CHALMERS

.NET Development for the web

Using Microsoft Office SharePoint Server 2007 and ASP.NET

Master of Science Thesis in Software Engineering and Technology

OSKAR JACOBSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, December 2009

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.
The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

.NET Development for the web
Using Microsoft Office SharePoint Server 2007 and ASP.NET.

OSKAR JACOBSSON

© OSKAR JACOBSSON, December 2009.

Examiner: SVEN-ARNE ANDREASSON

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden, December 2009

Abstract

For a company or organisation, it has become very important to 'exist on the web', that is to provide a webpage where information, goods available for purchase and services etc. are available for the customers. As a natural consequence of an increased interest for the Internet and its fast growth, software developing companies sometimes needs to adapt their solutions for the web. In this thesis, several methods and solutions for web development within .NET are proposed.

The solutions are in most cases related to *Microsoft Office Excel* where the possibilities and limitations for web publishing of workbooks are investigated. The main focus is on *Microsoft Office SharePoint Server 2007* and *ASP.NET* solutions. In SharePoint, several solutions are proposed for dealing with limitations with Excel Services and Excel Web Access when publishing workbooks. SharePoint is also examined further for solutions for database handling and COM automation. An application that publish Excel workbooks on the web is developed with ASP.NET, and presenting ways of displaying and interacting with a workbook published to a server. We also look at a way of presenting data visually on the web, using *Microsoft Chart Controls for .NET Framework 3.5* which can present data in chart format. *Office Open XML* are proposed as an alternative for COM automation when publishing Excel workbooks on the web.

Preface

This report concludes and describes the work of my master's thesis ".NET Development for the Web" which is a thesis written at the department of Computer Science and Engineering, Chalmers University of Technology. Examiner at Chalmers is Sven-Arne Andreasson. Supervisors at the company Excelspecialisten AB where the thesis was written, are Jesper Jonsteg and Niklas Jansson. I wish to thank them for their advice and feedback given during the work.

Contents

1 Introduction

1.1 Background

Internet has made an enormous impact on the society. Internet has changed the way we communicate, how we work, how we shop, how companies advertise, just to name a few areas of impact. For a company or organisation, it has become very important to 'exist on the web', i.e. provide a webpage where information, goods available for purchase, services etc. are available for the customers. As a natural consequence of an increased interest for the internet and its fast growth, software developing companies sometimes needs to adapt their solutions for the web. In this thesis, several methods and solutions for web development within .NET are proposed. This thesis main focus is on Microsoft Office SharePoint Server 2007 and ASP.NET.

1.1.1 ExcelSpecialisten XLS AB

Excelspecialisten is a software company located in Partille, a few kilometers east of Gothenburg. The company has more than 15 years of experience in software development and has developed more than 500 applications. Currently there are about 30 employees at the company. They specialize in development of Excel and other Office-based solutions. ExcelSpecialisten develops business related and customized solutions: report tools, analysis tools and business reports for companies all over Sweden. Currently VBA (Visual Basic For Applications) and Visual Basic is used as programming languages. For more information about the company see [19].

1.2 Motivation

Excelspecialisten wants to investigate the possibilities for integration of the company's current solutions to a more webbased environment. The company develops applications in .NET and Visual Basic for Applications (VBA) and wishes to investigate how to transfer as much of these applications functionality to the web as possible. This thesis should propose solutions for presenting this kind of data/applications on the web. This thesis consists of several implementations of various kinds in order to compare different methods for development. Internal Presentations of the work within the company is also included in the thesis.

1.3 Goals

Goals for the project, more specifically:

- **Example Project**

Example project that retrieves data from ExcelSpecialisten's database and presents it on the web or SharePoint.

- **Presentations within the company**

  Presentations where the methods and techniques are presented for the employees.

- **Publish data on the web**

  Analyze techniques for presenting different kinds of data on the web or SharePoint.

1.4 Limitations

In this thesis, development and research was limited to ASP.NET applications and SharePoint solutions.

2 Technology

In this section most of the technology related to this thesis is described. This section serves as a introduction to the concepts, technology and methods used during the work of the thesis.

2.1 Microsoft .NET

The .NET Framework is a system component and Microsoft's platform for building applications. The .NET Framework has two main components: the common language runtime (CLR) and the .NET Framework class library. The CLR is responsible for providing core services such as memory management, thread management and also handles security and robustness issues. The other core component of the .NET Framework is the class library. The class library is a collection of classes that the developer can use when developing applications.

2.1.1 Common Language Runtime (CLR)

The CLR Handles memory management, compilation, code execution, thread execution, code safety verification, and other system services. The CLR manages references to objects and releasing them when the object is no longer used.

The CLR makes it possible to design a component in one language and then make it interact with a component written in another language. It is for example possible to pass an instance of a class to a method of a class written in a different language. This cross-language integration is possible because language compilers and tools use a common type system (CTS) defined by the Common Language Runtime [3]. The CTS defines rules that languages must follow, which ensures that objects in different languages can interact.

2.1.2 .NET Framework Class Library

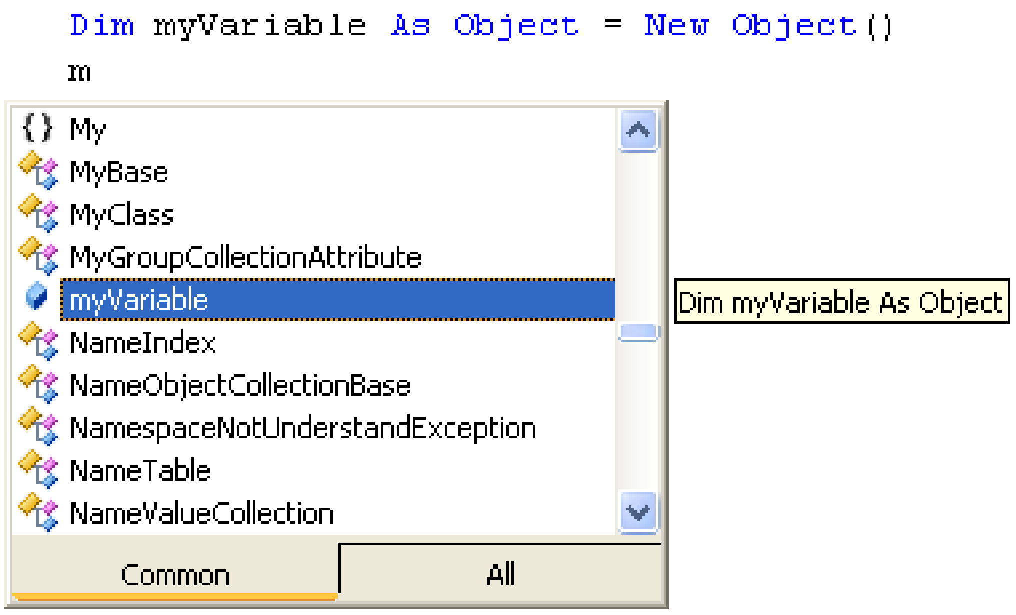
.NET Framework class library is a collection of classes that helps the developer with various programming tasks. String management, data collection, database connectivity and file access are examples of common tasks that the class library has support and classes for [4]. The .NET Framework class library also includes options that support several different development scenarios, for example: Console applications, Windows GUI applications (Windows Forms) and ASP.NET applications.

2.1.3 Visual Studio

Microsoft Visual Studio is an Integrated Development Environment (IDE) based development tool for building applications. There are support for several programming languages in Visual

Studio: Visual Basic, C# and Visual C++. Visual Studio can be installed on Windows Vista, Windows XP, Windows Server 2008, and Windows Server 2003 R2 [5]. Visual Studio is used as development tool when developing .NET applications and was used for all development within this thesis.

When coding in Visual Studio the developer don't need to spend so much time to lookup keywords or other language elements, as Visual Studio has built in support for this named IntelliSense. IntelliSense provides a set of options when the developer starts to type something and this way the developer can search directly for the keyword or class by just having to start typing the first letter(s) of the keyword/class [6]. IntelliSense comes up with suggestions for the typed combination of letters. IntelliSense can also complete the typing for the developer, speeding up development further.



```
Dim myVariable As Object = New Object()
m
```

Figure 1: Intellisense gives suggestions of available classes, methods, attributes and variables that can be used in the given context

2.1.4 ADO.NET

ADO.NET is a Microsoft technology built on the .NET Framework and provides consistent access to data sources such as Microsoft SQL Server, and other data sources accessed by OLEDB and XML. ADO.NET can be used in applications by the programmer to retrieve and manipulate data. There are two central components in ADO.NET: the DataSet and the .NET Framework Data Provider.

The DataSet is the core element of the ADO.NET architecture and is designed for data access from multiple and different data sources [7]. The structure of the DataSet is made up of a collection of one or several Datatable objects that are made up of rows and columns of data and primary/foreign key, constraint, and relation information about the data in the DataTable objects. [8]

2.1.5 ASP.NET

ASP.NET is a Microsoft technology part of the .NET Framework. ASP.NET is a successor to Microsoft's Active Server Pages (ASP) technology and is a web application framework used to build dynamic web sites, web applications and web services. ASP.NET is built on the Common Language Runtime (CLR), and thus allows the programmer to write ASP.NET code using any supported .NET language.

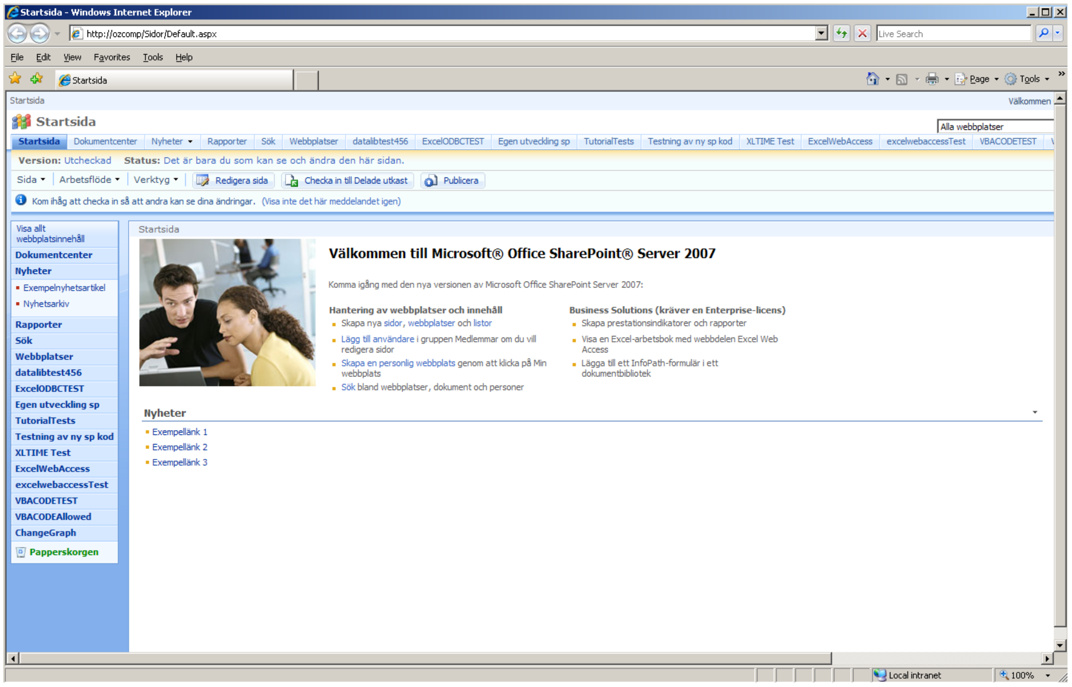
2.2 Microsoft Office Excel

Microsoft Office Excel is a widely used spreadsheet application. Microsoft Office Excel is used to create spreadsheets and has support for calculation, pivot tables, charts and a macro programming language VBA (Visual Basic for Applications) [9] that are covered in 2.2.1. The latest release of Excel for Microsoft Windows is Microsoft Office Excel 2007, even called version 12 of Excel, and is included in Microsoft Office 2007. A Microsoft Excel spreadsheet basically consists of numbered cells divided in rows and columns. Graphs can be created and is based on values in defined cells. Microsoft Excel contains built in functions that can be used for various task. There are functions for a variety of purposes, some examples of functions are math and trigonometry, statistical, financial, date and time functions. Users can also create their own custom functions and are called User Defined Functions. Microsoft Excel can be used for many purposes and situations like keeping track of finances, contact information and logs etc.

2.2.1 Visual Basic for Applications

Visual Basic for Applications (VBA) is an embedded programming language and is integrated in Microsoft Access, Microsoft Excel, Microsoft Outlook, Microsoft PowerPoint and Microsoft Word, which all are part of the Microsoft Office 2007 package [10]. VBA is used for creating custom solutions and extend the functionality for the application. VBA uses the full power of Microsoft Visual Basic programming language. Visual Basic for Applications 6.5 is available for the Microsoft Windows Server 2003, Windows XP, and Windows Vista operating systems on 32-bit Intel platforms.

From July 1, 2007 , Microsoft stop to offer VBA distribution licenses to new customers. Microsoft also announced that no significant improvements will be made to VBA in the future [11].



Version: Utcheckad **Status:** Det är bara du som kan se och ändra den här sidan.

Kom ihåg att checka in så att andra kan se dina ändringar. (Visa inte det här meddelandet igen)

Välkommen till Microsoft® Office SharePoint® Server 2007

Komma igång med den nya versionen av Microsoft Office SharePoint Server 2007:

Hantering av webbplatser och innehåll
- Skapa nya sidor, webbplatser och listor
- Lägg till användare i gruppen Medlemmar om du vill redigera sidor
- Skapa en personlig webbplats genom att klicka på Min webbplats
- Sök bland webbplatser, dokument och personer

Business Solutions (kräver en Enterprise-licens)
- Skapa prestationsindikatorer och rapporter
- Visa en Excel-arbetsbok med webbdelen Excel Web Access
- Lägga till ett InfoPath-formulär i ett dokumentbibliotek

Nyheter
- Exempellänk 1
- Exempellänk 2
- Exempellänk 3

Figure 2: SharePoint startpage

2.3 Microsoft Office SharePoint Server 2007

Microsoft Office SharePoint Server 2007 (MOSS 2007) is a server application that can help improve organizational effectiveness mostly by providing content management and information sharing. Office Sharepoint Server 2007 gathers organizations intranet, extranet and webapplications within one integrated platform, instead of having these as separate systems. The user interface of MOSS 2007 is a web interface accessed through a browser. Using MOSS 2007 helps to organize and store business documents in one centralized location. MOSS 2007 uses "Web Parts" to display content on the site. A Web Part is a "content container" used to display information on the site. It is possible to change the appearance of pages using Web Parts to add or delete content. The content that can be added to a webpart is listed when the user clicks "add a Web Part" somewhere on the page. Examples of Web Parts are lists, document libraries, key performance indicators, calendars etc. The program's developed for Sharepoint within this thesis have been using this webpart functionality. MOSS 2007 can be used to create web content using premade templates, and together with the web part structure, new webpages within the portal can be created easily. Registered Employees can create their own webpages within the portal and customize it as they like.

MOSS 2007 enables live, interactive business intelligence portals that can display dashboards, key performance indicators in order to track the status of different projects, products etc [12]. MOSS 2007 also offers a search center where users can search for people, content and business data.

2.3.1 Sharepoint and Excel

The most central functionality of Sharepoint according to this work is MOSS 2007s ability to display Microsoft Office Excel files within the portal. With Excel Services, described more in detail in section 2.5, it is possible to load and display Excel Workbooks within the portal. This functionality was examined and focused on in order to see Sharepoints possibilities and limitations when working with Excel documents.

2.3.2 Technical details

MOSS 2007 runs on Windows Server 2003 with Service Pack 1 or later, and Windows Server 2008 [13]. Internet Information Services (IIS) must be enabled on the host machine and Microsoft .NET Framework 3.0 installed with "ASP.NET 2.0 enabled" before installation.

Moss 2007 supports several common web browsers (Mozilla Firefox, Safari,Navigator) but is best used together with Microsoft's Internet Explorer 6.x and above for best performance [14].

2.4 COM Automation/COM Object

A COM (Component Object Model) object enables software components to communicate. In this thesis, an Excel COM object was used to gain access to Excels functionality. Lets pretend we created the COM object named "myExcelInstance".

Examples of usage of this "myExcelInstance" object is to use it to get/set a cell in a workbook, create new workbooks, save/open/close workbooks and other functionality that is possible with the "real" Excel application. This way we can modify Excel workbooks from .NET code.

2.5 Excel Services

Excel Services is a Microsoft Office Sharepoint technology that makes it possible to present, share, secure and use Microsoft Office Excel 2007 workbooks (in xlsx, xslb format) within a sharepoint site [15].

Excel services is composed of three components that interacts with each other to create the end result [16].

1. Excel Calculation Services (ECS) is the component where most of the work is done, ECS loads and calculates workbooks, refreshes external data and maintains sessions. ECS can also call custom code (user-defined functions).

2. Excel Web Access enables interaction with the excel workbook in a browser by using javascript and DHTML. With Excel Web Access, the workbook is displayed with a similar look and feel as an ordinary excel workbook in Excel. Excel Web Acces renders live Excel workbooks on a webpage and is the visible Excel Services component for the user. When using Excel Web Access it does not require anything to be installed on the user's client computer.

3. Excel Web services is a web service hosted in MOSS that provides methods for a developer to use as an API to build applications based on the workbook. With Excel Web Services developers can create applications that call methods from the API in order to set or extract values, refresh data connections, calculate values and more.

2.6 Excel Web Services API

The API offers developers more control when working with Excel Services. Excel Web Services API provides programmatic access to its web service. Developers can use the API to set, extract and calculate values from workbooks and write customized applications using the web service [16].

2.7 User Defined Function - UDF

User Defined Functions (UDF) are custom functions that extend the capabilities of Excel. With UDFs it is possible to create custom functions that are not built into excel. In the workbooks users can call UDFs from a cell through formulas just as if it were ordinary built-in functions in Excel. This way UDFs can overcome several limitations that Excel Services at the moment does not support [17].

3 Method

The main problem was analyzed and discussed together with my supervisor on how to divide the main problem into suitable sub problems to work with. Note that "Main problem" in this context can have different meanings. For example, one week the main problem could be to "Test database connectivity in SharePoint" and another week it could be referred to as "Convert a workbook to the web using ASP.NET". When a suitable sub problem was decided upon, literature studies were carried out for this particular topic and an implementation of this problem was made. When the development of the sub problem was done, the result was evaluated and discussed together with my supervisor. The evaluations purpose was to decide whether the sub problem was worth spending more time on, or select a new sub problem and focus on that. The discussion with the supervisor had the benefit that feedback could be given directly. The above method was iteratively applied during most of the work process.

3.1 Literature study

The source for the literature required for the work have been the Internet, mostly because of the technologies involved are rather new. The literature study have been carried out in paralell with the implementation and testing of the work. I did not find much information about "converting" excel workbooks to the web so the work started without influences from other sources. The literature studies was really important when I started to develop SharePoint solutions, as this was a new concept, there was few books about the subject.

3.2 Presentations

Several presentations was done within the company, where the current development and findings of the work was presented. These presentations gave some new ideas and was a good way to summarize topics that already had been investigated. The presentations was also a good way to check the current status of the work.

3.3 Work method

3.3.1 Excel

Part of the work is centered around Excel to see what features that can be supported in an Excel workbook in different environments. The most interesting and important features to investigate if they can be supported is listed here:

1. A workbook with diagrams and ordinary celldata (simply a very common standard Excel document) and named cells.

2. A workbook with links to another workbook
3. A workbook that have cells referencing an external data source (such as SQL Database).
4. A workbook containing VBA (Visual Basic for Applications) code.

The features in this list are to be investigated in several different environments to evaluate what features is supported in a specific environment. The different environments to test this functionality are Microsoft Office Sharepoint 2007 and ASP.NET. If a feature is not supported, workarounds for the problem will be searched for in order to make it work. The result from these evaluations will help to make a decision when a specific technique should be choosen. This thesis is not only focusing on Excel though, and there are other areas like database handling and graphs that also is covered.

4 Analysis

This chapter is divided into three major sections. The first section is all about SharePoint development. The second section covers ASP.NET and the third section covers development with Office Open XML.

4.1 Programming Language and development tools used

Visual Basic.NET (VB.NET) was used as programming language in Visual Studio 2008. C# and VB.NET is the most frequently used programming languages within .NET and the difference between the two languages is mostly on the syntax level. In Chalmers programming courses, Java is frequently used as programming language and a transition between Java and C# or VB.NET is relatively straightforward. The syntax in Java and C# are for example almost identical, but it can take some time though to get familiar with the .NET classes, the rich set of graphical components, and eventhandling, which is quite central in .NET. More comparisons between Java and .NET can be made; one major difference when it comes to working with User Interface (UI) or graphics is that Java's "Swing" classes can be quite frustrating to work with in order to get the desired end result (from personal experiences). In .NET on the other hand, creating UI's is really simple and can be created by a "drag and drop" technique to place the different components that Visual Studio offers.

4.1.1 VB.NET basic syntax

In the report there are code listed in some chapters, and in order to clarify the syntax of the VB.NET language, a brief explanation of the most frequently used syntax is listed here.

```
Imports Microsoft.VisualBasic

Public Class Class1

    ' Comments are written with '

    'declaring a variable by using 'Dim'
    Dim myVariable As String = "This is a String"

    'method declaration with two parameters
    Public Sub AMethod(ByVal firstParam As Integer, ByVal secondParam As String)
        'method body goes here
    End Sub

End Class
```

4.2 SharePoint

4.2.1 Motivation for evaluating Sharepoint 2007

The company had recently installed a Sharepoint 2007 solution to use as an intranet. It was therefore interesting to try and create a solution for the Sharepoint platform. Office Sharepoint Server 2007 was also interesting to evaluate due to several reasons:

- The company is working a lot with Excel and Sharepoint has Integrated support for publishing excel workbooks.
- Office SharePoint Server 2007 is designed to work effectively with other programs, servers, and technologies in the 2007 Office release.
- To see how SharePoint development can be done and evaluate if SharePoint is something for the company to work further with in the future.

So the beginning of the work with this thesis started with investigating how compatible Sharepoint 2007 is with the companys current solutions for presenting data. Sharepoint have support for displaying Excel workbooks so the possibilities for this feature was investigated in the beginning of the thesis. The first things to check was what kind of functionality that was supported within the excel workbooks when they are displayed within sharepoint, if they are editable, macros allowed, all functions allowed etc.

4.3 Sharepoint and Excel Services

Sharepoint's inherent possibilities to display an Excel file was tested with Sharepoint's built in feature Excel Services.

The evaluation of Excel Services was about trying to load several different workbooks, with different content, to see how they behaved in the sharepoint environment. The different workbooks tested contained following features: (Here reviewed from section 3.3)

1. A workbook with diagrams and ordinary celldata (simply a very common standard Excel document) and named cells.
2. A Workbook with links to another workbook
3. A Workbook that have cells referencing an external data source (such as a SQL Database).
4. A workbook containing VBA (Visual Basic for Applications) code.

4.4 Results of Excel services evaluation

1. A Workbook containing Charts and basic data in cells was properly displayed. The only difference from the original workbook was that the 3D diagram in the current example was converted to a 2D equivalent.

2. Loading a workbook containing a reference to another workbook failed, as external references (links between workbooks) isn't supported in excel services [2].

3. A workbook with cells that reference an external data source failed, support for this feature is currently missing.

4. A workbook containing VBA code failed to load into excel web access due to unsupported features for loading workbooks containing VBA code [2].

From Excelspecialisten's point of view, two very important features in Excel were not supported in Excel Services. Excelspecialisten works a lot with workbooks containing VBA code and links between worksheets and these two features were not supported. Displaying an Excel workbook can have it's uses and are gonna be used in certain cases, but more functionality is needed in order to achieve the companys requests of presenting and manipulating data. After some research, theory for solutions to problems (2) and (3) were found. But in order to solve problem number (4) there was no workaround found, so this problem is solved in another way described later in this chapter.

4.5 Development in SharePoint

Before the different solutions are presented, a brief description of SharePoint development is made. This section briefly describes what is required to get started with SharePoint development. Developers familiar with web development in ASP.NET and XML technologies is going to have good use for this knowledge, as SharePoint are built upon ASP.NET 2.0 and relies heavily on XML technologies. A personal reflection was that, with no prior knowledge of ASP.NET and XML technologies it is difficult to get familiar of the concepts.

4.5.1 Software requirements

Office SharePoint Server 2007 runs on Windows Server 2003 with SP1 or later. Before installation of Sharepoint 2007, Internet Information Services must be installed so that the computer acts as a web server, and it is also necessary to install Microsoft .NET Framework 3.0 and enabling ASP.NET 2.0 [20].

All content stored in SharePoint sites is saved in SQL Server databases, and developers do not need to interact with the database directly.

4.5.2 Development scenario 1

There are two development enviroment's that developers can choose from when developing SharePoint solutions. One is to work remotely, where developers are working locally on their own non-server operating system with Microsoft Visual Studio installed. Developers build solutions locally and then deploy the solutions to a server that have SharePoint installed. The advantages of working this way is that there are no need for the developers to install the server software required for running SharePoint on their local computers. By using one server, backup can also be more easily managed. This environment has several drawbacks though, probably the major one is that debugging the code remotely leads to many problems. When a developer debugs a site, all other developers are getting blocked from this site, leading to problems when more than one developer needs to debug the same site. Another problem is that developers needs privileges on the server while debugging.

4.5.3 Development scenario 2

The other alternative when developing SharePoint solutions is to work locally, where developers installs the software required on their workstations. This way of working is recommended by Microsoft and have several advantages over the remote solution [21]. When working locally, there is no risk that one developer blocks other developers when debugging, and no risk for possibly altering a solution made by a colleague. By working locally, the developer do not need to package the code and go through the deployment steps to the server before seeing the result, and this should be a great timesaver in the long run. A problem with this solution can arise when developers has other development work to do on the same workstation. This problem can be solved by running SharePoint on a Virtual Machine. A Virtual machine (VM) is a software implementation of a computer, and can be installed on the existing Operating System. By installing SharePoint on a VM, the developer can easily switch between the Sharepoint development environment and other environments when needed.

4.5.4 Building SharePoint Solutions Using Visual Studio 2008

Visual Studio 2008 includes tools for faster development of SharePoint applications by offering project templates for Web Parts and site definitions. In order to use these templates, an extension, named "Visual Studio 2008 Extensions for Windows SharePoint Services 3.0, v 1.2" for Visual Studio 2008 is required to be installed [22]. A Web Part is created by choosing the Web Part template as project type in Visual Studio. Visual Studio automatically creates the required references for the project and and also creates some minimum code required in order to run the Web Part without any configuration more than to specify where the Web Part should be deployed.

4.6 Solutions for Excel in Sharepoint

This section presents the results made for the Microsoft Office Sharepoint 2007 platform. Results for ASP.NET can be found in section 4.9.

4.6.1 Creating User Defined Functions

User Defined Functions (UDF) are custom functions that extend the capabilities of Excel 2007. Using UDF's it is possible to provide functionality that is not available with the native Excel function Library and build custom data feeds for unsupported data sources. Excel Services can be used to build UDFs based on the .NET Framework and deploy these UDFs to Microsoft Office Sharepoint Server 2007. These UDFs can then be called by Excel workbooks deployed to Excel Services. By using UDFs in this way can help solve two of the problems listed earlier, (2) External references to another workbook and (3) Allowing a SQL data connection in a workbook. Both of these features were as stated in section 4.4 not allowed in Excel Services, so by using managed-code UDFs for Excel Services there are possible workarounds to these limitations.

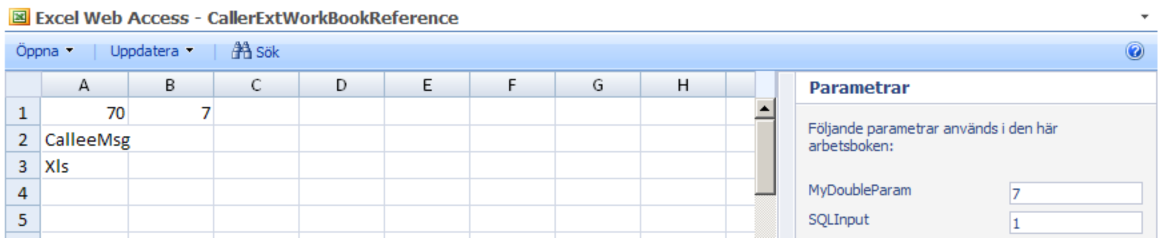
Short description in general on how to create managed code UDFs for Excel Services is as follows:

- Create a managed-code UDF project in Visual Studio (click Class library in the templates pane when creating new project).
- Add a reference for the project to the Excel Services UDF assembly.
- Create the custom functions.
- Configure Excel Services in Sharepoint (Central administration for Sharepoint, then choose SharedServices1) Under Excel services settings choose user-defined function assemblies, select add UDF and locate the User defined function.
- In order to create a workbook that calls the UDFs we can for example in cell A1 type the name of the UDF we wish to call, lets call it "Multiply(int number)" and this function takes an integer as argument and simply multiplies it with 10.
- To call the function Multiply(byval number as Double), type in cell A1: =Multiply(B1) , and type in 7 in cell B1. We have set the cell B1 as argument for the function, so in cell B1 we type in the value we want to send to the function as argument.
- In order to change the parameter, make cell B1 a named range by selecting it, clicking "define" in the defined name group (in the formulas tab) and type myDoubleParameter.

  By making B1 a named range it is later possible in Excel Services to make input in a textfield that passes the text to cell B1, this way the argument to the function Multiply(int number) can be changed directly from Sharepoint As seen in **??**

- The final step is to publish the workbook to Excel Services by clicking Microsoft Office Button in Excel and select ”Excel Services” and save the workbook to a trusted Sharepoint document library.

- The result after loading in Excel Services will be that cell A1 displays number 70 because we provided the function with parameter 7 in cell B1 and the function multiplied B1*10 which is 70 in our case.



	A	B	C	D	E	F	G	H
1	70	7						
2	CalleeMsg							
3	Xls							
4								
5								

Parametrar

Följande parametrar används i den här arbetsboken:

MyDoubleParam	7
SQLInput	1

Figure 3: In cell A1 the result from the function Multiply(byval number as Double) is shown with input 7. The value from the external workbook CalleeExtWorkbookReference.xlsx is shown in cell A2. SQL reference is also successful and displays its result in cell A3 with input CustomerID=1

4.6.2 Allowing External references in Excel Services

Excel Services does not allow that a workbook contains a reference to a cell in another workbook, as experienced when evaluating the functionality of Excel Services in different scenarios. This problem can however be solved using UDFs to provide the same functionality. The UDF that provide a solution for this problem opens the workbook that we want to reference and extracts the value in the cell we are looking for. The following code snippets illustrates how this can be done. For illustration purposes some code are left out in order to focus on the important aspects.

In the example we want to achieve the following: An Excel workbook ”CallerExtWorkBookReference.xlsx” is published to Excel Services and displayed in Sharepoint. This workbook contains a User Defined Function named ExternalReference() that opens another, external workbook ”CalleeExtWorkBookReference.xlsx” that we wanted to reference in the first place (Excel services did not allow this) , and return the value in cell A1 that is set to “CalleeMsg”. This way the same result can be achieved as if the workbook was referenced directly in the first place.

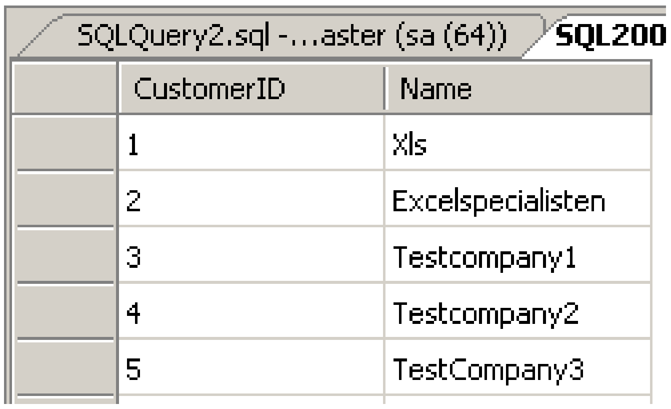
```
Imports Microsoft.Office.Excel.Server.Udf

'Mark the class with the UdfClass attribute when containing UDF methods
<UdfClass()> Public Class UDFMethodTest
```

```
    ' If a method in the UDF assembly are not marked with the UdfMethod attribute it is
    'because it is not considered as a UDF method
    <UdfMethod(isvolatile:=True, ReturnsPersonalInformation:=True)> _
    Function ExternalReference() As String

        ' path to the workbook that we want to reference and get data from
        Dim workbookPath As String = "http://ozcomp/Documents/CalleeExtWorkBookReference

        'initialize Excel Services
        Dim es As ExcelService = New ExcelServices.ExcelService()
        es.UseDefaultCredentials = True

        ' open the workbook
        Try
            sessionId = es.OpenWorkbook(workbookPath, "", "", status)

            'get the data that is inside cell A1 of the referenced workbook
            cellValue = es.GetCellA1(sessionId, "Blad1", "A1", True, status)

        Catch ex As Exception
            'exception handling
        End Try

        'return the cell value from the referenced workbook
        Return cellValue.ToString()

    End Function
End Class
```

4.6.3 SQL Connection in an Excel workbook

Excel services does not allow publishing a workbook that contains a reference to an external data source such as an SQL database. The same philosophy as with the problem with external workbooks applies to this SQL related problem. By using a User Defined Function that connects to an SQL database and fetches data it is possible to achieve the same result but with a little more work required. The purpose for this method is just to show that it is possible to reference SQL data from a workbook published in Excel Services. The data is retrieved from a table 'Customers' and returns the name of the Customer with the given customerID.

The method takes the customerID as argument.



	CustomerID	Name
	1	Xls
	2	Excelspecialisten
	3	Testcompany1
	4	Testcompany2
	5	TestCompany3

Figure 4: Customer Table

```
' mark method with UdfMethod attribute as earlier to indicate that it is an
<UdfMethod(isvolatile:=True, ReturnsPersonalInformation:=True)> _
   Function ReadFromSQL(ByVal customerID As Integer) As String

    Dim resultingValue As String

    'connection variables for database connection
    Dim connection As SqlConnection
    Dim command As SqlCommand
    Dim dataSet As DataSet
    Dim dataTable As New DataTable
    connection = Nothing
    command = Nothing

    Try
        'connection Info
        connection = New SqlConnection("Initial Catalog=UserProjects;Data Source=192.xxx
        Dim sqlCommand As New SqlCommand
        Dim storedProcedureAdapter As New SqlDataAdapter
        dataSet = New DataSet
        connection.Open()
        sqlCommand.Connection = connection
        sqlCommand.CommandType = CommandType.Text

        'The SQL query, retrieve a name from 'Customers' table that matches with Custome
        sqlCommand.CommandText = "SELECT Name FROM customers WHERE CustomerID=" + custom

        storedProcedureAdapter = New SqlDataAdapter(sqlCommand)
        storedProcedureAdapter.Fill(dataSet)
        dataTable = dataSet.Tables(0)
    Catch ex As Exception
        Return ex.Message
    End Try

    connection.Close()
    'get the value that resulted from the query
    resultingValue = dataTable.Rows(0).Item(0)
    Return resultingValue

End Function
```

4.6.4 Solution for using Excel documents containing VBA-Code (Macroactivated workbook) within Sharepoint Excel Web Access.

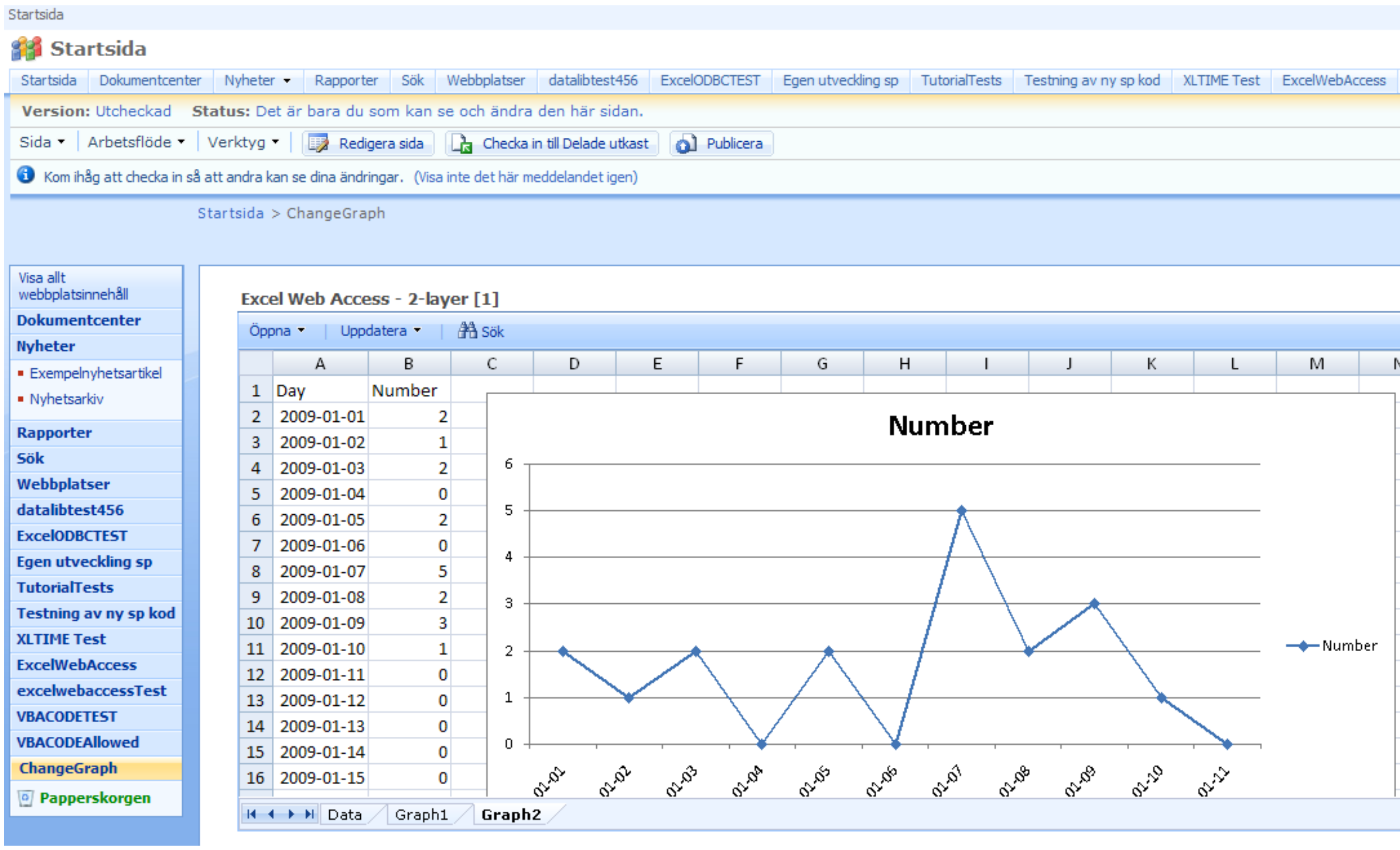
During the initial testing of Excel Services capabilities and limitations it was found that Excel Workbooks containing VBA-Code was not supported. This workaround was not solved using UDFs as was the case with the External workbook reference and SQL data source problem. A workbook cannot be loaded into Excel Web Access if it contains VBA code (macro-enabled), but nothing prevents us from opening the macro-containing workbook in a webpart in Sharepoint using a COM object and .NET using the COM object to copy the contents from the "forbidden" Macro workbook to a new workbook that is macro-free. The VBA-workbook opens in the background and after the VBA-code have executed, the content of the VBA-workbook is copied to a newly created, macro-free workbook that have permission to be loaded into Excel Services and published by Excel Web Access. The program for this solution creates a webpart containing an Excel Web Access area and loads the new workbook into this for display.

4.6.5 Generating new data dynamically in a Excel Workbook

An example program (webpart) was created to test if the following could be achieved within sharepoint: programmatically open a workbook saved in a trusted location in a sharepoint documentlibrary and alter some cells and save the workbook. After the change is done, the workbook loads into excel web access for display.

Excel workbook contents:

The workbook consists of 3 sheets, first sheet consists only of 2 columns, names and dates. Second sheet contains a graph of how many times a certain name from sheet1 is listed there. Sheet3 is also consisting of a graph, this time showing how many times a specific date has occured in sheet1. The main idea of the workbook is that sheet1 is the datalayer and sheet2 and sheet3 are acting like different presentation layers for the data in sheet1.


Startsida

Startsida

Startsida | Dokumentcenter | Nyheter | Rapporter | Sök | Webbplatser | datalibtest456 | ExcelODBCTEST | Egen utveckling sp | TutorialTests | Testning av ny sp kod | XLTIME Test | ExcelWebAccess

Version: Utcheckad **Status:** Det är bara du som kan se och ändra den här sidan.

Sida | Arbetsflöde | Verktyg | Redigera sida | Checka in till Delade utkast | Publicera

Kom ihåg att checka in så att andra kan se dina ändringar. (Visa inte det här meddelandet igen)

Startsida > ChangeGraph

Visa allt webbplatsinnehåll
Dokumentcenter
Nyheter
- Exempelnyhetsartikel
- Nyhetsarkiv

Rapporter
Sök
Webbplatser
datalibtest456
ExcelODBCTEST
Egen utveckling sp
TutorialTests
Testning av ny sp kod
XLTIME Test
ExcelWebAccess
excelwebaccessTest
VBACODETEST
VBACODEAllowed
ChangeGraph
Papperskorgen

Excel Web Access - 2-layer [1]

Öppna | Uppdatera | Sök

	A	B
1	Day	Number
2	2009-01-01	2
3	2009-01-02	1
4	2009-01-03	2
5	2009-01-04	0
6	2009-01-05	2
7	2009-01-06	0
8	2009-01-07	5
9	2009-01-08	2
10	2009-01-09	3
11	2009-01-10	1
12	2009-01-11	0
13	2009-01-12	0
14	2009-01-13	0
15	2009-01-14	0
16	2009-01-15	0

Figure 5: New values are randomly generated and written to the workbook every time the webpart is loaded

The current version of the program uses a COM object to load and modify the workbook. When the workbook has been loaded, new names and dates are randomly produced and written back to the workbook and saved. The program then creates a new Excel Web Access webpart on the page for display of the workbook. The program is a demonstration of the ability to write new data to a workbook published in Excel Web Access.

4.7 Testing SharePoint 2007 with SQL Database

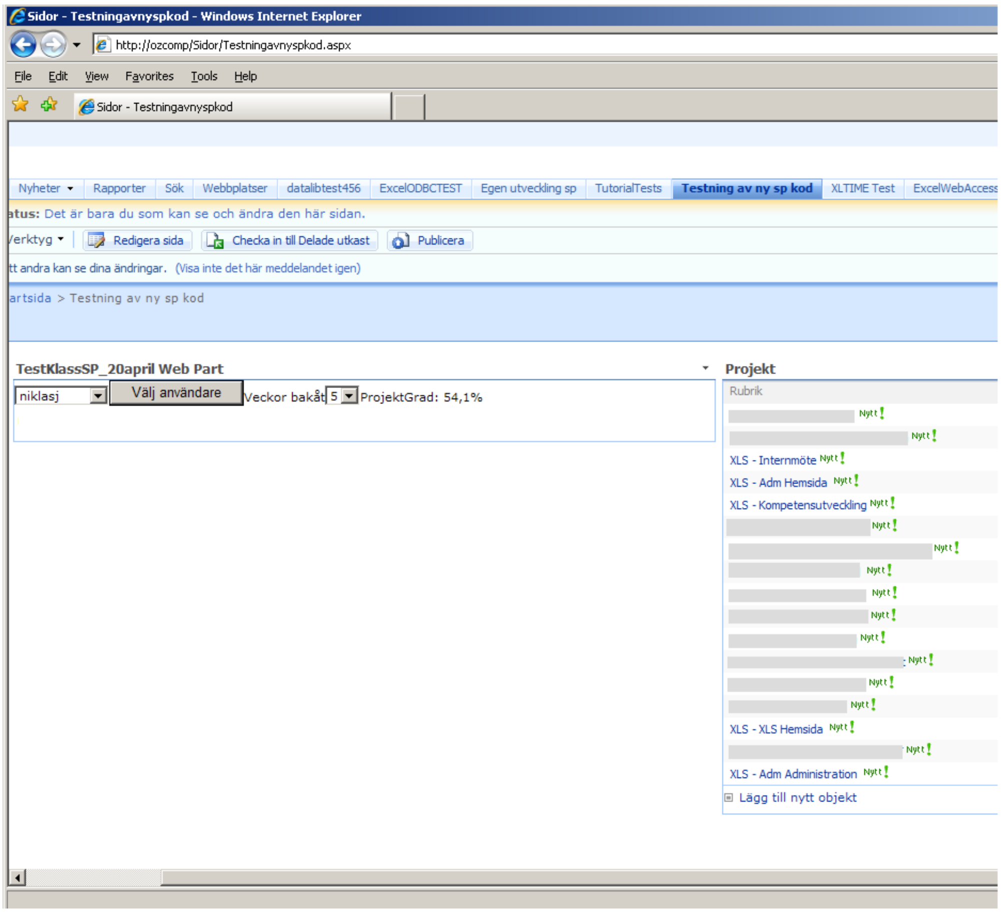
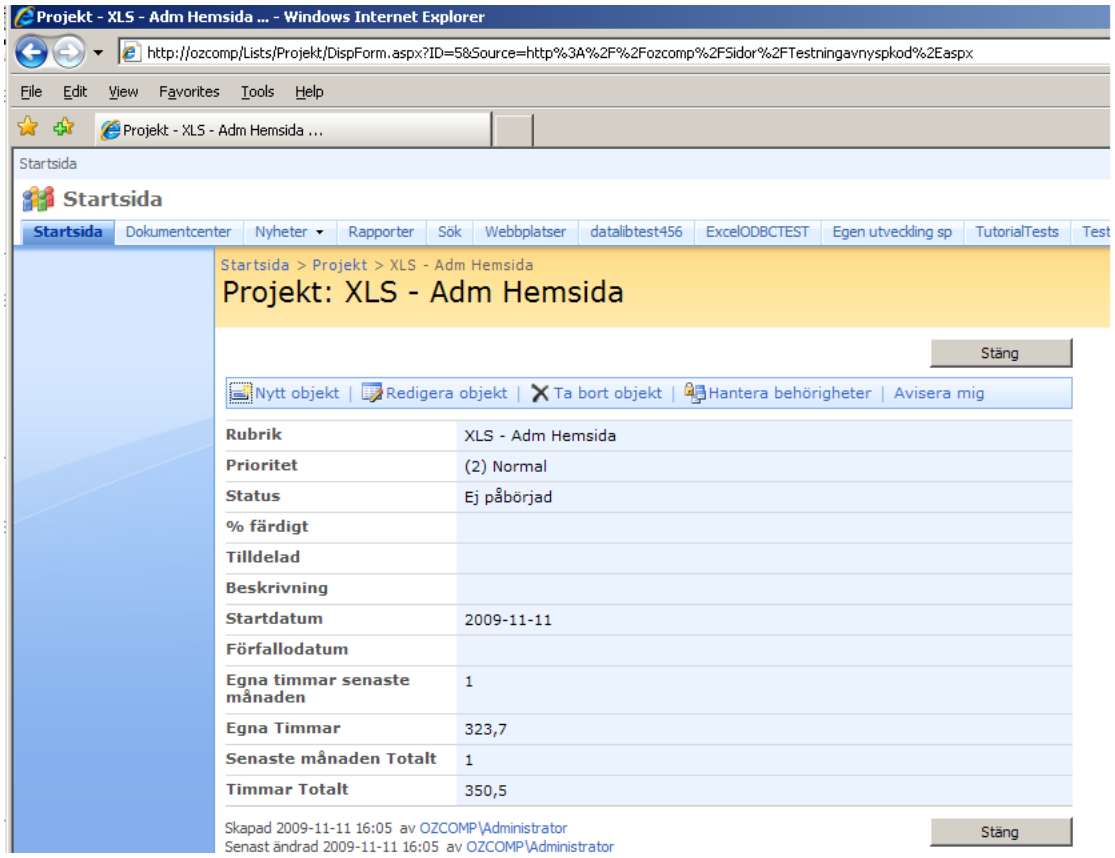

Sidor - Testningavnyspkod - Windows Internet Explorer
http://ozcomp/Sidor/Testningavnyspkod.aspx
File Edit View Favorites Tools Help
Sidor - Testningavnyspkod
Nyheter Rapporter Sök Webbplatser datalibtest456 ExcelODBCTEST Egen utveckling sp TutorialTests Testning av ny sp kod XLTIME Test ExcelWebAccess
atus: Det är bara du som kan se och ändra den här sidan.
Verktyg Redigera sida Checka in till Delade utkast Publicera
tt andra kan se dina ändringar. (Visa inte det här meddelandet igen)
artsida > Testning av ny sp kod
TestKlassSP_20april Web Part
niklasj Välj användare Veckor bakåt 5 ProjektGrad: 54,1%
Projekt
Rubrik
XLS - Internmöte Nytt!
XLS - Adm Hemsida Nytt!
XLS - Kompetensutveckling Nytt!
XLS - XLS Hemsida Nytt!
XLS - Adm Administration Nytt!
Lägg till nytt objekt

Figure 6: Screenshot of XLTime SharePoint, where user niklasj projects the last 5 weeks are displayed to the right. Several projects are hidden in this image in order to not reveal names of clients.

Excelspecialisten uses a time reporting system named XLTime, where employees record their worktimes on different project's they worked with. XLTime is developed by Excelspecialisten and is written in VBA. Sharepoint is used as an intranet within Excelspecialisten and the company thought that it could have its uses to present some data from the XLTime database in a sharepoint site. So an example project was developed with two purposes; To provide employees with useful information of their projects, and the second reason was to investigate limitations/possibilities of an sharepoint application. The example project are here referred to XLTime SharePoint.


Projekt - XLS - Adm Hemsida ... - Windows Internet Explorer
http://ozcomp/Lists/Projekt/DispForm.aspx?ID=5&Source=http%3A%2F%2Fozcomp%2FSidor%2FTestningavnyspkod%2Easpx
File Edit View Favorites Tools Help
Projekt - XLS - Adm Hemsida ...
Startsida
Startsida
Startsida Dokumentcenter Nyheter Rapporter Sök Webbplatser datalibtest456 ExcelODBCTEST Egen utveckling sp TutorialTests Test
Startsida > Projekt > XLS - Adm Hemsida
Projekt: XLS - Adm Hemsida
Stäng
Nytt objekt | Redigera objekt | Ta bort objekt | Hantera behörigheter | Avisera mig

Rubrik	XLS - Adm Hemsida
Prioritet	(2) Normal
Status	Ej påbörjad
% färdigt	
Tilldelad	
Beskrivning	
Startdatum	2009-11-11
Förfallodatum	
Egna timmar senaste månaden	1
Egna Timmar	323,7
Senaste månaden Totalt	1
Timmar Totalt	350,5

Skapad 2009-11-11 16:05 av OZCOMP\Administrator
Senast ändrad 2009-11-11 16:05 av OZCOMP\Administrator

Figure 7: Screenshot showing details for selected project "XLS Adm Hemsida"

4.7.1 XLTime SharePoint

The example projects main functionality is to retrieve data from the XLTime database and present it in Sharepoint. XLTime Sharepoint is a webpart to be used within an employees personal page in sharepoint. The webpart first checks which user is logged in and retrieves the users projects from the original XLTime database. The webpart retrieves the users most recent projects worked with (selected weeks from a drowdownlist) and creates a new SharepointList with the projects as listitems. By clicking on a list item (a specific project), more information about the project is displayed:

- The users worked hours on the project last month.
- The users total worked hours on project.
- The users projectgrade (percentage of projects that generates income for company).

- All users hours total for the project.

4.8 Sharepoint Business Data Catalog

Sharepoint Business Data Catalog (BDC) is a business integration feature in Microsoft Office Sharepoint Server 2007. BDC provides built-in support for displaying data from databases and webservices. One of the key features of BDC is that no coding except XML is required to present the data. XML is used to create an application definition file that is used to describe the data from the database, and it is quite a bit of work to develop this definition file by hand. A configuration and testing of a BDC was developed during this thesis. The BDC retrieved the data from a single table in an test database, and the result is shown below. The BDC is a good way of presenting data from databases, with filter options to even make smaller queries possible.

4.8.1 BDC Meta Man

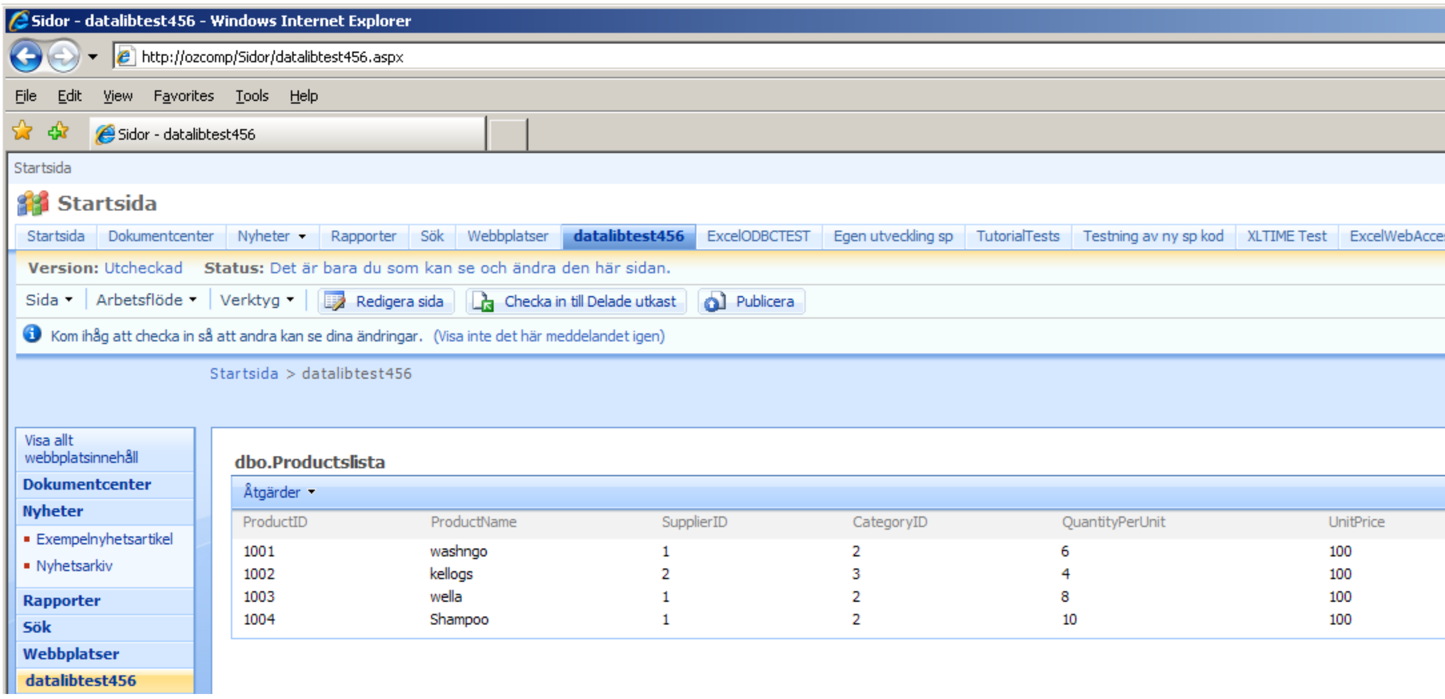
BDC Meta Man is a Business Data Catalog tool for Sharepoint that is created by LightningTools [26]. The version of the program that I used was a free trial version. BDC Meta Man can be used to generate an application definition file that is loaded into Sharepoint. This means that no XML code needs to be written and thus saves time and programming effort. Connection information (Server type,name, authentication, login and password) for the database is the only information needed as input and BDC Meta Man generates an application definition file in xml format. The generated application definition file is after that loaded into Sharepoint. To view the data one simply adds a new "Business Data List" web part and connects it to the application definition file. The instructions that are required to setup an BDC was created as an Microsoft PowerPoint presentation consisting of screenshots of the required steps, in order to make it easy to follow.

4.9 ASP.NET

4.9.1 Example project: Publishing Excel workbook on the Web

The application demonstrates some of the functionality that can be achieved when using Excel together with ASP.NET to create a connection between a worksheet and a webpage. In the current state, the application shows several examples of how to display and maintain some interactivity in a workbook presented on a webpage. The functionality in the application consists of the following features:

- Upload an existing workbook for display on the webpage. A .NET FileUpload component is used to let the user select a file on disk and upload it on the server for display. The workbook is also saved on the server.


Sidor - datalibtest456 - Windows Internet Explorer

http://ozcomp/Sidor/datalibtest456.aspx

File Edit View Favorites Tools Help

Sidor - datalibtest456

Startsida

Startsida

Startsida | Dokumentcenter | Nyheter | Rapporter | Sök | Webbplatser | datalibtest456 | ExcelODBCTEST | Egen utveckling sp | TutorialTests | Testning av ny sp kod | XLTIME Test | ExcelWebAcce

Version: Utcheckad Status: Det är bara du som kan se och ändra den här sidan.

Sida | Arbetsflöde | Verktyg | Redigera sida | Checka in till Delade utkast | Publicera

Kom ihåg att checka in så att andra kan se dina ändringar. (Visa inte det här meddelandet igen)

Startsida > datalibtest456

Visa allt webbplatsinnehåll
Dokumentcenter
Nyheter
- Exempelnyhetsartikel
- Nyhetsarkiv
Rapporter
Sök
Webbplatser
datalibtest456

dbo.Productslista

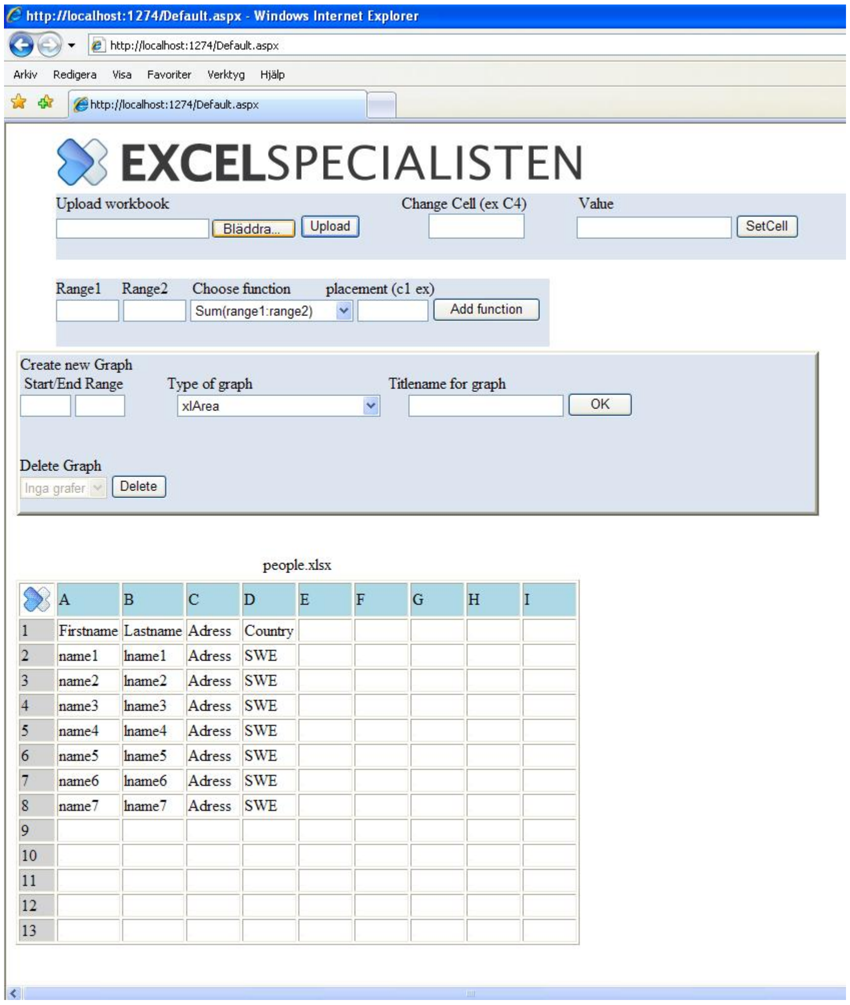
Åtgärder

ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice
1001	washngo	1	2	6	100
1002	kellogs	2	3	4	100
1003	wella	1	2	8	100
1004	Shampoo	1	2	10	100

Figure 8: A Business Data Catalog getting its data from the data table "dbo.Products"

- Change the cell content of any given cell in the workbook.
- Insert a function (for example Average value of a range of values) in the workbook. This feature is created by having two textboxes where the user defines a startrange and end range that defines where the data resides in the workbook. A third textbox is used to specify where in the workbook (in what cell) the result of the function call should be displayed. The ranges that should be specified are typed in the format "A1". To decide what kind of function to insert, there is a dropdown list with function names to choose from.
- Create a new graph and insert into the workbook. The graph is defined by a start and end range for the data in textboxes, the type of graph is selected from a dropdown list, and finally a title for the graph is given in a third textbox. The newly created graph is placed in the workbook just below the last graph in the workbook.
- Delete a graph

  The user can select from all graphs in the workbook from a dropdownlist containing all graph names and then press a button called 'Delete' to delete the selected graph from the workbook.
- Detect if a cell consists of a function or just an ordinary value like text or a number. In the webpage version of the workbook (same workbook, but webpage version have different design).



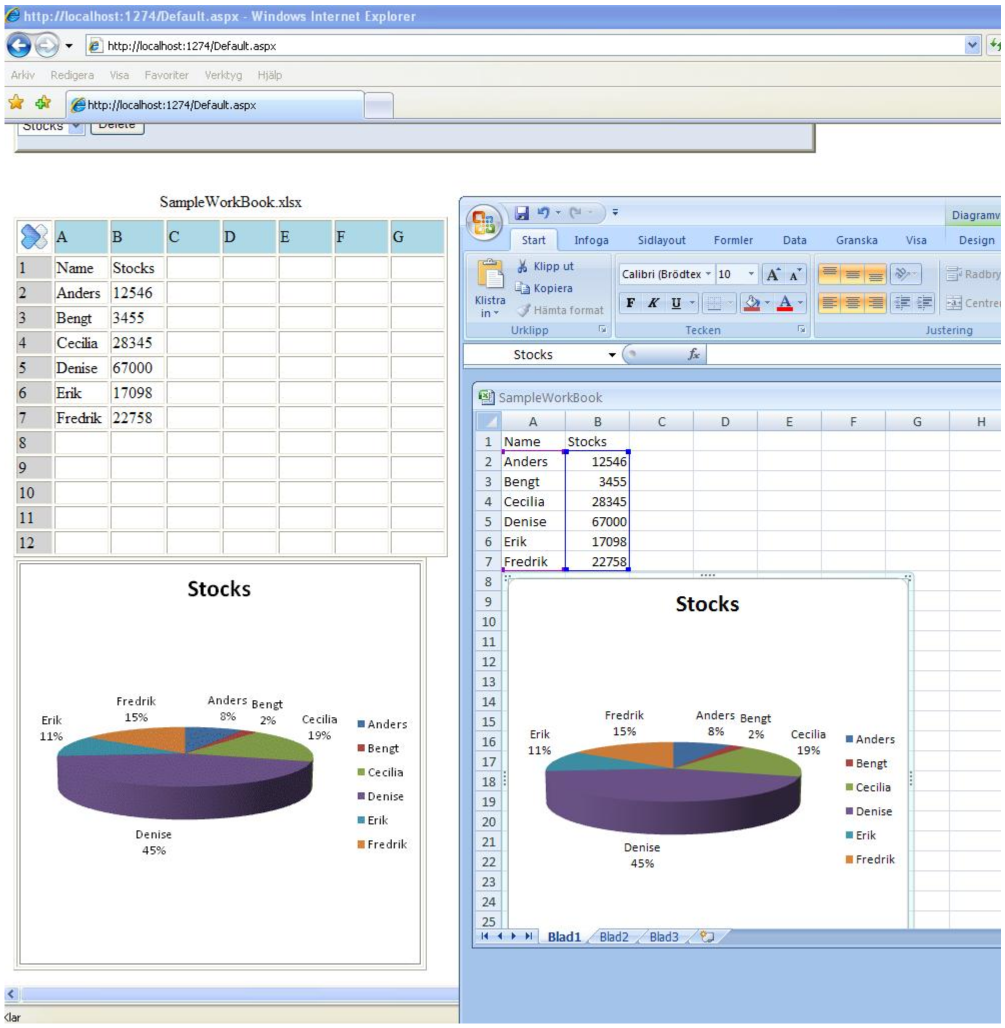
	A	B	C	D	E	F	G	H	I
1	Firstname	Lastname	Adress	Country					
2	name1	lname1	Adress	SWE					
3	name2	lname2	Adress	SWE					
4	name3	lname3	Adress	SWE					
5	name4	lname4	Adress	SWE					
6	name5	lname5	Adress	SWE					
7	name6	lname6	Adress	SWE					
8	name7	lname7	Adress	SWE					
9									
10									
11									
12									
13									

Figure 9: User Interface of the application

4.9.2 General description

The design of the 'webversion' of the workbook and the orginal workbook is a bit different. The "webversion design" of the workbook is made up of a HTML table with each tablecell representing the real cell in the workbook. If a workbook contains charts, these are converted to GIF images and placed in another table for display. The webversion workbook is trying to model the original workbook's data exactly, so that the user can be certain that the data in cell "A3" is the same in both versions of the workbook.

When working with Excel Com objects, it is important to release all related variables to these objects so that all Excel processes are closed. By closing the application and opening the Windows Task Manager it is easy to see if there are variables that are not released yet; if there still is an Excel process listed under 'processes' and no Excel program running, there is still a remain from the application that needs to be released.



SampleWorkBook.xlsx

	A	B	C	D	E	F	G
1	Name	Stocks					
2	Anders	12546					
3	Bengt	3455					
4	Cecilia	28345					
5	Denise	67000					
6	Erik	17098					
7	Fredrik	22758					
8							
9							
10							
11							
12							

Figure 10: Graph Creation: Graph published on web to the left and graph in the original workbook to the right

During the execution of the application, old Excel processes needs to be released so a call to CloseExcelProcesses() is made if there is a need to clear old Excel Processes.

4.9.3 Functionality and description of the program

- Creating graphs in Excel programmatically
- Displaying the graphs in the webpage

  The method Private Sub AddImages() is called in order to add all the graphs for display in the page in GIF-format. AddImages() creates a separate Html Table that is filled with all the images. The images in GIF-format is retrieved from the method Private Function GetChartsAndItsCoordinates() As List(Of Image) that returns a list of Images in GIF-format. What GetChartsAndItsCoordinates() does is exporting the charts in the worksheet to a GIF image and also maps each GIF image with its original placement in the workbook, (setting the image.ID attribute to the original coordinates) in order to know where the graph should be placed in the webversion of the workbook. The method also saves the GIF images to the server and then returns a list of the images in GIF format, that is used in AddImages() described above.

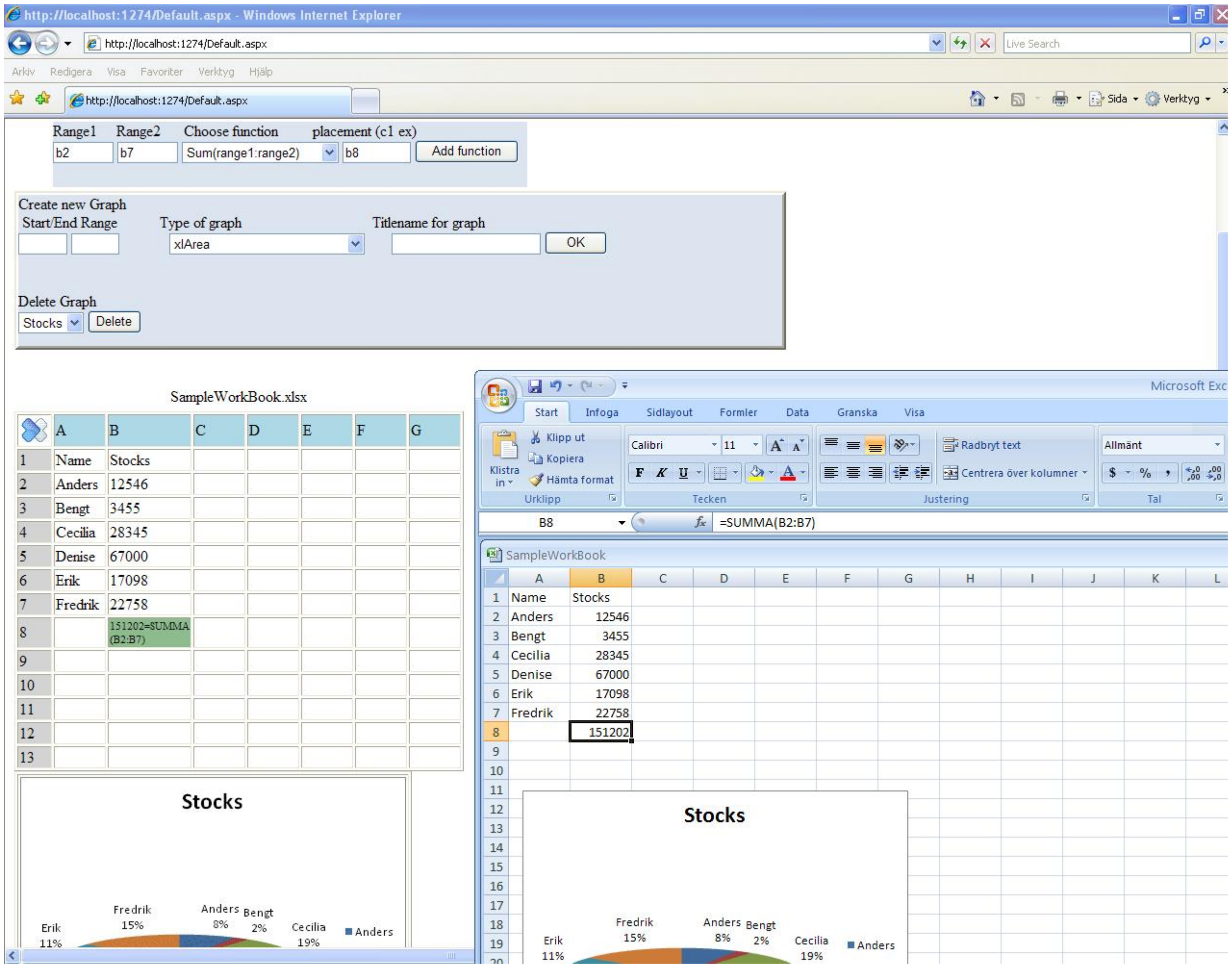
  Private Sub HandleGraphs() is then called to handle the placement of the newly created Graph, and places it below the graph that currently is placed furthest down in the workbook. HandleGraphs() calls DetectDiagrams() which retrieves the current worksheets Shape collection and returns a list with all the shapes (type:Excel.Shape) contained in that worksheet.

- Deletion of graphs

  The user selects a graph to be deleted from the workbook by choosing the name of the graph in a dropdown list control and presses the Button 'Delete Graph'. The eventhandler for button 'Delete Graph' is then called and deletes the selected Graph. The eventhandler also calls Private Sub DeleteGIFFile(ByVal filename As String) which delete's the corresponding GIF file to the Graph, to clean up GIF files that are not needed anymore.

- Upload the workbook

  A .NET FileUpload control is used in order to be able to upload a workbook for display on the webpage. If a valid file (currently .xls, .xlsm, xlsx formats) is choosen then it is saved to the server and is displayed on the webpage using the methods ExcelInit(FileUpload1.FileName) , ReadFromExcelToTable() and populateDropDownLists(). ExcelInit(FileUpload1.FileName) initializes all the Excel objects that are needed to communicate with Excel. ReadFromExcelToTable() then reads all cells from the selected workbook into a Html table and also maps the cells in the workbook with a corresponding cell in the Html table. PopulateDropDownLists() is populating the dropdownlists with the current workbooks graphs names.


http://localhost:1274/Default.aspx - Windows Internet Explorer
Arkiv Redigera Visa Favoriter Verktyg Hjälp
Sida Verktyg
Microsoft Exc
Start Infoga Sidlayout Formler Data Granska Visa
Klipp ut Kopiera Hämta format Klistra in
Calibri 11
Radbryt text Centrera över kolumner
Allmänt
Urklipp Tecken Justering Tal
Stocks
Erik 11%
Fredrik 15%
Anders 8%
Bengt 2%
Cecilia 19%
Anders

Range1	Range2	Choose function	placement (c1 ex)	
b2	b7	Sum(range1:range2)	b8	Add function

Create new Graph

Start/End Range		Type of graph	Titlename for graph	
		xlArea		OK

Delete Graph

Stocks Delete

SampleWorkBook.xlsx

	A	B	C	D	E	F	G
1	Name	Stocks					
2	Anders	12546					
3	Bengt	3455					
4	Cecilia	28345					
5	Denise	67000					
6	Erik	17098					
7	Fredrik	22758					
8		151202=SUMMA (B2:B7)					
9							
10							
11							
12							
13							

B8 =SUMMA(B2:B7)

SampleWorkBook

	A	B	C	D	E	F	G	H	I	J	K	L
1	Name	Stocks										
2	Anders	12546										
3	Bengt	3455										
4	Cecilia	28345										
5	Denise	67000										
6	Erik	17098										
7	Fredrik	22758										
8		151202										

Figure 11: Web and original representation of workbook with a created function that sums a range of numbers and places the result in cell B8.

The practical use with some minor changes to the application, is that it can be used to publish workbooks on the web, that allows some interactivity for the users. The application's main purpose is a demonstration of basic funtionality that can be developed when working with Excel and ASP.NET in conjunction.

Suggestions for future extensions and improvements to the application is to place textboxes in every cell of the table and map a corresponding cell in the workbook to the textbox, which makes it easier to input new data to the workbook and the user dont have to specify a cell specifically for input. There is also a possibility to use a .NET component called GridView to use. This requires a bit more work but is probably worth it if the application is gonna be used in a "real" situation, as it is possible to adjust the look of the GridView to almost look identical to Excel cells.

4.10 Using ADO.Net and Connectionstrings to read/write data

Reading and writing data is a major part of software development and can be done in many ways. If one method to read data works in a specific environment, nothing ensures that this method works good in other environments or situations. Multiple ways of writing and reading data has therefore been investigated in this thesis, to find methods that can be applicable in different situations.

4.10.1 Connection strings

When an application connects to a database or another datasource, it uses a driver or provider to initiate the connection [23]. The connection strings purpose is to provide the information that the driver or provider need to know in order to make the connection to the data source. A connection string consists of several keywords, separated by semicolons. There are different providers/drivers available and therefore a connectionstring can be written in many ways. A solution that used connectionstrings was created in order to demonstrate the ability to transfer data between a .NET application and a workbook. The ability to transfer data between a .NET application and a database was also tested. These methods worked good and can work as a way of reading data. Using connectionstrings to transfer data between a workbook and a .NET application is best suited for situations when there is only data that needs to be transfered. Graphs and Images cannot be transfered this way.

4.10.2 The .NET component GridView

In order to publish the content of an Excel workbook on the web, a powerful .NET component named GridView can be used. By using a combination of .NET classes together with a GridView it is possible to create a solution to publish the data contained in a workbook on the web (or elsewhere also). A connectionstring is passed to a OleDbConnection object, which represents an open connection to a data source. An OleDbCommand (represents an SQL statement) object is then created and associated with the OleDbConnection to know what datasource the SQL statement should be executed in. A OleDbDataAdapter object is created for acting as a bridge between a DataSet (created next) and the datasource (workbook in our case). The OleDbDataAdapter object is passed the OleDbCommand containing the SQL statement and after that we create a DataSet object to hold the information from the worksheet and then use this DataSet as the datasource for the GridView component. The end result is that the data part from the workbook (not diagrams or images) is displayed in the gridview on the webpage. A gridviews structure resembles the structure of a workbook and can therefore be suitable as a container for the workbook data.

4.10.3 Limitations

The technique discussed in this chapter is suited only for situations when data needs to be read or written. If a workbook contains Shapes, like graphs or Images, these will not be exported to the gridview, only the data will. To export graphs and solve other problems related to Excel, a more feature rich ASP.NET test application was developed (described earlier), which used a COM object to communicate with Excel. It is not possible to insert formulas in cells using ADO.NET.

4.10.4 Benefits over COM automation

The ADO.NET technique reads data faster due to the many interface requests that has to be done each time a cell is accessed when using a COM object.

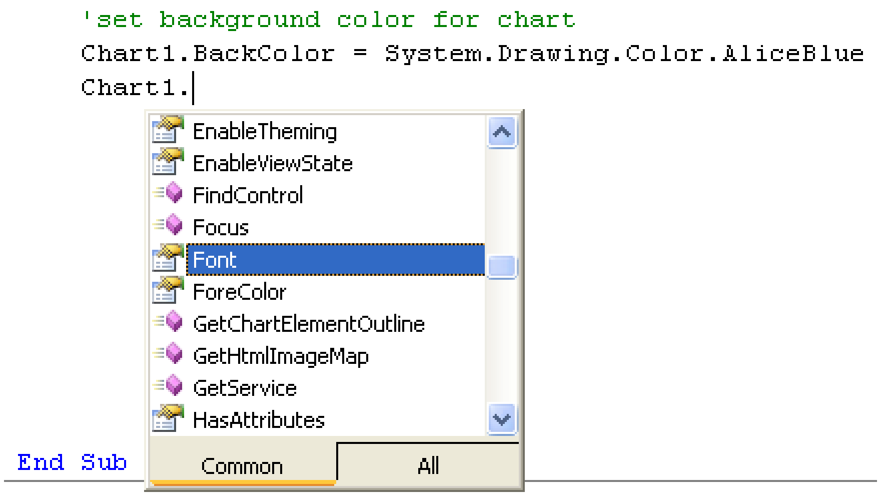
No need for Office to be installed on server (as there are no need to use COM object). More stable and simpler solution than with a solution that uses a COM object.

4.11 ASP.NET Charts

When displaying data of some type, it can often be preferred to present the data visually to the viewer rather than a table of numbers and values. Microsoft have created a free to use .NET control named “Microsoft Chart Controls” that have many good features and options for displaying quality graphs.

4.11.1 Microsoft Chart Controls for Microsoft .NET Framework 3.5

Microsoft Chart Control for .NET Framework provides a way for developers to create data visualization. Developers can use the Chart control to create charts in ASP.NET and Windows Forms applications [24].



```
'set background color for chart
Chart1.BackColor = System.Drawing.Color.AliceBlue
Chart1.
End Sub
```

Figure 12: Programmatically setting attributes of the chart control

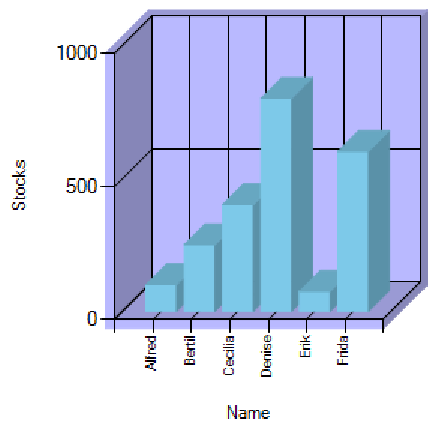
The Chart Control has many settings that can be configured directly in visual studio from the UI. However, The System.Web.UI.DataVisualization.Charting namespace contains methods and properties for the Chart control (for the ASP.NET version). Using the classes and methods in this namespace, developers have great control of customizing the charts programmatically. The Chart Control can be created by "drag and drop" from the UI or programmatically created in code.

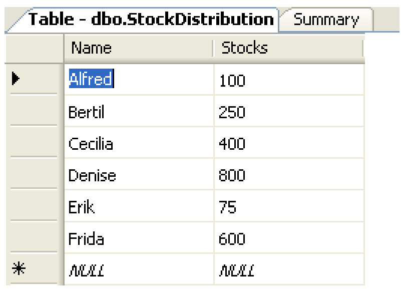
The Chart control can be configured to retrieve data from different types of data sources. The different data sources that are available:

- DataView
- Data readers (SQL, OleDB)
- DataSet
- DataTable
- Binding Source
- IDataSource
- Arrays
- Lists
- All Enumerable objects
- SqlCommand / OleDbCommand (DataSource data-binding only)

- SqlDataAdapter / OleDbDataAdapter (DataSource data-binding only)

In this example, data is retrieved from a Database where some peoples stock holdings are registered. (From test database)





Name	Stocks
Alfred	100
Bertil	250
Cecilia	400
Denise	800
Erik	75
Frida	600
NULL	NULL

Figure 13: Final chart with the Database Table that was used as the datasource

The Chart Control can also be used in Sharepoint but requires a bit more configuration in order to work properly.

Requirements:

- Supported Operating systems: Windows XP (SP3), Windows Vista, Windows Server 2008, Windows Server 2003 (SP2)
- .NET Framwork 3.5 with Service Pack 1 needs to be installed.
- Microsoft Chart Controls Add-on for Microsoft Visual Studio 2008

  This add-on provides toolbox integration and IntelliSense for the ASP.NET and Windows Forms Chart controls.

4.12 The Office XML File Format

Office XML is a file format built upon the open standard Extensible Markup Language (XML) and the ZIP format, and was introduced and used in Microsoft Office Excel 2007, Microsoft Office Word 2007, and Microsoft Office PowerPoint 2007 [25]. By using this file format, any application that supports XML can access and work with data stored in a Office XML file, without having Microsoft Office installed. The components that comprise a Office XML document is contained in a "package" and consists of several "parts", where a part corresponds to one file in the package.

4.13 Structure of the Office XML file format

The fileformat is based on the compressed ZIP file format and are made up of several parts. Most parts within the ZIP container (or package) are XML files that describe application data, but other non-XML parts may also be included within the container package. These non-XML files can be binary files representing images or OLE objects embedded in the document. There are also relationship parts in the package, that specify relationships between parts. To describe the structure of an Office Open XML file in short; The parts make up the content of the file and the relationships describe how the pieces of content work together. The experience as a user is still the same when opening a Office 2007 document in that the user is only working with just a single file. The modularity in the Office Open XML file format is an important characteristic, as modularity enables you to choose a specific part of the whole document and work only with that. This is also good if a part of a file is corrupt, then the other parts can still be intact.

4.14 Macro-enabled files (workbooks that contains VBA)

Macro-enabled files have the same file format as macro-free files but contains a binary part that stores the VBA project. It is possible to know in advance if a document contains VBA code or not by inspecting the package file and look for these specific VBA binary parts, and remove them to be sure of not running any potentially harmful code.

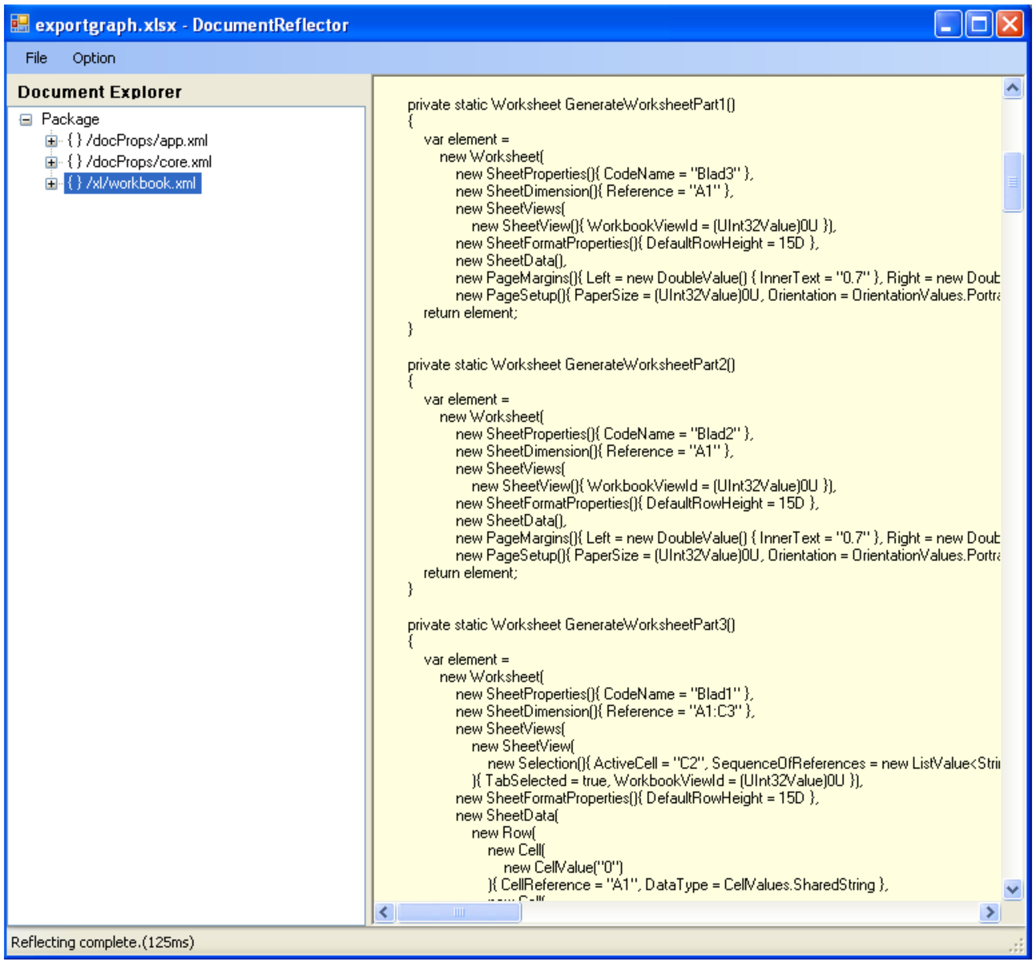
4.15 Developing applications using the Office XML formats

4.15.1 The Open XML SDK 2.0 for Microsoft Office

Microsoft offers developers to download and use the Open XML SDK 2.0 for Microsoft Office, which simplifies the task of manipulating Office Open XML files and complex operations can be achieved with only a few lines of code. The SDK is built on top of the System.IO.Packaging API and requires Microsoft.NET Framework version 3.5 to be installed.

4.15.2 DocumentReflector

The SDK contains a set of tools that can simplify Office Open XML development. DocumentReflector is a great tool to use to learn Office Open XML. With DocumentReflector it is possible to open Office Open XML files and see the underlying code that generated the whole document. This way it is possible to open for example an Excel file (.xlsx) and look how this file is structured and designed, and even take code snippets from it and use in own solutions. The code that is generated in DocumentReflector is written in C#.



```
private static Worksheet GenerateWorksheetPart1()
{
    var element =
        new Worksheet(
            new SheetProperties(){ CodeName = "Blad3" },
            new SheetDimension(){ Reference = "A1" },
            new SheetViews(
                new SheetView(){ WorkbookViewId = (UInt32Value)0U }),
            new SheetFormatProperties(){ DefaultRowHeight = 15D },
            new SheetData(),
            new PageMargins(){ Left = new DoubleValue() { InnerText = "0.7" }, Right = new Doub
            new PageSetup(){ PaperSize = (UInt32Value)0U, Orientation = OrientationValues.Portra
    return element;
}

private static Worksheet GenerateWorksheetPart2()
{
    var element =
        new Worksheet(
            new SheetProperties(){ CodeName = "Blad2" },
            new SheetDimension(){ Reference = "A1" },
            new SheetViews(
                new SheetView(){ WorkbookViewId = (UInt32Value)0U }),
            new SheetFormatProperties(){ DefaultRowHeight = 15D },
            new SheetData(),
            new PageMargins(){ Left = new DoubleValue() { InnerText = "0.7" }, Right = new Doub
            new PageSetup(){ PaperSize = (UInt32Value)0U, Orientation = OrientationValues.Portra
    return element;
}

private static Worksheet GenerateWorksheetPart3()
{
    var element =
        new Worksheet(
            new SheetProperties(){ CodeName = "Blad1" },
            new SheetDimension(){ Reference = "A1:C3" },
            new SheetViews(
                new SheetView(
                    new Selection(){ ActiveCell = "C2", SequenceOfReferences = new ListValue<Stri
                ){ TabSelected = true, WorkbookViewId = (UInt32Value)0U }),
            new SheetFormatProperties(){ DefaultRowHeight = 15D },
            new SheetData(
                new Row(
                    new Cell(
                        new CellValue("0")
                    ){ CellReference = "A1", DataType = CellValues.SharedString },
```

Figure 14: Screenshot of DocumentReflector

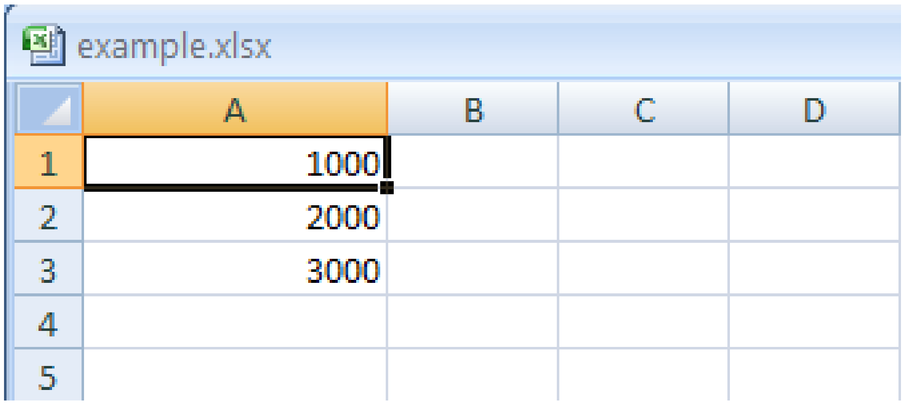


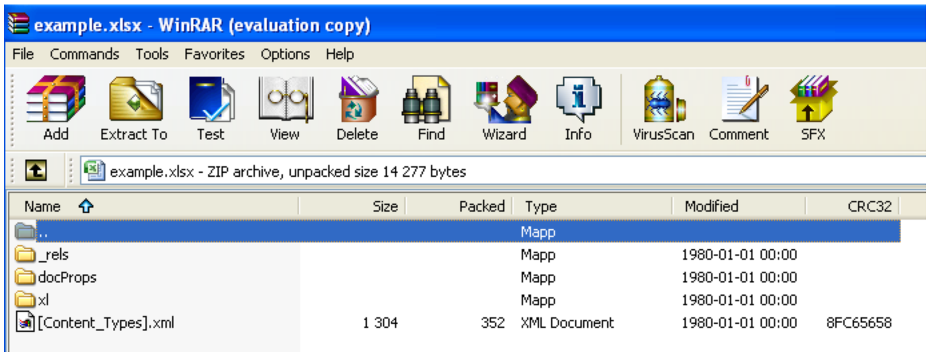
Figure 15: The contents of the Excelfile

4.15.3 The structure of an Office Open XML file

As mentioned earlier, the Office Open XML file format is based on the ZIP and XML file formats. It is therefore possible to open an Office Open XML as if it were a ZIP file, by using WinRAR or WinZip for instance. Just to get an impression of how the Office Open XML file format is structured, let's inspect an excel workbook to see how, for example the data in the cells are stored and represented. In this example, a new Excel workbook named "example.xlsx", with values written in cells A1, A2 and A3, was created and opened in WinRAR for inspection.

The file "example.xlsx" is with Office Open XML terminology called a package. A package contains two kinds of internal components, called parts and items. Items is subdivided into relationsip items and content type items, and in general an item contains metadata that describes the parts. A part contains content, where the majority is text files serialized as XML together with an XML schema. Parts can also consist of binary data, for example if a document contains an image or another media file. Inside the folder named "_rels" all relationship items are stored. Relationship items defines associations between parts. By having separate items that defines relationships between parts, there is no need to inspect the parts in question to discover relationships between them.

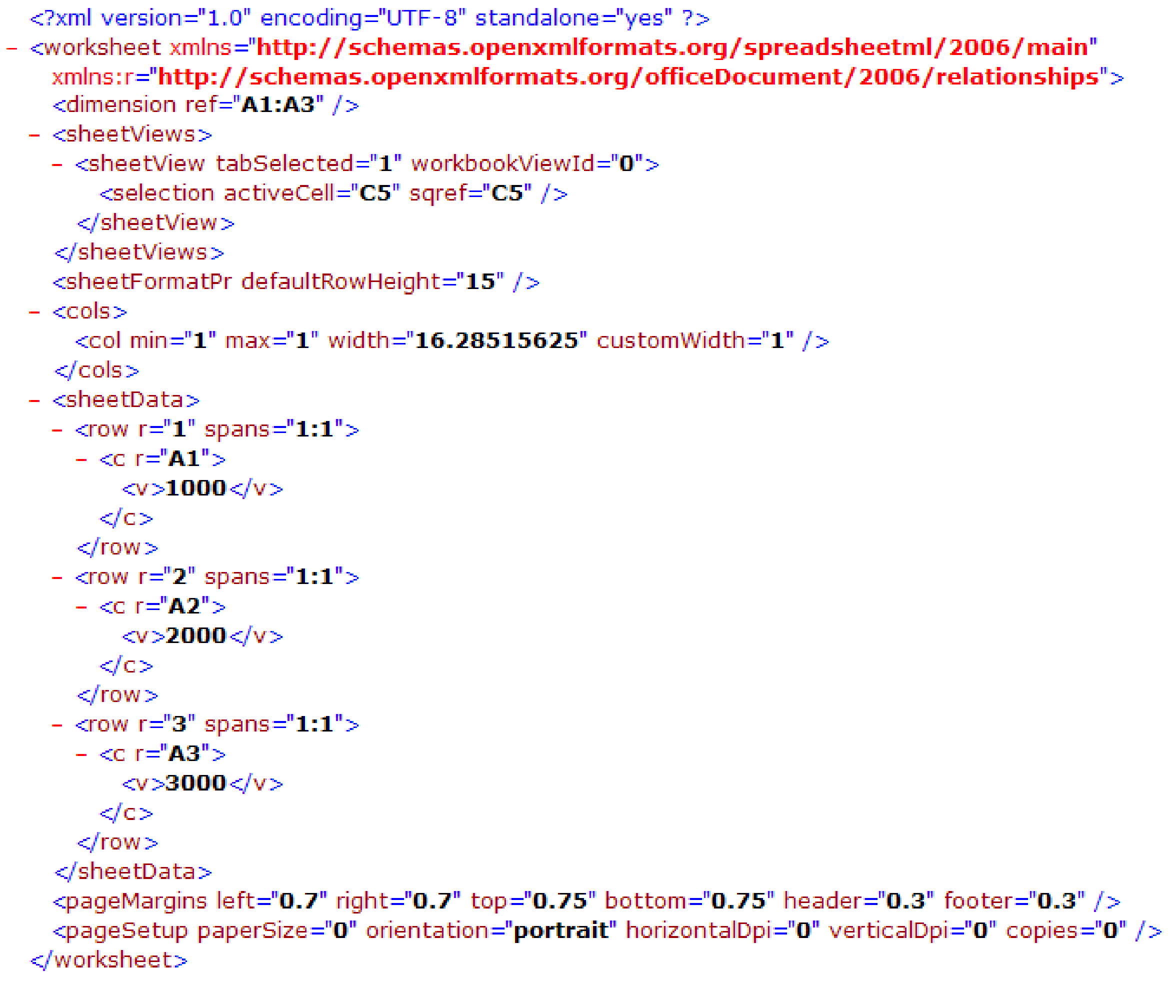
Now back to the example file "example.xlsx": If we navigate to the folder "example.xlsx/xl/worksheets" we find the worksheets for the document. The worksheets are represented as xml files named Sheet1.xml Sheet2.xml and Sheet3.xml as we currently have only three sheets in the workbook. By opening the XML file Sheet1.xml and inspecting it we can see how the contents of a workbook is represented in XML. The data is represented by a row and column number tags and the cells value, and we can see in the how the workbooks three values 1000, 2000 and 3000 are represented in XML format.


example.xlsx - WinRAR (evaluation copy)
File Commands Tools Favorites Options Help
Add
Extract To
Test
View
Delete
Find
Wizard
Info
VirusScan
Comment
SFX

example.xlsx - ZIP archive, unpacked size 14 277 bytes

Name	Size	Packed	Type	Modified	CRC32
..			Mapp		
_rels			Mapp	1980-01-01 00:00	
docProps			Mapp	1980-01-01 00:00	
xl			Mapp	1980-01-01 00:00	
[Content_Types].xml	1 304	352	XML Document	1980-01-01 00:00	8FC65658

Figure 16: The structure of an Office Open XML file, in this case an excel workbook, opened in WinRAR.



```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
    xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships">
    <dimension ref="A1:A3" />
  - <sheetViews>
    - <sheetView tabSelected="1" workbookViewId="0">
        <selection activeCell="C5" sqref="C5" />
      </sheetView>
    </sheetViews>
    <sheetFormatPr defaultRowHeight="15" />
  - <cols>
      <col min="1" max="1" width="16.28515625" customWidth="1" />
    </cols>
  - <sheetData>
    - <row r="1" spans="1:1">
      - <c r="A1">
          <v>1000</v>
        </c>
      </row>
    - <row r="2" spans="1:1">
      - <c r="A2">
          <v>2000</v>
        </c>
      </row>
    - <row r="3" spans="1:1">
      - <c r="A3">
          <v>3000</v>
        </c>
      </row>
    </sheetData>
    <pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75" header="0.3" footer="0.3" />
    <pageSetup paperSize="0" orientation="portrait" horizontalDpi="0" verticalDpi="0" copies="0" />
  </worksheet>
```

Figure 17: XML representation of some cells in a Excel workbook

4.15.4 Development

The development part within the Office Open XML consisted mostly of investigating the file system of Open XML and to try out some test methods to see that it actually worked as intended. These methods was assembled into a function library, to use as a starting point in order to get started with Office Open XML development. Listing of signatures of some methods in the function library:

```
'Method to create a new workbook
Public Sub CreateNewExcelWorkBook(ByVal filename As String, ByVal nrOfSheets As Integer)

    'Method to use when debugging, when developing in Office Open XML,
    'the files can be corrupted if relationships aren't set
    'properly. This method is a way to get a description of what is
    ' wrong with the corrupted file in question.

Private Function GetErrorDescription(ByVal filename As String) As String

    'A method to delete a specific sheet in a workbook.

Public Sub DeleteSheet(ByVal fileName As String, ByVal sheetToDelete As String)


    'writes to a specified cell in a workbook
    Public Sub WriteToCell(ByVal fileName As String,
                          _ByVal sheetName As String,
                          _ByVal addressName As String,
                          _ByVal value As String)


    'returns cell value from given cell
    Public Function GetCellValue(ByVal fileName As String,
                             _ByVal sheetName As String,
                             _ByVal addressName As String) As String


    ' exports a chart from Excel document to a chart representation in XML format
    ' The purpose of this method was to try exporting a chart into XML and then use
    ' the xml chart together with ASP.NET charts.
    ' ASP.NET Charts have an option to use XML as datasource,
    'but probably the XML data have to be structured
    ' in a certain way in order to work together with ASP.NET Charts.
    Public Sub ExportChartToXML(ByVal fileName As String,
                                _ByVal outputChartName As String,
                                _ByVal chartTitle As String)
```

Methods to build a word document was also developed, in order to learn about setting references and the general structure of a word document in Office Open XML format.

4.16 Benefits of using Office Open XML format

When developing Office-based solutions using Office Open XML, these are the advantages I found:

- There is no longer need for Office to be installed in order to create or modify a Office document.
- COM automation is not primarily designed for use on Servers. The applications within the Office family are designed for use on client computers and not in a server environment. In many situations the user is prompted for input and the program halts until the user have pressed on the “OK” button or a similar action. If the Office application resides on a server, no user is available to provide the proper input in order for the application to carry on. Therefore, many stability issues that can lead to program crashes or halts are avoided by using Office Open XML to develop instead.
- XML is a text-based format that compresses very well, and this together with the ZIP technology leads to significant reductions in file size of a Office XML document. File sizes can be up to 75 percent smaller compared to a comparable binary document, which leads to less disk space required to store files and less bandwidth needed to transport files over networks and across the web [25].
- The document format is used in Microsoft Excel 2007, Microsoft Word 2007 and Microsoft Powerpoint 2007. As these applications all use the same file format, interoperability between these documents is possible without using COM automation.

4.17 Limitations

I found no way to present Charts or Images on the web with the Office Open XML technology. As the Charts are stored in XML format, there probably has to be an instance of Excel available in order to render this XML as a chart.

5 Conclusions

5.1 Developed applications

One of the goals with the thesis was to develop an "Example project that retrieves data from ExcelSpecialisten's database and presents it on the web or SharePoint"

Two major example projects was developed, XLTime SharePoint and an ASP.NET application that provides web functionality for a workbook. From these projects many things was discovered and tested. Below is some conclusions that was drawn from these two projects and also from other projects developed in this thesis:

5.1.1 XLTime SharePoint

When I started to develop this SharePoint solution, my intention was to investigate the possibilities to write an webpart that retrieves data from a database and presents it in SharePoint.

The second reason, and almost as important, was to get a general feeling of how SharePoint development was. I felt that it could be interesting for the company (and for myself) to see if it was something worth continuing working with.

The results/conclusions from this project was

- Demonstrate and give examples of SharePoint development. In presentations show some example applications developed in SharePoint and to show the basics of how to get started with SharePoint development.
- Show that working with databases and SharePoint together works as in any other .NET project.
- The project can provide as an starting example to learn from if the company starts with SharePoint development.

5.1.2 ASP.NET application

This .NET application gave examples of some interactivity that can be achieved for a workbook presented on the web. The application was built with ASP.NET and COM automation. This application showed that it is possible to:

- Upload a workbook to the server and present its contents on the web.
- Present cell data (on web) that is stored in the uploaded workbook.

- Present Charts (on web) that exists in the workbook.
- Write new values into the original workbook from the webinterface.
- Insert new functions into the original workbook from the webinterface.
- Create new graphs dynamically in the workbook.

5.1.3 Developed webparts for Excel Services and Excel Web Access in SharePoint

These examples was created in order to see the possibilities (and possible limitations) for publication of Excel workbooks in SharePoint. The learning outcome from these webparts was that a lot of functionality in the workbooks was not supported if they were to be published in Excel Web Access. The webparts also shows several examples of functionality that can be created with help of the Excel Services API. With Excel Services API, workarounds for some of the unsupported features in workbooks was created.

- A solution for workbooks that contain a reference to another workbook.
- Similar solution for workbooks that contain a reference to a SQL database.
- A solution that serves as a way of keeping the design part of a workbook that contains VBA code.
- Solution that dynamically writes new data to a workbook presented in Excel Web Access.

5.1.4 Microsoft Chart Controls for .NET Framework 3.5

The chart controls was tested in order to provide an easy way to present data visually on the web in chart form. These charts proved to be really good and there are many different settings and types of graphs in this downloadable and free addon for Visual Studio. Charts can be customized both visually and programmatically.

5.1.5 Office Open XML format

COM automation can be unstable and slow when running on a server. The Office Open XML format provides an alternate way of manipulating Office files without an Office installation on the server. When writing applications that is going to run on a server, it is in many ways better to use Office Open XML technology to manipulate office files over COM automation. However, when working with Charts or Images in Excel, no way to present the Charts or

Images on the web was found with the Office Open XML technology. As the Charts are stored in XML format, there has to be an Instance of Excel available in order to render this XML, and that is unfortunately a problem that remains unsolved within this thesis. My suggestion here is if there is a need to present charts/images on the web with the Office Open XML format, one way to do it is to use COM automation only for this part and just convert the Chart(s) to GIF format with the COM object. The COM object is used as little as possible, only to convert the charts/images in the workbook to GIF and then terminate the COM object and do the rest with Office Open XML. By minimizing the use of the COM object, the risk for the application to crash is also minimized. If web applications that are communicating with Office files are to be developed, my recommendation is to use Office Open XML in as many situations as possible.

References

[1] Microsoft, *Introduction to Excel Services and Excel Web Access* , http://office.microsoft.com/en-us/sharepointserver/HA101054761033.aspx , 2009

[2] Microsoft, *Unsupported features in Excel Services* , http://msdn.microsoft.com/en-us/library/ms496823.aspx , 2009

[3] Microsoft, *Common Language Runtime Overview*, http://msdn.microsoft.com/en-us/library/ddk909ch.aspx , 2009

[4] Microsoft, *.NET Framework Conceptual Overview*, http://msdn.microsoft.com/sv-se/library/zw4w595w(en-us).aspx , 2009

[5] Microsoft, *Getting Started with Visual Studio*, http://msdn.microsoft.com/sv-se/vstudio/dd823311(en-us).aspx , 2009

[6] Microsoft, *Using IntelliSense*, http://msdn.microsoft.com/en-us/library/hcw1s69b(VS.71).aspx , 2009

[7] Microsoft, *ADO.NET Architecture*, http://msdn.microsoft.com/en-us/library/27y4ybxw

[8] Microsoft, *ADO.NET DataSet*, http://msdn.microsoft.com/en-us/library/zb0sdh0b

[9] Microsoft, *Office Excel 2007 product overview*, http://office.microsoft.com/en-us/excel/HA101656321033.aspx , 2009

[10] Microsoft, *Visual Basic for Applications Frequently Asked Questions*, http://msdn.microsoft.com/en-us/isv/bb190540.aspx , 2009

[11] Microsoft, *Visual Basic for Applications*, http://msdn.microsoft.com/en-us/isv/bb190538.aspx , 2009

[12] Microsoft, *Microsoft Office SharePoint Server 2007 product overview*, http://office.microsoft.com/en-us/sharepointserver/HA101656531033.aspx , 2009

[13] Microsoft, *Determine hardware and software requirements (Office SharePoint Server)*, http://technet.microsoft.com/en-us/library/cc262485.aspx , 2009

[14] Microsoft, *Plan browser support (Office SharePoint Server)*, http://technet.microsoft.com/en-us/library/cc263526.aspx , 2009

[15] Microsoft, *Introduction to Excel Services and Excel Web Access*, http://office.microsoft.com/en-us/sharepointserver/HA101054761033.aspx , 2009

[16] Microsoft, *Excel Services Architecture*, http://msdn.microsoft.com/en-us/library/ms582023.aspx , 2009

[17] Microsoft, *Understanding Excel Services UDFs*, http://msdn.microsoft.com/en-us/library/ms499792.aspx , 2009

[18] Microsoft, *Extending the Excel Services Programmability Framework*, http://msdn.microsoft.com/en-us/library/bb267252.aspx , 2009

[19] ExcelSpecialisten, *ExcelSpecialisten XLS AB Webpage*, http://www.excel-specialisten.se/ , 2009

[20] Microsoft, *Install Office SharePoint Server 2007 on a stand-alone computer*, http://technet.microsoft.com/en-us/library/cc263202.aspx , 2009

[21] Microsoft, *Development Tools and Techniques for Working with Code in Windows SharePoint Services 3.0 (Part 1 of 2)*, http://msdn.microsoft.com/sv-se/library/bb530302

[22] Microsoft, *Building SharePoint Solutions Using Visual Studio 2008*, http://msdn.microsoft.com/sv-se/library/cc537498(en-us).aspx , 2008

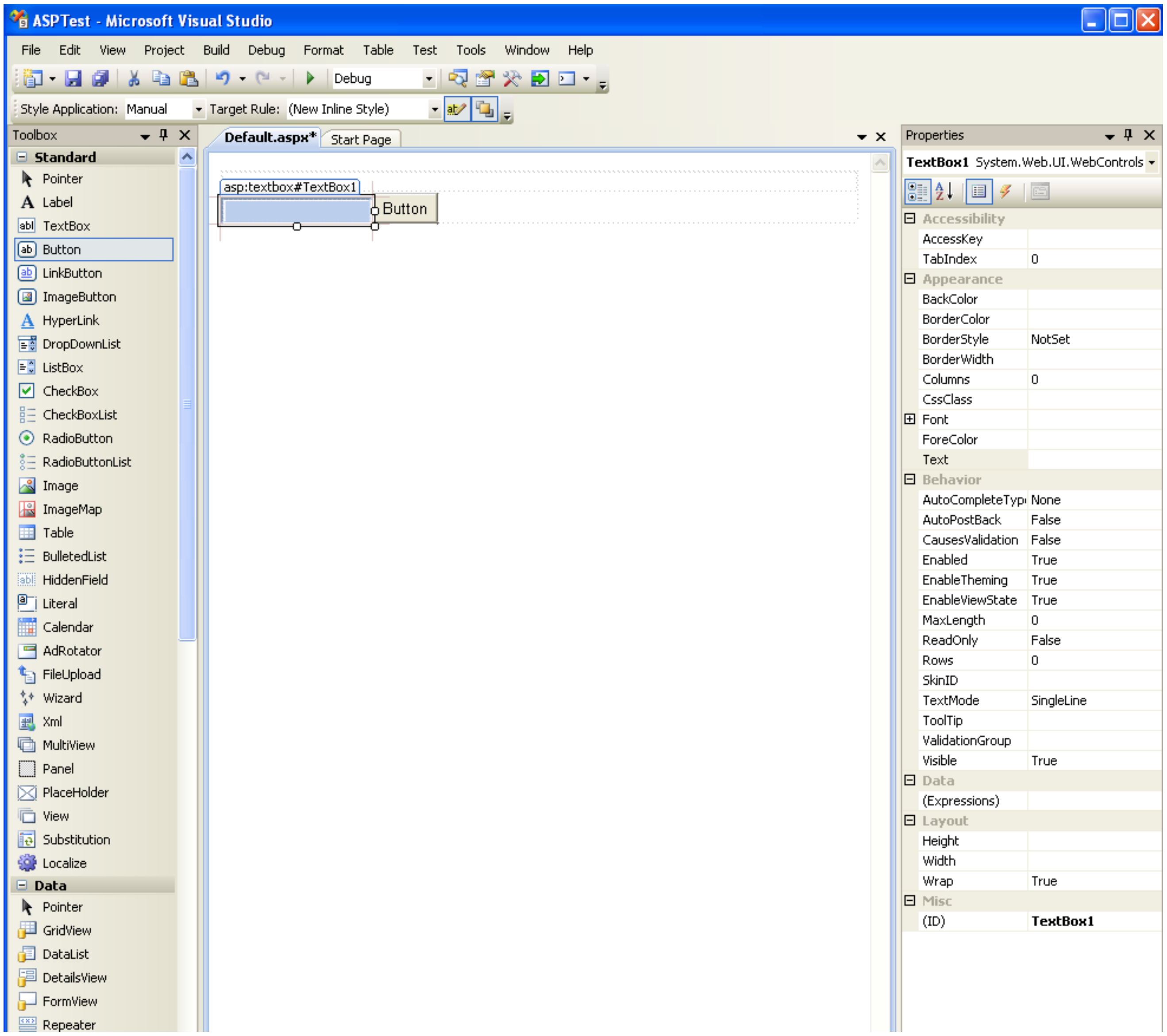
[23] Connectionstrings.com, *Connection Strings Explained*, http://www.connectionstrings.com/Articles/Show/what-is-a-connection-string , 2008

[24] Microsoft, *Microsoft Chart Controls for .NET Framework Documentation*, http://www.microsoft.com/downloads/details.aspx?FamilyId=EE8F6F35-B087-4324-9DBA-6DD5E844FD9F&displaylang=en , 2009

[25] Microsoft, *Introducing the Office (2007) Open XML File Formats*, http://msdn.microsoft.com/en-us/library/aa338205.aspx , 2009

[26] Lightning Tools, *Business Data Catalog Tools to get you working in seconds - BDC Meta Man*, http://www.lightningtools.com/bdc-meta-man/default.aspx , 2007

6 Appendix



TextBox1 System.Web.UI.WebControls

Accessibility	
AccessKey	
TabIndex	0
Appearance	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
Columns	0
CssClass	
Font	
ForeColor	
Text	
Behavior	
AutoCompleteTyp	None
AutoPostBack	False
CausesValidation	False
Enabled	True
EnableTheming	True
EnableViewState	True
MaxLength	0
ReadOnly	False
Rows	0
SkinID	
TextMode	SingleLine
ToolTip	
ValidationGroup	
Visible	True
Data	
(Expressions)	
Layout	
Height	
Width	
Wrap	True
Misc	
(ID)	**TextBox1**

Figure 18: Visual Studio 2008 Userinterface. To the left is the toolbar that can be used to "drag and drop" graphical controls. The right side displays the property window for the selected textbox control.