Implementation and Use of Rapid Development Kit for Database Driven Internet Applications

Svend Madsen

Kgs. Lyngby 2004

Technical University of Denmark
Informatics and Mathematical Modeling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS

Abstract

The purpose of this thesis is to describe the development of a Rapid Development Kit (RDK), which can assist developers in creating advanced database driven Internet applications. This can be done without the need for the developer to have knowledge in database development. This will decrease the number of layers/tiers which the developer needs to focus on from four to two.

Properties such as form validation, user management, picture upload, security and error are should be implemented as well. The RDK will support black box reuse, instead of the very often used white box reuse approach, by copying and pasting code.

The “religion” of choosing the right development environment Microsoft vs. Linux will also be discussed, and the pros and cons of each environment will be highlighted.

There are several systems on the market that can assist developers in creating advanced Internet web applications. The decision process of using Open Source, commercial or in-house development of an RDK, will also be discussed in this thesis.

Preface

This Master of Science Thesis presents the results of my work during the last 4 years. The thesis is the final part of my studies for the Master of Science degree at the Technical University of Denmark, DTU.

The writing of this thesis has been supervised by associate professor Hans Bruun and associate professor Jens Thyge Kristensen from IMM, DTU. Whom I would like to thank for comments, feedback and support during the writing of this thesis.

April 31st, 2004

Svend Madsen

Table of contents

1 Introduction

During my time as a self employed and free agent I have developed several advanced websites and intranets. The experience soon made me realize that there were several features that were used over and over again.

Many of the data models were also very similar; they often contained some user information such as usernames, passwords, name, e-mail and so on. The different groups of data are often represented in one main table such as an item for an e-shop and then some related tables containing different properties for this item, such as a table with colours. When developing such similar websites I thought that it would be nice to make a RDK that contained as much of these general properties as possible. And it should still be extremely flexible and not restrict the development to only the features supported by the RDK.

Normally advanced Internet applications are developed around a database and developers design new data models for every new application. The time and cost constraint often result in the data model being inflexible, and every time there is a change to the data model, the developers need to edit all the layers/tiers on top of the database to cohere with the new data model.

The changes needed often include the table design itself, Stored Procedures or SQL queries, the code for representing the data such as ASP or PHP, and finally the front-end design of the application. The process of doing these changes is extremely time consuming and the error rate is often very high. The testing process can be at the same level as it was for the original test of the whole system.

Skilled database developers can be hard to find, and by using programmers with little database experience, the final data model might be extremely difficult to maintain, and the performance is often critical.

Some properties that could be desirable in such a development kit are the ability to handle multilingual websites, as well as Unicode character sets.

1.1 How is this masters thesis structured

In this master's thesis I will give an introduction to Dynamic Websites in chapter 2.

The most common technologies used for web development today will be discussed in chapter 3, Choice of technologies.

Usually the development of an advanced dynamic website is under strong time and cost constraint. This will also be discussed in chapter 4, Development time and costs.

The most Common Features of web development are discussed in chapter 5.

A technical oriented Developer Manual aimed at the developer using the RDK is described in chapter 6.

The Tutorial in chapter 7 can assist the developer in using the RDK. This tutorial is guiding the developer from start to end in making a website for publishing food recipes.

A technical description of the RDK, called Practical implementation is placed in chapter 8.

In chapter 9, Examples of use, some of the systems already made with the RDK will be shortly described.

I will round up my experiences using the RDK in the Conclusion chapter 10.

1.2 How to read this masters thesis

The prerequisites for reading this masters thesis varies in the different chapters.

The first chapters up to and including chapter 5, can be read and understood by everybody with an interest in web development.

It is preferable to be familiar with Microsoft Active Server Pages, in order to understand chapter 6 and 7.

In chapter 8 it is necessary to have a good knowledge of database modelling in order to understand it to the fullest.

In code samples you will sometimes find parts marked in bold. The bold part is referenced in the text either before or after the code sample, and when referencing to the code, the text to find is marked with double quotes.

Some code samples are also inserted as figures, these samples are used to give the reader an insight in where in the code we are editing, without the need of looking through the original file.

Coloured arrows and boxes are added to some of the figures. These arrows and boxes are made to reefer to certain parts of the figures in the text.

References to other material are made in hard brackets, for example [1]. You can lookup the number in the chapter References to find the source of this material.

After the chapter References you can find a list of abbreviations. This is used as a help for the reader so he/she does not need to find the place in the text where the abbreviation is used for the first time in the thesis.

2 Dynamic Websites

The term dynamic website or page is in short the description of a web page being created on-the-fly, instead of static pages being served as is from the server to the client's browser[1]. The generation of a dynamic page can be achieved in many ways. There are lots of different technologies for providing dynamic content and they can both reside on the server and the client.

The reasons for providing dynamic websites can be diverse. It is important to catch your visitor's attention, or else the visitor will leave your site and never come back. If you want your visitor to revisit your site over and over again "something new needs to have happened" to your website since the user visited the website the last time.

One of the keys to get a successful site is to provide up-to-date dynamic content. This can be that your website is integrated with one or more backend systems so users can get up to date inventory lists and order statuses via the World Wide Web. A dynamic website can also be used to provide the user with a personalized experience so his or her settings are saved from visit to visit. Commercials can be represented depending on the user's interests and what has been bought earlier. Of course the website developers must be careful in the way the collected data are used so the user will not feel stalked. If you for example know the weight of your user, I don't think he or she likes to be introduced to commercials on how to loss weight every time he visits, but if you know your user is skinny, you can use your advertising space on other stuff than dieting products.

[1] A browser is an application that makes it possible to look at and interact with the information on the World Wide Web.

Usually the content for a dynamic website is situated in a database; this can be in form of text files, XML[2], or a relational database system like the database from Oracle[3], MySQL[4] or MSSQL[5].

2.1 Advantages of dynamic content

The reason why dynamic websites in the beginning were relatively uncommon was that it often cost more than the double of having a static website created, and many of the technologies for creating dynamic website were in their infant state.

If we take the example of an e-shop, with lets say 200 products, this would, on a static website, demand at least 200 separate static HTML[6] pages, one for each product. So every time a product changes or a new has to be added, a whole page has to be edited or added and often new references to the new page have to be made. This is often more than a normal physical store owner can handle. Due to this fact a lot of the first e-shops made were rarely updated, and didn't represent an up to date stock hold of the physical store.

[2] XML (Extensible Markup Language) is a way to create common information formats which makes it easier to share data on the World Wide Web, intranets, and elsewhere.

[3] Oracle is one of the world's leading suppliers of software for information management.

[4] MySQL is a popular open source relational database management system.

[5] MSSQL is the professional relational database management system provided by Microsoft.

[6] HTML (HyperText Markup Language) is the set of markup symbols or codes in a document intended for display on the World Wide Web.

Another problem with static web pages is that the content provider and the website developer is very seldom the same person. So the content provider needs to be taught how to edit and upload HTML pages, unless the company want to pay a HTML developer to copy / paste new content into the HTML pages, every time the content of the website needs to be updated.

It will also be a real pain to update the layout of this e-shop, imagine manually replacing the layout on the 200 product pages of your e-shop, and separating the content from the layout on each page.

All these problems can be overcome by creating a dynamic website. The “dynamic” part of a website can both be situated on the web-server and on the client, but due to browser incompatibility and the need to make an extra server request or page reload while fetching data, most developers decide to have most of the dynamic code executed server side.

Some will argue; keep away from the dynamic web pages where possible. They are less scalable, so every time a page is requested on the server it demands some processing time from the server, where as a static page only needs to be served to the client.

In my point of view the separation between content and layout, and the ability to have the content provider updating the website without programming skills is, in a professional environment, preferable due to the relatively low cost of server hardware, and the expensive cost of having a programmer updating the content at all times. Additionally the risk of making errors while updating the website is much less with at dynamic site, where input is typed into a form with automatic validation of the content, versus the risk of making errors when editing the HTML files manually.

2.2 Creating dynamic websites

When creating your dynamic website it is important to carefully consider, what functionality should go where, many often decide to leave stuff such as form validation to be handled by the client. This gives fast response time, because you do not need a server request to check if the information in the form was typed in correctly. The big hassle in leaving the clients browser to validate your form is that some validation functions might need to be written to the specific browser version or browser maker. So you need to check which browser is running on the client machine, and then choose the corresponding code to execute. The result of this approach is unfortunately that parts of your client side executed code need to be written specifically to all the browsers that the developer decides to support. The testing of this code is also very cumbersome. Many companies do not want to spend the money on testing and developing the website for a browser that maybe only 1% of the users have installed on their computer.

The client side executed code can be written in many languages such as JavaScript[7], VBScript[8] and Java applets[9]. In many occasions you will never use the client to handle and sort out data requested from the server. There is no need to send 1000 rows in a table to the client, when it may only need 10 of those. When the data need to be transferred over the Internet, the amount of data sent in this approach is unnecessary and time consuming.

[7] JavaScript is an interpreted script language originally implemented by Netscape.

[8] VBScript is an interpreted script language from Microsoft.

[9] An applet is a program that can be sent along with a Web page to be executed on the client's machine.

There are many ways and programming languages to make server side executed code, but common for them all, they provide a way to separate your content from your layout. Some of the most used ways of this separation are Server Side Includes (SSI[10]), Common Gateway Interface (CGI[11]), Active Server Pages (ASP[12]), Perl[13], Hypertext Preprocessor (PHP[14]) and Java Servlets[15]. Some of these technologies are later discussed in chapter 3 Choice of technologies.

The approach of creating the dynamic websites with these technologies can be diverse. One of the first approaches of providing dynamic content for the World Wide Web was done through CGI where you write your regular program and make sure to write your HTML code to the standard output. Maintaining the HTML code of these programs is not very easy and CGI is being used more and more rarely for new dynamic websites.

[10] SSI is a variable value or a reference to at file, that a server can include in an HTML file before it sends it to the requestor.

[11] CGI is a standard way for a Web server to pass a Web user's request to an application.

[12] ASP is a HTML page that includes one or more scripts that are processed on a Microsoft Web server before the page is sent to the user.

[13] Perl is a script programming language, and can be a good choice for developing CGI programs because it has good text manipulation facilities.

[14] PHP is a script language and interpreter that is freely available and used primarily on Linux Web servers.

[15] A Servlet is a small program that runs on a server.

You can also write placeholders into your regular HTML page, that when the page is requested makes sure to fill in the dynamic content of the page. This approach is often used with ASP and PHP. A big advance in this approach is that the designer can design the pages, and afterwards the program developer "just" fills in the application logic.

A more and more commonly used approach of creating dynamic content is the one using XML, which is the pure content and then a style sheet XSL[16] provides all the information on how to represent the data. The same effect can be obtained with ASP and Cascading Style Sheets (CSS[17]). The two last mentioned approaches are extremely effective in separating the content from the page layout.

2.3 References

The main references used for inspiration for this chapter are [1] and [2].

[16] XSL (Extensible Stylesheet Language) is a language for creating a style sheet that describes how data sent over the Web using XML is to be presented to the user.

[17] CSS is a style sheet style that describes how data sent over the Web using XML is to be presented to the user.

3 Choice of technologies

When deciding to build dynamic websites, we have to choose which technologies we want to use.

First of all I have to say that choosing which technologies to use is almost religious. Many have thousands of good points why the web technology they chose is the best and all the others are lousy. Often the arguments for a known programming language/technology are based on the little tweaks and the knowledge of the supported libraries. You would not know how to do this in the competing language. And the guy speaking against your favourite language will use the exact same arguments, just based on the knowledge of his preferred language.

I will try doing an objective comparison of the different technologies and afterwards give my own recommendations based on my own experience.

3.1 Choosing the database management system

The core of an advanced dynamic website is the database management system (DBMS). The best way to start out is to make your decision on which of the available technologies to use. Things that we need to take into account are availability. Can we find a hosting provider to host our DBMS, or do we want to host it our self? And do we want to pay the price of the more expensive solutions? What properties do we need to support in our website? Some of the properties we need to consider are the support of Stored Procedures[18], Transactions[19] and Unicode[20].

[18] A Stored Procedure is a set of SQL statements with an assigned name that's stored in compiled form in the database.

[19] A Transaction is a sequence of information exchange and related work that is treated as one unit for securing database integrity.

I have chosen to describe two different DBMS's in depth, MySQL and MSSQL. Other DBMS's that came into consideration was Oracle and MS Access, but these were quickly ruled out. Oracle has a very steep learning[21] curve; it is relatively expensive [24], [25] and is not supported by many hosting companies. MS Access has a very messy SQL[22] version, though it has a good visual representation of the queries. The resulting code is extremely hard to understand. If you do a query with more than 2 or 3 joins you need to spend a long time understanding the generated code. MS Access does not scale well, and is not intended for large professional multi-user databases [3]. Of course there are other DBMS's used for dynamic websites, but most of the ones used are MySQL and MSSQL.

[20] Unicode is a character set that supports most characters of the world's languages.

[21] Taken from recommendations on [3].

[22] SQL (Structured Query Language) is a programming language for getting information from and updating a database.

3.1.1 MySQL

MySQL is a very popular database system for web development. It is often used together with PHP[23] in Linux environments. MySQL is open source[24] but it lacks the support of Stored Procedures, Transactions and Unicode. MySQL is supported by most web hosting companies. MySQL can run on both Linux Servers and Windows servers.

The next version 4.1[25] of MySQL will support Unicode.

MySQL is free of charge for some purposes, please have a look at the license policy at [4].

[23] PHP is discussed later in this chapter.

[24] Open Source is a method and philosophy for software licensing and distribution designed to encourage use and improvement of software written by volunteers by ensuring that anyone can copy and modify it freely.

[25] In alpha release at the moment of writing.

3.1.2 MSSQL

The query language for the MSSQL is in opposition to MS Access very easy to understand, and you are able to write it without the need of visual aid even when you need multiple joins in your query. In addition to MySQL, MSSQL supports Unicode, Stored Procedures, Transactions, Views[26] and Triggers[27]. The downturn of MSSQL is that it only runs on Microsoft Windows servers. But you are able to connect from your Linux/PHP website to a separate server running MSSQL.

You are able to obtain a free version of MSSQL called MSDE. Please have a look at [5] for the license policy. And at [6] for the full version of MSSQL.

3.2 Choosing the server side language

To represent and change the data in the chosen DBMS, we need a server side programming or scripting language to make the connection to the DBMS, and produce the desired HTML code. Since it is the nature of server side scripting languages to be executed on the server, there is no need to think browser compatibility. It is possible to connect to the two described DBMS's from virtually every commonly used scripting or programming language. I have chosen to discuss 6 different server side programming or scripting languages. I need to stress again, that choosing of which language to use is very difficult. Many, if they have a free choice, will choose the language they are used to.

---

[26] In a DBMS, a view is a way of portraying information in the database. This can be done by arranging the data items in a specific order, by highlighting certain items, or by showing only certain items.

[27] A trigger is a set of SQL statements that automatically "fires off" an action when a specific operation, such as changing data in a table, occurs.

3.2.1 CGI/Perl

CGI is supported by most web servers. This allows you to communicate with the clients web browser in almost any programming language you like. One of the most commonly used programming languages through CGI is Perl. Perl is an Open Source project that has been running since late 90'Th. But the first version of Perl was released already in 1987[28].

Perl was designed as a scripting language to manipulate text easily. This was found well suited to do dynamic websites, when the first websites arrived on the scene. Perl was not designed for the Web, and that definitely leads to some drawbacks of using it for the programming for your dynamic website.

Since it is an Open Source project there is no formal support for it, but of course there are lots of forums and communities, that can offer you help. Perl tends to be pretty slow when it is running on a Microsoft server. And since it spawns a process for every single call from each client, it doesn't scale well either. Learning Perl can be pretty hard. Since it is fairly complex, you can write a piece of code in rather many ways and to times pretty non understandable for a third party.

[28] For further information on the history of Perl please have a look at [7].

3.2.2 ASP

ASP was first introduced by the name Denali, but in 1996 Microsoft changed its name to ASP as a version 3.0 and released it together with Microsoft's web server called Internet Information Server (IIS). In theory ASP is not a scripting language, but a framework where you can implement your scripting language of choice, for example VBScript or JavaScript. But it has been associated with VBScript since it is most commonly used with the ASP framework. ASP fast became popular, because it was much easier to learn than Perl. Especially for HTML developers without real programming experience that wanted to add some dynamic code to their HTML pages.

ASP is closely linked to the Microsoft Windows operating system, but you are able to execute ASP applications on other operating systems as well. For further information have a look at [8]. ASP is very extensible, though the default installation of ASP has not many components included. So if you need to use functionality such as file upload, emailing and others, you need to use a third party component or develop it yourself. You are able to develop these components in languages such as Visual Basic or C++, and there are a lot of components on the market. The downside is that most of it comes for a price.

3.2.3 PHP

PHP is a recursive acronym for Hypertext Preprocessor and is relatively new on the web development scene. It started to catch the web developer's attention in the late 90'Th. PHP has quickly become very popular, and is seen as a popular alternative to Microsoft's ASP. In opposition to ASP, PHP is free and Open Source and it supports almost every function desired such as dynamic graphics, file compression and PDF[29] file creation. If you want to add new functions to PHP, you need to be familiar with C/C++ programming language.

PHP is cross platform and is supported by all major web servers; this will make you independent of what web server your hosting provider has chosen to install. PHP is almost as easy to learn as the VBScript which also makes it a popular choice for new web developers.

3.2.4 ColdFusion

ColdFusion was developed by Allaire[30] in 1995 and focused on HTML developers wanting to make their website dynamic. So ColdFusion was made tag based just as HTML. Not to restrict more hardcore programmers you are able to write tags with included C/C++ or java code. This makes ColdFusion extremely easy to get started with for the non programmer, and still the experienced programmer is able to implement all the functionality he wants. Like ASP and .Net[31], ColdFusion is a commercial product but it still supports most popular platforms for web hosting, including IIS and Apache web servers.

---

[29] PDF (Portable Document Format) is a file format that has captured all the elements of a printed document.

[30] Allaire was later bought by Macromedia

[31] .Net is later described in this chapter.

3.2.5 Java/JSP

Java is unlike PHP and ASP a full-blown programming language which makes it quite difficult for beginners to learn. But once learned it is an extremely powerful tool. Java Server Pages (JSP) is in many ways just like ASP and PHP, but based on the Java syntax. On top of that you have the ability to add functionality with servlets, which is actual Java programs running on the server. With Java Enterprise Beans you also have the ability to write distributed programs. As well as with ASP you might need to buy components, for different functions, if you don't want to write it all yourself. Due to the complexity of the language, dynamic websites with Java might be quite time consuming to develop.

3.2.6 Microsoft .Net

The Microsoft .Net framework is the new kid on the block. As ASP it supports different languages like Visual Basic, C#[32] and J#[33]. This makes the .Net platform a tough competitor to the Java platform described above.

.Net is only supported by the Microsoft operating system, which is due to the close integration with the operating system.

.Net also offers language interoperability, so you are able to communicate seamlessly between applications written in different programming languages under the .Net framework. And the development environment Visual Studio .Net is for many developers a powerful development tool.

---

[32] C# or C-sharp is a new programming language from Microsoft, which aims to combine the computing power of C++ with the ease of Visual Basic.

[33] J# or Visual J# is a set of programming tools that allow developers to use the Java programming language to write applications that will run on Microsoft's .NET.

3.3 Conclusion on server side languages

From the descriptions of the server side languages, and the research I have made, I have completed a table (Table 1) characterizing the different technologies. The ratings are described below the table.

	Speed	Scalability	Operating system support	Learning curve	Support by hosting companies	Support	Maturity
Perl	%	%	+	%	+	-	+
ASP	+	+	-	+	+	+	+
.Net	++	++	-	+	-	+	-
PHP	+	+	++	+	+	+	+
ColdFusion	?	?	+	++	%	+	+
Java/JSP	+	++	+	%	+	+	+

Table 1 Comparison of the described server side scripting languages.

To the right you can se a description of the grades given in Table 1

%	Poor
-	Average
+	Good
++	Excellent

When all comes to all, the choice of technologies has something to do with what do your customers want. Some prefers to stick with the Microsoft technologies, and others prefer to use Open Source. There are many arguments of using both, and many reports have been in favour of both. So you need to think about what tool your developers are familiar with, and how much you want to spend in training, if you decide to introduce an unfamiliar server side language or DBMS. When discussing pricing of the different products, man tends to only compare the price of purchasing the necessary licenses of the chosen products. You really need to consider the expenses of the training of the development team, time to market and others to get the real cost of the chosen technologies as discussed in the next chapter (Chapter 4).

3.4 References

The main references used for inspiration in this chapter are [9], [10], [11], [12] and [26].

4 Development time and costs

There are many benefits of using a rapid development kit (RDK) for web application development, instead of building your application from scratch. There are a few exceptions where it might not be such at good idea; if your company only will develop one or two small web applications, then the learning curve and price of the RDK will exceed the cost of developing from scratch. But in general using a RDK will save you the costs of developing your applications from scratch. You rarely need the skilled developers and designers you need when starting from scratch, and the time to market will be shorter. The developers can focus on the business logic, and the testing is much easier when you have a robust RDK, because you do not need to test all the functionality embedded in the RDK. Using a RDK will often force your developers to build more consistent applications. The more web applications you build with one single RDK the lower the initial cost of purchasing and learning to use the RDK will be.

4.1 Choosing a RDK

It can be a hard choice to make, if you want to buy a commercial RDK, an open source RDK or develop the RDK yourself. You need to take a look at the preferred platform, the skills of your developers and the cost of the RDK to buy. Below in Table 2 is a decision matrix on what kind of RDK you want to base the development of your web applications on [13].

	Pros	Cons
Buy	1. Good documentation available. 2. Hardened. 3. High quality. 4. Good support available. 5. You save the time of building it yourself.	1. It's rare to find a RDK that meets all your requirements. 2. Many commercially available RDK's are too heavy and are overkill for many situations. 3. They often provide a steep learning curve. 4. Require vendor resources to educate developers in the use. 5. Depend on vendor for upgrades/new features etc. 6. Expensive. 7. Upgrades typically costly, increasing total cost of ownership.
Build	1. Build to your exact requirements. You get exactly what you need.	1. More time and high skill resources required to design, develop, test, support, and maintain the RDK. 2. Requires repeated use to harden and fine-tune. 3. Requires significant testing and quality assurance activities. 4. Documentation usually first to suffer if schedule slips. 5. Dependency on the developer(s)
Open source	1. Cheap. 2. Low total cost of ownership. 3. Typically reliable and stable because of large contributor and user base.	1. Typically lightweight framework. 2. Steep learning curve. 3. Requires you to evaluate quality before selecting a particular open source software RDK. 4. Depending on release, may not be hardened.

Table 2 Decision matrix for choosing a RDK.

4.2 Costs

The cost of producing a RDK can mainly be split into three different categories

- Total Cost of Development (TCD)
  TCD is based on five factors; time to market, number of software developers, designers, testers and managers, and the salaries of those.

- Associated Costs (AC)
  AC is based on various kinds of costs, be it cost of buying various software applications including the RDK, education of the developers, support and maintenance on the application in production, the lifespan of the application and the quality of the developed application.

- Extrinsic Costs (EC)
  EC is a very variable size. It is based on probability of project cancellation, cost of bad software designs leading to delays or redesign or removal of project features

It is important that you are able to add new components or functions to your framework, so you can continue to cut down development costs. You can either use white box[34] reuse which is the least effective way, where black box[35] reuse is much more effective. A good way to impose black box reuse is to use a RDK. This will automatically force the developers to think "this component/function could be nice to have in our RDK" and if it is a well designed RDK it is open for addition of new functions and components. When applying reuse to a web development project you can shorten the projects schedule, and thereby the man hours used on the project.

4.3 Windows vs. Linux

There have been many discussions on which environment is the cheapest to build applications in, and especially web applications. I will not come to a conclusion on this issue, but only describe which factors to take into consideration. Looking at the cost of the operating systems, development software and production environment. Generally Linux based software is a lot cheaper than Windows based software. But there are a lot of other factors to take into consideration.

[34] White box reuse is a simple form of reuse like copy and paste.

[35] Black box reuse is reuse where the developers do not need to look into the reused code. It can be reused from the documented interface.

- The price of a developer hour for the given choice of environment.
- The availability of developers for the given choice of environment.
- Where do you get the best development tools?
- Number of components available, the quality and price of these.
- Support on components / and development environment.
- The production cost / need of support for the given environment.
- What are the customer demands?

I think it is very hard to generalize on the price on each environment. It often comes to: What environment are the developers used to, and what does the customer prefers.

4.4 References

Most of the specified references used in this chapter are based on theory of frameworks, but since RDK's is a subset of frameworks much of the framework theory is still valid. The references used in this chapter are [13], [14], [15] and [16].

5 Common Features

While developing most websites a lot of features are used over and over again. In many web development houses these features are reused from project to project. But it often happens as white box reuse, which is in many cases easier than no reuse at all. Sometimes white box reuse introduces errors that could be avoided with black box reuse. So the optimal way of developing websites must be to do as much black box reuse of the most common features as possible. In this chapter I will describe some of the most common features of website development.

5.1 User management

Most dynamic websites features some kind of user management, this be registration for newsletters, user statistics or more detailed information on the specific user like name, address, Credit Card number and so on. The user management is often achieved by cookies, login/password or IP-recognition. The preferred way of recognizing the user depends on the security and environment of the web application.

Furthermore the system for user management should make the administrator able to add delete and edit user data. It should be fairly easy to add new properties to the user such as a phone number or address, if it was not included in the original design of the application.

5.2 Security

When implementing the security model of a website many things have to be taken into consideration. In this section I will not discuss operating system, hardware infrastructure, backup strategies and web server security, but only the implemented web application.

The main issues of security is to prevent unauthorized access to information, and that information only can be modified by the users intended and the users and their actions can be traced so that they can be held accountable for their actions.

While implementing the web application you must make sure that users can not get access to parts of the website that is not intended for them to see. On many websites with weak security you can just change the URL to access information such as pictures and files not intended for you to watch. And of course the security in the user management has to be tight.

At last you need to secure the program itself, so that no unintended person can edit the files of the web application, and the data in the database. This has of course something to do with choosing a professional web hosting company. You can also encrypt the data in the database, and there are several ways to encrypt the source code of the program itself.

5.3 Picture upload

The process of uploading pictures is often done by a third party component, but still you have to make several settings, where errors can occur. And to make the pictures available on the net, they often have to be resized and recompressed to perform nicely. Luckily there are also third party components for this process.

Most people using the Internet does not know how to edit and resize an image, so recompression should be a part of the picture upload, so you do not have a maximum size of for example 50kb on the picture that the user are uploading. Often people get stuck and irritated when trying to upload a picture of 1 MB, and after the 5 minutes it takes to upload the image they get a message like “The uploaded image is too large, the maximum size is 50kb!”.

So a RDK should of course include a black box reusable image upload component, supporting upload progress bar, automatically resize and recompression and ideally a web enabled cropping function.

5.4 Form validation

It is almost impossible to develop a website without the need of doing form validation, but it takes some time to do proper form validation. Of course you need to check the typed input. When the input is not typed correctly, the user needs to know what information is mistyped, and how it can be corrected. Way too often you get the screen that "something is wrong with the typed information, please go back and correct the error". Then the user has to go through the whole form again and try to find out what's wrong, which can often be hard to guess. Ideally the given error message should be in focus and have a clear description on what is wrong with the typed information.

Client- / Server side form validation

Most form validation is done by client side JavaScript, but sometimes you need to do the validation server side. The need for this is often due to more complicated validation, such as validation of an uploaded file or database comparisons. Ideally the client side and server side validation should look the same from a user point of view, and the user of the system should get similar error messages and not be forced to retype a lot of information.

5.5 Error handling

Many developers unfortunately make their websites without any error handling at all. This often causes the 500 error "unknown server error" screen to be shown, (Figure 1) and the user is stuck at this screen.

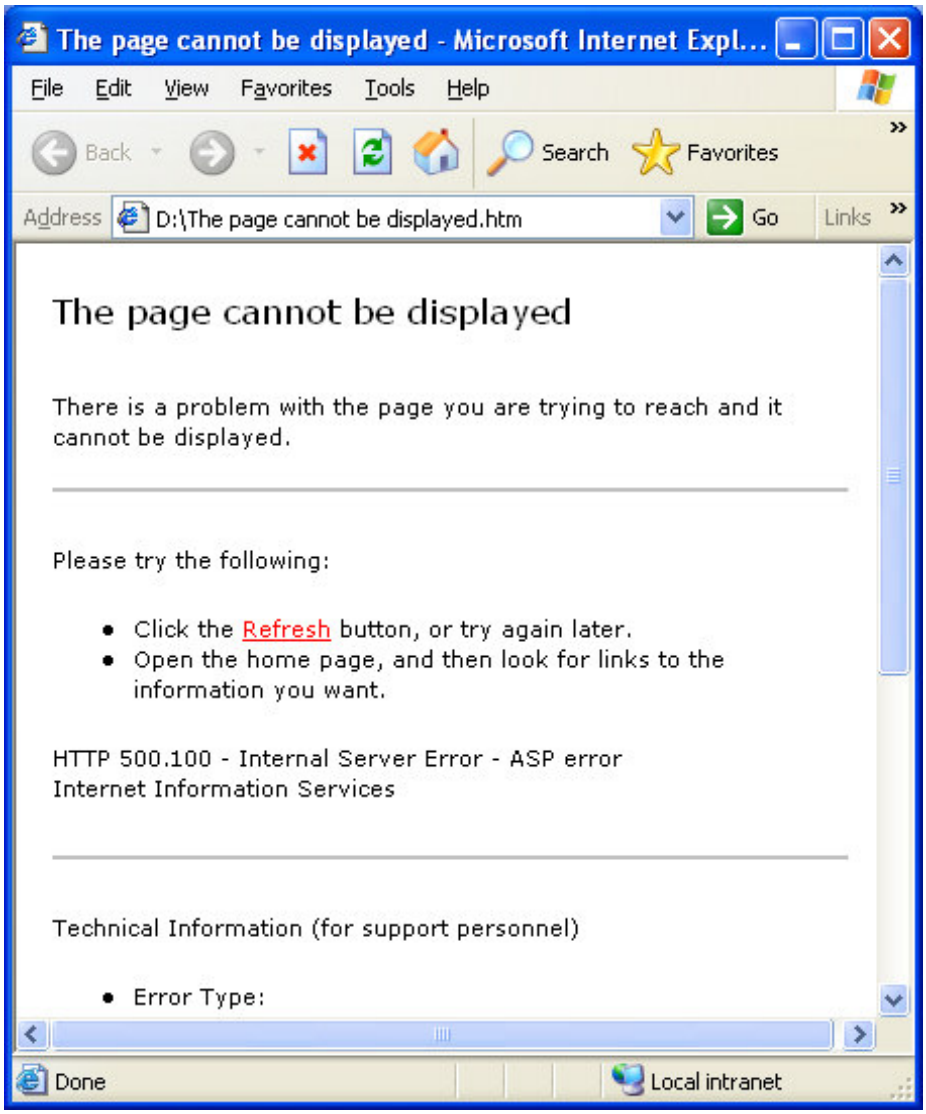


Figure 1 The typical 500 error.

First of all you often loose the user on this, and secondly the website administrator does not get notified that something went wrong on the website. The result is that the error never gets fixed.

This lack of error handling is often due to the nature of the development process and the customer relationship. Many web projects are made in a fire and forget manor. The project is specified, developed, tested, delivered to – and approved by the customer, and then forgotten. Only complaints from the customer will make the developers take up the project again.

It is utopia to believe that an error free web project is delivered every time. So the developers should be notified when an error occurs in the running website. The error should then be assessed, and a decision to ignore or fix the error should be taken. This decision process could either be done manually or by a software program, and the actions taken could depend on the customer's service level.

5.6 Database management

It should be easy for the developers to maintain the database for the website, and ideally this should be done from a web-enabled user interface. Supported functions could be cleanup operations of different kinds and maybe some statistics. Even minor changes to the application itself could be really nice to do on a website in production; e.g. addition of address to a users profile.

Search functions

Many websites have search functions, to search through articles, items in an e-shop or images. In a dynamic web application this is often done by a search through a database. Searches can be pretty complicated, but still it is possible to make reusable search functions so the developers do not need to write new search functions for every new web application.

5.7 Multilingual support

Many websites are developed for one language to begin with, and when the business is expanding the use of more languages on the website is often needed. This often results in development of an entirely new site for the second language. Doing the localization in this way, you need a translator that is able to distinguish the printed text from the websites formatting; this job can sometimes also be hard for an experienced web developer. A much easier way to translate the website is to keep all readable text in the database. Then the translator only needs to look at the printed text, and maybe some few codes for insertion of numbers and the like.

A much more complex issue in multilingual support is the sorting of words. Most databases gives you the option to set a specific sort-order based on a character-set for a specific language or region, but when combining character-sets from different regions or languages, there is no predefined way to sort these. For a closer look into this issue please have a look in the following: Appendix A.

5.8 Others

There are definitely many more features that can be seen as common. This can be advertising systems, mailing list managers, chat rooms, FAQ & Knowledgebase, Guest books, Shopping carts, Tests/Quizzes, Virtual Communities, Polls & Voting and Customer Support. These features should be able to be reused when they have been added to the RDK once, if they not already exist in the RDK.

5.9 References

The main reference for this chapter is [17].

6 Developer Manual

This manual is intended for the developer who wants to develop advanced web based database applications, without needing the knowledge of database development. The RDK can also be used for fast prototyping and "proof of concept" solutions.

The general idea in this RDK is to let the developer make configurations in a web based environment, in order to define the data structures needed on the website that the developer wants to make. For example if the developer defines a field as a TextBox. Automatically a textbox will be displayed in the web interface of the final system, without the developer needing to write any additional code. Validation of the text inserted in the TextBox will also be done automatically.

An interface for searching through the values inserted in the TextBox is also provided without the need of any additional code.

This developer manual is a technical description on how to use the RDK. If you get stuck in the technical aspects, it could be a good idea to read the tutorial in chapter 7 first. This will help you understand the practical use of the RDK better.

First the installation process of the RDK will be described in section 6.1, and then in section 6.2 the steps of developing new websites with the RDK is discussed.

6.1 Installation

The installation procedure of the RDK is made quite simple; it only requires normal knowledge of ASP web development and a very limited knowledge of MSSQL database management.

The installation procedure consists of 3 major parts.

- Install and upload of the database
- Configuring the Global.asa file
- Setting up custom error messages

6.1.1 Contents of the installation CD

The installation CD delivered with this manual contains the following directories associated to this manual.

/ASP	The directory containing all the ASP files and the default directory structure that needs to be uploaded to the web server.
/MSDE	This directory contains the free downsized copy of MSSQL server called MSDE. You only need this if you do not have access to an existing MSSQL server installation. It requires a bit of knowledge of MSSQL server to run the installation.
/Setup	Here you will find the installation program of the RDK. To run this it is required that you already have access to a running MSSQL server.
/Tutorial	The ASP files used in the recipe tutorial (chapter 7) are located in this directory.
/Sourcecode	The sourcecode of this master thesis. A description of the contents of this folder is placed in Appendix D.

6.1.2 Prerequisites

In order to install the RDK you need to have access to a MSSQL server. A copy of MSDE with Service Pack 3 is located in the /MSDE directory on the RDK CD. In order to install the database for the RDK you should have access to create tables, Stored Procedures and user defined functions[36] in the MSSQL database. If you have access to a MSSQL server at a hosting provider you should have been given these access rights.

You also need to have access to upload the ASP files used by the RDK to the web server you want to run the RDK from. You will normally upload the ASP files through the FTP[37] server of your hosting provider.

The web server should support execution of ASP files, and the components Dimac JMail from [22] and ServerObjects AspImage [20] should be installed in order to send emails from the RDK and to resize and recompress images automatically.

To sum it all up you need the following:

- Web server supporting ASP, and the possibility to upload files to the web server.
- MSSQL Server, with proper user rights.
- Dimac JMail and ServerObjects AspImage installed on the web server.
- Access to a SMTP server (outgoing mail server).

[36] User defined functions are functions implemented in the DBMS which is used by the Stored Procedures.

[37] File Transfer Protocol (FTP), a standard Internet protocol. It is a simple way to exchange files between computers on the Internet.

6.1.3 Install and upload of the database

To install the database you need to run the setup program from the installation CD located in the /Setup directory. When installation is complete go to the “Start” menu. Then click on “program files” and run the Setup RDK program. You will now see the prompt as shown on Figure 2, where you have to enter the SQL server name, the name of the database and a valid login and password of the MSSQL server you want to install the RDK database on. If you do not install the MSSQL server yourself, you should ask your hosting provider for the information needed to upload the database.

Now you can press the “Create RDK” button on the upper right on Figure 2 and the installation will begin. If the specified database does not exist you will be prompted if you want to create a new one.

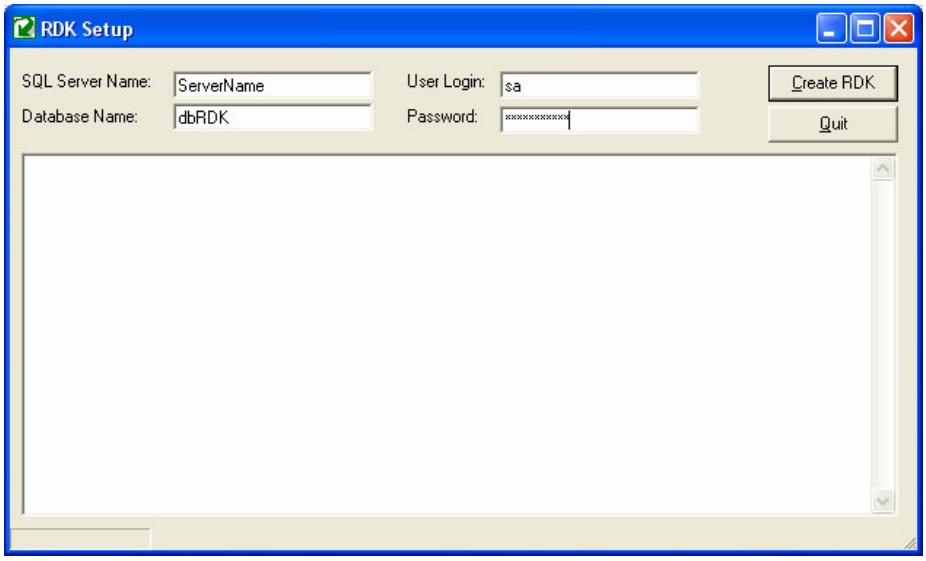


Figure 2 The install program for the RDK database.

The database for the RDK is now installed. Now we need to configure and upload the ASP files.

6.1.4 Configuration and upload of the ASP files

All the files and directories in the /ASP directory on the installation CD needs to be uploaded to the websites root directory. If you want the users of the RDK to use the image upload function, you need to make sure that the /Files directory on the web server has been given write access.

Now we need to edit the configuration files so the RDK knows the name of the MSSQL server, and the physical and virtual paths of the web server. All these settings are configured in the Global.asa[38] file located in the root of the web server directory containing the RDK. I will guide you through all the changes needed to be done to the Global.asa file, and also give a short introduction to the workings of the Global.asa file.

6.1.4.1 Configuration of Global.asa

The Global.asa file contains declarations of objects, variables, and methods that can be accessed by every ASP page in the RDK. Most settings in the Global.asa file are either application specific or specific for a single user session. There are also four events that are handled by Global.asa, these are described below.

Session_OnStart	This event is triggered when a new user is accessing the website. All user/session specific variables for the RDK are initialized when this event occurs. An example of these variables being initialized are AdminStatus and PersonID which are described in detail in section 6.2.1.

[38] The Global.asa file is an optional file that can contain declarations of objects, variables, and methods that can be accessed by every page in an ASP application.

Session_OnEnd	This event is triggered when a user logs off the website or when the user session is timing out. A timeout usually happens because the user have not been active on the website for a given time interval. In the RDK, cleanup and logout functions are called by the Stored Procedure spSetLogout, when this event is triggered. The Stored Procedures are described in section 8.2.4.
Application_OnStart	This event is triggered when the web server is starting up. Variables such as passwords for accessing the MSSQL database and some other general variables are initialized here.
Application_OnEnd	This event is triggered when the web server shuts down, such as when the server restarts. In the RDK no code is executed on this event.

In the following subsections is described which lines in the Global.asa you have to edit to make the RDK run on your server. The subsections regarding MSSQL database access, Physical and web paths and Email setup. We will start out with the MSSQL database access.

MSSQL database access

In the section Session_OnStart in Global.asa the following line should be edited, in order to connect to the MSSQL database:

```
Session("con_ConnectionString") = "Provider=SQLOLEDB.1;Data
Source=SQLServerName;Initial Catalog=dbRDK;User
ID=sa;Password=SQLPassword"
```

The “SQLServerName” should be replaced with the name of the SQL Server on which the RDK database is installed on.

The database name “dbRDK” should be changed if you are not using the default name for the database.

The default user login for the database administrator “sa” should be changed to the user login for the SQL Server and the “SQLPassword” should be changed to the password for the given SQL Server user.

Physical and web paths

In the Application_OnStart section of Global.asa you have to set the physical path of the directory where the RDK can upload images, as well as the www path. You also need to define where the log files written by the RDK should be located. You will first get a short description. Then the corresponding line in Global.asa is displayed. And the text you need to edit is marked in bold in order to get the RDK working on your web server.

The physical path of the images used and uploaded by the RDK.

```
Application("FieldImagePath") =
"C:\Inetpub\wwwroot\RDK\Files\"
```

The www path of the images used and uploaded by the RDK.

```
Application("FieldWWWImagePath") =
"http://localhost/rdk/files/"
```

The physical path of the log files.

```
Application("FieldLogPath") = "C:\Inetpub\wwwroot\WebLogs\"
```

The www path of the RDK application.

```
Application("FieldWWWPath") = "http://localhost/rdk/"
```

Email setup

If you want to send emails from the RDK you need to do some more changes to the Application_OnStart section in the Global.asa file.

The “localhost” in the following line needs to be changed to the address of your outgoing SMTP server.

```
Application("SMTPServer") = "localhost"
```

You also need to change the e-mail sender information, marked in bold, in the following lines. The information is used whenever the system sends e-mails, such as when a user has forgotten a password:

```
Application("SupportEmail") = "support@rdk.dk"
Application("SupportName") = "Support (rdk.dk)"
Application("SupportSubject") = "Support E-Mail from rdk.dk"
```

If you decide to activate the custom error messages you need to specify where the email containing the error messages should be sent. You do this by editing the following lines of Global.asa.

```
Application("ErrorEmail") = "error@rdk.dk"
Application("ErrorName") = "Error"
Application("ErrorSubject") = "Error occurred in the RDK: "
```

Now all the basics of the RDK are installed. In the following section you can define custom error messages redirection.

6.1.4.2 Custom error messages

Custom error messages are to prevent that the user of a website is met with errors like the one shown on Figure 1. Which for most is of no information at all. Instead you can show a more user friendly error message and eventually take other actions, like sending an email to the website administrator. You can enable custom error messages when you install the RDK, but I advice you to wait until you go into production. Then you can get the original IIS error messages while developing your website.

The RDK supports two ways of custom error messages: If your service provider supports asp files as custom error messages you can redirect all relevant errors to file “/error/AspErrorHandler.asp”. You can see a list of default errors in Appendix B. If your hosting provider only supports html error messages you can redirect each error to a specific page for example for an error 404 you should redirect to the file “/error/Error404.html”. In order to redirect the error messages, you have to contact the hosting provider, or if you are hosting the RDK on your own server you can have a look at [27] to see how to redirect the messages.

Enabling the custom error messages in the RDK will also provide you with an email, sent to the address specified in the Global.asa file, whenever errors occur.

6.2 Getting started with the development

Open the location of the RDK in your browser and if everything is installed correctly you should be met with the following screen (Figure 3). If the screen not appears, first you should disable the custom error messages if you enabled those. This will give you the error messages from the IIS on what is wrong. Eventually you can go through section 6.1.3 and 6.1.4 again and check your changes to Global.asa.

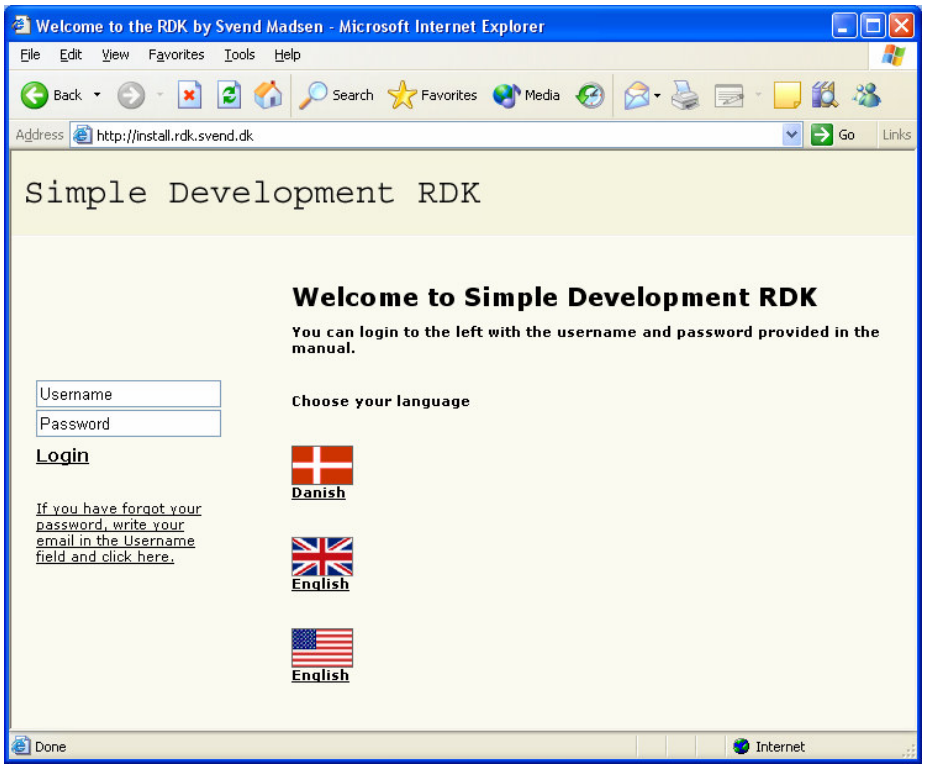


Figure 3 The start page of the RDK.

The administrator user of the RDK has the predefined username “admin”, so you are able to login as administrator to the RDK by entering the username: “admin” and password: “admin123”, without double quotes, into the fields to the left of Figure 3, and afterwards pressing enter or the “Login”.

6.2.1 The administrator menu

When logged in as an administrator you get the screen displayed in Figure 4.

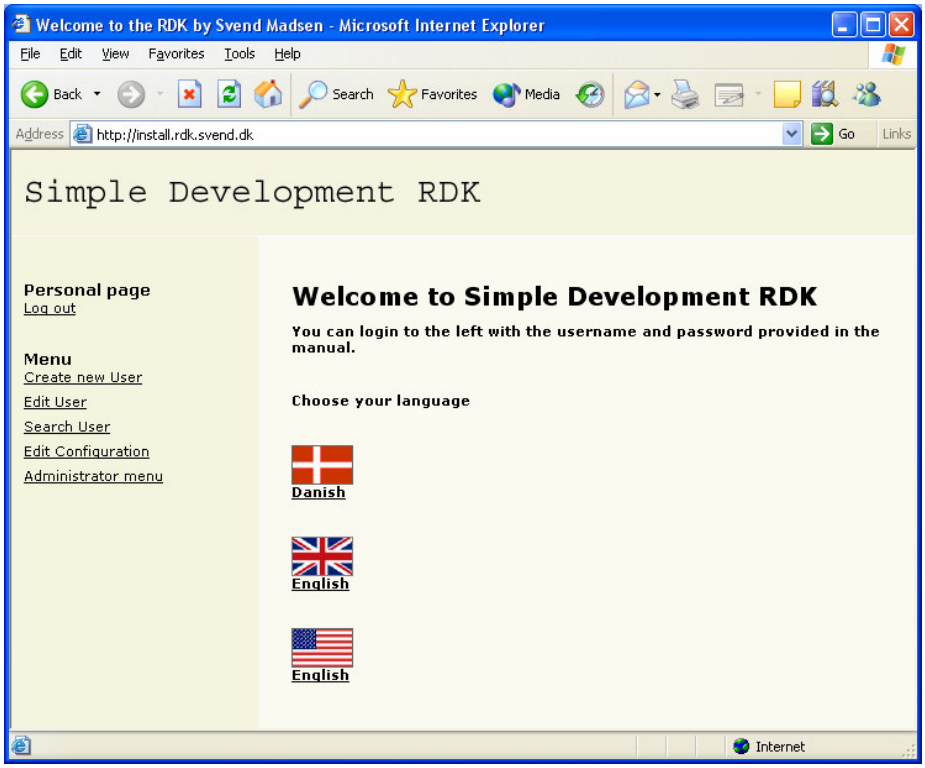


Figure 4 The Administrator menu link is at the bottom of the menu

You can now press the “Edit User” link in the menu to the left, and you get the screen displayed in Figure 5, the red arrows and the blue box is added for illustrative purposes used later in this section.

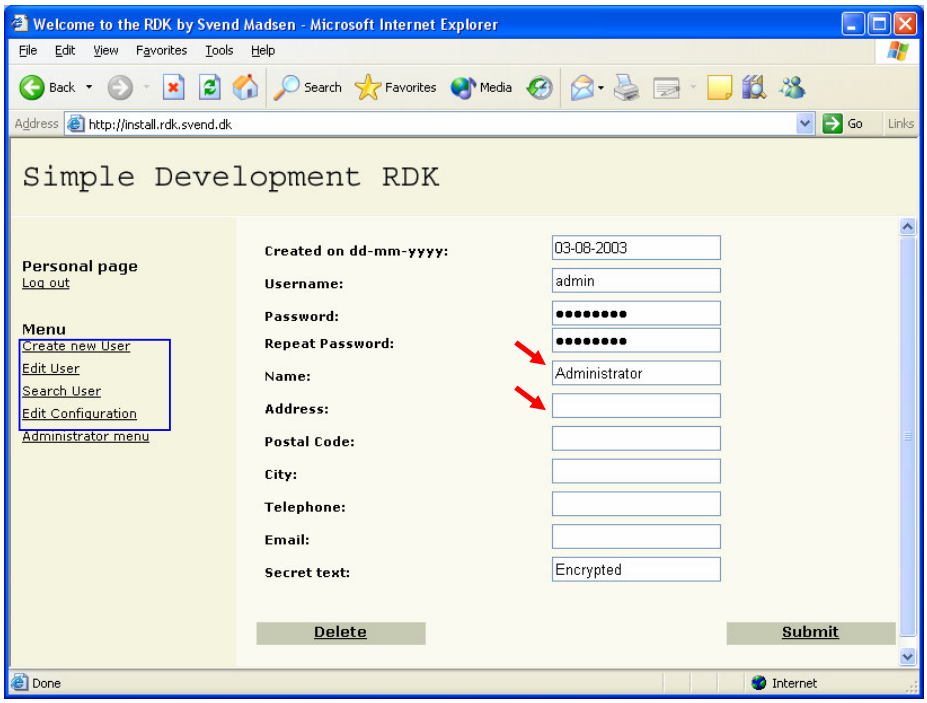


Figure 5 The User profile for the administrator.

You can now change the password to one that only you know, and save it by pressing the “Submit” button to the lower right in Figure 5.

The form you just edited was a profile of the type “User” this will be described later in this section in detail.

You can also try to login with the username “test” and password: “test123” to observe how the website looks like for the regular user.

As you properly noticed there is a link to the “Administrator menu” when you are logged in as an administrator (lower left in Figure 5).

If you click on the "Administrator menu" link, a page will popup where you have access to edit all the basic configurations in the RDK (Figure 6).

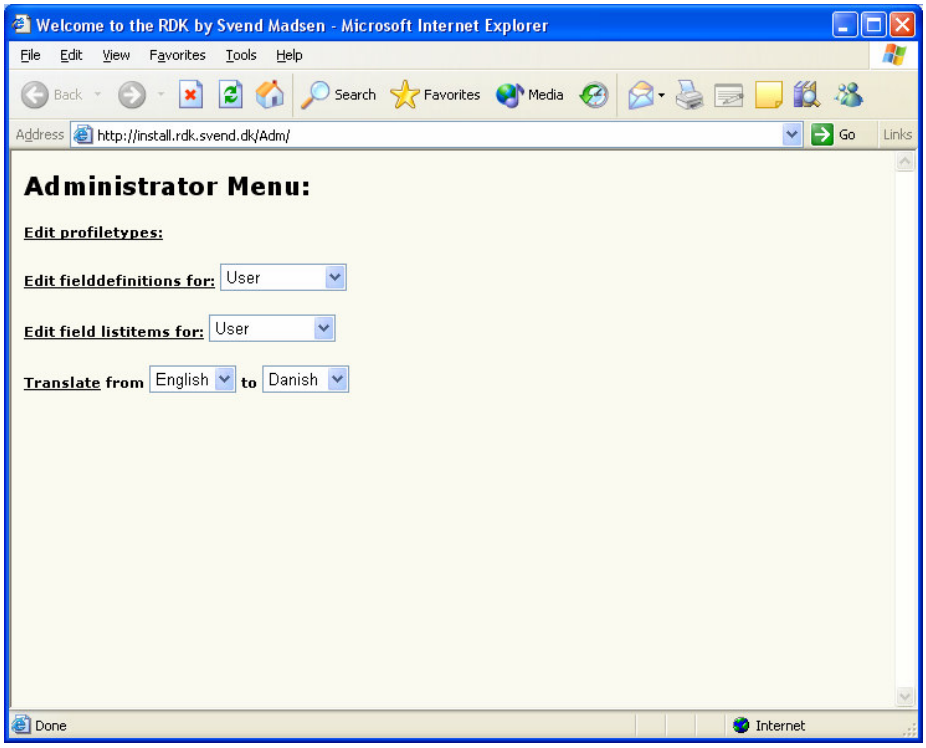


Figure 6 The administratior menu.

In Figure 6 there are four main options; I will go through them one by one. But first some definitions must be explained.

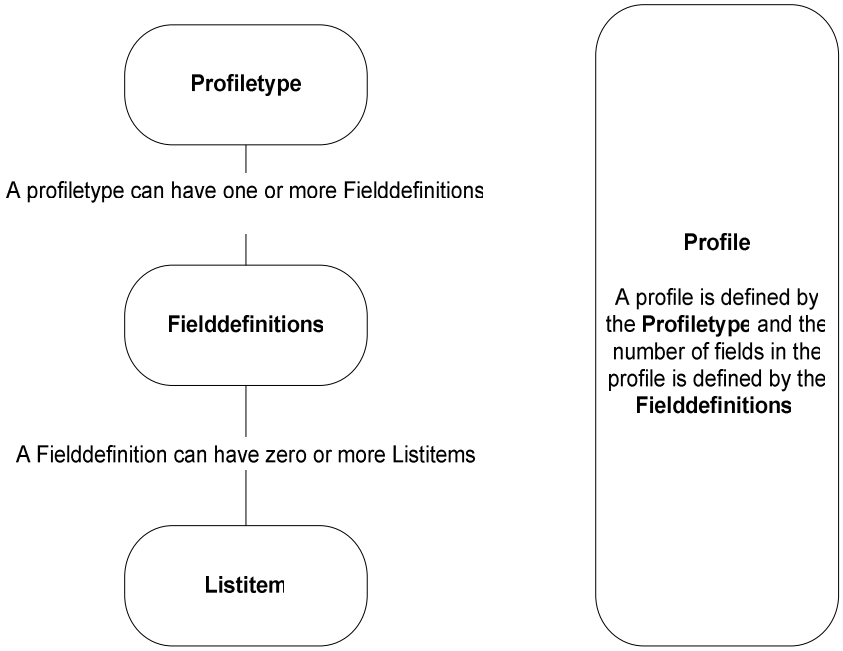


Figure 7 Overview of system objects.

The descriptions below can assist you in understanding the object hierarchy in Figure 7.

- Profiletype: A profiletype is the describer of a type of object; this object type could be users or items for an e-shop. A profiletype can contain one or more fielddefinitions described below. If you have a look at Figure 5 you can se the resulting profile of the profiletype "User" with the fields defined by the fielddefinitions.

- Fielddefinitions: These define how each profiletype looks like. The field definition for the "User" profiletype would include a "Name" and an "Address" field defined in the fielddefinitions. Both fields would be defined as the fieldtype: "TextBox" since we would like to have these fields represented as texts, and we would like to use a regular textbox to enter or edit the actual data for these fields. You can have a look at the red arrows in Figure 5 to see the result of the "name" and "address" field defined by the fielddefinitions. A fielddefinition contains properties such as fieldtype and name of the current field.

- ListItem: Fieldtypes such as "ComboBox" and "SelectList" contains some predefined values to choose from, these predefined values are defined as listitems.

- Profile: A profile is an object described by the profiletype and field definitions. A profile could be the actual admin user as displayed in Figure 5, or the item in an e-shop e.g. the guitar "Pearl River C-9". Each profile has an id those referrers to it. This id is called ProfileID.

To ensure that end users get the proper access rights there are also some definitions that must be explained.

- AdminStatus: This is an integer defining the user rights for the user logged in. A regular user has AdminStatus = 0 and the administrators has AdminStatus = 1000 as default. These settings can be changed, and you can also define levels in between 0 and 1000. This could be a person which is not the administrator, which should be allowed to add items to the e-shop. He could have an AdminStatus of 100.

- Owner: The owner of a profile is the user that has created the specific profile.

When the first option “Edit profiletypes” is chosen in the screen at Figure 6 the screen in Figure 8 appears.

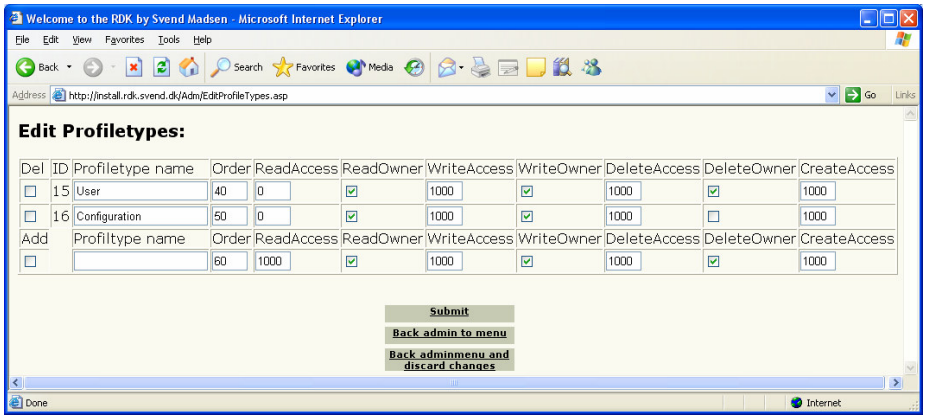


Figure 8 The edit profiletype screen.

In this screen you can create new profiletypes and define the access rights for them, based on the AdminStatus of the user trying to access a profile of this profiletype. The access rights given, has influence on which of the menu items marked with a blue rectangle on Figure 5 are displayed. The ID of the profiletype is given automatically when creating new profiletypes. Below is a more elaborated description of the configuration options.

Profiletype name:	Is the name of the profile type and will be displayed in menus as default for creating, editing and searching through profiles.
Order:	Defines the sorting of the profiletype when displayed in conjunction with other profiletypes.
ReadAccess:	Defines the minimum AdminStatus number that is allowed to read a profile of this type.
ReadOwner:	Defines if the owner should have read access to profiles of this type.
WriteAccess:	Defines the minimum AdminStatus number of who should have write access to profiles of this type.

WriteOwner: Defines if the owner should have write access to profiles of this type.

DeleteAccess: Defines the minimum AdminStatus number of who should be able to delete a profile of this type.

DeleteOwner: Defines if the owner should have delete access to profiles of this type.

CreateAccess: Defines the minimum AdminStatus number of who should be able to create a new profile of this type.

If you choose the profile type you want to edit, to the right of the link "Edit fielddefinitions for" in Figure 6 and click on the link. Then you will get the screen on Figure 9 and Figure 10, where you are able to edit all the fielddefinitions for that particular profiletype.

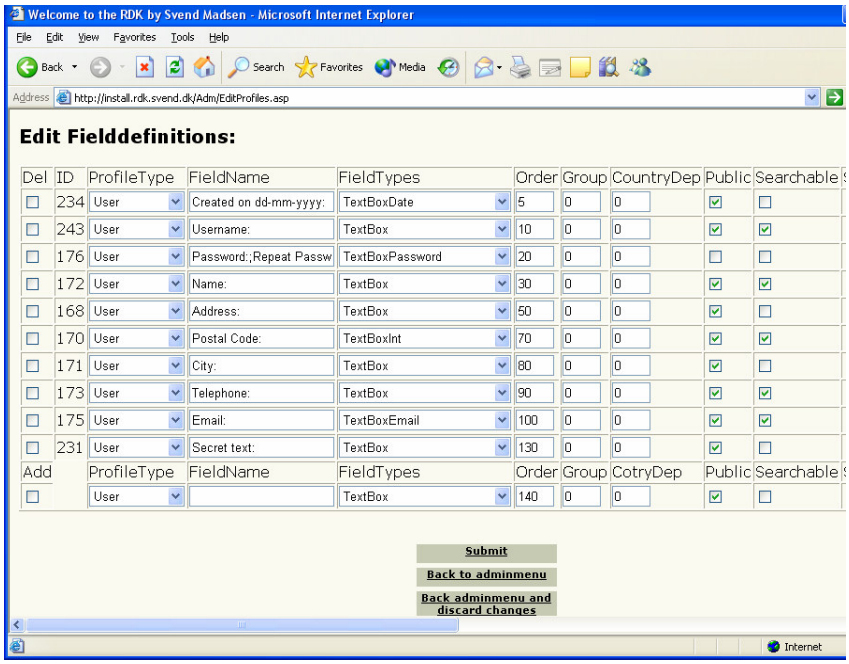


Figure 9 The fielddefinitions for the profile type "User" (scrolled to the left).

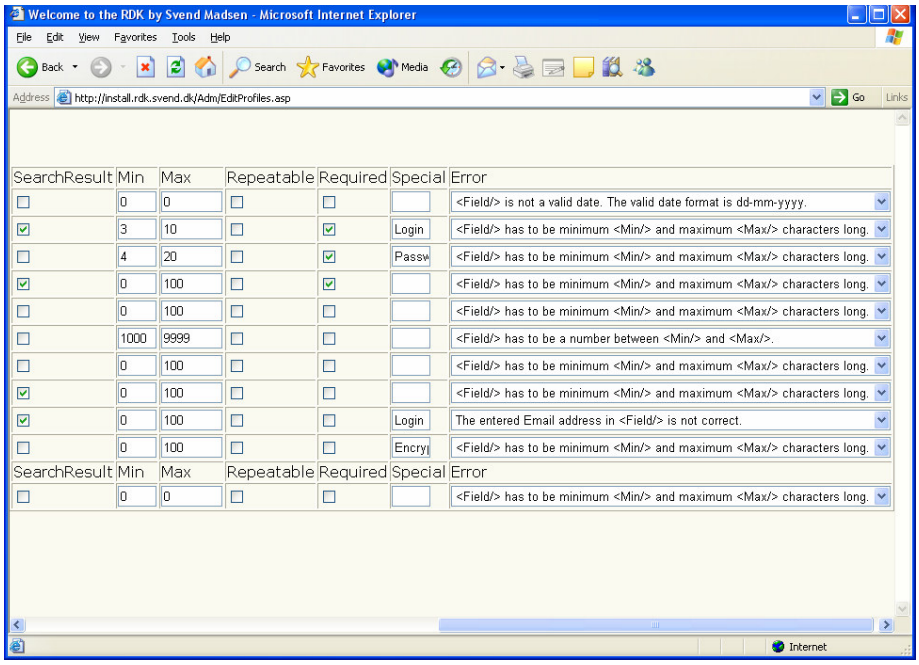


Figure 10 The fielddefinitions for the profile type "User" (scrolled to the right).

The available options for the fielddefinitions in Figure 9 and Figure 10 are described below:

ProfileType: (Figure 9) Is the profiletype that the field belongs to. If you change this value, the fielddefinition will be moved to the chosen profiletype.

FieldName: (Figure 9) Is the name of the field and the default text displayed in a specific profile. The FieldName of the field displayed in Figure 11 "TextBox" for illustration purposes. In practical use it could have been "First Name".

FieldType: (Figure 9) Defining the type of field. You can have a look at Figure 11 to Figure 21 to view the different field types.

Order: (Figure 9) Defines the order of the field in relation to the other fields in a specific profile. If two fields have the same order, they are ordered are determined by FieldName values.

Group: (Figure 9) Fields for a specific profile type can be grouped together. This field defines the group number of the field, if any. The group is only used for the graphical representation on the website.

CountryDep: (Figure 9) You can set this value to 1 if you need to list different default values in a list or combo box depending on the country or language you like. For example if you want the city name “Copenhagen” shown in a ComboBox when the user is English, and the city name “København” when the user is Danish.

Public: (Figure 9) Defines if a field should be available when retrieved by either the search or the profile view. A password would most likely not be public in either case.

Searchable: (Figure 9) Defines if the field should be shown in the default search page for the related profile. The search page is described in section 6.2.2.13.

SearchResult: (Figure 10) Defines if the field should be shown in the default search result page for the related profile.

Min: (Figure 10) Defines the minimum value for the field. Only needed for the fields of the following types TextBox, TextArea, TextBoxInt, TextBoxPassword, TextBoxEmail, TextBoxCurrency and HomePageUrl. For fields containing text the Min represents the minimum amount of characters allowed. For fields containing numbers Min represents the minimum number.

Max: (Figure 10) Defines the maximum value in the same way as described for Min.

Repeatable: (Figure 10) If Repeatable is set to 1 the field can be repeated indefinitely in a profile. For example; an image album needs one field with a title, and then multiple images. To allow this unknown number of images, we set the Repeatable value to 1 in the fielddefinition for the image field.

Required: (Figure 10) Defines if this field needs to be filled out when a profile is created or edited. The value only gives meaning for the fields of the following types; TextBox, TextArea, TextBoxInt, TextBoxPassword, TextBoxEmail, TextBoxCurrency, HomePageUrl and TextBoxDate.

Special: (Figure 10) Is used to define if the field is a special field.
To mark that a field is used as a username the value of Special must be “Login”.
To mark that the field should be matched against the password, when the user tries to login the value of Special should be “Password”.
You can also encrypt the value of the fields by setting Special to “Encrypted”.

Error: (Figure 10) Is the default error message if the data entered into the field is invalid.

The most characteristic thing for a fielddefinition is the FieldType (Figure 9). The FieldType defines what characteristics the field is going to have. The FieldType decides how the field is displayed when you edit or view a specific profile. In Figure 11 to Figure 21 you can see how each FieldType is represented when you edit a profile; this is a good reference to decide what FieldType to choose. The text and asterisk marked in red, is the default text displayed when there is an error in the typed input for the current field.

Figure 11 The fieldtype TextBox.

Figure 12 The fieldtype CheckBox.

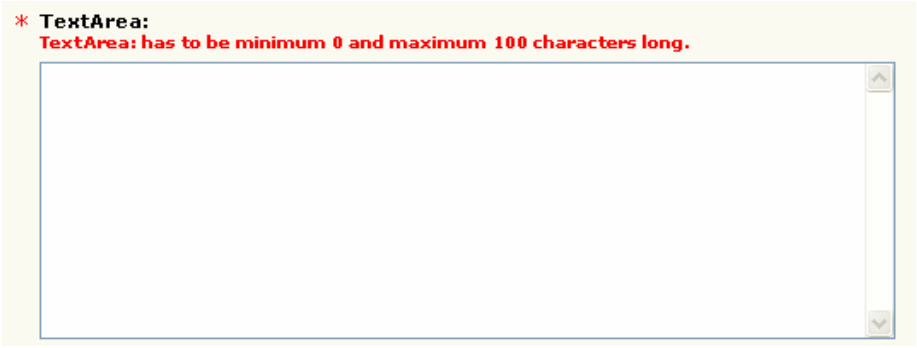


Figure 13 The fieldtype TextArea.

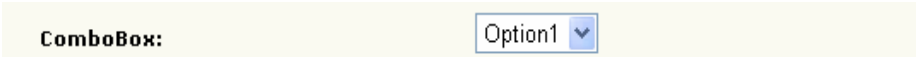


Figure 14 The fieldtype ComboBox.

A minimum of one ListItem[39] is required for each ComboBox. In Figure 14 the value of the ListItem is "Option1".

Figure 15 The fieldtype TextBoxInt.

[39] ListItems is described later in this chapter.

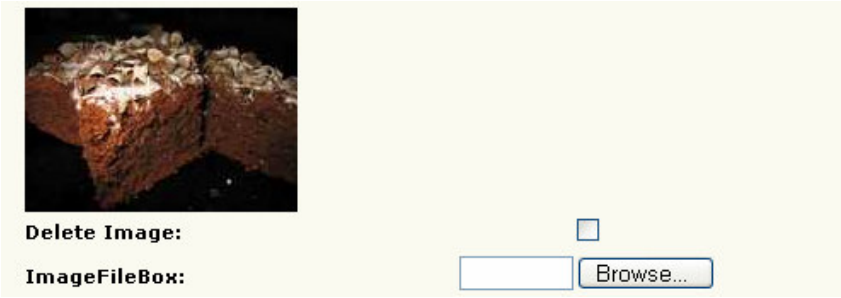


Figure 16 The fieldtype ImageFileBox.

The FieldName of an ImageFileBox must contain a ";" dividing the text for the "Delete Image" caption and the browse field description "ImageFileBox" as displayed in Figure 16.

Figure 17 The fieldtype TextBoxPassword.

The FieldName of a field with the type TextBoxPassword must contain a ";" dividing the text for the password caption "TextBoxPassword" and the "Repeat Password" caption as displayed in Figure 17.

Figure 18 The fieldtype TextBoxEmail.

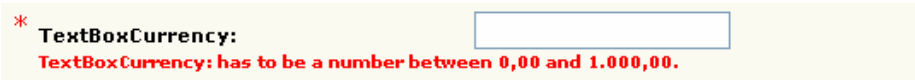


Figure 19 The fieldtype TextBoxCurrency.

Figure 20 The fieldtype SelectList.

A SelectList can have zero or more ListItems. In Figure 20 the ListItems are called “Option1” and “Option2”. The FieldName must contain XML tags for the different captions shown in Figure 20. The XML string for the example shown in Figure 20 is:

```
<Description>SelectList Description:</Description><Help>
SelectList Help</Help><Text>SelectList Text</Text><Value>
SelectList Value</Value><Add>Add</Add><Remove>Remove</Remove>
<MoveUp>&uarr;</MoveUp><MoveDown>&darr;</MoveDown>
```

Figure 21 The fieldtype HomePageUrl.

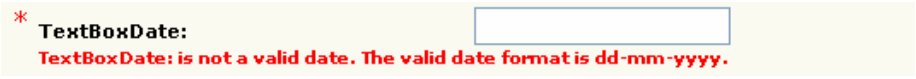


Figure 22 The fieldtype TextBoxDate.

The third link shown in the administrator menu (Figure 6) “Edit field listitems for” makes you able to add options to the ComboBox (Figure 14) and SelectList fields (Figure 20). Notice that all the listitems of the fielddefinitions of the selected profiletype are listed.

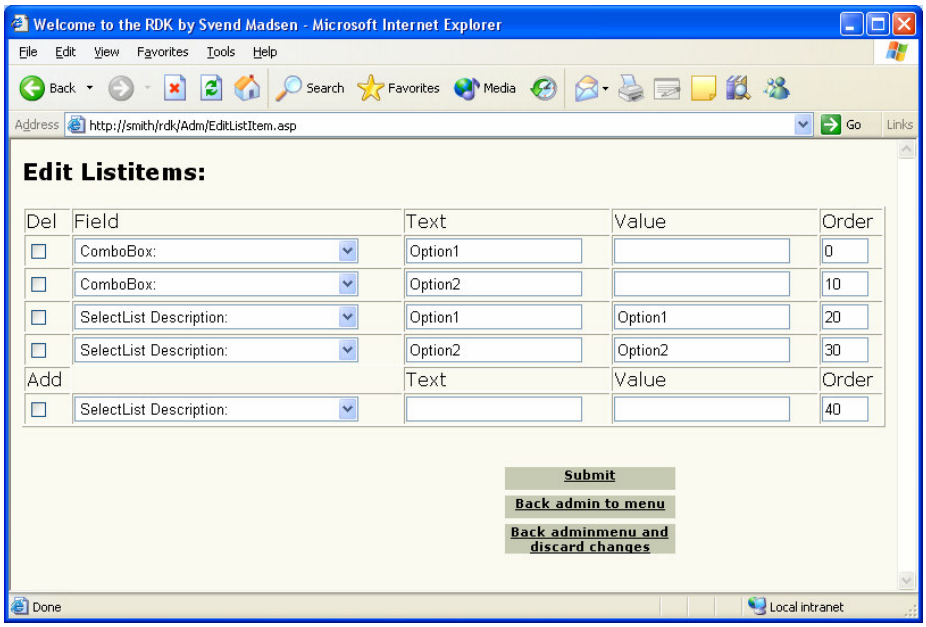


Figure 23 The ListItem view.

As seen in Figure 23 the ComboBox can only have a text, whereas the SelectList can both have a text and a value. The order field describes the order of the options when displayed. In the SelectList (Figure 20). You can see the 2 options added, “Option 1” and “Option 2”, these are defined by the two last lines in Figure 23.

The screen in Figure 24 is displayed when you click on the fourth link in the administrator menu (Figure 6) “Translate from one language to another language”.

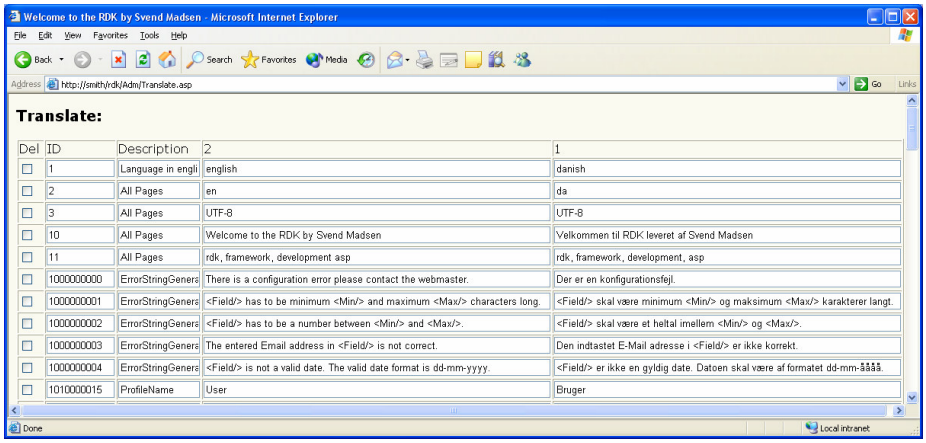


Figure 24 The translation window.

Here you are able to edit all text in the RDK. This can be FieldNames, error texts and menus. You are also able to add new texts. The ID field in Figure 24 is the reference to the text. The description field is for information only, so you are able to figure out where a text string is related.

In order to make sure that no one will edit the configuration of the RDK while going into production you can delete the /adm directory located under the /ASP directory on the installation CD as described in section 6.1.1. In that way it is not possible to edit the screens displayed in Figure 6 to Figure 10 and Figure 23 to Figure 24.

6.2.2 The asp files

As mentioned earlier, the front-end is written in ASP, but it could be converted to any other scripting language, as long as it supports connections to the MSSQL database. All the ASP files include the style sheet Style.css so you are able to edit the layout of all the ASP pages by editing only one style sheet. Additionally a style sheet with the same name as the .asp file is included in each ASP file, the purpose of this style sheet is to make page specific layout, which should not affect the layout of the other pages in the final application.

6.2.2.1 Default.asp

Default.asp is the first file that is being loaded, and it contains the definition of a frameset. The three files building the first frameset are TopMenu.asp, Login.asp and Welcome.asp.

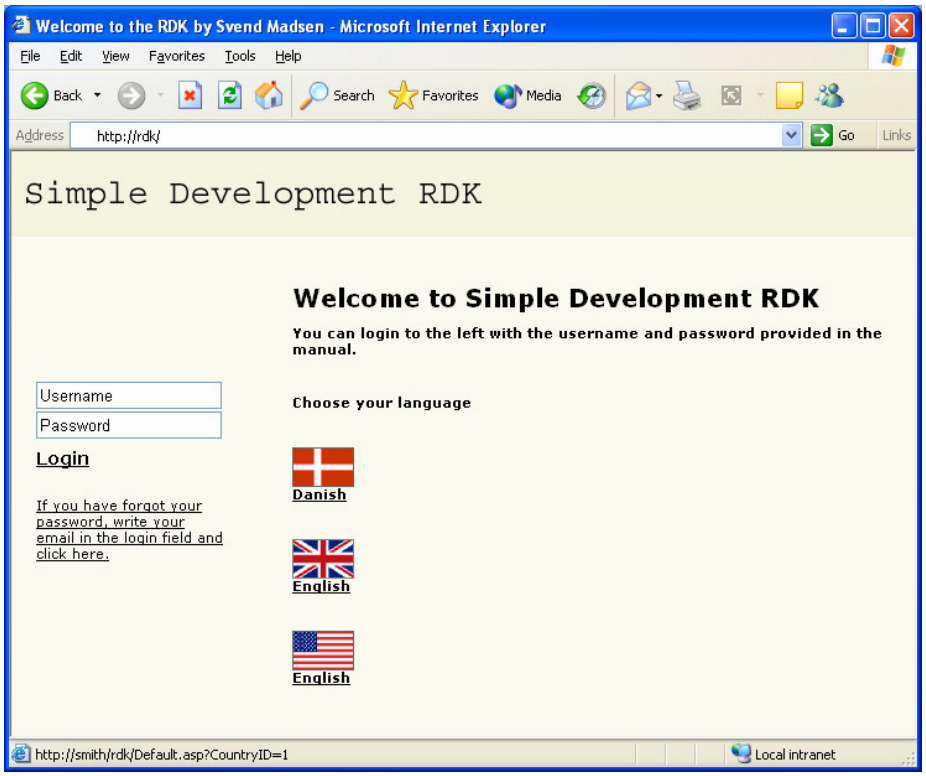


Figure 25 The start page of the RDK.

As seen in Figure 25 the slightly darker frame in the top containing the text "Simple Development RDK" is generated by TopMenu.asp. The fields to the left containing the "Username" and "Password" boxes are generated by the Login.asp file. And the main body containing the flags and the welcome text is generated by the file "Welcome.asp".

The frame definitions can be changed to match your website. You can for example choose to discard the frameset defined by Default.asp and rename the Login.asp to Default.asp so Login.asp will be loaded as the start page.

6.2.2.2 Welcome.asp

I will describe this page in detail, which will give you an overall view on how the ASP pages are constructed.

The lines:

```
<%@ codepage = 65001 LANGUAGE="VBSCRIPT" %>
<%Option Explicit%>
```

Defines that Codepage 65001 is chosen to allow all Unicode characters. And Option Explicit is to make sure that all variables are defined before they are used.

The lines:

```
<!--#include file="IncConnect.asp"-->
<!--#include file="IncLayout.asp"-->
```

Includes general functions and layout, these files are described later in this section. The layout can be edited in the admin profile when clicking on the "Edit Configuration" link. This link can be seen to the left on Figure 4.

The lines:

```
OpenRecordset("spGetLanguageStrings '"& session("CountryID")
&"', 2, 3, 10 , 11, 2001100003;")
dim strLanguage
strLanguage = DataRs("fldString")
DataRs.MoveNext
dim strCharSet
strCharSet = DataRs("fldString")
DataRs.MoveNext
dim strPageTitle
strPageTitle = DataRs("fldString")
DataRs.MoveNext
dim strPageKeywords
strPageKeywords = DataRs("fldString")
DataRs.MoveNext
dim strChooseLanguage
strChooseLanguage = DataRs("fldString")
CloseRecordset()
```

Retrieves text strings from the database given by the id and the desired language, and loads the strings into variables. In Figure 24 you can see the strings related to the id's 2, 3,10 and 11.

In the lines:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2
FINAL//<%=strLanguage%>">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=<%=strCharSet%>">
<META NAME="description" CONTENT="<%=strPageTitle%>">
<META NAME="keywords" CONTENT="<%=strPageKeywords%>">
<TITLE><%=strPageTitle%></TITLE>
<LINK REL=STYLESHEET TYPE="text/css" HREF="./style.css">
<LINK REL=STYLESHEET TYPE="text/css" HREF="./welcome.css">
</HEAD>
```

You can see how the string variables, marked with bold, retrieved from the database in the previous code snippet is loaded to the HTML page giving the head info in the correct language by the variables: strLanguage, strCharSet, strPageTitle and strPageKeywords. And the general style sheet "Style.css" and the page specific style sheet "Welcome.css" are loaded in the last two lines.

The strChooseLanguage variable is used in the code snippet below that generates the rest of the page:

```
<BODY>
<SPAN Class="WelcomeTitle"><%=GetLayout("WelcomeTitle")%>
</SPAN>
<%=GetLayout("WelcomeTextImage") & strChooseLanguage &
strFlagBar%>
</body>
```

The function GetLayout is called several places to retrieve data from the included file incLayout.asp described later.

The strFlagBar is generating the flags displayed on Figure 26. This bar is generated by a script not discussed further.

Figure 26 The page that is the result of Welcome.asp.

The Welcome.asp can be copied at renamed as a template for other pages needed.

6.2.2.3 Login.asp

This page is used to validate the user login. The ip address of the user is written to the log file AcceptedLogin.log or DeniedLogin.log on every login attempt, this makes it possible trace the user if illegal activities has been performed. The log files can be located in the path defined by Application("FieldLogPath") in the Global.asa file described under section 6.1.4.1. The default path is “C:\Inetpub\wwwroot\WebLogs\”.

6.2.2.4 TopMenu.asp

This files only purpose is to load the logo defined in the configuration profile. The profile is accessed from menu when logged in as administrator.

6.2.2.5 IncConnect.asp

This file contains functions for connecting to the database, converting variables in various ways, retrieving XML nodes from text, generating random passwords and others. If you make a new function you will use over and over again, this would be a good place to put it.

IncConnect.asp is called from most of the other ASP pages, since the functions in it is essential for most pages. For further details of the functions in this file you have view the file itself located in the /ASP directory of the installation CD.

6.2.2.6 IncLayout.asp

This file makes it possible to have a user or the administrator edit general layout features, such as the logo in TopMenu.asp or the text on the welcome page.

6.2.2.7 Menu.asp

This page is placed in the left frame when the user has logged in. It retrieves various menu items, such as creating new profiles, searching in profiles and editing profiles. The menu items are added due to the given access rights of the profiletype. As default there is also added a log out button.

6.2.2.8 EditFields.asp

When editing or creating a profile this file is used. If editing an existing profile the query string should contain the variable ProfileID containing the id of the profile to be edited. If creating a new profile the variable ProfileTypeID should contain the id of the profiletype to be created.

```
        SELECT CASE DataRS("fldFieldID")
'************ Start of commented area ****************
'Example of how to keep non administrators from editing the date.
'       Case 100 'FieldID
'           If Session("AdminStatus") >= 1000 Then
'               WriteField DataRS("fldFieldName"), DataRS("fldFieldID"), DataRS("fldFieldListNo"),
'           Else
'               WriteField "Show", DataRS("fldFieldID"), DataRS("fldFieldListNo"), DataRS("fldFiel
'           End If
'************ End commented area *********************

'************ Start of commented area ****************
'Example on how to autogenerate a password
'       Case 100 'FieldID
'           If DataRS("fldData") <> "" Then
'               WriteField DataRS("fldFieldName"), DataRS("fldFieldID"),DataRS("fldFieldListNo"),
'           Else
'               Response.Write("<INPUT TYPE=Hidden NAME=ctrl"& DataRS("fldFieldID") &" VALUE='"& C
'           End If
'************ End commented area *********************

'************ Start of commented area ****************
'Example of how to keep non administrators from viewing a field.
'       Case 100 'FieldID
'           If Session("AdminStatus") >= 1000  Then
'               WriteField "Show", DataRS("fldFieldID"),DataRS("fldFieldListNo"), DataRS("fldFieldC
'           End If
'************ End commented area *********************
        CASE ELSE
            'If error found on server insert errorvalue
            If trim(strFieldID) = trim(DataRS("fldFieldID")) Then
                WriteField DataRS("fldFieldName"), DataRS("fldFieldID"),DataRS("fldFieldListNo"),
            Else
                WriteField DataRS("fldFieldName"), DataRS("fldFieldID"),DataRS("fldFieldListNo"),
            End If
        END SELECT
        DataRS.MoveNext
    Loop
```

Figure 27 Line 203 to 241 of EditFields.asp.

If you want to change the way a field is edited, as displayed on Figure 11 to Figure 22, you can insert the field id in the select case statement displayed in Figure 27. There are already some examples that are commented out, shown as in green text.

6.2.2.9 Incfield.asp

This file is included in EditFields.asp, and is used to specify how each fieldtype should be displayed when editing.

6.2.2.10 EditFieldsSave.asp

This file is called when a user presses submit on a page generated with EditFields.asp. In this file all the data of the fields in EditFields.asp are saved into the database.

If one or more fields contain a picture, they are resized and recompressed using the third party component AspImage [20].

6.2.2.11 ShowProfile.asp

When displaying the data of specific profile this file is used. The display could be the information on at guitar in an e-shop. The Access rights for a user wanting to view the given profile are also checked.

```
IF DataRS.EOF AND DataRS.BOF THEN
    Response.Write(strNoRecord)
ELSE
    %><!--#include file="IncShowField.asp"--><%
    Do Until DataRS.EOF
        SELECT CASE DataRS("fldFieldID")
'       CASE 100
'           Do something
        CASE ELSE
            WriteShowField DataRS("fldFieldName"), DataRS("fldFieldID"), DataRS("fldFieldListNo")
        END SELECT
        DataRS.MoveNext
    Loop
    EndShowField()
    CloseRecordset()
```

Figure 28 Line 119 to 133 of ShowProfile.asp.

In the select case statement in Figure 28 you can include exceptions on how a specific field is displayed when a profile is shown. It works in the same way as for the file EditFields.asp.

6.2.2.12 IncShowField.asp

This file is included in ShowProfile.asp and is used to specify how a field of each type is displayed.

6.2.2.13 SearchFields.asp

When doing a search of a specific profile type this page is used to display the search input fields based on profiletype and field definitions.

6.2.2.14 SearchFieldsGet.asp

This file is used to display the search results.

```
Select Case DataRS("fldFieldID")
'Case 100
'    Do something
Case Else
    strResponse = strResponse & strTmpResponse
End Select
```

Figure 29 Line 175 to 180 of SearchFieldsGet.asp.

In Figure 29 you can include exceptions on how a field is displayed in the resulting search page. Notice that the fields are written to the string strResponse. strResponse in the later written to the HTML page.

6.2.2.15 IncUploadProgress.asp and UploadProgress.asp

These files are used to make a progress bar when uploading images. The source code for these files is from [23] and edited to match into the RDK.

6.2.2.16 DeniedAccess.asp

This file is called if the user tries to access a profile which is not allowed due to the lack of proper user rights based on AdminStatus.

6.2.2.17 Exit.asp

When this file is accessed the session is reset and the user is redirected to Default.asp

6.2.2.18 Maintenance.htm

File to display when maintaining the website.

6.2.2.19 robots.txt

The file used to define which pages spiders and search engines should index. A description of the use of robots.txt is included in Appendix C.

6.2.2.20 FormValidate.js

This file is called from EditFields.asp and SearchFields.asp and is used to do client side validation of the fields displayed in Figure 11 to Figure 22 when the user submits the page.

7 Tutorial

To get a better understanding of how easy it is to implement a relative advanced website I will make a step by step implementation of a website for food recipes. The tutorial is based on the result of a real implementation of the website http://ullasopskrifter.dk which is in production containing more than 350 recipes. The website in production is in Danish, but this tutorial will describe a similar website in English.

7.1 Installation of the database

In order to get started we need to create the MSSQL database for the RDK. This is done through the web interface at our hosting provider, which in this example is http://hosting.mitc.dk.

The database server at our hosting provider is called: “mssql3.mitc.dk“.

The name of the database we create is: “svend_dbRecipe”.

The username is: “svend_usrRDK” with the password “dRTger32”.

Using the install program described in the developer manual in section 6.1.3, we create the database needed for the RDK with the settings given above.

7.2 Global.asa

In order to have the RDK running at our hosting provider, we got to upload the asp files and do the following changes to Global.asa.

We have to edit line 44 marked with the red arrow in Figure 30 to correspond to the information of the MSSQL server given in chapter 7.1.

```
Sub Session_OnStart
    '************ Start of commented area ***************
    'User specific variables.
    Session("PersonID") = NULL
    Session("ProfileType") = NULL
    Session("ProfileID") = NULL
    Session("CountryID") =2
    Session("AdminStatus") =0
    Session.LCID = 2057
    '************ End commented area ********************

    '************ Start of commented area ***************
    'Database connection.
    session.Timeout = 60
    Session("con_ConnectionString") = "Provider=SQLOLEDB.1;Data Source=SQLServerName;Initial Cata
    Session("con_ConnectionTimeout") = 15
    Session("con_CommandTimeout") = 30
    Session("con_CursorLocation") = 3
    '************ End commented area ********************
End Sub
```

Figure 30 The Session_OnStart of Global.asa

When editing this line we get the following:

```
Session("con_ConnectionString") = "Provider=SQLOLEDB.1;Data
Source=mssql3.mitc.dk;Initial Catalog=svend_dbRecipe;User
ID=svend_usrRDK;Password=dRTger32"
```

Now we need to set up the path information to correspond with the new environment. This is done by editing the path information in Global.asa, beginning with the line marked with the red arrow in Figure 31.

```
Sub Application_OnStart
    '************ Start of commented area ***************
    'Definiton of application specific paths.
    'The physical path of the images used and uploaded by the RDK.
    Application("FieldImagePath") = "C:\Inetpub\wwwroot\RDK\Files\"
    'The www path of the images used and uploaded by the RDK.
    Application("FieldWWWImagePath") = "http://localhost/rdk/files/"
    'The physical path of the directory to write logevents to.
    Application("FieldLogPath") = "C:\Inetpub\wwwroot\WebLogs\"
    'The www path of the RDK application.
    Application("FieldWWWPath") = "http://localhost/rdk/"
    '************ End commented area ********************

    '************ Start of commented area ***************
    'Other constants.
    Application("UploadComponent") = "UploadProgress 1.0"
    Application("SMTPServer") = "localhost"
    Application("EmailComponent") = "JMail"
    Application("TimeZone") = 0
    '************ End commented area ********************

    '************ Start of commented area ***************
    'Application specific email definitons.
    Application("SupportEmail") = "support@rdk.dk"
    Application("SupportName") = "Support (rdk.dk)"
    Application("SupportSubject") = "Support E-Mail from rdk.dk"

    Application("ErrorEmail") = "error@rdk.dk"
    Application("ErrorName") = "Error"
    Application("ErrorSubject") = "Error occurred in the RDK: "
    '************ End commented area ********************
```

Figure 31 The beginning of the Application_OnStart in Global.asa

The paths on Figure 31 are edited to the paths marked in bold, in the code snippet below:

```
Application("FieldImagePath") =
"D:\hshome\svend\recipe.rdk.svend.dk\files\"
'The www path of the images used and uploaded by the RDK.
Application("FieldWWWImagePath") =
"http://recipe.rdk.svend.dk/files/"
'The physical path of the directory to write logevents to.
Application("FieldLogPath") =
"D:\hshome\svend\recipe.rdk.svend.dk\WebLogs\"
'The www path of the RDK application.
Application("FieldWWWPath") = "http://recipe.rdk.svend.dk/"
```

We decide to skip the setup of emails described in the developer manual section 6.1.4.1. So the RDK is now up and running at our new server.

7.3 Short recipe website system specification

We make a short system specification for our recipe website. We will not concentrate on the graphics, but just use the layout provided by the RDK.

For our recipe website we want a categorized list of recipes.

The administrator should be able to login and add new recipes.

One recipe should be associated to a category such as cakes or bread. We need a title, the number of persons the recipe is made for, the cooking time and preparation time. A recipe should also have an image displaying the finished food, an ingredients list and a description of how to prepare the dish. Additionally we want to be able to create a new recipe without making it public, or remove a recipe without deleting it.

We want a menu with all the recipe categories. When the user clicks on a category in the menu, for example “cakes”, all the recipes in the category “cakes” should be displayed with title, number of persons, preparation time and cooking time and a small image.

When the user clicks on a recipe all the details of the recipe and a larger image should be displayed.

Additionally we want a custom search where the user can search for the category, title, in the ingredients or in the preparation list.

7.4 Implementation of the recipe website

Since we already have installed the RDK at our hosting provider we are able to start configuring the RDK due to the system specification.

7.4.1 RDK configuration

According to the system specification we need to create a profiletype called recipe. So we open our browser and login to the website at http://recipe.rdk.svend.dk/ we use the username “admin” and password “admin123” as provided by the developer manual. We click on the “Administrator Menu” and “Edit Profiletypes” also described in the developer manual in section 6.2.1.

As displayed in Figure 32 we add the profiletype with the name “Recipe”, marked with the red arrow. Since we only want the administrator to be able to write to, delete and create new recipes, we set the numbers “WriteAccess”, “DeleteAccess” and “CreateAccess” to “1000”, which corresponds to the AdminStatus of the administrator. We want everybody to be able read the recipes, so we set the ReadAccess to ”0”.

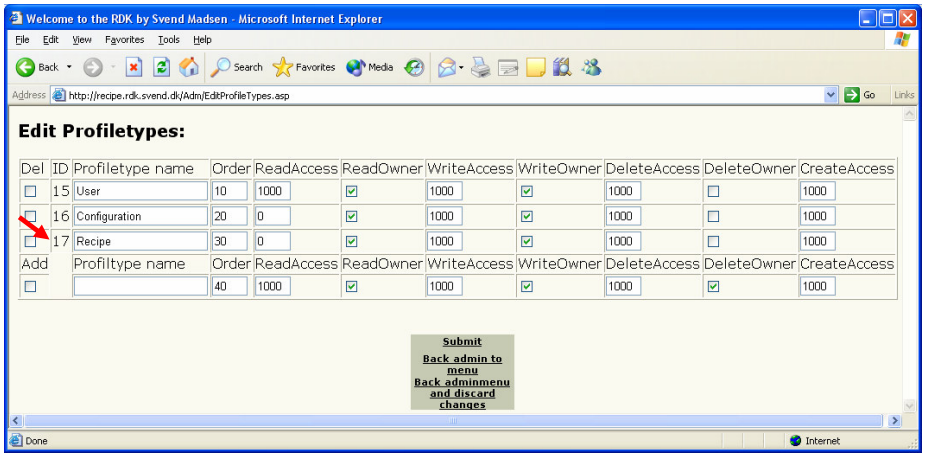


Figure 32 Addition of the ProfileType Recipe.

We can now go to the menu “Edit fielddefinitions for:” and choose “Recipe” from the combo box in the “Administrator Menu”. Then we can add the fields as specified in the system specification the result is displayed in Figure 33.

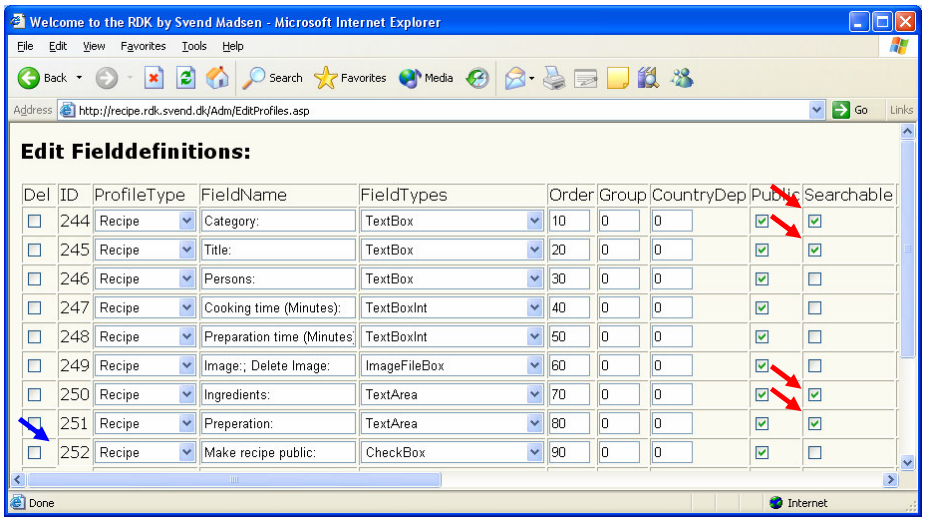


Figure 33 The fielddefinitions for the profiletype "Recipe" (scrolled to the left).

We have specified that we should be able to search through “Category”, “Title”, “Ingredients” and “Preparation”. This is achieved by ticking the checkbox “Searchable” for the given fields marked with a red arrow on Figure 33.

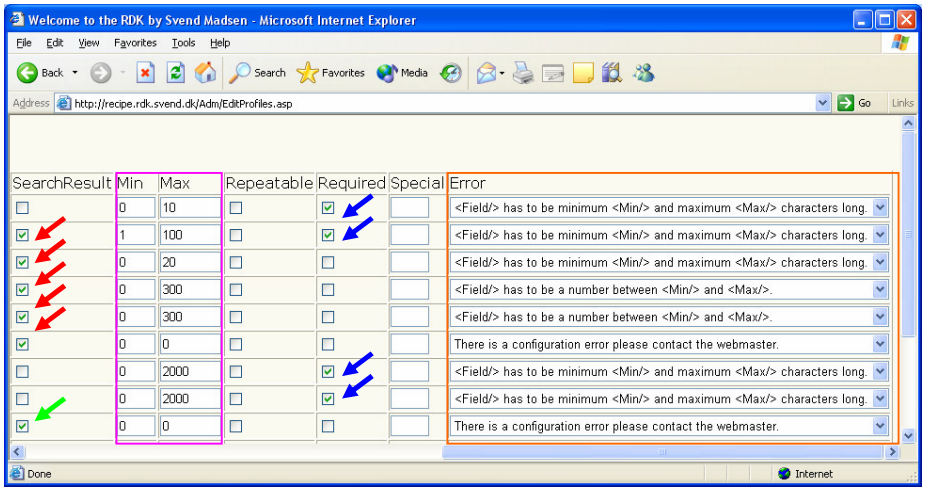


Figure 34 The fielddefinitions for the profiletype "Recipe" (scrolled to the right).

We also tick the field we want to be in the "SearchResult" as displayed on Figure 34, marked with red arrows. Notice the field "Make recipe public", marked with a green arrow, is also ticked. We need this value to determine to display the recipe or not.

In order to have a valid recipe we decide that the fields Category, Title, Ingredients, Preparation are required, so we tick the checkbox "Required" for these fields, marked with blue arrow on Figure 34. Now we only need the set the minimum and maximum values as displayed in Figure 34 in the pink frame, and select the proper error messages for each field, displayed in the orange frame of Figure 34.

Now the profiletype "Recipe" and the fielddefinitions are configured.

Then we need to make the administrator able to add the menu items corresponding to the recipe categories such as bread and cakes. The "Configuration" profiletype is predefined in the RDK.

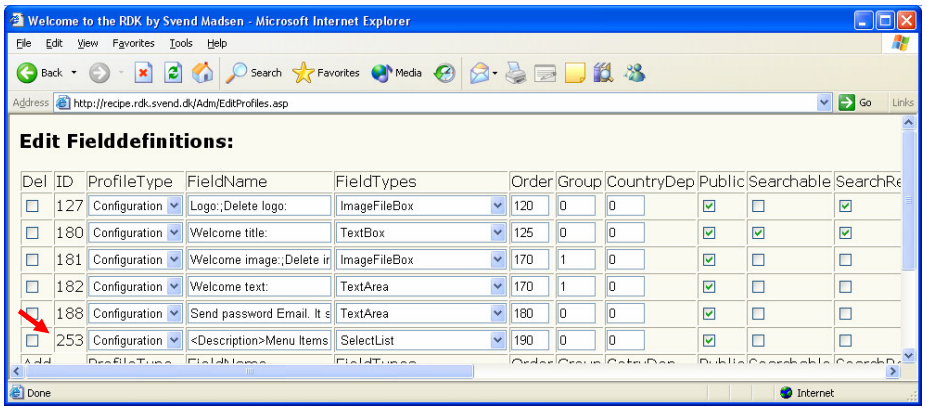


Figure 35 The added fieldtype "SelectList" is in the bottom of the figure.

So we only need to add a fielddefinition with the fieldtype "SelectList". The added fielddefinition can be seen in Figure 35 marked with a red arrow.

This was the configuration of the RDK. Now we need to edit the ASP files in order to match the requirements of the system specification.

7.4.2 ASP file edits

First of all we do not want the default login frame to the left as displayed in section 6.2.2.1 Figure 25. Instead we want the menu displaying the search option for the recipes. To do this we need to edit the Default.asp file.

Default.asp

In Default.asp we want to replace the link to the left frame from "./Login.asp" to "./Menu.asp".

```
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 FINAL//<%=strLanguage%>">
<html>
<head>
<link rel="shortcut icon" href="/favicon.ico">
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=<%=strCharSet%>">
<meta NAME="description" CONTENT="<%=strPageTitle%>">
<meta NAME="keywords" CONTENT="<%=strPageKeywords%>">
<title><%=strPageTitle%></title>
<frameset rows="70,*" frameborder="NO" border="0" framespacing="0" >
  <frame name="top" scrolling="NO" noresize src="./TopMenu.asp" bordercolor="#003333" frameborder
  <frameset frameborder="NO" border="0" framespacing="0"  cols=200,*>
    <frame name="left" src="./Login.asp" scrolling=no>
    <frame name="main" src="welcome.asp">
  </frameset>
</frameset>
</head>
</html>
```

Figure 36 Section of Default.asp

We edit the line marked with a green arrow in Figure 36, to the line displayed in the code snippet below.

```
<frame name="left" src="./Menu.asp" scrolling=auto>
```

Then the menu will be displayed instead of the login page.

In order to display the categories of the recipes as a menu we need to be able to retrieve the items added to the "SelectList" added to the profiletype "Configuration" during the configuration of the RDK. Since this menu is a part of the general layout, we will ad the code for the menu retrieval in incLayout.asp.

incLayout.asp

In the ASP file incLayout.asp there is a function called GetLayout with one input parameter that specifies what layout item the function should retrieve.

We define that the input parameter for retrieving the menu based on the "SelectList" in called "MenuBar".

```
Do Until DataRSLayout.EOF
    Select Case DataRSLayout("fldFieldID")
    Case 127 And strLayoutPart = "Logo"
        GetLayout = Application("FieldWWWImagePath") &"thumb_"& Session("Logo") & DataRSLayout("f
    Case 127 And strLayoutPart = "MailLogo"
        GetLayout = Application("FieldImagePath") &"thumb_"& Session("Logo") & DataRSLayout("fldD
    Case 180 And strLayoutPart = "WelcomeTitle"
        GetLayout = DataRSLayout("fldData")
    Case 181 And strLayoutPart = "WelcomeTextImage"
        If Not Isnull(DataRSLayout("fldData")) And DataRSLayout("fldData") <> "" Then
            If intTextImageLast = 182 Then
                GetLayout = GetLayout & "<DIV CLASS=WelcomeTextImage><SPAN CLASS=WelcomeImage><IM
            Else
                GetLayout = GetLayout & "</DIV><DIV CLASS=WelcomeTextImage><SPAN CLASS=WelcomeIma
            End If
            intTextImageLast = 181
        End If
    Case 182 And strLayoutPart = "WelcomeTextImage"
        If intTextImageLast = 181 Then
            GetLayout = GetLayout & "<SPAN CLASS=WelcomeText>" & TextAreaString(DataRSLayout("fld
        Else
            GetLayout = GetLayout & "<DIV CLASS=WelcomeTextImage><SPAN CLASS=WelcomeText>" & Text
        End If
        intTextImageLast = 182
    End Select
    DataRSLayout.MoveNext
Loop
```

Figure 37 The loop in incLayout.asp

In incLayout.asp there is a loop (Figure 37) that goes through the administrator's profile of the profiletype "Configuration", since we added the "SelectList" to the configuration profiletype and the "SelectList" got the id 253 as displayed in Figure 35 by the red arrow. We need to generate the menu from the data of the field with the id 253 when the function GetLayout is called with the parameter "MenuBar".

```
Case 253 And strLayoutPart = "MenuBar"
    Dim intLocation1, intLocation2, strMenuItem, strLastParent, boolSub
    GetLayout = GetLayout & "<script type=""text/javascript""><!-- " & vbCrlf & "function swap
    boolSub = false
    intLocation1 = 2
    intLocation2 = InStr(DataRSLayout("fldData"),"~#")
    'Seperate out the menu items and add them to a HTML based menu
    Do Until intLocation2 < 1
        strMenuItem = Mid(DataRSLayout("fldData"),intLocation1,intLocation2-intLocation1)
        ''Allow submenus seperated from the parent with a :
        If instr(strMenuItem,":") > 0 Then
            If strLastParent <> left(strMenuItem,instr(strMenuItem,":")-1) Then
                If boolSub Then GetLayout = GetLayout & "</div>"
                boolSub = true
                GetLayout = GetLayout & "<DIV CLASS=MenuHeader onClick=""swapClass(1,'mn"& lef
                GetLayout = GetLayout & "<DIV CLASS=MenuItem><A HREF=""SearchFieldsGet.asp?Fir
            Else
                GetLayout = GetLayout & "<DIV CLASS=MenuItem><A HREF=""SearchFieldsGet.asp?Fir
            End If
            strLastParent = left(strMenuItem,instr(strMenuItem,":")-1)
        Else
            If boolSub Then GetLayout = GetLayout & "</DIV>"
            boolSub = false
            GetLayout = GetLayout & "<DIV CLASS=MenuHeader><A HREF=""SearchFieldsGet.asp?First
            strLastParent = ""
        End IF
        intLocation1 = intLocation2 +2
        intLocation2 = InStr(intLocation1+2,DataRSLayout("fldData"),"~#")
    Loop
    If boolSub Then GetLayout = GetLayout & "</div>"
    GetLayout = GetLayout & "</SPAN>"
```

Figure 38 The code snippet generating the menu.

The code snippet in Figure 38 shows the menu generation and is inserted in incLayout.asp at the green arrow in Figure 37. The snippet is only displayed to give you an idea of the amount of code and business logic used for the menu, since it is cropped from the right. The code can be found in its full length and context in the /Tutorial directory of the installation CD.

When creating a new recipe we have to choose what category, and thereby under which menu item the recipe belongs to. We could make the user type in the text corresponding to the name of a menu item, but we rather want the user to select the category from a combo box generated from the actual menu items. For this we just need a way to retrieve the raw data from the "SelectList" containing the menu items. This is done by the code snippet below.

```
Case 253 And strLayoutPart = "ComboMenu"
      GetLayout = DataRSLayout("fldData")
```

As you can see in the code snippet marked with bold, you need to call the GetLayout with the parameter ComboMenu to retrieve the data. This data is used to ingenerate the combo box with the menu items/categories for the recipes.

Menu.asp

Since we are not using any menu headers before the menu with categories, we can delete the lines from 78 to 128 in the ASP file Menu.asp. To make the administrator able to login we need to add a login button on the page.

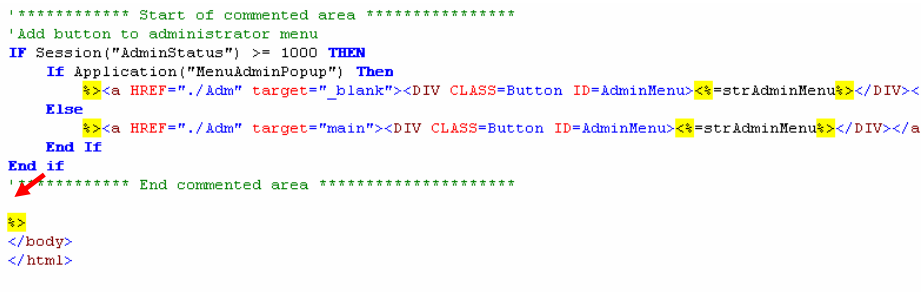


Figure 39 Part or the file Menu.asp

The code below is added in the line marked with a red arrow in Figure 39.

```
'************ Start of commented area ****************
'Add button to Log in and Out
IF Session("PersonID") <> "" THEN
      Response.Write("<a HREF=""./Exit.asp"" TARGET=_top>
      <DIV CLASS=Button ID=Login>"& strExit &"</DIV></a>")
Else
      Response.Write("<a HREF=""./Login.asp""><DIV
      CLASS=Button ID=Login>"& strLogin &"</DIV></a>")
End if
'************ End commented area *********************
```

This code snippet checks if a user is logged in and adds a login button, if the user is already logged in a logout button is added.

```
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=<%=strCharSet%>">
<meta NAME="description" CONTENT="<%=strPageTitle%>">
<meta NAME="keywords" CONTENT="<%=strPageKeywords%>">
<title><%=strPageTitle%></title>
<link REL="STYLESHEET" TYPE="text/css" HREF="./style.css">
<link REL="STYLESHEET" TYPE="text/css" HREF="./menu.css">
</head>

<body>
<%

'************ Start of commented area ****************
'Retrives data for new profiles to make edit search and create new menu items
Dim strProfileTypeID, strProfileID, strName, strOldProfileTypeID
OpenRecordset("spGetProfiles '"& session("CountryID") &"', '"& session("PersonID") &"';")
Do Until DataRS.EOF
    strProfileTypeID  = DataRS("fldProfileTypeID")
    strProfileID = DataRS("fldProfileID")
    strName = DataRS("fldString")
    If strProfileTypeID <> strOldProfileTypeID Then
        strOldProfileTypeID = strProfileTypeID
        If session("AdminStatus") >= 1000 And CheckAccess(strProfileID,"CreateAccess", false, str
            If Application("MenuCreatePopup") Then
                %><A HREF="javascript:void(0)" onClick="window.open('./EditFields.asp?ProfileType
            Else
                %><A HREF="./EditFields.asp?ProfileTypeID=<%=strProfileTypeID%>" target="main"><D
            End If
        End If
        If CheckAccess(strProfileID,"WriteAccess", false, strProfileTypeID)THEN
            If Application("MenuEditPopup") Then
                %><a HREF="javascript:void(0)" onClick="window.open('./EditFields.asp?ProfileID=<
            Else
                %><a HREF="./EditFields.asp?ProfileID=<%=strProfileID%>&ProfileTypeID=<%=strProfi
            End If
        End If
        If session("AdminStatus") >= 1000 And CheckAccess(strProfileID,"ReadAccess", false,strPro
            If Application("MenuSearchPopup") Then
                %><A HREF="javascript:void(0)" onClick="window.open('./SearchFields.asp?ProfileTy
            Else
                %><A HREF="./SearchFields.asp?ProfileTypeID=<%=strProfileTypeID%>" target="main">
            End If
        End If
    End If
    DataRS.MoveNext
Loop
CloseRecordset()
```

Figure 40 Part of Menu.asp

Now we can add the menu containing the recipe categories by calling the function we made in incLayout.asp. This function call we add at the red arrow in Figure 40.

```
'************ Start of commented area ****************
'Get MenuItems
Response.Write(GetLayout("MenuBar"))
'************ End commented area *********************
```

We also want a link in the menu so we are able to do a custom search.

This can be done either by adding a link to:

```
./SearchFields.asp?ProfileTypeID=17
```

This links tell the system that the search page should be opened for the profiletype with id 17, which is the profiletype we called Recipe. This can be seen at Figure 32 by the red arrow.

Another way which is a bit more complicated is to edit the line in Figure 40 marked with a green arrow, to the snippet below.

```
If CheckAccess(strProfileID,"ReadAccess",
false,strProfileTypeID) And strProfileTypeID <> 16 Then
```

This condition is a part of a loop going through all the different profiletypes starting at the pink arrow in Figure 40. This loop creates a link for creating a new profile, editing a profile, or search profiles of the given profiletype. The link is only displayed if the conditions marked with green and blue arrows in Figure 40 are met.

By removing the “session("AdminStatus") >= 1000 and” in Figure 40 marked with a green arrow. We allow other users than the administrator to have a search button in the menu. The search button is only displayed if the user has read access to the given profiletype.

In Figure 41 you can see the result of the menu after adding three recipe categories and doing a little formatting of the layout in the style sheet Menu.css that makes the “Login” in a smaller font, and separates it from the other part of the menu, so you can see it in the lower left corner of Figure 41.

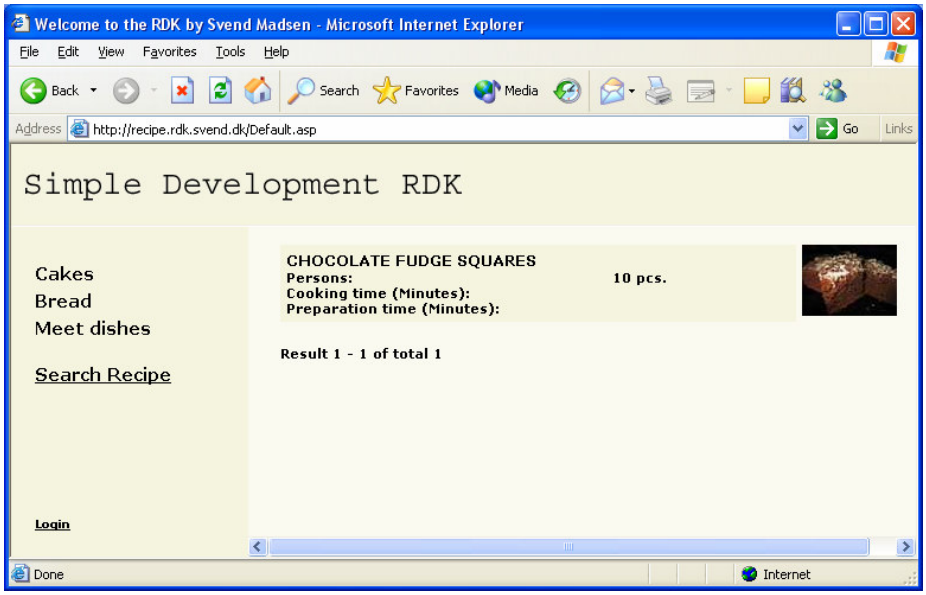


Figure 41 The resulting menu.

Now when we login and create a new recipe, we get the following screen pictured in Figure 42.

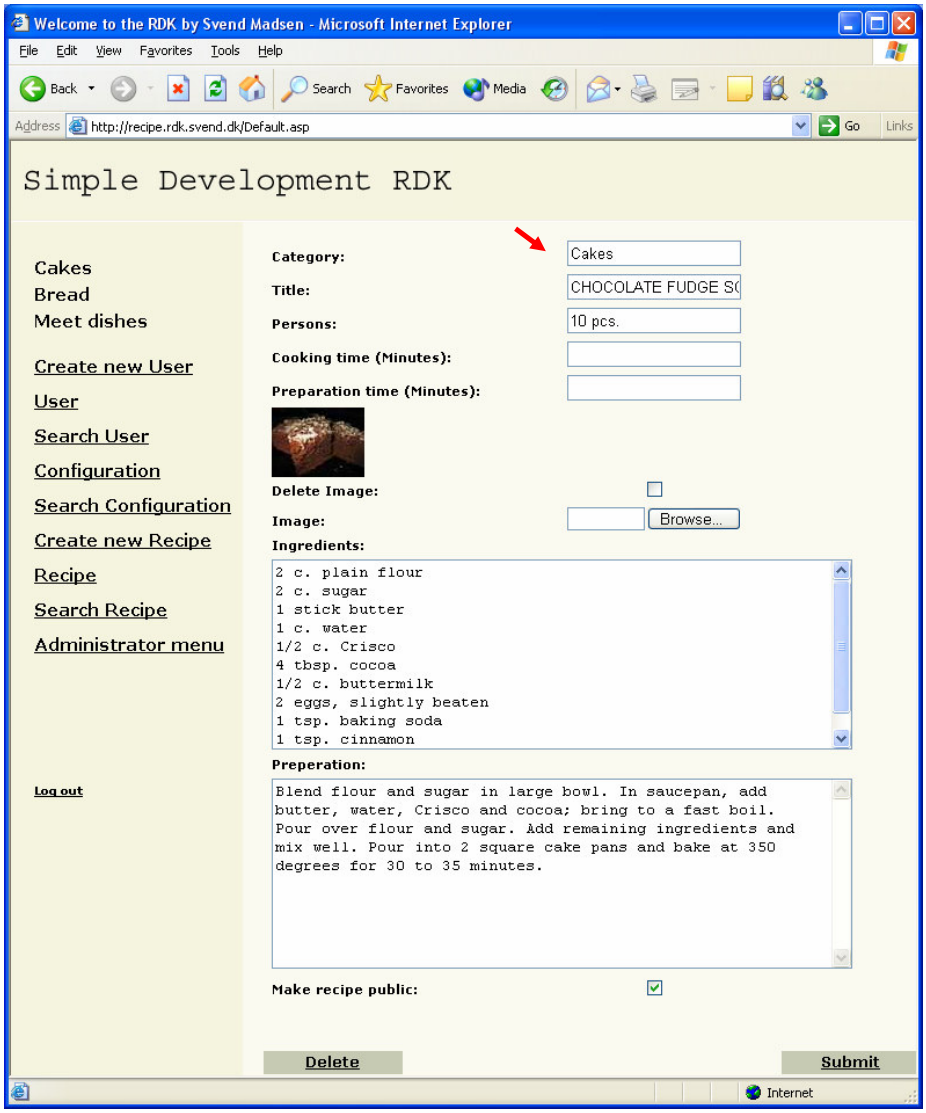


Figure 42 A new recipe has been created.

As we can see on Figure 42 marked by the red arrow, the category in the top is still just a text field as defined in the fielddefinition for the profiletype "recipe". We need to change this to a combo box so we can choose from the available categories corresponding to the items in the top of the menu to the left. How we do that will be described in the next section.

You should also notice that links for creating new users and recipes are added to the left menu. The items can be added and removed in the same way we added the search button for recipes in the previous section, editing the file Menu.asp.

EditFields.asp

To add a combobox instead of the textbox marked with the red arrow in Figure 42 we have to go and edit the ASP file EditFields.asp. We need to call the function GetLayout with the parameter “ComboMenu" which we created in the previous section describing incLayout.asp. Since the parameter “ComboMenu" only retrieves raw data we have to format it and put it into a combo box instead of the textbox marked with the red arrow in Figure 42.

```
<!--#include file="IncField.asp"--><%
    Do Until DataRS.EOF
        SELECT CASE DataRS("fldFieldID")
'************ Start of commented area ****************
'Example of how to keep non administrators from editing the date.
'       Case 100 'FieldID
'           If Session("AdminStatus") >= 1000 Then
'               WriteField DataRS("fldFieldName"), DataRS("fldFieldID"), DataRS("fldFieldListNo")
'           Else
'               WriteField "Show", DataRS("fldFieldID"), DataRS("fldFieldListNo"), DataRS("fldFie
'           End If
'************ End commented area *********************

'************ Start of commented area ****************
'Example on how to autogenerate a password
'       Case 100 'FieldID
'           If DataRS("fldData") <> "" Then
'               WriteField DataRS("fldFieldName"), DataRS("fldFieldID"),DataRS("fldFieldListNo"),
'           Else
'               Response.Write("<INPUT TYPE=Hidden NAME=ctrl"& DataRS("fldFieldID") &" VALUE='"&
'           End If
'************ End commented area *********************

'************ Start of commented area ****************
'Example of how to keep non administrators from viewing a field.
'       Case 100 'FieldID
'           If Session("AdminStatus") >= 1000  Then
'               WriteField "Show", DataRS("fldFieldID"),DataRS("fldFieldListNo"), DataRS("fldFiel
'           End If
'************ End commented area *********************
        CASE ELSE
            'If error found on server insert errorvalue
            If trim(strFieldID) = trim(DataRS("fldFieldID")) Then
                WriteField DataRS("fldFieldName"), DataRS("fldFieldID"),DataRS("fldFieldListNo"),
            Else
                WriteField DataRS("fldFieldName"), DataRS("fldFieldID"),DataRS("fldFieldListNo"),
            End If
        END SELECT
        DataRS.MoveNext
    Loop
    EndField()
    CloseRecordset()
    CloseConnection()
```

Figure 43 Code from EditFields.asp

This is done through inserting the code snippet below by the red arrow in Figure 43:

```
Case 244 'Add Categorycombo instead if textbox
      Dim strMenu, intLocation1, intLocation2
      strMenu = GetLayout("ComboMenu")
      intLocation1 = 2
      intLocation2 = InStr(strMenu,"~#")
      Do Until intLocation2 < 1
            WriteField "ComboBox", DataRS("fldFieldID"),
            DataRS("fldFieldListNo"),
            DataRS("fldFieldListGroup"), DataRS("fldString"),
            DataRS("fldMin"), DataRS("fldMax"),
            DataRS("fldRequired"), DataRS("fldData"),
            Mid(strMenu,intLocation1,intLocation2-
            intLocation1),
            Mid(strMenu,intLocation1,intLocation2-
            intLocation1), DataRS("fldErrorString")
            intLocation1 = intLocation2 +2
            intLocation2 = InStr(intLocation1+1,strMenu,"~#")
      Loop
```

A similar combo box has been made in the page we get when we press the "Search Recipe" button. To review this code you need to look in the ASP file SearchFields.asp in the /Tutorial directory of the installation CD.

To add the functionality where the display of a recipe is optional we have to look at the ASP file SearchFieldsGet.asp.

SearchFieldsGet.asp

In order to make the display of the recipe optional we have to link the value for the field called "Make field public" with the id 252, marked with the blue arrow in Figure 33, to the display of recipes in the search result.

```
    If intProfileTypeID <> 0 Then
        strDisplay = "true"
    End If
    Select Case DataRS("fldFieldID")
    'Case 100
    '    Do something
    Case Else
        strResponse = strResponse & strTmpResponse
    End Select
    strTmpResponse= ""
    intCurrentProfile = DataRS("fldProfileID")
    DataRS.MoveNext
    If DataRS.EOF then
        boolLoop = false
    else if intCurrentProfile <> DataRS("fldProfileID") then
            boolLoop = false
        else
            boolLoop = true
        end if
    end if
    strResponse = strResponse & vbcrlf
Loop
strResponse = strResponse & "</DIV>"
If intProfileTypeID <> 14  OR Session("AdminStatus") >= 10  Then
    strResponse = strResponse & "</a>"
End If

If Skip AND strDisplay = "true" Then
    If DispNr > (PageViews * PageNr) Then boolNext = true
    DispNr = DispNr + 1
else if (strDisplay = "true") Then
        Response.Write strResponse
        If DispStart = 0 Then DispStart = DispNr
        DispEnd = DispNr
        DispNr = DispNr + 1
    end if
```

Figure 44 Part of SearchFieldsGet.asp

In order to achieve this we only need to add a few lines of code shown below, to the ASP file SearchFieldsGet.asp, at the line marked with the red arrow in Figure 44:

```
Case 252    'Determin to display profile in searchresults
            'based on checkbox in fielddefinition
            strDisplay = DataRS("fldData")
```

The variable “strDisplay” which decides if a profile should be displayed in the search result is simply set to equal the value of the “Make field public” check box. A side effect of this is that even the administrator can not find the recipes which are not made public so we need to make the administrator able to do this when logged in. This is done by checking if the user has write access to the recipe and if so; make it visible in the search.

This is done by adding the following code to line marked with the green arrow in Figure 44.

```
'************ Start of commented area ****************
'If creator of profile or admin, display item even if not
'displayed in normal search
If CheckAccess(intCurrentProfile,"WriteAccess", false,0) Then
      strDisplay = "true"
End If
'************ End commented area ********************
```

To make the thumbnail images fit into the search result page shown in Figure 42 we need to change the default size of the thumbnail image. This is done in the ASP file EditFieldsSave.asp

EditFieldsSave.asp

In EditFieldsSave.asp is the function, "MakeResizedImages" that resizes images upon upload. The function is displayed in Figure 45:

```
Function MakeResizedImages(strField)
    '************ Start of commented area ****************
    ' Creating Different sizes of the image if the fieldname is matching
    ' This example is used for the logo
    If strField = "ctrl127" Then
        strResizeError= ResizeImage (Application("FieldImagePath") , strTemp, 500, 50,"thumb_", 1
        If strResizeError <> "" Then
            strError=strError + "</Field>"& replace(strField,"ctrl","") &"</Data>"& Server.URLEnc
        End If
        strResizeError= ResizeImage (Application("FieldImagePath") , strTemp, 500, 50,"medium_",
        If strResizeError <> "" Then
            strError=strError + "</Field>"& replace(strField,"ctrl","") &"</Data>"& Server.URLEnc
        End If
        strResizeError= ResizeImage (Application("FieldImagePath") , strTemp, 500, 50,"large_", 1
        If strResizeError <> "" Then
            strError=strError + "</Field>"& replace(strField,"ctrl","") &"</Data>"& Server.URLEnc
        End If
    '************ End commented area ********************
    Else
        strResizeError= ResizeImage (Application("FieldImagePath") , strTemp, 160, 120,"thumb_",
        If strResizeError <> "" Then
            strError=strError + "</Field>"& replace(strField,"ctrl","") &"</Data>"& Server.URLEnc
        End If
        strResizeError= ResizeImage (Application("FieldImagePath") , strTemp, 400, 300,"medium_",
        If strResizeError <> "" Then
            strError=strError + "</Field>"& replace(strField,"ctrl","") &"</Data>"& Server.URLEnc
        End If
        strResizeError= ResizeImage (Application("FieldImagePath") , strTemp, 800, 600,"large_",
        If strResizeError <> "" Then
            strError=strError + "</Field>"& replace(strField,"ctrl","") &"</Data>"& Server.URLEnc
        End If
    End If
    If Application("KeepOriginalImage") = false Then
        Dim fso, f
        Set fso = CreateObject("Scripting.FileSystemObject")
        Set f = fso.GetFile(Application("FieldImagePath")& strTemp)
        f.Delete
    End If
End Function
```

Figure 45 Part of the file EditFieldsSave.asp

As you can see in Figure 45 at the end of the line marked with a red arrow, the current size of the images are 160 x 120 we need to change these values to 80 and 60 in order to have the images fit into the search results as displayed in Figure 41. The line should be replaced with the one below:

```
strResizeError= ResizeImage (Application("FieldImagePath") ,
strTemp, 80, 60,"thumb_", 50)
```

The number 50 at the end of the code snippet above is a value telling the percentage of compression to use when recompressing the image.

The layout of the recipe website

The only thing we need now is to edit the style sheets for the search results SearchFieldsGet.css and the style sheet for the display of each recipe ShowProfile.css if we want to have another layout than the one given automatically by the RDK.

7.4.3 Gains of using the RDK

Now we have made a website for displaying recipes, we have a login for the administrator of the site, who is able to add recipes to the system. If we want, we can make it possible for all users to add recipes. The only thing needed is to change the value of create access form “1000” to “0” in Figure 32 chapter 7.4.1. A default value of the check box “Make recipe public” could be set to false so the administrator had the opportunity to validate each recipe before making it public.

We have achieved making this recipe website without having knowledge of database management, creation of tables and SQL or Stored Procedures.

The developer have not needed to worry about either image upload functions, recompression. The resizing of the images has only needed a change of two values because the standard thumbnail size was too large.

User management for validating the administrator, has only been a matter of configuring the proper access rights for the different profiletypes. And the developer has not even needed to think about form validation.

The website this tutorial is based on http://ullasopskrifter.dk, took only 4 hours to implement, excluding addition of the actual recipes, and has been proven robust. There have been no complaints from the customer. The only automatic error handling emails received as described in chapter 5.5, has been on two occasions when the MSSQL server has been rebooted, and the web server was not able to contact it.

8 Practical implementation

When creating a data model for websites it is hard to take all future changes into consideration. With my experience making databases for several websites and intranets and trying to make changes due to post production specification changes, I found that the changes needed were very comprehensive. First of all the tables in the database have to reflect the changes, this can result in addition of tables, rows or changes in data types and relation. Unfortunately it is extremely rare that you do not need to change the other layers in the application, so you need to change the Stored Procedures to reflect the changes in the data model. Often the execution of the Stored Procedures is very sensitive to changes in the data model, so you need to do a comprehensive testing after these changes. So now we have done all the changes in the DBMS, then our changes need to be done on the application level, a testing of these, and finally the GUI has to reflect the changes. This procedure is very time consuming. So this is where a well written RDK can assist you in enhancing development time and cost. The aim with implementing this RDK is that you only need to edit two layers in the application one is the business logic and the other is the layout. This will lead to no messing around with database tables and Stored Procedures.

8.1 Choice of technologies

Before starting to implement the RDK I needed to make a decision on what technologies to use. As DBMS I decided to use MSSQL[40], and ASP[41] as server side scripting language. Furthermore I had to decide how to separate content from layout in ASP and found the use of CSS to be the best way to do this. The main reason for the choices of technologies was that I feel most competent in using these technologies, combined with the popularity of these technologies in many companies. I also did some research on where I could host a website build on these technologies, and I found that many hosting providers supported these. So I would not have any problems in finding a suitable hosting provider for a reasonable price.

I found that it was not recommendable to implement the business logic in Dynamic Link Libraries (DLL)'s[42] because many hosting providers do not allow you to install custom made DLL's. And those who do often charge you extra for it, because they need to review the source code to make sure that they are not installing any hostile code on their web servers. The choice of using MSSQL as DBMS will restrict the suitable hosting providers a bit, but the only alternative that supported Unicode and transactions was Oracle, which is supported by very few hosting providers. It is very important that the DBMS supports Unicode for making multilingual websites. The support of transactions is possible, but not that important. It is a nice feature to have, to avoid errors in certain business critical parts of the application. Furthermore I have chosen to support Microsoft Internet explorer 4.0 and above as Internet browsers. This choice is based on the Table 3 below.

---

[40] MSSQL is discussed in section 3.1.2.

[41] ASP is discussed in section 3.2.2.

[42] DLL's are small executable programs that can help separating the business logic from the layout.

Browser	Usage
Microsoft IE 6.0	66.3%
Microsoft IE 5.5	14.5%
Microsoft IE 5.0	12.7%
Mozilla	1.6%
Microsoft IE 4.0	0.8%
Netscape Navigator 4	0.6%
Opera 6.0	0.6%

Table 3 Statistics of Internet browser usage from July 28 2003 source [18]

From Table 3 you can see that the Mozilla browser only has 1,6 percent of the market compared to for example users with VGA resolution (640x480) which is 1~2 percent. Please have a look at [19] for further information on browser statistics. It is important to take into consideration both support for browser maker, browser version and screen resolution when designing websites. The reason why I decided to support Internet explorer 4.0 which has a smaller part of the market than Mozilla is due to the fact that the majority of layout interpretation and functions supported by Internet Explorer 5.0 is also supported by Internet Explorer 4.0. The XGA resolution (800x600) is chosen as the minimum supported resolution, though it will be possible for the developer of the specific application to adapt the RDK to smaller screen resolutions by changing the style sheets.

8.2 System design

Developing the RDK has been an iterative process. First the initial design and development of the RDK was made, and then the first implementation of a specific website with the use of the RDK. This implementation helped me to correct errors, and to find new elements that were candidates for the RDK. These elements were then implemented into the RDK. Every time the RDK has been used in a new implementation, new elements that could be used to expand the RDK have been discovered, as well as errors to correct. The iterative process has been done approximately 6 times. One of these implementations has been a basis for the tutorial in chapter 7. It is important that the elements added to the RDK is not too business specific. If so, you will end up spending a lot of time on implementing a component that you will never use again.

8.2.1 Required features

In chapter 5 I made a list of common features desired in a RDK. These features can be implemented in many different ways. In this section the chosen strategies for implementing the different features will be described.

8.2.1.1 User management

In the user management we need to have some kind of administrator level, to set the users ability to administer or edit in the data for the given application. The number of fields describing the user should be variable, so it is possible to have a user in one application, with name, email and password. In another application maybe the user is only identified with fingerprint and social security no. There should also be an interface for the administrator to manage the users.

8.2.1.2 Security

When creating the application we need to be able to define what user level or group should be able to read, write, delete or create an object, and we also need to define the same rights for an object owner. If a user or administrator needs to assign rights to a specific user, we need to be able to do that as well. E.g. a user wants to give another user the rights to add images to an image album created by the first user.

Specific to MSSQL a way to enhance the security is to create one user for the administration of the database who has access to all tables and procedures. And another user who has access via the web application if the hosting provider supports multiple users for each web application. The web user should only have access to the Stored Procedures used by the web application, and no access to the tables.

MSSQL does not support native support for table data encryption. To allow encryption of table data we need to allow the developer to easily encrypt this data, so people with unauthorized access to the database tables is not able to read sensitive data such as credit card numbers and user passwords.

A way to enhance security in ASP is not to have any passwords for the database in the asp files, and if possible neither in the Global.asa file. There are known exploits that can list the contents of asp files and one has also been able to list the contents of the Global.asa file. It is recommended to keep the database password either in the system registry or in a DLL file, which can be accessed from Global.asa.

8.2.1.3 Picture upload

I have done a lot of research on components for picture upload, and I was not able to find any that supported handling of Unicode characters in the same form as the file upload form field[43]. But I found an "asp only" upload script, which I was able to edit so it supported Unicode.

When the picture has been uploaded it has to be resized and recompressed as described in chapter 5.3. For this task I have purchased a component called AspImage from ServerObjects, which is well suited for this task.

8.2.1.4 Form validation

When doing the form validation, it is desired if anything is mistyped or missing, the application needs to move focus to the incorrect form item, and come up with a descriptive error message. This should enhance the usability of the system. The use of different form items should, from the developer point of view, distinguish between client side or server side validation. But the end user should not feel any difference.

8.2.1.5 Error handling

If the hosting provider supports it, the RDK should make use of custom error pages, instead of the standard non user friendly pages described in chapter 5.5. This page should send an e-mail to an address specified by the developer so the developer or website administrator gets notified about the error that occurred.

[43] When uploading files you use a HTML form which often also contains other information passed to the server.

8.2.1.6 Database and search

Searching in the web application should be easy for the developer to implement; it should be possible to select what form fields should be enabled in the search. All this should be achieved without the need for the developer to write any SQL code.

8.2.1.7 Multilingual support

The multilingual support will be seen throughout the whole RDK. When the developer defines the different form fields, they should be given names in the different languages the implemented application supports. The specific translation should be possible by a translator with no programming experience.

8.2.2 The four tier approach

I have decided to implement the RDK in four tiers; many applications are implemented in three tiers, the database, business logic and the user interface. But the three tier model forces you to write SQL code in the business logic, which makes the business logic disorganized, and with MSSQL you can achieve much better performance by writing the SQL code in Stored Procedures, that are compiled and executed on a database level. This leads to the business logic being divided into two parts, one part in the Stored Procedures and the other part in the ASP code. A description of the four tiers can be seen on Figure 46.

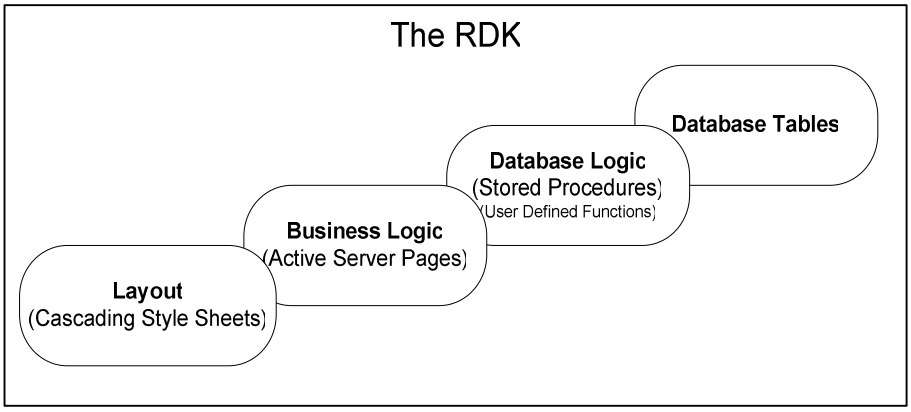


Figure 46 Model of the four tiers of the RDK.

In the following I will describe each of the four tiers: The database tables and relations, the Stored Procedures and a few user defined functions, the ASP code and at last the layout.

8.2.3 Database Tables

While designing the data model for the RDK, I have forced myself to do a very high level of abstraction edging to the extreme. It is very important that the RDK is so flexible that the developer will not feel restricted using it. When one of the goals in making this RDK, is to allow the developer to make advanced websites without needing to edit the tables and the Stored Procedures of the system, I need to compromise on some performance areas. But if you take an unskilled developer trying to do a good database design, you will definitely get a worse performance of the database.

Below in Figure 47 you can see the overall table design. The figure is divided into several parts named A-E which is described beneath the figure. The relations between the tables are one-to-many relations, where the yellow key represents the “one” in the relation.

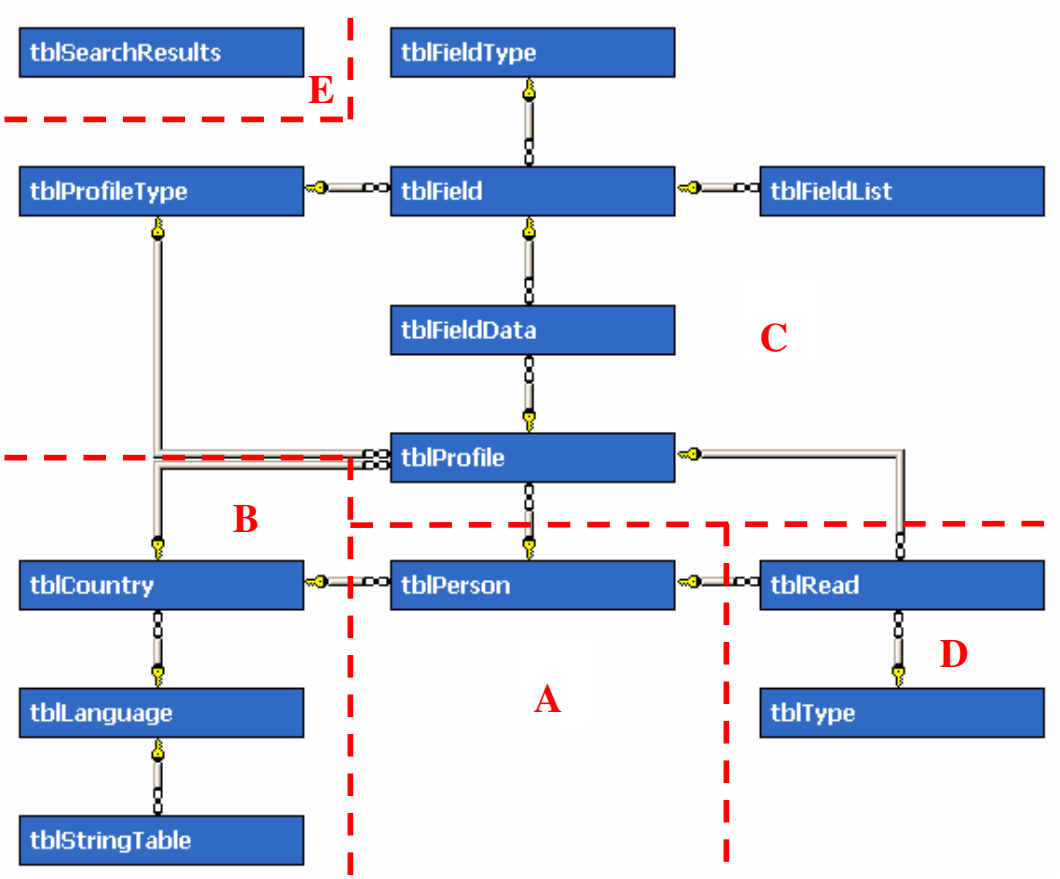


Figure 47 The overall table design.

The five major parts of the data model is described below. I will refer to figures in the developer manual section 6.2.1. In order to give you a better idea of what the resulting data of the tables is.

But first I will explain some of the data types of MSSQL that might not be known to the reader.

bit: The same as Boolean, 1 represents True and 0 False

varchar: A string of variable length.

nvarchar: A Unicode string of variable length.

Nearly all tables have a unique key named as the table, but with fld as prefix instead of tbl, and with ID as postfix.

8.2.3.1 Area A: The users

This is where all physical users are handled. The properties for each user are described tblPerson.

tblPerson

tblPerson is the table that stores the system data of the users, one row in this table describes one user. In Figure 48 you can see the column names and data types of tblPerson.

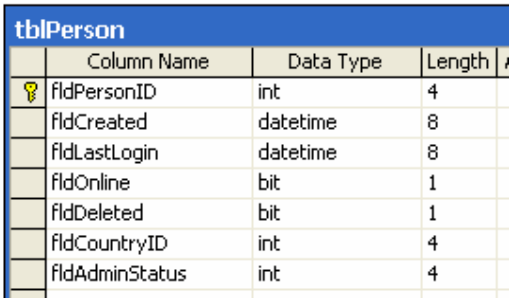


Figure 48 tblPerson

fldPersonID: Is the key for each user.

fldCreated: Stores the date of the creation of the user.

fldLastLogin: Stores the date of the last login to the system by the user.

fldOnline: Represents if the specific user is online.

fldDeleted: Makes it possible to delete a user without actually deleting the data.

fldCountryID: Reference to a specific country in tblCountry which makes it possible to retrieve the same language every time the user logs on to the system.

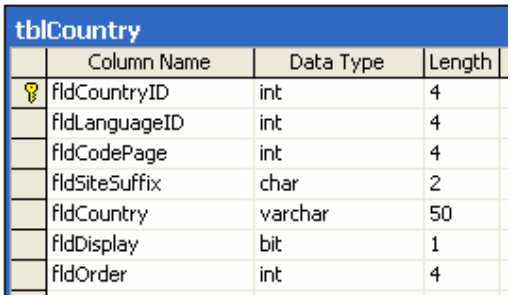
fldAdminStatus: Is an integer representing the access level, this level is used to ensure that the user only access the objects/profiles allowed.

8.2.3.2 Area B: The multilingual part

This is where all data related to the multilingual design is stored. tblStringTable is where all text strings for the RDK are stored and all languages are defined in tblLanguage. In tblCountry the countries are defined. Each country can be assigned a specific language. A user from tblPerson can be assigned a default country. Figure 24 in the developer manual displays the contents of these tables.

tblCountry

In Figure 49 you can see the column names and data types of tblCountry. One row in tblCountry describes the characteristics of a specific country.



tblCountry

Column Name	Data Type	Length
fldCountryID	int	4
fldLanguageID	int	4
fldCodePage	int	4
fldSiteSuffix	char	2
fldCountry	varchar	50
fldDisplay	bit	1
fldOrder	int	4

Figure 49 tblCountry

fldCountryID: Is the key for each country.

fldLanguageID: Reference to the prime language spoken in this country.

fldCodePage: The default code page for this country.

fldSiteSuffix: The default abbreviated name for this country e.g. uk or dk.

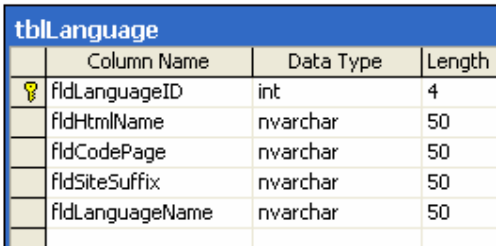
fldCountry: The name of the country.

fldDisplay: Defines if the country should be displayed when the Stored Procedure spGetCountries is called.

fldOrder: Defines the order of the languages when the Stored Procedure spGetCountries is called.

tblLanguage

One row in tblLanguage describes the characteristics of a specific language which is supported by the developer's implementation. In Figure 50 you can see the column names and data types of tblLanguage.



tblLanguage

	Column Name	Data Type	Length
🔑	fldLanguageID	int	4
	fldHtmlName	nvarchar	50
	fldCodePage	nvarchar	50
	fldSiteSuffix	nvarchar	50
	fldLanguageName	nvarchar	50

Figure 50 tblLanguage

fldLanguageID: Is the key for each language.

fldHtmlName: Is the name of the language used in the HTML doctype[44] definition e.g. en or da.

fldCodePage: The default code page for this language.

fldSiteSuffix: The default abbreviated name for this language e.g. en or da.

fldLanguageName: The name of the language

[44] The doctype definition is on top of a well formatted HTML page e.g. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 FINAL//en>

tblStringTable

tblStringTable is the table where all predefined text is put, This is all from error messages to printed text on the resulting pages. It is only the developer or translator may edit the contents of this table. In Figure 50 you can see the column names and data types of tblStringTable.

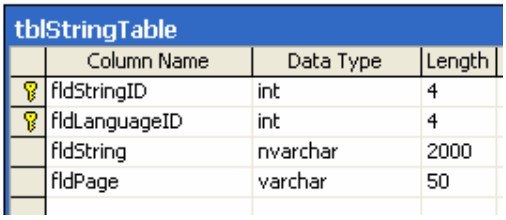


Figure 51 tblStringTable

fldStringID: This is the id used to reference to a specific string.

fldLanguageID: The id that specifies the language of the giving string.

fldString: The specific string.

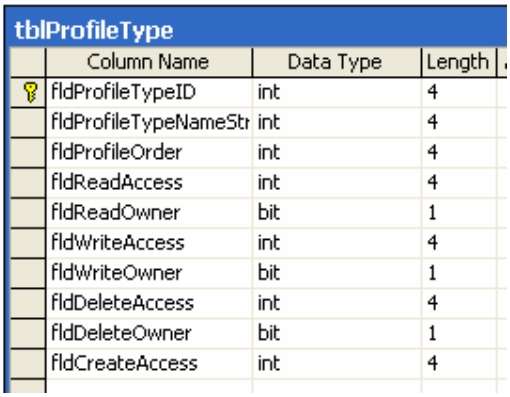
fldPage: A reminder for the developer on in what ASP file the text is used.

8.2.3.3 Area C: The core data

The majority of data is RDK lies here. A profiletype is located in tblProfileType (Figure 8 in the developer manual). This could be the general describer for a recipe on a website with a collection of food recipes. The properties such as ingredients and title are defined in tblField (Figure 9 and Figure 10 in the developer manual). Each field has a type defined in tblFieldType (Figure 11 to Figure 22 in the developer manual), e.g.: The title of the recipe would be defined as the field type named TextBox, since it is a regular text, which is represented as a normal textbox when entering new data. tblProfile represents the actual profiles. This could be the recipe of a chocolate cake; tblFieldData will then contain the actual name such as the title “Chocolate Cake”. The tblFieldList is made to represent choices such as in a combo box or option buttons.

tblProfileType

The profiletypes are contained in tblProfileType. This could be the definition of a recipe for a food recipe website. In Figure 52 you can see the column names and data types of tblProfileType.


tblProfileType

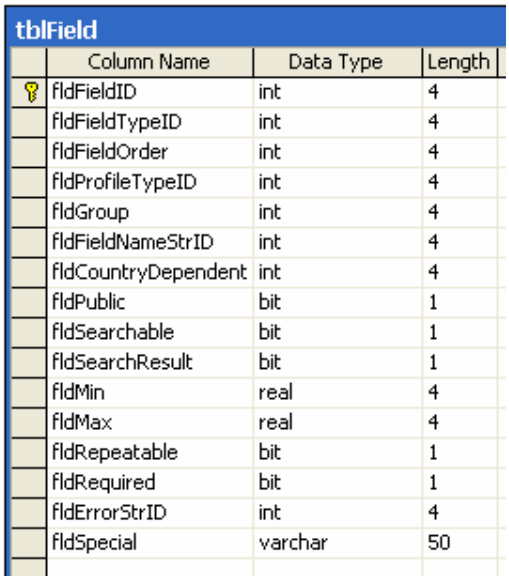
	Column Name	Data Type	Length
🔑	fldProfileTypeID	int	4
	fldProfileTypeNameStr	int	4
	fldProfileOrder	int	4
	fldReadAccess	int	4
	fldReadOwner	bit	1
	fldWriteAccess	int	4
	fldWriteOwner	bit	1
	fldDeleteAccess	int	4
	fldDeleteOwner	bit	1
	fldCreateAccess	int	4

Figure 52 tblProfileType

fldProfileTypeID:	Is the key for each profile type.
fldProfileTypeNameStrID:	Is the name of the profile type and refers to tblStringTable.
fldProfileOrder:	Defines the sorting of profile types.
fldReadAccess:	Defines the minimum AdminStatus number to read the profile, based on fldAdminStatus in tblPerson.
fldReadOwner:	Defines if the owner should have read access to profiles of this type, based on fldAdminStatus in tblPerson.
fldWriteAccess:	Defines who should have write access to profiles of this type, based on fldAdminStatus in tblPerson.
fldWriteOwner:	Defines if the owner should have write access to profiles of this type, based on fldAdminStatus in tblPerson.
fldDeleteAccess:	Defines who should have delete access to profiles of this type, based on fldAdminStatus in tblPerson.
fldDeleteOwner:	Defines if the owner should have delete access to profiles of this type, based on fldAdminStatus in tblPerson.
fldCreateAccess:	Defines who should be able to create a new profile of this type, based on fldAdminStatus in tblPerson.

tblField

tblField contains the fielddefinitions of the fields for a profiletype. This could be the title of a recipe. In Figure 53 you can see the column names and data types of tblField.



tblField

Column Name	Data Type	Length
fldFieldID	int	4
fldFieldTypeID	int	4
fldFieldOrder	int	4
fldProfileTypeID	int	4
fldGroup	int	4
fldFieldNameStrID	int	4
fldCountryDependent	int	4
fldPublic	bit	1
fldSearchable	bit	1
fldSearchResult	bit	1
fldMin	real	4
fldMax	real	4
fldRepeatable	bit	1
fldRequired	bit	1
fldErrorStrID	int	4
fldSpecial	varchar	50

Figure 53 tblField

fldFieldID: Is the key for each field.

fldFieldTypeID: Id referring to tblFieldType defining the type of the field.

fldFieldOrder: Defines the sort order of the field.

fldProfileTypeID: Referrers to the profile type the field ties to.

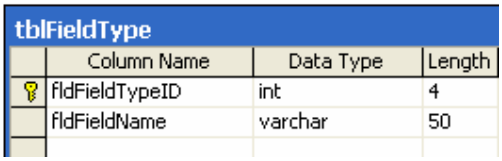
fldGroup: Fields for a specific profile type can be grouped together. This field defines the group number of the field, if any. The group is only used for the graphical representation on the website.

fldFieldNameStrID: Is the name of the field and referrers to tblStringTable.

fldCountryDependent:	You can set this value to 1 if you need to list different default values in a list or combo box depending on the country.
fldPublic:	Defines if a field should be available when retrieved by the search Stored Procedure spGetSearch or the default Stored Procedure to retrieve profiles spGetProfileData.
fldSearchable:	Defines if the field should be shown in the default search page for the related profile.
fldMin:	Defines the minimum value for the field. Described earlier in section 6.2.1.
fldMax:	Defines the maximum value for the field.
fldRepeatable:	If set to 1 this field can be repeated. Described earlier in section 6.2.1.
fldRequired:	Defines if this field needs to be filled out.
fldErrorStrID:	Is the default error message if the data entered into the field is invalid. Refers to tblStringTable.
fldSpecial:	Is used to define if the field is a special field, such as the login, the password or an encrypted field.

tblFieldType

In tblFieldType the individual types of fields are named. This naming is used to make the rest of the system aware of which type of field we are dealing with, be it a text box or a currency field. In Figure 54 you can see the column names and data types of tblFieldType.



tblFieldType

	Column Name	Data Type	Length
🔑	fldFieldTypeID	int	4
	fldFieldName	varchar	50

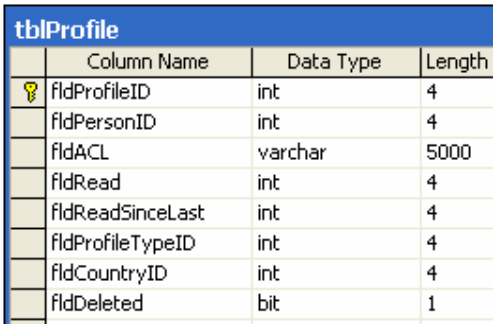
Figure 54 tblFieldType

fldFieldTypeID: Is the key for each field type.

fldFieldName: The common name for the field type.

tblProfile

tblProfile contains the system data for individual profiles created in the system. This could be a specific recipe called “Chocolate Cake”, for a website for food recipes. In Figure 55 you can see the column names and data types of tblProfile.



tblProfile

	Column Name	Data Type	Length
🔑	fldProfileID	int	4
	fldPersonID	int	4
	fldACL	varchar	5000
	fldRead	int	4
	fldReadSinceLast	int	4
	fldProfileTypeID	int	4
	fldCountryID	int	4
	fldDeleted	bit	1

Figure 55 tblProfile

fldProfileID: Is the key for each profile.

fldPersonID: Is the key for the person that owns the profile, usually the creator.

fldACL: A text field containing the access control list for this profile.

fldRead: Shows how many times the profile has been read.

fldReadSinceLast: Shows how many times the profile has been read since last time the owner has logged on to the system.

fldProfileTypeID: Defines the specific type of the profile.

fldCountryID: Defines the country of the owner when the profile was created. This value can be used to recognize what country and language the data of the profile was written in.

fldDeleted: Allow the profile to be deleted without the data being erased.

tblFieldData

This table contains all data entered into the RDK by all the users, this could be the details of the recipes for a website with food recipes, for example the string “Chocolate Cake” in one row and the category that the recipe belongs to “Cakes” in another row. In Figure 56 you can see the column names and data types of tblFieldData.

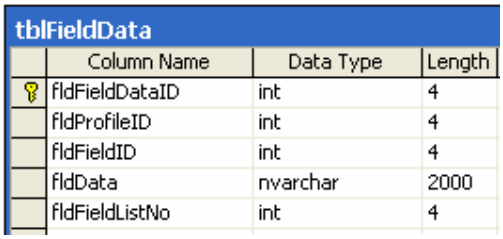


Figure 56 tblFieldData

fldFieldDataID: Is the key for each field related data.

fldProfileID: Is the key for the profile the field data relates to.

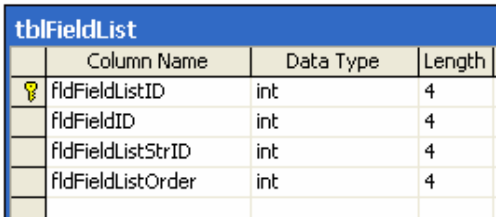
fldFieldID: Is the key for the field the field data relates to.

fldData: This is a Unicode field containing all entered data into the system that is not predefined in tblStringTable, like the title of a recipe.

fldFieldListNo: Contains the number in the list if the related field is defined as repeatable in tblField.

tblFieldList

tblFieldList is used to contain the default options that can be selected in an option or combo box .In Figure 57 you can see the column names and data types of tblFieldList.



tblFieldList

	Column Name	Data Type	Length
🔑	fldFieldListID	int	4
	fldFieldID	int	4
	fldFieldListStrID	int	4
	fldFieldListOrder	int	4

Figure 57 tblFieldList

fldFieldListID: Is the key for each field list item.

fldFieldID: Is the key to the corresponding field.

fldFieldListStrID: Is the reference in tblStringTable to the text of the field list item.

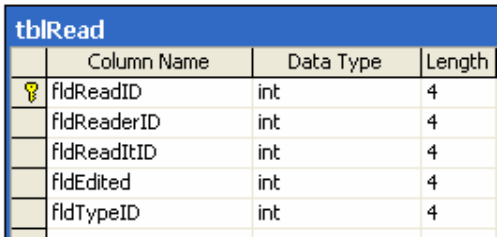
fldFieldListOrder: Defines the sort order of the list item.

8.2.3.4 Area D: User statistics

In area D every time a profile is read there will be added a record. This record can have a specific type defined in tblType, such as “Read from search” or “Read from link” if the profile is linked directly from another site. These tables are only made to gather statistics of how the users are exploring the website.

tblRead

In this table statistics are kept, to track which person has read or edited what profile. In Figure 58 you can see the column names and data types of tblRead.



tblRead

Column Name	Data Type	Length
fldReadID	int	4
fldReaderID	int	4
fldReadItID	int	4
fldEdited	int	4
fldTypeID	int	4

Figure 58 tblRead

fldReadID: Is the key for each read record.

fldReaderID: Reference to the person that has read the profile.

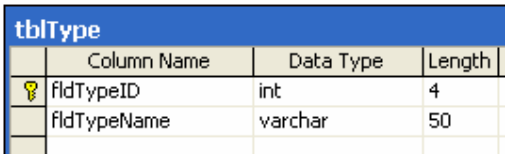
fldReadItID: The profile that has been read.

fldEdited: Is one if the profile was edited by the person that read it.

fldTypeID: Reference to tblType (described next) defining how the profile was read.

tblType

tblType simply is an identifier for how the profile has been read. This could be through a search or through the actual view of a profile. In Figure 59 you can see the column names and data types of tblType.



tblType

Column Name	Data Type	Length
fldTypeID	int	4
fldTypeName	varchar	50

Figure 59 tblType

fldTypeID: Is the key for each read type record.

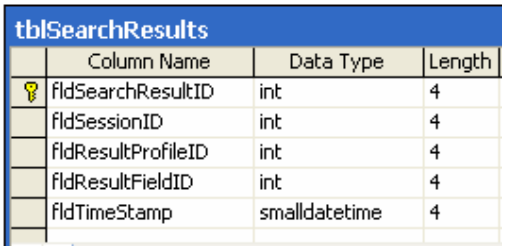
fldTypeName: Common name of how the profile have been read or accessed. E.g. via search or linked directly from another website.

8.2.3.5 Area E: Temporary search results

This table is made to store temporary search data in. The reason why a table for temporary search results is needed, is because we would get a very bad performance without it.

tblSearchResults

This table is used as a place to sort out temporary search results, for example when a user is searching for cakes in a website with food recipes. In Figure 60 you can see the column names and data types of tblSearchResults.



tblSearchResults

	Column Name	Data Type	Length
🔑	fldSearchResultID	int	4
	fldSessionID	int	4
	fldResultProfileID	int	4
	fldResultFieldID	int	4
	fldTimeStamp	smalldatetime	4

Figure 60 tblSearchResults

fldSearchResultID: Is the key for each search result.

fldSessionID: The ASP session id of the current user.

fldResultProfileID: The reference to the profile turning up in the result.

fldResultFieldID: The reference one field that is matching the search.

fldTimeStamp: A simple time stamp in order to clean up the right search results.

8.2.4 Stored Procedures

In the following I will describe all the Stored Procedures used by the RDK. Generally they can be divided in two groups. First all the Stored Procedures used by the system in production, and then all the Stored Procedures used by the developer for configuring the RDK, all the Stored Procedures for the developer I prefixed spAdm.

8.2.4.1 Stored Procedures used in production

spAddPerson

This Stored Procedure is used to add a new person to the system by adding a row to tblPerson.

Input variables:

intCountryID AS Int	The default country for the created person.
intAdminStatusID AS Int = 0	Optional, sets the AdminStatus.

Returns:

fldPersonID	Returns the identity of the newly created person.

spAddProfile

This Stored Procedure is used to add a new profile to the system, the rights for creating a new profile is also checked with the Stored Procedure spCheckAccess described later in this section. The call to spAddProfile results in an added row in tblProfile.

Input variables:

intPersonID AS Int	The id of person that should own the profile.
intProfileTypeID AS Int	The id representing the type of profile.
intCountryID AS Int	The id of the owners country.
intCreatePersonID AS Int = 0	Optional value: If the person that creates the profile is different than the owner of the profile, this should be set.

Returns:

fldProfileID	Returns the identity of the newly created profile.

spGetProfileData

This Stored Procedure retrieves the data for a given profile after checking if the user has access to it. The rights for reading the profile are also checked with the Stored Procedure spCheckAccess described later in this section. The data is mainly retrieved from tblFieldData.

Input variables:

nvchEncPass As NVarChar(2000)	The password for retrieving encrypted data.
intCountryID As Int	The id of country leading to the language that the data and descriptions are returned in.
intPersonID As Int	The id of person that want to display the profile.

intProfileID As Int	The id of the profile to be displayed.

Returns zero or more rows with the following fields:

fldFieldTypeID	The id referencing to the field type.
fldFieldID	The id of the field.
fldFieldListNo	The number in the list if repeated.
fldFieldListGroup	The group number used if repeated.
fldFieldName	The name representing the field.
fldString	The common name of the field.
fldFieldListStrID	The id of the list item if present.
fldListString	The common name of the list item if present.
fldData	The data for the field.
fldProfileID	The id of the related profile.
fldFieldOrder	The order of the field in comparison to the other fields.
fldFieldListOrder	The order of the list item if present in comparison to the other list items.
fldPersonID	The id of the person that owns the profile related to the field.

spSetDeleteProfile

This Stored Procedure is used to mark a profile as deleted. The rights for deleting the profile are also checked with the Stored Procedure spCheckAccess described later in this section.

intPersonID Int	The id of person that wants to delete the profile.
intProfileID Int	The id of the profile to be deleted.

Returns nothing.

spGetProfiles

This Stored Procedure is used to get a sorted list of all profiles owned by the person matching the input value intPersonID. If there is a profile type where the person does not have a profile in, one row is returned for this profile type with the profileID NULL.

Input variables:

intCountryID As Int	The id of country leading to the language that the profile names are returned in.
intPersonID AS Int	The id of person that wants to retrieve the profiles.

Returns zero or more rows with the following fields:

fldProfileTypeID	The id representing the type of profile.
fldString	The common name for the profile.
fldProfileID	The id of the profile to be displayed.
fldProfileOrder	The order of the profile in comparison to the other profiles.

spGetProfileTypes

This Stored Procedure is used to get a sorted list of all profiletypes in the RDK.

Input variables:

intCountryID As Int	The id of country leading to the language that the profiletype names are returned in.

Returns zero or more rows with the following fields:

fldProfileTypeID	The id representing the type of profile.
fldString	The common name for the profile.
fldProfileOrder	The order of the profile in comparison to the other profiles.

spSetACL

This Stored Procedure is used to set a new Access Control List for a profile. The rights for writing to the profile are also checked with the Stored Procedure spCheckAccess described later in this section.

Input variables:

intPersonID Int	The id of person that want to edit the ACL.
intProfileID Int	The id of the profile that should have edited the ACL.
txtData VarChar(5000)	The new ACL.

Returns nothing.

spAddFieldData

This Stored Procedure is used to add data to a profile. The rights for creating a new profile are also checked with the Stored Procedure spCheckAccess described later in this section. The call to this Stored Procedure results in a row added to tblFieldData.

Input variables:

nvchEncPass As NVarChar(2000)	The password for encrypting the data if needed.
intPersonID Int	The id of person that wants to add the data.
intProfileID Int	The id of the profile to add data to.
intFieldID Int	The id of the field to add.
txtData NVarChar(2000)	The data to add.
intFieldListNo int = NULL	If it is a repeatable field, this represents the number in the list.

Returns nothing.

spSetFieldData

This Stored Procedure is used to write data to a profile. If the data does not already exist spAddFieldData is called for this Stored Procedure. The rights for writing to a profile are checked with the Stored Procedure spCheckAccess described later in this section. The call to this Stored Procedure results in a row being edited in tblFieldData.

Input variables:

nvchEncPass As NVarChar(2000)	The password for encrypting the data if needed.
intPersonID Int	The id of person that want to write the data.
intProfileID Int	The id of the profile to write data to.
intFieldID Int	The id of the field to overwrite.

txtData NVarChar(2000)	The data to write.
intFieldListNo Int = NULL	If it is a repeatable field this represents the number in the list.

Returns nothing.

spSetCheckFieldExists

This Stored Procedure is used to check if a specific value already exists, such as a login. If bitWrite is set to 1, spSetFieldData is called to add the data.

Input variables:

nvchEncPass As NVarChar(2000)	The password for encrypting the data if needed.
intPersonID Int	The id of person that want to check/add the data.
intProfileID Int	The id of the profile to check/add data to.
intFieldID Int	The id of the field to check/add.
txtData nVarChar(2000)	The data to check/add.
bitWrite Bit	Indicate to write the field if it does not exists.

Returns:

fldOK	Is 1 if the field the field did not exist.

spGetFieldData

This Stored Procedure is used to retrieve the fields needed to create the form for creating a new or for editing an existing profile. The rights for writing to or creating a new profile are checked with the Stored Procedure spCheckAccess described later in this section. . The call to this Stored Procedure is mainly returning data from tblField and tblFieldData.

Input variables:

nvchEncPass AS NVarChar(2000)	The password for decrypting the data if needed.
intCountryID As Int	The id of country leading to the language that the data and descriptions are returned in.
intProfileTypeID As Int	The id of the profile type to be displayed.
intProfileID As Int	The id of the profile to be displayed.
intPersonID As Int	The id of person that want to retrieve the data.

Returns zero or more rows with the following fields

fldFieldID	The id of the field.
fldFieldListNo	The number in the list if repeated.
fldFieldListGroup	The group number used if repeated.
fldFieldName	The name representing the field.
fldMin	The minimum value of the field.
fldMax	The maximum value of the field.

fldRequired	Indicating if the field is required to be filled out.
fldErrorString	The string containing a general error message for the field.
fldString	The common name of the field.
fldFieldListStrID	The id of the list item if present.
fldListString	The common name of the list item if present.
fldData	the data for the field.
fldProfileID	The id of the related profile.

spSetDeleteField

This Stored Procedure is used to delete one of the fields of a repeatable field. The rights for writing to the profile are checked with the Stored Procedure spCheckAccess described later in this section.

Input variables:

intPersonID Int	The id of person that wants to delete the field.
intProfileID Int	The id of the profile containing the deleted field.
intGroup Int	The group of the field to be deleted.
intFieldListNo Int	The order of the field in comparison to the other fields.

Returns nothing.

spAddFieldListItem

This Stored Procedure is used to add an additional field to a field that is repeatable. The rights for writing to the profile are checked with the Stored Procedure spCheckAccess described later in this section.

Input variables:

intPersonID Int	The id of person that wants to add a field list item.
intProfileID Int	The id of the profile to where the added field list item is in.
intGroup Int	The group of the field to be added.
intFieldListNo int	The order of the field in comparison to the other fields.

Returns nothing.

spGetSearchFields

This Stored Procedure is used to retrieve the fields needed to create the form to search through profiles of a certain type. For example when the user wants to search for a recipe in a website with food recipes.

Input variables:

intCountryID As Int	The id of country leading to the language that the field descriptions are returned in.
intProfileTypeID As Int	The id of the profile type to retrieve the search fields for.

Returns zero or more rows with the following fields

fldFieldID	The id of the field.
fldFieldName	The name representing the field.
fldMin	The minimum value of the field.
fldMax	The maximum value of the field.

fldRequired	Indicating if the field is required to be filled out.
fldErrorString	The string containing a general error message for the field.
fldString	The common name of the field.
fldFieldListStrID	The id of the list item if present.
fldListString	The common name of the list item if present.

spGetSearch

This Stored Procedure is used to retrieve data from a requested search. For example to return the data of the chocolate cake and cheese cake when the user is searching for cakes in a food recipe website.

Input variables:

nvchEncPass AS NVarChar(2000)	The password for decrypting the data if needed.
intPersonID AS Int	The id of person that want to do the search.
intCountryID AS Int	The id of country leading to the language that the data and descriptions are returned in.
bitFirstSearch AS Bit	Indicates if it is the first page of the search.
intSessionID AS Int	The ASP session id used to distinguish between different searches.

intProfileTypeID AS Int	The id of the profile types to search through.
strSearch AS nVarChar(2000)	The string to search.
intOrderByFieldID AS Int = NULL	The field to sort by when returning the result.
bitOrderAsc AS bit = 1	Bit to indicate ascending or descending order of the result.

Returns zero or more rows with the following fields

fldFieldID	The id of the field.
fldString	The common name of the field.
fldPersonID	The person that owns the profile.
fldLastLogin	The date of the last login of the person that owns the profile.
fldProfileID	The id of the resulting profile.
fldData	The data for the resulting field.
fldFieldName	The name representing the field.
fldFieldListStrID	The id of the list item if present.
fldListString	The common name of the list item if present.

spGetLogin

This Stored Procedure is used to validate a user when logging in.

Input variables:

nvchEncPass AS NVarChar(2000)	The password for decrypting the password if needed.
txtLogin AS nVarChar(2000)	The login.
txtPassword AS nVarChar(2000)	The password.

Returns:

fldPersonID	The id of the person logged in. The value is NULL if login failed
fldCountryID	The id of the default country set by the person logged in.
fldAdminStatus	The AdminStatus of the person logged in.

spSetLogout

This Stored Procedure is used when a user logs out. It clears up the temporary search tblSearchResults table and sets fldOnline in tblPerson to 0.

Input variables:

intSessionID AS Int	The ASP session of the user.
intPersonID As Int = NULL	The id of person that wants to log out.

Returns nothing.

spCheckAccess

This Stored Procedure is used to check the access to a profile. Both the ACL's and the AdminStatus are used to check the access rights.

Input variables:

intPersonID As Int	The id of person to check the access for.
vchAccessType As VarChar(255)	The type of access to check.
bitRaiseError As Bit = 1	Indicate if an error needs to be raised if access denied.
intProfileID As Int = 0	The id of the profile that is being accessed.
intProfileTypeID As Int = 0	The id of the profile type that is being accessed.

Returns:

fldAccess	Value is 0 if the access is granted.

Raises the following errors:

'Can not assign access to this resource'

'The user does not have access to this resource'

spGetCountries

This Stored Procedure is used to return the available countries and languages available in the RDK.

Input variables: none.

Returns zero or more rows with the following fields:

fldCountryID	The id indicating the specific country.
fldCountrySuffix	A descriptor for the specific country.

fldLanguageName The name of the related language.

spGetLanguageStrings

This Stored Procedure is used to retrieve strings from the database in a specific language.

CountryID As Int	The id of country leading to the language that the strings are returned in.
strStringTableID1 As Int	The id of the string to return.
****** The line above repeated 24 times, with increasing number *****	
strStringTableID25 As Int = NULL	The maximum number of strings are 25.

Returns zero or more rows with the following fields:

fldString	The string in the given language.

spGetProfileFromLogin

This Stored Procedure is used to get the data of a profile from a login. The typical use of this Stored Procedure is for emailing the users password.

Input variables:

nvchEncPass nVarChar(2000)	The password for decrypting the data if needed.
txtData nVarChar(2000)	The login to retrieve the profile from.

Returns zero or more rows with the following fields:

fldFieldID The id of the field.

fldData The data for the field.

spGetPersonID

This Stored Procedure is used to get the person id of all persons having the data in their profile

Input variables:

nvchFieldData As nVarChar(4000)	The data to search for.
intFieldID As Int	The id of the field to search in.

Returns zero or more rows with the following fields:

fldPersonID	The id of the persons matching the data.

8.2.4.2 Stored Procedures used for the configuration

spAdmAddField

This Stored Procedure is used to add a field to a profile in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is adding the field.
intFieldID AS Int	Not used in this context.
intProfileTypeID AS Int	The profileTypeID of the field.
vchFieldNameStrID AS VarChar(255)	Not used in this context.
nvchFieldName AS NVarChar(1000)	The name of the field.
intFieldTypeID AS Int	The Type of the field.
intOrder AS Int	The sort order of the field.

intGroup AS Int	The associated group.
intCountryDependent AS Int	Defines if the field is country dependent.
bitPublic AS Bit,	Defines if the field is public.
bitSearchable AS Bit	Defines if the field should be in the search form.
bitSearchResult AS Bit	Defines if the field should be returned in the search results.
intMin AS Float	The minimum value of the field.
intMax AS Float	The maximum value of the field.
bitRepeatable AS Bit	Defines if the field is repeatable.
bitRequiered AS Bit	Defines if the field is required.
vchSpecial AS VarChar(50)	Defines if the field is a special field.
vchErrorStrID AS VarChar(255)	The id to the error text.
intFieldID2 AS Int	Not used in this context.

Returns nothing.

spAdmGetFields

This Stored Procedure is used to retrieve the fields for editing in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that retrieving the fields.
intCountryID As Int =1	The id of the country leading to the language the fields are retrieved in.
intProfileType As Int=9	The id of the profile to retrieve the fields for.

Returns zero or more rows with the following fields:

fldFieldID	The id of the field.
fldProfileTypeID	The id of the related profile type.
fldFieldTypeID	The type of the field.
fldFieldOrder	The order of the field in comparison to the other fields.
fldGroup	The group related to the field.
fldFieldNameStrID	The id of the name representing the field.
fldFieldName	The name representing the field.
fldCountryDependent	Defines if the field is country dependent.
fldPublic	Defines if the field is public.
fldSearchable	Defines if the field should be in the search.
fldSearchResult	Defines if the field should be returned in the search results.
fldMin	The minimum value of the field.
fldMax	The maximum value of the field.
fldRepeatable	Defines if the field is repeatable.

fldSpecial	Defines if the field is a special
fldRequired	Indicating if the field is required to be filled out.
fldErrorStrID	The id of the string containing a general error message for the field.
fldError	The string containing a general error message for the field.
fldFieldID	The id of the field repeated.

spAdmSetField

This Stored Procedure is used to edit a field to a profile in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is editing the field.
intFieldID AS Int	The id of the field to edit.
intProfileTypeID AS Int	The profileTypeID of the field.
vchFieldNameStrID AS VarChar(255)	The id of the string representing the field name.
nvchFieldName AS NVarChar(1000)	The name of the field.
intFieldTypeID AS Int	The Type of the field.
intOrder AS Int	The sort order of the field.
intGroup AS Int	The associated group.

intCountryDependent AS Int	Defines if the field is country dependent.
bitPublic AS Bit	Defines if the field is public.
bitSearchable AS Bit	Defines if the field should be in the search form.
bitSearchResult AS Bit	Defines if the field should be returned in the search results.
intMin AS Float	The minimum value of the field.
intMax AS Float	The maximum value of the field.
bitRepeatable AS Bit	Defines if the field is repeatable.
bitRequiered AS Bit	Defines if the field is required.
vchSpecial AS VarChar(50)	Defines if the field is a special field.
vchErrorStrID AS VarChar(255)	The id to the error text.
intFieldID2 AS Int	Not used in this context.

Returns nothing.

spAdmDeleteField

This Stored Procedure is used to delete a field from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person deleting the field.
intFieldID AS Int	The id of the field to delete.

Returns nothing.

spAdmAddFieldList

This Stored Procedure is used to add list items to a field from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is adding the list item.
intProfileTypeID AS Int	The profile type id of the field which related to the list item.
intFieldListID AS Int	The id of the list item.
intFieldID AS Int	The id of the field which related to the list item.
vchFieldListStrID AS VarChar(255)	The id of the name of the list item.
nvchFieldList AS NVarChar(1000)	The name of the list item.
intOrder AS Int	The order list item in respect to the other list items.
intFieldListID2 AS Int	Not used in this context.

Returns nothing.

spAdmDeleteFieldList

This Stored Procedure is used to delete a list item from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is deleting the list item.
intFieldListID AS Int	The id of the list item.

Returns nothing.

spAdmSetFieldList

This Stored Procedure is used to edit a list item related to a field from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is editing the list item.
intProfileTypeID AS Int	The profile type id of the field which related to the list item.
intFieldListID AS Int	The id of the list item.
intFieldID AS Int	The id of the field which related to the list item.
vchFieldListStrID AS VarChar(255)	The id of the name of the list item.
nvchFieldList AS NVarChar(1000)	The name of the list item.

intOrder AS Int	The order list item in respect to the other list items.
intFieldListID2 AS Int	Not used in this context.

Returns nothing.

spAdmGetFieldListItems

This Stored Procedure is used to retrieve the list items for editing in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that retrieving the list items.
intCountryID As Int =1	The id of the country leading to the language the list items are retrieved in.
intProfileTypeID As Int	The id of the profile type to retrieve the list items for.

Returns zero or more rows with the following fields:

fldFieldID	The id of the field which related to the list item.
fldFieldListID	The id of the list item.
fldFieldTypeID	The type of the field related to the list item.
fldFieldListStrID	The id of the name of the list item.
fldString	The name of the list item.
fldFieldListOrder	The order list item in respect to the other list items.

spAdmAddProfileType

This Stored Procedure is used to add a profile type from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is adding the profile type.
intProfileTypeID AS Int	Not used in this context.
vchProfileTypeNameStrID AS VarChar(255)	Not used in this context.
nvchProfileTypeName AS NVarChar(1000)	The name of the profile type.
intOrder AS Int	The sort order of the profile type.
intReadAccess AS Int	The AdminStatus for read access.
bitReadOwner AS Bit	Defines if the owner has read access.
intWriteAccess AS Int	The AdminStatus for write access.
bitWriteOwner AS Bit	Defines if the owner has write access.
intDeleteAccess AS Int	The AdminStatus for delete access.
bitDeleteOwner AS Bit	Defines if the owner has delete access.
intCreateAccess AS Int	The AdminStatus for create access.
intFieldID2 AS Int	Not used in this context.

Returns nothing.

spAdmGetProfileTypes

This Stored Procedure is used to retrieve the profile types for editing in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that retrieving the profile types.
intCountryID As Int =1	The id of the country leading to the language the profile types are retrieved in.

Returns zero or more rows with the following fields:

fldProfileTypeID	The id of the profile type.
fldString	The name of the profile type.
fldProfileTypeNameStrID	The id of the name of the profile type.
fldProfileOrder	The sort order of the profile type.
fldReadAccess	The AdminStatus for read access.
fldReadOwner	Defines if the owner has read access.
fldWriteAccess	The AdminStatus for write access.
fldWriteOwner	Defines if the owner has write access.
fldDeleteAccess	The AdminStatus for delete access.
fldDeleteOwner	Defines if the owner has delete access.
fldCreateAccess	The AdminStatus for create access.

spAdmSetProfileType

This Stored Procedure is used to edit a profile type from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is editing the profile type.
intProfileTypeID AS Int	The id of the profile type to be edited.
vchProfileTypeNameStrID AS VarChar(255)	The id of the name of the profile type.
nvchProfileTypeName AS NVarChar(1000)	The name of the profile type.
intOrder AS Int	The sort order of the profile type.
intReadAccess AS Int	The AdminStatus for read access.
bitReadOwner AS Bit	Defines if the owner has read access.
intWriteAccess AS Int	The AdminStatus for write access.
bitWriteOwner AS Bit	Defines if the owner has write access.
intDeleteAccess AS Int	The AdminStatus for delete access.
bitDeleteOwner AS Bit	Defines if the owner has delete access.
intCreateAccess AS Int	The AdminStatus for create access.
intFieldID2 AS Int	Not used in this context.

Returns nothing.

spAdmDeleteProfileType

This Stored Procedure is used to delete a profile type from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person deleting the profile type.
intProfileTypeID AS Int	The id of the profile type to delete.

Returns nothing.

spAdmAddTranslation

This Stored Procedure is used to add a translated text from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is adding the translated text.
intLanguage1 AS Int	The id of the first language.
intLanguage2 AS Int	The id of the first second language.
intStringID AS Int	Not used in this context.
intNewStringID AS Int	The id of the text string.
vchPage AS VarChar(255)	Text describing the context of the text usually the name of an ASP page.
nvchLanguage1 AS NVarChar(1000)	The text in the first language.

nvchLanguage2 AS NVarChar(1000)	The text in the second language.
intStringID2 AS Int	Not used in this context.

Returns nothing.

spAdmGetTranslation

This Stored Procedure is used to retrieve the text for translation in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is retrieving the translated text.
intLanguageID1 As Int = 2	The id of the first language.
intLanguageID2 As Int = 2	The id of the second language.

Returns zero or more rows with the following fields:

fldStringID	The id of the strings.
fldPage	Text describing the context of the text usually the name of an ASP page.
fldLanguage1	The text in the first language.
fldLanguage2	The text in the second language.

spAdmSetTranslation

This Stored Procedure is used to edit a text for translation from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is adding the translated text.
intLanguage1 AS Int	The id of the first language.
intLanguage2 AS Int	The id of the second language.
intStringID AS Int	The id of the texts.
intNewStringID AS Int	The id of the text string.
vchPage AS VarChar(255)	Text describing the context of the text usually the name of an ASP page.
nvchLanguage1 AS NVarChar(1000)	The text in the first language.
nvchLanguage2 AS NVarChar(1000)	The text in the second language.
intStringID2 AS Int	Used to validate the correct text.

Returns nothing.

spAdmDeleteTranslation

This Stored Procedure is used to delete a text from the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person that is deleting the translated text.
intStringID AS Int	The id of the texts to be deleted.

Returns nothing.

spAdmGetFieldTypes

This Stored Procedure is used to retrieve the available field types in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person requesting the field types.

Returns zero or more rows with the following fields:

fldFieldTypeID	The id of the field type
fldFieldName	The common name of the field type

spAdmGetErrors

This Stored Procedure is used to retrieve the available errors in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int	The id of person requesting the field types.
intLanguageID As Int =1	The id of the language the errors should be displayed in.

Returns zero or more rows with the following fields:

fldStringID	The id of the error text.
fldString	The error text.

spAdmGetLanguages

This Stored Procedure is used to retrieve the available languages in the administrator module. The rights for having administrator access are checked with the Stored Procedure spCheckAccess

Input variables:

intPersonID Int The id of person requesting the languages.

Returns zero or more rows with the following fields:

fldLanguageID The id of the language.

fldLanguageName The name of the language.

spAdmDeleteDeletedItems

This Stored Procedure is used to delete profiles, persons and data from tblRead, when they have 1 in fldDeleted. The rights for having administrator access are checked with the Stored Procedure spCheckAccess.

Input variables:

intPersonID Int The id of person that deleting the items.

Returns nothing.

8.2.5 User defined functions

fnCheckAccess

This function is similar to the stored procedure spCheckAccess and is used to check the access to a profile. Both the ACL's and the AdminStatus is used to check the access rights. At the moment of writing the function is only used by spGetSearch.

Input variables:

@intPersonID As Int	The id of person to check the access for.
vchAccessType As VarChar(255)	The type of access to check.
intProfileID As Int = 0	The id of the profile that is being accessed.
intProfileTypeID As Int = 0	The id of the profile type that is being accessed.

Returns:

intAccess	Value is 0 if the access is granted.

fnFieldData

This function is used to determine if a field is encrypted. If the field is encrypted it will return NULL else it will return the field data.

Input variables:

intFieldID Int	The id of the field
NVarCharFieldData NVarChar(2000)	The data of the field

Returns:

NVarCharFieldData	The field data if not encrypted.

fnRC4

This function is used to de/encrypt Unicode strings using reversible RC4 encryption. The function is taken from [21] and adapted to the RDK.

long_string	nvarchar(2000)	String to be en/decrypted
short_string	nvarchar(500)	En/decryption password

Returns:

out_string	The de/encrypted string.

8.3 Performance

The described data model yields some obvious areas where bad performance could be experienced. Since all data inserted into the RDK are converted to the MSSQL data type nVarChar, the benefits of searching through and sorting numbers and dates are left out. In more performance critical applications, a specific search table could be added to the database, containing the profiles that are used in the search functions, with columns containing the proper data types. This additional search table could either be updated in real time if needed or within a given time interval.

In most websites the searches and display of data are done much more often than insertions maybe on a factor 100:1 or 1000:1 so in general if optimisations are needed, the best place to do this work is on the search and display functions.

There might also be a performance gain in loading all the data from tblStringTable into a dictionary object[45] at application start up, instead of querying the database each time a text is needed by the RDK.

8.4 Future improvements

I have plans to implement the system in either VB.Net or C#.Net. This will make the system more attractive to a commercial market. But there are also some smaller improvements which are mentioned below.

[45] A dictionary object allows you to create word arrays to use for spell checkers, dictionaries, etc.

In the data model there are several areas that I need to improve. There exists some doublets of fields in tblCountry and tblLanguage, and some of them are not used at all. I need to have a closer look at this and remove some columns in these two tables. tblCountry also need to have a time zone and daylight saving info added. The support of time zones in ASP is very limited.

tblRead and tblType needs to be generalized a bit. I could imagine renaming them to tblAccess and tblAccessType. Then every time a profile is accessed, edited or deleted a row is added to tblAccess. This could be a part of an audit function that also logs the user's ip-address and maybe some info about screen resolution, operating system and web browser.

The administrator interface needs some more error handling, and the number of Stored Procedures prefixed spAdm could be reduced by merging the adding and editing Stored Procedures into one that does both.

To add more security, hashed[46] passwords could be used in some applications. Since it is a one way only encryption, you lose the possibility to send their password by email or SMS. Instead the system can assign a new password for the user and send that instead.

The definition of minimum and maximum value in tblField, could be changed to a regular expression[47] instead.

The RDK could also use some kind of predefined editable profiles such as a voting system, the general user, an image library, a shopping cart and many more.

---

[46] Hashing is the transformation of a string of characters into a fixed-length value or key that represents the original string.

[47] A regular expression is a way to express how a program should look for a specified pattern in a text.

There is no error handling for the RDK's client side JavaScript. This should be added as soon as possible.

Sorting for multiple simultaneously languages should be added. The same strategy as described in [appendix: Kosovo report] is preferred.

To enhance the usability for the developer, friendly names for each of the fields in a profile could be added, in stead of using field id's in the ASP code.

The maximum size of 4000 characters in a nVarChar field in MSSQL could be a problem in some applications. So the performance of the nText[48] data type should be examined and maybe used instead or combined with nVarChar.

[48] Variable-length Unicode data with a maximum length of 2^{30} - 1,073,741,824 characters.

9 Examples of use

The RDK has already been found very useful during production of several websites. Below I will make a short description of the websites and their project statuses.

9.1 Synergy Estates

Synergy Estates was a prototype for a website trading primarily Spanish holyday apartments.

Unfortunately the project was closed due to lack of my partner's ability to promote the website.

9.2 MobilCash

http://MobilCash.dk is a mobile payment project allowing private users, using their mobile phone, to transfer money. The project is a partnership of three people in the areas of marketing, management and software development. The prototype has been tested by a user group of 12 people in a period of 3 weeks.

The status on MobilCash is that we are awaiting a legal assessment of the project. We plan to go into production late this year.

9.3 Made in Kailua

Made in Kailua is a website for a trade organization on the island of O'ahu Hawaii.

The Kailua website is still being developed by a partner of mine at http://kailua.thorstein.nu.

9.4 Musikinstrumenter.net

http://Musikinstrumenter.net is a website selling musical instruments. The plan is to launch more similar e-shops, when agreements with the wholesalers have been obtained.

Musikinstrumenter.net is already in production.

9.5 E-invoice

http://E-Invoice.dk is the invoicing system that handles the credit card payments for Musikinstrumenter.net. Future plans for E-Invoice is to enable small and medium sized businesses to do much of their invoicing over the Internet.

You can only login to E-Invoice if you have ordered something on Musikinstrumenter.net.

E-Invoice.dk is already in production.

9.6 UllasOpskrifter.dk

http://UllasOpskrifter.dk is a website used by the teacher at Lyngby Private Skole teaching food preparation. The website is presenting recipes from the teaching, and is used as a basis for the Tutorial in this master thesis.

UllasOpskrifter.dk is already in production.

10 Conclusion

The RDK has proven its strengths based on all the implementations made. The database model has been extremely flexible. In the latest implementations with the RDK, there was no need to think about database development, which one of the main objective with the RDK.

The increase in development speed using the RDK is obvious. The website used as basis for the tutorial in chapter 7 was developed in half a day. It could have taken weeks to develop without the RDK, and still the robustness would not have been the same.

There are still improvements that can be made to the RDK, as mentioned chapter 8.4, Future improvements. But the general design of the RDK has reached a stage where most of the features have been tested in several websites and proven robust.

The ability to make a working prototype instead of just a mock-up has been proven useful in order to get intelligent venture capital for the mobile payment system MobilCash, which is planned to go into production at the end of this year.

A future business strategy for using the RDK is to launch prototypes and production websites rapidly, as an investment in innovative projects. The projects which prove to have a certain rate of success can be the focus of further development.

References

[1] Michael Corning, Steve Elfanbaum, David Melnick.1997. Working With Active Server Pages. Que

[2] Presented by developerWorks. Create dynamic sites with PHP & MySQL. http://ibm.com/developerWorks

[3] Various articles from http://www.experts-exchange.com

[4] http://www.mysql.com/products/licensing.html

[5] http://www.microsoft.com/sql/msde/howtobuy/default.asp

[6] http://www.microsoft.com/sql/howtobuy/default.asp

[7] http://history.perl.org/PerlTimeline.html

[8] http://wwws.sun.com/software/chilisoft/index.html

[9] http://marty.anstey.ca/programming/php_asp.html

[10] Various articles from http://php.weblogs.com/

[11] http://training.gbdirect.co.uk/courses/perl/comparison_php_versus_perl_vs_asp_jsp_vs_vbscript_web_scripting.html

[12] http://www.sitepoint.com/article/546/

[13] http://websphereadvisor.com/doc/11571

[14] http://www2.umassd.edu/SWPI/costmodeling/papers/03680322.pdf

[15] http://www.embeddedforecast.com/EMFTCD2003v3.pdf

[16] http://www.eds.com/thought/thought_web_framework.pdf

[17] Shan and Earle. 1998. Enterprise Computing With Objects. Addison Wesley

[18] http://www.onestat.com/html/aboutus_pressbox23.html

[19] http://www.upsdell.com/BrowserNews/stat_trends.htm

[20] http://www.serverobjects.com/products.htm#aspimage

[21] http://www.databasejournal.com/scripts/article.php/1496441

[22] http://www.dimac.net

[23] http://www.pstruh.cz/help/scptutl/upload.asp

[24] http://www.oracle.com/corporate/pricing/ePLext.pdf

[25] http://www.microsoft.com/sql/evaluation/compare/pricecomparison.asp

[26] http://www.theserverside.com/articles/ article.tss?l=J2EE-vs-DOTNET

[27] http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/Default.asp?url=/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/cerr_enablingcustom.asp

[28] http://www.searchengineworld.com/robots/robots_tutorial.htm

[29] Various articles from http://whatis.techtarget.com

Abbreviations

XML (Extensible Markup Language)

SQL (Structured Query Language)

HTML (HyperText Markup Language)

SSI (Server Side Includes)

CGI (Common Gateway Interface)

ASP (Active Server Pages)

PHP (Hypertext Preprocessor)

XSL (Extensible Stylesheet Language)

CSS (Cascading Style Sheets)

DBMS (Database Management System)

SQL (Structured Query Language)

IIS (Internet Information Server)

PDF (Portable Document Format)

JSP (Java Server Pages)

FTP (File Transfer Protocol)

DLL (Dynamic Link Libraries)

Appendix A Sorting of Multiple Character Sets

In this appendix you will find a description of how sorting of multiple character sets can be done.

The chapter is taken from a report written by me after my stay in Kosovo as Chief of Database Development for the general assembly election in 2001.

Due to the many minorities and the political necessity to treat all citizens equally, all material used and shown to the public, such as programs for the Counting and Results centre (C&RC)[49] and several reports such as Voter Lists, had to support the three following languages:

- Albanian
- Serbian (Latin)
- Turkish

The handling of the three languages separately does not cause any severe problems, but when mixed in reports and databases there is a lot of problematic issues.

Fortunately MS-SQL Server supports Unicode so we were able to store all data in Unicode, but a lot of our data sources e.g. the UN civil registry, did not store the data in Unicode, so we had to write character set conversion utilities, which in some cases were a big challenge, due to the mock up solutions used in the existing datasets.

Another issue related to the fonts were how to sort them. There are no predefined rules on how to sort combined character sets, so we first had to come up with a sort order for the combined character set, and then write sorting routines for the actual sorting.

In Figure 1 is a scheme of the used characters and the short-order we defined.

[49] Observers and users of the programs in C&RC should be able to use the programs for the counting and results processing in their own language.

A	1	H	17	RR	33
B	2	I	18	S	34
C	3	İ	19	SH	35
Ç	4	J	20	Š	36
Č	5	K	21	Ş	37
Ć	6	L	22	T	38
D	7	LL	23	TH	39
DŽ	8	LJ	24	U	40
DH	9	M	25	Ü	41
Đ	10	N	26	V	42
E	11	NJ	27	X	43
Ë	12	O	28	XH	44
F	13	Ö	29	Y	45
G	14	P	30	Z	46
GJ	15	Q	31	ZH	47
Ğ	16	R	32	Ž	48

Figure 1 the combined character set and the sort codes

As seen in Figure 1 some of the characters are combined of two characters from the Latin character set. This is because this combination of two letters is seen as one, just like the Danish AA was once used instead of Å.

Appendix B IIS Error Messages

In this appendix you will find a description error messages returned by IIS.

The list is taken from:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/iissdk/iis/customerrormessagesreference.asp

Error code	Error message
400	Cannot resolve the request.
401.1	Unauthorized: Access is denied due to invalid credentials.
401.2	Unauthorized: Access is denied due to server configuration favouring an alternate authentication method.
401.3	Unauthorized: Access is denied due to an ACL set on the requested resource.
401.4	Unauthorized: Authorization failed by a filter installed on the Web server.
401.5	Unauthorized: Authorization failed by an ISAPI/CGI application.
401.7	Unauthorized: Access denied by URL authorization policy on the Web server.
403	Forbidden: Access is denied.
403.1	Forbidden: Execute access is denied.
403.2	Forbidden: Read access is denied.
403.3	Forbidden: Write access is denied.
403.4	Forbidden: SSL is required to view this resource.

403.5	Forbidden: SSL 128 is required to view this resource.
403.6	Forbidden: IP address of the client has been rejected.
403.7	Forbidden: SSL client certificate is required.
403.8	Forbidden: DNS name of the client is rejected.
403.9	Forbidden: Too many clients are trying to connect to the Web server.
403.10	Forbidden: Web server is configured to deny Execute access.
403.11	Forbidden: Password has been changed.
403.12	Forbidden: Client certificate is denied access by the server certificate mapper.
403.13	Forbidden: Client certificate has been revoked on the Web server.
403.14	Forbidden: Directory listing is denied on the Web server.
403.15	Forbidden: Client access licenses have exceeded limits on the Web server.
403.16	Forbidden: Client certificate is ill-formed or is not trusted by the Web server.
403.17	Forbidden: Client certificate has expired or is not yet valid.

403.18	Forbidden: Cannot execute requested URL in the current application pool.
403.19	Forbidden: Cannot execute CGIs for the client in this application pool.
403.20	Forbidden: Passport logon failed.
404	File or directory not found. Note Tools like URLScan can be configured to block processing of certain file name extensions. Check your URLScan settings.
404.1	File or directory not found: Website not accessible on the requested port. Note The 404.1 error can occur only on computers with multiple IP addresses. If a specific IP address/port combination receives a client request, and the IP address is not configured to listen on that particular port, IIS returns a 404.1 HTTP error. For example, if a computer has two IP addresses and only one of those IP addresses is configured to listen on port 80, any requests received on the other IP address with port 80 result in IIS returning a 404.1 error. This error should be set only at the service level because it is returned to clients only when multiple IP addresses are used on the server.
404.2	File or directory not found: Lockdown policy prevents this request.
404.3	File or directory not found: MIME map policy prevents this

	request.
405	HTTP verb used to access this page is not allowed.
406	Client browser does not accept the MIME type of the requested page.
407	Initial proxy authentication required by the Web server.
410	File has been removed.
412	Precondition set by the client failed when evaluated on the Web server.
414	Request URL is too large and therefore unacceptable on the Web server.
500	Internal server error.
500.11	Server error: Application is shutting down on the Web server.
500.12	Server error: Application is busy restarting on the Web server.
500.13	Server error: Web server is too busy.
500.14	Server error: Invalid application configuration on the server.
500.15	Server error: Direct requests for GLOBAL.ASA are not allowed.

500.16	Server error: UNC authorization credentials incorrect.
500.17	Server error: URL authorization store cannot be found.
500.18	Server error: URL authorization store cannot be opened.
500.19	Server error: Data for this file is configured improperly in the metabase.
500.20	Server error: URL authorization scope cannot be found.
500 100	Internal server error: ASP error.
501	Header values specify a configuration that is not implemented.
502	Web server received an invalid response while acting as a gateway or proxy server.

Appendix C Robots.txt Tutorial

This appendix is a tutorial on the use of Robots.txt.

The tutorial is taken from [28]

Robots.txt Tutorial

Search engines will look in your root domain for a special file named "robots.txt" (http://www.mydomain.com/robots.txt). The file tells the robot (spider) which files it may spider (download). This system is called, *The Robots Exclusion Standard*.

The format for the **robots.txt** file is special. It consists of records. Each record consists of two fields : a User-agent line and one or more **Disallow:** lines. The format is:

```
<Field> ":" <value>
```

The robots.txt file should be created in Unix line ender mode! Most good text editors will have a Unix mode or your FTP client *should* do the conversion for you. Do not attempt to use an HTML editor that does not specifically have a text mode to create a robots.txt file.

User-agent

The **User-agent** line specifies the robot. For example:

```
User-agent: googlebot
```

You may also use the wildcard character "*****" to specify all robots:

```
User-agent: *
```

You can find user agent names in your own logs by checking for requests to robots.txt. Most major search engines have short names for their spiders.

Disallow:

The second part of a *record* consists of **Disallow:** directive lines. These lines specify files and/or directories. For example, the following line instructs spiders that it can not download email.htm:

```
Disallow: email.htm
```

You may also specify directories:

```
Disallow: /cgi-bin/
```

Which would block spiders from your cgi-bin directory.

There is a wildcard nature to the Disallow directive. The standard dictates that /bob would disallow **/bob.html** and **/bob/indes.html** (both the file bob and files in the bob directory will not be indexed).

If you leave the Disallow line blank, it indicates that ALL files may be retrieved. At least one disallow line must be present for each User-agent directive to be correct. A completely empty Robots.txt file is the same as if it were not present.

White Space & Comments

Any line in the robots.txt that begins with **#** is considered to be a comment only. The standard allows for comments at the end of directive lines, but this is really bad style:

```
Disallow: bob #comment
```

Some spider will not interpret the above line correctly and instead will attempt to disallow "bob#comment". The moral is to place comments on lines by themselves.

White space at the beginning of a line is allowed, but not recommended.

```
    Disallow: bob #comment
```

Examples

The following allows all robots to visit all files because the wildcard "*" specifies all robots.

```
User-agent: *
Disallow:
```

This one keeps all robots out.

```
User-agent: *
Disallow: /
```

The next one bars all robots from the cgi-bin and images directories:

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /images/
```

This one bans Roverdog from all files on the server:

```
User-agent: Roverdog
Disallow: /
```

This one bans keeps googlebot from getting at the cheese.htm file:

```
User-agent: googlebot
Disallow: cheese.htm
```

For more complex examples, try retrieving some of the robots.txt files from the big sites like Cnn, or Looksmart.

Extensions to the Standard

Although there have been proposed standards extensions such as an **Allow** line or robot version control, there has been no formal endorsement by the Robots exclusion standard working group.

Appendix D Sourcecode

An installation CD is enclosed with this master thesis. All contents of the CD are confidential.

The sourcecode for the RDK is placed on the installation CD in the following directory:

/Sourcecode

The contents of the subdirectories are the following:

/Sourcecode/ASP
This directory contains all the ASP files needed for the installation of the RDK it is an exact copy of the /ASP directory.

/Sourcecode/Install Program
This is the sourcecode used for the installation program in section 6.1.3.

/Sourcecode/Database Tables
The SQL code for creating the MSSQL database tables.

/Sourcecode/Stored procedures

The SQL code for creating the MSSQL Stored Procedures.

/Sourcecode/User Defined Functions

The SQL code for creating the MSSQL user defined functions.