records into each). When the upload is finished, the labels under each of the buttons will change text to confirm that the update was successful.

The third button is the testing button. The query speeds of the two databases are timed, and the times taken are displayed on the two labels down the bottom (The screenshot was taken at a later time, when other queries were measured hence only one shows the time).

The query to get entries from the table was conducted three times.

Modified	1.2	1.3	1.2
Original	3.4	3.5	3.2

Shown on the table above, we can observe a massive speedup with the new database. Because the tables that were removed had 3-5 entries in them, the retrieval speed from these tables will not have been great. However it was surprising to see how slow the join operation really is.

Throughout the papers done in the university, it was heard numerous times that the join operation is expensive and should be avoided when possible. It was interesting to see how badly the join operation can get even with small tables.

Database query speeds

It was pointed out earlier that there is the issue of query speeds of the database. With the potential that the database will grow to a large dataset with a lot of records in them, it is important that the common queries will execute within an acceptable time. Taking some of the most frequent queries that will be done, the query speeds were timed to check if the queries were done quickly enough.

The queries shown below are queries that are the most frequently used queries of the project. Each query was tested with three different values.

Country

```
Select c.CountrySummary, c.IEO FROM CountryInfo c, CountryList l WHERE l.CountryName = ? AND
c.CountryCode = l.CountryCode
```

Times: 0.23s, 0.18s, 0.21s.

Invasive

```
SELECT TaxonomicName, CommonNames, Synonyms, OrganismType, Habitat FROM InvasiveSpeciesMasterList
WHERE InvSpeciesName =?
```

Times: 0.14s, 0.15s, 0.16s

Native

```
SELECT l.TaxonomicName, l.Kingdom, l.Phylum, l.Class, l.Order, l.Family, l.Genus, l.Species, l.habitat,
t.TaxonomicGroupName, s.Subgroup FROM SpeciesMasterList l, TaxonomicGroup t, TaxonomicSubGroup s
WHERE l.SpeciesName = ? AND l.TaxonomicGroup = t.TaxonomicGroupCode AND l.TaxonomicSubGroup =
s.SubGroupId
```

Times: 0.16s, 0.17s, 0.16s

The timing was tested on a separate C# program, not the ASP project itself. It was to make the testing easier; it is the query speed that needed to be tested, not how well the server will cope with the queries.

Overall, the query speeds were satisfactory. The species tables having more records in them, it was expected that the country query will take much less time than the others, but taken less time.

Middleware for ISSG

ASP is becoming more and more user friendly as the new versions are released. Most of the controls can be data bound so the user does not have to deal with too much coding in the background. This is the case when the database is structured especially for the website.

The middleware (business logic) for ISSG had to be implemented fully on the C# behind code. Although the database was modified, it was in terms of cleaning, not optimising. Also, too many changes in the database will cause problems as the upload mechanism of the database is using a set format xls spreadsheet. The format is hard to change as it is the way done by the group.

Standard event handling methods are run on the pages. The page load events and button clicked events. On the page load event, the initial data binding of the controls are done. The connection string is called, the command formed according to the page, and controls on the page loaded with the query results as such. Because the page load event is fired on button click, it must not bind if the page has been loaded initially. An example is shown below.

```
protected void Page_Load(object sender, EventArgs e)
{

    if (rDropDown.SelectedIndex == -1)
    {
        String connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=" +
Server.MapPath(@"~\IBIS_5-5.accdb") + ";Persist Security Info=False";
        OleDbConnection conn = new OleDbConnection(connString);
        String query = "Select RegionName FROM Region ORDER BY RegionName";
        OleDbCommand cmd = new OleDbCommand();
        cmd.Connection = conn;
        cmd.CommandText = query;
        cmd.CommandType = System.Data.CommandType.Text;

        OleDbConnection conn2 = new OleDbConnection(connString);
        String query2 = "Select CountryName FROM CountryList ORDER BY CountryName";
        OleDbCommand cmd2 = new OleDbCommand();
        cmd2.Connection = conn2;
        cmd2.CommandText = query2;
        cmd2.CommandType = System.Data.CommandType.Text;

        try
        {
            conn.Open();
            OleDbDataReader reader = cmd.ExecuteReader();

            rDropDown.DataSource = reader;
            rDropDown.DataValueField = "RegionName";
            rDropDown.DataTextField = "RegionName";
            rDropDown.DataBind();
            conn.Close();

            conn2.Open();
            OleDbDataReader reader2 = cmd2.ExecuteReader();

            cDropDown.DataSource = reader2;
            cDropDown.DataValueField = "CountryName";
            cDropDown.DataTextField = "CountryName";
            cDropDown.DataBind();
```

```
            conn2.Close();
        }
        catch (Exception ex)
        {
        }
        rDropDown.Items.Insert(0, new ListItem("--All Countries--", String.Empty));
        rDropDown.SelectedIndex = 0;
    }

    if (rDropDown.SelectedIndex == -1)
    {
        Page.Response.Redirect("Countries.aspx");
    }
}
```

On most of the button clicked events, a session variable will be created to pass on the information to the navigating page. An example is shown below.

```
protected void LinkButton4_Click(object sender, EventArgs e)
{
    Session["country"] = cDropDown.SelectedItem.Text;
    Session["crcheck"] = "";
    Page.Response.Redirect("Country_Island.aspx");
}
```

Custom http response was also used to insure downloading of tables in csv format. A text stream was created, and then data was inserted using a loop. The file stream can now be sent to the user as an HTTP stream. Depending on the user's preference on how to view this (default is viewing on excel), the user's machine can handle the data appropriately.

Middleware for NZInstitute

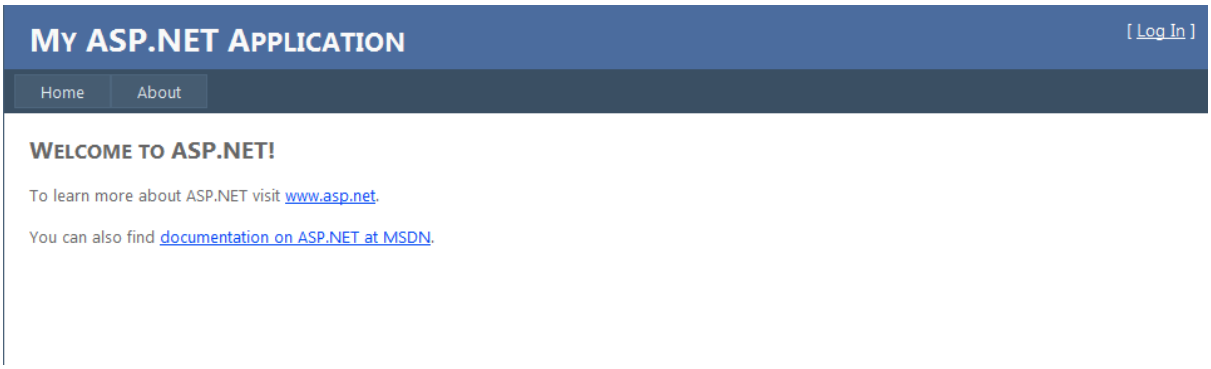
The database was made from scratch in this case, so some binding could be done on the default page of the website. The results page, depend on a lot of parameters and have to be dealt with dynamically, so the code is implemented like the NZahead website.

The results and graph page had to be made with C# code, as there are a numerous possibilities of how the user can ask for the data (different times and measures).

Frontend design methods

A blank ASP.net 4 project for a website provides a master page with the basic structure of the website formed. Since the focus of the website is not on making the appearance pretty but to look at the principles behind implementation, the master page was not modified to a great level of extent.

Customisation for each websites was done only on the image and stylesheet level. For each website, the header image was created depending on the needs, and the CSS file of the project modified to suit the colour scheme of each organisation. No extra elements were inserted to the CSS file.



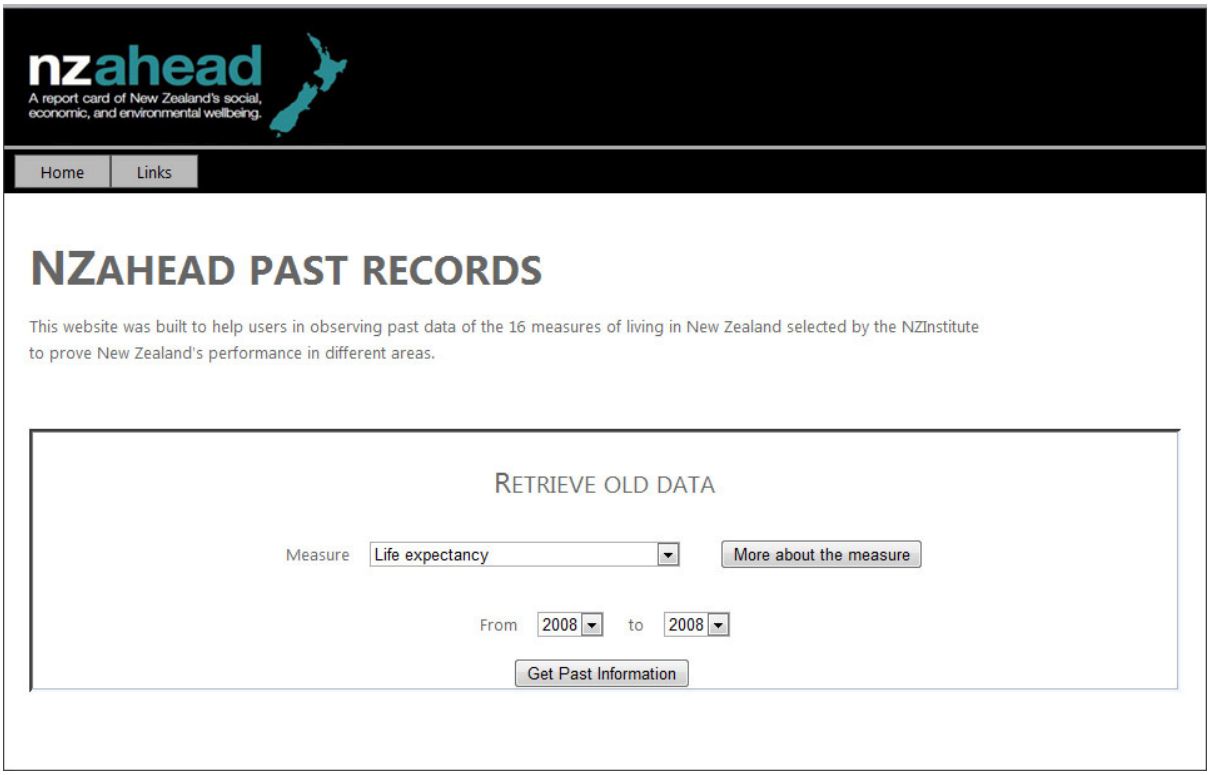
As seen above, a basic formatted website is already given in an ‘empty’ project. A header file is inserted in place of the title, and some colours have been changed accordingly.

After a query is executed, the results of the query are bound to the controls of the corresponding page, so each element of data retrieved are shown on the preset control.

Frontend design for NZInstitute

The user interface of NZInstitute's website is only capable of retrieving data from the database, and the underlying processing will be done separately. Due to this reason, the interface will be kept to a minimal, and the functionalities will be dealt by the users in an interesting manner.

Main page

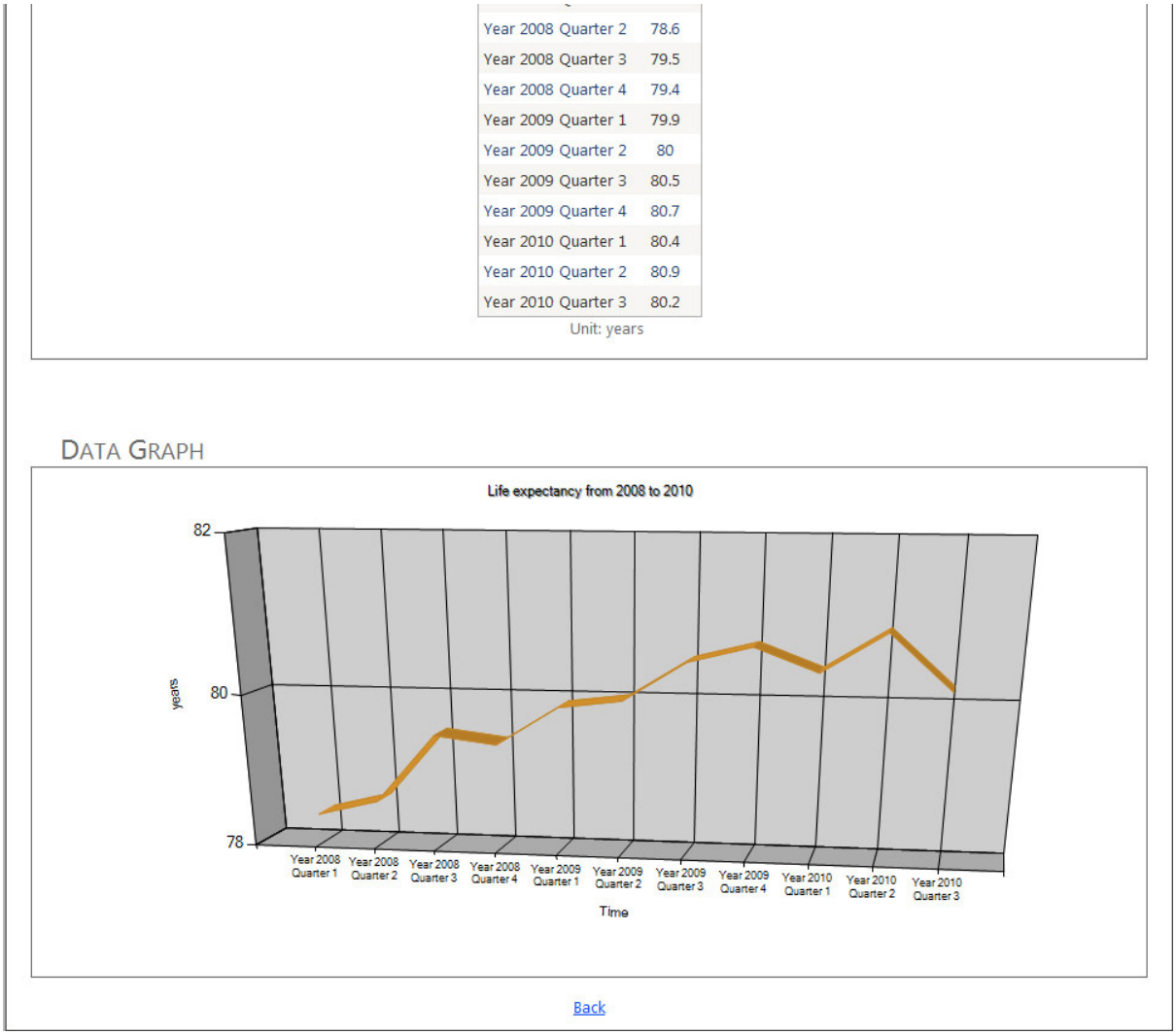


The picture above shows front page for the website. The CSS file has been touched from the default interface of an ASP project in VS2010 so the colour schemes match the scheme of NZahead's website. The colour selection was done via RGB values. To match the theme of the website properly, the three colour values were set to the same values (e.g. 4A4A4A, same red, green and blue) so the website is a monochrome theme from black-white.

Since it only requires an interface to retrieve data from one type of measure, the above example (or similar) is sufficient enough to do its job. Keeping things simple is always the safe option.

When the user selects the measure and clicks the 'more about' button, the user will be linked to the NZahead website measure page of the user's choice. When the 'Get Past Information' button is clicked, the user will be linked to the next page with the table and graph of past data within the time range will be shown.

Graph example

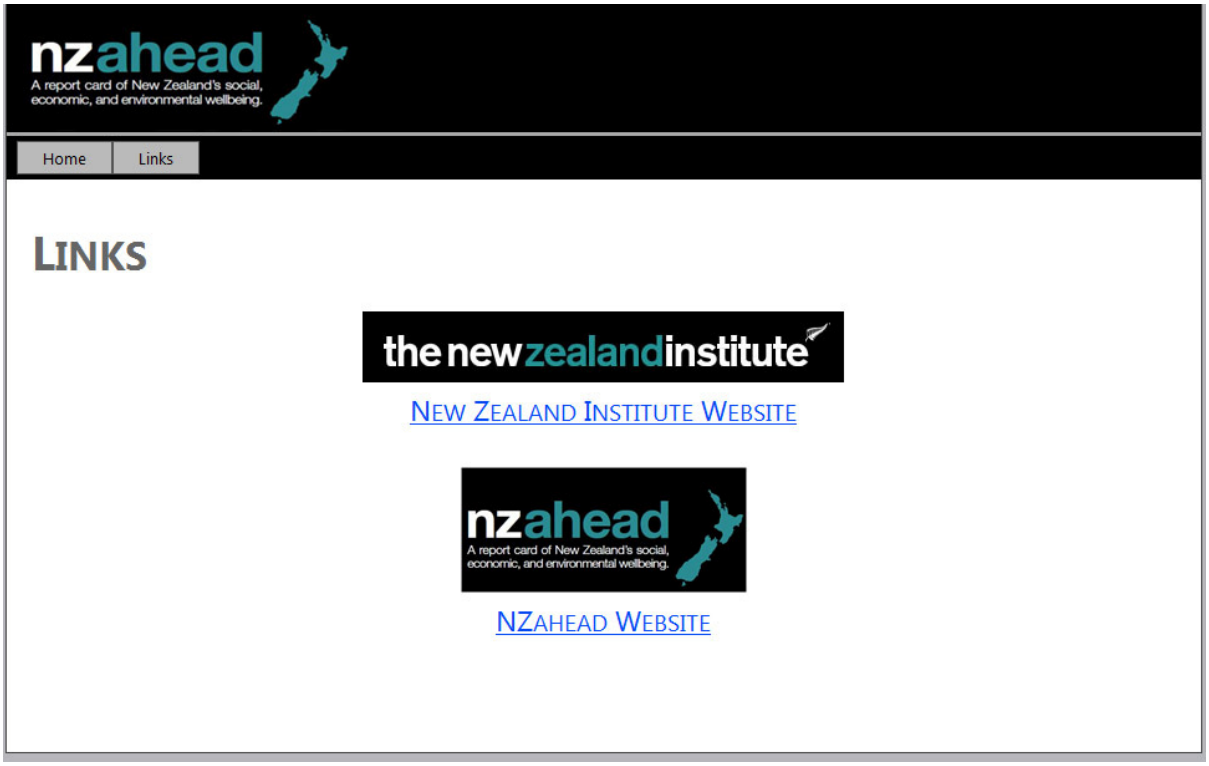


The initial plan at the beginning of the project was to include a silverlight application into the website of NZahead so the user could view a graph that shows the changes over the time period the user has selected. However, the requirement gathering and implementation of the ISSG website took longer than expected, and time to learn and implement a silverlight application was not reasonable.

To replace this, the chart control of ASP.net has been chosen. It is a new control that has been introduced with the release of ASP version 4. Due to this reason, there are not many examples that show how to implement this on the web.

The graph consists of a series, a chart area and title. The series is the actual data and the plot that results from the data. Colour and values bound are set with the series. The chart area is to do with the axis and the presentation of the field where the plot will be placed on. One option that is the most obvious is the 3d element of the graph, which consists of options such as the angle of tilt and the shadows to be included in the area. The title is the title of the area. The font and text that should go in the title of the graph is to be set with this attribute.

Links page



The website made by me is around the NZahead website which was launched March 2010. Due to this reason, the website needs to keep a strong relationship with the NZahead website (i.e. easily accessible) so it is easy for the user to move back and forward the two websites.

On the master page where the page links are made, a 'Links' button was added, so the user can access the links page anywhere the user might be.

Inside the page, links to the NZinstitute website and NZahead websites exist, in the forms of both a hyperlink and an imagelink so the clicks are easy for the user.

Frontend design ISSG

The website design for ISSG will be a lot more complicated, and more variables have to be taken into account. The design will have to follow the database structure, so the user can focus on the 'main' types of queries; i.e. data about the islands and species.

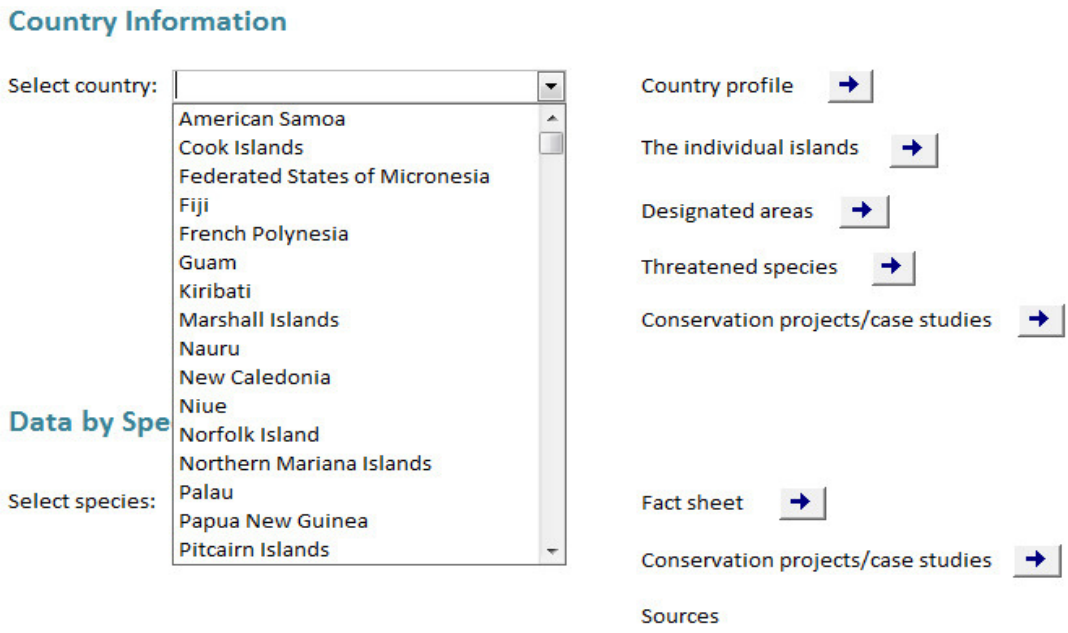
An example website is the red list, displaying information about species

http://www.iucnredlist.org/

Using the sample database query system that was provided and regular meetings with ISSG, the website was transformed bit by bit over five versions (more to come).

V1 –

The first version implemented to get a basic understanding of how ASP.net works. A simple version of the database was created, and a mini version of the sample query system provided was implemented.



This picture shows the screenshot of the interface implemented on the access database. Lots of entries and options need to be included in the interface, so keeping it simple while having all the functionalities is the key to the design.

V2 –

Second version developed with session variables, and now the classes inside the projects are now not needed and therefore removed from the project. The basic fundamentals such as binding data to controls and query execution are now working to some extent.

V3 –

Since the fundamentals are learnt and the basic idea of how to develop the website is understood, the real database that will be used is now included in the implementation. The main page functionalities are implemented.

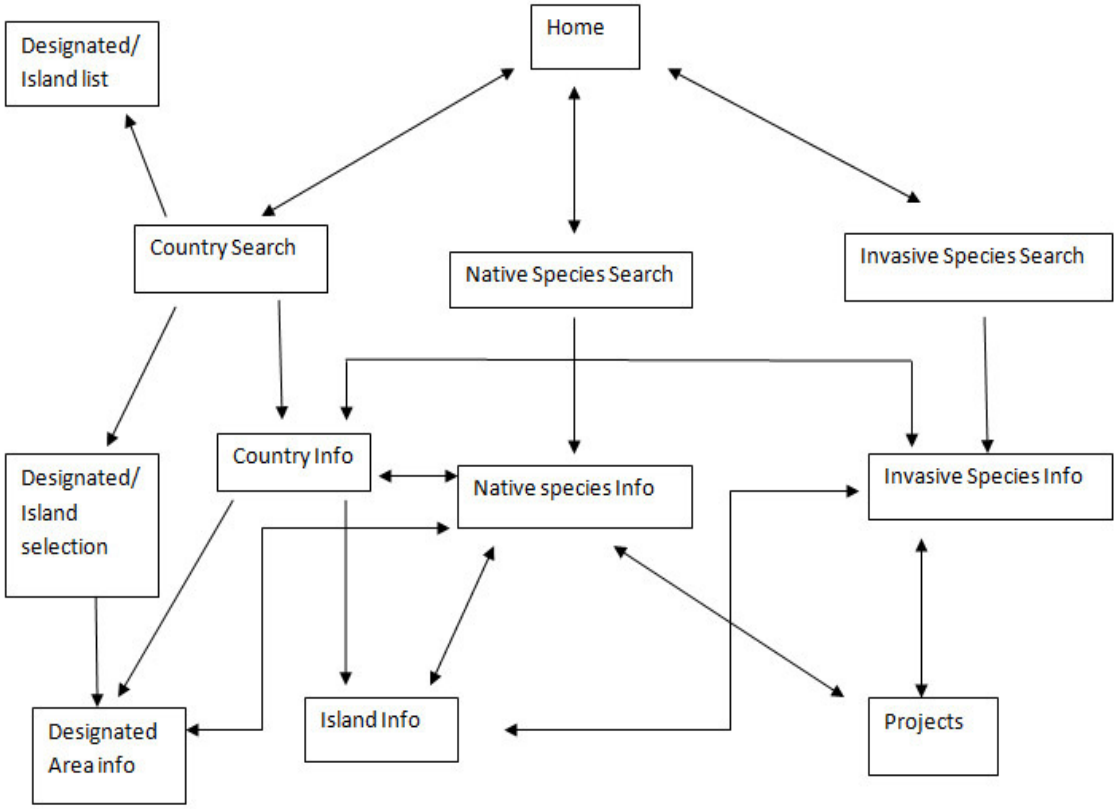
V4 –

Final version of the website with the same format as the sample query system has been completed. All functions and layouts of the pages are set, ready for use.

V5 –

The modified version is implemented, to suit the needs of ISSG more clearly. The user interface is thought of, and fully working. The working copy has now been handed over to ISSG for testing and waiting on feedback.

Website structure

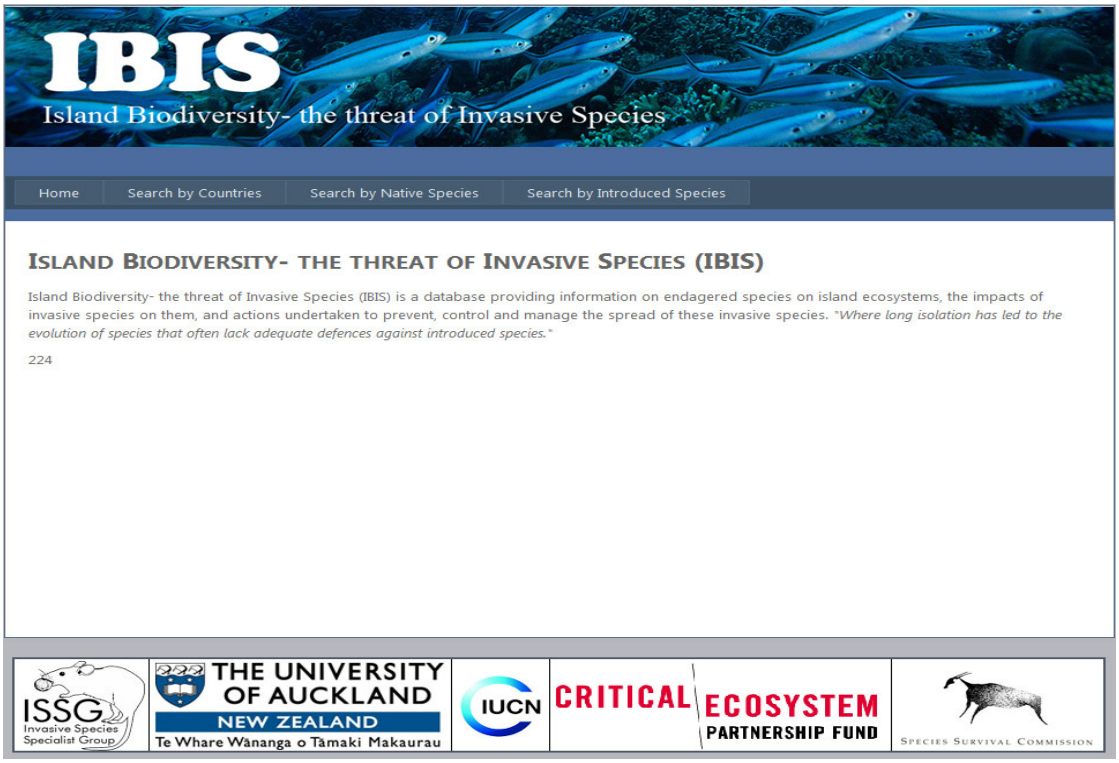


As the diagram above suggests, the website is a rather complex design. Due to this, a long time was spent on designing the website structure, so the user can follow through the website without much trouble. The design was formed through testing after the initial example format was converted to ASP format, and after the meeting was held to find out the exact requirements.

Screenshots

Show below are the main pages of the website, which are accessible from anywhere of the project. They have been designed for the functionality they require, and show only the functionality for simplicity reasons.

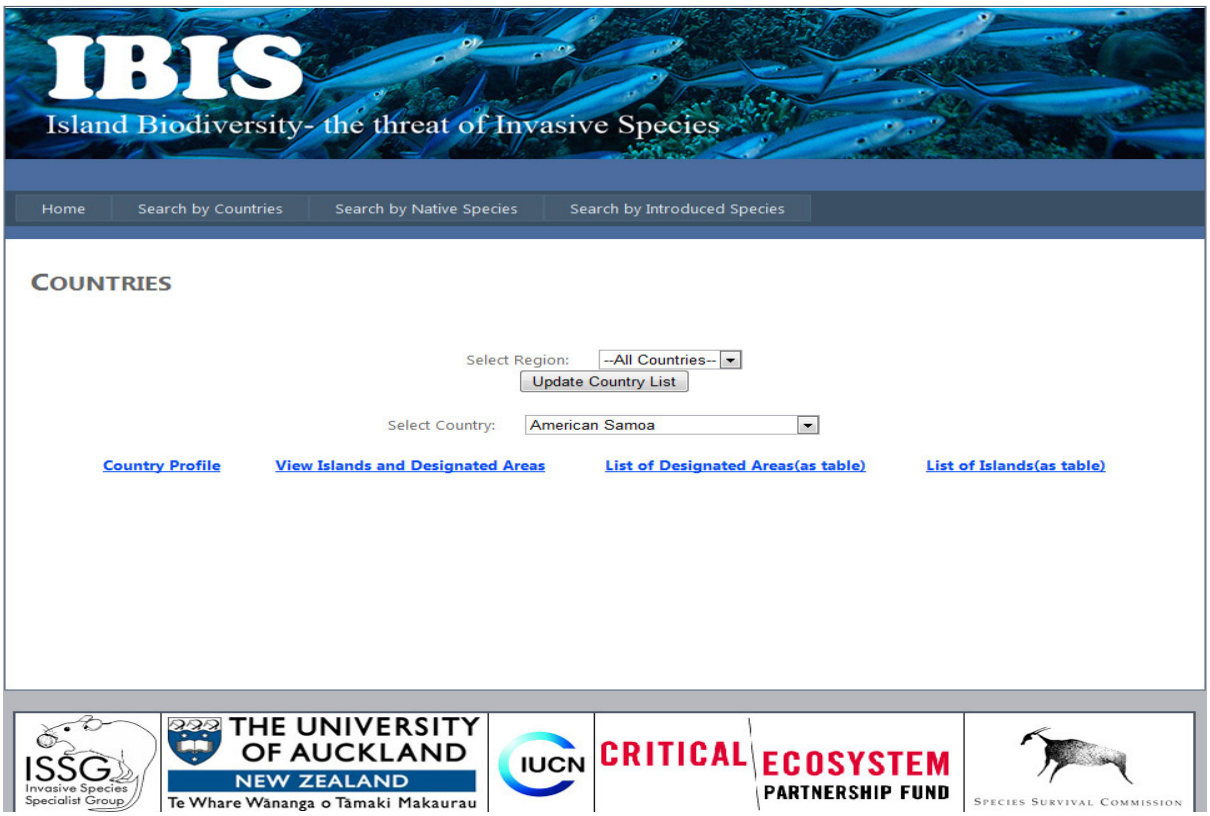
Home page



Screenshot of home page of the website

The four links that can be taken in any page of the project is the homepage, which is the page shown above, the Countries search page, Native Species search page and the Introduced Species page. The home page shown above is the default page, which is the starting point of the website. The three other links provide the search functions of the main aspects of the database. On the bottom, there is the image of the related organisations of the project, most of who are involved in the gathering of data, and hosting the website.

Country Page

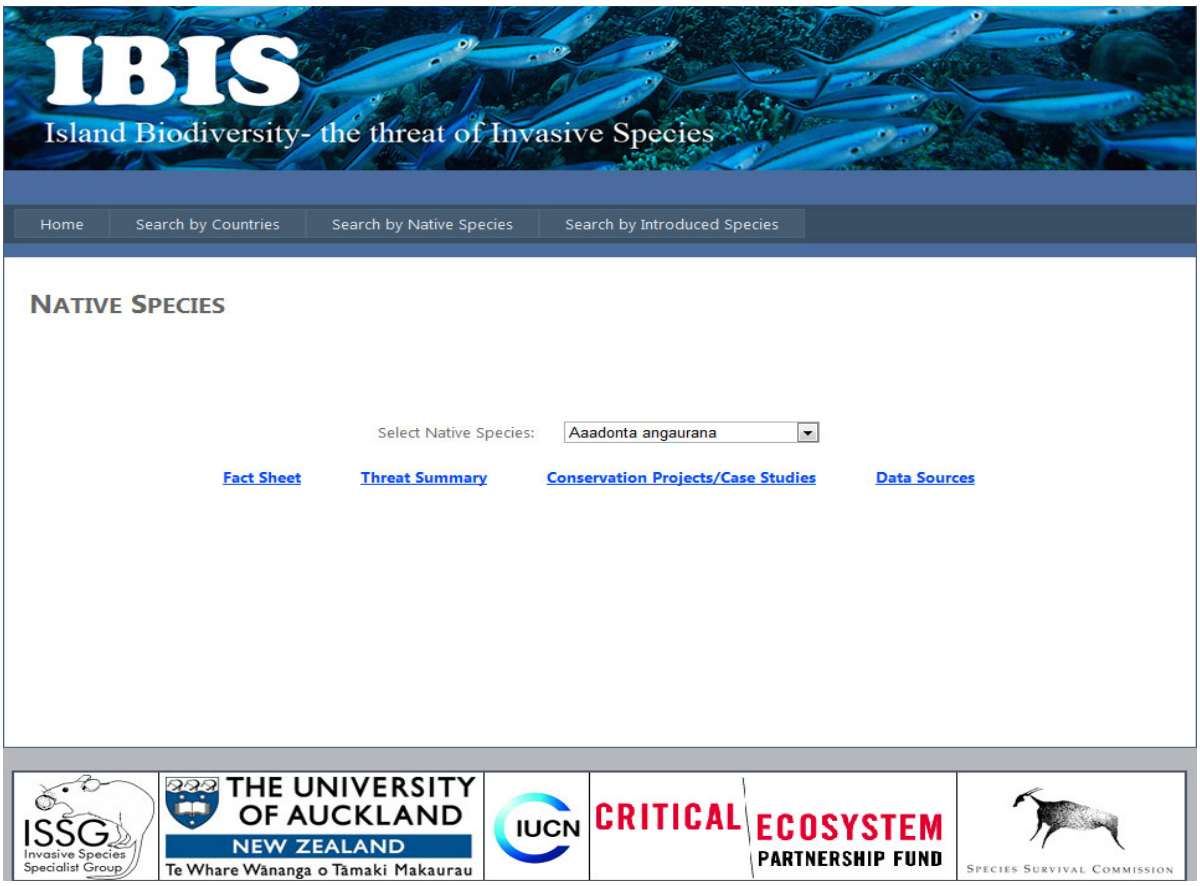


The country search page provides the interface to search for different functions. Initially the lower drop down box shows all the countries in the database. When the user changes the top drop down box (the region) and presses the update button, the page will refresh with the lower drop down list now showing only the countries within the region that the user has selected. A button pressed event has been selected over an index changed event due to the reason that the index changed method can over run the server very easily.

The page also has 4 links on them. One shows the country information, second shows the islands and designated areas in them, and the other two links show the table of islands and designated areas. The pages with the tables have the functionality for the user to download a csv file containing the information on the tables.

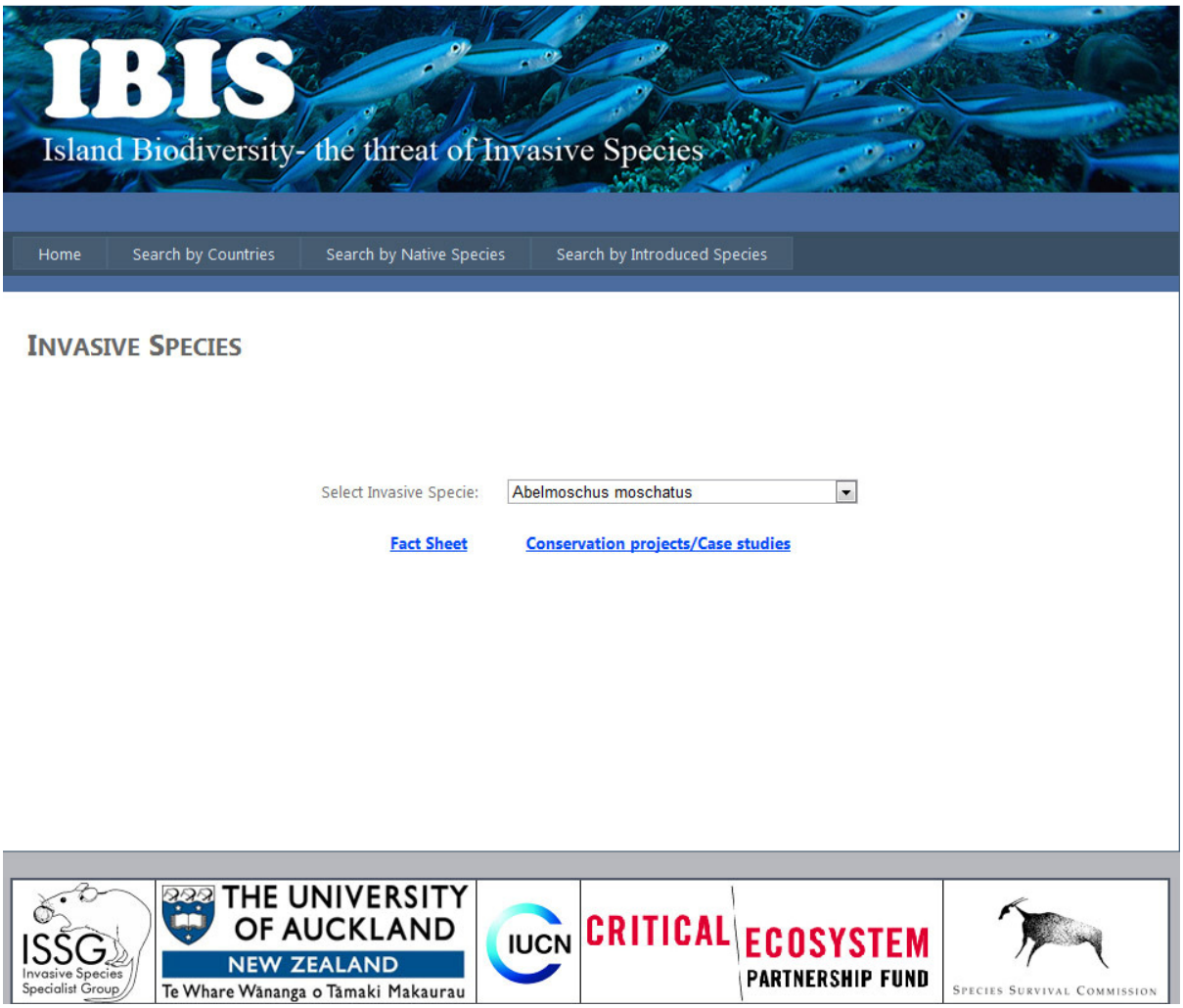
When the user navigates to the country, island or designated areas page, there is a drop down list that shows the list of species that live on them. If the specie information is viewed through these pages, extra information about the species specific to the area will also be shown.

Native species page



The structure of the native species is much simpler. It contains the four links to the pages on the information of the species.

Invasive species page



This page is even simpler than the native species page. The reason for this is there is more information relating to the native species due to the fact that this database is created to help maintain the habitat of native species.

Other techniques used

Other than just implementing the websites, many other factors had to be looked upon so the website is not only working, but ready for use in the public.

Parameterised queries

A common attack done to the dynamic websites with a database backend is sql injection. The user can 'attack' the database by inserting a sql command that will cause a table of the database to drop, or delete entries in the database.

A method to prevent this is to use parameterised queries. Instead of having a simple string as a query, parameters can be inserted into the queries so the queries that do not require parameters are blocked before execution. An example is shown below.

```
String query = "SELECT IslandName AS Island_Name, IslandType AS Island_Type, LatitudeDec AS
Latitude_in_Decimals, LongitudeDec AS Longitude_in_Decimals, LatitudeDeg AS Latitude_in_Degrees,
LongitudeDeg AS Longitude_in_Degrees FROM IslandInfo WHERE CountryName =?";

OleDbCommand cmd = new OleDbCommand();
cmd.Connection = conn;
cmd.CommandText = query;
cmd.Parameters.AddWithValue("@CountryName", temp)
```

The query here takes one parameter. With the statement Parameters.AddWithValue("@CountryName", temp), we can see the value of string temp inserted in place of the ? of the query. For the other queries, it is possible to insert more than one parameter in the query. The first parameter of the AddWithValue command needs to clearly specify where the parameter is inserted.

There is also another method to prevent sql injection. It is to have an array or words that are not allowed in the query/user input, and check the sql command before execution.

```
        Private blackList As String() = {"--", ";--", ";", "/*", "*/",
"@@", _
                                         "@", "char", "nchar", "varchar",
"nvarchar", "alter", _
                                         "begin", "cast", "create",
"cursor", "declare", "delete", _
                                         "drop", "end", "exec", "execute",
"fetch", "insert", _
                                         "kill", "open", "select", "sys",
"sysobjects", "syscolumns", _
                                         "table", "update"}
```

This will mean a query string will be checked with this array every time a query is run. Due to performance measures, parameterised queries were selected instead.

URL rewriting

Due to the nature of ASP websites, different pages show the page URL on the browser by default. This can cause problems in the projects, as some of the pages require one or more session variables to dynamically render the page. If the user navigates to a page via typing in the URL address, problems will be faced as the page does not have session variables to look at.

This is not a 'major' problem as ASP uses POST to transfer information between pages. The user cannot fraud the page by hacking this.

But to prevent it in the first place, URL rewriting can be implemented so the user does not know the URLs of the pages, just the home page is known.

URL rewriting 'tricks' the browser to think that the URL of the current page is not what it actually is.

For example, without URL rewriting the browser will show

http://www.samplehost.com/IBIS/default.aspx

but with URL rewriting, the last bits of the URL can be hidden as such.

http://www.samplehost.com/IBIS

This way, the user initially does not know the URL's of the inner pages, so cannot access it by typing in the URLs.

Note: The URLs given above are examples, and do not link to a proper page.

Common Elements

The most important part of the project was to come up with a superclass that could be used in all dynamic websites. ASP.net provides a nice interface so the data binding and page creation is easy to follow. However not all databases are tailored for use as a website backend. When the developer has to do most of the implementation via code, it can be time consuming. A few ideas have come up to create a superclass with elements that could be used in different pages and projects.

Clear all session variables

The way C# code gets the information (parameters) for queries is through session variables. If the variables are not managed in a clear way, they may overlap between some pages, and cause problems in rendering the page.

On loading some pages, the session variables might need to be cleared, and if lots of variables are used, it can get hard to keep track of all of them. It will be a good idea in some cases to clear all of the session variables. This can be done by getting the count of the variables that are stored in the session and running a loop to remove all of them (the session variable can be accessed by name or by int value).

Hide controls

When the query is unsuccessful, the page might not have to display all the controls in them. Again using the same idea of clearing the variables, the count of a certain control can be looked at and then removed.

This can be replaced with a custom ‘error’ page, so this may not be of use but the grouping of controls, partial queries can work.

Other than the stated, each layer (of 3 tier model) has been looked at to find the common elements. The database is made away from the website development, so is not really relevant, and the connection string can be save on the web.config file.

The middleware is ‘hard coding’, depending on the needs of the user, so the management of variables and controls is the key.

The frontend uses the master page for consistency among the different pages of the website. Given a basic master page on the creation of an ASP webpage project, the work that can be factored out is minimal. Once the master page is created, individual controls are placed on each page, which cannot be automised.

Problems (technical) encountered

Learning a new principle of coding and development can be a confusing task. Throughout the year, many (little) problems were faced while developing. Being the first time where the learning process was conducted by me, there were problems with the technical side that I could not understand why it was happening.

CSS changes not showing up straight away

During development, the frontend design was mostly touched on the CSS file. However, the problem with stylesheets was that the changes according to the CSS change did not show up immediately when the project was run. What was more confusing was that the preview of Visual Studio showed the changes right away. Changes to the browser cache were done so caching was turned off, but this did not make a change. A certain amount of time had to pass for the changes to show up on the browser, so the browser design was ignored and only the preview on Visual Studio was looked at to insure the colours show as expected.

Browser communication problems with non-IE and Visual Studio

Visual Studio and ASP.net are Microsoft technologies. Due to this reason, Visual Studio runs the best with Internet Explorer. However not knowing this, the default browser was set to Mozilla Firefox, and problems occurred between Visual Studio and Firefox. Opening and closing of test runs caused problems. Currently, the default browser on the computer is set to IE, so Visual Studio runs the project on IE. For all other purposes, Google Chrome or Mozilla Firefox is used due to webpage rendering speed reasons.

Button/linkbutton causing the page load method to run

When a button or link button is pressed, the page_load method is run first, then the event handler for the button pressed event fires. This is to do with page life cycles. Between the times the user first loads the page and the time user clicks on a button, changes can be made to the database, causing changes in the pages. If an automatic binding is implemented on the controls, the controls will not refresh because there is no C# code with binding. However in the case of IBIS website, manual binding is implemented so the binding is re-done on button click, changing the index of the user selection and causing the page to malfunction.

To get around this, the pages with dynamic binding has a check in them to see if the controls already have data bound to them and if they do, do not re-bind.

More on page life cycles on:

http://msdn.microsoft.com/en-us/library/ms178472.aspx

Asp4 free host not existent

It was only earlier in 2010 when ASP.net 4.0 was released. Due to this, many public servers that host websites and projects for free do not host ASP 4. Before ASP 4 was officially launched and in beta testing, MSDN provided a testing server for the developers. But after the official release, the test servers were closed and no other servers remained. Because of the lack of servers for testing, only screenshots could be given to the clients at one point in time.

Chart application library

Another problem that had arisen due to the fact that ASP 4 is new was that the example for the graph control was not readily available online, especially the example done with C#. The implementation had to be done with the example on the ASP official website. Although it was hard to find resources and develop, it was a good experience implementing something with not much examples.

Problems (non-technical) encountered

Also, being the first time programming in a business environment (outside the academic programming square), there were a few problems that I have encountered outside of the technical square.

Time management

Given a definite due date, the programming done in assignments in the previous years gave a clear outline of the time management, and the management was done (in a certain way) by the due date. However the project being a yearlong project, it was hard to manage time and to estimate how long the development will be.

Communication

An important factor in the implementation was the communications issue between me the implementer and the organisations, so the testing could be carried out efficiently. Because it is not clear when the reply will come and when I will be ready for the next meeting.

Due to the above reasons, although it may seem obvious, work needs to be done as soon as possible to avoid confusion and problems.

Future Work

Due to the problems encountered, the projects at the two organisations are not quite finished. Some work needs to be done in the projects to insure that the websites are ready to be deployed and working in the internet.

Database conversion

Since getting the website working was the first step to take, the databases of the projects are in access format, not yet converted to MS SQL. The converted version of the database should be used so the security problems are lessened.

Final implementation

The final version to be published is not yet confirmed by the organisations. The final changes to be made are still pending from the organisations.

Achievements

A numerous number of achievements have been earned through the project. One the technical side and on the business side, the project has given me good experience throughout.

Work experience

Being in an internship through the project was the first time I was involved in an employer-employee relationship. So it has been the first time I could have the chance to take responsibility in my work, and the first time someone depended on my implementation and my work. Since in the workforce this will be a procedure I will be involved in numerous times, the experience was priceless.

Organisation, management skills

Because it is not just about me implementing, it was important that I organised meetings, organised the project code for demonstration as such. Versions of the project had to be kept safe and in an organised way, and backed up regularly to avoid any problems.

Development skills

Although a bit of experience in programming was gained over the four years in university, it was the first time dealing with developing a dynamic website. Learning a new concept on my own was also a new experience.

Solution creation

Over the undergraduate years, most of the programming done was under circumstances where the exact requirement was given, sometime to fine details. However in the project, the requirement was given at a business logic level and the rest had to be developed by me, based on the knowledge gained throughout university study.

Progress log

5th March - first meeting with client

8th March - first weekly meeting for NZinstitute attended

9th March - first meeting with Mano

11th March - first project idea discussed

15th March - Draft idea of database of past data on NZahead introduced

22th March - project proposal formed

23th March - Second project for ISSG received

25th March - Introductory seminar, project with ISSG accepted.

26th March - Started to learn about ASP.net

4th April - database design for NZahead started

8th April - database design for NZahead finalised

9th April - First meeting with ISSG, Learnt database is in progress.

22th April - Been busy with Assignments, but still learning ASP.net. Draft ideas of website designs coming out.

7th May - Draft paper prototypes of website designs are being implemented.

18th May - First 'technical' meeting with ISSG. First database received.

4th June - Meeting with ISSG about database

1st July - Minor changes to the database structure

9th July - First soft copy of the project started, using draft access DB.

16th July - First draft using real data started to be implemented (v3)

30th July - The structure of the website to change a little, including the interface cleanup. (v4)

13th August - ISSG has talked about a need for a completely new structure of the website. Rebuild (v5)

2nd September - Version 5 of the website handed in. Waiting for feedback. In the mean time, Version 1 of NZahead website being developed.

19th September - First draft of the NZinst website handed in for testing, and awaiting responses.

10th October - Decision made to remove silverlight and keep with charting controls on ASP 4.

11th October - Preparation for final report/presentation.

20th October - Final presentation.

Miscellany work

Other than the project itself, I have been working at the NZInstitute as an intern helping with various miscellany works. Some of these include helping the institute in making suggestions about their web pages (they have another website launching soon), and finding gadgets they may be of good use in their websites like NZahead.

Modification of an existing gadget was part of the job. The example can be viewed on

http://www.nzinstitute.org/index.php/nzahead/measures/from_the_new_zealand_institute/

under "New Zealand's performance". The graph was modified from the motion chart of Google, to remove sidebars and unnecessary option tools.

Summary

The project progress has been slow, partly because the formation and acceptance of the projects being rather late, and partly due to failed time management in the earlier half of the year. However it was realised that lots of testing and implementation had to be done in order to get the implementation up to the standards of the organisations.

The second half of the project was all about implementing. Development and checking was done continuously to find the optimum design, and now heading towards completion.

Lots of lessons were learnt in various aspects. Learning how to learn for myself and developing in various situations and environments was one of the most difficult but memorable achievement and experience.

References

ASP
http://www.asp.net

PHP ASP comparison
http://www.me-u.com/php-asp/phpvsasp.htm

Access DB with C#
http://www.csharphelp.com/2006/01/ms-access-application-with-c/

Publishing websites
http://www.codeproject.com/KB/aspnet/IIS7ASPNet.aspx

Page lifecycles
http://msdn.microsoft.com/en-us/library/ms178472.aspx

Charting in VB
http://www.4guysfromrolla.com/articles/072209-1.aspx

Custom HTTP responces
http://msdn.microsoft.com/en-us/library/system.web.httpresponse.aspx

Acknowledgements

To conclude, I would like to thank a number of people who have helped me throughout the project.

First is the project co-ordinator and my academic mentor S. Manoharan. He has helped me and made lots of suggestions on the technical aspect, and answered all my questions in a constructive way.

Secondly, I would also like to thank Angela Chang for also helping me on the technical aspects. She was always willing to help, and thanks to this I have overcome many obstacles.

I would also like to thank Danielle Boven my industrial mentor. She has been patient with me all along while I get used to the working environment, and gave me suggestions when I was stuck with the designs. I would also like to thank all the staff at NZinstitute for making me comfortable.

And last but not least, Shyama and Sally at ISSG. They have been patient with me all along even when I could not meet some deadlines, and made me comfortable as possible in meetings.