INGENJÖRSHÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

Förbättra Internetsystem genom implementation av AJAX-teknik

Johan Pettersson

Jonas Kristiansson

EXAMENSARBETE 2006

Datateknik

INGENJÖRSHÖGSKOLAN

HÖGSKOLAN I JÖNKÖPING

Förbättra Internetsystem genom implementation av AJAX-teknik

Improving Internet systems by using AJAX techniques

Johan Pettersson

Jonas Kristiansson

Detta examensarbete är utfört vid Ingenjörshögskolan i Jönköping inom ämnesområdet Datateknik. Arbetet är ett led i den treåriga högskoleingenjörsutbildningen. Författarna svarar själva för framförda åsikter, slutsatser och resultat.

Handledare: Ragnar Nohre

Omfattning: 10 poäng (C-nivå)

Datum: 2006-06-15

Arkiveringsnummer:

Postadress: Box 1026 551 11 Jönköping

Besöksadress: Gjuterigatan 5

Telefon: 036-10 10 00 (vx)

Abstract

Today, many businesses and persons prefer to use web applications in their daily work. Until now, most of the websites are being constructed using an old traditional model. The model implies that whenever someone clicks on a link to get new information, a request is being sent to an information server. The server prepares, and returns the new requested information. For the client to take part of the new information, the client has to reload the whole page with all its content. By using AJAX-techniques the information flow will be more similar to an ordinary local application program. That is, only the information that needs to be updated is sent back to the client browser.

The purpose of this essay was to investigate whether AJAX-techniques can improve Internet systems, both in user-friendliness and in a development point of view. This was performed studying two typical cases of web applications. By comparing the two applications, before and after implementation of AJAX, we could see improvements with AJAX in all of our investigated aspects. The user can work more efficient in applications where AJAX is implemented. For example, the application can give more direct feedback on the user's actions by working with a much higher level of active communication. A whole new world of possibilities opens for the developer to construct applications that were not possible before. The new continuous communication between server and client creates higher demands on the developer than before. Since communication occur in the background, error handling is harder to troubleshoot. In our opinion the advantages of AJAX by far outnumbers the drawbacks. In the future, more and more developers will use AJAX which will lead to better solutions of debugging. This is a challenge to the Internet Community, a challenge that we believe will be met.

Sammanfattning

Webbapplikationer har en stadigt ökande användning i affärsvärlden. Fördelarna är många, några är enkelhet i teknisk implementering, geografisk spridning och underhåll. De flesta webbapplikationer och webbplatser har fram tills nu byggts på en gammal traditionell modell. Modellen innebär att användaren klickar på en länk för få ny information. En förfrågan skickas då till informationsservern som i sin tur behandlar och returnerar all information. Detta innebär att hela sidan måste laddas om och uppdateras för att visa den nya informationen. Genom att använda AJAX blir informationsflödet mer likt ett vanligt lokalt program. Det vill säga att bara den information som ändras och är relevant skickas tillbaka och uppdateras på användarens bildskärm.

Syftet med examensarbetet var att undersöka huruvida AJAX-tekniker kan förbättra Internetsystem. De två perspektiv som använts är användarvänlighet och utvecklingsmetodik. Undersökningen gjordes genom att studera två typfall av webbapplikationer ur dessa två perspektiv. Resultatet av undersökningen har bearbetats och visar en klar förbättring vad gäller båda de valda perspektiven.
Användaren kan arbeta effektivare vilket ger tidsvinster. Dessutom kan applikationen, genom den högre graden av aktiv kommunikation, ge mer direkt feedback på användarens handlingar. För utvecklaren öppnas nya möjligheter att skapa tillämpningar som tidigare inte var möjliga. Den kontinuerliga kommunikationen ställer också högre krav på programmeraren. Då kommunikationen mellan klient och server i större del sker i bakgrunden som inte syns i presentationsskiktet uppkommer även felen i bakgrunden. Det är därför svårare att felsöka en AJAX-baserad webbplats.

Vi anser att nackdelarna är små jämfört med alla möjligheter AJAX erbjuder. Lösningar för felhantering är en utmaning för Internetbranschen som den med all säkerhet kommer att lösa.

Nyckelord

AJAX, JavaScript, XML, ASP, Internetsystem, asynkron överföring

Innehållsförteckning

Figurförteckning

1 Inledning

Fördelarna med Internet och Internetapplikationer har blivit mer och mer uppenbara. Samtidigt som användningen av Internetapplikationer har ökat ställs också högre krav på både teknik och användarvänlighet. AJAX är en av de tekniker som har uppkommit utifrån dessa krav. Sedan Internets uppsving i mitten av 1990-talet har standarden för byggande av webbapplikationer, kallad Web 1.0, förändrats marginellt. Problemet med applikationer konstruerade enligt Web 1.0 är att det är svårt att uppdatera information som visas på en sida utan att ladda om hela sidan. AJAX-tekniken gör detta möjligt. En webbapplikation byggd med AJAX-teknik kan nyttja samma informationshantering som ett vanligt lokalt program. Ett föredömligt exempel på en AJAX-baserad webbapplikation är Google Maps (http://maps.google.com/).

1.1 Syfte och mål

Det primära syftet med examensarbetet är att undersöka huruvida Internetsystem kan förbättras genom tillämpning av AJAX-teknik. Ett mål är att beskriva fördelarna och nackdelarna med AJAX som teknik. AJAX är ett samlingsbegrepp som har flera olika beståndsdelar. I rapporten belyses användning av AJAX i samband med bland annat ASP, HTML, CSS, DOM och JavaScript. Vi kommer att använda två konkreta exempel. Med dessa som utgångspunkt diskuterar vi skillnaderna mellan traditionella webbapplikationer och applikationer uppbyggda med hjälp av AJAX-teknik, både från användare- och utvecklarperspektiv.

Målsättningen med undersökningen är att utvärdera vad den nya tekniken innebär för utvecklare och användare. Vad måste utvecklare ha i åtanke vid byggande av en applikation? Blir en webbapplikation användarvänligare, mer effektiv och mindre resurskrävande med AJAX-teknik?

1.2 Avgränsningar

Det finns en rad problemställningar i den gamla modellen av webbprogrammering som lösas med AJAX-teknik. I analysen utgår vi från två typfall av webbapplikationer vilka vi anser är tillämpningar av AJAX-teknik där Internetapplikationernas funktion och användbarhet ökar markant.

- **Listning av kunder från databas**
  Hämtning av kundinformation från en kund-databas.

- **Information med behov av kontinuerlig uppdatering.**
  Ett webb-baserat tidsplaneringssystem med gantt-diagram för visning av
- beläggning med kontinuerligt uppdaterad information.

1.3 Disposition

Rapporten är uppbyggd på följande sätt:

Ett teoriavsnitt där vi belyser ämnet och beskriver metodiken undersökningen baseras på. I avsnittet får läsaren en genomgång av ämnesområdet för att ge förståelse för begrepp och problematik i rapporten. Nästa avsnitt är genomförande, där beskrivs tillvägagångssätt och mera ingående problemställningar. Slutligen presenteras resultatet och de slutsatser vi kommit fram till. Rapporten avslutas och sammanfattas med en konkretiserande diskussion i slutsatsen.

2 Teoretisk bakgrund

Internet är ett världsomspännande informationsnätverk. Vi använder oss idag av Internet för att t.ex. göra bankärenden, handla varor eller sprida och söka information på hemsidor, bloggar eller liknande. En viktig och växande del av Internet är möjligheten att distribuera mjukvara. En av de största fördelarna med att använda Internetsystem är åtkomstmöjligheterna. Det enda som behövs är en Internetuppkopplad dator med en webbläsare.

Den klassiska webbapplikationen använder en grafisk webbläsare. Genom webbläsaren anger användaren en så kallad URL (se kapitel 2.3.1). En URL anger sökvägen till en informationssamling, vanligtvis en hemsida eller en webbapplikation. När användaren, klienten, efterfrågar en URL startas en kommunikation med målets server. Då kontakten är upprättad skickar servern tillbaka all information klienten efterfrågat. När överföringen är klar avslutas kommunikationen med servern och webbläsaren visar den mottagna informationen i korrekt form för användaren. Väljer användaren att hämta mer information genom att exempelvis klicka på en länk upprättas återigen kommunikationen med servern och proceduren upprepas.

2.1 HTML

Innan 1990 var Internet ett komplicerat och svårsökt informationsnätverk. Det var då fysikern Tim Berner-Lee tröttnade och uppfann ett nytt sätt att indexera och hantera informationen på nätet. Metoden kom att kallas för HTML (HyperText Markup Language) och använde sig av hypertextlänkar vilket gjorde att man kunde hoppa mellan olika informationskällor på ett snabbt sätt. Hans teknik blev snabbt känd och redan 1993 fanns det över 100 servrar på Internet som stödde denna teknik.

2.1.1 Tekniken

HTML-standarden är grunden för alla hemsidor som idag finns på webben. HTML är ett sätt att beskriva en texts utseende. Det görs genom att i texten lägga till så kallade formatkommandon.

Exempel på formatkommando som gör en text kursiverad:

```
<i>Exempel på kursiv text i HTML</i>
```

Ovanstående rad HTML-kod genererar följande resultat i webbläsaren:

Exempel på kursiv text i HTML

Under förutsättning att det finns en webbläsare installerad kan HTML-dokument läsas på nästan alla typer av plattformar. Dagens nya versioner av HTML erbjuder möjligheter att bygga upp stycken, tabeller, inmatningsformulär, händelsehantering m.m. [12]

2.2 CSS

CSS är en förkortning för ”Cascading Style Sheets” och används för att definiera stilmallar för HTML & XML dokument. CSS har funnits sedan oktober -94 men det började användas mer frekvent när Internet Explorer 3.0 började stödja CSS. Med en CSS-mall kan ett obegränsat antal sidors formatering styras. Definitionerna kan läggas in direkt i en webb-sida eller kan bifogas som en separat CSS-fil. Mallarna som beskriver utseendet på webb-dokumenten blir lätt stora och en separat fil är då att föredra.

I CSS kan utseendet på fördefinierade element i en HTML-sida bestämmas (se ”body” i Figur 1). Det är även möjligt att skapa egna klasser för exempelvis element såsom rubriker (se ”minaRubriker” i Figur 1), dessa utmärks med att sätta en punkt framför namnet på klassen.
[16]

```
body
{
   font-size : 10px;
   font-family : Verdana, Arial;
   font-weight : normal;
   font-style : none;
   color : #3A3A3A;
   text-decoration : none;
   background-color : #FFFFFF;
}

.minaRubriker
{
   font-size : 14px;
   font-family : Verdana;
   font-weight : bold;
   color : #FF0000;
   text-decoration :underline;
}
```

Figur 1. Exempel på stil-regler till klasser i stilmall.css. [16]

Båda av de två nedanstående exempel på implementation av stilmallar genererar följande resultat: **Min examensrapport**

```
<HTML>
<HEAD>
<LINK REL="StyleSheet" HREF="stilmall.css" TYPE="TEXT/CSS">
</HEAD>
<BODY>
<FONT CLASS=”minaRubriker”>
   Min examensrapport
</FONT>
</BODY>
</HTML>
```

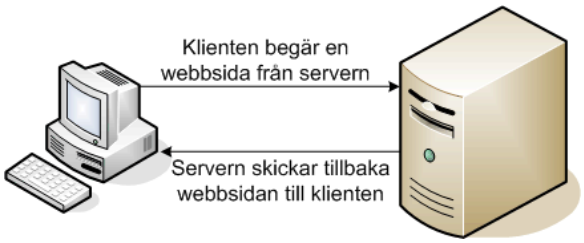
Figur 2. Exempel på stil-regler genom klasser i ett HTML-dokument. [16]

```
<HTML>
<HEAD></HEAD>
<BODY>
<FONT STYLE="font-size:14px; font-family:Verdana; font-
weight:bold; color: #FF0000; text-decoration:underline;">
          Min examensrapport
</FONT>
</BODY>
</HTML>
```

Figur 3. Exempel på stil-regler direkt i ett HTML-dokument. [16]

2.3 Klient/Serverhantering för webb

Internet styrs enligt klient/server-modellen, vilket betyder att två datorer arbetar gemensamt för att utföra en uppgift. Det är en teknik för distribuerade applikationer och dokument vilken innebär att användaren, klienten, kan koppla upp sig mot olika servrar för att hämta information. En serverdator står alltid redo att möta klientdatorns förfrågningar. Då en klientdator begär information processar serverdatorn informationen och skickar tillbaka den begärda, behandlade informationen.



Figur 4. Modell över en transaktion med klient/webbserver. [18]

2.3.1 Uniform Resource Locator

För att en klient på Internet skall kunna beskriva vilken server och vilket dokument som ska hämtas används ett speciellt adresseringssystem. Detta adresseringssystem kallas ”URL” (Uniform Resource Locator). En URL är en sökträng uppbyggd i fyra olika delar för att förklara hur och var information hämtas.

```
http://www.aftonbladet.se:80/artiklar/art2712.html
```

Figur 5. Exempel på en URL.

I exemplet kan man urskilja fyra olika delar. Först anges vilket protokoll som används (http).Därefter följer namnet på servern (www.aftonbladet.se). Efter detta anges vilken port som ska användas (80). Det behöver inte anges någon port om protokollets standardport används. Den fjärde och sista delen är dokumentidentifieraren (/artiklar/art2712.html) som talar om var på servern dokumentet finns att hämta. [11]

2.4 DOM - Document Object Model

DOM står för Document Object Model och är ett plattform- och språkoberoende API (*Application Programming Interface)* för att manipulera XML och HTML. DOM bistår med ett set av objekt och metoder som kan användas för att dynamiskt ändra struktur, utseende och information på ett webbdokument.

Till varje objekt i ett dokument kan en unik nyckel sättas, ett så kallat id. Genom DOM kan en referens till det unika objektet skapas. Via referensen kan när som helst objektet manipuleras. Det är även möjligt att skapa så kallade ”händelsehanterare”. De olika händelsehanterarna reagerar på saker som händer i dokumentet och kan då utföra olika saker. Exempel på detta är när en knapptryckning görs eller när man för muspekaren över något objekt, till exempel en bild. Då kan DOM genom en referens till ett text-objekt skriva ut texten ”Hej, du tryckte på en knapp eller förde musen över bilden och får nu inte se bilden mer”. Samtidigt som texten visas sätts genom en annan referens bildens synlighetsattribut till osynlig. [17]

2.5 XML - eXtensible Markup Language

Extensible Markup Language, förkortat XML, är ett markeringsspråk. Ett markeringsspråk är ett språk för att definiera uppmärkningsspråk och används för att kunna strukturera och separera delar av data i exempelvis ett informationsflöde. XML kan tolkas och bearbetas oberoende av programspråk eller datorplattform. Språket kom till då behovet uppstod av något mer kraftfullt än HTML. I XML finns möjlighet att skapa egna markeringstaggar. Detta gör att det är väldigt enkelt att skriva XML-kod. Trots detta relativt enkla sätt att arbeta kan ändå otroligt kraftfulla applikationer skapas genom att använda XML som databärare. [1]

2.5.1 Varför XML

Som vi nämnt tidigare är XML språk-, datorplattforms- och operativsystemsoberoende. Det behövs heller inte några tillägg för att få XML att fungera med befintliga webbprotokoll som exempelvis HTTP-protokollet. Uppbyggnaden av ett XML-dokument är relativt enkel, vilket underlättar förståelsen av ett dokument. XML har genom sin enkelhet blivit populärt vid lagring av data samt kopplingar mellan Internetapplikationer och databaser. Det är och kommer i framtiden bli en viktig del i Internets utveckling. [1]

2.5.2 Ett XML-dokuments uppbyggnad och uppmärkningar

För att strukturera och dela upp data används uppmärkningar. XML använder sig av ett ”mindre än”- (<) eller ett ”och”-tecken (&) för att visa att en uppmärkning börjar. Ett ”större än”-tecken (>) avslutar uppmärkningen. XML använder även apostrof (’) och citattecken (”) för uppmärkning. Vill man använda sig av något av dessa tecken i en vanligt text som skickas genom XML måste de ersättas med ersättningstecken. [1]

Tecken	Ersättningstecken
&	&amp;
<	&lt;
>	&gt;
’	&apos;
”	&quot

Figur 6. Tecken och dess ersättningstecken.

En hemsida

```
<?xml version=”1.0”?>
<web.site>
  <head>
    <title>
      My web site
    </title>
    <banner source=”topbanner.gif”/>
  </head>
  <body>
    <main.title>
      Welcome to my web site
    </main.title>
    <rule/>
    <text>
      <para>
        Web site is under construction
      </para>
    </text>
  </body>
  <footer source=”foot.gif”/>
</web.site>
```

Ett personregister

```
<?xml version=”1.0”?>
  <contacts>
    <contact>
      <name>
        <first>
          Jim
        </first>
        <last>
         Brown
        </last>
      </name>
      <address>
        <street>
          Main road 10
        </street>
        <city>
          Silicon Valley
        </city>
      </address>
      <email>
        jim@abc.com
      </email>
    </contact>
  </contacts>
```

Figur 7. Ovan visas två exempel på typiska XML-dokument. [1]

2.6 JavaScript

JavaScript är ett interpreterande skriptspråk som inbäddas i HTML-kod. Med ett interpreterande skriptspråk menas att webbläsaren tolkar koden i applikationen varje gång applikationen körs. Koden lämnas oförändrad, dvs. den kompileras inte och görs inte om till maskinkod. Interpreteringen kan göras rad för rad eller all kod samtidigt. JavaScript togs fram av ett företag som heter Netscape. I början var det enbart Netscapes egna webbläsare som kunde tolka JavaScript, men i dagsläget är de flesta fullt kompatibla med Javascript. Internet Explorer stödjer JavaScript från version 3.0.

Skriptspråket är objektorienterat och plattformsoberoende vilket öppnar många portar för olika användningsområden. I HTML är informationen statisk, det vill säga att informationen inte kan ändras. Genom skriptspråk som JavaScript kan ändringar utföras. JavaScript kan användas vid exempelvis beräkningar av data, styrning av DOM-objekts egenskaper, validering av formulär, eller rörliga menyer. JavaScript körs inte på webbservern, utan lokalt och använder sig av klientens datorkraft. I och med att skriptet körs lokalt och inte behöver kompileras av servern kan ändringar göras på sidan utan omladdning. Dock måste det ske en kommunikation med servern om ny information ska hämtas eller sparas. Sidan måste då laddas om, alternativt används tekniker som AJAX vilken sköter kommunikationen i bakgrunden. [20]

JavaScript-koden kan initieras på flera olika sätt (för exempel se Figur 8, Figur 9). För att strukturera och återanvända kod kan skript och funktioner även placeras i en separat, så kallad, .js-fil. Denna fil inkluderas i de webbdokument skripten ska användas i (Figur 10).

```
<HTML>
<TITLE>Exempel 1 av implementation av JavaScript</TITLE>
<HEAD>
   <SCRIPT language="JavaScript">
        function init()
        {
            alert(“Nu har sidan laddats!”)
   }
   </SCRIPT>
</HEAD>
<BODY onLoad=init();>
   En textruta dyker nu upp när sidan har laddats.
</BODY>
</HTML>
```

Figur 8. När sidan laddats åkallas en funktion som heter init. Funktionen visar en textruta, också kallad alert-ruta, med texten ”Nu har sidan laddats!”.

```
<HTML>
<TITLE>Exempel 2 av implementation av JavaScript</TITLE>
<HEAD>
</HEAD>
<BODY>
   <font onClick="alert(Date());">Klicka för datum</font>
</BODY>
</HTML>
```

Figur 9. När användaren klickar på texten ”Klicka för datum”, visas en alert-ruta som genom en inbyggd JavaScript-funktion visar dagens datum.

```
<HTML>
<TITLE>Exempel 3 av implementation av JavaScript</TITLE>
<HEAD>
   <SCRIPT type="text/javascript" src="minaScript.js"></SCRIPT>
</HEAD>
<BODY onLoad=init();>
   En textruta dyker nu upp när sidan har laddats.
   Funktionen är placerad inuti filen minaScript.js som är
   inkluderad.
</BODY>
</HTML>
```

Figur 10. En textruta dyker nu upp när sidan har laddats. Funktionen är placerad inuti filen minaScript.js som är inkluderad i webbdokumentet.

2.7 AJAX - Asynchronous JavaScript and XML

AJAX är en förkortning av ***A**synchronous **J**avaScript **a**nd **X**ML* och är ett samlingsnamn för ett antal olika tekniker. AJAX används för att bygga Internetapplikationer som kräver mer interaktivet än traditionella Internetapplikationer kan hantera.
En hemsida som behöver laddas om inför minsta lilla uppdatering är genom att använda AJAX-tekniken ett minne blott.

Teknikerna som AJAX utgörs av är:

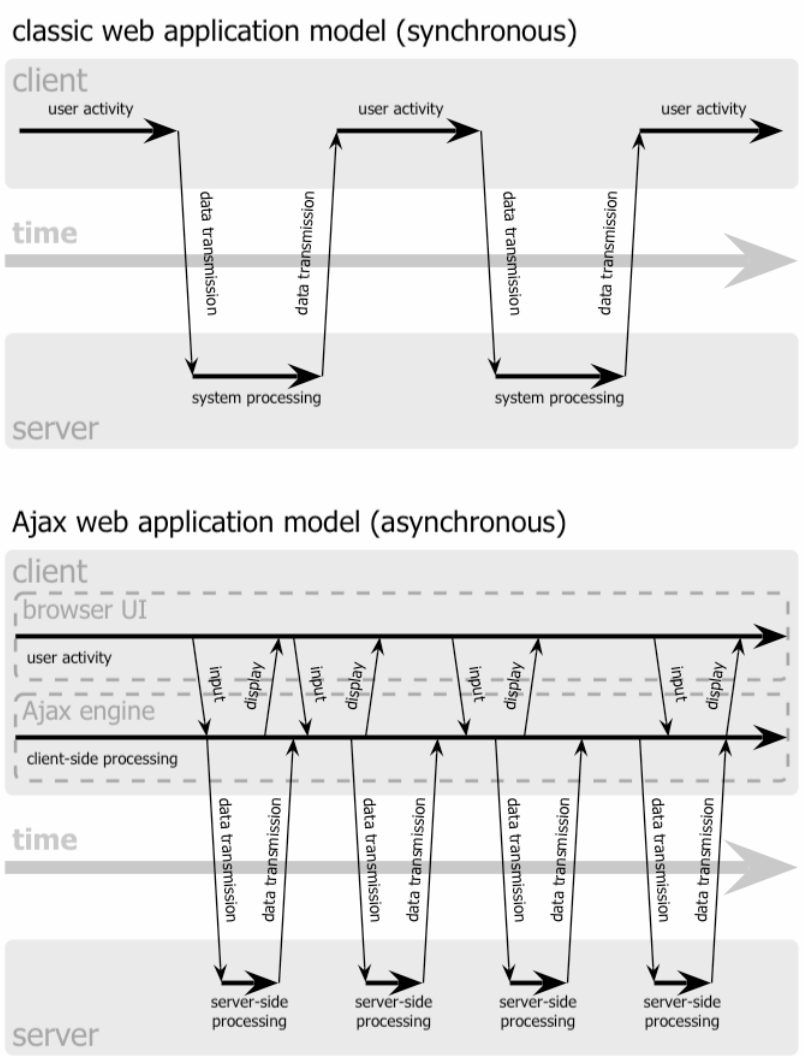
- HTML och CSS för att presentera innehåll.
- XMLHTTP för att skicka och ta emot data asynkront mellan server och klient.
- DOM för att på en webbsida dynamiskt kunna ändra innehåll.
- XML för att skicka och ta emot data.
- JavaScript för att integrera de olika teknikerna.

2.7.1 Historik om AJAX

AJAX myntades och fick sitt stora genombrott i februari 2005. En av de största anledningarna till att AJAX fick sitt uppsving just då var att Google, en stor aktör inom Internetteknik, lanserade tjänsten Google Maps som till stor del är baserad på denna teknik. Det som var mest sensationellt med Google Maps var att användarna kunde i ”realtid” förflytta sig i en karta, och detta med väldigt liten dataöverföring. Det Google Maps gjorde var att samtidigt som kartan visades för klienten förde applikationen en dialog med servern där kartinnehållet låg lagrat. Detta gjorde att när klienten ville förflytta visningen av kartan kunde Google Maps fortsätta visa den aktuella kartbilden samtidigt som förfrågan sändes till servern om nästa delbild. Med AJAX-teknik utökades kartan med den önskade delbilden, i realtid. Användaren upplevde att kartbilden förflyttades i önskad riktning utan omladdning av sidan. Den enda information som sändes igen var de delbilder som utgjorde förändringen av den efterfrågade kartbilden, inte hela den önskade kartbilden. AJAX-tekniken gör att Internetapplikationerna mer och mer upplevs fungera som ett program lokalt installerat på en dator. [3] [6] [7] [8]

2.7.2 Skillnader mellan asynkron och synkron överföring

Förfrågningar till en server kallas för ”requests” och för svar som returneras används termen ”response”. De flesta webbapplikationer använder sig av en modell för request/response som betyder att webbläsaren, klienten, hämtar all information på hela sidan varje gång en förfrågan görs, s.k. synkron överföring. AJAX möjliggör istället asynkron överföring av data, dvs sidans information kan sändas i mindre delar. AJAX kommunicerar asynkront med hjälp av objektet XMLHTTPRequest. XMLHttpRequest är den mest grundläggande delen av alla AJAX-relaterade objekt. Objektet är förinstallerat på de vanligaste webbläsarna, vilket gör att användaren inte behöver ladda ner några extra komponenter för att kunna använda AJAX.



Figur 11. Bilden överst illustrerar en synkron kommunikation av en traditionell webbapplikation, den nedre bilden visar hur en asynkron AJAX-applikation kommunicerar. [10]

AJAX är en viktig del i vad som benämns Web 2.0. I Web 1.0 refereras till den traditionella webben där det är en väldigt tydlig förfrågan- och svarsmodell (se övre bild i Figur 11). Ett exempel är webbplatsen amazon.com. Ponera att en sökning på en artikel ska göras. Det första som görs är att ladda söksidan och ett sökbegrepp skrivs in. Hittills är förfarandet detsamma som Web 2.0. Det är först när sök-knappen trycks ner som skillnaden märks. I en hemsida uppbyggd enligt Web 1.0 skickas en förfrågan till servern som i sin tur behandlar förfrågan och sedan skickar tillbaka ett svar som visas i webbläsaren. Det är dock inte bara informationen om just den sökta artikeln som skickas tillbaka, det är all HTML-kod, eventuella bild-objekt och liknande data som laddas om och senare visas på skärmen. På sidor byggda under Web 2.0 returnerar servern däremot endast ny informationen om artikeln och uppdaterar enbart det väsentliga på sidan. I detta fall information om artikeln, svaret på sökningen. Det görs alltså fortfarande förfrågningar och svar, men de sker i bakgrunden och uppdaterar bildskärmsinformationen när den är mottagen från servern.

För att klient och användare ska veta status på förfrågan mellan server och klient finns en variabel, ”readyState” (se Figur 13). Variabeln sätts till olika status med hjälp av en händelsehanterare. Med olika status menas till exempel om förfrågan initieras, laddas eller är färdig. Detta gör att när en förfrågan pågår kan en text som exempelvis ”information hämtas” visas.

```
XMLHttpRequest request = new XMLHttpRequest();
```

Figur 12. Så här ser skapandet av en XMLHTTPRequest ut.

- open(): Sätter upp en ny förfrågan till en server.
- send(): Sänder en förfrågan till en server.
- abort(): Avbryter den aktuella förfrågan.
- readyState: Ger status på den aktuella förfrågan.

  0 - Uninitialised
  1 - Loading
  2 - Loaded
  3 - Interactive
  4 - Completed

- responseText: Texten som servern skickar tillbaka som svar på en föfrågan.

Figur 13. Några metoder och objekt som vanligen används på XMLHTTPRequest. [9], [10]

Här visas ett exempel på hur en enkel AJAX-applikation kan se ut.

```
Hello World
```

Figur 14. Innehållet i textfilen helloWorld.txt

```
<html>
  <head>
    <script language="JavaScript">
      function ajaxHelloWorld()
      {
        var xmlHttpReq = false;
        var self = this;

        if (window.XMLHttpRequest) // Mozilla/Safari
        {
          self.xmlHttpReq = new XMLHttpRequest();
        }
        else if (window.ActiveXObject) // IE
        {
          self.xmlHttpReq = new ActiveXObject("Microsoft.XMLHTTP");
        }

        self.xmlHttpReq.open('POST', "helloWorld.txt", true);
        self.xmlHttpReq.setRequestHeader('Content-Type',
        'application/x-www-form-urlencoded');
        self.xmlHttpReq.onreadystatechange = function()
        {
          if (self.xmlHttpReq.readyState == 4)
          {
           alert(self.xmlHttpReq.responseText);
          }
        }
        self.xmlHttpReq.send();
      }
    </script>
  </head>
  <body>
    <input type=button name=myButton value='Hämta Hello World'
    onClick='ajaxHelloWorld();'>
  </body>
</html>
```

Figur 15. En applikation som med hjälp av AJAX-tekniker hämtar innehållet i textfilen helloWorld.txt och presenterar texten för användaren genom en ”alert-ruta”.

Figur 16. Figuren visar hur resultatet av AJAX-tillämpningen presenteras för användaren.

2.8 ASP – Active Server Pages

I takt med att Internet förvandlats till en informationsresurs för allmänheten har det dykt upp företag som säljer produkter och tjänster. I början användes bara HTML för att skapa oföränderliga webbsidor. Dessa hemsidor blir nu snabbt föråldrade. Om en stor e-handelsbutik enbart bestod av oföränderliga webbsidor skulle man inte kunna handla med hjälp av en kundkorg på hemsidan, inte söka igenom deras register och de skulle helt klart inte sälja lika mycket. Genom Microsofts lösning kallad Active Server Pages (ASP) löser man dessa problem och skapar dynamiska hemsidor. [18]

2.8.1 Vad är Active Server Pages?

Active Server Pages (ASP) är en teknik som används för att bygga in programkod i HTML-sidor, ett ramverk med objekt och rutiner. Den största fördelen är att sidorna kan göras dynamiska och interaktiva. Programkoden är normalt skriven i VBScript, en variant på programmeringsspråket Visual Basic. Den inbyggda koden ryms inom tecknen <% och %> och kan vid en första anblick förefalla ostrukturerad. Arbetssättet är dock effektivt för den utvecklare som är kunnig i både programspråket och HTML.

ASP är serverbaserat vilket betyder att koden exekveras på servern och returnerar resultatet till klienten i form av ett HTML-dokument. En stor fördel med denna teknik är att den är helt plattformsoberoende, vilket betyder att koden kan exekveras och resultat returneras oberoende av operativsystem, webbläsare eller dator. [13], [14]

Figur 17. Växelverkan mellan klient och server för ASP-filer. [18]

Nedan följer ett exempel på ASP-kod som ansluter till en databas och hämtar information samt skriver ut detta till ett HTML-dokument.

```
<html>
<title>Exempel, hämta data från MySQL databas</title>
<body>
   <%
   ’Öppna databasen
   set db = Server.CreateObject("ADODB.connection")
   db.open "driver={MySQL ODBC 3.51 Driver}; Server=localhost;
Uid=xx; Pwd=xx; Database=xx"

   ’Skapa recordset
   Set rs = server.createobject(”adodb.recordset”)
   Sql = “SELECT * FROM tabell;”

   Rs.open sql
   Do while not rs.eof
       Response.write(“Data:”& rs(“falt1”) &”<br>”)
       Rs.movenext
   loop
   Rs.close
   Set rs = Nothing
   db.close
   set db = Nothing
   %>
</body>
</html>
```

Figur 18. Exempelkod i ASP.

2.9 Databaser

Uppgifter av olika slag kallas *data*. Information skiljer sig ofta från data och kan beskrivas som tolkad data. Ett exempel av data kan vara 14 samtidigt som informationen är att ”Frölunda hade 14 skott mot mål”.

Med ordet databas menar man oftast:

- En samling data som hör ihop.
- Data som modellerar en del av en värld, till exempel ett företag och dess verksamhet.
- Att data är persistent, det vill säga inte försvinner när man avslutar programmet.

2.9.1 Databashanterare

Program som har som uppgift att lagra data och hantera databaser kallas oftast för databashanterare, databashanteringssystem, förkortat DBHS. Många av dessa är stora och mycket komplexa. Några av de mest kända databashanterarna är MS Access, MySQL och MS SQL.

Ett alternativ till databashanterare är att spara data i filer. Det är dock arbetskrävande att bygga system som kan strukturera dataflödet. Därför finns det många fördelar med att använda en databashanterare istället.

De viktigaste fördelarna med att använda en databashanterare är:

- Enkelhet – standardiserade format, snabbt igång.
- Kraftfull – kan hantera stora mängder data utan kapacitetsproblem.
- Flexibiliteten – Relativt enkelt att utöka tabeller med ytterligare fält.

Databasteknik möjliggör samtidig åtkomst av data
En databashanterare förhindrar skadliga krockar om flera användare läser och skriver i samma informationsmängd samtidigt. Databashanteraren kan låsa specifika poster till specifika användare. Risk för fel beroende på överlagring etc. minskas härigenom markant.

Databasteknik möjliggör återställande efter krascher
Vid datahaveri på grund av yttre eller inre faktorer såsom hårddiskkrascher, strömavbrott eller liknande riskeras dataförlust. Används traditionell metod med arbetsfiler i primärminnet för senare lagring på hårddisken kan stora skador uppstå vid ett eventuellt haveri. Databashanterare har dock funktionalitet för att hantera detta genom att hålla data kontinuerligt uppdaterad och korrekt.

Databasteknik gör det lättare att möta olika användares behov
Då olika användare av informationen har olika behov erbjuder normalt databashanterare flera olika gränssnitt, det vill säga möjligheter för användaren att kommunicera med databasen.

Databasteknik möjliggör bättre säkerhet
Med hjälp av behörighetshantering, som de flesta databashanterare stödjer, kan rättigheterna styras för respektive användare. Behörighetshanteringen används även för att skydda databasen mot obehörig åtkomst.

2.9.2 SQL - Frågespråket

En sökning i databasen benämns i dagligt tal en ”fråga” eller en ”query”. Språket som frågan ställs med benämns frågespråk. SQL ”Structured Query Language” är ett standardiserat frågespråk för relationsdatabaser. Idén till frågespråk kom från dr Edgar Codds uppsats från 1970 som handlade om stora delade databaser. SQL utvecklades från början av IBM men utvecklades snabbt till flera varianter av olika programvaruutvecklare. Språket antogs formellt av ANSI 1986 och av ISO 1987. 1992 omarbetades standarden och standarden SQL-92 beslutades.

SQL består av två delar. Dels de satser som används för att konstruera databasen, benämnt Data Definition Language (DDL). Dels frågespråket för hantering av data i den färdiga databasen, benämnt Data Manipulation Language (DML).

De fyra vanligaste satserna i en SQL fråga är:

```
SELECT fält1, fält2 FROM tabell WHERE villkor ORDER BY fält
```

Figur 19. Från tabellen (tabell) hämtas angivna fält (fält1 och fält2) från alla rader där villkoren (villkor) är uppfyllda. [15]

```
DELETE FROM tabell WHERE villkor
```

Figur 20. Ta bort rader ur tabellen (tabell) där villkoret (villkor) är uppfyllt. [15]

```
UPDATE tabell SET fält1 = 'värde', fält2 = 'värde' WHERE villkor
```

Figur 21. Ändra fält1 och fält2 till värdet (värde) för alla rader där villkoret (villkor) är uppfyllt. [15]

```
INSERT INTO tabell (fält1,fält2) VALUES ('data1','data2')
```

Figur 22. Lägg till en rad i tabellen tabell. [15]

3 Genomförande

Arbetet genomfördes under våren 2006. Vi ville fördjupa oss i AJAX-tekniken för att undersöka användbarheten inom området webbapplikationer, med betoning på användarvänlighet, interaktivitet och applikationsutveckling.

3.1 Studier

Inledningsvis studerade vi AJAX-tekniken inklusive alla berörda språk och tekniker. Informationen sökte vi i böcker och på Internet. På Internet fanns en uppsjö av diskussioner om just tekniken och dess fördelar respektive nackdelar. Tonvikten i informationssökningen lade vi på att finna förbättringar och förenklingar för såväl utvecklare som användare.

Vi studerade vilka möjligheter AJAX-tekniken ger från en utvecklares perspektiv. Det fanns många frågor vi ville få besvarade. Ger AJAX möjlighet till mera interaktiva webbapplikationer? Vilka fallgropar finns? Är det möjligt att snabbare utveckla applikationer med hjälp av AJAX?

En annan viktig aspekt för oss var användarperspektivet. Nya tekniker har alltid sina fördelar men överarbetas de kan resultatet bli överdrivet krångliga lösningar. Märker användaren någon skillnad? På vilket sätt blir det enklare? Går användarens arbete i en webbapplikation snabbare med AJAX-tekniken?

3.2 Praktik

Med utgångspunkt i våra litteraturstudier inledde vi praktiska laborationer med tekniken. En dator med operativsystemet Windows XP Professional konfigurerades i ett befintligt nätverk. Dessutom konfigurerades tjänsten IIS (Internet Information Services), datorn var därmed en webbserver.

Det var på denna server vi utvecklade applikationerna. Utvecklingen gjordes med hjälp av ett vanligt textediteringprogram, EditPad Pro. Koden vi skapade testades därefter med avseende på studiens utgångsperspektiv, användarvänlighet och utvecklingsmetodik, genom en klientdators webbläsare.

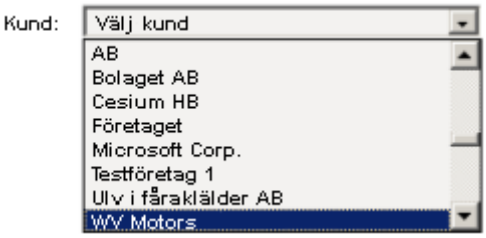
Under utvecklingstiden besvarades många av våra frågor, samtidigt såg vi intressanta tillämpningar för AJAX-tekniken. Mera om frågor och tillämpningar redovisas nedan. Själva utvecklandet anpassades efter AJAX-tekniken. Vi lade ner mycket arbete på att skapa generella funktioner med bästa tänkbara användbarhet. Generella i den meningen att funktionen kan återanvändas i andra applikationer. Optimala i den betydelse att funktionen ger största möjliga användarvänlighet. Genom jämförelse mellan utvecklingstid kontra prestanda och användbarhet i lösningarna bedömde vi till vilken användbarhet AJAX-tekniken har utifrån de två perspektiven användarvänlighet och utvecklingsmetodik.

De två typer av applikationer vi utvecklade har utan användning av AJAX-teknik stora brister för både användare och utvecklare. Vi byggde dessa både med och utan AJAX-teknik. Parametrar för undersökningen avseende utvecklingsmetodik var

tidsåtgång och hur förutsebart utvecklingsarbetet var, ju större risk för fallgropar ju svårare att förutse tidsåtgång och metodik. Med ökad risk för oförutsedda problem ökar utvecklingstiden. Vi undersökte även vilka möjligheter och problem de olika lösningarna gav ut användarens perspektiv.

3.2.1 Listning av kunder från databas

Vår funktion hanterar kunder i en applikation där användaren behöver välja en kund. Funktionen är byggd med HTML och ASP, och används för att hämta kundposter från en databas. Sökningen görs i en rullgardinsmeny, även kallad ”dropdown”. Problemet för användaren uppstår då mängden kunder ökar och gör funktionen svårarbetad. Användaren måste helt enkelt leta igenom hela listan.



Figur 23. En rullgardinsmeny för hämtning av kundposter.

Programkoden (*Figur 24*) inleds med att anslutning till databasen skapas. HTML-kod för systemets meny genereras med hjälp av kommandot ”response.write”. En förfrågan sänds till databasen med begäran om alla kundposter presenterade med kundens ID-nummer (customerId) och kundens namn (name). Systemet skapar utifrån kundposterna ett menyval i rullgardinsmenyn per kundpost. Slutligen avslutas förfrågan och anslutningen till databasen stängs.

```
<%
dim db, rs, sql
set db = Server.CreateObject("ADODB.connection")
db.open "driver={MySQL ODBC 3.51 Driver}; Server=localhost;
Uid=systemLogin; Pwd=xxxx; Database=customerDB"
response.write("Kund:")
response.write("<select name=customerId>")
sql = "SELECT customerId, name FROM customer;"
set rs = server.createobject("adodb.recordset")

rs.open sql, db
do while not rs.eof
response.write("<option name=customerId value='"& rs("customerId")
&"'>"& rs("name") &"</option>")
          rs.movenext
loop
rs.close
set rs = Nothing
response.write("</select>”)
db.close
set db = Nothing
%>
```

Figur 24. Programkod som hämtar kundposter från databasen och placerar dem i en ”dropdown”

Funktionen är relativt enkel att bygga med hjälp av ASP, alla kundposter hämtas från databasen och läggs i menyn. Funktionen genererar dock ett stort informationsutbyte mellan server och klient. I ett fall med 560 kunder i databasen blev storleken på den genererade HTML-koden över 40 kb.

För att förbättra funktionen undersökte vi olika tillämpningar av AJAX-tekniken. Dels för att minska den stora trafikmängden som genereras av ”dropdown-menyn” och dels för att öka användbarheten för användaren. Genom att enbart hämta de kundposter som är aktuella för användaren ökar användarvänligheten och trafiken minskas.

Vi uppnådde detta genom att lägga in ett inmatningsfält i HTML och koppla en JavaScript-funktion till fältet som initieras varje gång en tangent släpps upp i inmatningsfältet (onKeyUp). Inmatningsfältet kompletterades även med ett dolt fält som, när kund har valts, fylls i med det unika id-numret för den valda kunden.

```
<input size=40 name="customerName" type="text"
onKeyUp='doSearch(this.value);'>
<input type=hidden name=”customerId” value=””>
```

Figur 25. HTML-kod för inmatningsfältet, en händelsehanterare vid uppsläppning av en tangent samt ett dolt fält för kund-id.

Funktionen kallar på ett XML-objekt som tillsammans med söksträngen gör en förfrågan till servern. Förfrågan tas emot av servern och ett ASP-skript initieras som ansluter till databasen och utifrån vad användaren angivit söker upp de tio första kunderna som matchar den del av söksträngen användaren hunnit knappa in. Dessa returneras till klienten.

```
function doSearch(strSearchWord)
{
    var xmlHttpReq = false;
    var self = this;
    // Mozilla/Safari
    if (window.XMLHttpRequest) {
        self.xmlHttpReq = new XMLHttpRequest();
    }
    // IE
    else if (window.ActiveXObject) {
        self.xmlHttpReq = new ActiveXObject("Microsoft.XMLHTTP");
    }

    strURL = "getData.asp?action=new&searchWord="+
escape(strSearchWord);
    self.xmlHttpReq.open('POST', strURL, true);
    self.xmlHttpReq.setRequestHeader('Content-Type',
'application/x-www-form-urlencoded');
    self.xmlHttpReq.onreadystatechange = function()
    {
        if (self.xmlHttpReq.readyState == 4)
        {
            var arr = self.xmlHttpReq.responseText.split(',');
            document.getElementById('result').innerHTML =
            htmlFormat(arr);
            document.getElementById('result').style.display =
            'block';
        }
    }
    self.xmlHttpReq.send();
}
```

Figur 26. Programkod som gör sök-förfrågan till servern och tar emot svaret.

Resultatet från servern tolkas genom funktionen htmlFormat (se Figur 27). Funktionen lägger de returnerade kundposterna i en tabell. Händelsehanterarna kopplas även till de olika tabellraderna. Beroende på om muspekaren befinner sig över tabellraden eller inte ändras CSS-klassen på den aktuella tabellraden.

```
function htmlFormat(arr)
{
    var output = '<table class="border" width=200
bgcolor=#ffffff>';
    for (var i=0;i<arr.length;i++)
    {
        output = output + '<tr
onmouseover="this.className=\'mouseOverTR\';"
onmouseout="this.className=\'mouseOutTR\';"
onclick="getData(\''+ arr[i+1] +'\',\''+ arr[i] +'\')">' + '<td>'
+ arr[i] + '</td>' + '</tr>';
        i = i + 1
    }
    output = output + '</table>';
    return output;
}
```

Figur 27. Programkod som bygger upp en tabell i HTML genom ett DOM-objekt.

I vår CSS-mall definierar vi två olika klasser som visas i figuren nedan (Figur 28). När muspekaren placeras över tabellraden färgas bakgrunden grå samtidigt som muspekaren byts ut till en hand för att förtydliga att man kan klicka på raden.

```
.mouseOverTR
{
    background-color:#eeeeee;
    cursor:'hand';
}

.mouseOutTR
{
    background-ccolor:#ffffff;
}
```

Figur 28. *CSS-klasserna som används i tabellen.*

All HTML som tidigare skapades i funktionen htmlFormat sänds till DOM-objektet, placerat precis under inmatningen. Placeringen gör att tabellen upplevs som en meny. När musen förs över menyn markeras den aktuella raden med grå färg. När användaren sedan klickar på raden startar funktionen getData (se Figur 29). Funktionen utför kommandot genom att placera den valda kundpostens id-nummer i den gömda variabeln, och tömmer DOM -objektet. Härigenom försvinner menyn.

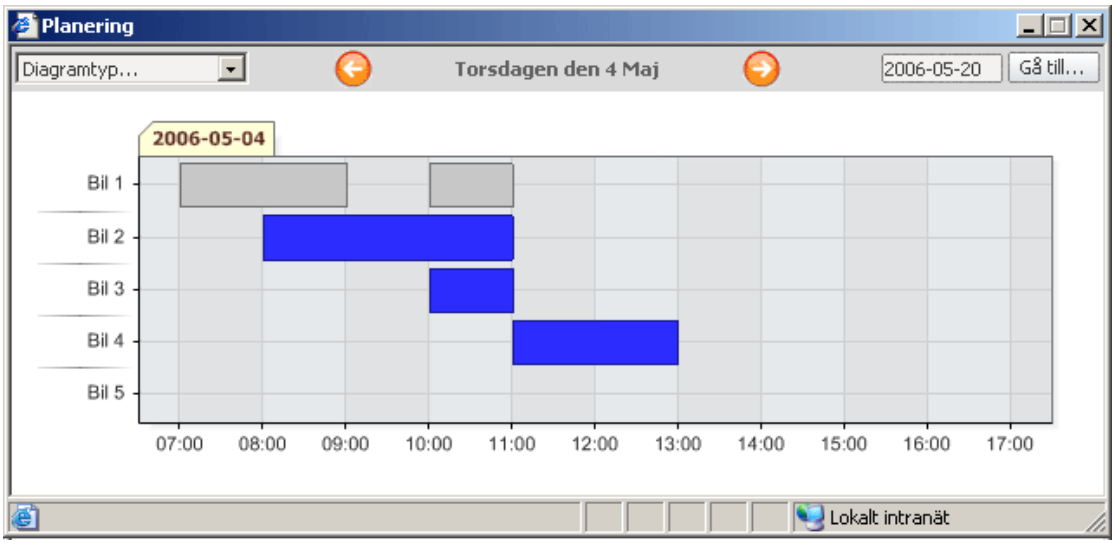
```
function getData(theDataId,theInfo)
{
    if (theDataId != 0)
    {
        //Tar bort menyn
        document.getElementById('result').innerHTML = "";

        //Lägg in namnet i textrutan
        document.searchCustomer.customerName.value = theInfo;

        //Lägg in id-nummret i det dolda fältet
        document.searchCustomer.customerId.value = theDataId;
    }
    else
    {
        //Tar bort menyn
        document.getElementById('result').innerHTML = "";

        //Tar bort namnet i textrutan
        document.searchCustomer.customerName.value = "";

        //Tar bort id-nummret i det dolda fältet
        document.searchCustomer.customerId.value = "";
    }
}
```

Figur 29. Programkod som väljer kund då användaren klickat på denna.

Figur 30. Listningen ses här i en tabell skapad med hjälp av AJAX-teknik

3.2.2 Information med behov av kontinuerlig uppdatering

För ett webb-baserat tidsplaneringssystem med visning av beläggning via gantt-diagram finns behov av kontinuerligt uppdaterad information. I gantt-diagrammet (*Figur 31*) visas flera olika objekt och vilka tider objekten är belagda. I vårt exempel visas beläggningen för fem bilar, alla har olika beläggning över en viss dag. Med traditionell teknik kräver lösningen att hela sidan med jämna mellanrum uppdateras. Diagrammet blir härmed svårläst, uppdateringen tar ca två till tre sekunder. All information måste skickas varje gång sidan skall laddas om, vilket genererar stor datatrafik.



Figur 31. Visar gantt-diagram över bilbeläggning.

```
<meta http-equiv="refresh" content="20">
```

Figur 32. HTML-kod för att uppdatering av hela sidan var 20:e sekund.

Vi skapade en funktion som uppdaterade hela sidan var 20:e sekund (se Figur 32). Klienten sänder en förfrågan till servern som gör beräkningar, sparar ner diagrammet och därefter returnerar detta tillsammans med den HTML-kod som behövs för visningen.

Objekt	Storlek
Bild	~ 30 kb
HTML-kod	~ 16 kb
Summa	**~ 46 kb**

Figur 33. Tabell på överföringsmängd med traditionell uppdatering för gantt-visning.

Med 20 sekunders uppdateringsfrekvens överförs 8-9 Mb data per timma. Detta medför en stor belastning på både server och klient, och är i våra ögon sett ett resursslöseri.

Med detta problem som utgångspunkt försökte vi göra förbättringar med hjälp av AJAX-tekniken. Första tanken var att vi med hjälp av AJAX-teknikens dolda informationslager försökte hitta möjligheter att göra överföringen mindre synlig för användaren. Vidare skissade vi på möjligheter att minimera antalet uppdateringar genom att uppdatera endast då ändringar skett i systemet.

Vi förbättrade lösningen genom att skapa en kolumn i databasen som loggar när senaste uppdatering skett för de olika posterna. Loggposten gör det möjligt att kontrollera om någon uppdatering skett sedan föregående förfrågan.

	workEventId	workOrderId	workGroupId		startDate	startTime	stopDate	stopTime	lastUpdate
1	129	103	3		2006-04-22	08:00	2006-04-22	13:00	2006-04-27 15:14:50
2	130	104	3		2006-04-20	07:00	2006-04-20	08:00	2006-04-28 10:22:02
3	131	104	3		2006-04-20	08:00	2006-04-20	09:00	2006-04-28 10:22:51
4	132	104	3		2006-04-20	09:00	2006-04-20	10:00	2006-04-28 10:25:54
5	133	104	3		2006-04-20	10:00	2006-04-20	12:00	2006-04-28 10:26:31
6	134	104	3		2006-04-20	13:00	2006-04-20	14:00	2006-04-28 10:27:09
7	135	104	3		2006-04-20	14:00	2006-04-20	15:00	2006-04-28 10:27:28
8	137	107	2		2006-04-20	14:00	2006-04-20	16:00	2006-04-28 10:45:03
9	119	87	1		2006-04-20	08:00	2006-04-20	12:00	2006-04-26 10:50:00
10	122	89	2		2006-04-20	13:00	2006-04-20	14:00	2006-04-26 10:50:16
11	116	83	1		2006-04-20	13:00	2006-04-20	16:00	2006-04-20 17:20:00
12	117	84	2		2006-04-20	08:00	2006-04-20	12:00	2006-04-20 17:21:00

Figur 34. Figuren visar databasen där vi lagt till kolumnen ”lastUpdate”. Den innehåller tidpunkten för senaste uppdateringen av en post.

Visningen av gantt-diagrammet förändrades så att de statiska elementen i diagrammet visades med hjälp av HTML-ramar. Då behöver inte de elementen uppdateras, utan enbart diagraminformationen. Genom att använda en JavaScriptsfunktion initieras uppdateringen var tionde sekund. Figur 35 visar denna kod.

```
function checkUpdate()
{
window.setTimeout('generalXMLhttpPost("save.asp?action=checkTimetabl
eUpdate&startDate='+ startDate +'&lastUpdate='+ lastUpdate +'")',
10000);
}
```

Figur 35. *Funktion som var tionde sekund initierar en kontroll.*

Funktionen använder sig av AJAX-teknik för att ansluta till servern och kontrollera om uppdateringar gjorts sedan föregående förfrågan. Om ingen uppdatering är gjord förändras inte diagrammet. Samma fråga ställs igen tio sekunder senare. Har ändringar skett uppdateras diagrammet med all information.

```
function generalXMLhttpPost(strURL)
{
    var xmlHttpReq = false;
    var self = this;

    // Mozilla/Safari
    if (window.XMLHttpRequest) {
        self.xmlHttpReq = new XMLHttpRequest();
    }

    // IE
    else if (window.ActiveXObject) {
        self.xmlHttpReq = new ActiveXObject("Microsoft.XMLHTTP");
    }

    self.xmlHttpReq.open('POST', strURL, true);
    self.xmlHttpReq.setRequestHeader('Content-Type',
'application/x-www-form-urlencoded');
    self.xmlHttpReq.onreadystatechange = function()
    {
        if (self.xmlHttpReq.readyState == 4)
        {
           var result = self.xmlHttpReq.responseText;
           if (result != "0")
           {
                     parent.main.location.reload();
                     lastUpdate = result;
           }
        }
    }
self.xmlHttpReq.send();
checkUpdate()
}
```

Figur 36. Funktion för att skapa en förfrågan till servern med hjälp av AJAX.

4 Resultat

Generellt gör AJAX-tekniken det möjligt att utveckla mer användarvänliga systemfunktioner. I tekniskt avseende är fördelarna främst att mängden data som överförs minskar, även belastningen på serverdatorn minskar. Utveckling med AJAX-teknik ställer högre krav på utvecklare än vid användning av traditionell webbutveckling. Omfattningen på programmeringsarbetet ökar då komplexiteten i en applikation med AJAX-teknik är betydligt större. Dessutom krävs goda kunskaper i flera utvecklingsmiljöer. En AJAX-utvecklare behöver med andra ord bredare kunskaper och ungefär dubbelt så lång tid till sitt förfogande jämfört med traditionell webbutveckling.

4.1 Utvecklaren

För att arbeta med AJAX-teknik krävs kunskap om hela utvecklingsprocessen för att med rimlig tidsåtgång lösa problem som kan uppstå. Utvecklaren behöver utveckla nya tankesätt för felsökning och felhantering. Som exempel finns ett informationslager vilket är dolt för användarens presentationslager. För felsökning vid programmering krävs åtkomst till informationslagret från presentationslagret. Åtkomsten löses genom att programmera så att systemet hela tiden informerar om de frågor som användningen skapar. Sökvägar för de olika händelserna visas i applikationen. De skapade sökvägarna kan kopieras och användas i ett nytt webbläsarfönster för att härigenom finna de fel som uppstått.

Optimering av programkod är dock inte lika väsentlig för en AJAX-utvecklare då det tillkommande informationslagret gör att servern kan utföra kommandon i bakgrunden under tiden användaren arbetar. Detta medför att användaren ofta slipper vänta på svar från servern. Därmed uppfattas applikationerna som snabbare, modernare och mer välprogrammerade.

4.2 Användaren

En webbapplikation byggd med AJAX-teknik upplevs av användaren ha många fördelar jämfört med traditionellt utvecklade webbapplikationer. Dels upplevs applikationen snabbare då all information inte behöver bearbetas när användaren väntar utan mycket kan göras redan innan. Dels upplevs att systemet är aktivt hela tiden och ger snabb respons utan väntetid. Webbapplikationer utvecklade med AJAX-teknik upplevs mera likna vanliga lokala applikationer än webbapplikationer utvecklade med traditionell teknik. Samtidigt har AJAX-applikationen, till skillnad från lokalt installerade applikationer, den stora fördelen att den nås från alla Internetuppkopplade datorer.

4.3 Listning av kunder från databas

Genom att ta bort dropdown-menyn för val av kunder minskade storleken på HTML-koden i vårt exempel markant. Menyn ersattes med ett inmatningsfält för text. Användaren matar in namnet på en kund som eftersöks. En förfrågan startas. Med hjälp av AJAX-teknik hämtar applikationen de kundnamn som matchar de bokstäver som matats in. När önskat kundnamn visas i listan klickar användaren på namnet och kunden väljs. Då sökning sker i alla delar av namnet behöver användaren inte veta exakt kundnamn. Inte heller behöver användaren bläddra igenom en lång lista för att finna kunden. Användaren upplever en enklare hantering i sökningen av kunder.

Trafiken minskade i vårt exempel genom att data som returneras från servern är cirka 36 kb mindre, en minskning med cirka 90 %. Serverns processorbelastning är likartad i båda lösningarna då mängden instruktioner är likvärdig, beräknat på en medelstor sökning som genererar tre olika sökförslag.

4.4 Information med behov av kontinuerlig uppdatering

Med hjälp av AJAX-teknik och uppdateringsinformation i databasen lyckades vi minska trafiken markant. Med den nya lösningen uppdateras informationen enbart när ändringar skett i beläggningsdata. Användaren slipper störande moment då diagrammet uppdateras var 20:e sekund. Både server och klient är betydligt mindre belastade.

Med traditionell metod hanterades ungefär 8-9 Mb/tim per klient, samtidigt gjordes uppdatering var 20:e sekund. Med hjälp av AJAX-teknik har datamängden minskats till ungefär 0.6 Mb/tim. Det störande uppdateringsmomentet har för användaren reducerats med ungefär 98 % då ingen uppdatering sker förrän underliggande data har förändrats. Dessutom är informationen snabbare tillgänglig, då uppdateringsfrekvensen dubblerats, uppdatering sker var tionde sekund. Sifforna är beräknade med antagandet att i snitt görs fyra datauppdateringar per timma i systemet.

5 Slutsats och diskussion

Vår undersökning har visat att fördelarna med användning av AJAX-teknik för både utvecklare och användare är mycket stor. För användaren främst genom stora tidsvinster och ökad användarvänlighet, då applikationen arbetar mera aktivt. Genom att dataöverföringen sker i mindre stycken, under tiden som användaren arbetar i applikationen, upplevs denna vara betydligt snabbare jämfört med traditionellt utvecklade webbapplikationer. Även applikationer som kräver kontinuerlig uppdatering kan med hjälp av AJAX-tekniken effektiviseras och enbart göra uppdateringar då det krävs. Detta upplevs av användaren som en klar förbättring både på grund av minskade störande automatiska bildskärmsuppdateringar och snabbare uppdaterad information.

För utvecklare innebär AJAX-tekniken många möjligheter. Den öppnar nya dörrar och ger förutsättningar för att skapa nya bättre webbapplikationer. Samtidigt ges kreativiteten ett större spelrum då möjligheternas gräns flyttas framåt. Tekniken kräver dock större kunskaper hos utvecklaren. Denne måste behärska alla delar av AJAX väl för att programmera och hantera applikationer på ett effektivt sätt. Då tekniken tillåter ett aktivare informationsflöde måste utvecklaren ha beredskap för tidigare okända problemställningar som kan uppkomma. Arbetet har även visat på svårigheter vid felsökning av AJAX-applikationer. Till en början ett krävande merarbete, men med lite erfarenhet löser den rutinerade utvecklaren även de frågeställningarna.

Slutsatserna styrks med de två konkreta exempel som detta arbete belyst. En minskning av datatrafik med mellan 90-98 % och möjlighet till betydligt mer användarvänliga applikationer gör AJAX till en teknik som kommer att revolutionera webbprogrammeringen och som därmed är här för att stanna.

6 Referenser

[1] North, Simon; Hermans, Paul (1999) *Sams lär dig XML på 3 veckor*. ISBN 91-636-0517-1.

[2] Susning.nu - XML http://www.susning.nu/XML (Acc. 2006-05-11)

[3] Susning.nu - AJAX http://www.susning.nu/AJAX (Acc. 2006-05-11)

[4] Susning.nu http://www.susning.nu (Acc. 2006-05-11)

[5] Susning.nu http://www.susning.nu (Acc. 2006-05-11)

[6] Carl-Johan Nordqvist, *Ajax – ingen ren teknik*, februari 06, 2006 http://internetworld.idg.se/webbstudio/pub/artikel.asp?id=326 (Acc 2006-05-11)

[7] Se Ajax nu - http://www.seajax.nu/ (Acc. 2006-05-11)

[8] Wikipedia – AJAX, http://sv.wikipedia.org/wiki/AJAX (Acc. 2006-05-11)

[9] IBM – Mastering Ajax, http://www-128.ibm.com/developerworks/web/library/wa-ajaxintro2/?ca=dgr-lnxw07AJAX-Request (Acc. 2006-05-11)

[10] Adaptive Path – Ajax, http://www.adaptivepath.com/publications/essays/archives/000385.php (Acc. 2006-05-16)

[11] Lysator – Datorhandbok, http://www.lysator.liu.se/local/datorhandbok/www/www.html (Acc. 2006-05-16)

[12] Webbkunskap.com, http://www.webbkunskap.com/kurser/html/introduktion/introduktion.asp (Acc 2006-05-14)

[13] Susning.nu, http://susning.nu/Active_Server_Pages (Acc 2006-05-16)

[14] Webeye - http://www.webeye.nu/default2.asp?que=asp/default.asp&exec=asp/artiklar/asp1.asp (Acc. 2006-05-16)

[15] Databasteknik.se – Webbkursen, http://www.databasteknik.se/webbkursen/databaser/index.html (Acc. 2006-05-16)

[16] World Wide Web Consortium – CSS, http://www.w3.org/Style/CSS/ (Acc. 2006-05-12)

[17] Cover Pages – DOM, http://xml.coverpages.org/dom.html (Acc. 2006-05-12)

[18] Mitchell, Scott; Atkinson, James (2000) *Sams lär dig Active Server Pages 3.0*. ISBN 91-636-0625-9.

[19] Webbprogrammering.se – JavaScript, http://webbprogrammering.se/javascript (Acc. 2006-05-12)