CHALMERS

Development of support web applications in .NET

For Visit Technology Group

Master of Science Thesis in Software Engineering and Technology

ANDERS CLAESSON
CHRISTOPHE DUBRAY

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

Development of support web applications in .NET
For Visit Technology Group

Anders Claesson
Christophe Dubray

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 June 2009

Acknowledgments

We would like to thank Visit Technology Group for the opportunity to do this thesis at their company. We would also like to thank all personal of Visit for a warm welcome and an exciting working environment, especially our supervisor Jonas Olsson. Finally we would like to thank our examiner Sven-Arne Andreasson.

Table of Contents

Sammanfattning

Det finns två populära ramverk för webbutveckling för .NET plattformen, båda utvecklade av Microsoft. Det största av dem är ASP.NET Web Forms vilket grundar sig på Windows Forms modellen för att erbjuda möjligheter till snabb utveckling av webbapplikationer. Det andra ramverket, ASP.NET MVC, är signifikant tunnare och lättare att skala. ASP.NET MVC stödjer endast de mest fundamentala HTML-komponenterna i kontrast till Web Forms som levererar flertalet serverkomponenter.

Under examensarbetets olika faser har både ASP.NET Web Forms och ASP.NET MVC använts. De har utgjort stommen för arbetet men även kombinerats med tekniker som remoting, dependency injection och object relational mapper för att uppnå en robust applikationsarkitektur. I rapporten förekommer även en jämförelse mellan .NET och Java Enterprise Edition plattformen från SUN på en övergripande nivå.

Första fasen av examensarbetet består av vidareutveckling och förbättring av supportverktyget Citybreak Support, utvecklat i ramverket ASP.NET Web Forms. Citybreak Support används av företaget Visit Technology Group för kommunikation med sina kunder. Andra fasen består av utvecklandet av ett mindre Customer Relational Management system i ASP.NET MVC ramverket. Fördelar och nackdelar med dessa två ramverk kommer att diskuteras och utvärderas i denna rapport.

Abstract

There are two popular web application frameworks for the .NET platform, both developed by Microsoft. One of them being ASP.NET Web Forms which takes the event-driven Windows Forms model approach to rapid web application development. The other framework is ASP.NET MVC, which is significantly more extensible and lightweight. ASP.NET MVC supports only the most fundamental HTML components; in comparison Web Forms comes with several server controls.

Both ASP.NET Web Forms and ASP.NET MVC were used in the thesis work during different phases. While these frameworks were the major technologies used, other techniques such as remoting, dependency injection, and object relational mapper were used in conjunction to achieve solid application architecture. A high level comparison between the .NET and Java Enterprise Edition platform is also conducted.

First phase of the thesis work consists of further development of the support tool Citybreak Support, developed in ASP.NET Web Forms, used by the company Visit Technology Group to interact with their customers. Second phase includes development of a lightweight Customer Relationship Management system using the ASP.NET MVC framework. The strengths and weaknesses of the two web frameworks are also evaluated and discussed.

Keywords: ASP, NET, MVC, C#, Visit, dependency injection, repository pattern, LINQ, customer relationship management, object relation mapper, remoting, Java, web, application, Microsoft, SQL, Citybreak support

1 Introduction

This report will describe the steps taken and some of the techniques behind the further development of Citybreak Support, the support tool used by Visit Technology Group.

"Visit Technology Group designs Internet based solutions that enhance the business performance and profitability of tourist organizations, destination management organizations, inbound tour operators, hotel groups and resorts worldwide.[1]"

At this point the in-house developed web-based support tool, Citybreak Support, originally developed in 2006, mainly implemented in ASP.NET 1.0, plays a central part of Visit's process management, development and customer relations. It is the foundation of communication between Visit Technology Group and their customers.

The support system is a place for dialog that handles everything from regular questions about the Citybreak products to requests for further development of Citybreak products. It involves and affects all Visits personal from call center support to in-house developers. Since it is a core function for Visit Technology Group, it is of outmost importance that it works effectively and gives strong support in their daily work.

However Citybreak Support in its current form does not fully meet the requirements and needs of its users. As the customer base has grown, the amount of data contained in the system has increased dramatically. This has resulted in degraded system performance and trouble grasping an overview as the amount of data has increased. In addition, new requirements and wishes have surfaced during the years of usage. This combination of issues has resulted in slower production rate and higher costs for the company.

Visit Technology Group has provided a long list of new features they would like to add and current features that need to be improved. The specific details about the current support system and its functionality and the requests for further improvement will be disclosed later in this introduction.

1.1 About the Company

Visit Technology Group is a privately held company incorporated in 1999. It was founded by its current Chief Execute Officer (CEO) Magnus Emilson. At that point the company only had one employee but grew rapidly. Today, February 2009, Visit Technology Group has 27 employees with offices in three different cities. One located in Stockholm with four employees, one in Nantes, France with two employees and the head office and development center is located in

[1] Visit Technology Group, About the company, http://www.visit.com/about 2009-05-26

Göteborg with 21 employees. The company also has a sister company by the name ITicket that works with online booking of tickets.

Today Visit Technology Group is a one of the leading suppliers of internet based travel solutions for the tourism and travel industry. Citybreak was launched in 2000 and has been adopted by both large and small industry players from Northern Europe to Southern Africa and USA.[2]

1.2 Current state of Citybreak Support

This section gives you a short introduction to how the support system works today, which functionality it offers and which restrictions it has. Currently, there are two types of users in the system. First there is the company employee user, the administrator, which has no restrictions and can do everything in the system. Next there is the customer user, which has heavy restrictions on what they may create, read, update or delete and is limited to issues that only concern their organization.

The basic idea is that a user creates an issue whenever he or she has a concern related to a Citybreak product. These issues can be of three different types: question, request or error. Each issue is also bound to a severity level: urgent, medium or low. Here the options end for the customer user. However as an administrator it is possible to attach several additional attributes to an issue such as project, responsible person and which support line it belongs to.

The support system is built up with three different support lines where the first one is non-technical support, second one is technical support and third one is for development and bug fixes. All new issues created by a customer are automatically assigned to first line support. If they fail to solve the problem they pass it on to the second line which makes a deeper investigation and either solves the problem or pass it onto the third line. Third line is the final stop and the issue has to be resolved here.

An administrator may assign someone as responsible for an issue and the issue may in turn have several staff members assigned to it in order to complete it. Since an issue can be anything from a small question to a request for new development, it is possible to divide an issue into smaller tasks. If there is a larger development involved, an administrator is able to create a project. A project may contain many issues, which may in turn contain many tasks.

Every user of the support system has their own inbox connected to them, which can be used to send and receive private messages concerning issues. It is also possible to create public comments related to an issue, in that case no private message will be sent.

[2] Visit Technology Group, About the company, http://www.visit.com/about 2009-05-26

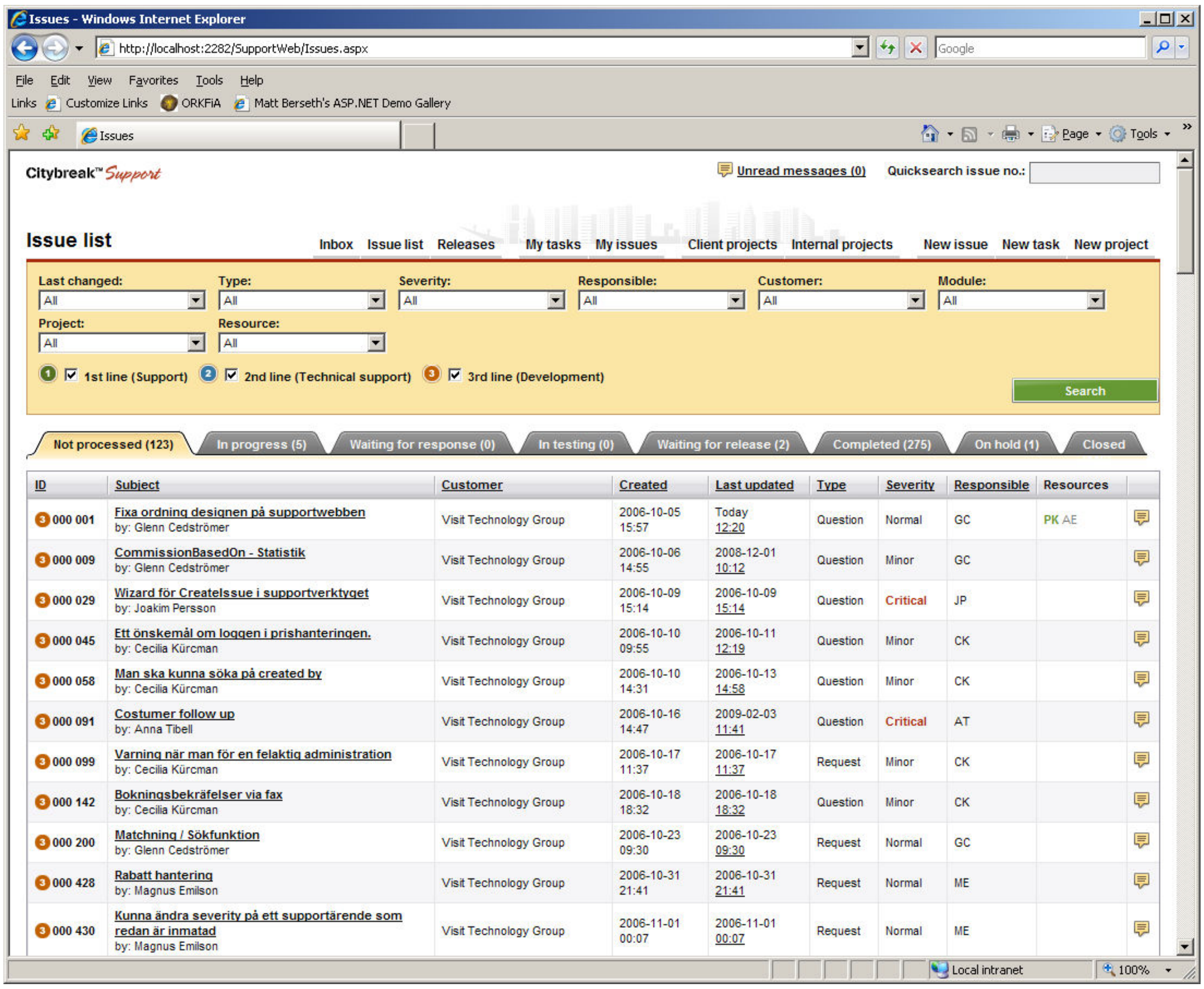
To keep track of what happens to an issue after it has been created, all issues have a status connected to them. At the time of creation, all issues have the status Not Processed. This means that the issue is waiting to be assigned to someone that will take responsibility for handling the issue. An issue may, after its initial status, be changed to one of the following, depending on where in its lifecycle it currently resides:

- **In progress** - Someone is assigned to work on solving the issue.
- **In testing** - The problem has been solved and is now in the testing phase.
- **Waiting for release** - The issue has passed the testing phase and will be a part of the next release of the product.
- **Waiting for response** – A solution to the issue has been suggested and is now waiting for response from the customer.
- **Completed** - The issue is completed.
- **On hold** - The issue is on hold for undecided time.
- **Closed** - When status has been set to completed, a user can close the issue which means the customer has accepted the suggested solution and the concern has been solved.

All users can login to the system at all times to see the status of their issues.

One very important feature of the support system is the possibility to sort and filter listings by different attributes. Because of the large amounts of issues that exists and are constantly being created, there exists several predefined listings for the administrators to gain an overview.

- **My Issues** - Shows a list of all issues where the current user is assigned to as a resource and does not have their current status set to completed or closed
- **Inbox** - Shows all issues that have not yet been assigned any staff members as resources at all
- **Issue List** - Shows all issues and provide many different viewing options for how to filter and sort the displayed issues. Issue list also exists for a non-administrator user but is limited to only show issues from the user's organization and provides a smaller set of sort and filtering options.


Issues - Windows Internet Explorer
http://localhost:2282/SupportWeb/Issues.aspx
Google
File Edit View Favorites Tools Help
Links Customize Links ORKFiA Matt Berseth's ASP.NET Demo Gallery
Issues
Page Tools
Citybreak™ Support
Unread messages (0)
Quicksearch issue no.:
Inbox Issue list Releases My tasks My issues Client projects Internal projects New issue New task New project
Search
Local intranet
100%

Issue list

Last changed: All
Type: All
Severity: All
Responsible: All
Customer: All
Module: All
Project: All
Resource: All

1 ☑ 1st line (Support) 2 ☑ 2nd line (Technical support) 3 ☑ 3rd line (Development)

Not processed (123) | In progress (5) | Waiting for response (0) | In testing (0) | Waiting for release (2) | Completed (275) | On hold (1) | Closed

ID	Subject	Customer	Created	Last updated	Type	Severity	Responsible	Resources	
3 000 001	Fixa ordning designen på supportwebben by: Glenn Cedströmer	Visit Technology Group	2006-10-05 15:57	Today 12:20	Question	Normal	GC	PK AE	
3 000 009	CommissionBasedOn - Statistik by: Glenn Cedströmer	Visit Technology Group	2006-10-06 14:55	2008-12-01 10:12	Question	Minor	GC		
3 000 029	Wizard för CreateIssue i supportverktyget by: Joakim Persson	Visit Technology Group	2006-10-09 15:14	2006-10-09 15:14	Question	Critical	JP		
3 000 045	Ett önskemål om loggen i prishanteringen. by: Cecilia Kürcman	Visit Technology Group	2006-10-10 09:55	2006-10-11 12:19	Question	Minor	CK		
3 000 058	Man ska kunna söka på created by by: Cecilia Kürcman	Visit Technology Group	2006-10-10 14:31	2006-10-13 14:58	Question	Minor	CK		
3 000 091	Costumer follow up by: Anna Tibell	Visit Technology Group	2006-10-16 14:47	2009-02-03 11:41	Question	Critical	AT		
3 000 099	Varning när man för en felaktig administration by: Cecilia Kürcman	Visit Technology Group	2006-10-17 11:37	2006-10-17 11:37	Request	Minor	CK		
3 000 142	Bokningsbekräfelser via fax by: Cecilia Kürcman	Visit Technology Group	2006-10-18 18:32	2006-10-18 18:32	Question	Minor	CK		
3 000 200	Matchning / Sökfunktion by: Glenn Cedströmer	Visit Technology Group	2006-10-23 09:30	2006-10-23 09:30	Request	Normal	GC		
3 000 428	Rabatt hantering by: Magnus Emilson	Visit Technology Group	2006-10-31 21:41	2006-10-31 21:41	Request	Normal	ME		
3 000 430	Kunna ändra severity på ett supportärende som redan är inmatad by: Magnus Emilson	Visit Technology Group	2006-11-01 00:07	2006-11-01 00:07	Request	Normal	ME		

Figure 1-1: A screenshot of the old Issue List page

1.3 Purpose

There are many purposes with the work that is to be carried out. First and foremost to deliver an improved and new version of the support tool that will meet as many requirements from Visit Technology Group as possible within the set time frame. Since there is a limited time frame the most important features will be prioritized by Visit Technology Group.

Another purpose is to get a broader perspective of web development. Prior to this master thesis we have only done web development using Java EE environments. This makes it interesting to compare our prior experience with the one in ASP.NET. The purpose of the report is to describe the development process, tools, techniques and design decisions being used while performing the task.

1.4 Limitations

We realized from the beginning that we were not going to be able to implement all of the requirements that Visit Technology Group had. Our strongest limitation for that decision is time. The plan is to spend around 20 week's fulltime on the company working with the development of Citybreak Support. During this time we decided to work in iterations treating the most important (prioritized by the company) functionality first.

Another very strong limitation is the current implementation of the Citybreak Support. Since we are not making a complete new version from scratch (mostly to save time), we have to adapt to the current implementation. This implementation is in ASP.NET Web Forms using code behind with the language C#, .NET Remoting and Microsoft SQL Server among others. These techniques will be discussed more in detail later.

2 Method

Visit Technology Group provided a requirement specification, a list of what they wanted improved and added to Citybreak Support. The requirement specification was a high level description in general terms. Due to the vague specification of requirements and lack of previous experience concerning ASP.NET development, the decision was made to use an agile development.

For each phase, Visit Technology Group created a prioritized list of requirements they wished to have included in that phase. The contents of that list were then discussed more in-depth together with Visit Technology Group to get a better picture. To minimize the risks for the company, an iterative process was chosen and a rather small portion of functionality was selected from the prioritized request list for implementation.

A mutual decision was made to not treat any other functionality until the prioritized functionality was completed. These decisions were based on the fact that the largest risk for the company was to end up with a lot of incomplete functionality, which would be unusable. Further reason for the decision was the fact that we had no prior experience, which made it very hard to estimate the time it would take to implement the requested functionality.

Every phase in itself was also an iterative process with weekly meetings where the last week's works was presented and feedback was received. Acceptance testing was used in order to decide whether certain functionality was completed or not. When all functionality for a phase was accepted that phase was considered completed and the work moved on to the next phase.

3 Requirement Specification

3.1 First Phase

Citybreak Support is currently developed using the ASP.NET Web Forms framework in the language C#, .NET Remoting and Microsoft SQL Server. Hence all further development in this phase will be conducted in this environment. This phase only contains requirements for improvements and changes to Citybreak Support.

- Change the customer priorities (low, medium, urgent) into something more descriptive
- Add functionality for grouping issues together
- Add functionality so that when a grouped issue is updated all issues grouped with it gets the same update
- Add functionality that makes it possible to set an internal priority of issues, not shown to customers
- Add bugs, minor features and wishes as support lines to issues
- All support lines should have their own inboxes so users can send messages to a specific support line
- All private messages should be possible to be marked as read or unread
- Add free text search functionality for issues
- Add paging to the listings of issues
- Additional history for issues, all actions performed are added to history
- Add declined as an issue status
- Change the listings of issues to not show responsible and severity.
- Add new column for amount of sub issues as a column of issue listings
- Prevent unused parts such as projects, tasks and releases from being displayed
- Add functionality to search for how long a customer has waited for response since their last comment on an issue
- When a customer adds a comment to an issue, it should automatically appear in its current support line mailbox
- Always display unread messages at the top of the listings and make it possible to sort

3.2 Second Phase

The second phase was to develop a light customer relationship management (CRM) system, which should be capable of integrating with Citybreak Support. This was a fresh start and the development started from scratch. A decision was made to use the ASP.NET MVC framework instead of Web Forms, since it seemed like an extensible framework and also a great opportunity to gain a new set of skills. The requirements for this phase were:

- It should be possible to CRUD (Create/Read/Update/Delete) list of tasks for project leader and sales department
- It should be possible to add more detailed information about each customer
- It should be possible to set responsible persons for every customer
- There should be a checklist of tasks concerning the project leader and sales department for each customer
- There should exist text fields which you can freely edit for each customer: business model, key factors, projects in the future, milestones, time plan and vision
- A list containing an overview of the agreement for every customer should be present
- There should exist a customer activity plan containing: name, start- and end date and comments
- There should exist a reminder function for upcoming, unfinished tasks that will alert you when you login to the system.
- It should be possible to CRUD person positions, roles within company and their attitude towards Citybreak
- It should be possible to add contact persons in each company. Each person should have a number of attributes and possibility to be member of groups
- Add functionality to search for and filter customer contacts
- Add functionality to export customer contact information to excel filtered by customer contact attributes and membership
- Add functionality for adding a quick comment on both customers and customer contacts
- Add functionality for grouping customers together
- Add functionality to CRUD customer groups
- Add functionality to CRUD customer business model, customer goals, customer vision, customer key factors and customer projects in a “word like” editor
- Add login functionality
- Add functionality for uploading files to each customer

4 System Design

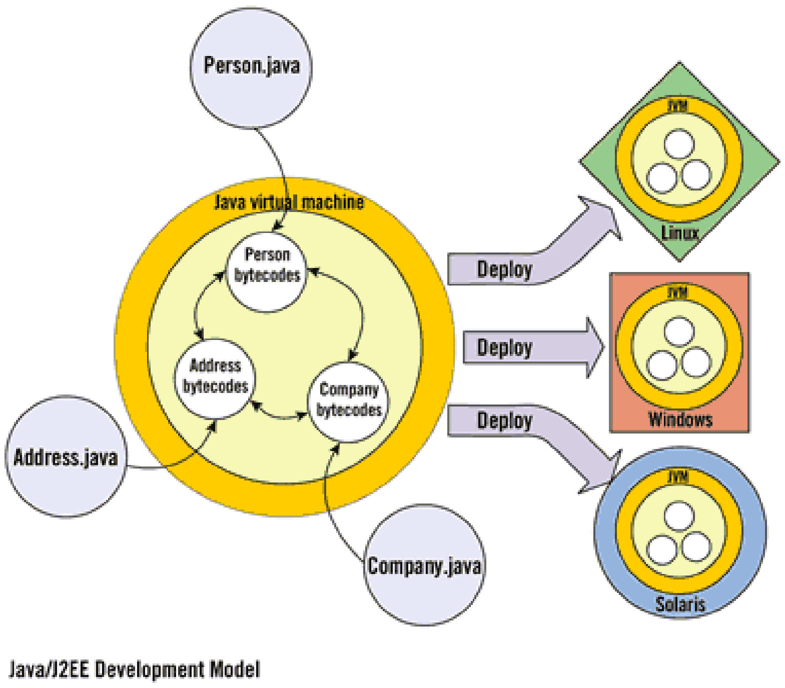
In its current state the system consists of a 3-tier ASP.NET web application. N-tier applications imply that they are organized into N major disjunctive layers where each layer refers to logical separation of responsibility. A layer communicates with other layers by providing or extracting information from other layers. Advantages of 3-tier architecture are that it is easier to maintain, while offering better code reuse and looser coupling compared to, for example, two-tier architecture. It also has more to offer in terms of flexibility and scalability in a client-server environment. The layers of the support system are:

- **The presentation layer** is the front-end which is responsible for presentation logic. In this case the layer largely consists of ASP.NET web forms and pages. This layer communicates with the business logic layer and transforms the results into front-end material.
- **Business logic layer** allows control of business logic by isolating it from the other layers of the application. This layer communicates the presentation layer and the data access logic layer, generally requesting data from the data access logic layer and the passing it on to the presentation layer.
- **Data access logic layer** provides access to the database. Does not contain business rules, it is merely a reusable interface to the database.

The support system uses .NET Remoting, a technique that will be explained in the next section, to gain scalability by allowing distributing the workload over several machines. Of course multiple tiers can reside on a single computer but by distributing the tiers to multiple machines the workload is better distributed.

4.1 .NET vs. Java EE

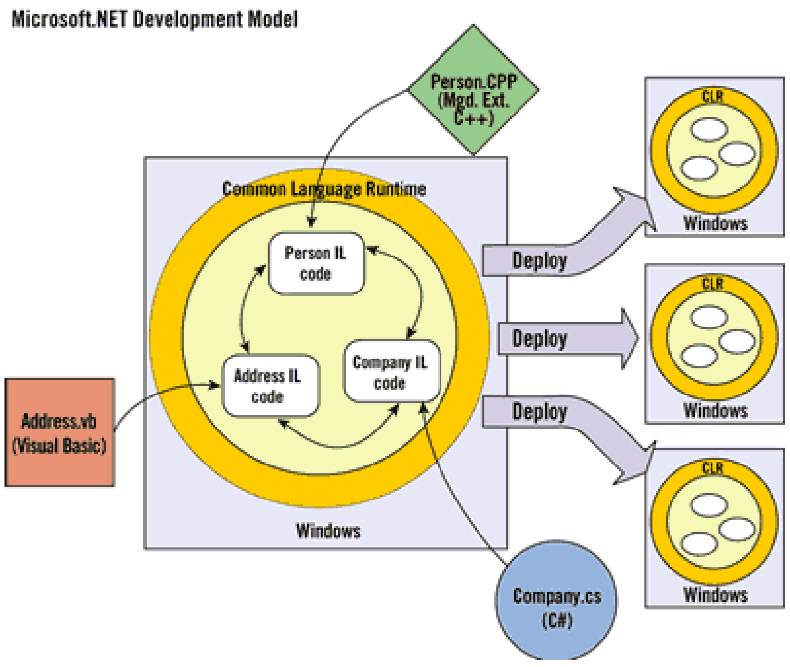
The Java 2 Platform, Enterprise Edition (J2EE) is an industry standard initiated by Sun Microsystems. The J2EE is a framework that is based on the core of the Java environment along with its Java Virtual Machine (JVM) and its core APIs. The programming model of Java compiles class descriptions in Java into platform-independent byte codes according to the JVM specification. These byte codes are then interpreted and executed by an implementation of the JVM targeted for a particular platform.



4-1 J2EE Development Model[3]

In comparison, Microsoft's .NET framework is based on its Common Language Runtime (CLR), which specifies the language-independent intermediate language (IL). First source code is translated into IL, which is analogous to Java byte code. The IL code then needs to be interpreted and translated into a native executable by the CLR, which is analogous to the JVM. The difference between the two is that .NET targeted code can be written in any language that supports the CLR component model. A specific compiler must be developed in order to compile the code into IL. [4]

[3] Jim Farley .NET vs. J2EE, http://www.ddj.com/java/184414710, 2009-05-26
[4] Jim Farley .NET vs. J2EE, http://www.ddj.com/java/184414710, 2009-05-26



4-2 .NET Development Model[5]

Java is designed to be a unified programming model that is platform independent; .NET on the other hand is platform dependent but language independent.

"Java is language-specific and platform-independent,

and dot-NET is language-independent and platform-specific."

However, this is an oversimplification since numerous J2EE implementations aren't entirely cross-platform.

The goal of both of these frameworks is to provide a lot of the groundwork required for enterprise applications such as security, interoperability, load balancing and transactions. Rather than writing all of the previously mentioned, the developer writes an application that runs within a container that provides those services.

Another important aspect when choosing framework is their interoperability with legacy systems. To achieve integration J2EE relies upon the J2EE Connector Architecture (JCA), a specification for plugging in resource adapters that understand how to communicate with existing systems. If such an adapter is missing it is possible to write one. These resource adapters are reusable in any container that supports the JCA.

[5] Jim Farley .NET vs. J2EE, http://www.ddj.com/java/184414710, 2009-05-26

.NET offers legacy integration through the Host Integration Server**.** COM Transaction Integrator (COM TI) can be used for collaborating transactions across systems. Another technique for integrating with legacy systems is Microsoft Message Queue (MSMQ).

One of the strengths of Java development is that it offers a variety of tools, products and applications. Providing more functionality in total than any one vendor could ever provide. This strength can also be considered to be a disadvantage, since you have to evaluate and choose with the risk of making the wrong choice. While the toolset of the Java community as a whole supersedes the functionality provided by Microsoft, one has to remember that they are not 100% interoperable.

However Visual Studio IDE is not free. Neither is SQL Server, Microsoft's operative systems (OS) or Internet Information Services (IIS). In contrast it is possible to get a Linux OS and a highly performing application server such as JBoss and a good IDE such as Eclipse or NetBeans for free. What favors Microsoft in this situation is the amount of integration they offer as a single vendor solution, minimizing the amount of low-level hacking required by developers.

Choosing the framework that is right for an organization depends upon a lot of factors. Microsoft offers a one package solution. If choice and flexibility of system components is important, J2EE becomes interesting. Another factor is the personal preference and skills of the developers of the organization.

If the system only needs to support Windows then .NET is a good alternative. However if one needs to support a number of different servers or if one wants to keep all options open in areas of tools, application servers, component vendors then J2EE is a good choice.

4.2 .NET Remoting

.NET Remoting is an application programming interface (API) for inter-process communication released in 2002 along with .NET Framework 1.0. It should be noted that it has been succeeded by the Windows Communication Foundation (WCF) introduced with .NET Framework 3.0.

Remoting allows applications to transform objects into remotable objects in order to make them available across boundaries such as application domains, processes or network. Objects can be made remotely available using two different methods, either Marshal By Reference or Marshal By Value. Marshalling is the process of transforming the memory representation of an object to a format suitable for storage or transmission.

What happens when an object is passed as Marshal By Reference by the remoting server is that the client will receive a proxy object. A proxy object is a transparent object that forwards operations performed on the proxy to the actual object, which resides in the memory of the Remoting server. By inheriting from the abstract class MarshalByRef in C# the object inheriting will be passed as Marshal By Reference and allow remote applications to instantiate the object.

The other alternative is called Marshal By Value, which is implemented by serialization. A serializable object knows how to transform its state into a byte array, which can be sent across boundaries along with its type name as a string. At the destination a new object can now be instantiated by using the type name and the state can be copied from the byte array – at this point two identical objects exists, one on each side of the boundary.

When comparing the two alternatives in terms of performance the usage scenario is the key. If the boundary consists of a network, each call to an object passed as Marshal By Reference will require a network connection to be made. If however the inter process communication takes place within the same machine the overhead of Marshal By Reference is negligible. If an object has a lot of properties or a lot of objects are to be accessed and the boundary is a network Marshal By Value is preferred.[6]

4.3 The Repository Pattern

The repository pattern is an architectural pattern based on the assessment that a complex domain model often benefits from a layer that isolates domain objects from details of the data storage access code. In such systems it can be worthwhile to build another layer of abstraction over the mapping layer where query construction code is concentrated. This becomes more important when there are a large number of domain classes or heavy querying. In these cases particularly adding this layer helps minimize duplicate query logic. [7]

A repository acts like a collection, except with more elaborate querying capability. Objects of the appropriate type are added and removed by the machinery behind the repository interface.
The client can communicate through a simple, intention-revealing interface, and ask for what it needs in terms of the model. The infrastructure behind the interface might be complex, but the interface remains simple and conceptually connected to the domain model. The repository pattern provides the illusion of an in-memory collection of all objects and provides methods that selects object based on certain criteria and return fully instantiated objects.

[6] Farhan Muhammad and Matt Milner, Real World ASP.NET Best Practices, Apress, 2003
[7] Martin Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley Professional, 2002

Conceptually, a repository encapsulates the set of objects in a data store and the operations performed over them, providing a more object-oriented view of the persistence layer. A repository also supports the objective of achieving a clean separation and one-way dependency between the domain and data mapping layers.[8]

Implementation will vary greatly, depending on the technology being used for persistence and the infrastructure you have. The idea is to hide all the inner workings from the client, although not from the developer of the client, so that client code will be the same whether the data is stored in an object database, a relational database, or simply held in memory.

Repositories reduce the amount of code needed to deal with querying and the clients of a repository never need to think in terms of, for example SQL, and can focus on handling of pure objects. A well-specified repository interface also enables the developer to swap the data source for domain objects, for example changing it from a database to a XML file. The repository will delegate to the appropriate infrastructure services to get the job done.

A great example is when the developer is interested in using simple in-memory data storage, a common situation when a developer wants to run a suite of unit tests entirely in memory for better performance. Tests that would otherwise take long time due to database access can run significantly faster by implementing the repository interface as in-memory object storage.

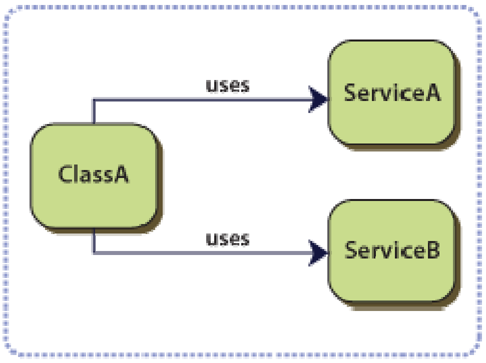
Repositories have many advantages, including the following: [9]

- They present clients with a simple model for obtaining persistent objects and managing their life cycle.
- They decouple application and domain design from persistence technology, multiple database strategies, or even multiple data sources.
- They communicate design decisions about object access.
- They allow easy substitution of a dummy implementation for use in testing, typically using an in-memory collection.

4.4 Dependency Injection

Dependency Injection (DI) is useful when you have layered dependencies whose types can be specified at compile time.

[8] Martin Fowler, Patterns of Enterprise Application Architecture, Addison Wesley, 2002
[9] Eric Evans, Domain-driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2004



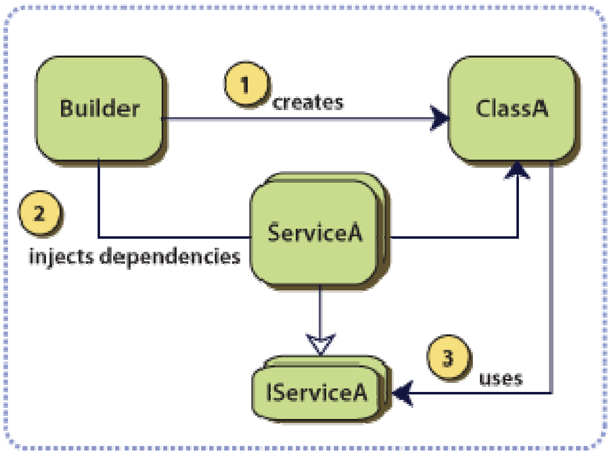
4-3 ClassA has dependencies on ServiceA and ServiceB[10]

In order to replace or change the dependencies of the above the developer has to modify every location where the concrete class is specified. The services become harder to test since they require dependencies that cannot be replaced without modifying the classes using the dependencies. The system above will also have repetitive code for creating, locating, and managing their dependencies.

DI allows the developer to decouple classes from their dependencies so that dependencies can be replaced or updated with minimal or no changes to the source code. It also enables the developer to write classes that depend on classes whose concrete implementation is not known at compile time. By decoupling the classes and programming by interfaces it increases testability. It also liberates classes from the responsibility of locating and managing the lifetime of dependencies. Managing the lifetime of dependencies allows the DI framework to limit how many instances or for how long a dependency should exist, for example, by managing a dependency as a Singleton or by the lifetime of the current thread.

When using DI the dependencies are not explicitly instantiated in the class using the dependency. Instead it uses a Builder/Object Factory object to obtain valid instances of the required dependencies and passes them to the objects requiring them upon creation or initialization.

[10] Microsoft Developer Network, Dependency Injection, http://msdn.microsoft.com/en-us/library/cc707845.aspx, 2009-05-25



4-4 Example on how a Dependency Injection builder class works[11]

When specifying the dependencies of classes it is common practice to use interfaces instead of concrete classes, this enables easy replacement of the dependency's concrete implementation without modifying your classes source code. Two common forms of dependency injection are constructor injection and setter injection.

Constructor injection works as follows, the developer expresses the dependencies as the object's constructor parameters, these are injected by the Object Factory/Builder object. With Setter Injection the dependencies are expressed as setter properties that the builder object uses to pass dependencies to an object during initialization. It is best practice to use constructor injection for required dependencies and setter injection for optional dependencies. [12]

Summarized benefits of Dependency Injection

- Decreased coupling
- Greater flexibility
- Separation of concerns
- Interface driven design
- Dependency management
- Manage lifetime of objects

[11] Microsoft Developer Network, Dependency Injection, http://msdn.microsoft.com/en-us/library/cc707845.aspx, 2009-05-25
[12] Microsoft Developer Network, Dependency Injection, http://msdn.microsoft.com/en-us/library/cc707845.aspx, 2009-05-25

4.5 ASP.NET MVC vs. ASP.NET Web Forms

ASP.NET MVC, MVC standing for Model-View-Controller, is an alternative approach to web application development compared to ASP.NET Web Forms. ASP.NET MVC is not to be considered as a replacement for ASP.NET Web Forms, it is even possible to use the two frameworks together.

The approach of ASP.NET Web Forms has many similarities with Windows Form Development. The Windows Form Development model is mostly event-driven, something that ASP.NET Web Forms shares. Besides being mainly event-driven, ASP.NET Web Forms also introduced two new terms: ViewState and Postback.

When developing web applications one has to remember that the foundation they reside on is the stateless HTTP protocol. ASP.NET Web Forms gives the developers the illusion of a stateful environment; one of these illusions was introduced in ASP.NET 1.0 called ViewState. ViewState helps the developer to maintain the state of objects and controls of a page through its life cycle by storing the state of the one control in a hidden form field that is rendered as a part of the HTML. ASP.NET Web Forms is restricted to one form element per page and this is where the hidden form field is attached. This form field generally has the name "_ViewState" and the value of this form field is the ViewState for the current page. By default ViewState is enabled in all included server controls shipped with ASP.NET Web Forms.

The ViewState is basically a StateBag that enables you to save and retrieve objects as key-value pairs. As objects are added, ASP.NET Web Forms serializes and persists them as an encrypted string, which is pushed to the client as the hidden form field _ViewState. The process of sending data back to the server is generally known as a page Postback and when it occurs the ViewState can be restored by using the hidden form field. If the ViewState grows to be very large one should consider alternative methods for storing the ViewState such as a database. The storing mechanism is however flexible and it is possible to derive your own storing mechanism for the ViewState. One thing to keep in mind is that certain types of data are optimized for use in the ViewState and not all types can be safely persisted. By default, the ViewState is sent to the client with a salted hash to prevent tampering. In practice this means that the ViewState data has been appended with a unique value before encoding.

Another variant of the ViewState is the ControlState, a place for storing critical server control information across Postbacks in order to ensure a working server control. The reason for this private ViewState per control, called ControlState, is that developers are able to disable or enable ViewState for all server controls. Usually when optimizing ASP.NET Web Forms pages developers tend to disable the ViewState for many controls whose state is not important.

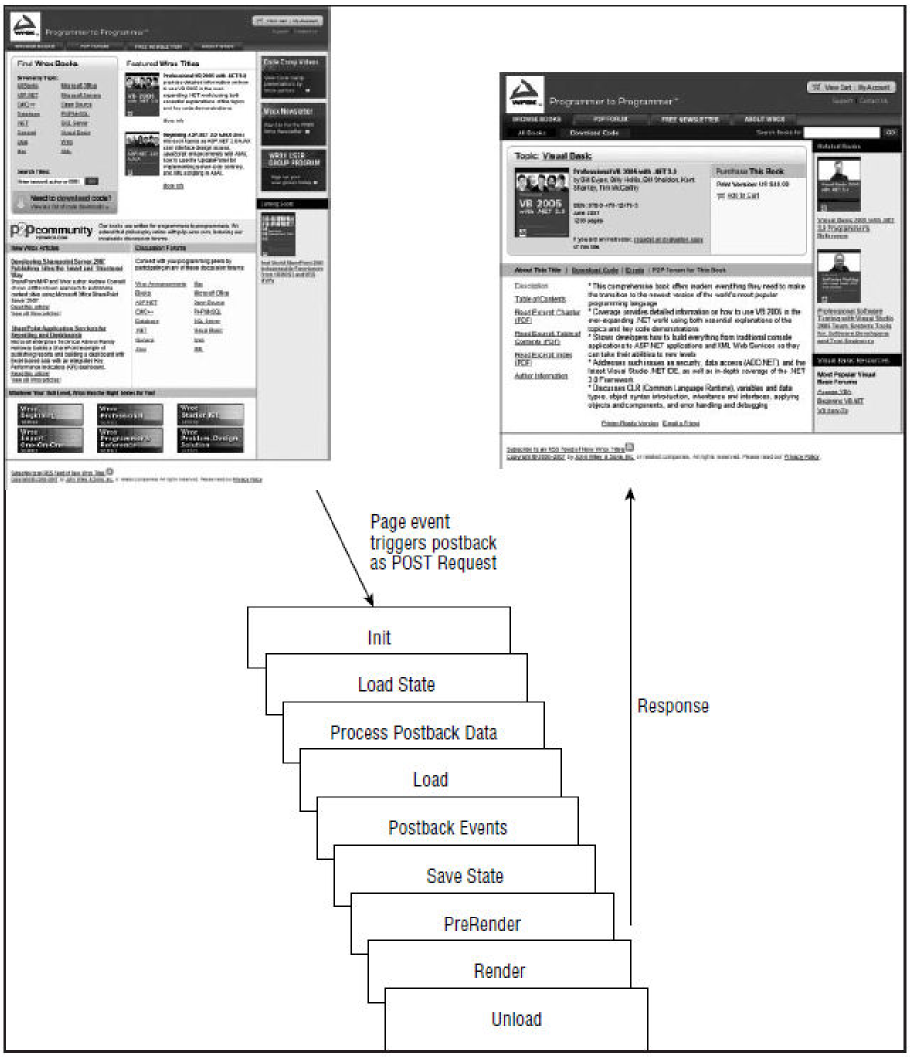
ViewState and ControlState, although not secure, is a good place to store small bits of data and state that don't quite belong in a cookie or the Session object. If the data that must be stored is relatively small and local to that specific instance of your page, ViewState is a much better solution than littering the Session object with lots of transient data.[13]

By the usage of ViewState and ControlState ASP.NET Web Forms offers a powerful way for server controls to emit their state to the client. This however a one-way communication since the server is pushing data to the client. ASP.NET Web Forms provides great support for Postback handling from ASP.NET Web Forms pages by enabling the developer to use controls that can raise events on the server-side. To initiate a Postback, client-side JavaScript is used and almost all server controls that are included by default support offer Postback handling.

These techniques help in some situations but they both try to make up for the stateless nature of the HTTP protocol. Both ViewState and Postbacks increase complexity of the web application and can affect performance. The heavy usage of server controls in ASP.NET Web Forms result in developers not having fine-grained control of the HTML that server controls generate. A problem is poorly written server controls that generate non-standard HTML. Another problem with ASP.NET Web Forms is the integration of JavaScript frameworks, such as jQuery, due to naming conventions of the rendered HTML and the Postback handling of ASP.NET Web Forms.

The lifecycle of an ASP.NET Web Forms page is complex; the view of an ASP.NET Web Forms page is also tightly coupled with its code-behind making it increasingly hard to unit test. Unit testing is an important aspect of modern software development, especially of agile methodologies.

[13] Bill Evjen, Scott Hanselman, Devin Rader, Professional ASP.NET 3.5 in C# and VB, Wiley Publishing, Inc, 2008



4-5 Lifecycle of an ASP Web Forms page as a Postback occurs[14]

ASP.NET MVC is a lighter framework than ASP.NET Web Forms and very extensible. The very core of the ASP.NET MVC framework is the Model-View-Controller pattern. Many of today's web applications retrieve data from data storage and display it to the user, afterwards the user changes the data and the data storage updates. Because the information flows between the data store and the user interface one might be inclined to tie these two together. However one thing to keep in mind is that the user interface tends to change much more frequently than the logic for data storage.

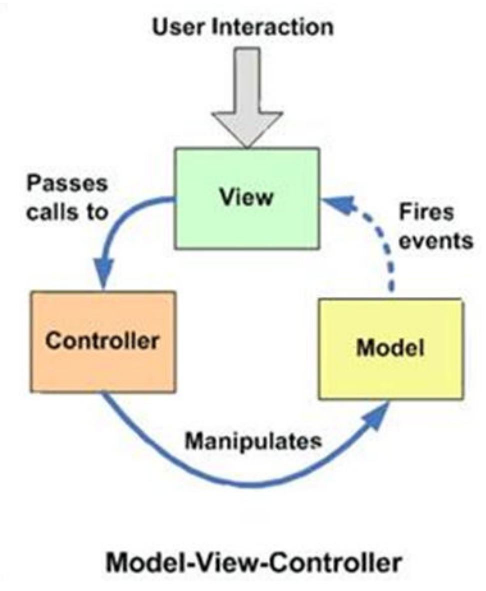
[14] Bill Evjen, Scott Hanselman, Devin Rader, Professional ASP.NET 3.5 in C# and VB, Wiley Publishing, Inc, 2008

Another problem with coupling the data and user interface pieces is that business applications tends to incorporate business logic that is far more complex then data transmission, resulting in very tightly coupled code.

A web application using MVC is structured into three individual parts, the Model, View and Controller, each with their own responsibility. By having three different layers, the chances to introduce errors when changing either of them are less likely. MVC separates the concern of storing, displaying, and updating data into three components that result in smaller and less coupled layers that can be tested individually. It also has the advantage of enabling web designers to design without any greater knowledge of the programming language used.

The MVC pattern separates the modeling of the domain, the presentation and the actions based on user input into three separate classes.[15]

- **Model** - The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).
- **View** - The view manages the display of information.
- **Controller** - The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate.

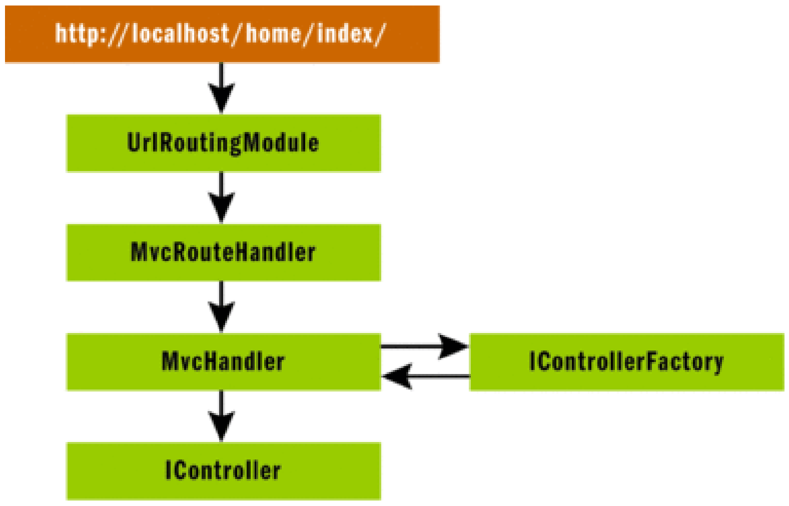


4-6 Depicts the structural relationship between the three objects[16]

[15] Microsoft Developer Network, Model-View-Controller, http://msdn.microsoft.com/en-us/library/ms978748.aspx, 2009-05-25
[16] Alex Homer, Design Patterns for ASP.NET Developers, http://www.devx.com/dotnet/Article/33695/1954, 2009-05-25

One important aspect is that both the view and the controller depend on the model. It should be noted that the model does not depend on either the view or the controller. Separation of the model allows for it to be built and tested independent of the presentation.

ASP.NET MVC replaces the event-driven model with controller actions and offers complete control of the rendered HTML. It also includes routing by default resulting in greater control and more intuitive URLs. There are no such things as server controls, ViewState or Postback handling in ASP.NET MVC.



4-7 Request flow for an ASP.NET MVC controller[17]

- The UrlRoutingModule class matches an HTTP request to a route in an ASP.NET application.[18]
- The MvcRouteHandler class creates an object that implements the IHttpHandler interface and gives it the request context.[19]
- The MvcHandler class first extracts the controller parameter as a string from the route data. Then it passes on the route data to the controller factory. When the ControllerFactory has returned a reference to a Controller, the MvcHandler will execute the corresponding action method on this Controller. When the Controller is done

[17] Scott Allen, Extreme ASP.NET: Dissecting an ASP.NET MVC Controller, http://msdn.microsoft.com/en-us/magazine/dd695917.aspx, 2009-05-25
[18] Microsoft Developer Network, UrlRoutingModule Class, http://msdn.microsoft.com/en-us/library/system.web.routing.urlroutingmodule.aspx, 2009-05-25
[19] Microsfot Developer Network, MvcRouteHandler Class, http://msdn.microsoft.com/en-us/library/system.web.mvc.mvcroutehandler.aspx, 2009-05-25

executing its action it will return an ActionResult, which could be of many different types.[20]

ASP.NET Web Forms Strengths

- Mature technology
- Designer support in Visual Studio.
- Abstracts the need to understand HTTP, HTML, CSS, and in some cases JavaScript.
- Easy state management with ViewState and Postback model.

ASP.NET Web Forms Weaknesses

- Display logic coupled with code, through code-behind files
- Harder to unit test application logic, because of the coupled code-behind files
- Added complexity of ViewState and Postback
- State management of controls leads to very large and often unnecessary page sizes

ASP.NET MVC Strengths

- Provides fine grained control over HTML
- Clear separation of concerns
- Provides application layer unit testing
- Can support multiple view engines
- Easy integration with JavaScript frameworks like jQuery or Yahoo UI frameworks
- Ability to map URLs logically and dynamically
- RESTful interfaces are used by default, helps with Search Engine Optimization
- Not bounded by the ViewState and Postback model
- Supports the entire core ASP.NET features, such as authentication, caching, membership, etc.
- Extensible

[20] Scott Allen, Extreame ASP.NET: Dissecting an ASP.NET MVC Controller, http://msdn.microsoft.com/en-us/magazine/dd695917.aspx, 2009-05-25

ASP.NET MVC Weaknesses

- Not event driven by the framework, so it may be more difficult for ASP.NET Web Forms developers to understand
- Requires the need to understand, at least at the basic level, HTTP, HTML, CSS, and JavaScript
- Third party library support is not as strong
- No direct upgrade path from Web Forms [21]

4.6 Language Integrated Query (LINQ)

LINQ was introduced with .NET 3.5 and offers strong typing and full object oriented support for query languages such as SQL with syntax specifically designed for query operations. With LINQ, querying becomes a first class concept whether the source is objects, LINQ to XML (XLINQ) or LINQ to SQL (DLINQ). Each flavor of LINQ uses the same basic query syntax to dramatically simplify the querying. LINQ allows the developer to perform complex query operations against any enumerable object that implements the IEnumerable interface. The purpose of LINQ was to address many of the shortcomings of querying collections of data by offering a more high level and abstract approach, it does so by not requiring the developer to specify exactly how the query is to be executed.

The simplest form of query is to only specify the type of which you want LINQ to return and leave the rest up to .NET and the compiler to determine exactly how the query will be run. LINQ also offers a dramatic improvement in readability and understandability of code. Another interesting feature of LINQ is Delayed Execution. In practice it means that LINQ will delay the actual execution of the query statements until a specific point in the code is accessed.

LINQ also makes it easier to include paging logic with the methods Skip and Take. Skip allows the developer to skip a defined number of records in the result set. The take method specifies the number of records to return. By combining Skip and Take paging becomes trivial.

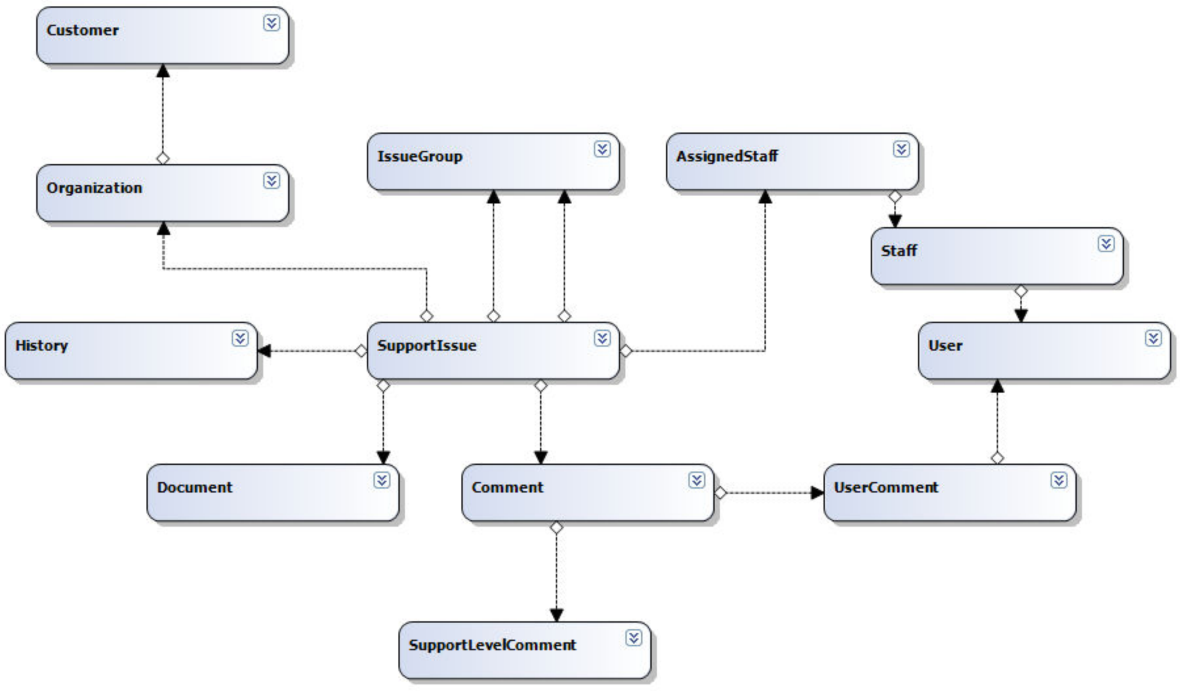
[21] Nick Berardi, Introducing the ASP.NET MVC (Part 2) - ASP.NET MVC vs. ASP.NET Web Forms, http://www.coderjournal.com/2008/12/introducing-aspnet-mvc-part-2-aspnet-mvc-vs-webforms/ 2009-05-25

5 Implementation

This section is intended to describe the choices made when developing a function or enhancement to Citybreak Support and the general idea behind the CRM system. In cases where one implementation was made but later changed, all implementations will be discussed, as well as the reason for change. Only nontrivial functionality containing some interesting decisions will be treated. After each description screenshots with the final result of the implementation will be provided.

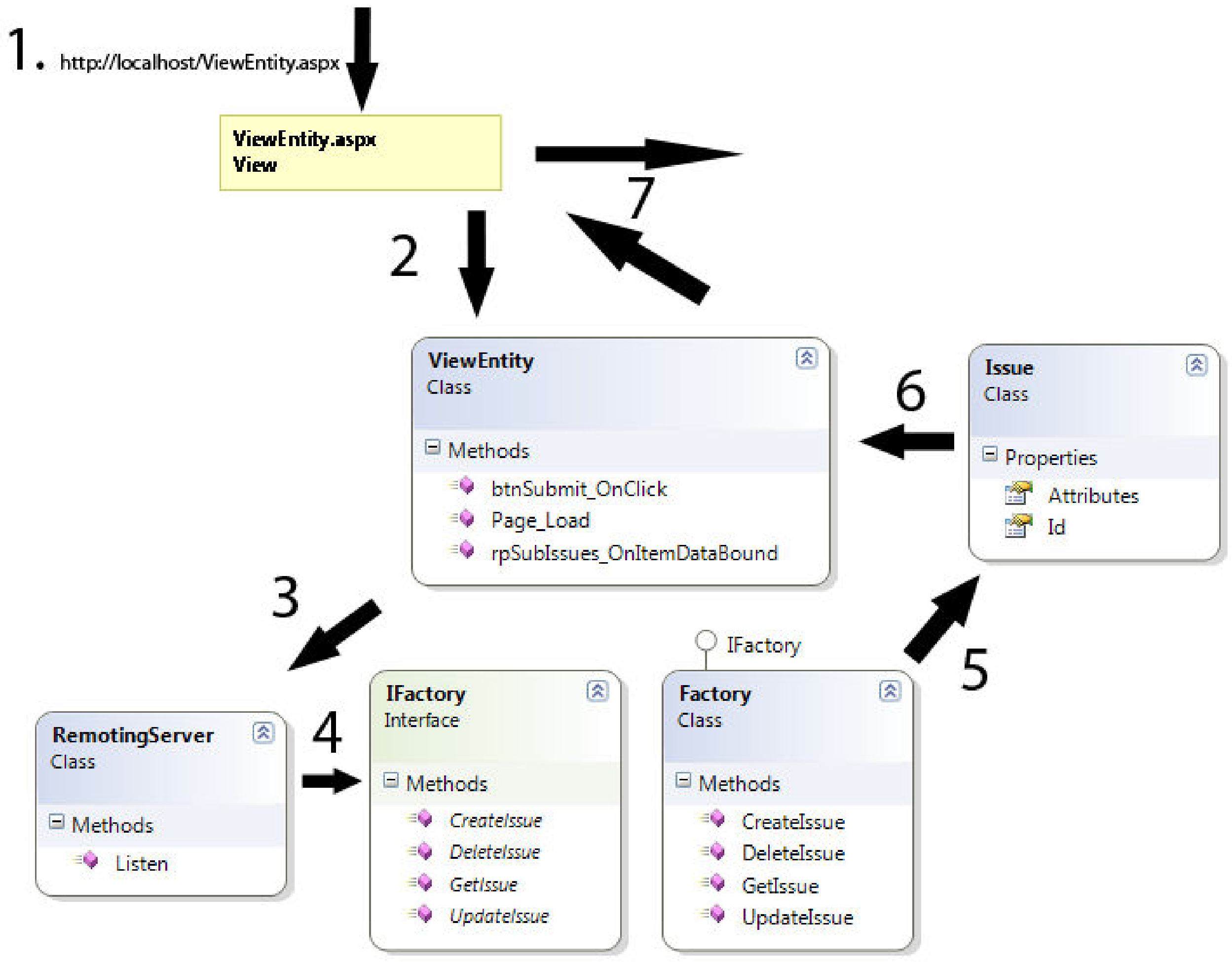
5.1 First Phase – New Functionality

5.1.1 Domain Model



5.1.2 Architecture

The architecture of Citybreak Support consists of ASP.NET Web Forms, .NET Remoting and an object factory. Below is a picture which describes an example of how the techniques are used to serve a user as they browse to a certain URL.



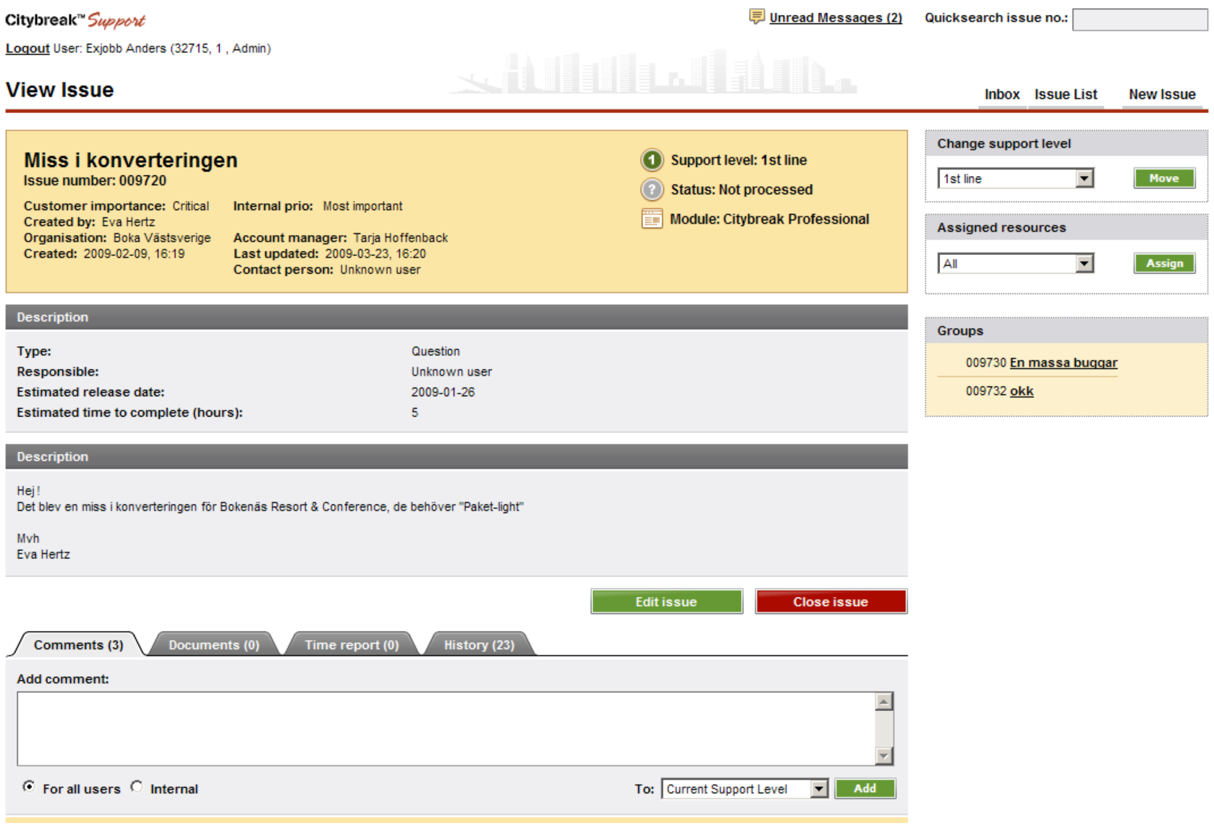
1. The user navigates to the URL, the view calls the code behind which is the presenter. Also server controls located in the view may trigger events connected to the code behind.
2. As the method Page_Load is invoked, ViewState and ControlState have been populated. If it is a Postback the ViewState and Control State can be used to bind the server controls.
3. In order to retrieve data from the database, a connection to the remoting server is made.
4. After connecting to the remoting server, the code behind requests an IFactory object which is passed as Marshal By Ref.
5. By usage of the IFactory, object model entities are retrievable by the methods of IFactory. These model entities are mostly passed as Marshal By Value.
6. Server controls are bound with the data retrieved from the IFactory if it is not a postback.

7. View is rendered and sent as response to the client.

5.1.3 Grouping Issues

The purpose of grouping issues was to make it easier to get an overview of the large amount of issues that already existed as well as new ones. There were two types of groups that were common and needed to be supported. Nearly identical issues created by different customers should be grouped when discovered. Issues about a certain area should be possible to group; for example, issues about translations into different languages should be possible to put in a translations group.

The first natural idea was to add functionality, which made it possible to create groups and then add issues to them. However, that solution would not support the grouping of nearly identical issues in a clear way. You would have to make up a lot of extra groups in order to be able to group such issues. The second idea was to mark an issue as a master issue and then connect other issues as sub issues. This solution seemed more natural and existing code for managing issues could be used. Therefore, the second solution was the one being implemented. It had some restrictions, based on a request from the company, so that only issues created by Visit Technology Group could become master issues in order to prevent customers from viewing other customer's issues. Also no master issues could be sub issues to prevent a hierarchy of groups.


Citybreak™ Support

Logout User: Exjobb Anders (32715, 1 , Admin)

Unread Messages (2) Quicksearch issue no.:

View Issue

Inbox Issue List New Issue

Miss i konverteringen
Issue number: 009720

Customer importance: Critical
Created by: Eva Hertz
Organisation: Boka Västsverige
Created: 2009-02-09, 16:19

Internal prio: Most important
Account manager: Tarja Hoffenback
Last updated: 2009-03-23, 16:20
Contact person: Unknown user

Support level: 1st line
Status: Not processed
Module: Citybreak Professional

Description

Type:	Question
Responsible:	Unknown user
Estimated release date:	2009-01-26
Estimated time to complete (hours):	5

Description

Hej!
Det blev en miss i konverteringen för Bokenäs Resort & Conference, de behöver "Paket-light"

Mvh
Eva Hertz

Add comment:

For all users Internal To: Current Support Level

Change support level: 1st line

Assigned resources: All

Groups

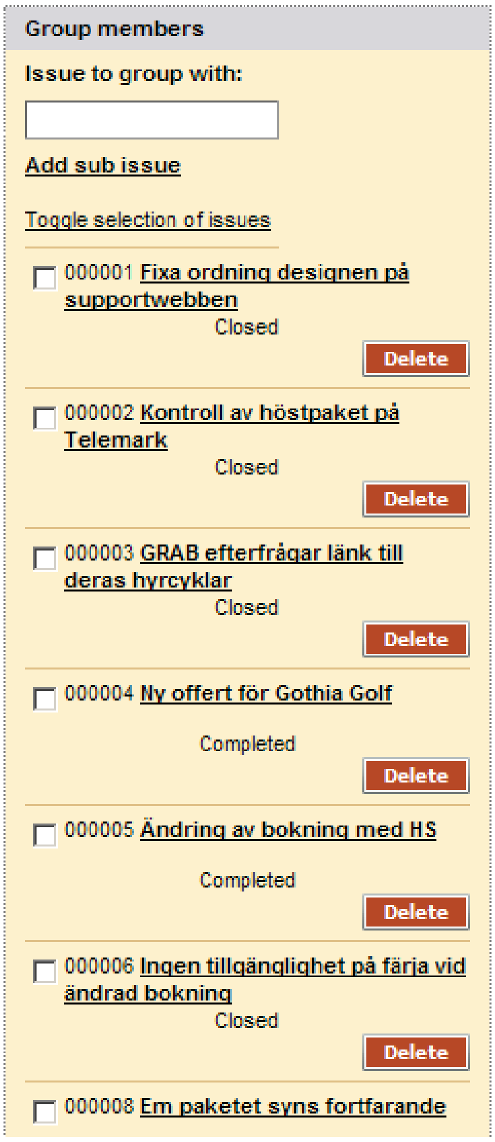
009730 En massa buggar
009732 okk

5-1 Screenshot from a sub issue showing group to the right

5.1.4 Updating Groups of Issues

The basic request from the company was that when you set certain statuses on a master issue all sub issues would be set to the chosen status as well. One of the main problems with Citybreak Support was that very much were hardcoded and difficult to change. Therefore, a decision was taken to make this functionality a bit more flexible and easier to change in case the company discovered that they wanted something more updated on the sub issues when set on a master issue.

Also one could imagine that only a selected set of the sub issues should receive the update made to the master issue. With these considerations in mind the implementation need to have checkboxes next to all sub issues and perform the update only on those that had their checkbox filled.


Group members

Issue to group with:

Add sub issue

Toggle selection of issues

000001 Fixa ordning designen på supportwebben
Closed
Delete

000002 Kontroll av höstpaket på Telemark
Closed
Delete

000003 GRAB efterfrågar länk till deras hyrcyklar
Closed
Delete

000004 Ny offert för Gothia Golf
Completed
Delete

000005 Ändring av bokning med HS
Completed
Delete

000006 Ingen tillgänglighet på färja vid ändrad bokning
Closed
Delete

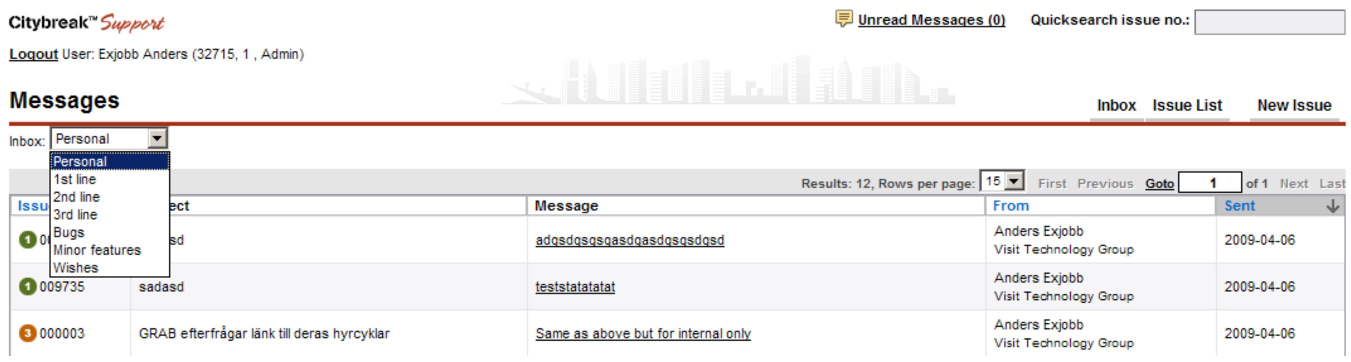
000008 Em paketet syns fortfarande

5-2 Screenshot showing the checkboxes next to sub issues

5.1.5 Line Messages/Inboxes

An additional requirement was that all lines should have an inbox and that all Visit users should have access to them. This was to prevent the customers from long response times in the case of someone on the staff being away from work for a longer period. The old solution with only having personal inboxes and lines not having any inboxes could result in long response times. Messages related to issues would now by default end up in the line inbox, which line it ends up in depends on the current line of the issue.

Before adding inboxes for lines, the old personal message page was changed to use the same paging as the new listing of issues did. After this a dropdown list was implemented and only visible for Visit users and made it possible for them to select which inbox they wanted to view.


Citybreak™ Support

Logout User: Exjobb Anders (32715, 1 , Admin)

Unread Messages (0) Quicksearch issue no.:

Messages

Inbox Issue List New Issue

Inbox: Personal

Personal
1st line
2nd line
3rd line
Bugs
Minor features
Wishes

Results: 12, Rows per page: 15 First Previous Goto 1 of 1 Next Last

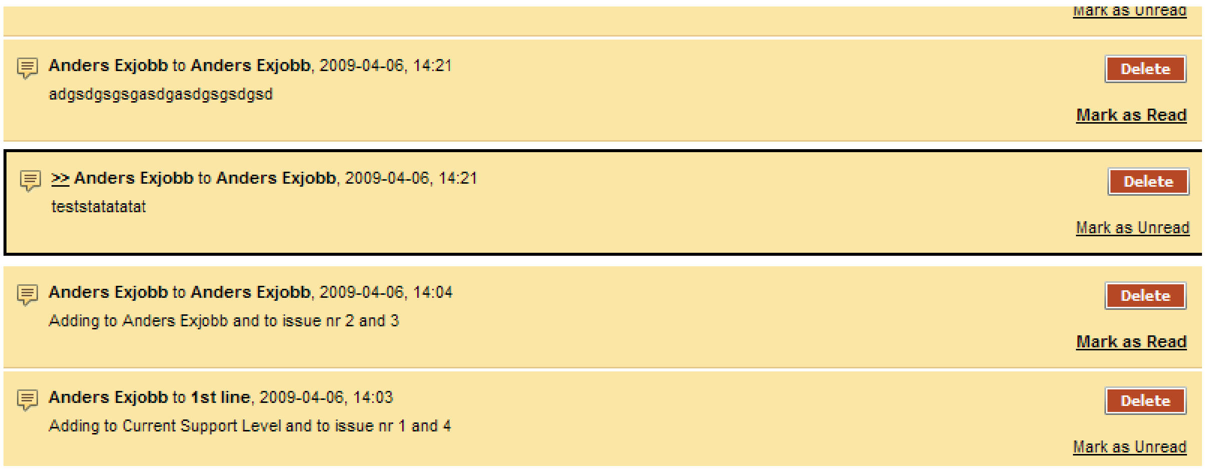
Issu	ect	Message	From	Sent
0	sd	adgsdgsgsgasdgasdgsgsdgsd	Anders Exjobb Visit Technology Group	2009-04-06
009735	sadasd	teststatatatat	Anders Exjobb Visit Technology Group	2009-04-06
000003	GRAB efterfrågar länk till deras hyrcyklar	Same as above but for internal only	Anders Exjobb Visit Technology Group	2009-04-06

5-3Screenshot on message page with the dropdown open

5.1.6 Marking Private Messages as Read or Unread

This requirement was based upon the huge differences in how time-consuming treating different messages could be. Some messages just needed a short reply while others could take hours to complete. When discovered that the time available wasn't enough to treat the message directly, one would wish for a function to mark the message as unread so that you could leave it for now but still have a reminder to treat the message later.

All messages in Citybreak Support are comments added to an issue. If a specific user were chosen when added the comment, it would appear as a message in that user's inbox. The typical usage of the mark as unread function would be to read the comment and then decide whether one have enough time or not to deal with it. To give as strong support as possible to this typical usage, a decision was made to put a link next to every comment allowing marking it as read or unread.


Mark as Unread

Anders Exjobb to Anders Exjobb, 2009-04-06, 14:21
adgsdgsgsgasdgasdgsgsdgsd
Delete
Mark as Read

>> Anders Exjobb to Anders Exjobb, 2009-04-06, 14:21
teststatatatat
Delete
Mark as Unread

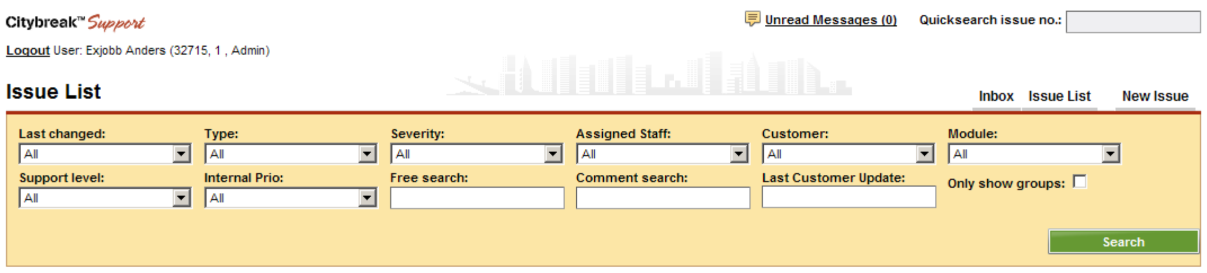
Anders Exjobb to Anders Exjobb, 2009-04-06, 14:04
Adding to Anders Exjobb and to issue nr 2 and 3
Delete
Mark as Read

Anders Exjobb to 1st line, 2009-04-06, 14:03
Adding to Current Support Level and to issue nr 1 and 4
Delete
Mark as Unread

5-4 Screenshot showing comments and mark as read/unread button

5.1.7 Search Issues by Text

More specific the possibility to search for a free text in comments, descriptions, subject and customer was requested. This should also be able to combine with the already existing filtering search for issues. The large amount of data made the search too slow if it had to collect all data from the database and then search through it. Therefore, the search had to be done directly in the SQL query. To do this the FULLTEXT[22] index search supported by SQL was used. For as simple usage as possible, the desire was to make one text field that searched all the requested areas. For performance reasons it turned out to be necessary to break out comment search to its own text field and make that search separately.


Citybreak™ Support
Logout User: Exjobb Anders (32715, 1 , Admin)
Unread Messages (0) Quicksearch issue no.:

Issue List
Inbox Issue List New Issue

Last changed: All	Type: All	Severity: All	Assigned Staff: All	Customer: All	Module: All
Support level: All	Internal Prio: All	Free search:	Comment search:	Last Customer Update:	Only show groups: ☐

Search

5-5 Screenshot of the filter options with new options added

[22] Microsoft Developer Network, Full-Text Search, http://msdn.microsoft.com/en-us/library/ms142571.aspx, 2009-05-25

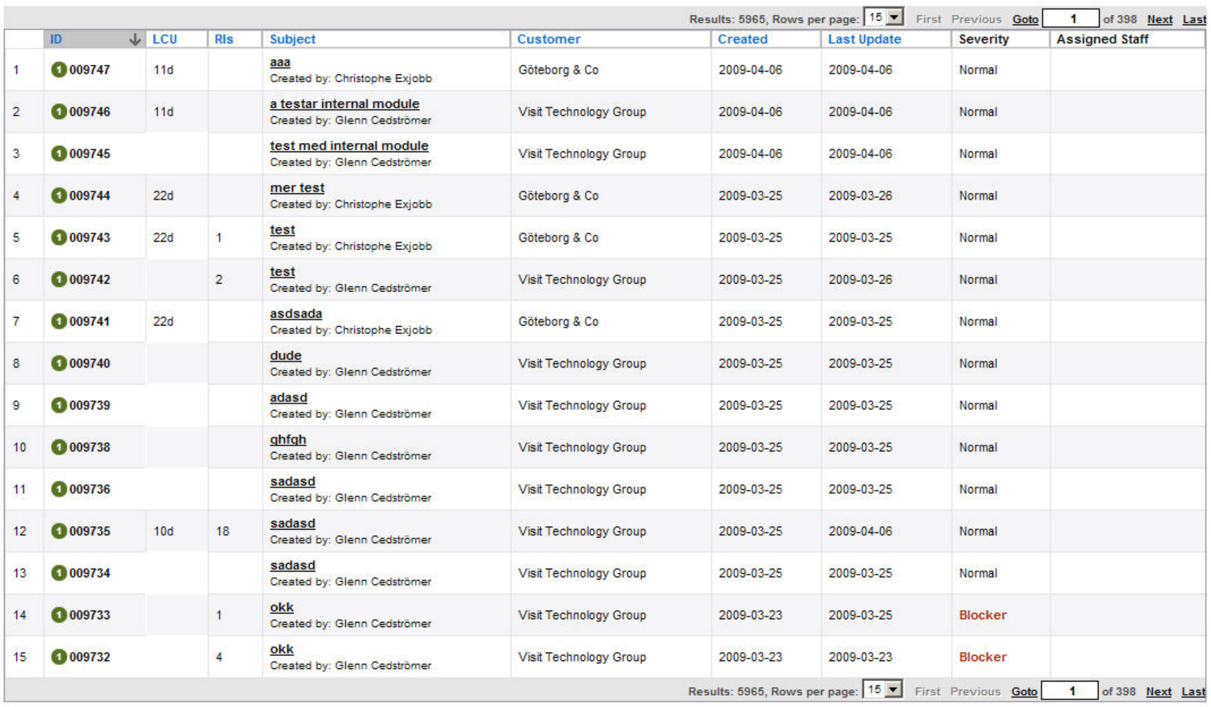
5.1.8 Paged Lists

This requirement turned out to be much more troublesome than expected. In addition to adding paging to the listings, it was essential to make them a lot faster. Loading the listing with all existing issues could take up to 45 seconds. A first guess was that generating thousands of ASP.NET server controls was the cause of the huge loading time. Therefore adding the paging would take care of that automatically.

The first implementation made, in order to solve the paging problem, was using the ASP.NET control GridView[23] that had built-in support for both paging and sorting among others. This component was during the development replaced with the newer ListView[24] component which had all the same advantages as GridView. To take advantage of the built-in paging, these components required the complete set of data at once. Unfortunately, that data consisted over almost ten thousand issues with all information connected to them. The size of those issues was almost 3 MB, which made the listing very slow to load.

To speed up the listing it was necessary to ask the database to only give back those rows that matched the current paging selected. This made the built-in functionality of both GridView and ListView useless and those components could be replaced by a regular Repeater[25] control. The paging needed to be handled manually and after a lot of effort and help from our supervisor we created our own user control to handle the paging. This final solution to the listing improved the speed tremendously.

[23] Microsoft Developer Network, GridView Class, http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx, 2009-05-25
[24] Microsoft Developer Network, ListView Web Server Control Overview, http://msdn.microsoft.com/en-us/library/bb398790.aspx, 2009-05-25
[25] W3Schools, ASP.NET - The Repeater Control, http://www.w3schools.com/ASPNET/aspnet_repeater.asp, 2009-05-25



Results: 5965, Rows per page: 15 First Previous Goto 1 of 398 Next Last

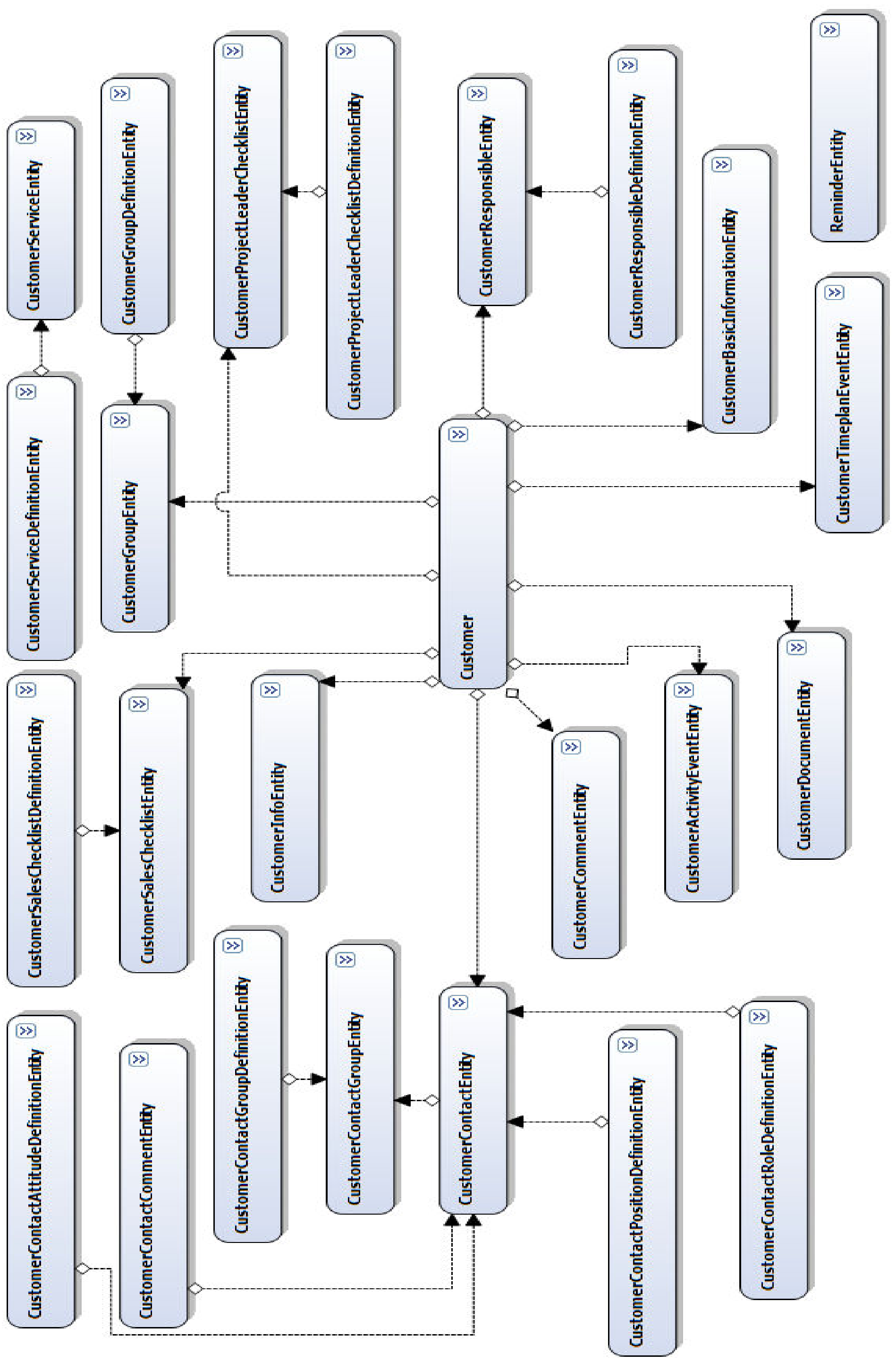
	ID	LCU	RIs	Subject	Customer	Created	Last Update	Severity	Assigned Staff
1	009747	11d		aaa Created by: Christophe Exjobb	Göteborg & Co	2009-04-06	2009-04-06	Normal	
2	009746	11d		a testar internal module Created by: Glenn Cedströmer	Visit Technology Group	2009-04-06	2009-04-06	Normal	
3	009745			test med internal module Created by: Glenn Cedströmer	Visit Technology Group	2009-04-06	2009-04-06	Normal	
4	009744	22d		mer test Created by: Christophe Exjobb	Göteborg & Co	2009-03-25	2009-03-26	Normal	
5	009743	22d	1	test Created by: Christophe Exjobb	Göteborg & Co	2009-03-25	2009-03-25	Normal	
6	009742		2	test Created by: Glenn Cedströmer	Visit Technology Group	2009-03-25	2009-03-26	Normal	
7	009741	22d		asdsada Created by: Christophe Exjobb	Göteborg & Co	2009-03-25	2009-03-25	Normal	
8	009740			dude Created by: Glenn Cedströmer	Visit Technology Group	2009-03-25	2009-03-25	Normal	
9	009739			adasd Created by: Glenn Cedströmer	Visit Technology Group	2009-03-25	2009-03-25	Normal	
10	009738			ghfgh Created by: Glenn Cedströmer	Visit Technology Group	2009-03-25	2009-03-25	Normal	
11	009736			sadasd Created by: Glenn Cedströmer	Visit Technology Group	2009-03-25	2009-03-25	Normal	
12	009735	10d	18	sadasd Created by: Glenn Cedströmer	Visit Technology Group	2009-03-25	2009-04-06	Normal	
13	009734			sadasd Created by: Glenn Cedströmer	Visit Technology Group	2009-03-25	2009-03-25	Normal	
14	009733		1	okk Created by: Glenn Cedströmer	Visit Technology Group	2009-03-23	2009-03-25	Blocker	
15	009732		4	okk Created by: Glenn Cedströmer	Visit Technology Group	2009-03-23	2009-03-23	Blocker	

Results: 5965, Rows per page: 15 First Previous Goto 1 of 398 Next Last

5-6 Screenshot of the new listing with paging at both top and bottom

5.2 Second Phase - CRM Development

5.2.1 Domain Model

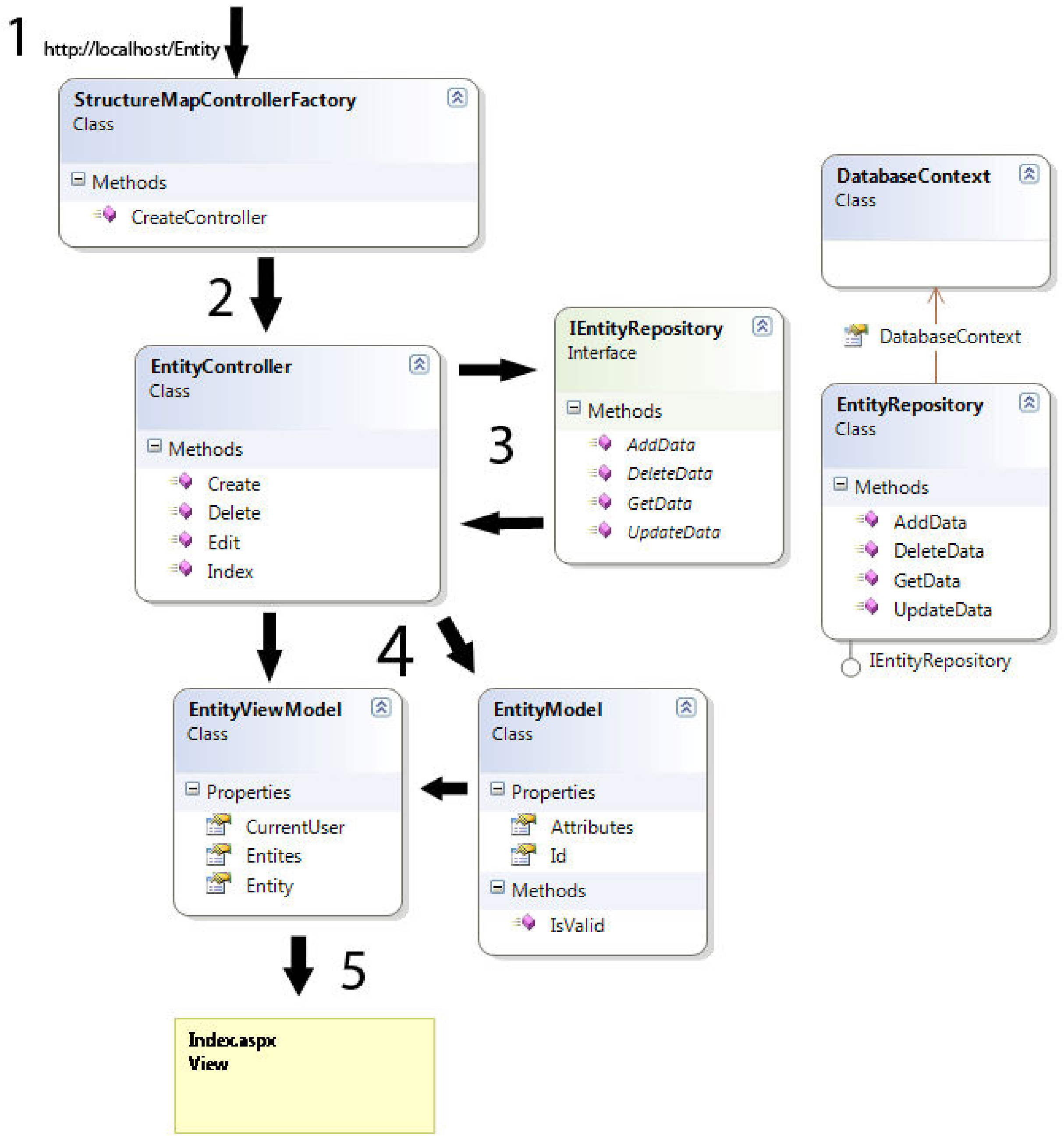

CustomerServiceEntity
CustomerGroupDefinitionEntity
CustomerProjectLeaderChecklistEntity
CustomerProjectLeaderChecklistDefinitionEntity
CustomerResponsibleEntity
CustomerResponsibleDefinitionEntity
CustomerBasicInformationEntity
ReminderEntity
CustomerServiceDefinitionEntity
CustomerGroupEntity
Customer
CustomerTimeplanEventEntity
CustomerSalesChecklistDefinitionEntity
CustomerSalesChecklistEntity
CustomerInfoEntity
CustomerCommentEntity
CustomerActivityEventEntity
CustomerDocumentEntity
CustomerContactAttitudeDefinitionEntity
CustomerContactCommentEntity
CustomerContactGroupDefinitionEntity
CustomerContactGroupEntity
CustomerContactEntity
CustomerContactPositionDefinitionEntity
CustomerContactRoleDefinitionEntity

5-7 Domain Model for the CRM system

All objects in the domain model has the suffix Entity but this suffix will be left out when mentioning them. As shown in figure 5-7, Customer is the central part in the CRM system, directly or indirectly connected to all other model entities with the exception of Reminder. The reason for Reminder not being connected to any other entity is due to the fact that those entities belong to another domain model, which is mentioned in phase one.

5.2.2 Architecture

Aim of this section is to describe the general architecture of the CRM system; how dependency injection, repository pattern and DLINQ are woven together into the architecture. It should be noted that the picture below has omitted the classes that handle route handling in ASP.NET MVC previously described in the chapter ASP.NET MVC vs. ASP.NET Web Forms.



5-8 Architecture of the CRM system

1. As the user enters the URL into their browser the ASP.NET MVC route handler figures out the name of which controller should be instantiated. The name is then passed to the StructureMapControllerFactory, responsible for instantiating controllers. It does so by looking for a class with the controller name and with the suffix “Controller” which implements the IController interface.

The reason for the name StructureMapControllerFactory is that it uses a DI framework which goes by the name StructureMap. After finding the appropriate controller class, the DI framework is ordered to find the appropriate constructor of the controller that it has been instructed how to construct. In the case of the CRM system this is done at application startup, at this point the DI framework is instructed to, for example, always instantiate a requested object type of IEntityRepository with the concrete class EntityRepository. It is also instructed to manage the lifetime of the DatabaseContext by the HTTP request context, or if there is no HTTP request context manage it by the lifetime of the local thread. In this example it is asked to construct an EntityController, which constructor takes an IEntityRepository. When the constructor has been located, the DI framework realizes that IEntityRepository is represented by the concrete class EntityRepository, which constructor takes a DatabaseContext as parameter. If a controller uses multiple repositories the DI framework will ensure that only one DatabaseContext will exist per HTTP request context[26].

2. After the EntityController has been instantiated, the appropriate action method, specified in the route data, will be invoked.
3. At this point new entities can be created, updated or retrieved through the EntityRepository. The controller validates the entities by invoking IsValid of an entity to ensure that only valid entities are persisted in the repository.
4. Before rendering the View, the controller populates a view model used by the View to render data retrieved from the repository.
5. View is rendered and is sent as a response to the client by the server.

5.2.3 Overview

The CRM system is based around the customers but also contains some parts that are not strongly connected to a customer. Each part not strongly connected to a customer is represented with its own tab when the user is logged into the system. Above the tabs is a quick search for customers, placed to always ensure quick access to a customer's profile, since they play the central part of the system. The other three tabs are: Reminders, Contact search and Control Panel.

5.2.4 Structure

There is lot of information about customers that is to be stored in their profile, which made it impossible to display everything on the same page. Links on the top of every customer page to different sections of the profile are used to navigate through different customer information. This

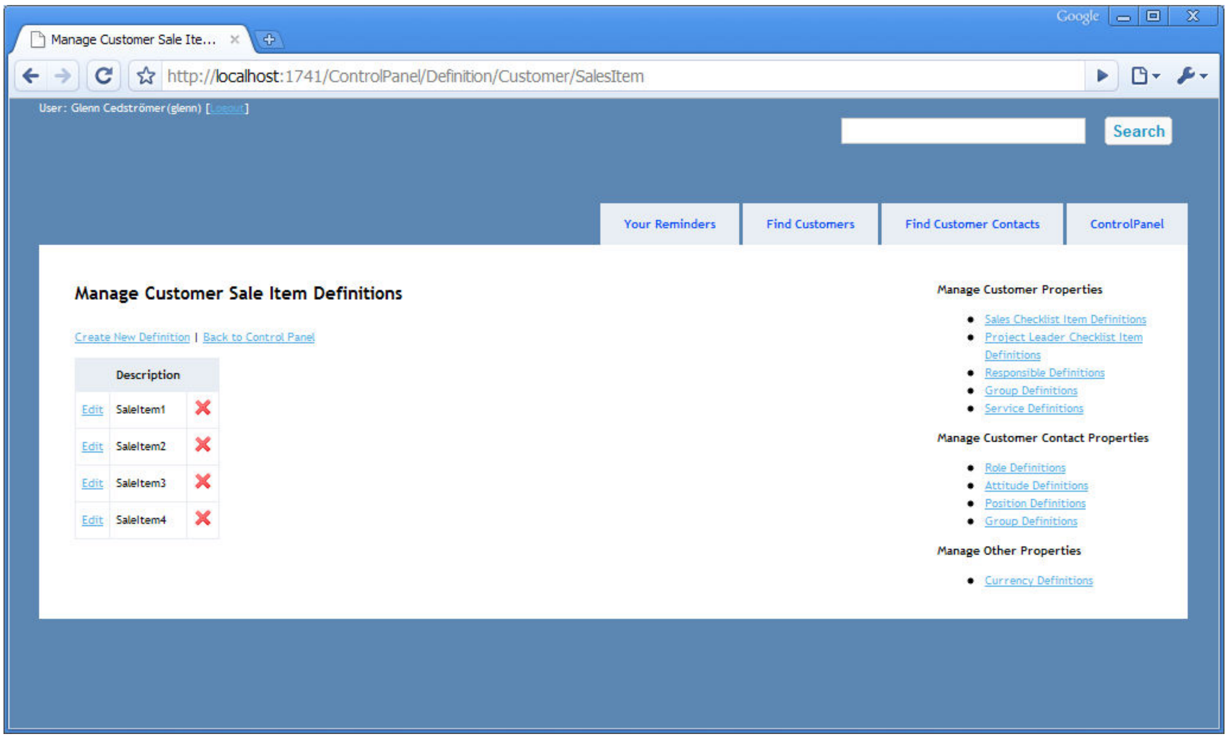
[26] Jeremy D Miller, StructureMap, http://structuremap.sourceforge.net, 2009-05-25

layout is based on discussions with the company where an agreement that this was the best way to display the information was reached. The first page you come to in every section is a view of the information contained in that section.

The first page is just for viewing the information. A link is provided on the top right corner to an edit page where all the customer details are editable. This was chosen in order to have a cleaner presentation view. On the edit page, one of the benefits with .NET MVC compared to .NET Web Forms is used namely the ability to have several forms on one page. Each separate piece is contained in its own form and submitted separately.

5.2.5 Control Panel

In order to allow for easy creation, removal and updates of information in the system, all definition items were requested to possess the possibility to be CRUD from inside the system. For this purpose a control panel was created and accessible for all administrator users. The control panel contains all the list items that have an effect on every customer and customer contact in the system. Future plans are to let this panel contain all actions that affects all customers and customer contacts.



Manage Customer Sale Ite...
http://localhost:1741/ControlPanel/Definition/Customer/SalesItem
User: Glenn Cedströmer (glenn) [Logout]
Search
Your Reminders
Find Customers
Find Customer Contacts
ControlPanel

Manage Customer Sale Item Definitions

Create New Definition | Back to Control Panel

	Description	
Edit	SaleItem1	❌
Edit	SaleItem2	❌
Edit	SaleItem3	❌
Edit	SaleItem4	❌

Manage Customer Properties

- Sales Checklist Item Definitions
- Project Leader Checklist Item Definitions
- Responsible Definitions
- Group Definitions
- Service Definitions

Manage Customer Contact Properties

- Role Definitions
- Attitude Definitions
- Position Definitions
- Group Definitions

Manage Other Properties

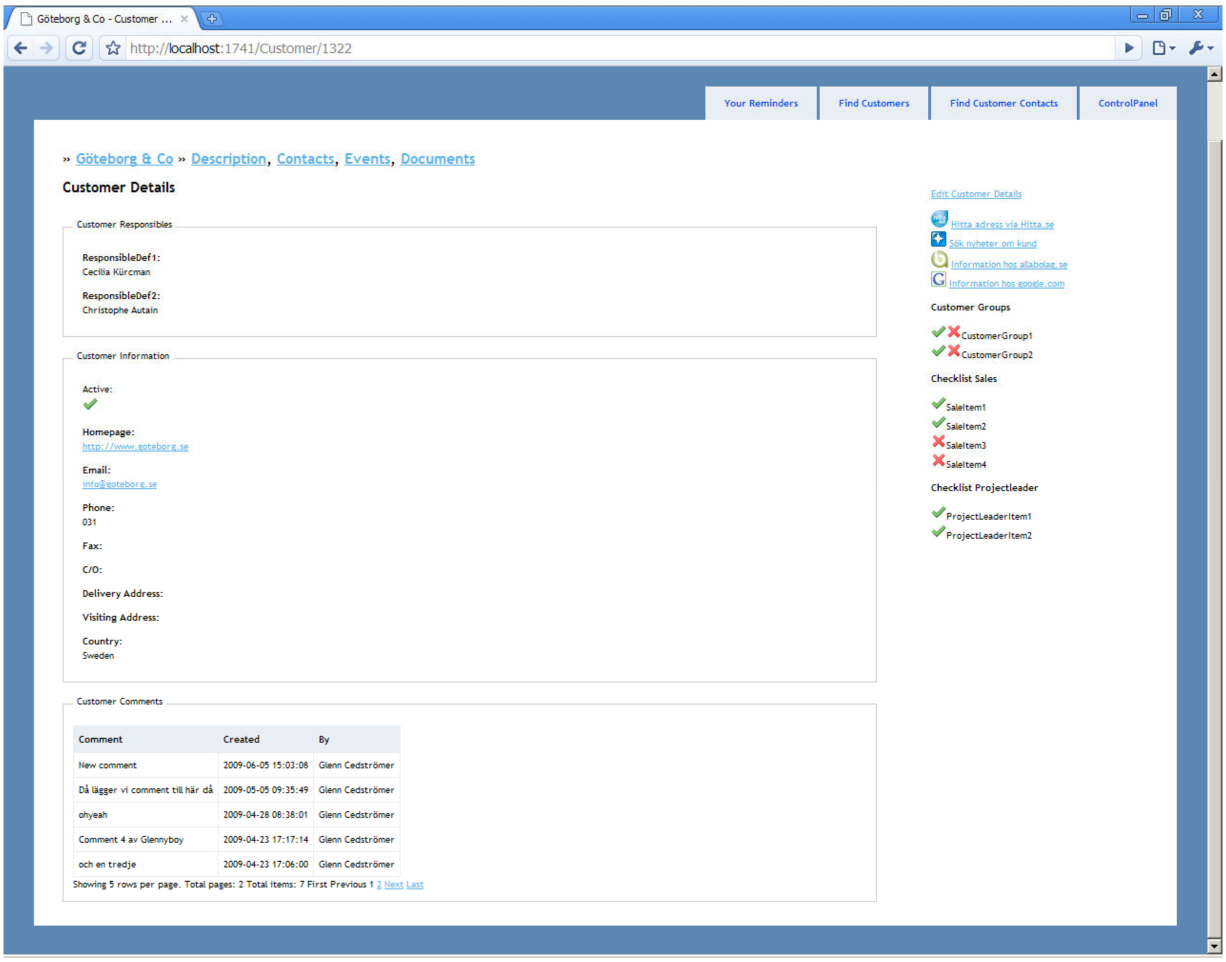
- Currency Definitions

5-9 Controlpanel managing customer sale item definitions

5.2.6 Customer Details

There were some discussions about which information was most important to see directly when viewing a customer's details. After a few meetings with the company a decision was taken to put the following information on the first page:

- **Customer responsible** – The items to choose from are dynamically configured as mentioned earlier and one person from visit staff is assigned as responsible for each task.
- **Basic customer information** – Basic information such as active customer, customer mail, customer homepage, phone etc.
- **Checklist Sales** – Checklist of sale items
- **Checklist project leader** – Checklist of project leader items
- **Customer groups** – Checklist of all customers groups
- **Customer comments** – Display 5 latest comments added to this customer with corresponding user and date.
- **Customer quick links** – Links to www.allabolag.se, www.hitta.se, www.google.se and www.hitta.se for the corresponding customer



Göteborg & Co - Customer ...
http://localhost:1741/Customer/1322
Your Reminders
Find Customers
Find Customer Contacts
ControlPanel

» Göteborg & Co » Description, Contacts, Events, Documents

Customer Details

Customer Responsibles

ResponsibleDef1:
Cecilia Kürcman

ResponsibleDef2:
Christophe Autain

Customer Information

Active:
✓

Homepage:
http://www.goteborg.se

Email:
info@goteborg.se

Phone:
031

Fax:

C/O:

Delivery Address:

Visiting Address:

Country:
Sweden

Customer Comments

Comment	Created	By
New comment	2009-06-05 15:03:08	Glenn Cedströmer
Då lägger vi comment till här då	2009-05-05 09:35:49	Glenn Cedströmer
ohyeah	2009-04-28 08:38:01	Glenn Cedströmer
Comment 4 av Glennyboy	2009-04-23 17:17:14	Glenn Cedströmer
och en tredje	2009-04-23 17:06:00	Glenn Cedströmer

Showing 5 rows per page. Total pages: 2 Total items: 7 First Previous 1 2 Next Last

Edit Customer Details

Hitta adress via Hitta.se
Sök nyheter om kund
Information hos allabolag.se
Information hos google.com

Customer Groups

✓ ✗ CustomerGroup1
✓ ✗ CustomerGroup2

Checklist Sales

✓ SaleItem1
✓ SaleItem2
✗ SaleItem3
✗ SaleItem4

Checklist Projectleader

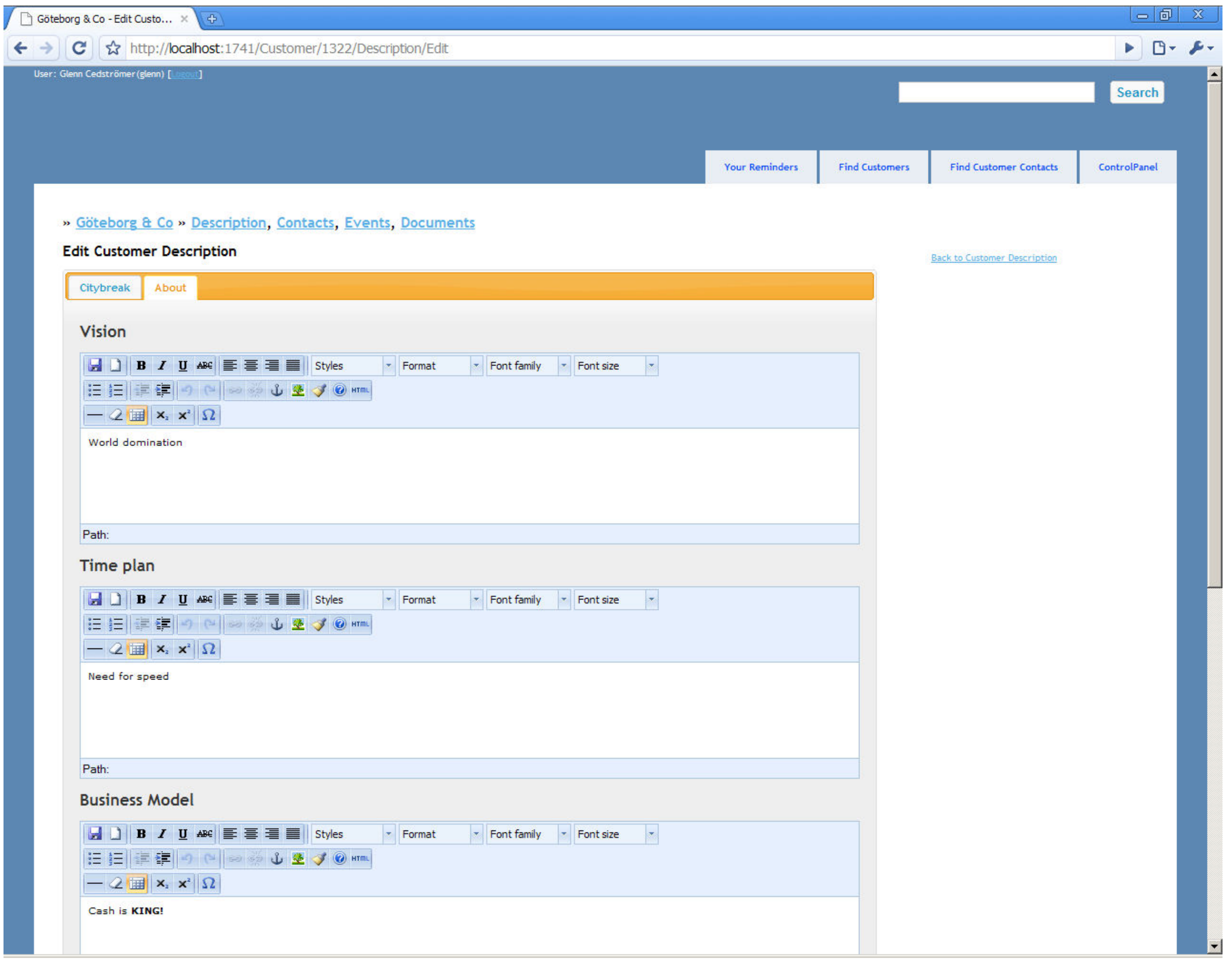
✓ ProjectLeaderItem1
✓ ProjectLeaderItem2

5-10 Customer details

5.2.7 Customer Description

This section of the customer profile is containing the requested free text functionality for business model, key factors, projects in the future, time plan and customers vision. The company had requested it to be similar to Microsoft Word when editing. Therefore a decision was taken to use Tiny Moxiecode Content Editor (TinyMCE). TinyMCE is a free text editor developed by Moxiecode Systems AB[27].

[27] Moxiecode Systems, TinyMCE, http://tinymce.moxiecode.com/, 2009-05-25


Göteborg & Co - Edit Custo...

http://localhost:1741/Customer/1322/Description/Edit

User: Glenn Cedströmer (glenn) [Logout]

Search

Your Reminders | Find Customers | Find Customer Contacts | ControlPanel

» Göteborg & Co » Description, Contacts, Events, Documents

Edit Customer Description

Back to Customer Description

Citybreak | About

Vision

World domination

Path:

Time plan

Need for speed

Path:

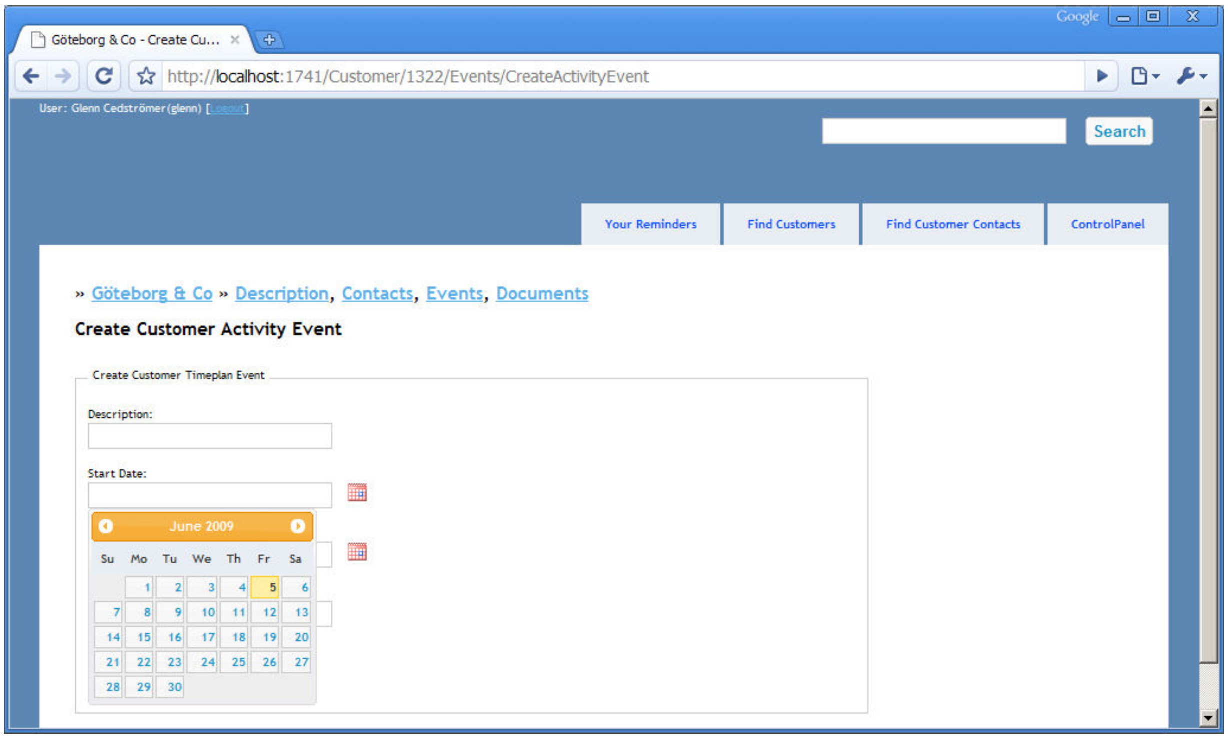
Business Model

Cash is **KING!**

5-11 Customer description

5.2.8 Customer Events

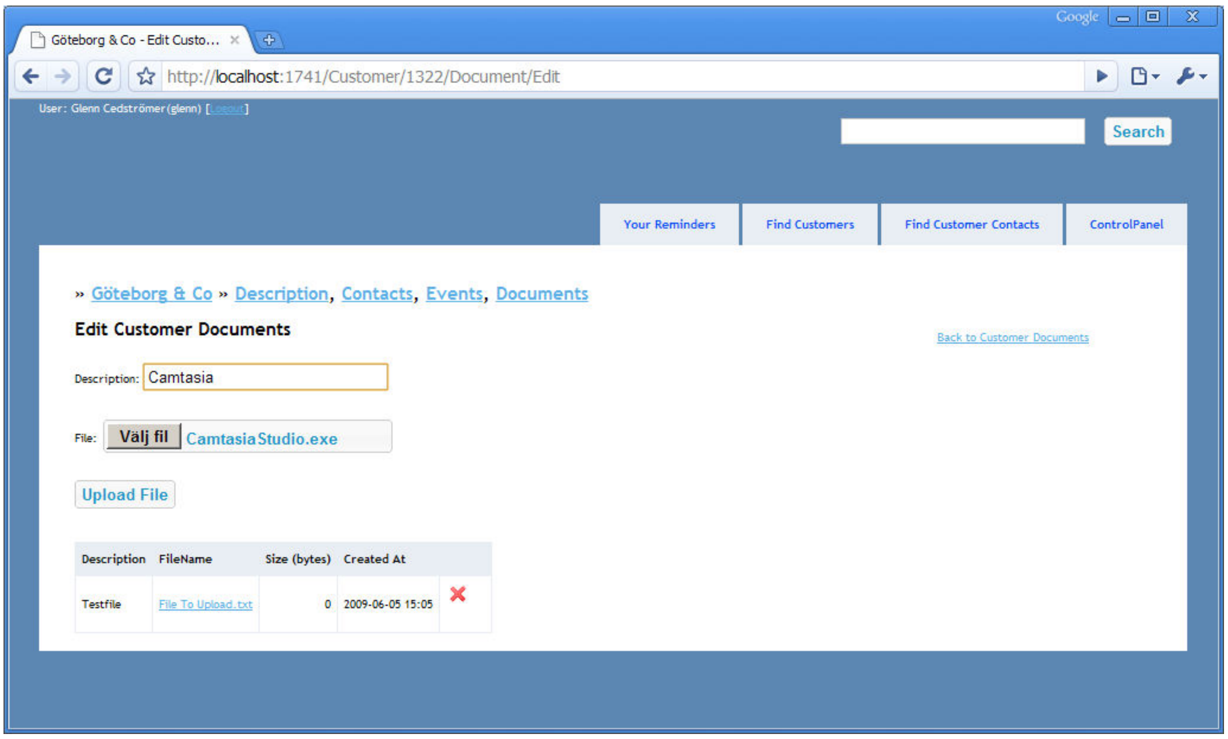
At first there were two types of customer events, customer time plan events and customer activity events. Later customer time plan changed to be a free text instead and only activity events remained. Temporarily before a better display is worked out, all ongoing activity events are displayed in green and all future are displayed in red. Activity events in the past are not displayed at all.


Göteborg & Co - Create Cu...
http://localhost:1741/Customer/1322/Events/CreateActivityEvent
User: Glenn Cedströmer(glenn) [Logout]
Search
Your Reminders
Find Customers
Find Customer Contacts
ControlPanel
» Göteborg & Co » Description, Contacts, Events, Documents
Create Customer Activity Event
Create Customer Timeplan Event
Description:
Start Date:
June 2009
Su Mo Tu We Th Fr Sa

5-12 Customer events

5.2.9 Customer Documents

Here is where the user has an opportunity to upload files connected to a customer. Along with the filename, size and the time the file was uploaded is saved and displayed.


Göteborg & Co - Edit Custo...
http://localhost:1741/Customer/1322/Document/Edit
Search
Your Reminders
Find Customers
Find Customer Contacts
ControlPanel
Back to Customer Documents
Välj fil
Upload File

User: Glenn Cedströmer (glenn) [Logout]

» Göteborg & Co » Description, Contacts, Events, Documents

Edit Customer Documents

Description: Camtasia

File: CamtasiaStudio.exe

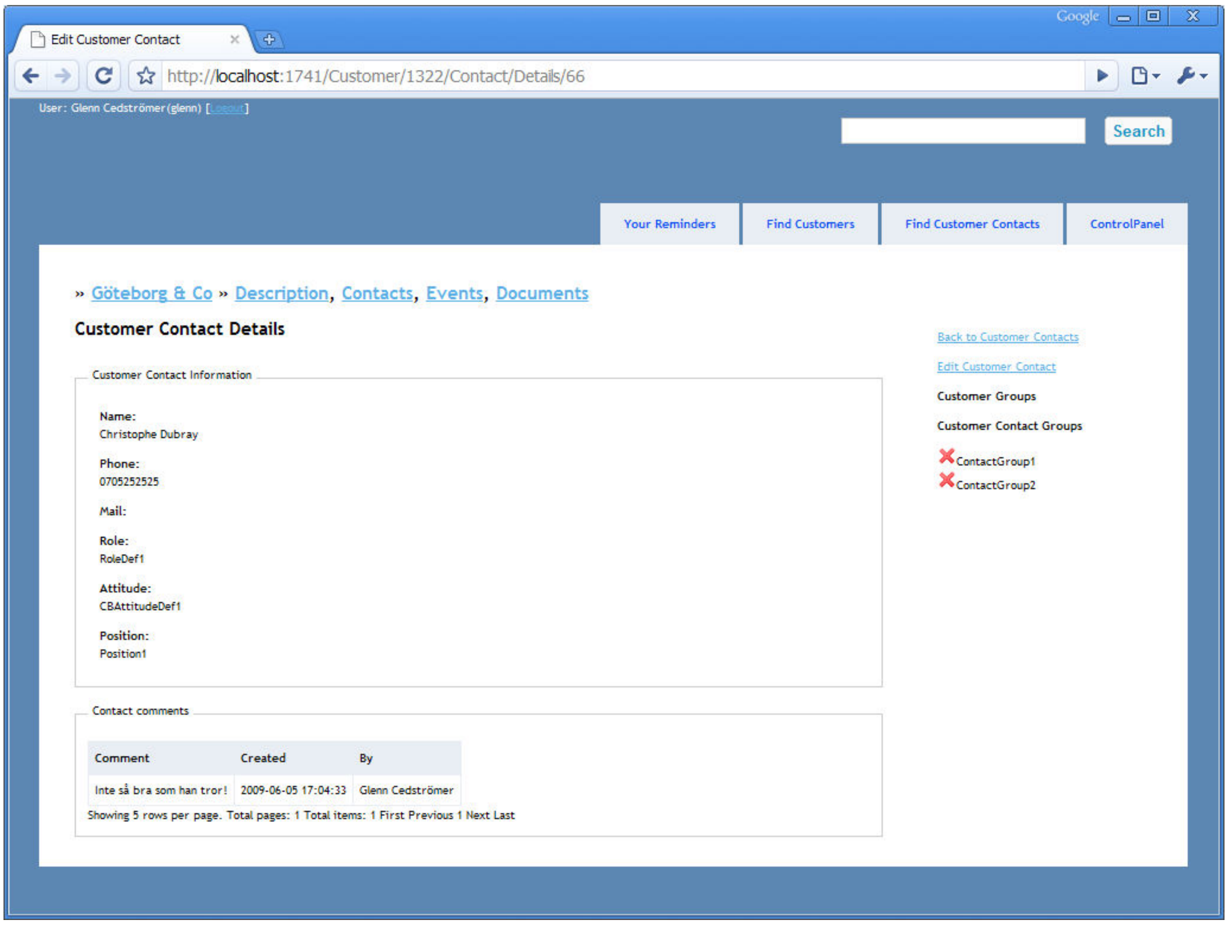
Description	FileName	Size (bytes)	Created At	
Testfile	File To Upload.txt	0	2009-06-05 15:05	✗

5-13 Customer Documents

5.2.10 Customer Contacts

Each customer may have a number of contact persons. On this page the user can see a list of all the current contacts that exists for the chosen customer. Roles, positions and Citybreak Attitude are chosen from defined lists, the definitions are managed in the control panel). This was made to allow easy search and filtering functionality for customer contacts. Other attributes on a customer contacts are name, phone and e-mail.

By clicking on a contacts name, the user will come to the details page where all attributes on that contact is displayed. Besides contact attributes, comments and group belongings for that selected contact is displayed and by clicking the edit link everything becomes editable.


Edit Customer Contact
http://localhost:1741/Customer/1322/Contact/Details/66
User: Glenn Cedströmer(glenn) [Logout]
Search
Your Reminders
Find Customers
Find Customer Contacts
ControlPanel

» Göteborg & Co » Description, Contacts, Events, Documents

Customer Contact Details

Customer Contact Information

Name:
Christophe Dubray

Phone:
0705252525

Mail:

Role:
RoleDef1

Attitude:
CBAttitudeDef1

Position:
Position1

Back to Customer Contacts

Edit Customer Contact

Customer Groups

Customer Contact Groups

ContactGroup1
ContactGroup2

Contact comments

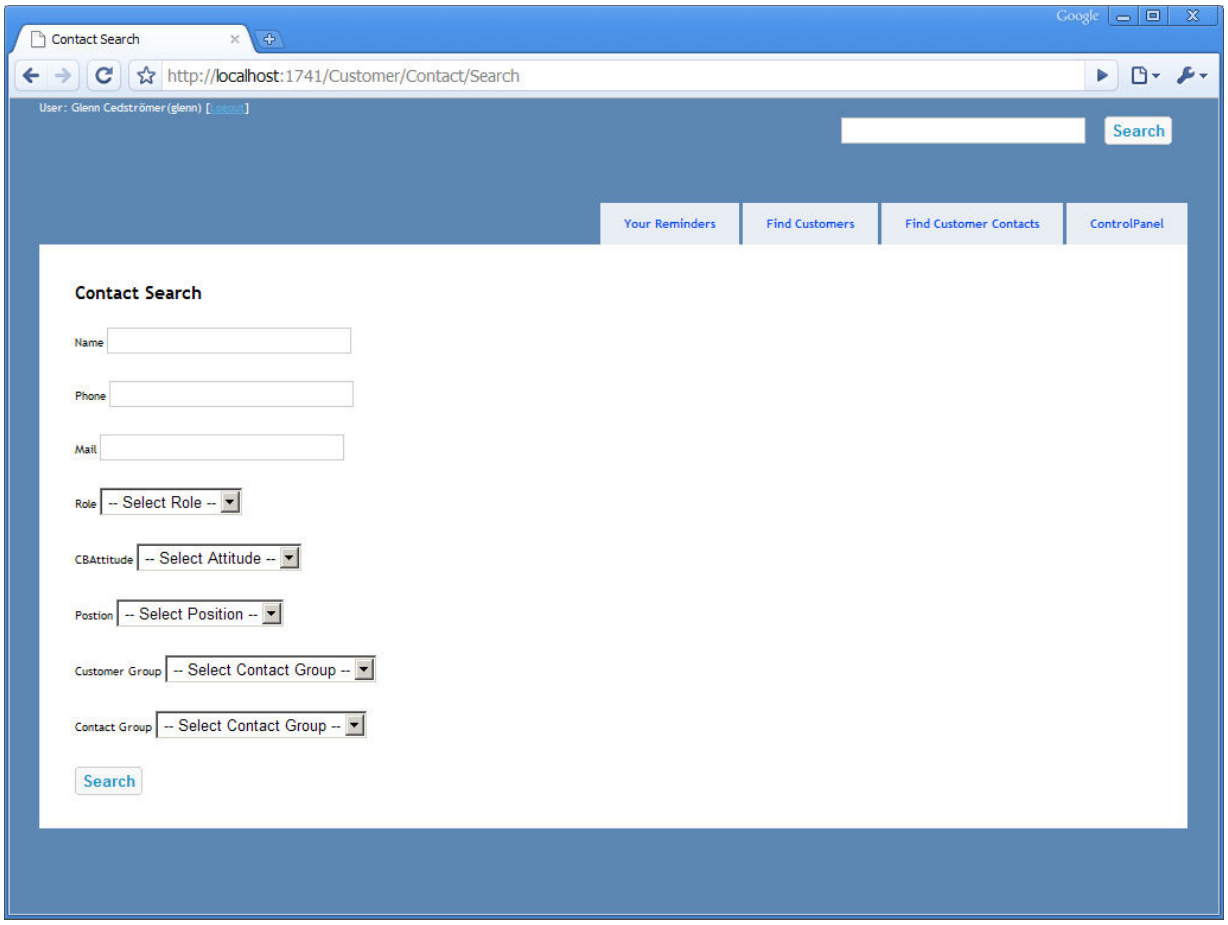
Comment	Created	By
Inte så bra som han tror!	2009-06-05 17:04:33	Glenn Cedströmer

Showing 5 rows per page. Total pages: 1 Total items: 1 First Previous 1 Next Last

5-14 Customer contact page

5.2.11 Customer Contact Search

In this tab, search and filtering for customer contacts are performed. The user can filter on any attribute a customer contact can possess in the search. If the user does not specify anything, it will list all customer contacts currently present in the system. The result will be displayed in a paged list and the user will also have the option to export the result as a Microsoft Excel document by clicking on a link. This will generate an Excel document containing all information about every contact that matched the search.


Contact Search
http://localhost:1741/Customer/Contact/Search
User: Glenn Cedströmer (glenn) [Logout]
Search
Your Reminders
Find Customers
Find Customer Contacts
ControlPanel

Contact Search

Name	
Phone	
Mail	
Role	-- Select Role --
CBAttitude	-- Select Attitude --
Postion	-- Select Position --
Customer Group	-- Select Contact Group --
Contact Group	-- Select Contact Group --

Search

5-15 Customer contact search

5.2.12 Reminders

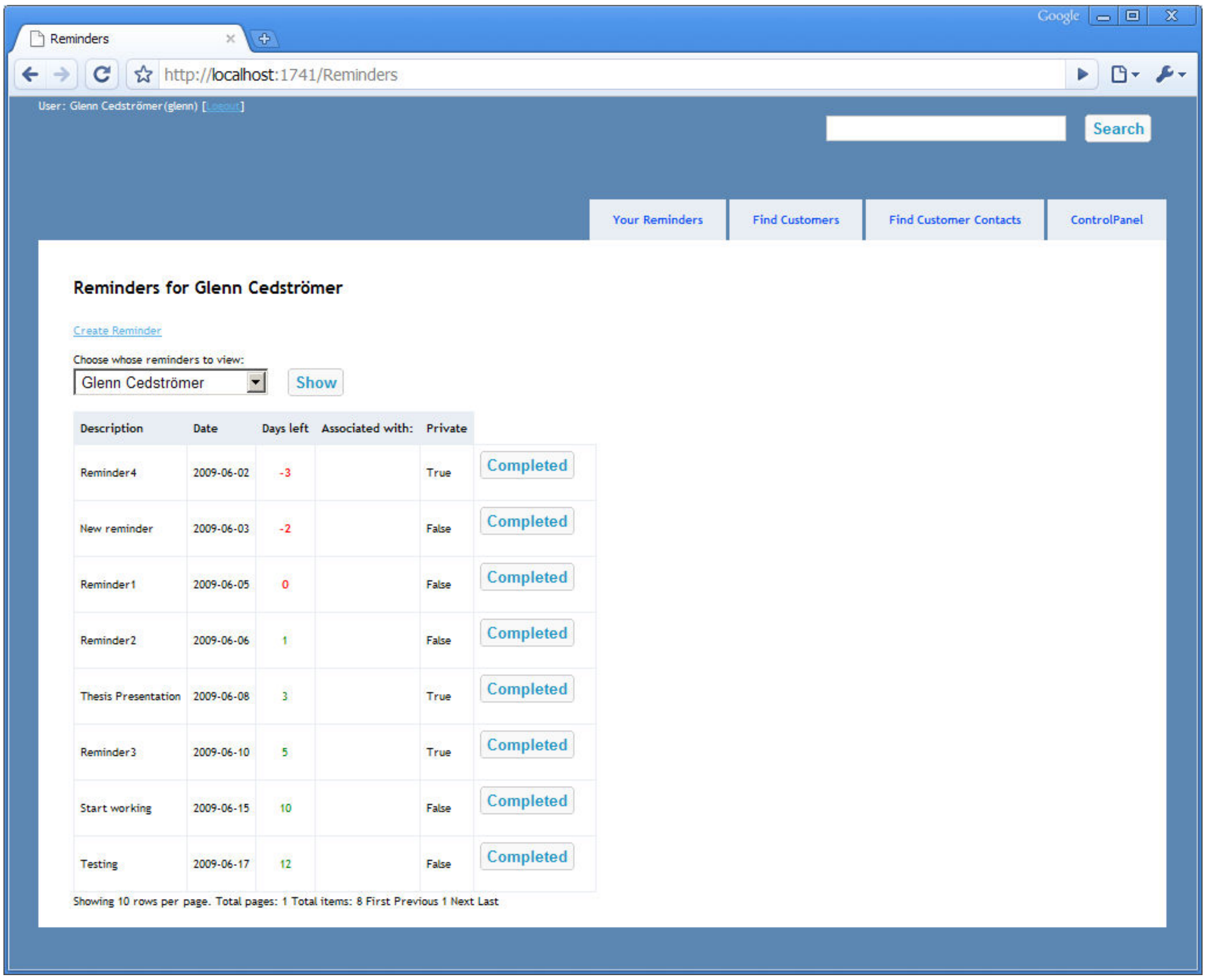
A reminder belongs to a user, however, unless it is marked as private, all users can see each other's reminders. When a reminder is no longer valid, it can be set as completed and will no longer be displayed in the reminders list. The ten most urgent, the ones with shortest time left, will be displayed, but the user will have the option to paginate through the rest of the reminders if they wish. Critical reminders, zero days left or less, are marked with red color and others in green color.

There was also a request by the company to have different kind of reminders. For starters four types of reminders are treated:

- **Default** – The reminder is not connected to anything
- **Support issue** – The reminder is connected to a support issue and should provide a link to that issue

- **Customer** – The reminder is connected with a customer and should provide a link to that customers profile
- **Customer contact** – The reminder is connected with a customer contact and should provide a link to that contacts details page

Since the CRM is not yet integrated with Citybreak support no link is provided if the reminder is of type support issue. Instead a text saying “Support issue id: id” is displayed.



Reminders for Glenn Cedströmer

Create Reminder

Choose whose reminders to view: Glenn Cedströmer Show

Description	Date	Days left	Associated with:	Private	
Reminder4	2009-06-02	-3		True	Completed
New reminder	2009-06-03	-2		False	Completed
Reminder1	2009-06-05	0		False	Completed
Reminder2	2009-06-06	1		False	Completed
Thesis Presentation	2009-06-08	3		True	Completed
Reminder3	2009-06-10	5		True	Completed
Start working	2009-06-15	10		False	Completed
Testing	2009-06-17	12		False	Completed

Showing 10 rows per page. Total pages: 1 Total items: 8 First Previous 1 Next Last

5-16 User's reminders

6 Conclusions

The work that has been conducted consists of further development of the already existing Citybreak Support and development of a completely new Customer Relationship Management (CRM) system for Visit Technology Group. Already from the beginning the company was aware of the fact that they had a too large amount of requirements possible within the timeframe of a master thesis. Therefore an iterative development process, where the client prioritized the requirements, was chosen.

All the highly prioritized requirements mentioned in this thesis were treated and completed. However, there is still a large amount of less prioritized requirements, not mentioned in this thesis, from the company that are still to be implemented. One example of such a requirement is localization.

The greatest limitations of both solutions developed are the set techniques and frameworks used. In the case of Citybreak Support it is developed using the ASP.NET Web Forms framework and the .NET Remoting technology and no real Object Relational Mapper (ORM). Using remoting adds another level of complexity to development of the application, but it is a tradeoff, since remoting could offer benefits such as better scalability. In the current situation, however, with Citybreak Support and the remoting server residing on the same machine, the possible benefits of remoting are not achieved. It is therefore possible to remove the usage of .NET remoting and gain a simpler application structure.

The lack of ORM also increases the complexity and reduces the maintainability of the application. The tight coupling between the view and code behind in .NET Web Forms also increases complexity of unit testing. As for the ASP.NET MVC framework, while having great extensibility, testability and separation of concerns, it lacks infrastructure and only supports the most fundamental HTML components. A limitation concerning LINQ-to-SQL (DLINQ) ORM is that it officially only supports the Microsoft SQL dialect.

Concerning architecture of the CRM system, most of the best practices have been followed in order to deliver a solid, maintainable and testable application.

7 References

Visit Technology Group, About the company, http://www.visit.com/about 2009-05-26

Bill Evjen, Scott Hanselman, Devin Rader, Professional ASP.NET 3.5 in C# and VB, Wiley Publishing, Inc, 2008

Jim Farley .NET vs. J2EE, http://www.ddj.com/java/184414710, 2009-05-26

Martin Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley Professional, 2002

Eric Evans, Domain-driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional, 2004

Microsoft Developer Network, Dependency Injection, http://msdn.microsoft.com/en-us/library/cc707845.aspx, 2009-05-25

Microsoft Developer Network, Model-View-Controller, http://msdn.microsoft.com/en-us/library/ms978748.aspx, 2009-05-25

Nick Berardi, Introducing the ASP.NET MVC (Part 2) - ASP.NET MVC vs. ASP.NET Web Forms, http://www.coderjournal.com/2008/12/introducing-aspnet-mvc-part-2-aspnet-mvc-vs-webforms/ 2009-05-25

Microsoft Developer Network, Full-Text Search, http://msdn.microsoft.com/en-us/library/ms142571.aspx, 2009-05-25

Microsoft Developer Network, GridView Class, http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx, 2009-05-25

W3Schools, ASP.NET - The Repeater Control, http://www.w3schools.com/ASPNET/aspnet_repeater.asp, 2009-05-25

Moxiecode Systems, TinyMCE, http://tinymce.moxiecode.com/, 2009-05-25

Farhan Muhammad and Matt Milner, Real World ASP.NET Best Practices, Apress, 2003

Scott Allen, Extreme ASP.NET: Dissecting an ASP.NET MVC Controller, http://msdn.microsoft.com/en-us/magazine/dd695917.aspx, 2009-05-25

Microsoft Developer Network, UrlRoutingModule Class, http://msdn.microsoft.com/en-us/library/system.web.routing.urlroutingmodule.aspx, 2009-05-25

Microsfot Developer Network, MvcRouteHandler Class, http://msdn.microsoft.com/en-us/library/system.web.mvc.mvcroutehandler.aspx, 2009-05-25

Jeremy D Miller, StructureMap, http://structuremap.sourceforge.net, 2009-05-25

Alex Homer, Design Patterns for ASP.NET Developers, http://www.devx.com/dotnet/Article/33695/1954, 2009-05-25