2. Introduction

It is certain fact that e-commerce systems have been bringing great facilities into our life. However, main complaints concerning to present e-commerce sides have focused on two themes: firstly, it is impossible to reach every product in the real life without any range and amount restriction; secondly; shipping time and security still are the most challenging problems. In addition, customers are dependent on their web browsers to do shopping from traditional e-commerce sides.

Although *Mobile Store* can be deemed as an adaptation of standard e-commerce systems into mobile platforms at the first sight, it has huge differences from typical e-commerce systems. The first aim of the Mobile Store project is to enable people doing shopping from mobile devices and PCs without any restriction in variety and amount of goods. Besides, as Mobile Store reduces the shipping time it raises the security in both delivery of goods and payment.

Consequently, Mobile Store project offers new approaches to classical e-commerce systems and exhibits a model to introduce the approaches.

3. Project Scope and Description

Mobile Store is basically an e-commerce system that is designed for a virtual city named as Mobile City. Aim of this project is offering a new approach to classical e-commerce systems and providing a model to introduce this approach. Although mobile store project was developed for a virtual city to demonstrate a model, it is quite simple adapting the project to real world. The only thing that should be done to adapt the project to real world is adding map and suppliers information of a region into project database and announcing the project to public.

The mobile store project consists of customers, carriers (transporters), stores, store managers, executive software and system administrator as major objects.

- Customers can give order and view their order status using Internet Web Browsers or Mobile Phones.

- Carriers take order information from executive software with their Mobile Phones, buy products in customer shopping-list from member stores and transport these products to customers. In addition, carriers are responsible for updating the order status and road status using their mobile phones. There are currently four carriers in the system but system administrator can add easily more carriers to the system.

- Stores are the suppliers of the system. Any store that accepts the membership agreement can take part in the system.

- Store Managers add/remove product, update the product information and announce the campaigns using Internet Web Browsers.

- Executive software adds customers to the system, takes orders from customers, determines the carrier, route of carrier and stores where products will be bought, and informs the carrier about new order updating the carrier WAP page.

- System administrator adds stores, carriers and roads to the system, removes customers from the system.

Customer orders, carrier positions, map information, points of suppliers and status of goods in suppliers are required input objects for the system. After system receives a customer order via WAP or HTTP, it determines the carrier who enables the shopping in shortest time. Then, it determines the route of the carrier and stores placed in this route according to customer desires. Finally, the information comprised what carrier should do is sent to carrier chosen. This informing process is the main data object produced by the system as output.

3.1 Project Innovation & Creativity

Mobile Store project is a unique project in Turkey and currently it is not known whether there is a similar example in the world. Mobile store project offers a new way to classical e-commerce systems.
The current e-commerce systems attempt to include goods in their bodies and to employ exterior shipping companies to deliver goods. This situation causes an important limitation in range and amount of goods obtainable from e-shops. Also, since classical systems use third-party shipping companies, occurring of shipping-time problems is inevitable. However, mobile store does not attempt to include products. It attempts to include retailers and to deliver goods using its own shipping system.
Thus, mobile store offers shopping without any limitation in range and amount of goods along with shortest shipping method.

Mobile Store project enables the inhabitants of a region to do shopping without any variety restriction. That is, customers have the capability of buying everything that is sold in the stores. Furthermore, they do shopping using their mobile phones as well as using their web browsers thus a mobile phone having WAP property becomes enough tool for shopping. In addition, mobile store project offers the quickest shipping method in the present systems due to the fact that carriers are assigned to pre-determined regions that allows the carriers to reach the suppliers in the shortest time. Also, the system presents three shipping options to the customers.

In the first option, customers determine the suppliers (store names) manually and software sends the route that enables the carriers to do shopping from stores preferred by customers and to arrive at customer address in the shortest time.

In the second option, customers choose " shortest time " then software determine the suppliers reducing the shopping time and the route taking shortest time automatically. In this case, the time becomes the most important factor and stores are selected according to route.

In the third option, customers choose " cheapest form " then software determines the suppliers that provide most economic prices for the products in shopping-list and generates the appropriate route. In this case, cost becomes most important factor and route is determined according to stores.

Finally, since mobile store communicates via WAP with carriers, an additional GPS informing system and a wireless system are not necessary to make a communication between the carriers and the system. As the carriers move, they inform the system about their current locations and traffic status of the roads so the system has the latest information about the region traffic. That means, our customers can use the system additionally as a city-traffic-info stuff that can be reached from mobile phones or web browsers and supported by another .NET web service.

3.2 Social Responsibility or Commercial Viability

If we compare the mobile store project with classical e-commerce systems, we notice that mobile store has significant advantages. Firstly, a large amount of capital is crucial requirement to put a classical e-commerce project into practice because this kind of systems have to start to serve without any deficiency in number and variety of goods. Otherwise, the first impression on customers never becomes positive and it is almost impossible to gain the interest of a customer who had a negative feeling about your site later. However, the only thing to put the mobile store project into practice is to rent a small number of carriers for a few months and for a well-determined region plus to announce the existence of the project to inhabitants of region. If the project fails the things that are lost are not so much compared to the things that are lost when a traditional e-commerce project fails. But, if project becomes successful you can enlarge the system adding more regions, more carriers along with more stores into system and making more profitable agreements with suppliers. . Finally, the failing possibility of the project is very low if it is designed and planned well since it brings the quickest shipping method (at most a few hours), unrestricted product variety compared to real world, the shopping change with use of mobile phones as well as PCs and same prices with the stores if customers do shopping over a pre-determined limit. (Otherwise, they pay a little amount of shipping money)

3.3 Division of Labor

Hüseyin Hasçelik is responsible for implementation of the database on SQL Server and Web Services with the web site of Mobile Store Project.

Zülküf Genç is responsible for implementation of Mobile Part of the Mobile Store Project along with the Mobile Store Simulator.

4. Asp.Net And .Net Framework

4.1 Learning from the History of ASP

We can trace the history of ASP right back to 1995 and the momentous occasion when Microsoft realized they were falling behind in a fundamental shift in the industry by not embracing the Internet. Up until that point Microsoft had been developing their proprietary technologies, tools, and network protocols for the Microsoft Network; all of a sudden they needed an Internet strategy and fast. Microsoft has gone from a position of playing catch-up to one close to dominance, with the Internet Explorer Web browser having a strangle-hold on the Web browsing market, and Internet Information Server (IIS) installed at the majority of Fortune 1000 companies.

4.2 The Origins of ASP

Back in the mid '90s, when the commercial Web world was still young, there was not a great deal of choice of tools for the Web developer who wanted to make his or her Web site a truly useful place to do business.The choices were limited in both available server-side programming platforms and also desktop development tools to produce the solutions. In the end, the programmer was stuck with clumsy Common Gateway Interface (CGI) programs using compiled languages such as C, Delphi, and Visual Basic, or interpreted scripting languages like Perl or Rexx, and operating system shell scripts on systems such as UNIX. In early 1996 Microsoft had a first stab at improving the situation by including the Internet Server Application Programming Interface (ISAPI) technology as part of Internet Information Server. ISAPI is an extension to the Windows Win32 API. It was developed as a way to create Web server software that interacts with the inner workings of Internet Information Server, bringing what was claimed to be a five-fold increase in performance. As you can well imagine from this description, as well as the immediate performance increase, it also had a side effect of increasing the complexity of the development for the programmer. It wasn't for the faint hearted, and it takes some serious hardcore programming knowledge to do ISAPI applications right.As well as ISAPI, Microsoft encouraged developers to embrace their Internet Database Connector (IDC) technology.This was a new way to connect Web sites to back-end databases through Open Database Connectivity (ODBC). The ISAPI and IDC technologies lifted Microsoft's youthful and as yet unproven Web server from being a glorified file server to being a basic interactive application server platform for the first time. Other vendors had tools out there, and several were very popular, such as Netscape Livewire. Livewire was a technology that ran under Netscape's Web server and used a version of JavaScript for page logic, and also used Java components. Unfortunately, Livewire had similar limitations to ISAPI in that it was a compiled technology and the server needed stopping and starting to make changes visible.

4.3 Why ASP Was Needed

Not all Web developers have the programming skills needed to write ISAPI applications, and because ISAPI requires the compilation of programs, there are extra steps in producing an ISAPI-based site that slow development down. Novice and intermediate programmers found the need to learn an industrialstrength language, such as C++, and compile even the simplest of their page logic into .dll files a real barrier. Visual Basic programs, although easier to develop, when used for CGI, performed poorly and the overhead hogged resources. Other languages such as Perl require the Web server to launch a separate command-line program to interpret and execute the requested scripts, increasing page-load time and reducing server performance. CGI itself hogs resources because every page request forces the Web servers to launch and kill

new processes and communicate across these processes. This is time consuming and also uses up precious RAM.

Another problem facing development teams in the mid '90s was the fact that a Web site is a mixture of Hypertext Markup Language (HTML) and logic.They needed a way to mix the programmer's code with the designer's page-layout HTML and designs without one messing up the other.There were many solutions to this problem, ranging from custom template systems to Sever Side Include (SSI) statements that told the server to execute code based on special HTML comment tags. Database-driven interactivity was another challenge.The demand for complex Web sites had just kicked off, and developers needed to supply that demand in a manageable fashion, but the tools available did not make this an easy task.Those who could achieve it demanded rewards that matched the difficulty of what they were being asked to do. What was needed was a solution for the rest of us. It needed to be a simple scripted text-based technology like Perl, so developers could tweak and alter their pages without compilation and with simple text-editing tools such as Notepad. It needed to have low resource requirements while keeping high performance; therefore it needed to be executed within the server environment just like ISAPI, but without the complexity. Designers and cross-discipline teams demanded that it should include SSI and template features to make integrating page layouts simpler to manage.To be truly popular, it should run off a language that would be easy to pick up and was familiar to a large community of developers. Enter Active Server Pages!

4.4 Why ASP Was Not Originally Embraced

Active Server Pages was not an overnight success, though understandably it did capture the imagination of a large sector of the development community, particularly those already well versed in Visual Basic programming or Visual Basic for applications scripting. Others who did not have an investment in Visual Basic knowledge found the limitations of Visual Basic, and by extension Visual Basic Scripting, reasons to avoid the technology. Faults included poor memory management, the lack of strong string management abilities, such as Regular Expressions, found in other established languages.When compared to CGI with Perl,ASP was found lacking. At that time, Internet Information Server was in its infancy, and take-up was low, despite Microsoft's public relations juggernaut going into full flow after the company's much-reported dramatic turnaround. In comparison to current versions of the software it seems very poor, but it was still competitive on performance. Until 1997, back-end Web programming was pretty much owned by CGI and Perl. High-performance Web sites usually had a mix of C-compiled programs for the real business engine, and Perl for the more lightweight form processing. There was a fair amount of doubt and suspicion around Microsoft's Internet efforts, including IIS and Internet Explorer, and ISAPI had not done all that much to bring across a huge sector of the development community. Despite this uncertain atmosphere, Microsoft saw many Windows NT 4 licenses being bought specifically for Web hosting and development increasing.Third-party support for anything other than small components was initially slow, but, as with all Microsoft products, after the first couple of releases they usually get things right, and ASP was no exception. Whereas Perl had a huge community of developers led by the heroic figure of Larry Wall, the ASP developer was not yet well supported.A Perl programmer was encouraged from the top to share and make his or her code open, so the community thrived, with every conceivable solution or library just a few clicks away at the Comprehensive Perl Archive Network (CPAN) site, or at one of the many other Web sites and news groups. Contrast this with the ingrained competitive and financially led philosophies of the third-party component vendors in the Windows Distributed Internet Applications (DNA) world. Of course, it did not take the ASP community long to grow to be the loving, sharing success it is now.

4.5 The ASP Timeline

- **December 1995** Microsoft makes a dramatic U-turn and announces that their whole product lineup will be refocused to embrace the Internet. Up until this point they had largely ignored the Internet market and had fallen dangerously behind the competition.

- **February 1996** Microsoft releases Internet Information Server to the public for free download. Microsoft spokespeople claim that the server offers a four-fold increase in performance over Netscape Netsite server. IIS includes ISAPI and IDC technologies. With the release of Windows NT 4, IIS version 2 is bundled, while IIS 1 is available for Windows NT 3.51.

- **October 1996** Microsoft releases the public beta for IIS 3 as an optional upgrade to IIS 2.The major change with this version is the inclusion of a new development environment called Active Server Pages, formerly known under its project name of "Denali." As part of their public relations campaign, Microsoft claims they are beating Netscape 2- 1 in the server market. IIS no longer supports MIPS and NT 3.51.

- **August 1997** Microsoft releases ASP 2 with IIS 4. IIS now includes the Microsoft Management Console (MMC) to make administering the server more straightforward, and the SMTP server is now bundled, having previously been a part of the Commercial package. IIS and ASP are now tightly integrated with Microsoft Transaction Server, and this is seen as a real step forward in making the platform a credible choice for large-scale deployment.

- **1998–2000** Microsoft started releasing incremental versions of the language Scripting Engines, adding language features and functionality without the need for full ASP version updates, such as the addition of Regular Expressions for VBScript programmers. With the release of Windows 2000 with IIS 5, Active Server Pages 3 became available. ASP 3 allowed for server-side redirects, better error support, ADO 2.5 with support for XML, and caching of compiled code. IIS 5 enabled the administrator to finely separate processes to prevent crashing of the server.

- **July 2000** .NET makes their first public announcement, revealing their new C# language, promising to deliver better functionality and flexibility than ever before, and promising support for a wide variety of Internet standards.

4.6 Final Changes to Original ASP Model

With version 3, Microsoft introduced the concept of server scriptlets.These were COM objects that were developed as Extensible Markup Language (XML)-based text files.This enabled programmers to rapidly prototype multi-tiered application business logic without the "change, recompile, upload, stop the server, register, test, change" cycle of component development.ASP and ActiveX Data Objects (ADO) were given a boost in capability with the addition of XML-processing abilities. XML was, at this point, a massive deal in the developer community, and Microsoft wanted to appear to be fully embracing it, and so the whole of Microsoft's product line seemed to be receiving an XML makeover. As well as the new script execution changes mentioned earlier, it included many other performance improvements, such as the ability of the Web server to self-tune, checking adding threads when needed, and having response buffering on by default.

4.7 Weaknesses in the ASP 3 Model

Despite the great achievements of Active Server Pages, particularly in the areas of speed and stability, the platform was still based on incomplete scripting languages of VBScript and JScript, and third-party languages such as Perl. Scripting languages required the developer to compromise coding standards and bolster the application with components written in a second language, usually C++ or VB.The languages were not properly object oriented, although they were object-aware, and could never perform very well whenever they required an interpreter to execute. The reliance on the systems administrator for Web server configurations was also a problem; the administrator must register components, settings, and permissions on the server, and so deployment was not as simple as just uploading your files. Programmers were bound to ask, after several years of Java programmer colleagues evangelizing Java Server Pages,"What is Microsoft going to do?"

4.8 The Need for a New ASP Model

It was evident that Microsoft would require a fundamental change to bring ASP up to the standard of industrial-strength programming. Active Server Pages was a technology based on the foundations of COM. ActiveX and COM technology provided much of its strength, but also many of its limitations. Microsoft would need to have a long hard look at COM to see how it could improve, and these changes would be bound to affect ASP. At the same time,Microsoft realized that the developers' playing field was changing, with new standards arriving all the time, particularly in information-sharing and distributed applications using XML, such as Simple Object Access Protocol (SOAP) and XML-RPC.Web services were becoming all the rage; Java was everywhere, and XML was taking the developer community by storm.A new version of ASP was not going to be enough to meet these demands; the changes must be more far-reaching if they were not just going to catch up but also take the lead against such tough challenges. ASP and Windows DNA, being based on early 1990's COM and Win32 API technologies, did not provide a very coherent technical architecture roadmap for modern distributed applications, whereas with Java 2 Enterprise Edition (J2EE), Sun had a suite of technologies that developers could follow, starting small with Standard Edition projects and scaling up to full Enterprise JavaBeans. In today's world, we do not have to contend just with different Web browsers but also with different distribution channels and modes of operation, with mobile phones and computers, interactive digital TV, intelligent appliances, digitally networked homes, and possibly moving from Web pages to disposable applications and Web services. No doubt, as Microsoft was looking at their own technologies they must have analyzed the competition. As they announced the .NET framework, they also introduced a new language for the twenty-first century, C#. C# and .NET would address all of the criticisms, provide for a whole new way of looking at applications and the Web, and replace everything that had gone before, including Microsoft's flagships Visual C++,Visual Basic, and Active Server Pages.

4.9 Reviewing the Basics of the ASP.NET Platform

Microsoft has done a great job of bringing ASP and their older languages into the twenty-first century with .NET.ASP.NET, using VB.NET, is now a full-fledged object-oriented Web application development platform, and has seen many improvements; but the past legacy languages should not hold back a new initiative as massive as .NET, so Microsoft developed a new headline-grabbing language for the .NET Framework, called C#. C# was built from scratch as *the* .NET language.While it has features familiar to C programmers, and it has some of the great RAD features so beloved by Visual Basic programmers, it is completely new. Some have said that C# is Microsoft's "me too" language to compete with Sun's Java. If Microsoft does one thing well, that is building developer tools, (remember,the product that first put Microsoft on the map was their version of Basic), and C# with Visual Studio.NET certainly lives up to expectations. C# is a truly modern language with all the features you could wish for, such as full object-orientation (unlike the C++ bolted-on approach), automatic memory management, and housekeeping.

4.10 Some key points about ASP.NET

- ASP.NET is a key part of the wider Microsoft .NET initiative, Microsoft's new application development platform.

- .NET is both an application architecture to replace the Windows DNA model and a set of tools, services, applications and servers based around the .NET Framework and common language runtime (CLR).

- Rather than just being ASP 4 or an incremental upgrade,ASP.NET is a complete rewrite from the ground up, using all the advanced features .NET makes available.

- ASP.NET can take advantage of all that .NET has to offer, including support for around 20 or more .NET languages from C# to Perl.NET, and the full set of .NET Framework software libraries.

- Web applications written in ASP.NET are fast, efficient, manageable, scalable, and flexible, but, above all, easy to understand and to code!

- Components and Web applications are all compiled .NET objects written in the same languages, and they offer the same functionality, so no need to leave the ASP environment for purely functional reasons.

- You'll have less need for third-party components.With a few lines of code,ASP.NET can talk to XML, serve as or consume a Web service, upload files, "screen scrape" a remote site, or generate an image.

With the .NET Framework and ASP.NET,Microsoft has not just shown itself to be a contender in Web development technologies, but many commentators also believe Microsoft has taken the lead.ASP.NET is well equipped for any task you want to put to it, from building intranets to e-business or e-commerce megasites. Microsoft has been very careful to include the functionality and flexibility developers will require, while maintaining the easy-to-use nature of ASP.

- With ASP.NET you now have a true choice of languages. All the .NET languages have access to the same foundation class libraries, the same type of systems, equal object orientation and inheritance abilities, and full interoperability with existing COM components.

- You can use the same knowledge and code investment for everything from Web development to component development or enterprise systems, and developers do not have to be concerned about differences in APIs or variable type conversions, or even deployment.

- ASP.NET incorporates all the important standards of our time, such as XML and SOAP, plus with ADO.NET and the foundation class libraries, they are arguably easier to implement than in any other technology, including Java.

- An ASP.NET programmer still only needs a computer with Notepad and the ability to FTP to write ASP code, but now with the .NET Framework command-line tools and the platform's XML-based configuration, this is truer than before!

- Microsoft has included in the .NET Framework an incredibly rich feature set of library classes, from network-handling functions for dealing with Transmission Control Protocol/Internet Protocol (TCP/IP) and Domain Name System (DNS), through to XML data and Web Services, to graphic drawing.

- In the past, the limitations of ASP scripting meant components were required for functionality reasons, not just for architectural reasons. ASP.NET has access to the same functionality and uses the same languages in which you would create components, so now components are an architectural choice only.

- A .NET developer is shielded from changes in the underlying operating system and API, as the .NET technologies deal with how your code is implemented; and with the Common Type System, you don't have to worry whether the component you are building uses a different implementation of a string or integer to the language it will be used in.

4.11 Converting Code into Multiple Languages

As supplied by Microsoft,ASP.NET and the .NET Framework consist of three main languages: JScript.NET,VB.NET, and C#. Other vendors have available or have announced many more, such as Perl.NET, COBOL.NET, and a version of Python.

JScript has been updated to be a full-fledged language and to take account of the object-oriented nature of .NET. Experienced JScript developers should feel very at home and be pleasantly surprised at the new additions.

VB.NET replaces VBScript support, but is similar enough in operation that it isn't too steep a learning curve for VBScript programmers, and as with JScript above, it provides you with full access to all that .NET has to offer, including, for the first time, full object orientation.

C# has been (perhaps unfairly) described as J++ mark 2.There is more to it than that. C# is effectively C++ built from scratch.The problems with C++ are well documented, so there is no need to go into them here, but suffice it to say that in C++, object orientation was an optional bolted-on afterthought, whereas in C#, it was built in from the ground up.

All the functionality and support of the .NET Framework is available to any of the .NET languages, and in addition, objects written under one language can be used, inherited, and extended under any of the others.This is a very powerful concept and

introduces the idea of language independence.This is achieved through the Common Language Runtime technology.

The CLR takes your .NET language code and converts it into an intermediate language (Microsoft Intermediate Language [MSIL]), and this intermediate language is then compiled to target machine-specific binary code.The Intermediate Language specification is one of the many .NET technologies that have been submitted to standards bodies, and several projects are under way to transport the software over to non-windows platforms, such as Mono and Portable.NET in the open source community, and to developments from Corel and Borland.

4.12 Comparing Improvements in ASP.NET to Previous ASP Models

The first difference an experienced ASP developer will notice is that VBScript support has been dropped in favor of VB.NET.This is not as much of a hurdle as it sounds like, as the syntax is quite similar, and VB.NET is a full-fledged language and so provides a lot richer environment than VBScript ever could.

As described above, all ASP.NET languages are object oriented, event driven, and server compiled.This brings many benefits, especially where improvements were needed most, namely performance, stability, scalability, and manageability.

With Classic ASP, you pretty much had to code your whole application from scratch.ASP.NET has several labor-saving additions to make life easier.Web forms introduce a new Visual Basic Rapid Development-style way of looking at forms in Web pages.With Web Forms, the developer uses new form components that you can add in the traditional way or through code, and they enable the programmer to call on server-side event-driven programming and true separation of layout and logic.You can separate the layout code and functions by using *code behind* pages that use inheritance to add methods to the form. .NET form controls maintain the session state so the users input remains when the page is submitted, and the controls' property values are available to the ASP code without resorting to querying the request object.

The framework foundation class libraries contain exciting new features, previously only available from third parties such as the System.Drawing tools, which enable you to build dynamic images on the fly, built-in browser-based file upload and system network services for working with TCP/IP and DNS.

With Web Services and built-in support for SOAP you can distribute code and applications.Your ASP.NET scripts can consume services across the Web, and publish and expose routines as services just as easily.

Deployment, including server configuration, is mostly just a matter of transferring files with configuration that was previously only available from the MMC now implemented with XML files. Now you do not need to register and unregister components, and the server can handle multiple versions of the same component without conflicts.

Mission critical services has increased support with load balancing and several state-management options, including the ability to store state information in an SQL Server database and pass the session ID on the URL to avoid requiring the user to have cookies.

4.13 How Web Servers Execute ASP Files

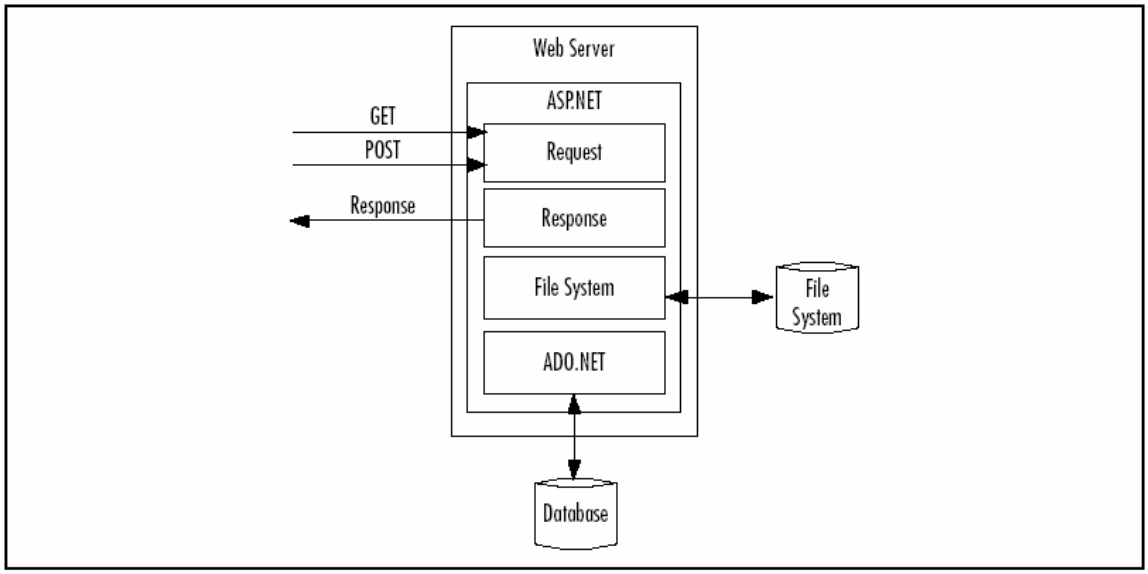
When a site visitor requests a Web page address, the browser contacts the Web server specified in the address URL and makes a request for the page by formulating a HTTP request, which is sent to the Web server.The Web server on receiving the request determines the file type requested and passes processing to the appropriate handler.ASP.NET files are compiled, if necessary, into .NET Page classes and then executed, with the results sent to the client's browser.

Compilation means that on first load ASP.NET applications take longer to display than previous versions of ASP, but once compiled they are noticeably faster.

4.14 Client-Server Interaction

ASP.NET applications are a mixture of client side markup and code, and server side processing.When an ASP.NET Web form page is downloaded to the visitor's Web browser, additional code is included to previous ASP versions.This extra code enables richer form functionality, including server and client side events, validation, and the ability to maintain form value state.The server determines the visitor's browser type and sends markup to match the browser's abilities.

Some client interactions will be dealt with within the visitor's browser, while others will require information to be posted to the server for processing and the altered page returned. As form responses are received, the form values are maintained in a new facility of ASP.NET "State Bags" and are compressed into a hidden form element containing the page "Viewstate."This allows the form elements that the visitor has interacted with to maintain the same values as when the page was submitted. As illustrated in figure below, the browser can request information from and send information to the server using two HTTP methods, *GET* and *POST*.



GET is simply the method in which the browser compiles a URL.A typical URL in this context will consist of a protocol, for example, HTTP for hypertext or FTP for file transfer, a fully qualified domain name, such as "www.aspalliance.com," followed by a path, such as "/chrisg/", and then the page to *GET*, such as "default.asp" or "index.html."You can add information as parameters, called a *querystring*.This is

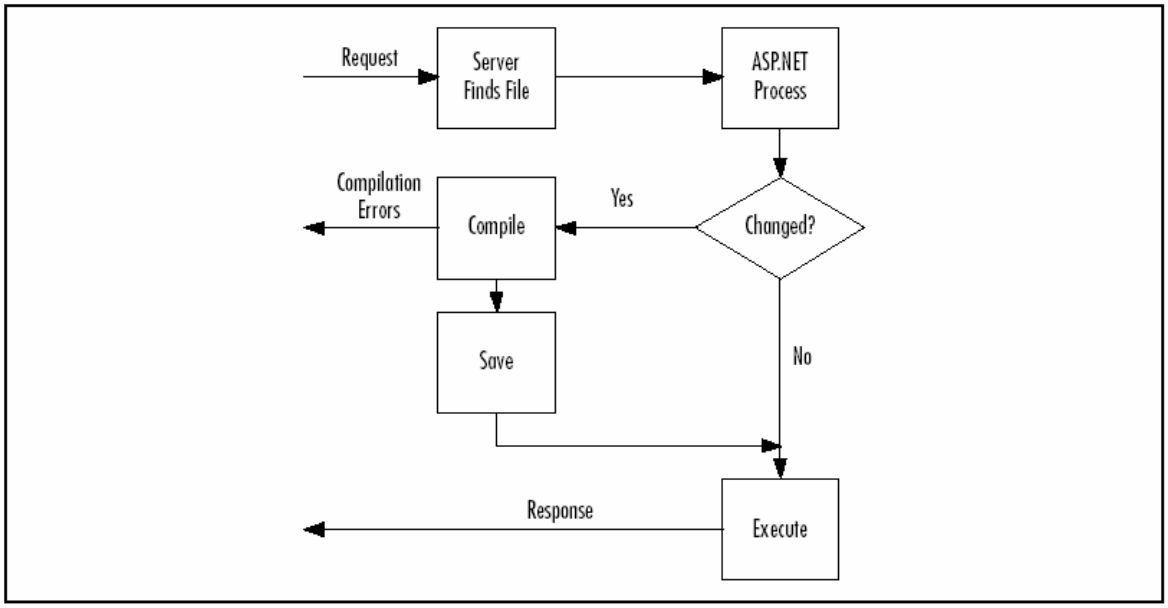
separated from the rest of the URL with a question mark, and the parameters take the form of keywords and values such as "keyword=value," for example,"article=5." Multiple parameters are separated with ampersands, so if we have two parameters, *foo* and *bar*, they would be presented like **foo=a&bar=z**. So, a full GET request including querystring could be **http://www.abcxyz123.com/site/index.asp?page=5**.

When a browser sends information using the *POST* method, the parameters are compiled in the same way but sent separately in the HTTP header, and so are not seen in the URL portion of the browser like *GET* requests are. Forms often use *POST* for this very reason.

Other information goes into the HTTP request header, such as what browser the user is using and so on. As you will see later, your ASP can pick up this header information and the querystring parameter values.

4.15 Server-Side Processing

When the server receives this request, it will find the page that was requested using the path information specified, and the relevant system will process the page. In the case of Classic ASP, there was not much to this process, although a certain amount of caching happened. As you will see in Figure 1.2, with ASP.NET the process is a fair amount more involved but provides for much faster processing and delivery.



The server will process the ASP.NET page using a special .dll especially for ASP.NET.As with previous versions of ASP,ASP.NET has a large collection of objects that deal with processing certain functions such as the HTTP request, databases, the file system, and forming the response. When the response is complete, it is flushed back out to the user's browser, usually as HTML but not necessarily, and the browser renders this page as it arrives as the page on screen.

4.16 Compiling and Delivering ASP.NET Pages

The process of compiling and delivering ASP.NET pages goes through the following stages:

1. IIS matches the URL in the request against a file on the physical file system (hard disk) by translating the virtual path (for example, /site/index.aspx) into a path relative to the site's Web root (for example, d:\domains\thisSite\wwwroot\site\index.aspx).

2. Once the file is found, the file extension (.aspx) is matched against a list of known file types for either sending on to the visitor or for processing.

3. If this is first visit to the page since the file was last changed, the ASP code is compiled into an assembly using the Common Language Runtime compiler, into MSIL, and then into machine-specific binary code for execution.

4. The binary code is a .NET class .dll and is stored in a temporary location.

5. Next time the page is requested the server will check to see if the code has changed. If the code is the same, then the compilation step is skipped and the previously compiled class code is executed; otherwise, the class is deleted and recompiled from the new source.

6. The compiled code is executed and the request values are interpreted, such as form input fields or URL parameters.

7. If the developer has used Web forms, then the server can detect what software the visitor is using and render pages that are tailored to the visitors requirements, for example, returning Netscape specific code, or Wireless Markup Language (WML) code for mobiles.

8. Any results are delivered back to the visitor's Web browser.

9. Form elements are converted into client side markup and script, HTML and JavaScript for Web browsers, and WML and WMLScript for mobiles, for example.

5. Net XML Web Services Overview

5.1 What Are Web Services

- "Software as a Service".
- Programmable components that provide business logic to web or desktop applications using XML for message formatting and the standard Internet protocols (HTTP, SSL, etc.) for communications (SOAP). Kind of the "Internet-Enabling" of the Windows DNA architecture.
- Not a new concept - and .NET is not required to build SOAP servers or clients. The MS vision is simply that .NET becomes the easiest way (thus most popular).
- MS doesn't own the standards.
- Part of the .NET initiative is "My Services" (formerly known as "Hailstorm"), which are a set of Microsoft .NET Foundation Web Services. The first example of this is Passport. Another good example is Microsoft's TerraServer.

5.2 Why Use Web Services

- No physical distribution of software. Less cost, easier maintenance.
- Easier to implement than DCOM, Sun's RMI, CORBA. Web Services do not rely on any proprietary standards or platform.
- Designed with a loose-coupling between the service provider and the service consumer. One can run without the other.
- XML is the defacto standard for data interoperability.

5.3 Wire Formats

- Web Services use the standard Internet communication protocols (HTTP, TCP/IP). This enables basic communication between providers and consumers, regardless of platform.
- SOAP is a messaging protocol that uses XML for formatting service requests and replies.
- SOAP defines an envelope formatting and processing mechanism for complex message structures and allows for loose coupling.

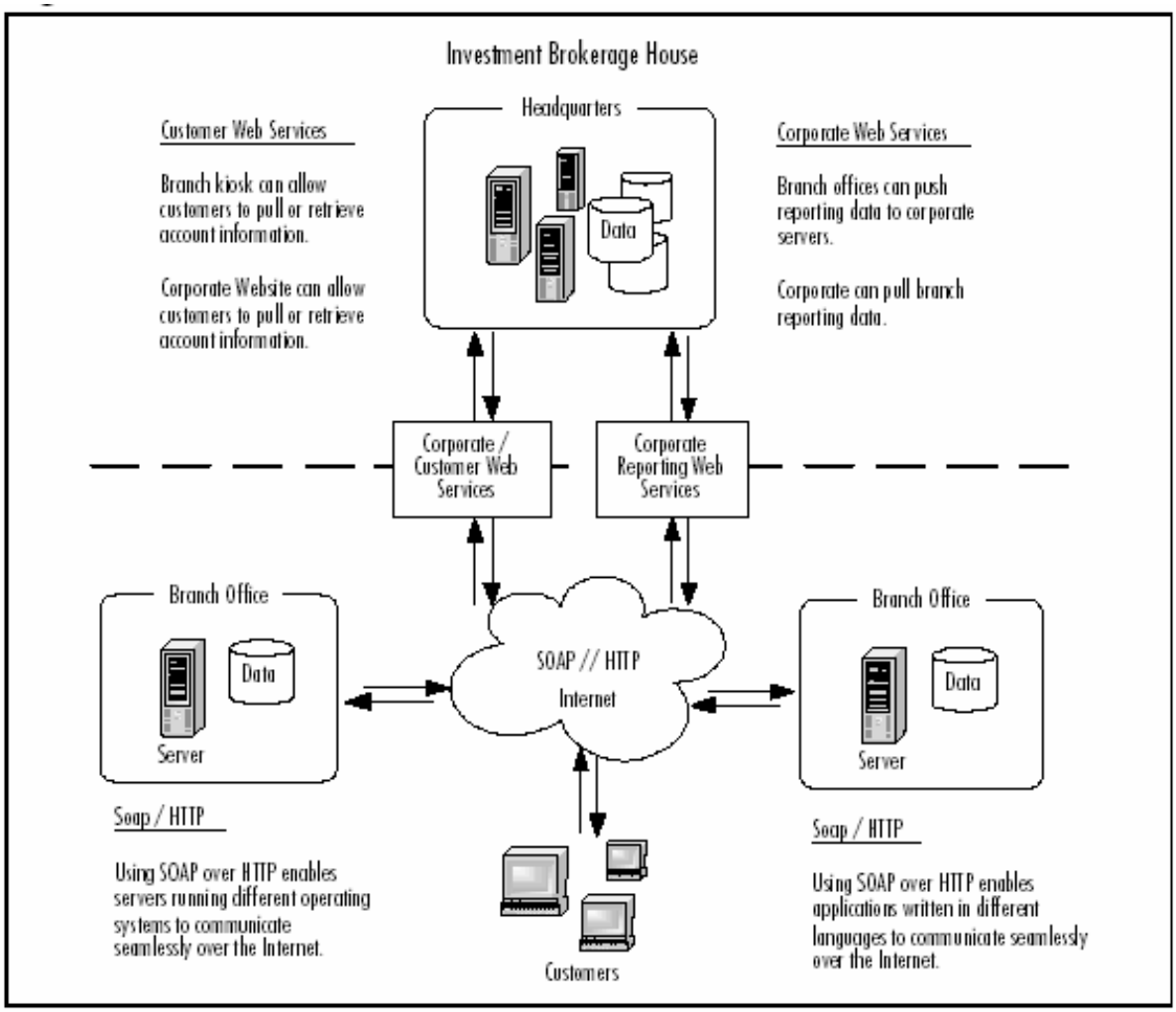
5.4 Web Service Description

- A Web Service Description defines all the supported methods that a Web Service provides.
- This is WSDL (Web Service Description Language).
- WSDL is an XML grammar that developers and development tools use to represent the capabilities and syntax of a Web Service.
- The .NET Framework has utilities for dealing with WSDL, such as Wsdl.exe.
- Visual Studio.NET can retrieve the WSDL for a Web Service and incorporate the service into your project.

5.5 Web Service Discovery

- This is the aspect of making the presence and capabilities of a Web Service known to the world.
- This is the UDDI (Universal Discovery, Description, and Integration) business registry service.
- Initiated by Ariba, IBM, and Microsoft. Supported by more than 130 companies.
- Provides a standard place to register Web Services. Check out www.uddi.org.
- UDDI is a specification built on SOAP/XML and defines a document format and protocol for searching and retrieving discovery documents.
- DISCO (Discovery of Web Services) is a Microsoft protocol for retrieving the contracts for Web Services (WDSL documents).

Where Do Web Services Fit In?



5.6 Creating Web Services

- Incredibly easy using the .NET Framework. Start by opening a new Visual Studio "Web Service" project, or simply code from scratch in Notepad.
- Like Web Forms, Web Services are part of ASP.NET. You create your Web Service as an ".asmx" file with the following directive:

```
<%@ WebService Language="C#" Class="MyClass" %>
```

- Code can be in any .NET language. Code can be in the ".asmx" file or in a separate module.
- Your class always inherits from System.Web.Services.WebService.
- To expose a method, you use the [WebMethod] attribute and make the method public.

5.7 Calling a Web Service via HTTP Get

- By requesting a Web Service URL from the IE Address field, ASP.NET will respond with a neatly-formatted page that describes the Web Service and it's methods. This page even provides a simple means to run the methods.

- This is not UDDI or DISCO or SOAP. Just a nicely-formatted page built from the metadata.

- You can get the WSDL by appending "?wsdl" to the URL.

- To call a method, append the method name and parameters to the URL like this:

```
/MethodName?Parm=Value&NextParm=NextValue…
```

- Calling the service in this manner will result in a simple XML response containing the return value.

5.8 Calling a Web Service via HTTP Post

- The WSDL describes the requirements for doing this.

- The Web Service expects that the incoming parameter values be contained in FORM fields with specific names. Therefore, your FORM has to contain INPUT elements named according to the WSDL. The ACTION attribute names the method:

```
<FORM Method="Post" Action="StockTicker.asmx/GetStockPrice">
```

- The response is an XML string.

5.9 Calling a Web Service via SOAP

- Using SOAP is the most comprehensive manner to invoke a Web Service, but can be complicated.

- Fortunately, the .NET Framework provides utilities to make this easier.

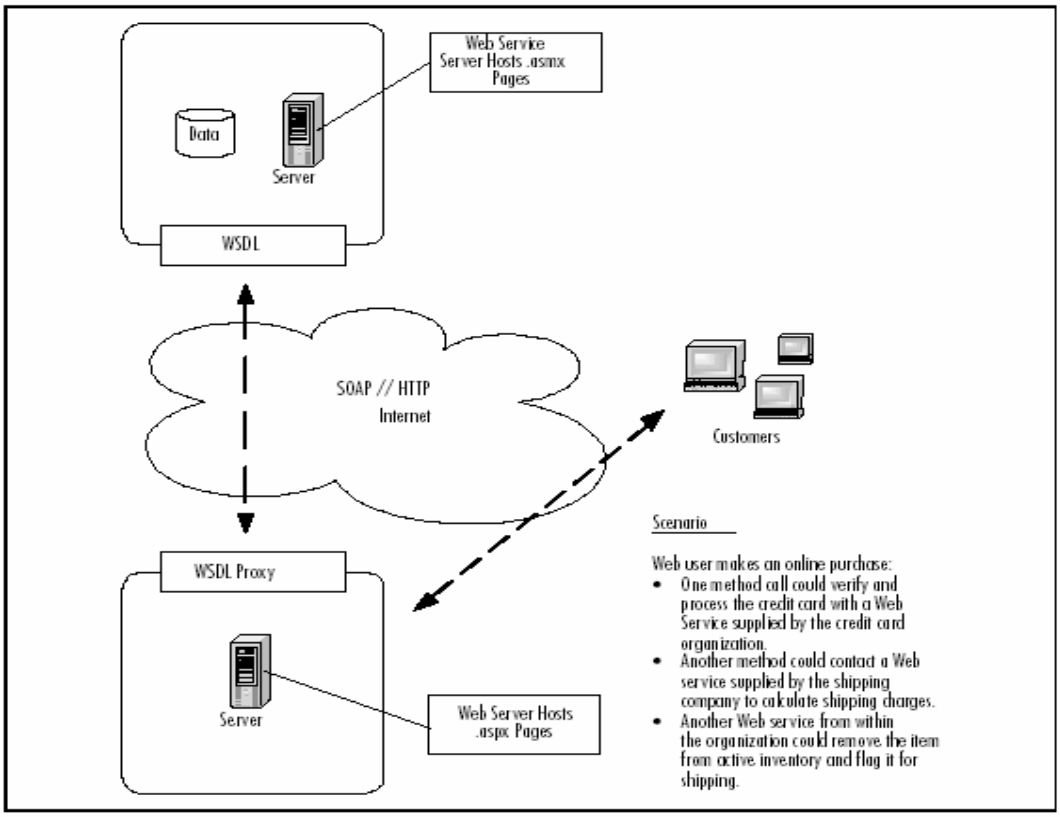
- You build a "proxy" class which is a mirror image of the Web Service. It is run locally on your system, and appears to your application as the Web Service.

- The proxy handles the SOAP communications with the actual Web Service.

- You generate the proxy class using the "Web Services Description Language" tool (wsdl.exe).

- Note that Visual Studio will do this for you simply by adding a "Web Reference" to your project.

Where WSDL and WSDL Proxies Fit into the Internet User Page Request Process



5.10 Web Services Summary

The power of Web Services is due to its foundation in nonproprietary protocols and standards.

Web Services would not be as useful if it were not built on XML for defining data and structure, XSD for defining structure, SOAP for defining a messaging transport mechanism over the well-established HTTP, WSDL for defining method interfaces in XML, Universal Description, Discovery, and Integration (UDDI, a Web Service discovery mechanism), and DISCO, the Web Service discovery description document.

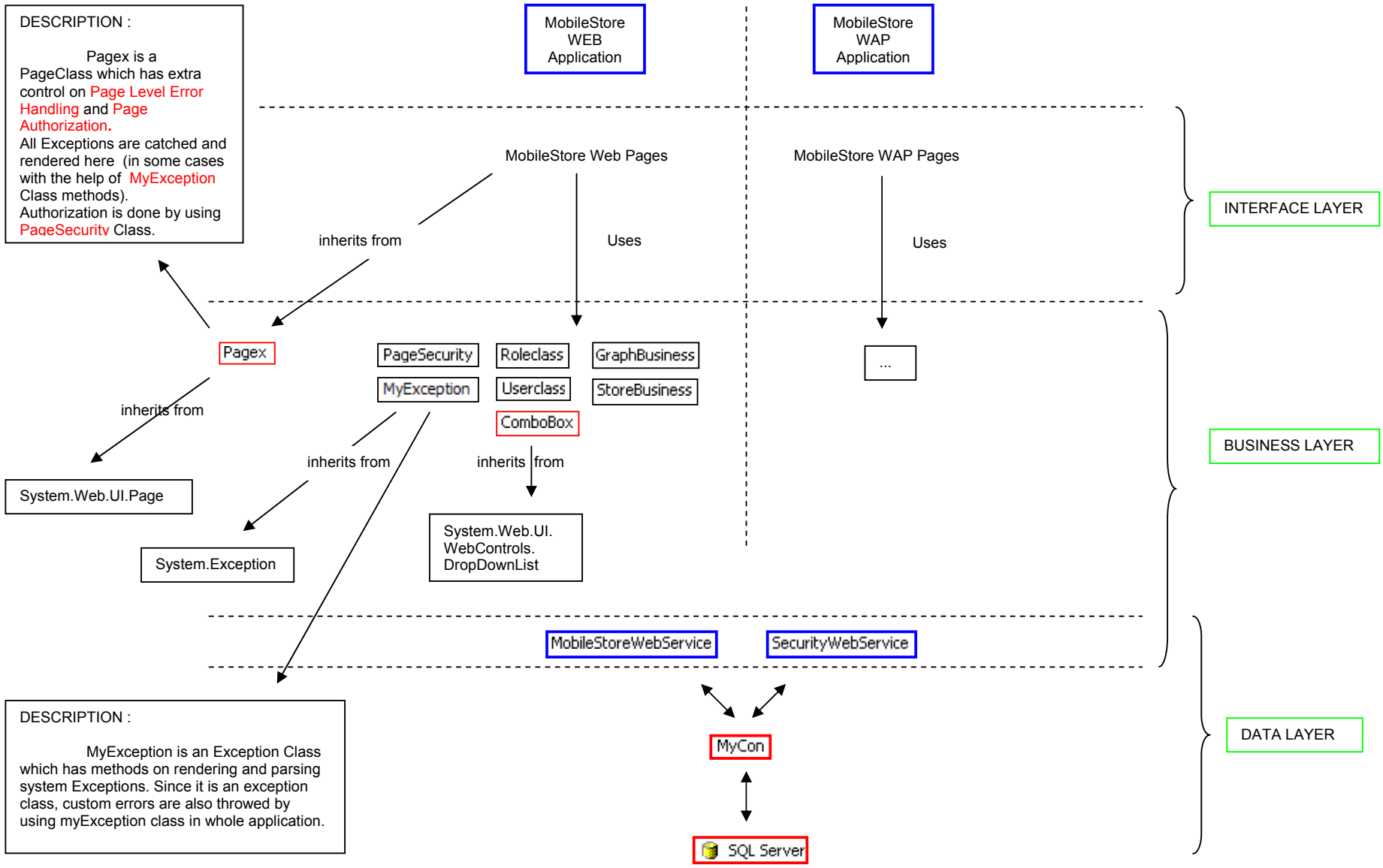
5.11 Advantages Of Web Services Model And .Net For The Project

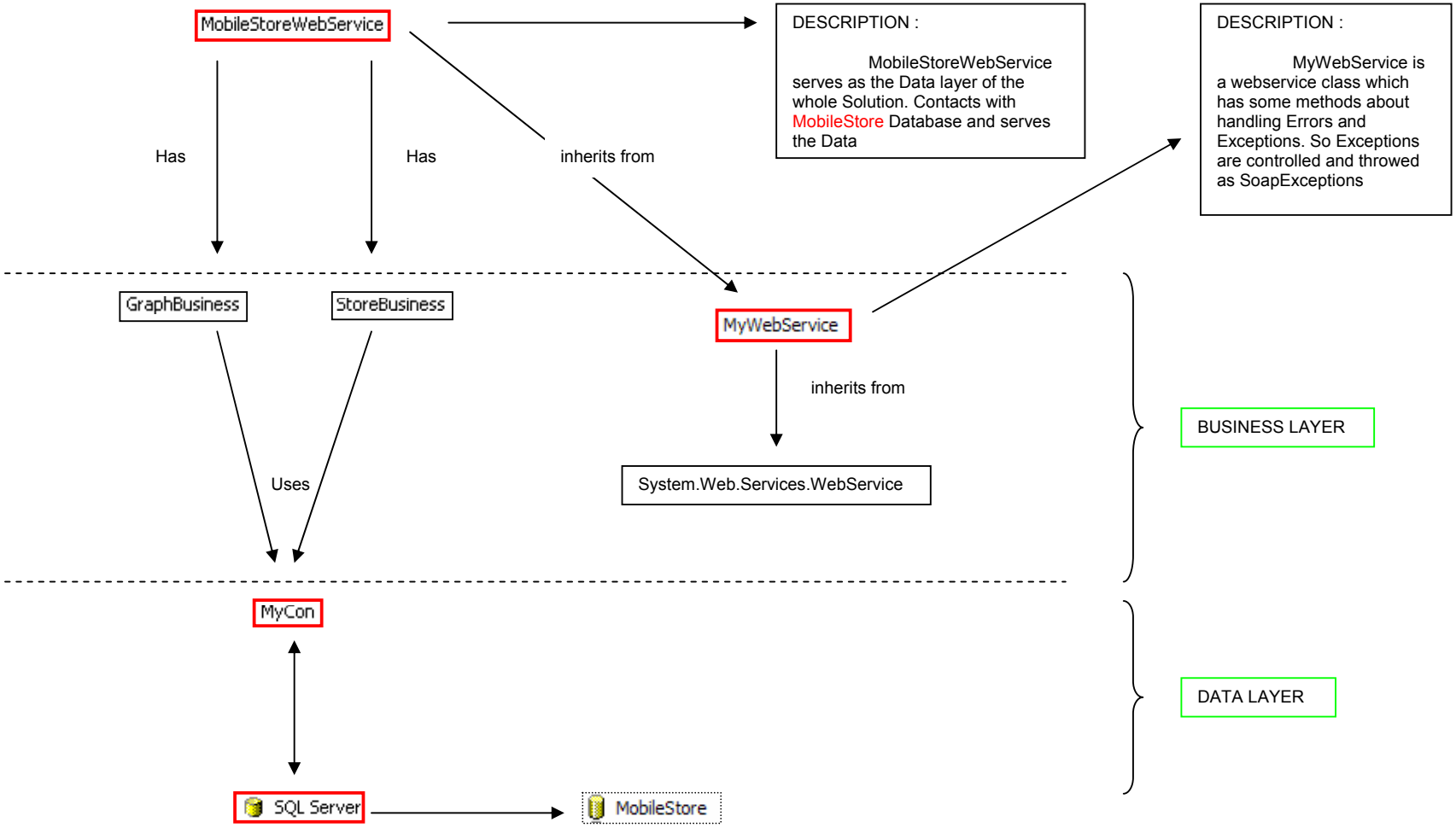
As the purpose of web services, using web services makes the project data public (but only for authorized users). We use same web service method for MobileStore WEB Application, MobileStore WAP Application and also the new existing application "No-Traffic WEB Aplication". Since the data transform is done with XML which is the defacto standard for data interoperability and there are explanations for all methods by WSDL, the Project Data (Market statistics, customer statistics, City traffic status statistics, Carrirer statistics and other application statistics) can be served to any other application by just authorizing it to get the data.

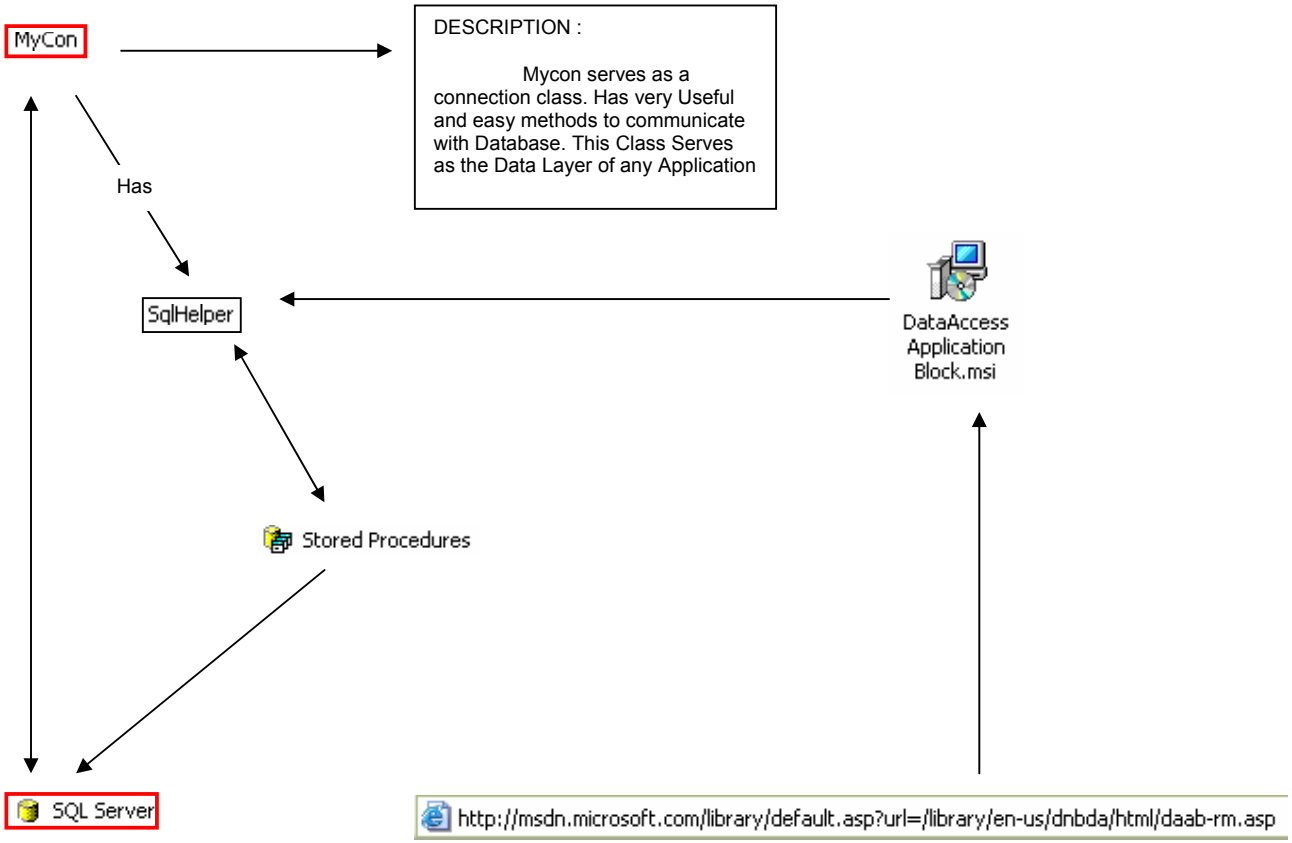
Another advantage of using web services for our project and also for every other projects is "no physical distribution of the software" . So this will cause less cost and easier maintenance.

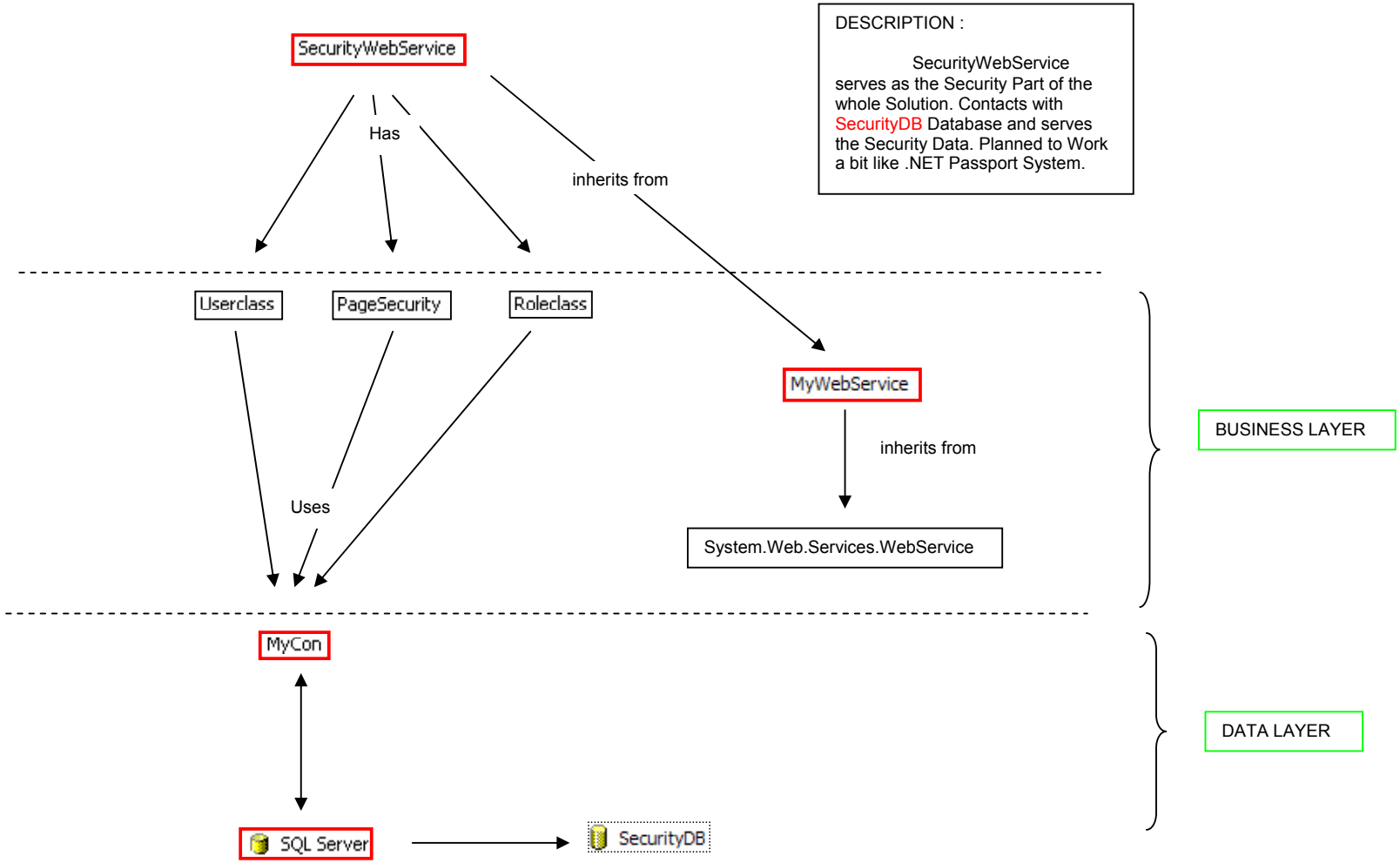
On the other hand, .NET provided us a great development medium. We had the chance to developed a fully object oriented web application and to use numerous classes which decrease the development time and increase development output. We also utilized many advantages of the Visual Studio .NET.

6. Mobile Store SoftWare Architecture


DESCRIPTION :
Pagex is a PageClass which has extra control on Page Level Error Handling and Page Authorization.
All Exceptions are catched and rendered here (in some cases with the help of MyException Class methods).
Authorization is done by using PageSecurity Class.
MobileStore WEB Application
MobileStore WAP Application
MobileStore Web Pages
MobileStore WAP Pages
inherits from
Uses
Uses
INTERFACE LAYER
Pagex
PageSecurity
MyException
Roleclass
Userclass
ComboBox
GraphBusiness
StoreBusiness
...
inherits from
inherits from
inherits from
System.Web.UI.Page
System.Exception
System.Web.UI.
WebControls.
DropDownList
BUSINESS LAYER
MobileStoreWebService
SecurityWebService
DESCRIPTION :
MyException is an Exception Class which has methods on rendering and parsing system Exceptions. Since it is an exception class, custom errors are also throwed by using myException class in whole application.
MyCon
SQL Server
DATA LAYER


MobileStoreWebService
DESCRIPTION :
MobileStoreWebService serves as the Data layer of the whole Solution. Contacts with MobileStore Database and serves the Data
DESCRIPTION :
MyWebService is a webservice class which has some methods about handling Errors and Exceptions. So Exceptions are controlled and throwed as SoapExceptions
Has
Has
inherits from
GraphBusiness
StoreBusiness
MyWebService
inherits from
Uses
System.Web.Services.WebService
BUSINESS LAYER
MyCon
DATA LAYER
SQL Server
MobileStore


MyCon
DESCRIPTION :
Mycon serves as a connection class. Has very Useful and easy methods to communicate with Database. This Class Serves as the Data Layer of any Application
Has
SqlHelper
DataAccess Application Block.msi
Stored Procedures
SQL Server
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/daab-rm.asp


SecurityWebService
Has
inherits from
DESCRIPTION :
SecurityWebService serves as the Security Part of the whole Solution. Contacts with SecurityDB Database and serves the Security Data. Planned to Work a bit like .NET Passport System.
Userclass
PageSecurity
Roleclass
MyWebService
inherits from
Uses
System.Web.Services.WebService
BUSINESS LAYER
MyCon
SQL Server
SecurityDB
DATA LAYER

7 Mobile Store Mobile Side

The mobile part of the Mobile Store Project is designed for the use of project in mobile devices.

7.1 Development Environment

Mobile Internet Designer

The Mobile Internet Designer extends the Microsoft Visual Studio .NET integrated development environment (IDE) to easily build mobile Web applications. Using the designer, you can take advantage of the adaptive rendering, customization, and extensibility features in the Microsoft Mobile Internet Toolkit. It provides the standard Visual Studio IDE design tools: the Toolbox, forms creation, debugging capability, code windows (HTML View and Code View), interactive design (Design View), and more.
To start building your mobile Web application, first open a mobile Web Forms project in Microsoft Visual Basic .NET or Microsoft Visual C# .NET. Next, drag a mobile Web Forms control from the Mobile Web Forms tab in the Visual Studio .NET Toolbox and drop it on the *Design View* panel. You can set the properties and event handlers for the control using the Properties window. Use the standard Visual Studio .NET functionality to build and preview your application. You can also use HTML View to customize the page and Code View to build the application logic within the designer.
Because the Mobile Internet Controls Runtime automatically adapts the rendering of your application to different devices, you build your application by logically grouping controls and arranging them to match the desired user experience. Another difference from desktop design is that you cannot resize controls manually. The runtime handles the resizing of controls when it generates the appropriate markup. To see how the application renders on a specific device, view it on an emulator for the device, or on the actual device.

The Mobile Internet Designer displays an abstract representation of pages. It does not emulate the rendering of any specific device. As you develop applications, the designer provides you with visual cues that indicate the current property settings of mobile controls on the page being viewed. However, this does not mean that the page appears at run time exactly as you see it at design time. The target device may not support every control property. It may support the property, but not the setting you chose. In addition, some properties are provided strictly for extensibility. For example, most of the controls have a BackColor property, but only the Form control currently uses it. The Mobile Internet Toolkit enables you to develop controls that utilize the BackColor property. Developers writing custom device adapters can use this property while rendering controls.

You can optimize the toolkit's automatically generated markup for a specific device through the customization model. Indeed, the Mobile Internet Toolkit provides you with powerful tools that enable you to customize the application's output for specific devices by overriding property values and by creating a specialized rendering based on device capabilities.
The extensibility model of the toolkit enables new device support to be added without requiring that the mobile Web application be modified. You can add support for new devices by updating configuration file settings or by deploying new device adapters. This greatly increases the lifespan of your applications because they continue to work with the latest devices.

7.1.1 Development System Requirements

To develop mobile Web applications with the Mobile Internet Toolkit, your computer must have the following software:

- Microsoft Windows NT 4.0 Workstation with Service Pack 6a (SP6a) or later, as a client for remote development
- Microsoft Windows NT 4.0 Server with SP6a or later, as a client for remote development
- Microsoft Windows 2000 Professional with Service Pack 2.0 (SP2) (including Internet Information Services (IIS))
- Microsoft Windows 2000 Server with SP2
- Microsoft Windows 2000 Advanced Server with SP2
- Microsoft Windows XP Professional
- Microsoft .NET Framework (including ASP.NET)
- Microsoft Mobile Internet Toolkit
- Microsoft Visual Studio .NET (optional)
- Visual Studio .NET is required to use the Mobile Internet Designer that integrates with the Visual Studio .NET developer environment (IDE).
-

You can install the Mobile Internet Toolkit on a server that does not have IIS installed. However, if you install IIS at a later time, the Mobile Internet Toolkit will not work properly. To use the Mobile Internet Toolkit after you install IIS, you must uninstall and then reinstall the Mobile Internet Toolkit.

7.1.2 Deployment System Requirements

To deploy mobile Web applications with the Mobile Internet Toolkit, your computer must have the following software:

- Microsoft .NET Framework (including ASP.NET)
- Microsoft Mobile Internet Toolkit (only the Mobile Internet Controls Runtime component is required)

7.2 Mobile Controls Overview

The Microsoft Mobile Internet Toolkit contains server-side technology that extends ASP.NET to deliver content to a wide variety of mobile devices. These devices include WML and CHTML cell phones, HTML pagers, and personal digital assistants (PDAs) like the Pocket PC.

The Mobile Internet Toolkit contains a complete suite of tools for rapidly building mobile Web applications for wireless devices and for extending the toolkit with your own device-specific adapters.

Users can build mobile Web applications using the graphical interface provided by the Mobile Internet Designer in Visual Studio .NET or can author applications in a text editor using any language supported by the common language runtime.

7.2.1 List Of Mobile Controls

AdRotator
The AdRotator control offers advertisement rotation functionality in the same way that the Web Forms AdRotator control does, but provides adaptive rendering for mobile devices.
Calendar
The Calendar control offers date-picking functionality in the same way that the Web Calendar control does, but appears on mobile devices.
Command
The Command control provides a way to invoke Microsoft ASP.NET event handlers from UI elements.
CompareValidator
The CompareValidator control compares one control to another by using a specified comparison operator.
CustomValidator
The CustomValidator control allows the developer to provide a custom method to validate another control's field.
Form
The Form control is a container for one or more controls within a MobilePage object.
Image
The Image control specifies an image to display on a mobile device.

Label
The Label control creates a text-based control that displays output-only text on a mobile device.
Link
The Link control creates a text-based, output-only control that represents either a hyperlink to another Form control on a mobile page, or an arbitrary URL.
List
The List control renders a list of items to a mobile device.
MobilePage
This is the base class for all mobile ASP.NET pages. As such, the MobilePage control provides the outermost layer of all the containers in a mobile Web Forms control application. It is the only container associated with a URL and primarily contains style and context information common to all controls.
ObjectList
The ObjectList control provides a feature-rich view of a list of data objects.
Panel
The Panel control provides a grouping mechanism for organizing controls. Panel controls can be recursively nested within a form — the Panel control's container. There is no rendering associated with a Panel control.
PhoneCall
The PhoneCall control generates device-dependent interactive UI for automatically calling or displaying telephone numbers.
RangeValidator
The RangeValidator control validates that the values of another control fall within an allowable range, where the minimum and maximum are provided either directly or by reference to another control.
RegularExpressionValidator
The RegularExpressionValidator control validates that the values of another control match a specified expression.
RequiredFieldValidator
The RequiredFieldValidator control validates that user input has been entered in another control.
SelectionList

The SelectionList control provides a UI rendering capability that allows a user to select from a variety of choices.
StyleSheet
The StyleSheet control has no visual representation and is used to organize styles that will be applied to other controls.
TextBox
The TextBox control generates single-line text boxes.
TextView
The TextView control displays large fields of text. Unlike the TextBox control, this control does not support editing.
ValidationSummary
The ValidationSummary control displays a summary of all the validation errors that occurred during the rendering of a form.

7.2.2 Comparing Web Controls and Mobile Controls

Mobile Web Forms are based on Microsoft ASP.NET Web Forms. The Microsoft Mobile Internet Toolkit provides a flexible toolset that enables you to create content sites and Web applications intended for a wide variety of mobile devices. You can take advantage of the adaptive rendering of the mobile Web Forms controls while having the flexibility to customize the display for specific devices or types of devices, such as a handheld computer or a mobile phone.

The following table provides a side-by-side comparison of the controls used for Web Forms controls and mobile controls.

Web Forms control	Mobile control	Comments or differences
AdRotator	AdRotator	Similar functionality. Mobile control adds ImageKey and NavigateUrlKey properties.
Button, ImageButton, LinkButton	Command	Mobile control combines the functionality of the Web Forms Button, ImageButton, and LinkButton controls.
Calendar	Calendar	Similar functionality. Mobile control does not provide HTML-specific properties directly but exposes an underlying Web Forms Calendar control through the WebCalendar property.
[no equivalent control]	PhoneCall	Used to actively drop the data line and initiate the call on dial-capable devices. This is similar to the use of the mailto scheme for electronic mail

		addresses, which starts your e-mail client.
CompareValidator	CompareValidator	Validation is identical.
CustomValidator	CustomValidator	Validation is identical.
DataList, Repeater	List	Similar functionality. Mobile control can apply templates on a per-device basis.
DataGrid	ObjectList	Similar functionality. The ObjectList control provides multiple views to show the data collections
[no equivalent control]	DeviceSpecific	Used to enable property overrides and templates for mobile Web Forms controls.
[no equivalent control]	Form	Similar to a page in a Web Forms control. Mobile Web Forms pages can contain multiple Form controls.
Image	Image	Similar functionality. Mobile control can select an image from a set of device-specific images.
Label	Label	Same functionality.
Hyperlink	Link	The runtime cannot render the mobile control as an image. Use the Image control to create an image link (by specifying the NavigateUrl property on the Image control).
Panel	Panel	Mobile panels can be used to provide device-specific rendering by using the ContentTemplate device templates to replace the panels.
RangeValidator	RangeValidator	Validation is identical.
RegularExpressionValidator	RegularExpressionValidator	Validation is identical.
RequiredFieldValidator	RequiredFieldValidator	Validation is identical.
CheckBox, CheckBoxList, DropDown, DropDownList, ListBox, RadioButton, RadioButtionList	SelectionList	Mobile control combines the functionality of the corresponding Web Forms controls. Use the SelectType property (and the associated ListSelectType

		enumeration) to define the type of selection list button to render. For example, the Mobile control Checkbox SelectType corresponds to the Web Forms control CheckBox and CheckBoxList; Dropdown is the same as DropDown and DropDownList. Use the Rows property to specify the number of items shown in the list when the SelectType property is the ListBox or MultiSelectListBox control.
[no equivalent control]	StyleSheet	Web Forms use cascading style sheets rather than StyleSheet controls.
Table	[no equivalent control]	Use the List, ObjectList, and SelectionList mobile controls
TextBox	TextBox	Similar functionality. Mobile control does not provide automatic postback, read-only, or multiline functionality.
[no equivalent control]	TextView	Used to display large blocks of text. Supports basic text formatting.
ValidationSummary	ValidationSummary	Same functionality. Mobile control shows error messages of validators on a particular form (through the FormToValidate property).

7.3 Mobile Store Mobile Forms

Mobile Store Project contains two kinds of forms. These are Customer and Carrier forms. When customers or carriers enter the Mobile Store mobile page via mobile phones or pocket PCs, they firstly encounter the login form. If they are successfully authorized then system recognize which form is loaded according to their authentication information. If user has a carrier username and password then system redirects the user to Carrier Forms else the user is redirected to Customer Forms.

7.3.1 Customer Forms

These forms were designed for customers to enable them to make shopping and to view their order statuses via their mobile phones.

7.3.1.1 Form_select

This form is the first form after login page and customers have two options in use of the Form_select. First option is viewing status of orders given before. Second option is making new order.

7.3.1.2 Form_orderstatus

This is the place where customers can get information about their orders. Form_orderstatus reflects the order status that is updated by the carrier who is delivering the order.

7.3.1.3 Form_main

This form displays the main shopping options to customers. Customers continue to shopping selecting a main category in this form.

7.3.1.4 Form_category

Form_category contains the sub options according to selected option in previous form. Customers select a main category in this form.

7.3.1.5 Form_subcategory

This form exhibits the sub categories of selected category in the form of Form_category.

7.3.1.6 Form_products

This form displays the products placed in selected sub category in previous form.

7.3.1.7 Form_productdetails

This form gives information related to price of product and the stores selling the product. Customers can learn where the product is sold and price of the product in the form of Form_productdetails.

7.3.1.8 Form_quantity

Form_quantity enables the customers to add the product selected into basket. In addition, customers can add the products as much as they want up to 9.

7.3.1.9 Form_end

This form offers three options to customers. The first option is submitting the selected products. After submission, system informs the carrier and carrier starts shopping if he is not hold by another order. The second option is continuing shopping. When customer selects this option, system keeps the products bought by customer in his shopping basket and redirects the customer to Form_main. Thus, the customer can continue buying new products. The third option is removing a product from shopping basket. If customer selects this option then system redirects the customer to Form_product_delete.

7.3.1.10 Form_product_delete

In this form, customer can remove any product entering its list order.

7.3.1.11 Form_thanks

This is the last form in the customer forms. Its aim is showing the gratefulness of company and declaring the end of shopping.

7.3.2 Carrier Forms

These forms were designed for the carriers to enable them to take the orders from customers, to update the order and road statuses.

7.3.2.1 Form_intro

This form offers four options to carriers. The first option is inquiring the system to learn whether there is a new order waiting for carrier or not.
The second option is reading the orders. He third option is updating the road status. The last option is inform the system about the order was completed.

7.3.2.2 Form_location

Thanks to this form, carriers send their location information to the system and they get their route information from the system.

7.3.2.3 Form_route

The route information is displayed in this form

7.3.2.4 Form_roadstatus

Carriers can update the road statuses entering road id in this form.

7.4 Mobile .NET Terminology

This table defines only terms that are specific to the Mobile Internet Toolkit or Web-based languages.

Term	Definition
Card	In WML, a Web page is called a card. WML devices can either display the contents of a card on a single screen or, when necessary, provide scroll bars so that the entire contents of the card can be viewed. Developers need not worry about manipulating cards or decks (groups of cards) because the Microsoft Mobile Internet Toolkit handles formatting, including pagination, for targeted devices.
CHTML	Compact HTML is a mark-up language used on some cell phones. CHTML is a subset of HTML tags with additional tags added to enhance mobile functionality.
Code-behind class	A page class contained in a code-behind file that implements the program logic of a Web Forms or mobile Web Forms application.
Code-behind file	A code file containing the page class that implements the program logic of a Web Forms or mobile Web Forms application. For more information about using a code-behind file to develop your applications, see Developing User Controls in a Code-Behind File.
Code-behind page	See *code-behind file*.
Comparison evaluator	A filter that compares a device capability name to a value.
Composite control	A custom server control that consists of a custom collection of other server controls as child controls.
Container control	A type of mobile Web Forms control that contains other controls and is used to provide visual groupings of controls and content.
Control template	A template associated with a control. See *templated control.*
Data bind	The association of a data source with a server control.
Deck	A group of one or more cards.
Default unit system	The default unit system is based on one line equaling 100 units, as indicated by the DefaultWeight field in the ControlPager class.
Delegated evaluator	A filter that uses a custom method to evaluate the specified data. See also evaluator delegate filter.
Device adapter	A Mobile Internet Controls Runtime class that adapts the behavior of mobile pages and controls based on the target device.
Device capabilities	The set of device functionality available through the HasCapability method or the <Choice> element.
Device definition	The characteristics of a device available through the MobileCapabilities class and the DeviceSpecific control.
Device filter	Provides a named construct to a comparative evaluation of MobileCapabilitites properties or to a delegate evaluation method that utilizes the MobileCapabilitites object.
End tag	A markup language tag that closes an element: </>. An end tag follows the syntax </*Name*>, where *Name* matches the element name declared in the start tag.

Equality comparison filter	A device filter that compares a device capability to a literal value.
Evaluator delegate filter	A device filter that calls a static method of a class to perform an evaluation.
External style	A style in an external style sheet.
External style sheet	A style sheet defined in a user control in an ascx file.
Microsoft .NET Framework	Common infrastructure designed to assist development, such as those provided by Microsoft ASP.NET page framework and the Microsoft Mobile Internet toolkit.
Internal style	A style in an internal style sheet.
Internal style sheet	A style sheet contained in a mobile Web Forms page.
Mobile Internet Controls Runtime	An extension of the Microsoft ASP.NET runtime environment that supports the creation of mobile Web applications for wireless devices.
Mobile Internet Designer	An extension to the Microsoft Visual Studio .NET Integrated Development Environment (IDE) that provides an environment in which to create mobile Web applications for wireless devices.
Mobile user control	A mobile Web Forms control derived from the System.Web.UI.MobileControls.MobileUserControl class. User controls provide containers for custom controls built from other mobile Web Forms controls.
Mobile Web Forms	Extensions added to Microsoft ASP.NET Web Forms that target mobile devices from cell phones to Pocket PCs.
Mobile Web Forms controls	See Mobile Internet Controls Runtime.
Pagination	A mechanism that automatically separates the content in mobile Web forms into smaller groups of rendered pages that are targeted to fit a specific device. It also renders UI elements that a user can use to browse to other pages.
Postback	When a Web page sends data back to the server to access the same page.
Private view state	State information that is written out as a hidden field, such as the form that is currently active or the pagination information for a form.
Property bag	A category of properties in the Properties window. For example, the Appearance property bag contains such properties as Alignment and BackColor.
Property editor	A dialog box the Mobile Internet Designer in which developers can set a control's properties.
Start tag	The opening tag that begins an element. The general syntax for a start-tag is <Name *attributes*>, where *Name* is the name of the element being defined and *attributes* is a set of name-value pairs. All start tags in XML must either have end-tags or use the empty element syntax, <name *attributes*/>
Tag	A component of markup used to delineate element beginnings and endings. For example, the tag, <A> is the start tag for the A element, </A> is the end-tag for the A element, and <B/> is an empty tag representing the B element.
Template	A markup language construct associated with controls and created with the <DeviceSpecific> and <Choice> tags. It is

	used to customize output for specific types of hardware devices
Templated control	A control that supports one or more templates. A single templated control can refer to multiple sets of templates, where each template set is defined through device-specific criteria.
Template set	A collection of templates associated with a templated control.
Text writer	A mechanism that allows device adapters to write their output through an object. A text writer object is instantiated from the TextWriter base class.
Uniform Resource Identifier (URI)	A number or name that uniquely identifies an element or attribute. URIs includes both Uniform Resource Names (URNs) and Uniform Resource Locators (URLs). URIs are a more-general scheme for locating resources on the Internet that focuses more on the resource and less on the location. In theory, a URI could find the closest copy of a mirrored document or locate a document moved from one site to another. When discussing XML today, URIs are URLs in nearly all cases, although it is expected that URNs will become more common in the future.
WAP	Wireless Application Protocol, a group of standards for wireless devices proposed by the WAP Forum. A standard protocol for providing Internet communications and advanced telephony services on phones, pagers, PDAs, and other wireless terminals.
WML	Wireless Markup Language, a markup language designed to specify user-interfaces on wireless phones. WML is part of WAP. An existing, XML-based markup language, intended for use in specifying content and the user interface for narrowband devices, including cellular phones and pagers.
Wrap	To render text and graphics so the user does not have to scroll horizontally (or that the text and graphics is not truncated at the right margin).

8 Mobile Store Simulator

Mobile Store Simulator is designed for displaying the activity of carriers. It was developed in .NET environment as an html page and it contains two kinds of objects as the main parts of simulator. These are Input and Textarea objects. Input objects were used as the roads of the simulator. Textarea objects were used as the corner identifiers of the simulator.

8.1 Use of Web Services in Simulator

The connection between simulator and system was achieved with use of XML Web Services since use of web services does not require full-page refresh for web pages. Thus, each action of carriers can be displayed in simulator very quickly. Otherwise, if simulator web page was posted back to show each change in location of carriers, simulator web page could not be viewed in runtime because of high volume of data transferred between server and client. To use a XML Web Service in client-side script (simulator web page) a behavior object must be defined inside the Java Script code in the Simulator Web page.

8.2 Web Service Behavior

The Web Service behavior enables client-side script to invoke remote methods exposed by Web Services, or other Web servers, that support the Simple Object Access Protocol (SOAP) and Web Services Description Language (WSDL) 1.1. The Web Service behavior uses the SOAP protocol to communicate with Web Services, yet its purpose is to provide a simple way to take advantage of this protocol without requiring expert knowledge of SOAP. The Web Service behavior supports the use of a wide variety of data types, including intrinsic SOAP data types, arrays, objects, and Extensible Markup Language (XML) data. The Web Service behavior is implemented with an HTML Component (HTC) file as an attached behavior, so it can be used in Microsoft® Internet Explorer 5 and later versions.

8.3 What the Web Service Behavior Does

The Web Service behavior enables a method call to be made to a Web Service using a simple scripted syntax, as shown in the following snippet.

```
iCallID = myService.MyMath.callService("add", int1,int2);
```

To invoke a method on a Web Service, the author first attaches the Web Service.HTC file to any element in the page. Once the behavior is attached, the Web Service behavior enables either synchronous or asynchronous calls to be made to Web Services from client-side script. The asynchronous nature of a remote method invocation means that there is a delay between the execution of the method and the arrival of its returned result. Using the synchronous mode of processing means that the client script processing halts until the callService method has completed. The asynchronous mode of method invocation is the default mode of the Web Service behavior.

Note Synchronous calls to remote services lock the user interface while the call is pending and, therefore, aren't practical for browser-based applications.

The Web Service behavior handles the process of calling the method and receiving the raw XML data packets from the Web Service. The user has the option of using either an event handler or a callback handler function to process the results. If an event handler is used, the Web Service behavior fires the onresult event, which occurs when the Web Service receives the response from a method call. Alternatively, if a callback function is used to process results, a result object is passed directly as an input parameter to the callback function.

Any client script using the Web Service behavior should always test the error property to determine if the method invocation was successful. When an error is

encountered, an errorDetail object is also exposed; this object has properties that can be evaluated to help identify the source of the error. The different techniques for handling returned results from method calls are described in using the Web Service Behavior.

The Web Service behavior cannot directly invoke a method on a Web Service that is hosted in a different domain from the machine hosting the Web page. Nevertheless, a Web Service running on the Web server hosting the Web page can be configured to act as a proxy for other remote Web Services. For more information, see Calling Methods on Remote Servers.

8.4 Benefits of Web Service Behavior

The primary benefit of the Web Service behavior is that it provides a simple way for you to call methods that are exposed by Web Services using the SOAP protocol. The Web Service behavior enables you to call a remote method using a few, straightforward, client-side scripting methods exposed by the behavior. Navigating to another URL is unnecessary when using the Web Service behavior to deliver data to a page because DHTML is used to update the page's content. This approach enhances the browsing experience significantly, compared to traditional browsing approaches that require a full-page refresh.

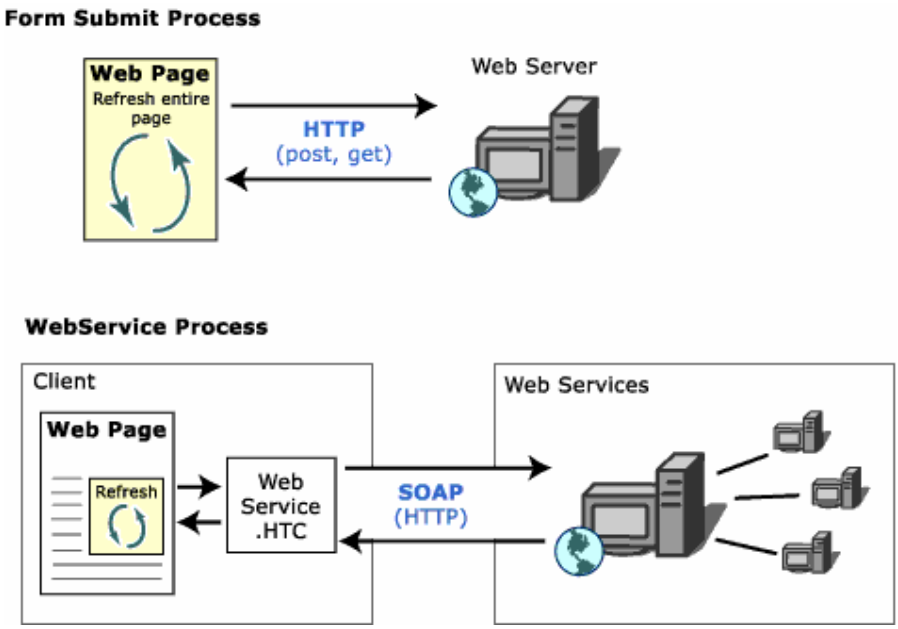
The Web Service behavior is implemented as an attached behavior, as opposed to an element behavior, and, therefore, can be used in Internet Explorer 5 and later versions. The Web Service behavior is a reusable component, so its encapsulated capabilities help reduce the need to duplicate code, thus improving the overall organization of the client-side script and the Web application. All protocol-specific code, and most of the code required to handle the data transactions, is encapsulated from the client-side script in the main Web page, which is a general benefit of using DHTML behaviors. You only need to attach the Web Service behavior once in order to invoke methods from one or more different Web Services.

This behavior enables Internet Explorer 5 users to take advantage of some of the latest cross-platform programming techniques. Web Services can reside anywhere on the Internet and encapsulate building blocks of capability, which can be assembled, packaged, or presented in various ways in a Web page. Web Services site provides access to a variety of tools and resources for designing and using Web Services. Using such a distributed architecture offers improved scalability because data- or CPU-intensive tasks can be organized into dedicated Web Services, freeing the client from unnecessary burden. Therefore, the Web Service behavior can help enhance the client browser experience and improve the overall organization of the Web application.

The Web Service behavior provides a more streamlined approach to the problem of delivering information from the Web server to Internet Explorer 5 and later. Using the Web Service behavior to access Web Services simplifies things on the client side, making the use of Web Services more appealing. The behavior can be updated and adapted as the SOAP standard evolves, without requiring major changes to client-side script in the main Web page.

8.5 Comparing the Web Service Approach to Forms

To help explain why the Web Service behavior is so useful, it's worth comparing the Web Service behavior technique with the approach commonly used to deliver data-driven Web sites today. The following diagram shows the basic process used by each technique.



The first part of the illustration shows how a Web page containing a form commonly uses either the get or post method through HTTP to update a Web page. Each time a form is submitted, the client navigates to a new URL, after which the browser downloads and renders the entire page. This method is widely used today but is inefficient because the Web page refreshes and re-renders an entire page of content, even if only a small percentage of the page has actually changed. Web surfers commonly encounter this behavior when browsing e-commerce and data-driven pages.

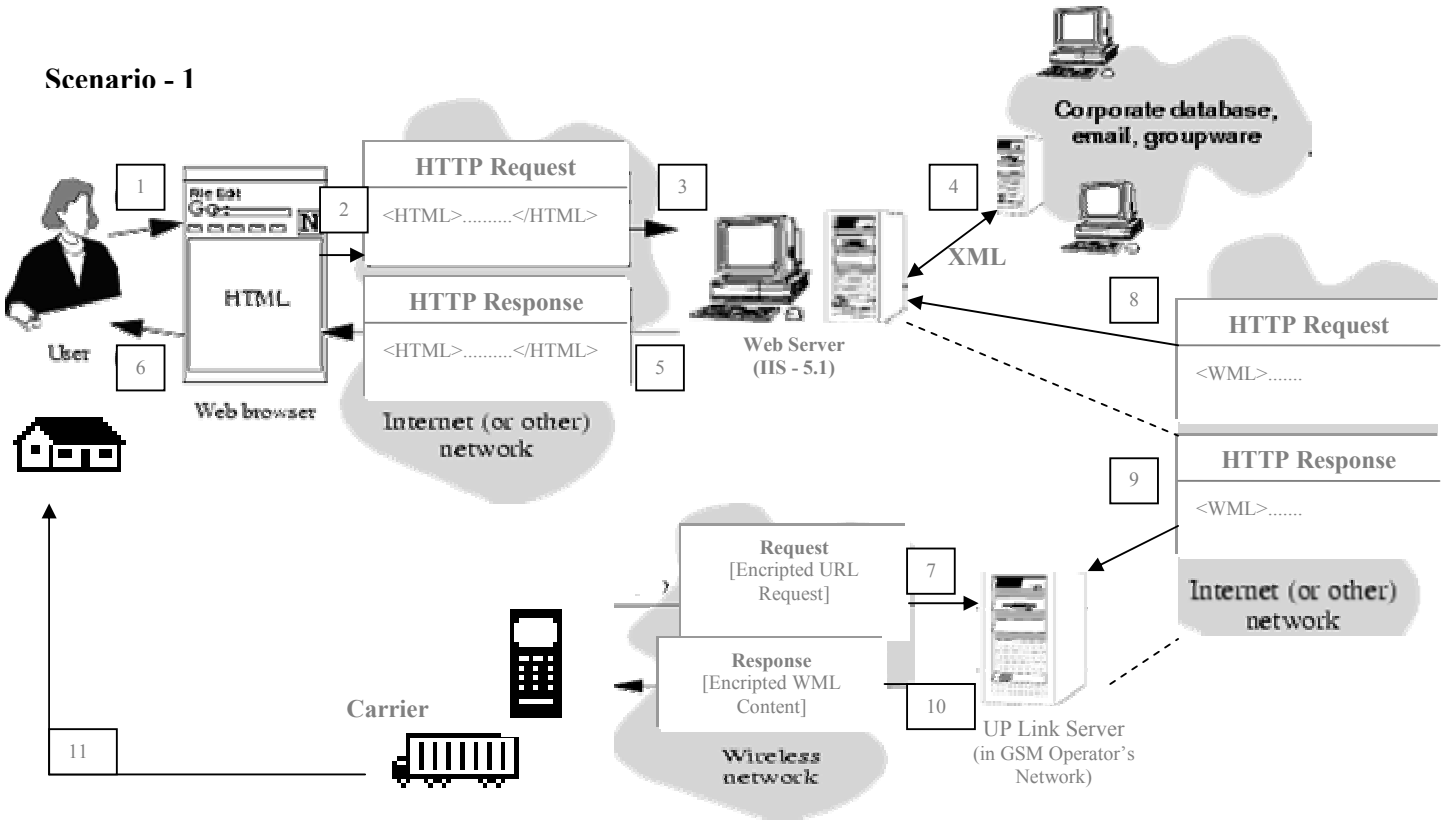
When there's a need to browse numerous items and pages, such as when searching a catalog or search engine, the delays and waste of resources can be significant.

The second part of the diagram illustrates how a Web page can use the Web Service behavior to avoid the drawbacks associated with the traditional form submit approach. The Web Service behavior receives method calls from the client-side script and sends a request to the Web Service. The results are returned to the client script, and processing continues. The Web page can then use the information in whatever context is required, such as updating some portion of page rendering using DHTML.

A key feature of the Web Service behavior is that it enables client-side script to access a Web Service without requiring navigation to another URL. Using the Web Service behavior approach, the portions of the page that are indicated by the user's inputs can be dynamically updated using DHTML, providing a significant improvement in the browsing experience.

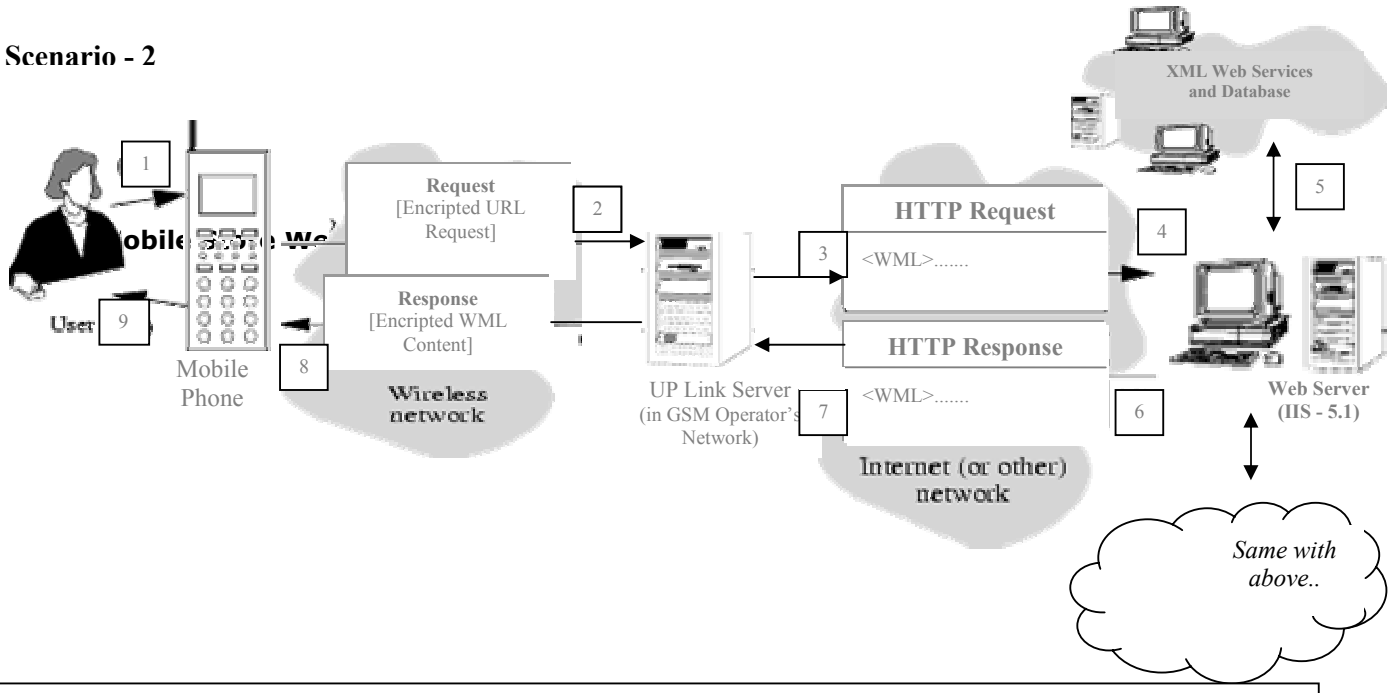
9 Mobile Store Sample User Scenarios

Scenario - 1



In Scenario-1,
-user gives an order using his/her web browser
-web server determines the appropriate carrier and other required information for carrier to transmit the order
-carrier always sends requests to web server to learn whether there is an order or not
-if there is an order given by a customer, web server informs the carrier via wap
-carrier prepares the order and then takes it to customer's address

Scenario - 2



In Scenario-2,
-user gives an order using his/her Mobile Phone
-web server determines the appropriate carrier and other required information for carrier to transmit the order
-carrier always sends requests to web server to learn whether there is an order or not
-if there is an order given by a customer, web server informs the carrier via wap
-carrier prepares the order and then takes it to customer's address

10. Mobile Store Solution Database Architecture

10.1 Solution Database Overview

Microsoft SQL Server 2000 is used as database server for the project. There are 2 databases for Solution.

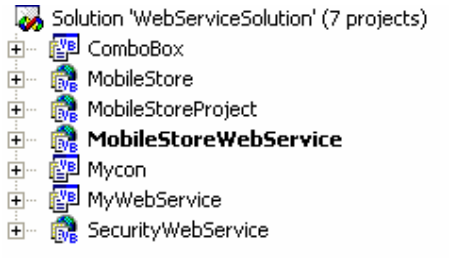
One of Them is **MobileStore** Database. MobileStore Database Keeps Business Data And Has Functions As Stored Procedures To Apply Business Logic. MobileStore Database şs used by **MobileStore Web Service**.

Other One is **SecurityDB** Database, which keeps Web Application Security Data, User Information and User Rights for the project. SecurityDB Databse is used by **Security Web Service.**

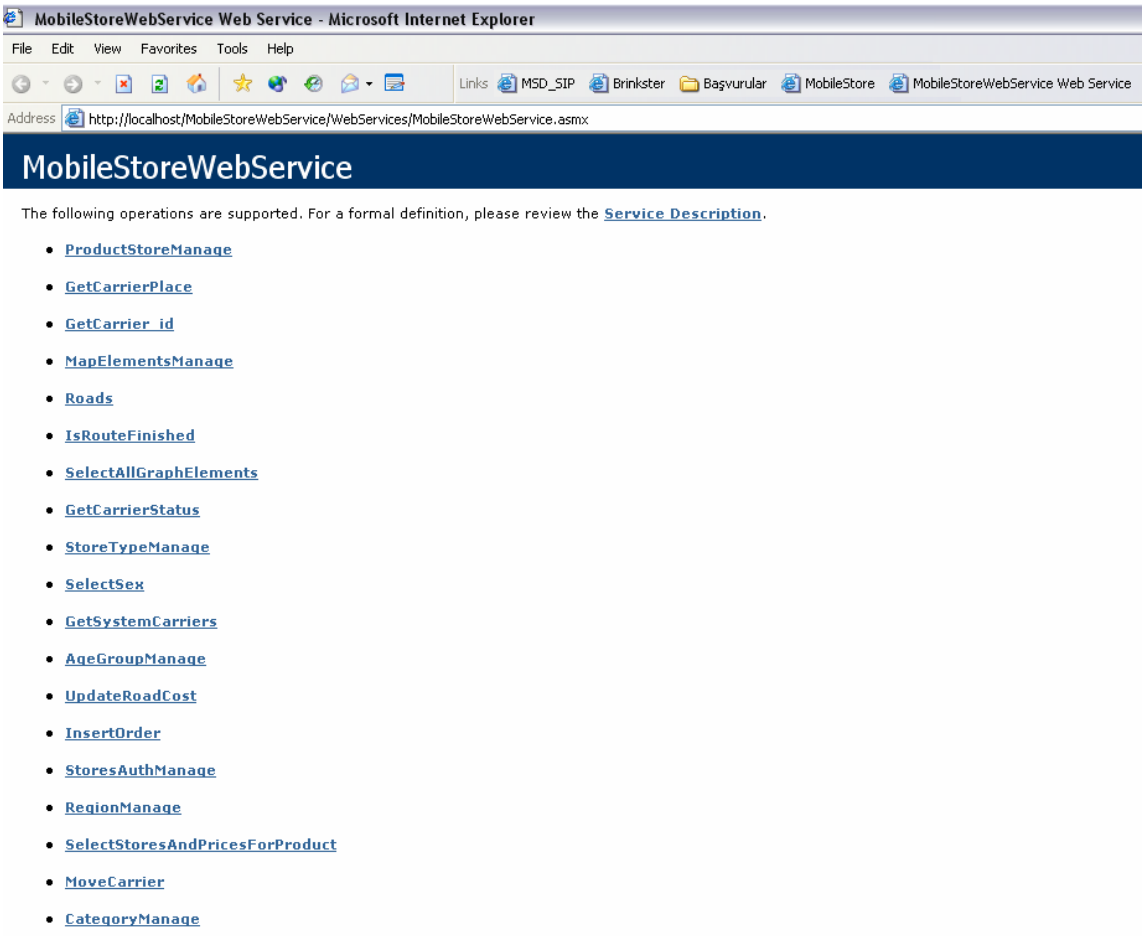
10.2 SecurityDB Database Architecture

10.3 MobileStore Database Architecture

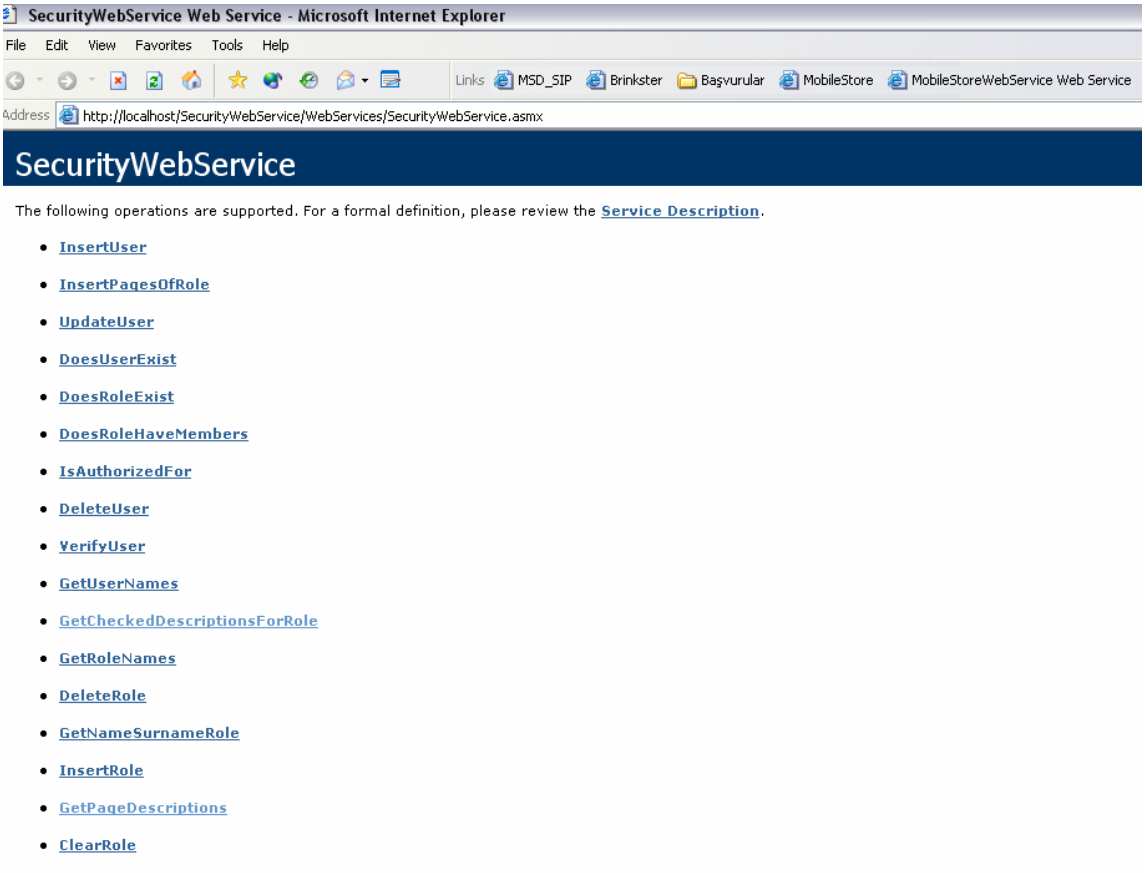
11. MOBILE STORE SYSTEM USAGE AND SCREENSHOTS



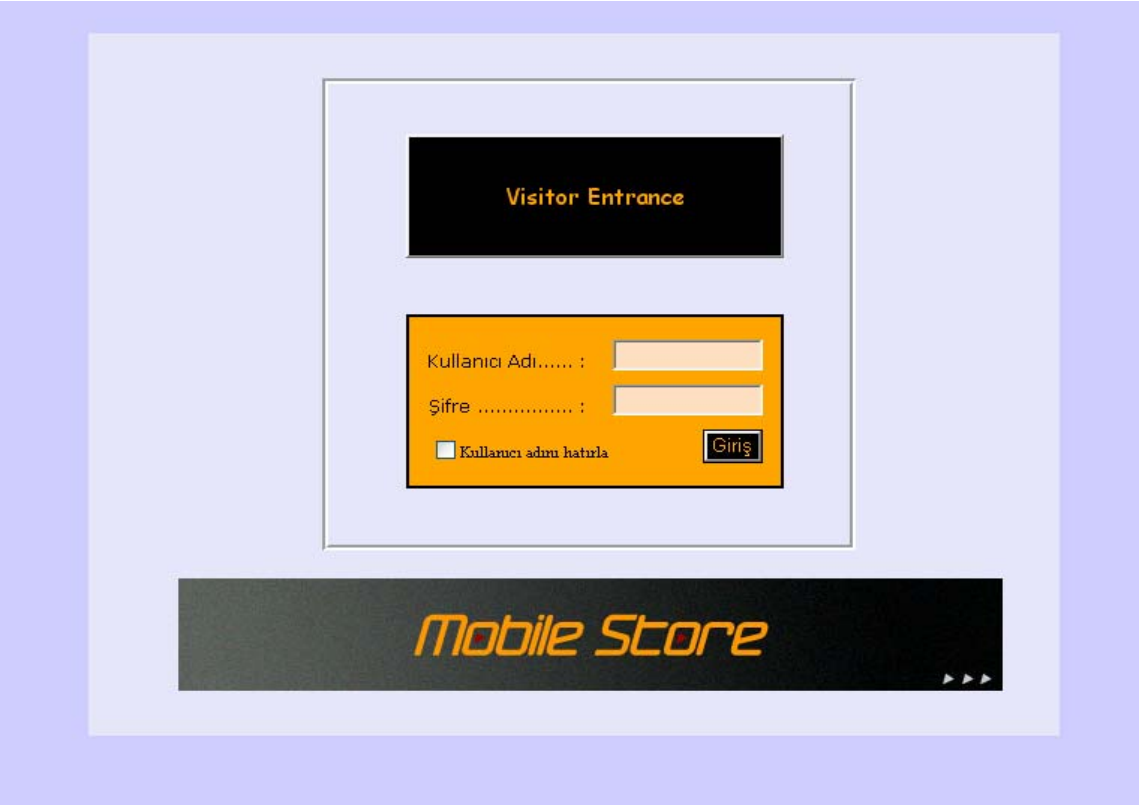
MobileStoreWebService



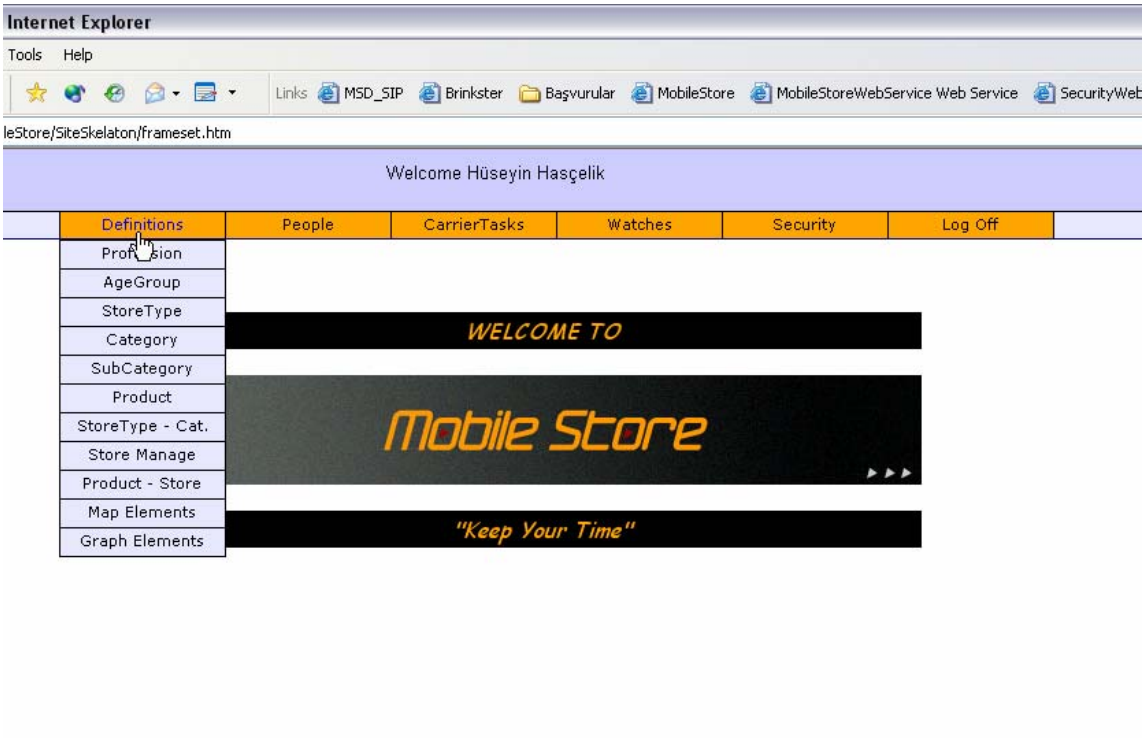
Security Web Service :



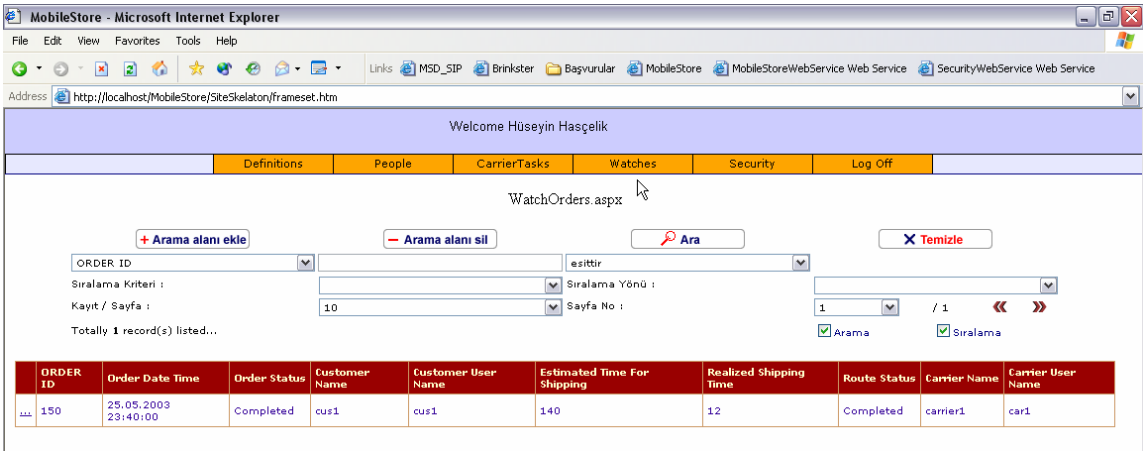
Interface For Internet Users :



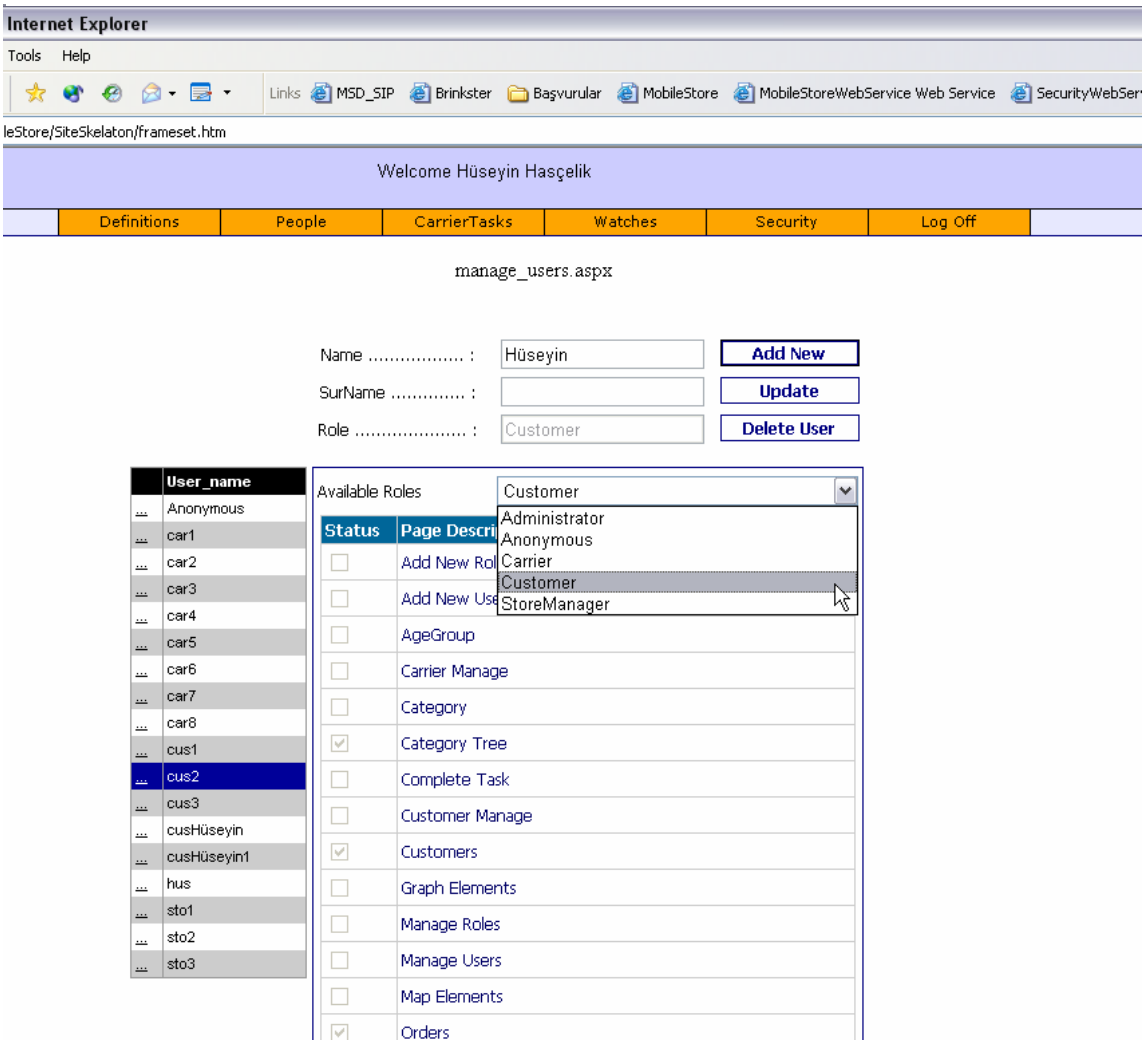
Make Definitions To Customize Mobile Store System...



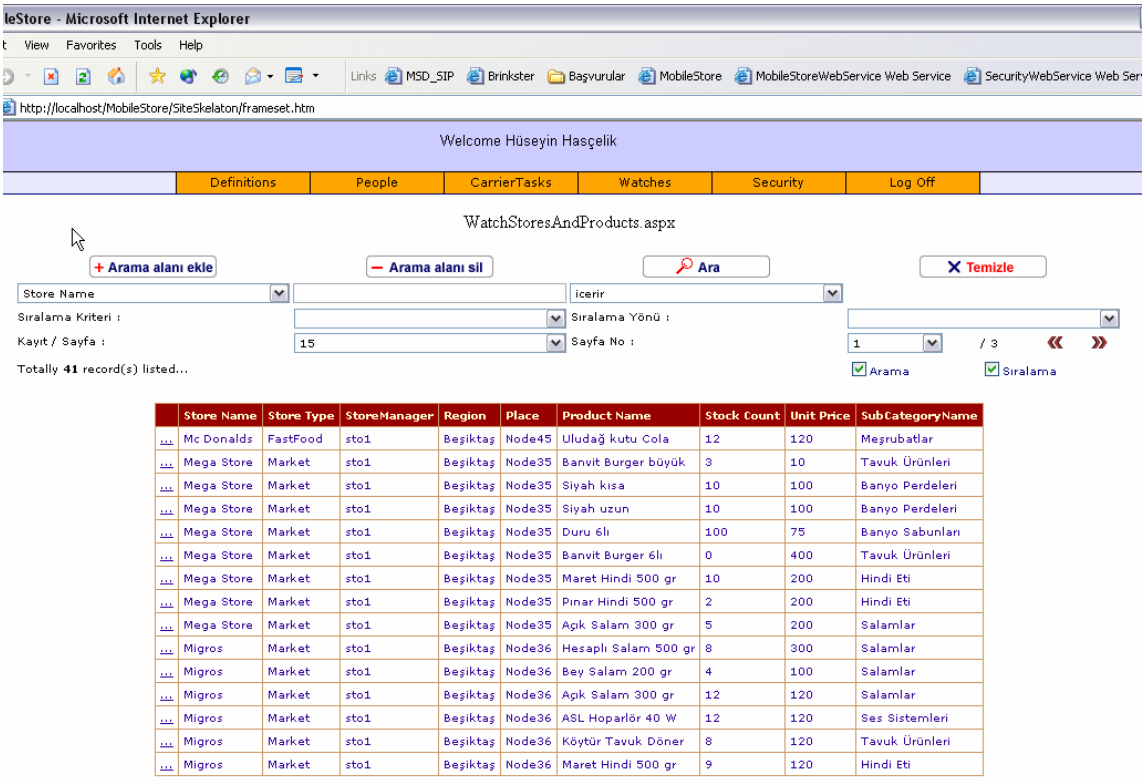
Watch Orders...



Manage Users...

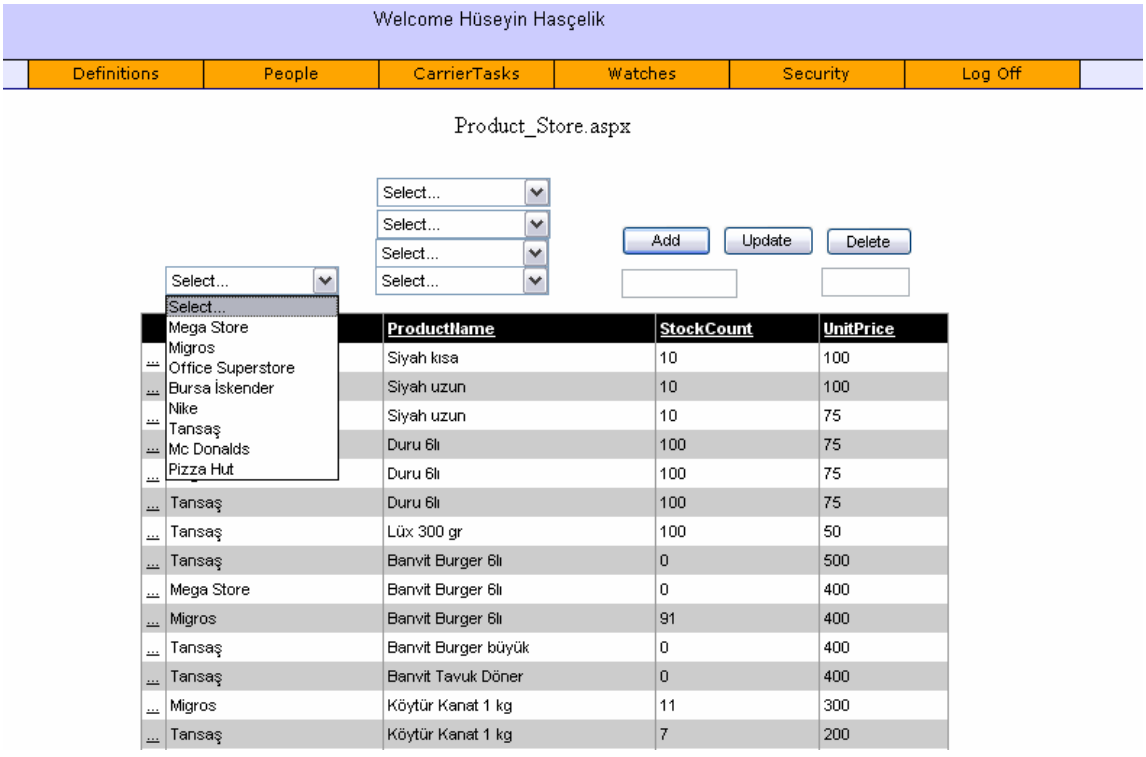


Watch Stores And Products...

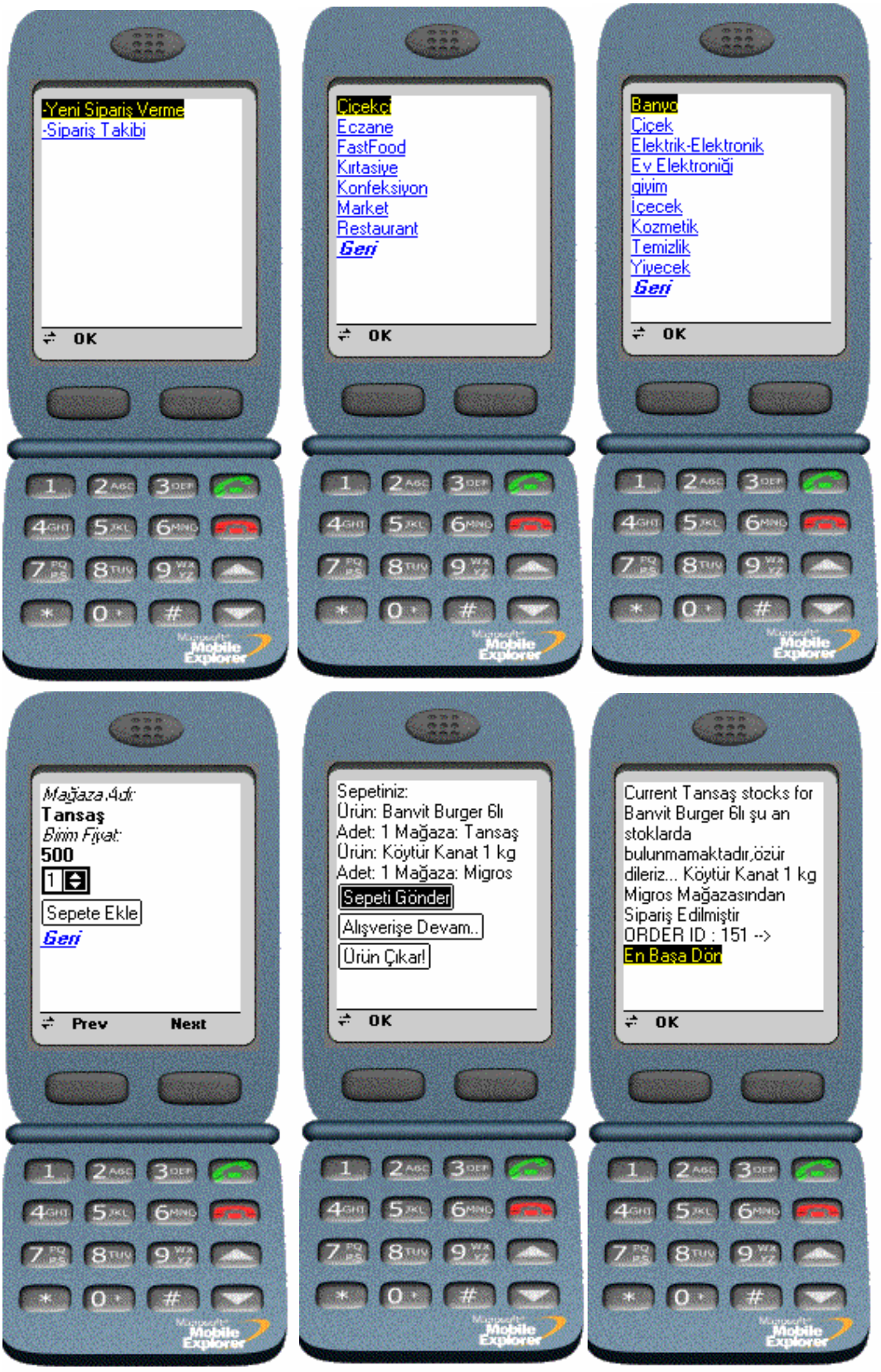


	Store Name	Store Type	StoreManager	Region	Place	Product Name	Stock Count	Unit Price	SubCategoryName
...	Mc Donalds	FastFood	sto1	Beşiktaş	Node45	Uludağ kutu Cola	12	120	Meşrubatlar
...	Mega Store	Market	sto1	Beşiktaş	Node35	Banvit Burger büyük	3	10	Tavuk Ürünleri
...	Mega Store	Market	sto1	Beşiktaş	Node35	Siyah kısa	10	100	Banyo Perdeleri
...	Mega Store	Market	sto1	Beşiktaş	Node35	Siyah uzun	10	100	Banyo Perdeleri
...	Mega Store	Market	sto1	Beşiktaş	Node35	Duru 6lı	100	75	Banyo Sabunları
...	Mega Store	Market	sto1	Beşiktaş	Node35	Banvit Burger 6lı	0	400	Tavuk Ürünleri
...	Mega Store	Market	sto1	Beşiktaş	Node35	Maret Hindi 500 gr	10	200	Hindi Eti
...	Mega Store	Market	sto1	Beşiktaş	Node35	Pınar Hindi 500 gr	2	200	Hindi Eti
...	Mega Store	Market	sto1	Beşiktaş	Node35	Açık Salam 300 gr	5	200	Salamlar
...	Migros	Market	sto1	Beşiktaş	Node36	Hesaplı Salam 500 gr	8	300	Salamlar
...	Migros	Market	sto1	Beşiktaş	Node36	Bey Salam 200 gr	4	100	Salamlar
...	Migros	Market	sto1	Beşiktaş	Node36	Açık Salam 300 gr	12	120	Salamlar
...	Migros	Market	sto1	Beşiktaş	Node36	ASL Hoparlör 40 W	12	120	Ses Sistemleri
...	Migros	Market	sto1	Beşiktaş	Node36	Köytür Tavuk Döner	8	120	Tavuk Ürünleri
...	Migros	Market	sto1	Beşiktaş	Node36	Maret Hindi 500 gr	9	120	Hindi Eti

Manage Store And Their Products...



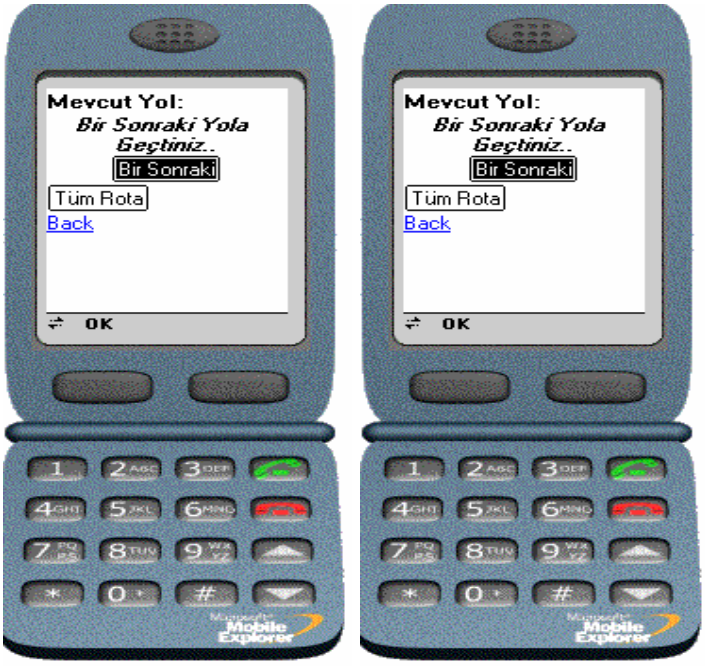
Wap Interface For Customers From Cell Phones...

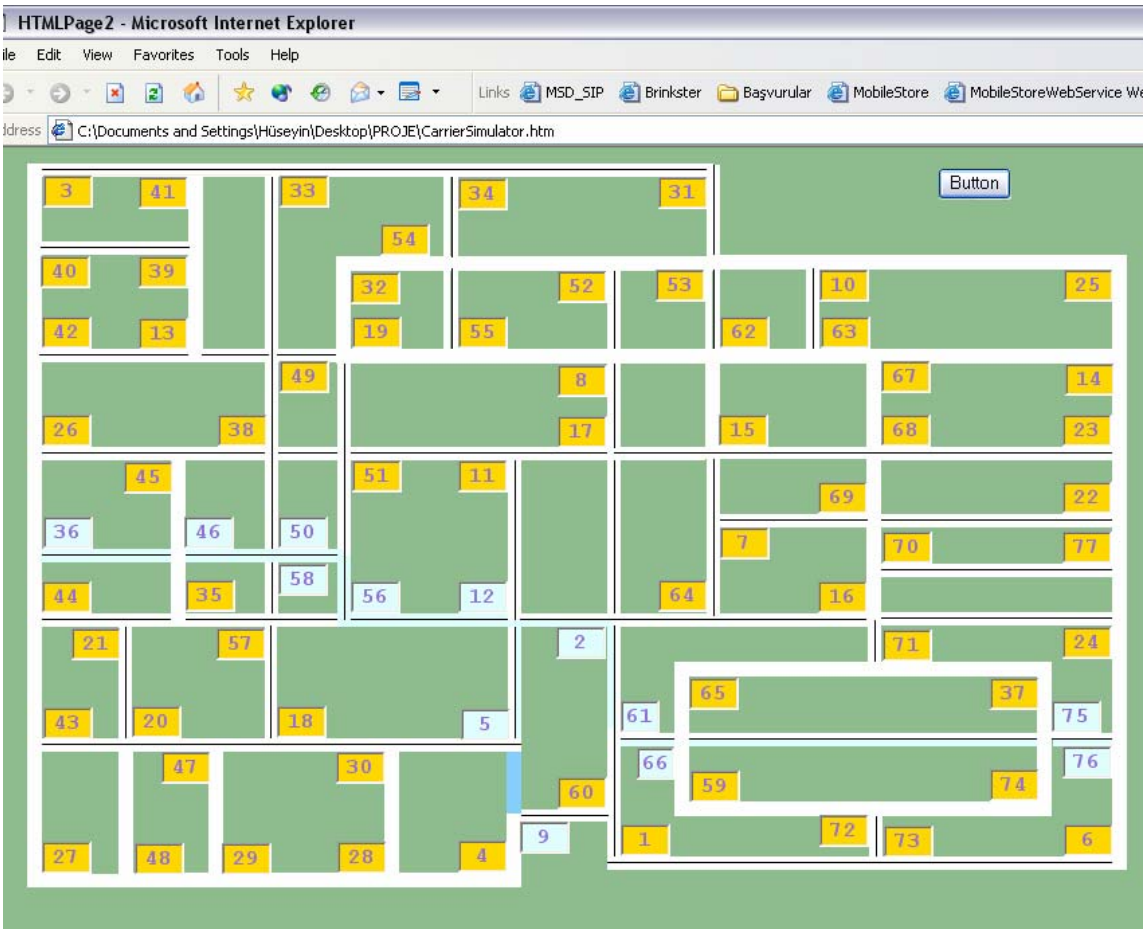


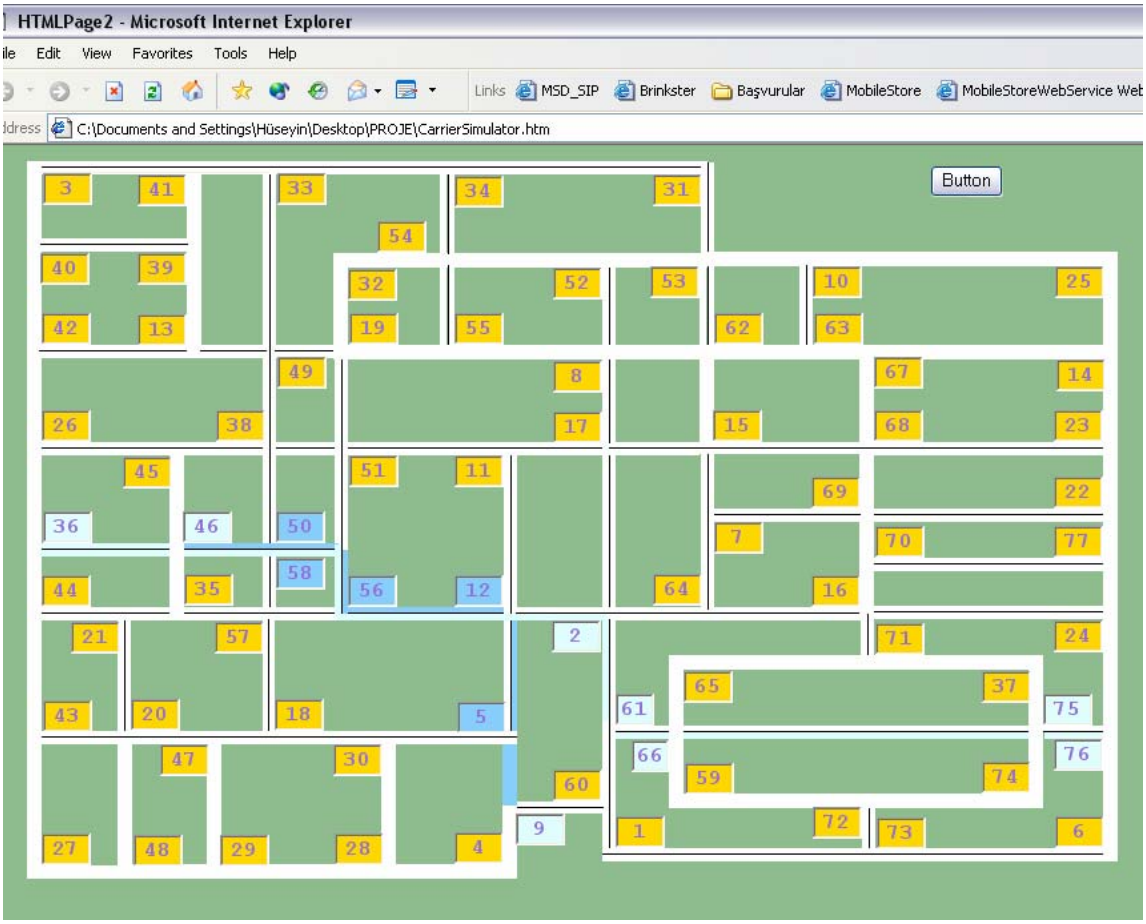
- Order is assigned to most suitable carrier according to shortest path calculations

Wap Interface For Carriers From Cell Phones...

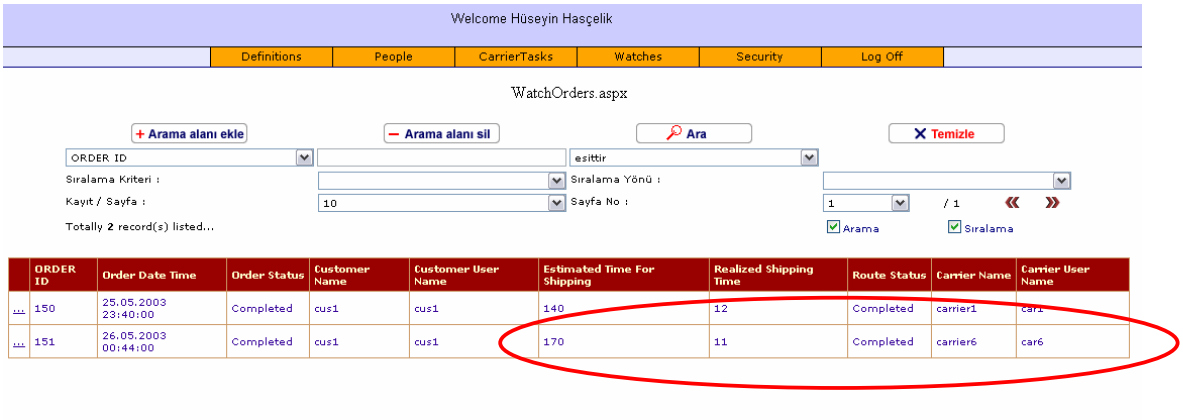
As the Carrier Moves...





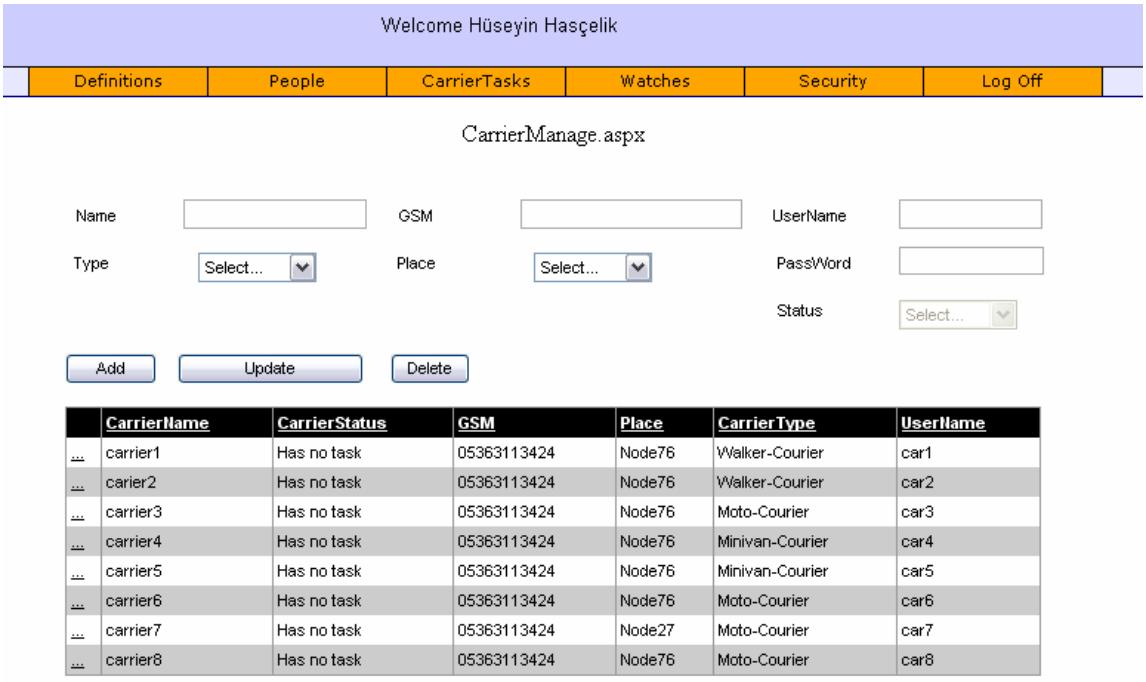


When Carrier Completes his task ...



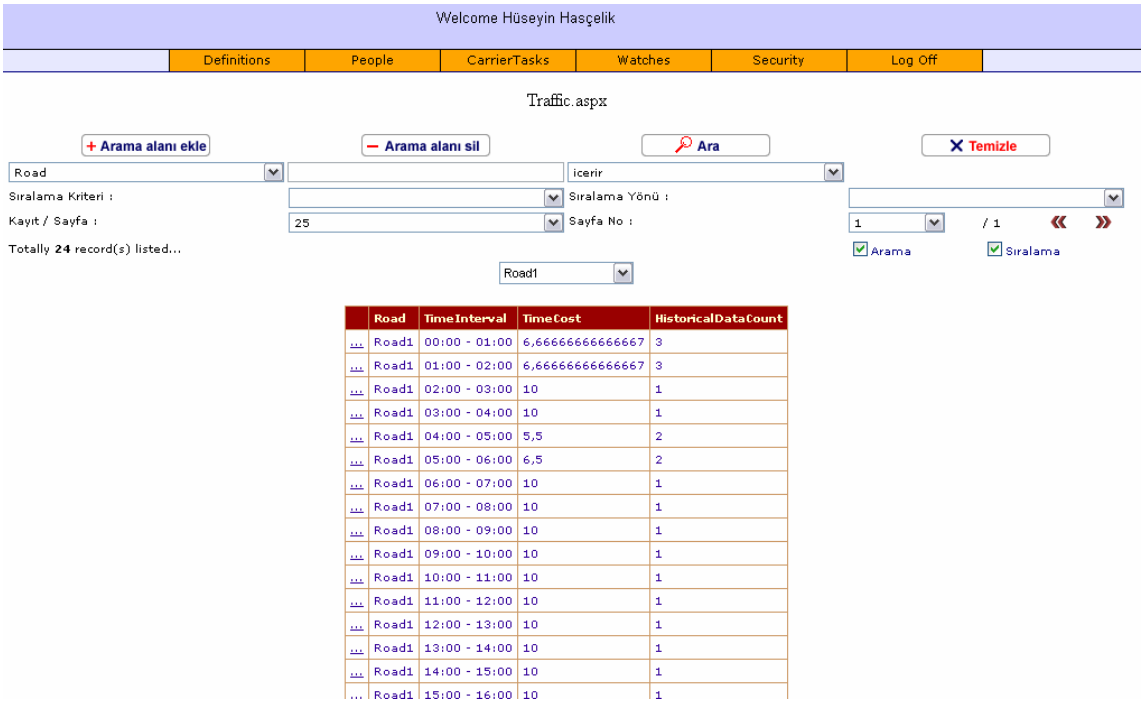
We Can see him as he completes his task from web interface...

In fact we can monitor all Carriers and customers from web interface...



As carriers update road information while they are routing :

We have a Historically updated huge City Traffic information which will be used for next shortest Path Route calculations :



	Road	TimeInterval	TimeCost	HistoricalDataCount
...	Road1	00:00 - 01:00	6,66666666666667	3
...	Road1	01:00 - 02:00	6,66666666666667	3
...	Road1	02:00 - 03:00	10	1
...	Road1	03:00 - 04:00	10	1
...	Road1	04:00 - 05:00	5,5	2
...	Road1	05:00 - 06:00	6,5	2
...	Road1	06:00 - 07:00	10	1
...	Road1	07:00 - 08:00	10	1
...	Road1	08:00 - 09:00	10	1
...	Road1	09:00 - 10:00	10	1
...	Road1	10:00 - 11:00	10	1
...	Road1	11:00 - 12:00	10	1
...	Road1	12:00 - 13:00	10	1
...	Road1	13:00 - 14:00	10	1
...	Road1	14:00 - 15:00	10	1
...	Road1	15:00 - 16:00	10	1

12 Technical Details

- We have used the **shortest path algorithm** and its derivatives in our project in methods which finds the suitable route, store or carriers…
- We have used **3-layered structure** to increase the software maintenance and independence.
- We have used **object oriented** features of .NET like custom web components, user controls and inheritance.
- We have had some difficulties about web services security (web services authorization) as we could not find a strong solution about this issue.

12.1 Technology Used:

ADO.NET, ASP.NET, Mobile Internet Toolkit, .NET Web Services

12.2 Programming Languages:

Visual Basic.NET

12.3 NET Enterprise Servers

Microsoft SQL Server 2000, Microsoft Internet Information Server 5.0

12.4 Platforms

All platforms which is supported by .NET Framework

12.5 Standards

in addition to SOAP and XML, we use some custom coding standards and project folder management standarts. We use 3-layered structure as a development standart.

12.6 Mobile Extensions

Mobile Internet Toolkit

13. Result

Mobile Store project is quite appropriate for improving. The only process to enlarge the current region where the system services is to add the new road and store information to Mobile Store database. Also to add customers and carriers into project or delete from project is required only a few mouse clicks.

The next stage that is considered to be included in Mobile Store is a traffic information web service. Thanks to this service, our members who have the proper mobile devices can get the chance to view the status of important roads in runtime from their devices. To provide this functionality, a video web service that takes the latest image captures from remote cameras and broadcasts this information in XML format must be attached to system. Thus, Mobile Store can inform the customers about road status in the current region using its video web service.

14 Resources and References

- ASP.NET Web Developer's Guide - SYNGRESS
- ASP.NET Unleashed - SAMS
- ASP.NET Professional - WROX
- Building XML Web Services for the Microsoft .NET Platform - Microsoft
  ISBN : 0-7356-1406-7
- Building .NET Applications For Mobile Devices – Wigley Roxburg
- MSDN
- Writing Secure Code - Michael Howard and David LeBlanc
  ISBN : 0-7356-1588-8
- http://www.orie.cornell.edu/~or115/handouts/handout3/handout3.html
  (For Shortest Path Algorithm)
- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/daab-rm.asp (Data Access Application Block)