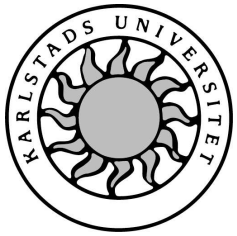


Datavetenskap

Birsen Yilmaz
Haris Drapic

Webbgränssnitt för hantering av BibTeX-databas

Examensarbete

2003:19

Webbgränssnitt för hantering av BibTeX-databas

Birsen Yilmaz
Haris Drapic

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är vårt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Birsen Yilmaz

Haris Drapic

Godkänd, 2003-06-04

Handledare: Stefan Alfredsson

Examinator: Stefan Alfredsson

Sammanfattning

I detta arbete har ett webbgränssnitt mot referensdatabasen Bibliography for TEX (BibTEX) implementerats i PHP:Hypertext Processor (PHP). BibTEX, ett system som hanterar referenser, används tillsammans med LATEX, ett typsättningsprogram, till att skapa dokument. Referenserna som används i ett dokument lagras i BibTEX-databasen via webbgränssnittet. Användaren har möjlighet att utföra diverse operationer på BibTEX-databasen via webbgränssnittet exempelvis lägga till, ta bort, ändra och söka referenser.

Syftet med detta webbgränssnitt var att utöka vissa funktioner som inte förekom i det webbgränssnitt som används för närvarande av forskningsgruppen för distribuerade och kommunicerande system. Dessa funktioner är följande: låsning av referensdatabasen när någon utför en operation, utökning av antalet referenstyper,borttagning, förändring samt automatisk revisionshantering. Webbgränssnittet implementerades med stöd för två språk, svenska respektive engelska.

Abstract

In this project a web interface for Bibliography for TEX (BibTEX) has been implemented in PHP:Hypertextex Processor (PHP). BibTEX, a system which handles references, is used together with LATEX, a typesetting program, to create documents. The references which are used in a document can be put in the BibTEX-database via a web interface. The user has the possibility to perform various operations on the BibTEX-database via the web interface, namely inserting, deleting, modifying and searching references.
The purpose with this web interface was to extend some functions which don't appear in the current web interface used by the research group for distributed and communicating system. These functions are the following: lock the reference database when someone executes a operation, extend the number of reference types and manage automatic revision control. The web interface is available in two languages, Swedish and English.

Innehåll

Figurer

Tabeller

1 Inledning

I de flesta rapporter och avhandlingar används hänvisningar/referenser till tidigare utgivna artiklar. När det blir för många referenser är det med fördelaktigt att hantera dem i en databas (av något slag) i stället för att skriva dem för hand i dokumentet. De blir automatiskt sorterade i önskad ordning, flera dokument kan använda samma referenstext och det blir enklare att ändra ett fel om många referenser används från olika rapporter eftersom dessa finns på ett ställe. Listan med fördelar kan göras lång.

Ett system för att hantera referenser är BibTEX, som används tillsammans med LATEX för att skapa dokument. BibTEX använder en eller flera textfiler som innehåller referenserna. Ett exempel på en sådan fil[1] finns tillgänglig vid forskningsgruppen (DISCO) för distribuerade och kommunicerande system vid ämnet datavetenskap.

Ett generellt problem med att många användare ändrar i samma fil samtidigt, är att förändringar riskerar att försvinna om inte låsning används. Detta gäller även BibTEX-databasen. I dagsläget sker i princip bara tillägg till filen via ett webbgränssnitt, där tillägget sker så snabbt att det förhoppningsvis inte inträffar två skrivningar samtidigt. I undantagsfall när någon detalj ska ändras så sker det sent på kvällen när det antas att ingen annan använder den, eller så koordineras ändringar på gruppmöten. Det nuvarande webbgränssnittet är också något begränsande eftersom det bara tillåter att en viss typ av referens kan läggas till.

Resten av rapporten är indelad enligt följande: I kapitel två ges en introduktion till hur ett webbgränssnitt kan vara uppbyggt i antingen statiskt eller dynamiskt utförande. Sedan beskrivs hur skapandet av dessa sidor kan gå till med hjälp av olika språk. Kapitel tre presenterar LATEX tillsammans med referenshanteringssystemet BibTEX för att ge användaren en insikt över hur dessa två fungerar ihop samt beskriver standard-formatet på BibTEX. I kapitel fyra ges lösningen av konstruktionen, det vill säga olika lösningar diskuteras fram på hur webbgränssnittet skall läggas upp för att åstadkomma så bra användargränssnitt som

[1] <http://www.cs.kau.se/cs/prtp/prtp.bib>

möjligt. Efter att gränssnittet har beskrivits fördjupar vi oss in på användarbeskrivning av applikationen. Här beskrivs hur samtliga operationer genom gränssnittet kan utföras på BibTEX-databasen. I kapitel fem beskrivs designen av implementationen, det vill säga filer som bygger upp applikationen tas upp. I kapitel sex fokuserar vi oss på implementationen av de olika filerna som nämndes i det föregående kapitlet. I kapitel sju ges en beskrivning till hur automatiskt revisionshantering, dubbelinmatning och låsning av filen löstes. För att åstadkomma vissa effekter på vår applikation har Java Applet och JavaScript använts på vissa ställen. En av php-filerna har även implementerats på ett objektbaserat sätt, detta för att illustrera på vilket sätt Object Oriented Programming (OOP) kan kombineras med PHP. Dessa tre (OOP, Java och JavaScript) kommer att behandlas i kapitel åtta.Slutligen kommer slutsatserna i kapitel nio.

2 Webbgränssnitt

I dagsläget finns det olika språk som används för att skapa webbsidor på Internet. Valet av språk beror på vad gränssnittet ska ha för funktionalitet, det vill säga webbgränssnittet är antingen statiskt eller dynamiskt. Detta kapitel beskriver skillnaden mellan statiska och dynamiska webbsidor samt presenterar de språk som används till att skapa de respektive funktionaliteterna. De språk som tas upp i detta kapitel är HyperText Markup Language (HTML), Common Gateway Interface (CGI), Active Server Pages (ASP) och PHP.

2.1 Statiska och dynamiska webbsidor

Tidigare var webbsidor mest statiska informationsplatser som enbart kommunicerade åt ett håll, det vill säga mot användaren. Exempel på statiska webbsidor är HTML-sidor. HTML-sidor är dokument som klienten begär från sin webbläsare, till exempel hämtas från en server och visas för klienten. Att en HTML-sida är statisk innebär alltså att webbläsaren visar webbsidans innehåll och inte dess beteende. HTML är alltså ett sidbeskrivningsspråk,

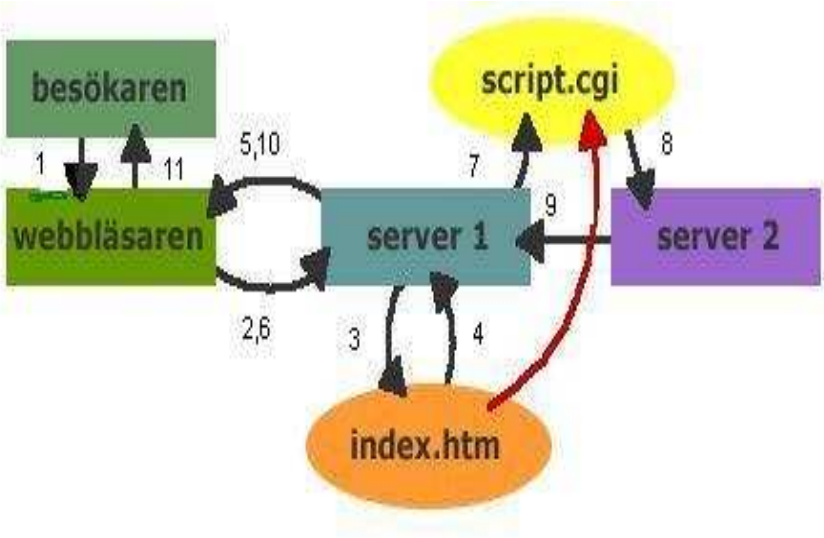
ett markeringsspråk, som kan användas när statiska webbsidor skapas.
Önskas i stället dynamiska, och/eller interaktiva webbsidor så krävs det att HTML kombineras med ett skript- eller programmeringsspråk. Valet av språk beror på vart programmen körs, det vill säga antingen hos klienten (webbläsaren) eller på servern. Vissa av de språk som körs på server sidan använder sig av gränssnittet CGI tillsammans med ett skript. Skript är små program, skrivna i ett programmeringsspråk, som körs direkt i datorn utan att kompileras.
Ett av de språk som används tillsammans med CGI är Perl. Med detta språk är det relativt enkelt att skapa webbapplikationer som till exempel formulärhanterare och besöksräknare. Men Perl är i jämförelse med nyare programmeringsspråk, exempelvis PHP och ASP, långsamt. Detta på grund av att programmet exekveras som ny process för varje förfrågan på webbservern, som leder till att det kan vara mycket resurskrävande för servern. Därför anses Perl av många vara ett utdöende programmeringsspråk för webben. Konkurrerande tekniker till gränssnittet CGI är framförallt ASP, som mest används på Microsofts Windows plattformar, och PHP, som är standarden för Linux-servrar[2].
Exempel på programmeringsspråk som körs hos klienten är JavaScript och Java. Dessa är dock betydligt mindre kraftfulla jämfört med PHP, men har fördelen att de kan användas på vilken webbserver som helst eftersom det är webbläsaren som tolkar och exekverar koden [3].

2.2 CGI - introduktion

CGI är ett standardiserat gränssnitt för att låta en webbserver kommunicera med ett externt program. Kommunikationen sker ofta med hjälp av formulär som användaren fyller i och sedan skickar till en webbapplikation för bearbetning. Till skillnad från HTML så har CGI ett separat program på servern som körs och resultatet av körningen skickas

[2]Mer om detta kan läsas i <http://www.jonasweb.nu/dhtml/cgi.html>
[3]Mer om detta kan läsas i <http://websajt.nu/start.htm>

till klienten som svar på dess förfrågan. Så länge det standardiserade gränssnittet följs, kan CGI-programmet vara skrivet i vilket språk som helst. Antingen tolkas den direkt av webbservern eller kompileras för den. Det i särklass vanligaste språket för CGI-program är Perl, men det förekommer också CGI-program i C, C++, AppleScript, Python, Tcl och Visual Basic med flera. CGI har dessutom den fördelen att det inte spelar någon roll vilken webbläsare klienten använder, eftersom allting utförs på servern. Nackdelen är då att programmen oftast måste vara rätt skräddarsydda för varje server. CGI-program exekveras på själva webbservern och kan vara mycket resurskrävande och kan på så sätt vara en orsak till att en webbserver är långsam.
CGI fungerar enligt följande figur 2.1:



Figur 2.1: Principen för hur CGI-program fungerar

När användaren knappar in adressen (punkt 1) till en hemsida, *index.htm*, söker webbläsaren upp (punkt 2) den server där sidan finns (server 1). Servern returnerar (punkt 3-5) filen och webbläsaren tolkar den. Om det finns bilder på sidan då frågar webbläsaren efter dessa och servern returnerar då rätt bildfiler. På figuren ser man också att *index.htm* pekar på en särskild skriptsida, till exempel för att starta en räknare. När webbläsaren

i HTML-koden stöter på anropet, *script.cgi*, till CGI-programmet anropar den (punkt 6) servern igen och begär att servern skickar tillbaka rätt information. Servern läser då CGI-programmet (punkt 7) för att kunna returnera rätt information till webbläsaren. Denna läsning involverar (punkt 8) ytterligare en server (server 2). I CGI-filen finns nämligen ett kommando, som säger att ett program skall startas och detta program finns på den andra servern. Server2 returnerar (punkt 9) sedan resultatet till den första servern, som i sin tur skickar (punkt 10) resultatet tillbaka till webbläsaren. Besökaren ser (punkt 11) vad som exempelvis kan vara besökarens nummer.

CGI-funktionerna är på det här sättet mycket intressanta och användbara, eftersom de kan starta särskilda program på servern. Funktionerna är därför många fler än att till exempel använda klientbaserade-skript, som körs direkt i webbläsaren. De är också ganska mycket snabbare än till exempel *applets*, eftersom skripten körs oftast på snabbare maskiner än klienten [4].

CGI används framförallt på Unix-servrar, även om funktionen finns på Windows NT och andra system. Filändelserna brukar vara olika beroende på vilket språk som används. Perl ger till exempel filändelserna .pl eller .cgi och C++ ger .cgi eller .exe. Någon tydlig regel finns inte och ändelsen beror väldigt mycket på vad det är för typ av fil. För att kunna använda en CGI-funktion, krävs dels att användaren lägger in HTML-tag samt har tillgång till sökvägen för ett CGI-script.

I HTML-dokumentet kan det till exempel se ut så här:

```
<FORM method="get" action="/cgi-bin/counter.cgi">
```

När användaren sedan klickar på skicka-knappen, ner i formuläret, aktiveras skriptet med *get-metoden* och servern letar upp och läser filen *counter.cgi*.

De flesta webbhotell tillhandahåller liknande sökvägar till gratis skript. Då behöver användaren inte bry sig om att installera själva skriptfilen.

[4] Mer om detta kan läsas i `<http://www.jonasweb.nu/dhtml/cgi.html>`

2.3 ASP - introduktion

ASP är en skriptmiljö som körs på servern. Det används till att skapa interaktiva webbsidor och kraftfulla webbprogram. ASP är en vidareutveckling av CGI, som i stället för att starta ett program varje gång användaren vill se på en sida (vilket kan innebära prestandaproblem i riktigt stora tillämpningar), ligger en tolk inbyggd i själva webbservern som läser ASP-sidan i fråga. ASP-tolken går även att köra som ett vanligt CGI-program om användaren inte kan eller vill integrera den direkt i servern.
ASP-koden bäddas in i HTML-dokumentet som tolkas och bearbetas av webbservern innan resultatet på skriptet skickas till användarens webbläsare. För att webbservern skall stödja ASP krävs särskilda programvaror, det vill säga Microsofts Internet Information Server (Microsofts IIS) och Personal Web Server (PWS).
Dessutom är Unix servrar som webbserver överlägsna NT både i tillförlitlighet, stabilitet och funktionalitet. Vill användaren omedelbart använda sig av ASP är det NT som gäller. Det finns förvisso ASP moduler till Apache som bland annat körs under Unix, men ASP fungerar bäst på en NT/IIS plattform, det är så Microsoft vill ha det[5].

2.4 PHP - introduktion

PHP stod för Personal Home Page och skapades år 1995 av Rasmus Lerdorf. Allteftersom funktionaliteten växte, kom förkortningen snarare att stå för PHP:Hypertext Processor, som innebär att PHP behandlar data innan det blir HTML.
Även PHP är en vidareutveckling av CGI, vilket innebär att den har en inbyggd tolk i själva webbservern som läser PHP-sidan i fråga. PHP är ett HTML-inbäddat, serverbaserat och ett flerplattforms-skriptspråk. PHP kan även beskrivas med termen interpreterande, tolkande, språk. Att PHP är serverbaserat menas att allt som PHP gör händer på servern. PHP har fördelen gentemot traditionella CGI-program att ett PHP-skript är helt portabelt och går att köra på alla servrar som har PHP installerat, det vill säga den är plattform-

[5]Mer om detta stycke kan läsas i <http://websajt.nu/start.htm>

soberoende. De operativsystem som PHP kan köras på är bland annat *Unix*, *Windows NT*, *Macintosh* och *OS/2*. Med ett skriptspråk menas att PHP är designat för att utföra en funktion endast då en handling utförs, exempelvis då en användare skickar ett webbformulär eller går till en Uniform Resource Locator (URL). Till skillnad från programmerinsspråk såsom Java, C och Perl, involveras ett skriptspråk i webbsidorna. PHP skiljer sig från ett skript skrivet i andra språk, som till exempel Perl eller C. I stället för att skriva ett program med många kommandon som i sin tur skriver ut HTML-kod, skrivs i stället HTML med inbäddad PHP-kod som utför något. PHP-koden omsluts av speciella start- och sluttaggar som gör det möjligt att växla mellan PHP-kod och HTML-kod. PHP-koden tolkas och bearbetas av webbservern som i sin tur skickar tillbaka resultatet till användarens webbläsare.

När en användare besöker en webbsida som är skriven i PHP är det inte PHP-koden som användaren får se utan ren HTML. Programtolken på webbservern skriver en helt ny webbsida i traditionell HTML-kod utan PHP-kod när den har bearbetat/exekverat PHP-koden. Användaren kommer därför aldrig att få se någon PHP-kod. PHP-baserade webbsidor behandlas precis som en vanlig HTML-sida. Sidorna kan skapas och ändras på samma sätt som ett HTML-dokument. Det bästa med PHP är att det är oerhört enkelt att lära sig, men ändå erbjuder avancerade funktioner för professionella programmerare. Med PHP har användaren friheten att välja både operativsystem och webbserver. Vidare kan användaren välja algoritm-orienterad programmering eller objekt-oriented programmering, eller en blandning av dem. Även om inte alla OOP-funktioner finns implementerade i den nuvarande versionen av PHP är många bibliotek och större applikationer skrivna endast med hjälp av OOP-kod. PHP används i praktiken till exempelvis diskussionsforum och andra databas-baserade sidor, men kan användas till nästan vad som helst som har med webbutveckling att göra. Dess användning och popularitet ökar kraftigt världen över. Speciellt kraftfullt blir PHP i kombination med databasen *mySQL*. Både PHP och *mySQL* är fria programvaror, program som är gratis att använda och som på egen hand kan vidareutveck-

las. Dessa måste installeras på webbservern och i regel krävs ett dedikerat webbhotell för att stödja det.
Mer om PHP kan läsas i [1, 3, 10, 13, 16, 17].

3 LaTeX och BibTeX

BibTEX, ett program som hanterar literaturreferenser i en text, används tillsammans med textformateringsspråket LATEX för att typsätta dokument. LATEX dokument är baserade på en logisk struktur istället för en visuell struktur, vilket innebär att användaren fokuserar sig på att beskriva strukturen i dokumentet och låter en kompilator sköta all layout.
Kapitlet ger en kort beskrivning av LATEX. Vidare ges en introduktion till formatet för BibTEX genom ett exempel, samt förklaringar till dess referenstyper och fält som kan förekomma i BibTEX formatet.

3.1 LATEX - introduktion

LATEX är ett typsättningsspråk som skapades av Leslie Lamport. Detta språk bygger på TEX som i sin tur skapades av Donald Knuth för att typsätta bokserien *The Art of Computer Programming*. Till skillnad från programverktyg designade för konceptet What You See Is What You Get (WYSIWYG) som till exempel välkända Microsoft Word, så överlåts formateringen av dokumentet till en LATEX-kompilator. Typsättningsspråket är inte alltför olikt HTML. LATEX-koden, såsom HTML-koden, skrivs i ett textredigeringsprogram där fokuseringen sker på vad som ska stå i dokumentet och inte dess layout. Resultatet av dokumentet erhålls genom kompilering som utförs med LATEX-kompilatorn.
Systemet kan i princip användas till alla typer av dokument, men verktyget visar sin stora styrka i samband med vetenskapliga rapporter. Det finns inbyggt stöd för att skapa innehållsförteckningar, korsreferenser och referenslistor. LATEX är gratis och finns så gott som till alla plattformar. En mycket stor fördel är att filformatet är samma för alla plat-

tformar, alltså kan dokument utan större problem flyttas från en Windows-miljö till Unix och vice versa. Mer om LaTeX och BibTeX kan läsas i [2, 4, 7, 8, 9, 11, 14].

3.2 BibTeX - introduktion

BibTeX är ett filformat skapat år 1985 av Oren Patashnik och Leslie Lamport för dokument systemet LaTeX. BibTeX formatet är helt textbaserat, vilket innebär att det kan användas av de flesta texthanteringsprogram. Dessutom är formatet fältbaserat och utbyggbart, det vill säga användaren kan även lägga till sina egna fält.
BibTeX används som ett referenshanteringssystem för LaTeX. Genom att i LaTeX dokumentet ange vilken referensdatabas som ska användas, byggs det upp en referenslista utifrån de referenser som citerats i texten. Detta förenklar arbetet, då användaren bara behöver skriva in information om sina referenser i en databas en gång, och sedan återanvända denna databas för många framtida dokument.
Om användaren vill använda BibTeX i sin .tex fil[6], måste följande två kommandon inkluderas:

```
\bibliographystyle{stil}
\bibliography{bibfil}
```

stil i *bibliographystyle* refererar till style.bst, en fil som tar hand om olika stilar, och *bibfil* i *bibliography* refererar till en .bib-fil (databas).
De standardiserade stilarna är:

- *alpha* - Referenslista är sorterad i alfabetisk ordning. Resultatet av referenslistan erhålls efter att rapporten har behandlats med LaTeX-kompilatorn. Referenserna i rapporten motsvaras av varsin etikett som består av författarens namn samt utgivningsår. Exempel på en etikett är [For95].

[6] .tex är filändelsen på LaTeX dokument

- *plain* - Referenslista är sorterad i alfabetisk ordning. Resultatet är etiketter som är numeriska, till exempel [1].
- *unsrt* - Referenslista är osorterad, det vill säga referenserna visas i referenslistan i samma ordning som de anges i texten. Även här är etiketterna numeriska.
- *abbrv* - Samma som *plain*, men med förkortade referenser.

3.3 Beskrivning av BibTEX formatet

I nedanstående exempel visas formatet på en post, en referenstyp med tillhörande fält, som lagras i BibTEX-databasen (filen).

Exempel på formatet av referenstypen Article

```
@Article {Gettys90,
  author =  {Jim Gettys and Phil Karlton and Scott McGregor },
  title=    {The X Window System, Version 11},
  journal=  {Software Practice and Experience},
  volume =  {20},
  number =  {S2},
  year =    {1990},
  annote =  {A technical overview of the X11 functionality.
  This is an update of the X10 TOG paper by Scheifler & Gettys.}
}
```

3.3.1 Referenstyper som kan lagras i BibTEX-databasen

@Article En artikel från en tidskrift eller magasin.

@Book En bok med en explicit utgivare.

@Booklet Ett verk som är tryckt och bundet, men utan utgivare eller någon institution som sponsor.

@InBook Ett avsnitt i boken, exempelvis ett kapitel och/eller ett antal sidor.

@InProceedings En artikel i en konferens, *proceedings*, en samling av verk som ges ut på konferenser.

@Manual Teknisk dokumentation.

@MastersThesis En magistersavhandling.

@Misc Denna referenstyp används om inget annat stämmer överens.

@PhDThesis En doktorsavhandling.

@Proceedings Publikation med artiklar presenterade vid en konferens.

@TechReport En rapport som en skola eller en institution publicerar.

@Unpublished Ett dokument som har en författare och en titel, men som inte är formellt publicerat.

3.3.2 Fält som kan finnas i de olika referenstyperna

address Vanligtvis utgivarens- eller institutionens adress (skolans adress).

annote En anmärkning.

author Författarens namn.

bibdate Datumet då posten sparades (lades) i databasen.

chapter Kapitlets nummer.

edition Bokens utgåva, till exempel *Second edition.*

institution Institution som sponsrade en teknisk rapport.

ISBN The International Standard Book Number.

ISSN The International Standard Serial Number. Används för att identifiera en tidskrift.

journal Tidskriftens namn. Förkortningar kan användas.

language Språket i vilket dokumentet är skrivet.

location En plats, till exempel staden där en konferens ägde rum.

month Månad då verket är publicerat, eller för ett opublicerat verk, när det är skrivet.

number Numret på tidskriften eller numret på en teknisk rapport. En tidskriftsutgåva är identifierad med en volym och/eller ett nummer. Organisationen som ger ut en teknisk rapport markerar vanligtvis rapporten med ett nummer.

organization En organisation som sponsrar en konferens eller publicerar en manual.

pages Ett eller flera sidnummer eller ett sidintervall, sådana som 42-111 eller 7, 41, 73-97.

publisher Utgivarens namn.

school Skolans namn, där avhandlingen är skriven.

submitter Namn på personen som har lagt till posten i databasen.

title Titel.

type Typ av en teknisk rapport, till exempel *Research Note*.

URL URL adressen av en post.

volume Volym av en tidskrift eller en multivolym bok.

year Utgivningsår, eller för ett opublicerat verk, året när det är skrivet

Vissa fält är utelämnade som exempelvis *pris-fältet* eftersom det ändras hela tiden. Naturligtvis kan även egna fält läggas till. I vårt fall skapades två fält, nämligen fälten *bibdate* och *submitter*. Dessa fält skapades på grund av att användaren skall kunna kontrollera vem som har lagt till en viss post och vilken tid/datum denna post har lagts till i BibTEX-databasen.

4 Beskrivning av konstruktionslösningen

I detta kapitel diskuteras olika lösningar på layouten för webbgränssnittet för BibTEX. Den layout som ansågs vara bäst lämpad har implementerats. Det tas även upp en generell beskrivning om hur layouten på gränssnittet ser ut samt vilka regler som har följts vid dess utförande.
Därefter ges instruktioner till användaren om hur BibTEX-databasen ska användas för att utföra diverse operationer via ett webbgränssnitt.

4.1 Beskrivning av gränssnittets utformning

Till webbgränssnittet har *header* och *footer* som finns på Karlstad Universitets webbsidor valts. Det här valet gjordes på grund av att denna användarmiljö är bekant för målgruppen. Samma gränssnitt kommer att användas genom hela arbetet. Men dessa är bara en fördefinierad (*default*), layout på *header* och *footer* som innebär att det är möjligt för användaren att ändra layouten på dessa. Detta kan göras genom att i php-koden ändra namnen på *header*- och *footer* filen samt ändra filernas innehåll så att layouten blir enligt användarens önskemål.
Till menysidan diskuterades två olika lösningar på hur användargränssnittet kunde se ut. En lösning var att placera ut en knapp för varje operation och en annan lösning var att dela in fönstret i två ramar (frames) med hjälp av en vertikal linje. I lösningen med ramar hade knappar placerats på vänster sida av linjen och högra sidan hade använts till att visa layouten av varje operation (referenstyp). Layouten skulle då bestå av ett formulär för varje operation som bestod av olika inmatningfält beroende på referensens typ. I denna lösning skulle ett stort antal knappar och textrutor placerats på en php-sida. Det skulle leda till att användaren skulle bli tvungen att använda sig av rullningslistan för att kunna se hela sidan. I och med denna nackdel valdes därför lösningen med knappar.
I menysidan valdes en knapp för varje operation. De operationer som går att utföra på

BibTEX-databasen är följande: lägga till (*Insert*), ta bort (*Delete*), redigera (*Change*) eller söka externt efter en referens i den aktiva BibTEX-databasen (*Search_ extern*).
För operationerna insättning respektive borttagning valdes att ha en varsin html-sida. Detta för att undvika att användaren av misstag klickar på fel knapp, vilket skulle leda till att fel operation utfördes. Det är nu inte möjligt att göra detta fel i den valda lösningen. Däremot valdes att ge borttagnings- och redigeringsoperationerna samma söksida, det vill säga samma användargränssnitt. Formuläret i dessa operationer består endast av en textruta där användaren får fylla i en specifik nyckel, som identifierar en post, eller ett sökord. Dessutom övervägdes olika lösningar angående låsning på filen, i vilken BibTEX-databasen placeras. En lösning gick ut på att en *lock* fil skapas när användaren kommer in på någon av sidorna. Därefter skulle filen med BibTEX-databasen kopieras till denna _lock fil. Efteråt skulle rättigheterna på BibTEX-databasen ändras och enbart läsrättigheter skulle ges till användaren. Dessutom skulle också ett skript skapas utanför applikationen som skulle varna användaren om att filen är låst för skrivning. I detta skript skulle det undersökas om det fanns en _lock fil och i så fall skriva ut ett lämpligt meddelande till användaren, om att filen är låst för skrivning"eller något liknande. I slutet, när html-sidorna som användes för inmatning, borttagning eller ändring lämnades, skulle rättigheterna på BibTEX-databasen ändras. Innehållet i _lock filen kopieras till BibTEX-databasen och till slut tas _lock filen bort. I den lösning som till slut implementerades används den inbyggda PHP funktionen *flock* med parametern *LOCK_ EX* som låser filen för alla andra användare förutom för den som är på sidan och utför något. När flera användare är på sidan samtidigt, då har den första användaren skrivrättigheter till filen.

4.2 Layout

Det gjordes ett försök att få sidornas utseende att följa vissa regler. Textrutorna som används oftast är *author*, *key* och *title*. Dessa finns i alla tolv *insert* sidorna och är placerade i, eller nära, formulärets övre vänstra hörn. Detta på grund av att det övre vänstra hörnet

är det ställe som användaren först tittar på. Mer om detta kan läsas i [5, 6, 12].
Ett avstånd gjordes mellan knapparna *Save* och *Main menu* som finns på InsertXX.php filerna för att användaren inte av misstag skulle klicka på fel knapp. Samma sak gäller för knapparna *Search* och *Main menu* på Search.php respektive knapparna *Delete* och *Previous* på Delete.php. På menysidan gjordes ett avstånd mellan de fyra knapparna (*Delete*, *Change*, *Search_extern* och *To start page*) av samma anledning. Textrutor, rullgardinsmenyer och etiketter är placerade i tabeller och på detta sätt åstadkoms symmetri. Informationsmeddelanden skrivs ut i blå färg och felmeddelanden skrivs ut i röd färg. Textstorleken på informationsmeddelandena är lite mindre än textstorleken på felmeddelanden. På detta sätt blir det lättare för användaren att känna igen om det är informationsmeddelande eller felmeddelande. De tolv *insert* knapparna som finns på menysidan tillhör samma grupp, och de är placerade i kontrollpanelen med blå färg.
Den bästa layouten på applikationen erhålls med webbläsarna *Internet Explorer* och *Netscape version 7.0*.

4.3 Beskrivning av gränssnittets funktionalitet

Användaren kommer först till startsidan där användaren kan välja mellan två språk, engelska respektive svenska. Varje språk representeras i sin tur av två olika flaggor, nämligen en Info-flagga och en Meny-flagga. Genom att klicka på någon av två flaggor erhålls antingen information om applikationen eller vidarebefordras till menyn på respektive språk. På infosidan erhålls information om hur BibTEX-databasen ser ut och hur olika operationer kan utföras. De operationer som går att utföra på BibTEX-databasen är följande: lägga till, ta bort, redigera eller söka externt i den aktiva BibTEX-databasen. För att kunna utföra dessa operationer har en knapp för varje operation placerats ut. Om valet är något av de ovanstående möjligheterna så kommer användaren till ett nytt fönster där vald operation utförs. Varje nytt fönster består av ett formulär som användaren får fylla i. Formuläret består av obligatoriska fält och valfria fält. De obligatoriska fälten måste vara ifyllda för

att information ska kunna lagras på BibTEX-databasen. Om kravet inte uppfylls kommer det upp ett nytt fönster med ett tomt formulär och en lämplig text som talar om för användaren om att det inte gick att utföra operationen. De obligatoriska fälten skiljer sig beroende på vilken referens användaren vill lägga till.

4.3.1 Lägga till en post

Om en post skall läggas till, klickar användaren på en av de tolv olika *Insert* knapparna i menyn och blir således omdirigerad till en ny sida. Den nya sidan har ett lättförståeligt namn, på formen InsertXX.php[7]. InsertXX.php sidan består av ett formulär som användaren får fylla i (se figur C.2). I formuläret finns ett fält, nyckel, som måste bli unik för varje post eftersom den identifierar en specifik post. Om den inmatade nyckeln eller den inmatade posten redan existerar i BibTEX-databasen får användaren ett felmeddelande. Detta leder till att ingenting sparas i BibTEX-databasen och användaren uppmanas till att göra ett nytt försök. Samma sak händer i fallet om användaren glömmer att fylla i någon av de obligatoriska fälten. I annat fall får användaren ett meddelande om att den nya posten är tillagd i BibTEX-databasen, samtidigt som användaren får ett tomt formulär med möjligheten att lägga till ytterligare poster. Det finns även möjlighet att gå tillbaka till menysidan genom att klicka på knappen *Main menu*.

4.3.2 Ta bort en post

För att ta bort en post i taget, klickar användaren på knappen *Delete* som är placerad längst ner i menyn bland tre andra knappar (*Change*, *Search_ extern* och *Main menu*). När valet har gjorts omdirigeras användaren till en ny sida där operationen utförs. Den nya sidan består av en textruta och två knappar, *Search* och *Main menu* (se figur C.4). I textrutan skrivs sökordet in, som tillhör den post som användaren vill ta bort. När användaren har skrivit in sökordet, söker programmet igenom följande fält i BibTEX-databasen efter en

[7]XX är namnet på den referenstyp som skall läggas till

förekomst av sökordet:

- *Key*
- *Author*
- *Title*
- *Year*

Följande kan hända när sökningen är avslutad:

1. Sökordet existerar i BibTEX-databasen
2. Sökordet existerar inte i BibTEX-databasen.

I det första av de två fallen ovan visas den post eller de poster som har matchats. Posterna staplas upp efter varandra med varsin radioknapp (se figur C.6). Användaren har då möjligheten att ta bort den önskade posten genom att markera radioknappen och sedan klicka på knappen *Delete*. När en post är borttagen från BibTEX-databasen tas den även bort bland de staplade posterna vilket leder till att alternativen av poster att välja mellan minskar med ett. I fallet där det är endast en post att radera omdirigeras användaren automatiskt tillbaka till söksidan efter borttagningen, vilket leder till att en ny sökning kan göras. Så länge det finns mer än en post att ta bort har användaren möjligheten att fortsätta sin borttagning bland de kvarstående posterna eller också bara välja att avsluta borttagningen med knappen *Main menu*.
I det andra fallet ovan kan användaren antingen försöka söka på nytt eller gå tillbaka till menysidan.

4.3.3 Redigera en post

Om användaren klickar på knappen *Change* så omdirigeras användaren till söksidan. Denna sida har samma användargränssnitt som operationen *'ta bort en post'*, det vill säga den

består av en textruta och två knappar, *Search* och *Main menu* (se figur C.4). I textrutan får användaren ange en nyckel som identifierar en specifik post för att sökningen ska gå igenom. När användaren klickar på knappen *Search* sker en sökning i BibTEX-databasen efter den inmatade nyckeln. Om den inmatade nyckeln är hittad, det vill säga överensstämmer med en nyckel i BibTEX-databasen, öppnas en sida. Denna sida har en textyta med den post som innehåller den matchade nyckeln (se figur C.5). Om användaren däremot skulle klicka på knappen *Main menu* så omdirigeras användaren till menysidan. På sidan som användaren omdirigerades till efter lyckad sökning, får användaren två valmöjligheter. Det ena är att ändra postens innehåll och spara de förändringar som utförts, med knappen *Save Change*. Det andra är att användaren lämnar sidan utan att spara några ändringar med knappen *Previous*. I båda fallen öppnas söksidan som används för att söka efter den önskade posten som användaren vill redigera.

4.3.4 Söka externt efter en post

Om användaren klickar på knappen *Search_extern*, omdirigeras denna till en ny sida där användaren kan utföra sin externa sökning. Användargränssnittet har ett exakt likadant layout med det gränssnitt som *DISCO* gruppen använder sig av i dagsläget (se figur 6.7). På denna sida kan användaren ange ett sökord. Användaren kan välja bland olika sökningskriterium. Med sökningskriterium menas vilket fält som sökningen ska utföras i och och vilken typ sökordet ska matchas med. De olika fälten som finns är *author*, *title* eller alla fälten. De olika söktyper som finns är att hela sökordet, en delsträng eller ett reguljärt uttryck skall matchas. Programmet söker efter det inmatade sökordet i den aktiva BibTEX-databasen som används av *DISCO* gruppen. När användaren klickar på knappen *Submit Query*, sker en sökning efter det inmatade sökordet i den aktiva BibTEX-databasen. Om användaren klickar på knappen *Main menu*, omdirigeras användaren till menysidan.

5 Generell beskrivning

I detta kapitel beskrivs filstrukturen på applikationen och filernas sammanhang, det vill säga hur användaren tar sig från en fil till en annan genom respektive operation. Det ges även en generell beskrivning på InsertXX filernas gränssnitt, det vill säga hur formulären i samtliga InsertXX filer ser ut och vad som skiljer de olika formulären åt.

5.1 Filer som bygger upp applikationen

Applikationen består av följande filtyper:

- PHP-filer
- HTML-filer
- Java Class-filer

Förutom ovanstående filer används JavaScript till *pop-up window(s)* och utskrift på statusfältet. Php- och html-filerna beskrivs mer ingående i nästföljande kapitel.

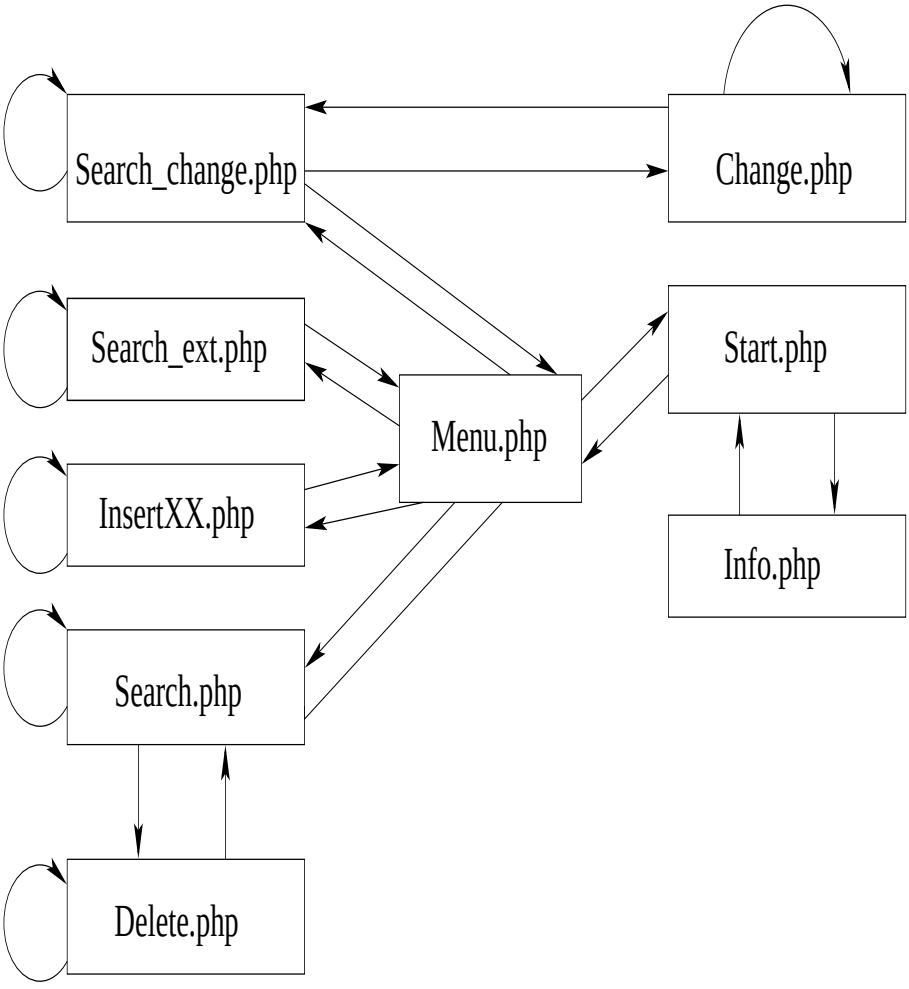
Följande php-filer implementerades:

1. Search.php
2. Delete.php
3. Menu.php
4. Start.php
5. Info.php
6. Flera php-filer för att lägga till en post i bib-databasen

7. Search_ext.php

8. Change.php

9. Search_change.php



Figur 5.1: Schema över applikationens filer - flödesdiagram

Filernas namn valdes omsorgsfullt så att namnen representerar dess funktion. Ett flödesdiagram med alla filer som bygger upp applikationen visas i figur 5.1.

5.2 Beskrivning av informationsfilen

Från startsidan (Start.php) kommer användaren till informationssidan (Info.php) efter sitt val. I denna fil ges en utförlig användarbeskrivning. Denna användarbeskrivning omfattar instruktioner, exempel på hur en referenstyp lagras i BibTEX-databasen samt referenstyper med tillhörande fält förklaras. För ytterligare läsning se informationssidan (Info.php) som hittas under infoflaggorna.

5.3 Beskrivning av menyfilen

Från startsidan (Start.php) kommer användaren till menysidan (Menu.php) efter sitt val. Huvudmenyn placerades i menyfilen där användaren kan utföra diverse operationer på BibTEX-databasen (se figur C.1). För att utföra någon operation på BibTEX-databasen får användaren först klicka på en knapp bland de valbara operationerna och därefter dirigeras användaren om till en php-fil vars namn representerar dess funktion. Samtliga operationer som är möjliga på BibTEX-databasen tas upp mer ingående i de nästföljande avsnitten.

5.3.1 Beskrivning av insättningsfilerna

Som nämndes tidigare så finns det tolv olika referenstyper att välja bland på menysidan (Menu.php). Innan den generella bilden av insättningsfilerna tas upp beskrivs en av de referenstyper som finns, nämligen InsertBook.php. Knappen *InsertBook* används till att lägga till en post av referenstypen bok. Efter att ha klickat på denna knapp omdirigeras användaren från Menu.php till InsertBook.php. Vidare från denna sida kan användaren antingen gå tillbaka till Menu.php eller är kvar på samma sida.
Alla InsertXX.php filer har liknande användargränssnitt. Det som skiljer de olika InsertXX filerna åt är att inmatningsfälten i formulären är olika beroende på vilken typ av referens det är. Vi försökte ge de här sidorna ett standardiserad utseende. Alla formulär i InsertXX.php filerna består av några gemensamma fält (*Common Fields*). *Common Fields*

är egentligen etiketter, textrutor och rullgardinsmenyer som är gemensamma för alla referenser. Därefter placerades ett visst antal referensspecifika fält ut. Antalet referensspecifika fält varierar beroende på vilken referenstyp det gäller. Under de referensspecifika fälten placerades textytan *annote*. Som namnet antyder, är det en textyta där anmärkningar kan skrivas. Den här textytan skulle placeras tillsammans med *Common Fields*, eftersom den också är gemensam för alla referenstyper. Men vi valde att placera den längst ner därför att det gav sidorna ett bättre utseende. Om textytan placeras mittemellan *Common Fields* och referensspecifika fält, skulle den bryta mot regeln att liknande element skulle sitta tillsammans[12]. Några InsertXX.php sidor visas i bilaga C.

5.3.2 Beskrivning av borttagningsfilen

Bland de valbara alternativen i menysidan (Menu.php) kan även operationen ta_bort väljas. Borttagning av en post sker en i taget. När användaren har klickat på knappen *Delete* i menysidan öppnas söksidan (Search.php). Från denna sida kan användaren antingen länkas vidare till borttagningssidan (Delete.php), där borttagning sker eller gå tillbaka till huvudmeny (Menu.php). Länkningen till (Delete.php) sker efter att det sökta ordet på söksidan matchas med något i BibTEX-databasen. Från Delete.php kan användaren antingen välja att gå tillbaka till söksidan (Search.php) eller fortsätta med sin borttagning. När det inte längre finns någon post att ta bort på borttagningssidan, omdirigeras användaren automatiskt till söksidan (Search.php).

5.3.3 Beskrivning av den externa sökningsfilen

Från menysidan (Menu.php) kan användaren välja att gå till sidan Search_ext.php för en extern sökning av posten, med knappen *Search_ extern*, där sökning i den aktiva BibTEX-databasen sker.

5.3.4 Beskrivning av redigeringsfilen

Dessutom finns det möjlighet för användaren att länkas till söksidan Search_change.php, för redigering av posten, från menyn (Menu.php) med knappen *Change record*. Till denna söksida används samma gränssnitt som för operationen ta_bort. Efter att användaren har skrivit in sitt sökord omdirigeras användaren till en ny sida (Change.php) när sökningen har gått igenom. På denna sida gör användaren sina ändringar av posten och sparar den redigerade posten i databasen. När användaren har klickat på *save* knappen sparas den redigerade posten i BibTEX-databasen och användaren dirigeras till Search_change.php. Användaren kan också dirigeras till Search_change.php genom knappen *previous* utan att några ändringar sparas.

6 Ingående beskrivning av html- respektive php-filer

I detta kapitel beskrivs samtliga html- respektive php-filer ingående. Vi går först igenom alla html-filer och beskriver dess innehåll. Php-filerna för varje operation presenteras utförligt genom att beskriva de viktigaste funktionerna som är implementerade i filerna. För att ge en bild över hur funktionerna hänger ihop kommer olika diagram att visas. Några skärmdumpar som visar webbgränssnittet på sidorna är inkluderade. När beskrivningen av insättningsfilerna ges, tas dubbletter, låsning av filen samt automatisk revisionhantering upp, vilka förekommer i insättningsproceduren. Dessa koncept förklaras mer detaljerat i nästföljande kapitel.

6.1 Html-filer som används i applikationen

Alla html-filer placerades i katalogen html. I html-filerna header.htm och footer.htm lagras *header* respektive *footer*. I commonFields.htm lagras de fält som är gemensamma för alla sorters referenser.

De fält som är gemensamma är följande:

- Author
- Key
- Year
- Title
- Language
- URL
- Submitter

Alla dessa fält representeras med varsin länk och en textruta eller med en länk och en rullgardinsmeny.
Förutom dessa, finns även referensspecifika html-filer:

- InsertArticle.htm
- InsertBook.htm
- InsertMastersThesis.htm
- InsertPhDThesis.htm
- InsertInProceedings.htm
- InsertManual.htm
- InsertTechReport.htm
- InsertProceedings.htm
- InsertUnPublished.htm
- InsertBoklet.htm

- InsertInBook.htm

Alla ovanstående html-filer har också en motsvarighet på svenska. Det som skiljer sig är tillägget _s på den engelska. Exempelvis blev då den svenska motsvarigheten till InsertBoklet.htm i stället InsertBoklet_s.htm. Samma sak gäller för .php filerna. Dessutom, finns det två html-filer som ger information till användaren. Dessa två filer är, Info.htm och Info_s.htm, som tillhandahåller en bruksanvisning till användaren om hur de olika referenstyperna är uppbyggda, samt vilka fält som existerar i BibTEX-databasen. Förutom dessa två filer finns det ett antal Info_XX[8].htm filer som förklarar innebörden av ett specifikt fält. Alla Info_XX.htm filer används som *pop-up window(s)*.

Referensspecifika html-filer omfattar etiketter, textrutor och rullgardinsmenyer för de fält som är specifika för en viss referenstyp. Dessutom finns en annote.htm där en etikett, *Annote:*, en textyta och två knappar lagras. Dessa knappar är *save* för att spara posten i BibTEX-databasen och *main menu* för att återgå till huvudmeny.

Submitter fältet och *year* fältet som finns i commonFields.htm filen förekommer även i separata filer, nämligen i submitted.htm och year.htm. I dessa lagras etiketter och rullgardinsmenyer för *submitter* fältet och *year* fältet. Dessa filer kommer till användning då användaren har matat in nyckel som redan finns i BibTEX-databasen. Valet i rullgardinsmenyerna får då väljas om på nytt och användaren får mata in en ny nyckel. Resterande inmatad text valdes att behålla kvar, detta på grund av att spara tid åt användaren som då inte behöver fylla i textrutorna på nytt.

Eftersom både *submitter* fältet och *year* fältet finns i två olika filer vardera måste ändringar av dessa göras på båda ställena. Exempelvis om ett nytt årtal ska läggas till i rulgardinsmenyn måste detta göras i två olika filer, nämligen i commonFields.php och year.htm filerna.

Användaren måste alltså ändra på två olika ställen beroende på vad som skall ändras:

- commonFields.php

[8]XX står för namnet på ett specifikt fält

- submitter.htm eller year.htm

I den svenska delen är denna funktionalitet borttagen, men då räcker det med att ändra på commonFields.php.
Alla html-filer som nämndes läses med hjälp av följande php-kod:

```
function printHeaderFooterPage($filename) {

    if (!$fp = fopen($filename, 'r'))
    {
        print "Cannot open file ($filename)";
        exit;
    }
    while(!feof($fp))//till slutet av filen
    {
        $row=fgets($fp,1024);//en rad i taget
        print $row;
    }
    fclose($fp);
}
```

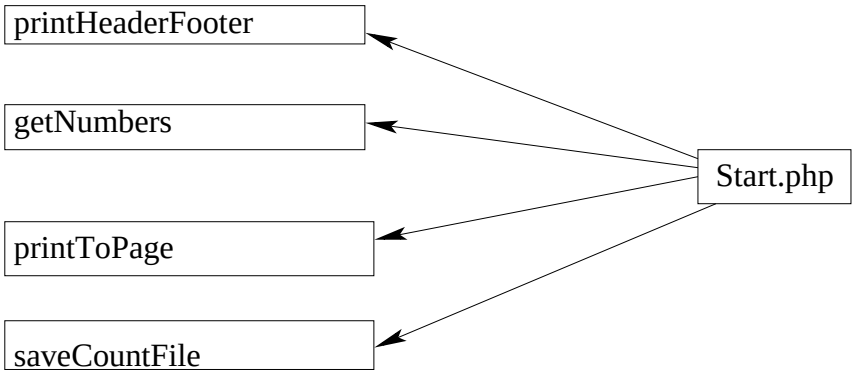
Angående html-filerna övervägdes också ett annat alternativ. Alternativet var då att skriva ut html-koden med hjälp av *print* eller *echo* kommandona och inte använda separata html-filer. All html-kod skulle då placeras i php-filerna. För att förenkla förändring valdes det första alternativet. Inga förändringar av php-koden behöver då göras om till exempel om ett år skall läggas till, eller om *header* och *footer* ska ändras.

6.2 Php-filer som används i applikationen

6.2.1 Beskrivning av Start.php

Start.php är, som själva namnet antyder, applikationens startsida. Funktionerna som implementerades i startsidan visas i figur 6.1
De fyra funktionerna som implementerades i start.php är följande:

1. *printHeaderFooter* - skriver ut header och footer



Figur 6.1: Funktionerna som implementerades i Start.php

2. *getNumbers* - extraherar enstaka siffror från ett femsiffrigt tal

3. *saveCountFile* - sparar antalet besökare i filen

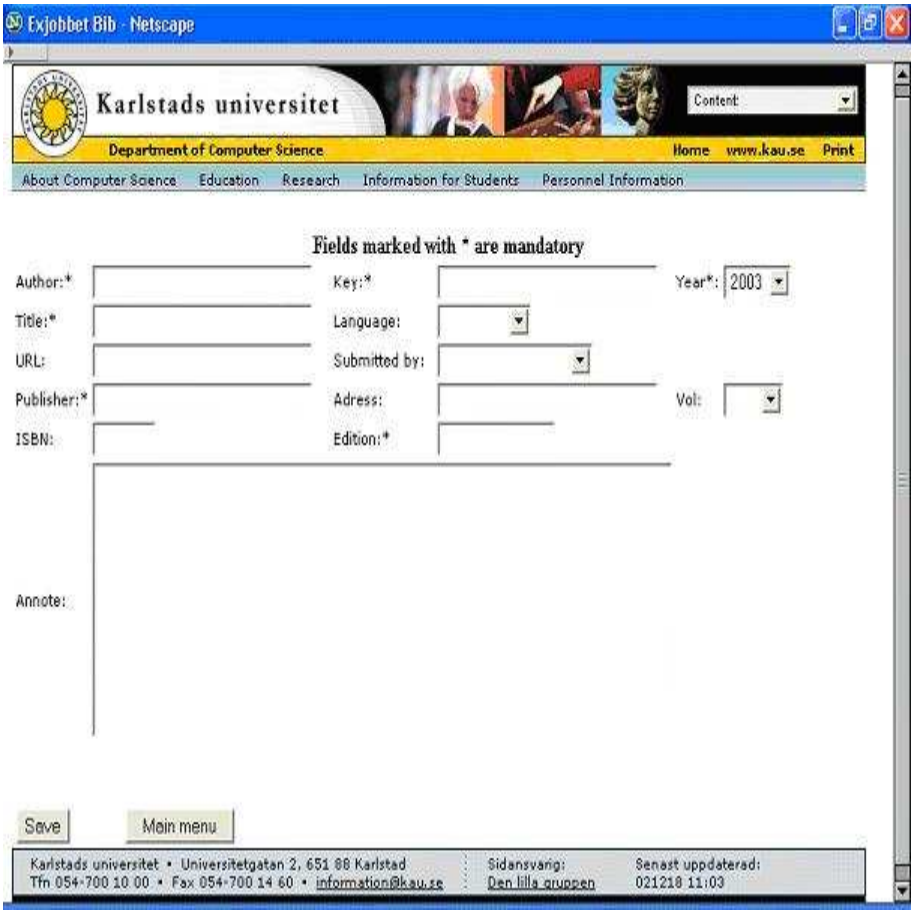
4. *printToPage* - skriver ut sidan, det vill säga allt förutom *header* och *footer*

På startsidan finns, förutom *header* och *footer* en rullande textremsa, fyra symbolflaggor och en räknare som räknar antalet besökare. Textremsan är en Java *Applet* och en mer omfattande beskrivning av *appleten* ges i avsnitt 9.2. Själva hastigheten på textremsan (hastigheten med vilken text rullar på remsan), textstorleken och färgerna kan lätt justeras i källkoden för Java. Omkompilering krävs om några förändringar görs.
JavaScript användes för att åstadkomma vissa effekter på startsidan såsom en rullande text på statusfältet.

6.2.2 Insättning av en post

Den första lösningen, och även den valda, innefattade tolv olika InsertXX.php filer vilket innebar tolv olika knappar. Sidorna liknar varandra ganska mycket men skiljer sig i antalet specifika fält, det vill säga antalet parametrar som funktionerna har. En andra lösning övervägdes också, där alla tolv filerna skulle slås ihop till en enda fil där en *hidden* parameter skulle skickas från menysidan. I denna lösningen skulle referenstypen väljas från en

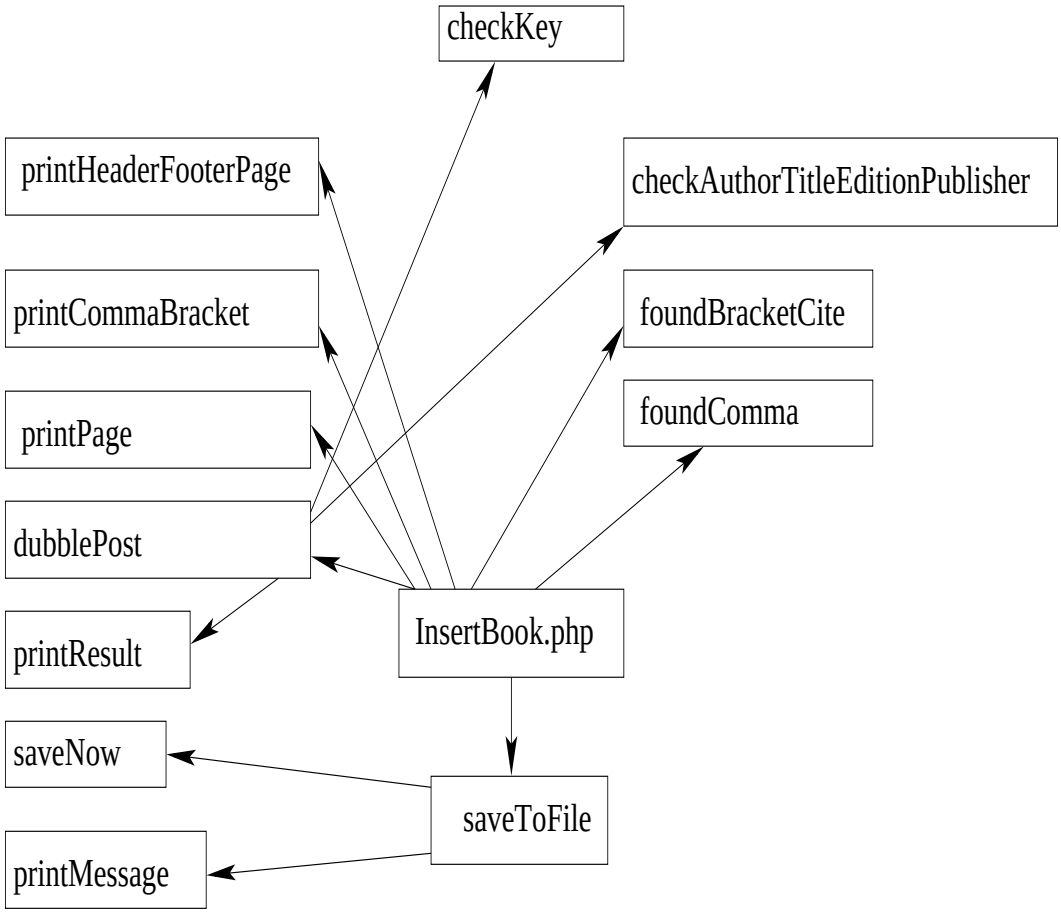
rullgardinsmeny. Denna skulle finnas på menysidan vilket kanske skulle vara en liten förbättring i jämförelse med tolv knappar i lösning ett. En tredje lösning diskuterades också. Denna lösningen gick ut på att sidan skulle laddas om automatiskt eller direkt genom en rullgardinsmeny. Men det visade sig att de två sistnämnda lösningarna skulle bli mer komplexa än den första. Lösning ett valdes då eftersom att de andra två lösningarna är mindre strukturerade på grund av att insättningen av alla referenstyper sker på en och samma sida. Den valda lösningen med tolv olika filer minskar komplexiteten och programmet blir lättare att överblicka samt att ändra, vilket leder till att läsbarheten ökar. En skärmdump av InsertBook.php visas i figur 6.2.



Figur 6.2: Användargränssnittet för InsertBook.php

I alla tolv filerna utförs först kontroll av dubbelinmatning med hjälp av funktionen *dubblePost* som visas här under. Antalet parametrar i denna funktion varierar beroende på vilken referenstyp det handlar om.

```
function dubblePost($author,$title,$key,$number,$journal,$filename,
                    &$dubble,&$key_dubble)
```



Figur 6.3: Funktionerna som implementerades i en InsertXX.php

I funktionen *dubblePost* läses en rad i taget från filen. Inläsningen görs med den fördefinierade funktionen *fgets*. Referenstypen extraheras om den nya posten börjar med följande mönster, *ref = @referenstyp*. När referenstypen matchas med mönstret sätts alla booleska variabler till falskt (markerad med 1 i figur 6.4). Variablerna motsvarar fälten som tillhör

en specifik referenstyp. Dessa avgör om en post i BibTEX-databasen är identiskt med de inmatade fälten.
Därefter används den fördefinierade funktionen *strtok* för att extrahera nyckeln från filen. *Trim*, en fördefinierad funktion, används till att ta bort eventuella blanktecken som kan finnas i nyckelns början eller slut. Detta gjordes därför att användaren kanske av misstag råkar lägga till några blanktecken på dessa ställen när användaren manuellt redigerar i filen. För att jämföra den inmatade nyckeln med den som finns i filen används den fördefinierade funktionen *strcmp*. En temporär variabel, *temp_ref*, har använts för att lagra referenstypen temporärt tills den nya posten börjar. Denna variabel behövs eftersom typen på referensen, *$ref*, ändras hela tiden (2 i figur 6.4).
Därefter, för en viss referenstyp, extraheras fältets namn från dess värde och även här används funktionen *trim* för att ta bort eventuella blanktecken från värdet av samma anledning som med nyckeln (här även mer rättfärdigat). Värdena av vissa fält från en post i filen jämförs (3 i figur 6.4) med de inmatade värdena. Funktionen *checkX* anropas, där X är namnen på de fält som jämförs. Vilka fält som undersöks beror på referenstypen (se dubbelinmatning avsnittet i nästa kapitel).
I slutet av denna (4 i figur 6.4) funktion skrivs varning(ar) ut i tre olika fall:

1. Om inmatad post är en dubblett
2. Om nyckeln i den inmatade posten är en dubblett, det vill säga den inmatade nyckeln redan finns i BibTEX-databasen
3. Om alla obligatoriska fält inte är ifyllda och om användaren har matat in en post eller nyckel som redan finns i BibTEX-databasen

Utskrift av varningarna sker i funktionen *printResult*. De funktioner som implementerades i en InsertXX.php visas i figur 6.3.

Om följande villkor är uppfyllda anropas funktionen *saveToFile*:

```
function dubblePost($author,$title,$key,$number,$journal,
                    $filename_database,&$dubble,&$key_dubble)
{
   if (!$fp = fopen($filename_database, 'r'))//öppna filen för läsning
   {
       print "Cannot open file ($filename_database)";
       exit;
   }
   while(!feof($fp))//till slutet av filen
   {
       $row=fgets($fp,1024);//en rad i taget eller 1024 tecken
       $ref=strtok($row,"{");//extrahera referenstyp

       if((strcmp($ref,"@Book")==0 )||(strcmp($ref,"@Article")==0)||
          (strcmp($ref,"@InProceedings")==0)||(strcmp($ref,"@MastersThesis")==0)||
          (strcmp($ref,"@Misc")==0)||(strcmp($ref,"@Manual")==0)||
          (strcmp($ref,"@TechReport")==0)||(strcmp($ref,"@PhDThesis")==0)
          ||(strcmp($ref,"@Proceedings")==0)||(strcmp($ref,"@InBook")==0)||
          (strcmp($ref,"@Booklet")==0)||(strcmp($ref,"@Unpublished")==0))
       {
          $author_title_same=false;//------------------------------(1)
          $author_same=false;
          $author_title_journal_same=false;
          checkKey($temp_ref,$key,$key_dubble,$ref);//-------------(2)
       }
       if(strcmp($temp_ref,"@Article")==0)//om den handlar om en
                                          //viss referenstyp
       {
          $field_name=strtok($row," ");//Extrahera fältets namn
          $token_second=strtok("{");//Ta andra token, från slutet av
                                    //fältets namn till{
             $field_value=strtok("}");//Extrahera fältets värde
             $field_value=trim($field_value);//Ta bort eventuella blanktecken i
                                             //början och slutet
             checkAuthorTitleJournal($field_name,$field_value,$author,$title,
                                     $author_same,$author_title_same,
                                     $dubble,$journal);//----------(3)
          }
       }
       fclose($fp);
       printResult($key_dubble,$dubble,$key,
                   $journal,$title,$author,$number);//--------------(4)
 }
```

Figur 6.4: Php-koden för dubblePost

- Att användaren inte har matat in nyckel eller post som redan finns i BibTEX-databasen
- Att det inte finns ',' i den inmatade nyckeln[9]
- Att det inte finns } eller " i *title* eller *year* fälten[10]

I denna funktion öppnas filen för skrivning (markerad med 1 i figur 6.5).
All inmatad information sparas i slutet av filen om:

- Alla obligatoriska fält är ifyllda
- Nyckeln är inte längre än ett ord, det vill säga inga mellanslag får förekomma

Flock, en fördefinierad funktion, tar hand om samtidig åtkomst till filen. *flock* tas upp mer ingående i avsnitt 8.2.
Därefter undersöks om alla obligatoriska fält (2 i figur 6.5) och nyckeln är ifylld. Nyckeln får bestå av max ett ord, alltså inga mellanslag är tillåtna (3 i figur 6.5). Automatisk revisionshantering görs med hjälp av kommandot *ci -l filens namn*. Vi tar upp automatisk revisionshantering mer ingående i avsnitt 8.3. Lämpliga meddelanden skrivs ut beroende på den inmatade nyckelns- och de obligatoriska fältens innehåll (4 i figur 6.5).
Hur alla anrop hänger ihop kan ses i figur 6.3. Anropen mellan funktionerna visas med hjälp av pilar, till exempel funktionen *saveToFile* anropar funktionerna *printMessage* och *saveNow*. Dessutom finns det länkar bredvid alla textrutor och rullgardinsmenyer. En länk tillsammans med en textruta eller en rullgardinsmeny, representerar ett fält i BibTEX-databasen. Om användaren klickar på någon av de länkarna som finns, öppnas ett *pop-up window* som innehåller information om det fältet. Detta utfördes med hjälp av JavaScript som beskrivs mer utförligt i avsnitt 9.3.
URL-adresser av de olika artiklarna ändras ganska ofta. Men om de olika artiklarna sparas

[9]För att undersöka detta anropas *foundComma* - se bilaga A
[10]För att undersöka detta anropas *foundBracketCite* - se bilaga A

```
function saveToFile($author,$year,$title,$filename_database,$key,$adress,
                  $journal,$annote,$vol,$sub,$number,$issn,$month,$url)
{
      $saved=false;   //om posten är sparad=true
      $key_two=false; //om nyckeln består av 2 ord=true
      if (!$fp = fopen($filename_database, 'a'))//-----------------(1)
      {
            print "Cannot open file ($filename_database)";
            exit;
       }
       flock($fp,2);//filen låses
       $today=date("F j,  Y,  g:i a");//aktuellt datum och tid
       $mand_fields_filled=false;//om alla obligatoriska fält är fyllda
       //{ eller " är hittat i author eller titel

       if((strcmp($author,"")!=0)&&(strcmp($title,"")!=0)&&
          (strcmp($key,"")!=0)&&(strcmp($journal,"")!=0)&&
          (strcmp($year,"")!=0)&&(strcmp($number,"")!=0))//--------(2)
       {
              $mand_fields_filled=true;
              $key_1=strtok($key," ");
              $key_2=strtok("");
              if(!$key_2)//---------------------------------------(3)
              {
                 //allt inmatad information skrivs i filen
                 saveNow($author,$year,$title,$fp,$key,$adress,
                         $journal,$annote,$vol,$sub,$number,$issn,
                         $month,$url,$today);
                 $saved=true;

              }
              else
              {
                 $key_two=true;
              }
       }
       flock($fp,3);//filen låses upp
       printMessage($saved,$key_two);//----------------------------(4)
       fclose($fp);
       if($saved)
       {
          system("ci -l ".$filename_database);//automatisk revisionshantering
       }
}
```


Figur 6.5: Php-koden för saveToFile

i en lokal katalog skulle det underlätta för användaren om URL-fältet innehöll filens namn istället för URL-adressen. För att åstadkomma detta lade vi ut ett *file element* i InsertArticle.php. *File element* är en kombination av en textruta och en *browser* knapp. Denna används oftast till filuppladdningar men i vår applikation används den till att spara filens namn. Användaren kan alltså med hjälp av *file element* spara artikelns namn i BibTEX-databasen.

```
<input name="url" type=file>
```

För att *file element* skulle funka, adderades också *enctype* i formuläret.

```
print"<FORM  name=\"form1\"  action=\"".$_SERVER["PHP_SELF"]."\"
method=post enctype=\"multipart/form-data\">";
```

Till en början skapade sparandet av artikelns namn problem för *browsern*. I stället för att filens namn sparades, så sparades namnet av en temporär fil i BibTEX-databasen. Men efter noggrann genomläsning av *PHP referens dokumentation*[11] kom vi fram till att det inte var tillräckligt med det skrivsätt som vi har visat ovan, det vill säga *input name="url"*.
Istället för att använda variabeln direkt, så måste $url skrivas på följande sätt för att komma åt filens namn:

```
$_FILES['url']['name']
```

6.2.3 Borttagning av en post

De funktioner som implementerades i Delete.php visas i figur 6.6.
Om användaren klickar på knappen *Search* söker programmet efter det inskrivna ordet i BibTEX-databasen.
Sökningens resultat kan vara något av följande fem alternativ:

1. Det inskrivna ordet är en nyckel

[11] <http://www.php.net>

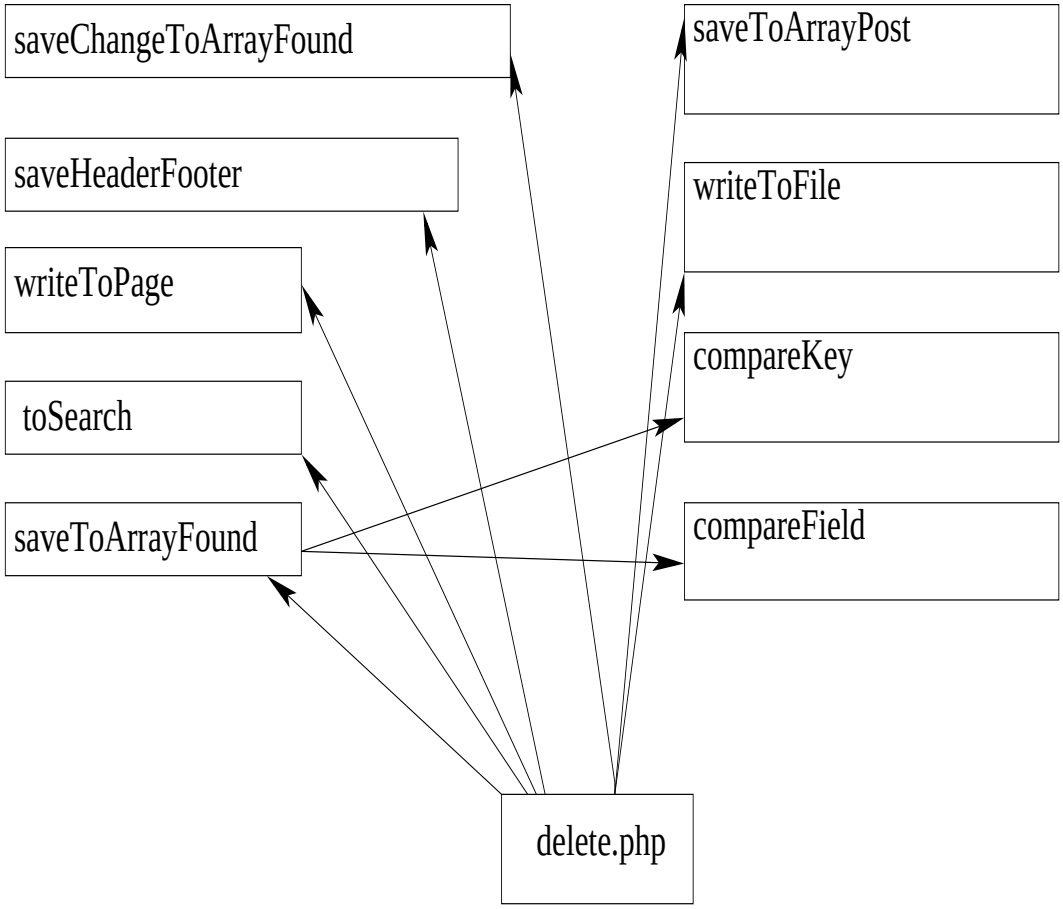
2. Det inskrivna ordet är ett ord i *author* fältet

3. Det inskrivna ordet är ett ord i *titel* fältet

4. Det inskrivna ordet är ett årtal

5. Det inskrivna ordet är inte något av de fyra ovanstående fallen

Sökningen utförs i hela BibTEX-databasen, det vill säga det söks i alla poster oberoende av vilken referens en viss post har.



Figur 6.6: Funktionerna som implementerades i Delete.php

Om sökordet inte hittas, öppnas söksidan Search.php på nytt med ett lämpligt meddelande, exempelvis "*Nothing found*", eller motsvarande på svenska. Detta görs genom anrop till

funktionen *toSearch* med hjälp av Search.php sidan öppnas automatiskt.

```
function toSearch($search,$delete)
   {
      print"<!doctype html public \"-//W3C//DTD HTML 4.01//EN\">
            <html>
              <head>
                 <title>redirect</title>
                 <meta http-equiv=\"refresh\" content=\"0;
                 URL=Search.php?search=".$search."&delete=".$delete."\">
              </head>
            <body>
              <p>
                If you aren't forwarded to the search page
                <a href=\"Search.php?search=".$search."&delete=".$delete."\">
                click here</a>.
              </p>
            </body>
            </html>";
   }
```

Om sökordet hittas lagras posten, i vilken sökordet är hittat, i variabeln *array found*. Det som lagras i variabeln är följande, nämligen fälten *author, title, year* och referenstypen. Allt detta utförs i funktionen *saveToArrayFound* som visas här nedan.

```
function saveToArrayFound($key,$filename_database,&$found,&$number_post_found)
{
        if (!$fp = fopen($filename_database, 'r'))//filen öppnas för skrivning
        {
             print "Cannot open file ($filename_database)";
             exit;
        }
        while(!feof($fp))//till slutet av filen
        {
           $row=fgets($fp,1024);//en rad i taget
           $ref=strtok($row,"{");//ta första token
           $field_name=strtok($row," ");//extrahera fältets namn
           if((strcmp($ref,"@Book")==0 )||(strcmp($ref,"@Article")==0)||
              (strcmp($ref,"@InProceedings")==0)||(strcmp($ref,"@MastersThesis")==0)||
              (strcmp($ref,"@Misc")==0)||(strcmp($ref,"@Manual")==0)||
              (strcmp($ref,"@TechReport")==0)||(strcmp($ref,"@PhDThesis")==0)||
              (strcmp($ref,"@InBook")==0)||(strcmp($ref,"@Unpublished")==0)||
```

```
            (strcmp($ref,"@Proceedings")==0)||(strcmp($ref,"@Booklet")==0))
        {
            $found_key=false;   /*I början av posten sätts de alla till
                                falsk,det vill säga ingenting är hittad*/
            $found_author=false;
            $found_title=false;
            $found_year=false;
            $ref=strtok($row,"{");
            compareKey($ref,$key,$found_key,$number_post_found,$found,$temp_found);
        }
         //sökordet letas i tre olika fält(author,year och titel)
        else if(strcmp($field_name,"author")==0)
        {
                compareField($found,$number_post_found,$found_author,
                             $found_key,"\tAuthor: ",$temp_found,"",$key);
        }
        else if((strcmp($field_name,"title")==0))
        {
                compareField($found,$number_post_found,
                            $found_title,($found_key||$found_author),
                            "\tTitle: ",$temp_found,"",$key);
        }
        else if((strcmp($field_name,"year")==0))
        {
                compareField($found,$number_post_found,$found_year,
                            ($found_key||$found_author||$found_title),
                            "\tYear: ",$temp_found,"",$key);
        }
      }
      fclose($fp);
}
```

Erhållna resultat av sökningen visas med hjälp av funktionen *writeToPage* (se bilaga A). Om någon av radioknapparna markeras och användaren klickar på knappen *Delete* som finns i Delete.php, tas detta element bort från arrayen *found* och motsvarande post tas bort från filen. Den första delen, borttagning av element från *array found*, utförs i funktionen *changeArrayFound* (se bilaga A).

Den andra delen, borttagning av posten från filen, utförs i funktionen *saveToArrayPost* som visas här under:

```
function saveToArrayPost(&$post,&$number_post,$resultat,$filename_database)
{
   if (!$fp = fopen($filename_database, 'r'))
   {
         print "Cannot open file ($filename_database)";
         exit;
   }
   $hit=false;
   while(!feof($fp))//till slutet av filen
   {
       $row=fgets($fp,1024);//en rad i taget
       $referens=strtok($row,"{");//extrahera referenstyp
       $key_dat=strtok(",");//extrahera nyckeln
       $key_dat=trim($key_dat);//eventuella blanktecken bort
       if((strcmp($referens,"@Book")==0 )||(strcmp($referens,"@Article")==0)||
          (strcmp($referens,"@InProceedings")==0)||
          (strcmp($referens,"@MastersThesis")==0)
          ||(strcmp($referens,"@Misc")==0)||(strcmp($referens,"@Manual")==0)||
          (strcmp($referens,"@TechReport")==0)||(strcmp($referens,"@PhDThesis")==0)
          ||(strcmp($referens,"@InBook")==0)||(strcmp($referens,"@Unpublished")==0)||
          (strcmp($referens,"@Proceedings")==0)||(strcmp($referens,"@Booklet")==0))
       {
           $hit=false;//för varje ny post $hit=false,det vill säga nyckeln är inte hittat
           $number_post++;//starta ny post
           if(strcmp($key_dat,$resultat)==0)//hittat nyckel
           {
                   $number_post--;//den posten raderas
                   $hit=true;     //nyckeln är hittat
           }
           else
           {       /*Om nyckeln är inte hittat,lägg till raden i array post*/
                   $post[$number_post]=$row;
           }
      }
      else if(!$hit)
      /*Om nyckeln(posten) är inte hittat, då fortsätt
        att lägga till rader i arrayen post*/
      {
                   $post[$number_post]=$post[$number_post].$row;
      }
   }
   fclose($fp);
}
```

6.2.4 Extern sökning

Användargränssnittet som används till extern sökning visas i figur 6.7.

Figur 6.7: Användargränssnittet för extern sökning

Det skapades också en php-sida för sökning i den aktiva BibTEX-databasen. Om användaren klickar på knappen *Search extern*, öppnas sidan search_ext.php. Ett ord kan matas in, samt olika sökningkriterium kan väljas. Det inmatade ordet kan antingen sökas i alla fält, eller enbart i *author*- eller *title* fälten. Sökordet kan antingen vara en delsträng, det vill säga en del av ett ord, ett helt ord eller ett reguljärt uttryck. Sökningen utförs med CGI-filen.

```
<form action="http://www.cs.kau.se/cs/prtp/bibsearch.cgi" method=post>
```

6.2.5 Redigering av en post

Från menysidan kan användaren komma till Search_change.php efter att ha klickat på knappen *Change*. På den här sidan matas nyckel av posten som skall redigeras. Användaren kan också lämna den här sidan och gå tillbaka till menysidan genom knappen *Main menu*. När nyckeln har matats in och användaren har klickat på knappen *Search*, öppnas sidan Change.php med posten som innehåller den inmatade nyckeln. Om det inte finns någon post som innehåller den inmatade nyckeln, öppnas Search_change.php. För att åstadkomma detta används samma funktion som hos Delete.php, nämligen funktionen *toSearch*, som används för automatisk omdirigering. Den ändrade posten sparas i filen efter att användaren har klickat på knappen *Save Change*. Sidan kan även lämnas utan att någon ändring utförs i filen, genom knappen *Previous*.

Första gången när php-koden exekveras, sätts variabeln *search=true*, och programmet letar efter den inmatade nyckeln i filen. Om den inmatade nyckeln hittas i filen, *post_found=true*, skrivs posten som innehåller nyckeln i en textyta. För att åstadkomma detta, används funktionen *writeToPage*. Om nyckeln inte hittas öppnas Search_change.php med hjälp av funktionen *toSearch*. Efter de gjorda förändringarna och verifiering genom knappen *Save Change*, *save_change=true*, anropas följande funktioner: *saveToArrayPost,writeToFile* och *addNewPostToFile*, för att spara de gjorda förändringarna i filen.

```
if($search)
{
       //om posten med inmatat nyckel är hittat eller inte
       $post_found=false;
       //spara posten(om den hittas) i variabel resultat
       saveResult($key,$filename_database,$post_found,$resultat);
       if($post_found!=0) //hittat någonting
       {
           $search=false;
           printHeaderFooter($filename_header);
           writeToPage($search,$resultat,$key);
           printHeaderFooter($filename_footer);
       }
       else
```

```
        {
            toSearch($search,$save_change);
        }

}
else if($save_change)
{
        $number_post=0;
        //den gamla posten tas bort från filen
        saveToArrayPost($post,$number_post,$key,$filename_database);
        writeToFile($post,$number_post,$filename_database);
        //den nya(redigerade) posten sparas i array
        addNewPostToFile($record,$filename_database);
        toSearch($search,$save_change);
}
```

7 Dubbelinmatning, fillåsning och revisionshantering

I det föregående kapitlet nämndes dubbletter, låsning av filer och automatisk revisionshantering. I detta kapitel tas dessa tre upp i detalj. I början beskrivs hur dubbletter tas omhand i insättningsfilerna, sedan presenteras det allmänna problemet med låsning av filer. För att göra användaren uppmärksam på detta problem visas ett exempel som illustrerar varför låsning av filer är nödvändig. För att underlätta för användaren i skapandet av en ny revision ges en beskrivning av automatiskt revisionshantering.

7.1 Dubbelinmatning

Problemet med dubbletter löstes på så vis att de ej sparas i BibTEX-databasen. Detta gjordes för att det inte finns någon mening med att ha två identiska poster i BibTEX-databasen. Detta löstes genom att vi jämförde olika fält beroende på referenstyp för att undersöka om två poster av samma referenstyp är identiska. Om den inmatade posten är identisk med någon av de poster som finns i BibTEX-databasen, sparas alltså inte den inmatade posten i BibTEX-databasen. Efter noggrannt övervägande, gjordes ett antagande

för varje referenstyp.
Dessa antaganden är följande:

1. Två böcker är samma om följande fält hos de båda är identiska: *title, author, edition, publisher* och eventuellt *volume* (hos multi-volym böcker).

2. Två artiklar är samma om följande fält hos de båda är identiska: *title, author, journal* och eventuellt *volume* (hos multi-volym artiklar).

3. Två *proceedings* är samma om följande fält hos de båda är identiska: *title, author, year* och *location.*

4. Två *inProceedings* är samma om följande fält hos de båda är identiska: *title, author, year* och *location.*

5. Två tekniska rapporter är samma om följande fält hos de båda är identiska: *title, author, year* och *institution.*

6. Två magistersavhandlingar är samma om följande fält hos de båda är identiska: *title* och *author.*

7. Två doktorsavhandlingar är samma om följande fält hos de båda är identiska: *title* och *author.*

8. Två opublicerade verk är samma om följande fält hos de båda är identiska: *title* och *author.*

9. Två oklassificerade verk är samma om följande fält hos de båda är identiska: *title,author* och *year.*

10. Två outgivna böcker är samma om följande fält hos de båda är identiska: *title* och *author.*

11. Två bokavsnitt är samma om följande fält hos de båda är identiska: *title, author, edition, chapter, pages* och eventuellt *volume* (hos multi-volym böcker).

12. Två manualer är samma om följande fält hos de båda är identiska: *title, author* och *year*.

7.2 Låsning av filer

Ett klassiskt problem uppkommer när flera användare läser och skriver till en fil samtidigt. Exempel på ett scenario kan vara följande:
Säg att användare X läser in en fil och utför en ändring. Direkt efter X har läst in filen, läser användare Y in samma fil och utför en ändring. Användare X skriver därefter sin ändring till filen. I det läget har användare Y kopia av filen inte den ändring som X gjorde, vilket innebär att enbart ändringen utförd av Y kommer att skrivas till filen.
En lösning på detta problem är att låsa filen, så att endast en användare i taget får tillgång till den. I PHP kan funktionen *flock()* användas för att åstadkomma denna låsning. Notera att hela systemet som webbapplikationen körs på måste använda samma princip för låsning av filer. Alla program som har tillgång till den aktuella filen, måste alltså hantera låsta filer på samma sätt, annars kommer det inte att fungera.

```
bool flock(int filhandtag, int operation);
```

Funktionen *flock()* tar emot två argument: ett filhandtag och den operation som ska utföras på filen.
Följande operationer finns att välja mellan (se tabell 7.1):

För att inte *flock()* ska blockera filen under tiden den utför låsning, kan tillägget LOCK_NB läggas till operationen med hjälp av ett plustecken (+).

Operation	Utför	Beskrivning
LOCK_SH	Delad låsning	Låser så att flera användare kan läsa filen samtidigt
LOCK_EX	Exklusiv låsning	Låser så att endast en användare kan skriva till filen
LOCK_UN	Låser upp	Låser upp filen som har varit låst

Tabell 7.1: *flock* operationer

Följande kod för olika operationer bör därmed användas (se tabell 7.2):

Ändamål	Operationskod
Läsning	LOCK_SH + LOCK_NB
Skrivning	LOCK_EX + LOCK_NB
Upplåsning	LOCK_UN + LOCK_NB

Tabell 7.2: *flock* operationskoder

Funktionen *flock()* returnerar true vid lyckad låsning och false om den misslyckas. Det är rekommenderat att avsluta webbapplikationen ifall låsningen inte fungerade, för att undvika att *flock()* hänger sig.
Denna avslutning kan utföras med hjälp av funktionen *die()*, som kan ses i följande exempel:

```
$fil = fopen("namn.txt", r);
flock($fil, LOCK_SH + LOCK_NB) or
die("Kunde inte låsa filen");
fclose($fil);
```

Detta exempel öppnar en fil för läsning och sätter därefter en delad låsning på den. Ifall låsningen inte går att utföra kommer funktionen *die()* att skriva ut ett felmeddelande och därefter avsluta webbapplikationen. Notera att *flock()* inte behöver låsa upp filen, eftersom funktionen *fclose()* gör det.
Mer om *flock* kan läsas i [3, 10, 13, 17].

7.3 RCS

Revision Control System (RCS), automatisk revisionshantering, tar hand om multipla versioner av textfiler. Walter F. Tichy och Paul Eggert skrev RCS.
RCS är en fri programvara och tillhandahålls av GNU Free Software Foundation. Den finns på de flesta system eller så kan *freeware* versionen installeras[12]. RCS automatiserar lagring, loggning, identifikation och sammanslagning av revisioner. RCS är lämplig för text som revideras ofta, till exempel källkod, dokumentation etc.
Två grundläggande kommandon som används är *ci* och *co*.
Ci, förkortning för *check in*, lagrar textfilens innehåll i en arkiveringsfil som kallas för RCS-fil och därefter raderas textfilen. En RCS-fil innehåller alla revisioner av denna textfil. RCS-filer slutar med *,v* som står för versioner.
Co, förkortning för *check out*, extraherar sista revisionen från RCS-filen och skriver denna revision till textfilen.
Med *ci -l*, som används i vår applikation, utförs de två ovanstående operationerna samtidigt (*ci* och *co*), men textfilen raderas inte. Dessutom utför detta kommando låsning av RCS-filen med växeln *-l*. Om växeln *-u* används istället för *-l*, utförs ingen låsning.
Följande operationer kan utföras med hjälp av RCS:

- Spara alla nya revisioner i en RCS-fil och extraherar gamla revisioner från RCS-filen. RCS sparar alla revisioner på ett sådant sätt att utrymme används effektivt. Förändringar förstör inte originalet, därför att föregående revisioner är tillgängliga. Revisioner kan hämtas med hjälp av revisionsnumret, *author*, datum etc.

- Ta hand om förändringarnas historia.
  RCS loggar alla förändringar automatiskt. Förutom texten av varje revision, kan RCS lagra datum, tid, *author* och även ett logg-meddelande som summerar förändringen. RCS logg-meddelanden kan skrivas ut med *rlog*. Dessutom kan *rcsclean* radera filer

[12] `<prep.ai.mit.edu>`

som inte är ändrade.

Två revisioner kan jämföras med *rcsdiff*.

- Ta hand om revisionsträd.

  RCS lagrar revisioner i ett revisionsträd som representerar rot-barn förhållande mellan revisioner.

- Slå samman flera revisioner (*rcsmerge*)

- Kan ge symboliska namn till revisioner

- Automatiskt identifiering av varje revision med hjälp av datum, tid, *author* etcetra.

Största fördelen med RCS är att den sparar minnesutrymme därför att RCS behöver minnesutrymme bara för att spara skillnader mellan revisioner.

7.3.1 Revisionsnummer och grenar

Olika nummer kan tilldelas till nästkommande revisioner.
För att starta en ny utgåva, *release*, till exempel revision 2, används följande två kommandon:

- ci -r2 textfilens_namn

- ci -r2.1 textfilens_namn

Det sista kommandot tilldelar numret 2.1 till den nya revisionen, de nästkommande revisionerna kommer då ha numren 2.2,2.3 etc. Med *co -r2.1 texfilens_namn* extraheras revisionen 2.1 från RCS-filen.
I RCS har användaren möjlighet att starta en gren från en existerande revision. Detta görs för att undvika att skapa en ny revision om ändringarna inte är så stora. För att starta en ny gren i revision 1.3 skrivs följande:

```
ci -r1.3.1 texfilens\_namn
```

Detta kommando startar en gren i revision 1.3 och numret 1.3.1.1 tilldelas till den nya revisionen.

7.3.2 Exempel av RCS-filen

I exemplet under visas RCS-filen för vår BibTEX-databas. Med nyckelordet *State* kan användaren beskriva till vilken utvecklingsfas en viss revision tillhör, till exempel *Exp=experimental*, *Stab=stable*, *Rel=release*. *Exp* är det fördefinierade värdet till de föregående. *Author* är *user name* av användaren som lade till, *check in*, denna revision. Med RCS kan revisionen plockas från filen. Med kommentarerna som är lagda förklaras på vilket sätt RCS sparar revisionerna. Kommentarerna lades bakom /* till slutet av raden.

```
head    1.3; access;/*säger vem som får låsa filen
symbols;   /*symboliska namn för olika revisioner (här finns inget
           /*symboliskt namn)
 locks
    apache:1.3; strict ;/*automatisk låses med hjälp av ci -l
                        /*1.3 aktuell (sista) revision
                        /*strikt låsning
comment @# @;/*kommentarer
 1.3 date 2003.03.09.12.14.29;/*datum när revisionen sparades i
                              /*filen
author apache;      /*author - apache server (ingen author anges)
state Exp;          /*Exp = experimental branches;
 /*inga grenar
next 1.2;           /*föregående revisionsnummer
  1.2 date 2003.03.09.12.10.53;
  author apache;
  state Exp;
  branches; next
1.1; 1.1 date 2003.03.09.12.08.19; author apache; state Exp;
branches; next ;
 desc @@
1.3 log @*** empty log message ***
 @
```

```
 text
 @@@Book{a,
 author =    {a},
 title =     {a},
 year =      {2003},
 edition =   {a},
 publisher = {a},
 bibdate =   {March 9,  2003,  1:14 pm}
 }
 @
1.2 log @*** empty log message ***
 @
 text
 @d7 1/*d7=delete7, d7 = tar bort sjunde raden från revision 1.3.1 och 1 anger hur många rader
      /*framåt det skall tas bort (i detta fall 1)
 a7 11/*a7=append7, a7 = lägger till, men i det här fallet kan man tänka sig
      /*mer som insert, med början av sjunde raden och 11 rader som
      /*som läggs till (visas här under).
    bibdate =    {March 9,  2003,  1:08 pm},
    annote =     {asdasdas}
    }
    @@Article{b,
    author =      {a},
    title =       {a},
    journal =     {asdas},
    year =        {2003},
    number =      {asd},
    adress =      {asd},
    bibdate =     {March 9,  2003, 1:10 pm}
    @
1.1 log @Initial revision
    @
    text
    @d10 9/*med början av raden 10 (revision 1.2) ta bort 9 rader
    @
```

Mer information om RCS återfinns i [2, 15].

8 Applikationens utökningar

I detta kapitel implementeras en php-sida på ett objektbaserat sätt. Den objektbaserade lösningen bygger på en redan existerande sida som redan är beskriven, nämligen InsertArticle.php. Detta för att illustrera hur PHP och OOP fungerar tillsammans. Sedan ges en beskrivning av den Java Applet som användes till att implementera textremsan samt JavaScript som användes bland annat vid skapandet av *pop-up windows*.

8.1 Objekt-orienterad programmering med PHP

En klass består av metoder och attribut. Attributen börjar med det reserverade ordet *var* som står för en variabel. I klassens konstruktor initialiseras attributen, till exempel

```
function DubblePostChecker($fd)
{
                $this->dubble=false;
                $this->key_dubble=false;
                $this->filename_database=$fd;
                $this->author=$GLOBALS['author'];
                $this->title =$GLOBALS['title'];
                $this->key=$GLOBALS['key'];
                $this->journal =$GLOBALS['journal'];
                $this->vol =$GLOBALS['vol'];
                $this->number =$GLOBALS['number'];
}
```

$GLOBALS är globala variabler, i vårt fall de som finns i formuläret. För att skapa ett objekt av en klass används nyckelordet *new* som visas här under:

```
    $form_intro=new FormIntroduktion();
```

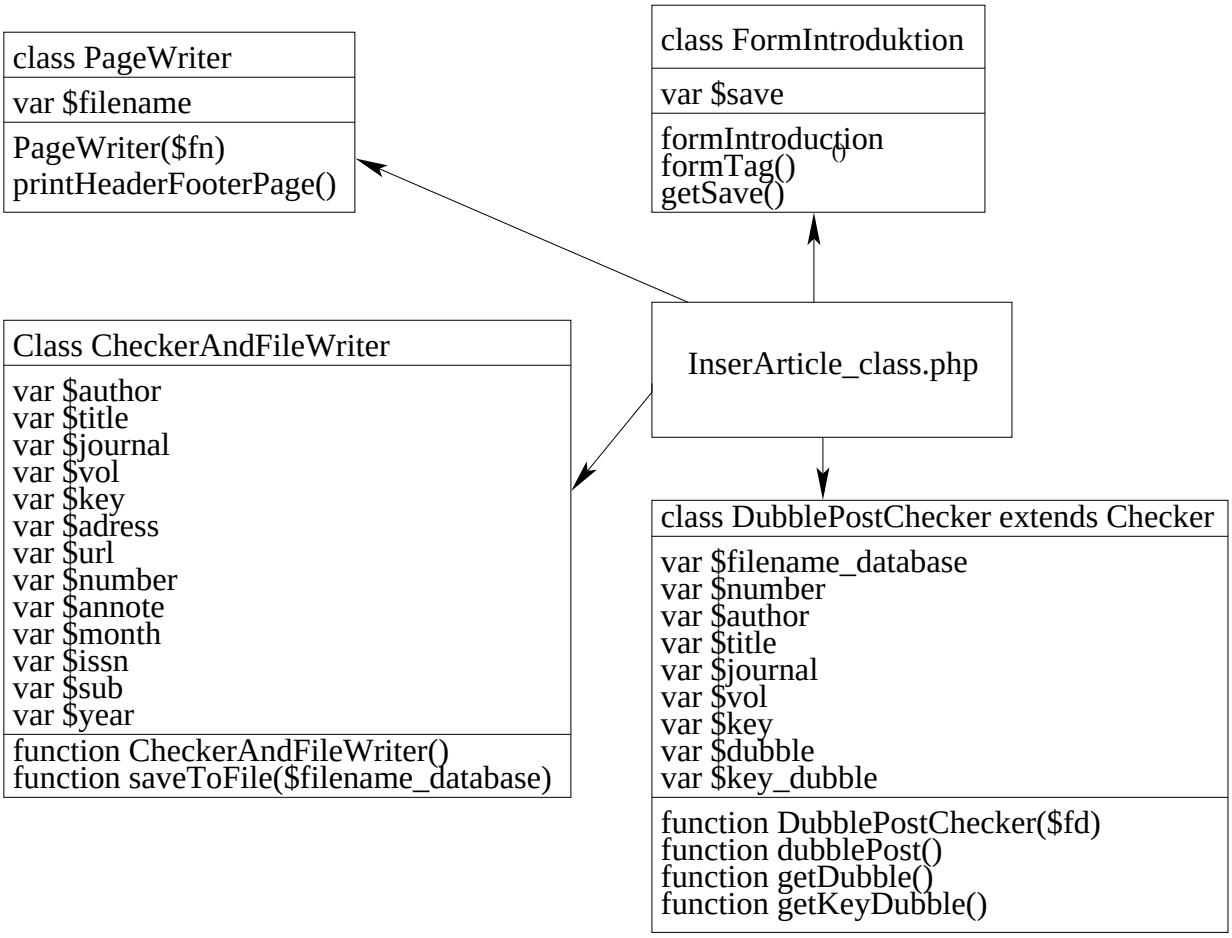
För att anropa en metod av en klass användspiloperatorn som visas i följande mönster:

```
$res->printResult($this->key_dubble,$this->dubble,$maybe_all_same,
                  $this->key,$this->journal,$this->title,$this->author,
                  $this->number);
```

Arv åstadkoms med hjälp av nyckelordet *extends*, till exempel

```
class DubblePostChecker extends Checker
```

I själva sidan skapas objekt av fyra klasser (se figur 8.1). Klass *DubblePostChecker* ärver klassen *Checker* och innehåller klassen *Result* (se figur 8.2). Klass *CheckerAndFileWriter* innehåller klasserna *BracketCiteChecker*, *FileSaver* och *MessageWriter*.

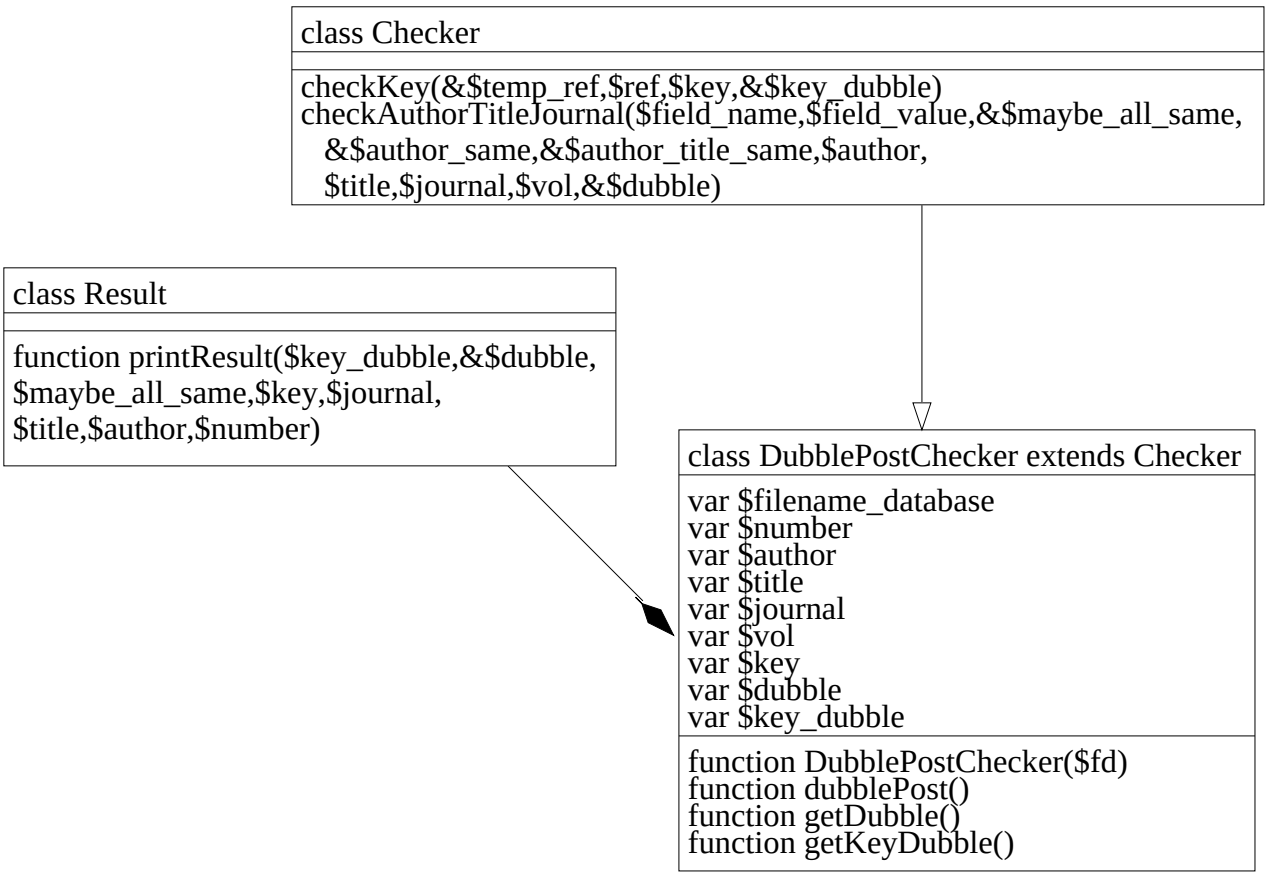


Figur 8.1: OOP - Klasser som finns på InsertArticle_class.php

PHP stöder bara enkelt arv och inte multipelt arv.

InsertArticle.php valdes slumpmässigt för att implementeras på ett objekt-orienterat sätt. Detta gjordes för att illustrera på vilket sätt OOP görs med PHP. OOP lösningen byggs

på den existerande lösningen som redan är beskriven. Php-sidan implementerades genom nio olika klasser. Klass diagrammen visas i figur 8.2 och figur 8.3.



Figur 8.2: OOP - Första delen av klassdiagrammet

Följande klasser implementerades:

1. **PageWriter** är ansvarig för att skriva ut innehållet från olika filer (html-kod: header, footer och layout av själva sidan) på skärmen.
   Metod - *printHeaderFooterPage*
   Metod - *getSave*

2. **FormIntroduktion** skriver ut *form tag* och tar hand om variabeln *save* (save-knapp).

Metod - *formTag*

3. **Checker** undersöker nyckeln och följande fält: *author,title,journal*
   Metod - *checkKey*: undersöker nyckeln, det vill säga om den inmatade nyckeln redan finns i BibTEX-databasen (*key_ dubbel=true*)
   Metod - *checkAuthorTitleJournal*: fälten *author, title* och *journal* undersöks (samma post kan redan finnas i BibTEX-databasen - *dubbel=true*).

4. **DubblePostChecker** undersöker om samma post (eller nyckel) som den inmatade finns i BibTEX-databasen och, om det behövs, skriver ut viss(a) meddelande(n). Den här klassen ärver egenskaper av klassen *checker*.
   Metod - *dubblePost*: undersöker om samma post eller nyckel existerar i BibTEX-databasen. De olika felmeddelanden skrivs ut i mån av behov.
   Metoder - *getDubble* och *getKeyDubble*: returnerar *dubble* respektive *key_ dubble*

5. **Result** skriver ut olika felmeddelanden. Ett objekt av denna klass skapas i metoden *dubblePost* av klassen *DubblePostChecker*.
   Metod - *printResult*

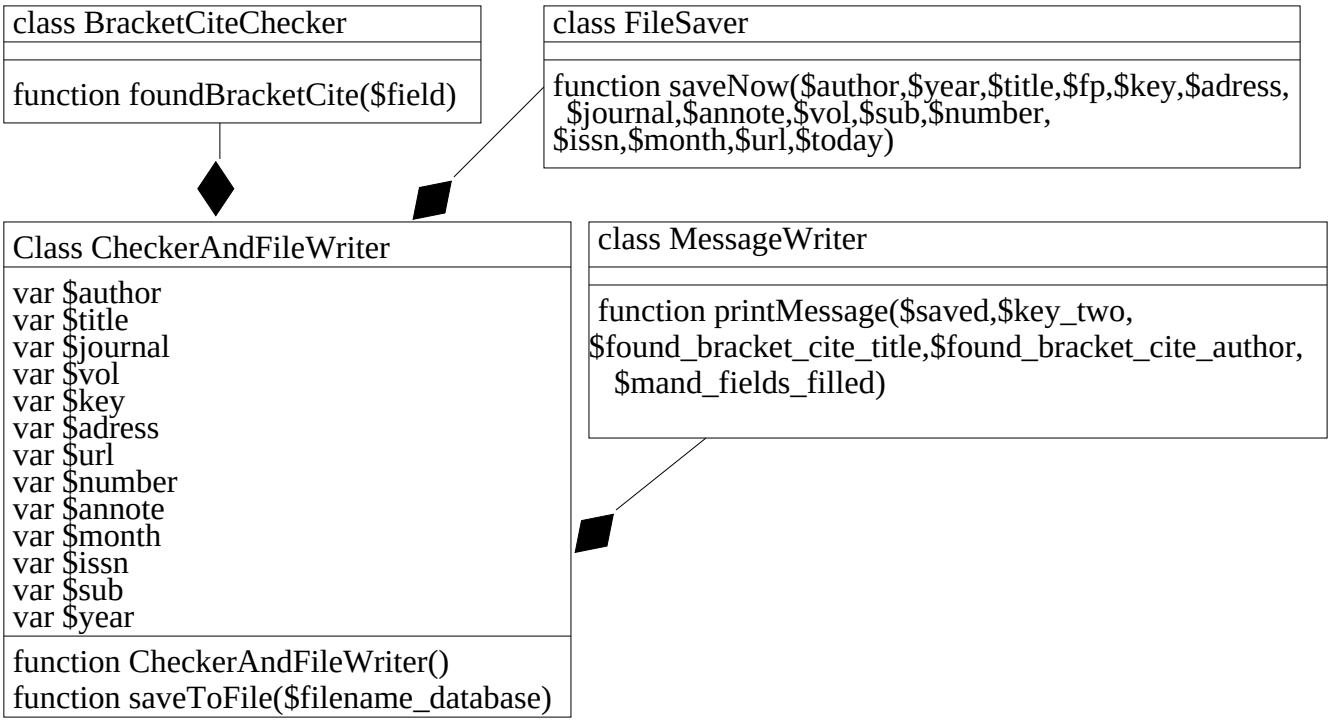
6. **CheckerAndFileWriter** undersöker först villkor, obligatoriska fält, { eller " i *author* eller *title* fälten, nyckel med två ord, och om alla villkor tillfredsställs sparas filen.
   Den här klassen använder metoder från tre klasser - *BracketCiteChecker*, *FileSaver* och *MessagePrinter*.
   Metod - *saveToFile*

7. **MessageWriter** skriver ut olika meddelanden. Ett objekt av denna klass skapas i metoden *saveToFile* av klassen *CheckerAndFileWriter*.
   Metod - *printMessage*

8. **FileSaver** sparar posten i BibTEX-databasen. Ett objekt av denna klass skapas i metoden *saveToFile* av klassen *CheckerAndFileWriter*.

Metod - *saveNow*

9. **BracketCiteChecker** undersöker om det finns } eller " i *title* eller *author* fälten. Ett objekt av denna klass skapas i metoden *saveToFile* av klassen *CheckerAndFileWriter*.
   Metod - *foundBracketCite*



Figur 8.3: OOP - Andra delen av klassdiagrammet

8.2 Applets beskrivning

Först initialiseras appleten sedan plockas parametrarna *text* och *font* från applet taggen och dubbelbuffring utförs. *Start* och *stop* metoderna hanterar tråden. *Run* metoden består av en oändlig loop i vilken metod *moveText* anropas med de olika parametrarnas värden.

```
public void moveText(int speed,int size,int mode,Color
Color1,Color Color2)
```

Förklaring av *moveText* metodens parametrar:

- Med parametern *speed* kan hastigheten på texten ändras
- Med parametern *mode* kan riktningen på texten ändras

Texten kan ställas in i tre olika riktningar:

- Om *mode*=MOVE_HOR, då rör sig texten i vågrätt riktning
- Om *mode*=MOVE_VER, då rör sig texten i lodrätt riktning
- Om *mode*=MOVE_CURVE, då rör sig texten enligt kurvan (Y=2X).

Textens storlek varierar mellan FONT_MIN och FONT_MAX och remsan målas i två valda färger.

8.3 JavaScript

JavaScript används till *pop-up windows* i InsertXX.php filerna.
För att få fram *pop-up windows* används funktionen *winpop*:

```
function winpop(file)
{
   window.open(file,"","height=20,width=300,status=no,toolbar=no,
               directories=no,menubar=no,location=no,
               resizable=yes,scrollbars=no,left=450,top=30");
}
```

Winpop består av, som användaren kan se från koden, en enda sats. Det öppnar ett nytt fönster med en given storlek (*height* och *width*) som kan justeras med knapparna för *resizing*. Dessa knappar är placerade på en given position på förälder fönstret (InsertXX.php). Med parametern *file* anges vilken av Info_XX.htm filer som ska öppnas.
Funktionen anropas på följande sätt:

```
<a href="javascript:winpop('html/Info_author.htm')">
```

Följande Info_XX.htm filer gjordes som är placerade i html-katalogen:

- Info_year.htm
- Info_key.htm
- Info_title.htm
- Info_language.htm
- Info_publisher.htm
- Info_adress.htm
- Info_vol.htm
- Info_url.htm
- Info_edition.htm
- Info_isbn.htm

JavaScript används också för att få information utskriven på status fältet på startsidan.

```
<input type=image name="imm" src="Graphic/britain.gif"
onMouseOver="window.status='Menu - english';return true;">
```

Dessutom visas rullande text på status fältet på menysidan med hjälp av JavaScript. Funktionen *rullaText* används för detta ändamål.

```
function rullaText(blank)
  {
    var text = 'This is a menu page.'
    utText = ' ';
    if(blank<text.length)
    {
```

```
    //visa bara en del av strängen i början av status fält
    utText+=text.substring(0,blank);
    self.status = utText;//visa på statusrad
    blank++;
    var anrop = 'rullaText(' + blank + ')';
    //visa texten var 100 ms(rullaText utförs var 100 ms)
    textVisare = setTimeout(anrop,100);
}
else if(blank <150)
{
    // Lägg till blanktecken innan texten.
    for (i = 0 ; i < (blank-text.length); i++)
    {
                utText += ' ';
    }
    utText+= text;//konkatera text och blank
    self.status = utText;//visa på statusrad
    blank++;
    var anrop = 'rullaText(' + blank + ')';
    textVisare = setTimeout(anrop,100);
}
else
{
    // Ta bort ett tecken ur texten.
    if (blank <(150+text.length))
    {
        // Texten inte borttagen (helt).
        for (i = 0 ; i < (blank-text.length); i++)
        {
                utText += ' ';
        }
        //visa bara en del av strängen
        utText+= text.substring((blank-150),text.length);
        self.status = utText;
        blank++;
        var anrop='rullaText(' + blank + ')';
        textVisare = setTimeout(anrop,100);
    }
    else
    { // Texten helt borttagen, börja om.
        self.status = ' ';
        textVisare = setTimeout('rullaText(0)',100);
    }
```

```
    }
}
```

9 Slutsats

Detta kapitel omfattar uppräkningen av ett antal problem som har uppstått under arbetet, en uppräkning av de kunskaper som har erhållits genom detta arbete och även vår bedömning om vi har uppnått målet.

9.1 Problem

Under arbetsgången uppstod följande större problem:

1. Det tog ganska lång tid att hitta någon fungerande lösning för borttagningssidan (Delete.php).

2. Det tog längre tid än förväntat att implementera *file element* i InsertArticle.php sidan.

3. Det var ganska svårt i början att rita bilder (diagram) i programmet *xfig* på grund av att vi inte visste att vi var tvungna att ha båda musknapparna nertryckta samtidigt när vi skulle utföra en viss operation.

4. Det visade sig att skriva rapporten i LaTeX blev inte så lyckat. Det blev nästan samma sak som att lära sig ett helt nytt programmeringsspråk. Detta på grund av LaTeX har många kommandon som man måste använda sig av för att formatera texten, lägga till bilder etc.

5. Att lägga till skärmdumpar i rapporten blev ganska krångligt. Eftersom skärmdumparna först måste sparas i .jpg format och därefter omvandlas till .eps format.

Det uppstod även ganska många små problem. Ett av problemen var när tab användes för att åstadkomma ett önskat format av posten i BibTeX-databasen. Ett annat problem

uppstod vid försök att åstadkomma en önskad layout av formuläret med HTML taggen $</table>$ etc.

9.2 Erfarenheter

Många inbyggda PHP funktioner användes för att konstruera funktionerna i applikationen. De flesta av dessa var olika stränghanteringsfunktioner och olika filhanteringsfunktioner som exempelvis funktionerna strtok() respektive fopen(). Förutom dessa inbyggda funktioner implementerades också många egna funktioner såsom exempelvis printHeaderFooterPage() som skriver ut filens innehåll.

Att skriva rapporten i LaTeX visade sig vara en nyttig erfarenhet eftersom vi fick lära oss ett nytt sätt att skriva rapporter på. Men det visade sig att det inte var så effektivt att skriva rapporten i LaTeX. Detta på grund av att till exempel bilderna placerades inte på önskade ställen i rapporten eftersom LaTeX själv hanterar detta genom att placera bilden där den passar in storleksmässigt. Skrivandet av rapporten underlättades när textredigeringsprogrammet *Winedit* `<http:www.winedit.com>` tillsammans med *MikTex*, som gör det möjligt för användaren att förhandsgranska texten, upptäcktes. WinEdit har ett grafiskt gränssnitt som tillhandahåller stavningskontroll, färdiga kommandon med mera. Användaren behöver således inte känna till alla kommandon i LaTeXför att formatera texten. De flesta av kommandona skrivs ut automatisk genom att användaren klickar på knapparna som finns i *Winedit*.

Ett program som underlättar skrivandet av php-koden, *Php- coder*[13] upptäcktes också. *Php coder* är en relativt bra editor och gratis. *Php coder* gör det möjligt för användaren att skriva php koden samt förhandsgranska php sidorna.

Förutom de ovanstående delarna har vi erhållit ytterligare kunskap inom HTML samt hur formateringen av bilder från .jpg till .eps utförs.

Som slutord kan vi säga att vi lärde oss många nya saker, programmeringsmässigt som

[13] `<http:www.phpide.de>`

formateringsmässigt, vid framställningen av denna rapport. Men det skall också tilläggas att vi även utnyttjade den kunskap som vi besitter för att nå lösningar på de problem som uppstått i samband med rapportskrivningen.

9.3 Bedömning av projektet

Vi är nöjda med vår insats och det uppsatta målet har uppnåtts vilket var att skapa ett webbgränssnitt för BibTEX-databasen.

Referenser

[1] Ganesh Prasad Chris Ullman. *Beginning PHP.* Wrox Press, 2000.

[2] Mike Loukides Dan Bolinger. *Applying RCS and SCSS:From source control to project control.* Thomson Learning, 1993.

[3] Adam Trachtenberg David Sklar. *PHP Cookbook.* O'Reilly and Associates, 2002.

[4] Antoni Diller. *LaTeX: line by line:Tips and Techniques for document processing.* Wiley,John and Sons, 1999.

[5] Susan L. Fawler. *GUI Style Guide.* Science and Technology Books, 1994.

[6] Wilbert O. Galitz. *An Introduction to GUI design principles and techniques.* Wiley John and Sons, 2002.

[7] George Gretzer. *Math into LaTeX.* Birkhauser Boston, 1998.

[8] Jane Hahn. *LaTeX for everyone.* Prentice Hall incorporated, 1993.

[9] Patrick W Daly Helmut Kopka. *Guide yo LaTeX :Document preparation for beginners and advanced users.* Pearson Education, 1999.

[10] Viktor Jonsson. *Webbprogrammering med PHP.* Studentlitteratur, 2001.

[11] Donald E. Knuth. *Computer and Typesetting.* Addison-Wesley, 2000.

[12] Alan Moore. *Graphical User Interface Design and Evaluation.* Prentice Hall, 1995.

[13] Kevin Tatroe Rasmus Jay Lerdorf. *Programming PHP.* O'Reilly and Associates, 2002.

[14] J. Kenneth Shultis. *LaTeX Notes.* Prentice Hall, 1994.

[15] Walter F. Tichy. Design,implementation and evaluation of a revision control system. September 1982.

[16] Larry Ullman. *Visuell snabbguide PHP.* Studentlitteratur, 2001.

[17] Janet Valade. *PHP and MySQL for dummies.* Wiley John and Sons, 2002.

A PHP - källkod

A.1 InsertArticle_s.php

```
<?php
 $filename_header="html/header1.htm";
 $filename_js="html/head_java.htm";
 printHeaderFooterPage($filename_js);
 printHeaderFooterPage($filename_header);


 /****************************************/
 /*pre - true                            */
 /*post - filens innehåll (html kod) visat  */
 /*       på skärmen                     */
 /****************************************/

 function printHeaderFooterPage($filename)
 {
         if (!$fp = fopen($filename, 'r'))
         {
              print "Cannot open file ($filename)";
              exit;
         }
         while(!feof($fp))//till slutet av filen
         {
            $row=fgets($fp,1024);//en rad i taget
            print $row;
        }
        fclose($fp);
 }
?>
<?php
  $filename_database = "db/database.bib";     //databasen(filen)
  $filename_page="html/Article_s.htm";       //referens-specifika fält
  $filename_common="html/commonField1_s.htm"; //gemensamma fält
  $filename_annote="html/annote_s.htm";      //"annote" fält
  $dubble=false;    //om boken finns redan i databasen,dubble=true
  $key_dubble=false;//om nyckeln redan finns i databasen,key_dubble=true
  if($save)
  {
          dubblePost($author,$title,$key,$number,$journal,$vol,
                     $filename_database,$dubble,$key_dubble);
          $found_bracket_cite_title=foundBracketCite($title);
          $found_bracket_cite_author=foundBracketCite($author);
          $found_comma=foundComma($key);
          printCommaBracket($found_bracket_cite_title,
                            $found_bracket_cite_author,$found_comma,
                            $number,$year,$title,$author,$journal,$key);
          if((!$dubble)&&(!$key_dubble)&&
             (!$found_bracket_cite_title)&&(!$found_bracket_cite_author)
             &&(!$found_comma))

          {
             saveToFile($author,$year,$title,$filename_database,
                        $key,$adress,$journal,$annote,
                        $vol,$sub,$number,$issn,$language,$month,$url);
          }

    }
```

```
print"<FORM  name=\"form1\"  action=\".$_SERVER["PHP_SELF"]."\"
       method=post enctype=\"multipart/form-data\">";
printHeaderFooterPage($filename_common);
printHeaderFooterPage($filename_page);
printHeaderFooterPage($filename_annote);

/*************************************************************************/
/*pre- true                                                              */
/*post-funktionen kollar om det finns en post i databasen som har samma  */
/*     fältens(author,title,year,journal)värden som inmatade värdena i   */
/*     motsvarande textrutorna.                                          */
/*************************************************************************/

function dubblePost($author,$title,$key,$number,$journal,$vol,
                    $filename_database,&$dubble,&$key_dubble)
{

     if (!$fp = fopen($filename_database, 'r'))//öppna filen för läsning
     {
          print "Cannot open file ($filename_database)";
          exit;
     }
     $maybe_all_same=false;
     while(!feof($fp))//till slutet av filen
     {
        $row=fgets($fp,1024);//en rad i taget
        $ref=strtok($row,"{");//extrahera referenstyp
        //kolla om det handlar om ny post
        if((strcmp($ref,"@Book")==0 )||(strcmp($ref,"@Article")==0)||
           (strcmp($ref,"@InProceedings")==0)||
           (strcmp($ref,"@MastersThesis")==0)||
           (strcmp($ref,"@Misc")==0)||(strcmp($ref,"@Manual")==0)||
           (strcmp($ref,"@TechReport")==0)||(strcmp($ref,"@PhDThesis")==0)||
           (strcmp($ref,"@Unpublished")==0)||(strcmp($ref,"@Proceedings")==0)||
           (strcmp($ref,"@InBook")==0)||(strcmp($ref,"@Booklet")==0))
        {
               if($maybe_all_same)
               {
                      $dubble=true;
               }
               $author_title_same=false;
               $author_same=false;
               $maybe_all_same=false;
               checkKey($temp_ref,$key_dubble,$key,$ref);
        }
        if(strcmp($temp_ref,"@Article")==0)//om den handlar om artikeln
        {
               $field_name=strtok($row," ");
               $token_second=strtok("{\"");//ta andra token
               $field_value=strtok("}\"");//extrahera fältets värde
               $field_value=trim($field_value);//ta bort blanktecken
               //värden från olika fälten jämförs med de
               //inmatade värdena(textrutornas innehåll)
               checkAuthorTitleJournal($field_name,$field_value,
                                       $author,$title,
                                       $maybe_all_same,$author_same,
                                       $author_title_same,$dubble,
                                       $journal,$vol);
        }
    }
    fclose($fp);
    //skriv ut varningar(om de finns) om dubbel post eller dubbel nyckel
```

```
   printResult($key_dubble,$dubble,$maybe_all_same,$key,
   $journal,$title,$author,$number);
 }
/*************************************************************************/
/*pre-true(anropas av dubblePost)                                        */
/*post-funktionen kollar om den inmatade nyckeln är lika med en av       */
/*     de nycklarna som finns i databasen                                */
/*************************************************************************/
 function checkKey(&$temp_ref,&$key_dubble,$key,$ref)
 {
              $key_database=strtok(",");//extrahera nyckel
              $key_database=trim($key_database);//ta bort blanktecken
              if(strcmp($key_database,$key)==0)//nyckeln kollas
                      $key_dubble=true;
              $temp_ref=$ref;/*temporär variabel för att lagra
                              referenstypen tills ny post kommer*/
 }

 /*************************************************************************/
 /*pre-true(anropas av dubblePost)                                        */
 /*post-funktionen kollar om de inmatade värdena(textrutornas innehåll) är */
 /*     lika med motsvarande fältens värden av en post i databasen.       */
 /*     Följande fälten kollas:author,title,year och journal              */
 /*************************************************************************/

 function checkAuthorTitleJournal($field_name,$field_value,$author,$title,
                                  &$maybe_all_same,&$author_same,
                                  &$author_title_same,&$dubble,$journal,$vol)
 {
              if(strcmp($field_name,"author")==0)
              {       //jämför inmatat 'author' med den som finns i posten
                      if(strcmp($field_value,$author)==0)
                      {
                           $author_same=true;
                      }
              }
              else if((strcmp($field_name,"title")==0))
              {     //jämför 'titel'(i posten och inmatat)
                    if(strcmp($field_value,$title)==0 && $author_same)
                    {
                      $author_title_same=true;
                    }
              }
              else if((strcmp($field_name,"journal")==0))
              {     //jämför 'journal'(i posten och inmatat)
                    if(strcmp($field_value,$journal)==0 && $author_title_same)
                    {
                      $maybe_all_same=true;// author,titel och tidning träffat
                    }
               }
               else if((strcmp($field_name,"volume")==0))
               {   //jämför 'volume'(om den finns)
                   if(strcmp($field_value,$vol)==0 && $maybe_all_same)
                   {
                     $dubble=true;
                   }
                   else
                   {
                     $maybe_all_same=false;
                   }
              }
}
```

```
 /*************************************************************************/
 /*pre-true(anropas av dubblePost)                                        */
 /*post-olika varningar utskrivna om följande hittades:                   */
 /*     1)dubbel nyckel                                                   */
 /*     2)dubbel post                                                     */
 /*     3)dubbel nyckel eller dubbel post och ett av obligatoriska fält   */
 /*       är inte ifyllt                                                  */
 /*************************************************************************/

 function printResult($key_dubble,&$dubble,$maybe_all_same,$key,$journal,
                      $title,$author,$number)
 {
     if($dubble || $maybe_all_same)
     {
               print "<font color=red size=5><B><Center>Du har försökt
               att spara artikeln som redan
               finns i databasen</center></B></font><br>";
               $dubble=true;
     }
     if($key_dubble)
     {
        print "<font color=red size=5><B><Center>Dubbel nyckel!!
               Byt den!!</center></B></font><br>";
     }
     if(((strcmp($journal,"")==0)||(strcmp($title,"")==0)||
         (strcmp($author,"")==0)||(strcmp($number,"")==0))&&
         ($key_dubble||$dubble))
     {
               print "<font color=red size=5><B><Center>Alla
               obligatoriska fält är inte
               ifyllda!!Försök igen!</center><B></font><br>";
     }

 }
 /*************************************************************************/
 /*pre-true                                                               */
 /*post-returnerat true om ,  finns i den inmatade nyckeln , annars false  */
 /*************************************************************************/
function foundComma($key)
{
    for($i=0;$i<strlen($key);$i++)
    {
        if((strcmp($key{$i},",")==0))
        {
                return true;
        }
    }
    return false;
}
 /*************************************************************************/
 /*pre-true                                                               */
 /*post-returnerat true om } eller " finns i 'title' eller 'author' fält   */
 /*     (inmatat),annars false                                            */
 /*************************************************************************/
function foundBracketCite($field)
{
    for($i=0;$i<strlen($field);$i++)
    {
        if((strcmp($field{$i},"}")==0)||(strcmp($field{$i},'"')==0))
        {
                return true;
        }
```

```
    }
    return false;
}
 /*************************************************************************/
 /*pre-Den inmatade posten innehåller inte:                               */
 /*    1) } i 'title' eller 'author' fält,                                */
 /*    2) , i nyckeln                                                     */
 /*    3)  nyckeln som redan finns i databasen                            */
 /*       och databasen innehåller redan inte en likadan post             */
 /*post-inmatade värdena sparade i filen(en ny post skapades) om:         */
 /*     1)alla obligatoriska fält är ifyllda                              */
 /*     2)inmatad nyckeln består av bara ett enda ord                     */
 /*************************************************************************/

function saveToFile($author,$year,$title,$filename_database,$key,$adress,
                    $journal,$annote,$vol,$sub,$number,$issn,
                    $language,$month,$url)
{
    $saved=false;   //om posten är sparad=true
    $key_two=false; //om nyckeln består av 2 ord=true
    //filen öppnas för skrivning(append)
    if (!$fp = fopen($filename_database, 'a'))
    {
          print "Cannot open file ($filename_database)";
          exit;
     }
     flock($fp,2);//filen låses
     $today=date("F j,  Y,  g:i a");//aktuellt datum och tid
     $mand_fields_filled=false;

     //obligatoriska fältt kollas
     if((strcmp($author,"")!=0)&&(strcmp($title,"")!=0)&&
        (strcmp($key,"")!=0)&&(strcmp($journal,"")!=0)&&
        (strcmp($year,"")!=0)&&(strcmp($number,"")!=0))
     {
            $mand_fields_filled=true;
            $key_1=strtok($key," ");
            $key_2=strtok("");
            if(!$key_2)
            {         //allt inmatad information skrivs i filen

                 saveNow($author,$year,$title,$fp,$key,$adress,
                         $journal,$annote,$vol,$sub,$number,
                         $issn,$month,$language,$url,$today);
                 $saved=true;
            }
            else
            {
                   $key_two=true;
            }
     }
     flock($fp,3);//filen låses upp
     //en eller flera meddelande(n) skrivs ut
     printMessage($saved,$key_two,$mand_fields_filled);
     fclose($fp);
     if($saved)
     {
          //automatisk revisionshantering
          system("ci -l db/database.bib");
     }

}
```

```
/*************************************************************************/
/*pre-true                                                               */
/*post-de inmatade värdena sparade i filen                               */
/*************************************************************************/

function saveNow($author,$year,$title,$fp,$key,$adress,
                 $journal,$annote,$vol,$sub,$number,$issn,$month,
                 $language,$url,$today)
{
        fwrite($fp, "@Article{".$key.",\n");
        fwrite($fp, str_pad("author =",14)."{".$author."},\n");
        fwrite($fp, str_pad("title =",14)."{".$title."},\n");
        fwrite($fp, str_pad("journal =",14)."{".$journal."},\n");
        fwrite($fp, str_pad("year =",14)."{".$year."},\n");
        fwrite($fp, str_pad("number =",14)."{".$number."}");
        if(strcmp($adress,"")!=0)
        {
           fwrite($fp, ",\n". str_pad("adress =",14)."{".$adress."}");
        }
        if(strcmp($vol,"")!=0)
        {
           fwrite($fp, ",\n". str_pad("volume =",14)."{".$vol."}");
        }
        if(strcmp($issn,"")!=0)
        {
           fwrite($fp, ",\n". str_pad("ISSN =",14)."{".$issn."}");
        }
        if(strcmp($language,"")!=0)
        {
            fwrite($fp, ",\n". str_pad("language =",14)."{".$language."}");
        }
        if(strcmp($month,"")!=0)
        {
            fwrite($fp, ",\n". str_pad("month =",14)."{".$month."}");
        }
        if(strcmp($sub,"")!=0)
        {
            fwrite($fp, ",\n". str_pad("submitter =",14)."{".$sub."}");
        }
        if(strcmp($_FILES['url']['name'],"")!=0)
        {
            fwrite($fp, ",\n". str_pad("URL =",14).
                   "{".$_FILES['url']['name']."}");
        }
        fwrite($fp, ",\n". str_pad("bibdate =",14)."{".$today."}");
        $annote=trim($annote);
        if(strcmp($annote,"")!=0)
        {
            fwrite($fp, ",\n". str_pad("annote =",14).
                   "{".$annote."}\n}\n");
        }
        else
        {
            fwrite($fp,"\n}\n");
        }
}
/*************************************************************************/
/*pre-true(anropas av saveToFile)                                        */
/*post-en eller flera meddelande utskrivna.Tre olika meddelande existerar:*/
/*     1)alla obligatoriska fält är inte ifyllda                         */
/*     2)inmatade nyckeln består av mer än ett ord                       */
```

```
     /*     3)ny post sparades i databasen                                  */
     /*********************************************************************/

    function printMessage($saved,$key_two,$mand_fields_filled)
    {
         if(!$mand_fields_filled)
         {
                   print "<font color=red size=5><B><Center>Alla obligatoriska
                   fält är inte ifyllda!!Försök igen!!</center><B></font><br>";
         }
         if($key_two)
         {
                   print "<font color=red size=5><b><center>Två ord i nyckel!!
                   Försök igen!!</B></center></font><br>";
         }
         if($saved)
         {
                   print"<font color=blue size=4><b><center>Den nya artikeln är
                    sparad i databasen</center></font></b><br>";
         }
   }
     /*********************************************************************/
     /*pre-true                                                           */
     /*post-en eller flera meddelande utskrivna.Tre olika meddelande existerar:*/
     /*     1) } eller " finns i author fält(inmatat)                     */
     /*     2) } eller " finns i title fält(inmatat)                      */
     /*     3) ,  finns i den inmatade nyckeln                            */
     /*     4) obligatoriska fält är inte fyllda                          */
     /*********************************************************************/
    function printCommaBracket($found_bracket_cite_title,
                               $found_bracket_cite_author,$found_comma,
                               $number,$year,$title,$author,$journal,$key)
    {
         if($found_bracket_cite_title)
         {
                   print"<font color=red size=5><b><center> } eller \" inmatat
                   i <I>title</I> fält</center></b></font><br>";
         }
         if($found_bracket_cite_author)
         {
                   print"<font color=red size=5><b><center>  } eller \" inmatat i
                    <I>author</I> fält</center></b></font><br>";
         }
         if($found_comma)
         {
                   print"<font color=red size=5><b><center> , inmatat i nyckeln
                   </center></b></font><br>";
         }
         if(((strcmp($number,"")==0)||(strcmp($title,"")==0)
            ||(strcmp($author,"")==0)||(strcmp($year,"")==0)
            ||(strcmp($journal,"")==0)||(strcmp($key,"")==0))&&
            ($found_bracket_cite_title||$found_bracket_cite_author||$found_comma))
        {
                 print "<font color=red size=5><B><Center>Alla obligatoriska fält
                 är inte ifyllda!!Försök igen!</center><B></font><br>";
        }
    }
?>
<?php
 $filename_footer="html/footer.htm";
 printHeaderFooterPage($filename_footer);
?>
```

A.2 Delete_s.php

```
<?php
 function saveHeaderFooter($filename)
 {
         if (!$fp = fopen($filename, 'r'))
         {
              print "Cannot open file ($filename)";
              exit;
         }
         while(!feof($fp))//till slutet av filen
         {
            $row=fgets($fp,1024);//en rad i taget
            print $row;
         }
         fclose($fp);
 }
?>

<?php
    $filename_header="html/header.htm";
    $filename_footer="html/footer.htm";
    $filename_database = "db/database.bib";
    $number_post=0;//antal post i post array
    $resultat="";//nyckel av denna post som kommer att tas bort
    $whole_result="";
    //om man trycker på sök knapp
    if($search)
    {
              $number_post_found=0;//antalet träffar=0
              saveToArrayFound($key,$filename_database,$found,
                              $number_post_found);
              if($number_post_found!=0) //hittat någonting
              {
                    $search=false;
                    saveHeaderFooter($filename_header);
                    writeToPage($found,$number_post_found,
                                $search,$whole_result);
                    saveHeaderFooter($filename_footer);

              }
              else
              {
                    toSearch($search,$delete);
              }

    }
    else if($delete)
    {
             //den valda posten tas bort från arrayen found och filen
             saveChangeToArrayFound($found,$number_post_found,
                                    $result,$resultat,$whole_result);
             saveToArrayPost($post,$number_post,$resultat,
                             $filename_database);
             writeToFile($post,$number_post,$filename_database);
             if($number_post_found!=0) //hittat någonting
             {
                   saveHeaderFooter($filename_header);
                   //skriv till  sidan återstående radioknappar
                   writeToPage($found,$number_post_found,
                               $search,$whole_result);
                   saveHeaderFooter($filename_footer);
```

```
        }
        else
        {
               toSearch($search,$delete);
        }

}

/*********************************************************************/
/*pre-om ingenting hittades eller finns ingenting mer att ta bort     */
/*post-'search_s.php' öppnat                                          */
/*********************************************************************/

function toSearch($search,$delete)
{

   print"<!doctype html public \"-//W3C//DTD HTML 4.01//EN\">
         <html>
               <head>
                   <title>redirect</title>
                   <meta http-equiv=\"refresh\" content=\"0;
                   URL=Search_s.php?search="
                   .$search."&delete=".$delete."\">
               </head>
         <body>
           <p>
             If you aren't forwarded to the search page
              <a href=\"Search_s.php?search=".$search."&delete="
              .$delete."\"> click here</a>.
           </p>
         </body>
         </html>";
}
/*********************************************************************/
/*pre-anropas när man har klickat på knappen 'delete'                */
/*post-vald post borttagen från arrayen found                        */
/*********************************************************************/

// I variabeln resultat  lagras nyckeln av den posten som ska tas bort
function saveChangeToArrayFound(&$found,&$number_post_found,
                                $result,&$resultat,&$whole_result)
{
     for($count=1;$count<=$number_post_found;$count++)
     {   //om du kommer till radio knappen som är ifylld
         if($count==$result)
         { //spara hela resultat(inte bara nyckeln)
           $whole_result=$found[$result];
           $resultat=strtok($found[$result]," ");//extrahera nyckel
           $resultat=trim($resultat);//bort med blanktecken
           //vald post tas bortfrån found array
           for($count_1=$count;$count_1<$number_post_found;$count_1++)
           {
                 $found[$count_1]=$found[$count_1+1];
           }
         }
     }
     $number_post_found--;//antalet hittade poster minskas
}

/*********************************************************************/
/*pre-anropas när man har klickat på knappen 'delete'                */
/*post-alla poster utom valda(som ska raderas) sparade i array post  */
```

```
/*********************************************************************/

function saveToArrayPost(&$post,&$number_post,
                         $resultat,$filename_database)
{
     if (!$fp = fopen($filename_database, 'r'))
     {
          print "Cannot open file ($filename_database)";
          exit;
     }

     $hit=false;

     while(!feof($fp))//till slutet av filen
     {
        $row=fgets($fp,1024);//en rad i taget
        $referens=strtok($row,"{");//extrahera referenstyp
        $key_dat=strtok(",");//extrahera nyckeln
        $key_dat=trim($key_dat);//eventuella blanktecken bort
         if((strcmp($referens,"@Book")==0 )||
            (strcmp($referens,"@Article")==0)||
            (strcmp($referens,"@InProceedings")==0)||
            (strcmp($referens,"@MastersThesis")==0)||
            (strcmp($referens,"@Misc")==0)||
            (strcmp($referens,"@Manual")==0)||
            (strcmp($referens,"@TechReport")==0)||
            (strcmp($referens,"@PhDThesis")==0)||
            (strcmp($referens,"@Proceedings")==0)||
            (strcmp($referens,"@Unpublished")==0)||
            (strcmp($referens,"@InBook")==0)||
            (strcmp($referens,"@Booklet")==0))
        {
               $hit=false;//nyckeln är inte hittat
               $number_post++;//starta ny post
               if(strcmp($key_dat,$resultat)==0)//hittat nyckel
               {
                   $number_post--;//den posten raderas
                   $hit=true;     //nyckeln är hittat
               }
               else
               {
                   $post[$number_post]=$row;/*Om nyckeln är inte hittat,
                                              lägg till raden i array post*/
               }
        }
        else if(!$hit)/*Om nyckeln(posten) är inte hittat, då fortsätt
                        att lägga till raderna i arrayen post*/
        {
                   $post[$number_post]=$post[$number_post].$row;
        }

     }
     fclose($fp);

}

/*********************************************************************/
/*pre-anropas när man har klickat på knappen 'delete'                */
/*post-alla poster utom valda(som ska raderas) sparade i filen(databasen)*/
/*********************************************************************/

function writeToFile($post,$number_post,$filename_database)
```

```
{
    if (!$fp = fopen($filename_database, 'w'))//filen öppnas för skrivning
    {
          print "Cannot open file ($filename_database)";
          exit;
    }
    //alla poster från arrayen post sparas i filen
    flock($fp,2);
    for($j=1;$j<=$number_post;$j++)
    {
           fwrite($fp,$post[$j]);
    }
    flock($fp,3);
    fclose($fp);
}

/*************************************************************************/
/*pre-anropas när man har klickat på knappen 'Search'                    */
/*post-alla poster som innehåller inmatat sökord (i fältena author,year  */
/*     eller title eller i nyckel) sparade i arrayen found.I arrayen     */
/*     sparas bara de följande fältena-author,title och year och nyckel  */
/*     och referenstyp.                                                  */
/*************************************************************************/

function saveToArrayFound($key,$filename_database,
                          &$found,&$number_post_found)
{
     if (!$fp = fopen($filename_database, 'r'))//filen öppnas för läsning
     {
          print "Cannot open file ($filename_database)";
          exit;
     }
     while(!feof($fp))//till slutet av filen
     {
        $row=fgets($fp,1024);//en rad i taget
        $ref=strtok($row,"{");//ta första token
        $field_name=strtok($row," ");
        if((strcmp($ref,"@Book")==0 )||(strcmp($ref,"@Article")==0)||
           (strcmp($ref,"@InProceedings")==0)
           ||(strcmp($ref,"@MastersThesis")==0)||
           (strcmp($ref,"@Misc")==0)||(strcmp($ref,"@Manual")==0)||
           (strcmp($ref,"@TechReport")==0)||
           (strcmp($ref,"@PhDThesis")==0)||
           (strcmp($ref,"@InBook")==0)||(strcmp($ref,"@Unpublished")==0)||
           (strcmp($ref,"@Proceedings")==0)||(strcmp($ref,"@Booklet")==0))
        {
              $found_key=false;   /*I början av posten sätts de alla till
                                    falsk,det vill säga ingenting är hittad*/
              $found_author=false;
              $found_title=false;
              $found_year=false;
              $ref=strtok($row,"{");
              compareKey($ref,$key,$found_key,
              $number_post_found,$found,$temp_found);
        }
        //sökordet letas i tre olika fälten(author,year och titel)
        else if(strcmp($field_name,"author")==0)
        {
              compareField($found,$number_post_found,$found_author,
                           $found_key,"\tAuthor: ",$temp_found,"",$key);

       }
```

```
        else if((strcmp($field_name,"title")==0))
        {
                compareField($found,$number_post_found,
                            $found_title,($found_key||$found_author),
                            "\tTitle: ",$temp_found,"",$key);

        }
        else if((strcmp($field_name,"year")==0))
        {
                compareField($found,$number_post_found,$found_year,
                            ($found_key||$found_author||$found_title),
                            "\tYear: ",$temp_found,"",$key);
        }
      }
      fclose($fp);
.
}

 /*************************************************************************/
/*pre-anropas när man har klickat på knappen 'search'                     */
/*post-inmatade sökordet jämförs med de nycklarna i databasen(filen)      */
/*************************************************************************/

function compareKey($ref,$key,&$found_key,
                    &$number_post_found,&$found,&$temp_found)
{               //extrahera referenstyp
                $take_ref=substr($ref,1,(strlen($ref)-1));
                $key_database=strtok(",");//extrahera nyckeln
                /*Lagra innehållet i temp variabeln,ifall om du inte hittar
                sökordet i nyckeln,men hittar det i author eller titel eller
                år*/
                $temp_found=$key_database." ".$take_ref."  ";

                if(strcmp($key_database,$key)==0)
                {
                         $found_key=true;     //nyckeln hittat
                         $number_post_found++;//antalet hittade ökas
                         //informationen sparas i arrayen found
                         $found[$number_post_found]=$key_database.
                                                  " ".$take_ref."  ";
                }
}

function compareField(&$found,&$number_post_found,&$found_field,
                      $found_check,$print,&$temp_found,$temp_field,$key)
{
    $temp_second=strtok("{\"");//tills {
    $temp_field=strtok("}\"");//extrahera author
    $temp_field=trim($temp_field);//alla blanktecken bort
    if($found_check)//om sökordet är redan hittat
    {
            $found[$number_post_found]=$found[$number_post_found].
                                    $print.$temp_field;
    }
    $field_array=explode(" ",$temp_field);//bryta ner fältens värde i ord
    for($counter=0;$counter<count($field_array);$counter++)
    {
             /*Jämför fältets värde(i filen och inmatat),om det är redan
               hittat då jämförelsen inte (undviker dubbletter)*/
             if(strcmp($field_array[$counter],$key)==0 && !$found_check)
             {
                  $found_field=true;//sökordet hittat
```

```
                    $number_post_found++;
                    $found[$number_post_found]=$temp_found.
                                        $print.$temp_field;

                }
        }
        //om sökordet finns i nästa fält
        $temp_found=$temp_found.$print.$temp_field;
    }

    /*********************************************************************/
    /*pre-true                                                           */
    /*post-radioknappar med beskrivande text utskrivna på sidan          */
    /*      tillsammans med knapparna 'delete' och 'previous'            */
    /*********************************************************************/

   function writeToPage($found,$number_post_found,$search,$whole_result)
   {
        //print"<FORM name=\"form1\"
                ACTION=\"".$PHP_SELF."\" METHOD=POST>";
        $default=false;
        print "<FORM ACTION=\"".$_SERVER["PHP_SELF"]."\" METHOD=POST>";
        if(strcmp($whole_result,"")!=0)
        {
           $whole_result_1=strtok($whole_result," ");
           $whole_result_2=strtok("");//resten(allt utan nyckeln)
           print"<font color=blue size=4>".
                 $whole_result_2." deleted from the
                 Bib database</font><br>";
        }
        print "<font color=blue size=4>Sökning resultat</font>";
        print "<Table>";
        for($j=1;$j<=$number_post_found;$j++)
        {
           $cut_first=strtok($found[$j]," ");//ta bort nyckel
           //resten ska bli text vid radio knapparna
           $cut_second=strtok("");
           print "<tr><td><input type=radio
                 name=\"result\" value=".$j.">".$cut_second."<br></td>";
           print "<TD><input type=hidden
                 name=\"found[".$j."]\" value=\"".$found[$j]."\"></tr>";
        }
        print "<tr><TD><input type=hidden
              name=\"search\" value=".$search.">";
        print "<TD><input type=hidden
              name=\"number_post_found\" value=".$number_post_found."></tr>";
        print "<tr><td colspan=3><input type=submit
              name=\"delete\" value=Radera></td>";
        print "</form></b>";
        print "<td><form  name=\"form2\" action=\"Search_s.php\" method=post>";
        print "<input type=submit
              name=\"Back\" value=Föregående></center></td>";
        print "<td><input type=hidden
              name=\"search\" value=".$default."></td>";
        print "<td><input type=hidden name=\"delete\" value=".$default.">";
        print "</td></form></tr>";
        print "</table>";
   }
?>
```

B Java Applet - källkod

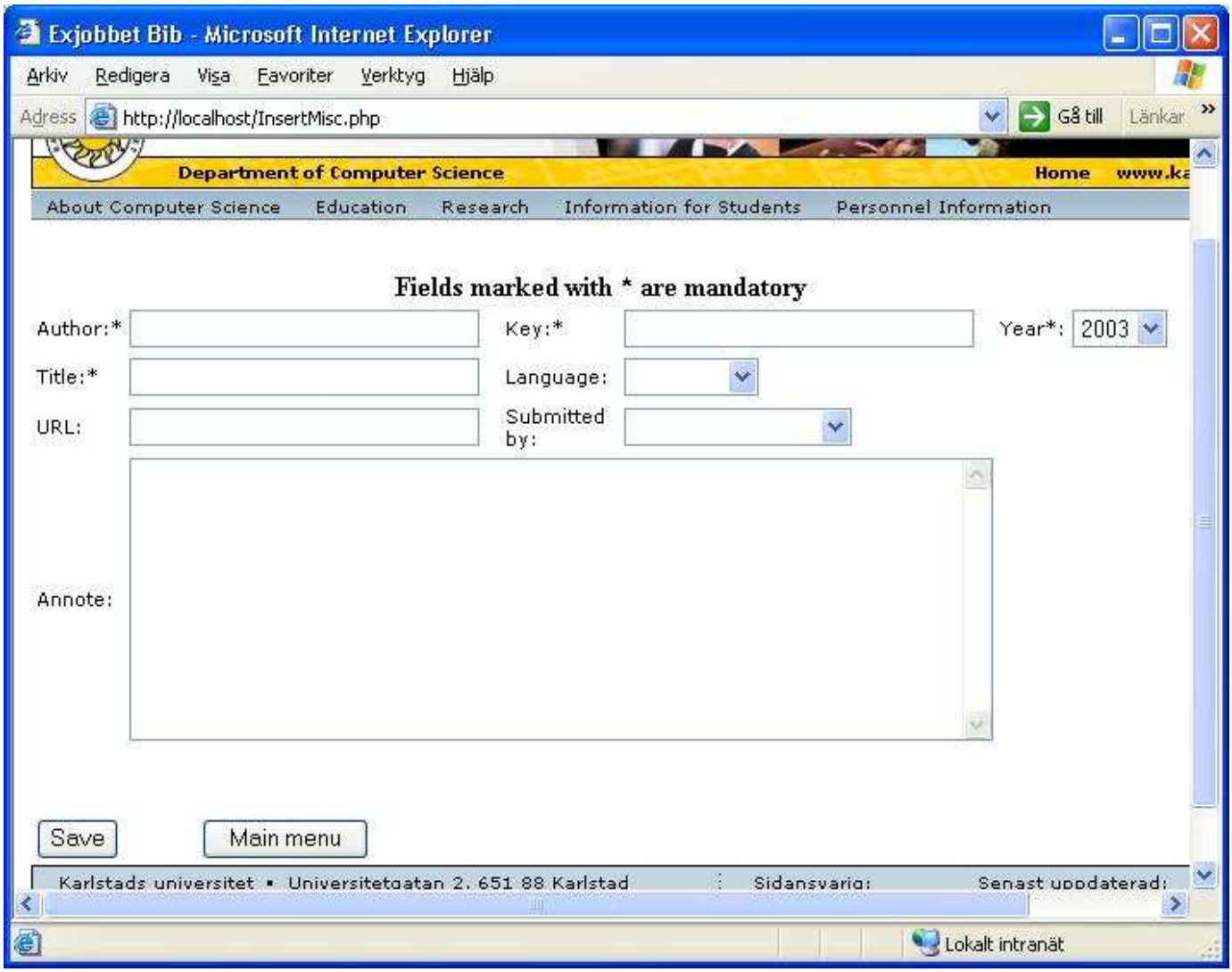
```
 /*                 Textremsa
  För att använda,skriv följande i din HTML sida:
  <applet code=Textremsa.class width=600 height=30>
  <param name=TEXT VALUE="Din text här!">
  <param name=FONT VALUE="Courier">
   </applet>
Parameter - beskrivning:
  TEXT: specificerar textsträngen som ritas på remsan
  FONT: specificerar fonten som används
        för att rita textsträngen*/
import java.awt.*; import java.applet.*; import java.awt.event.*;
public class Textremsa extends Applet implements Runnable {
    Thread runner;
    final int FONT_MAX=24;//max och min fontstorlek
    final int FONT_MIN=8;
    final int MOVE_HOR=1;//texten rör sig på tre olika sätt
    final int MOVE_VER=2;
    final int MOVE_CURVE=3;
    final int SPEED_MAX=30;
    final int SPEED_NORMAL=150;
    final int SPEED_MIN=300;
    String font;
    Image bufferScreenImg;// buffrad skärmbild (dubbel-buffrad animation)
    Graphics Gfx;
     String textString;// Textsträng som kommer att visas i
                       // appleten lagras här
    //initialisera appleten
    public void init()
    {
         super.init();
        textString = getParameter("TEXT");//hämta texten från applet tagg
        if (textString == null)//om ingen text anges
        {
            textString = "NOTHING";//default värde(text)
       }
       font = getParameter("FONT");//hämta font från applet tagg
       if (font==null)//om ingen sträng anges
                font=new String("Courier");//default värde(font)
        //Gör gömd skärm för dubbelbuffring av animation
        bufferScreenImg = createImage(getSize().width,getSize().height);
        Gfx = bufferScreenImg.getGraphics();

    }
    //skapa nytt tråd och starta tråden
    public void start()
    {
        if (runner == null)
        {
            runner = new Thread(this);
            runner.start();
        }
    }
    //stoppa tråden
    public void stop()
    {
        if (runner != null)
        {
            runner.stop();
            runner = null;
```

```
    }
}
/*I den här metoden flyttas texten i olika riktningar.
  Den anropas när programmet(applet) exekveras.*/
public void run()
{
   while (true)
   {
     moveText(SPEED_MAX,getSize().width,
              MOVE_HOR,Color.black,Color.red);
     moveText(SPEED_MIN,getSize().height,MOVE_VER,
              Color.blue,Color.green);
     moveText(SPEED_NORMAL,(getSize().height/4),
              MOVE_CURVE,Color.blue,Color.green);
   }
}
 /*Texten flyttas uppåt(mode=1),nedåt(mode=2),
   eller enligt y=2*x kurvan (mode=3).*/
 public void moveText(int speed,int size,int mode,
                      Color Color1,Color Color2)
 {
   int text_size=0;//initialisera textens storlek
      boolean up=true;
      //ändra textposition(bredd och höjd)
     //textstorlek varierar mellan FONT_MIN och FONT_MAX
     for (int text_position = 0; text_position < size;text_position++)
     {
       if(((text_size+FONT_MIN)<FONT_MAX)&&up)
       {
             text_size++;
             if((text_size+FONT_MIN)==FONT_MAX)
             {
                    up=false;
             }
       }
       else if(((text_size+FONT_MIN)>FONT_MIN)&&!up)
       {
             text_size--;
               if((text_size+FONT_MIN)==FONT_MIN)
             {
                   up=true;
             }
          }
          //öka storleken på texten
       Font myfont=new Font(font,Font.BOLD,FONT_MIN+text_size);
       //sätt font, färg  på appleten(hälften svart,hälften röd färg)
       Gfx.setFont(myfont);
       Gfx.setColor(Color1);
          Gfx.fillRect(0,0,getSize().width,getSize().height/2);
          Gfx.setColor(Color2);
          Gfx.fillRect(0,(getSize().height)/2,
                       getSize().width,getSize().height/2);
          //ändra färgen på texten(med hjälp av random generator)
          Gfx.setColor(new Color((int)(255*Math.random()),
                      (int)(255*Math.random()),(int)(255*Math.random())));
          //visa texten på två ställe(textremsan)
          if(mode==1)
          {
            Gfx.drawString(textString,text_position,
                             (getSize().height)/4);
            Gfx.drawString(textString,text_position,
                          3*(getSize().height)/4);
```

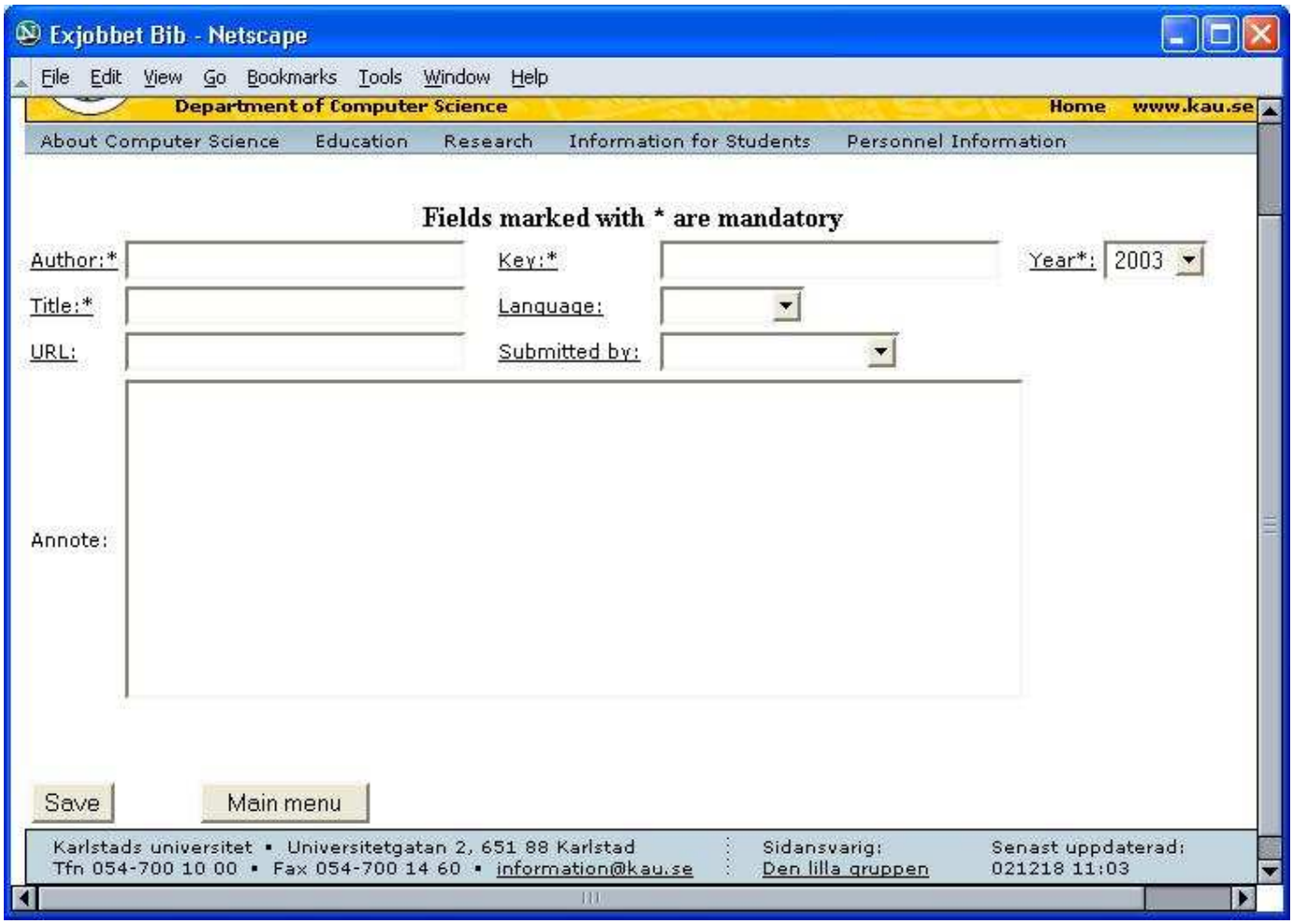
```
                repaint();
            }
            else if(mode==2)
            {
                     Gfx.drawString(textString,0,text_position);
                    Gfx.drawString(textString,(getSize().width/2),
                                   text_position);
                 repaint();
            }
            else
            {
                   Gfx.drawString(textString,text_position,
                                  4*text_position);
                 repaint();
            }
            try
            {
                  Thread.sleep(speed,speed);
            }
            catch( InterruptedException e)
            {
            }
          }
    }
    // Rita buffer på skärmen (dubbel-buffrad animation)
    public void paint(Graphics screenGfx)
    {
          screenGfx.drawImage(bufferScreenImg,0,0,this);
    }
    public void update(Graphics g)
     {
        paint(g);
    }
}
```

C Skärmdumpar

Figur C.1: Användargränssnittet för Menu.php

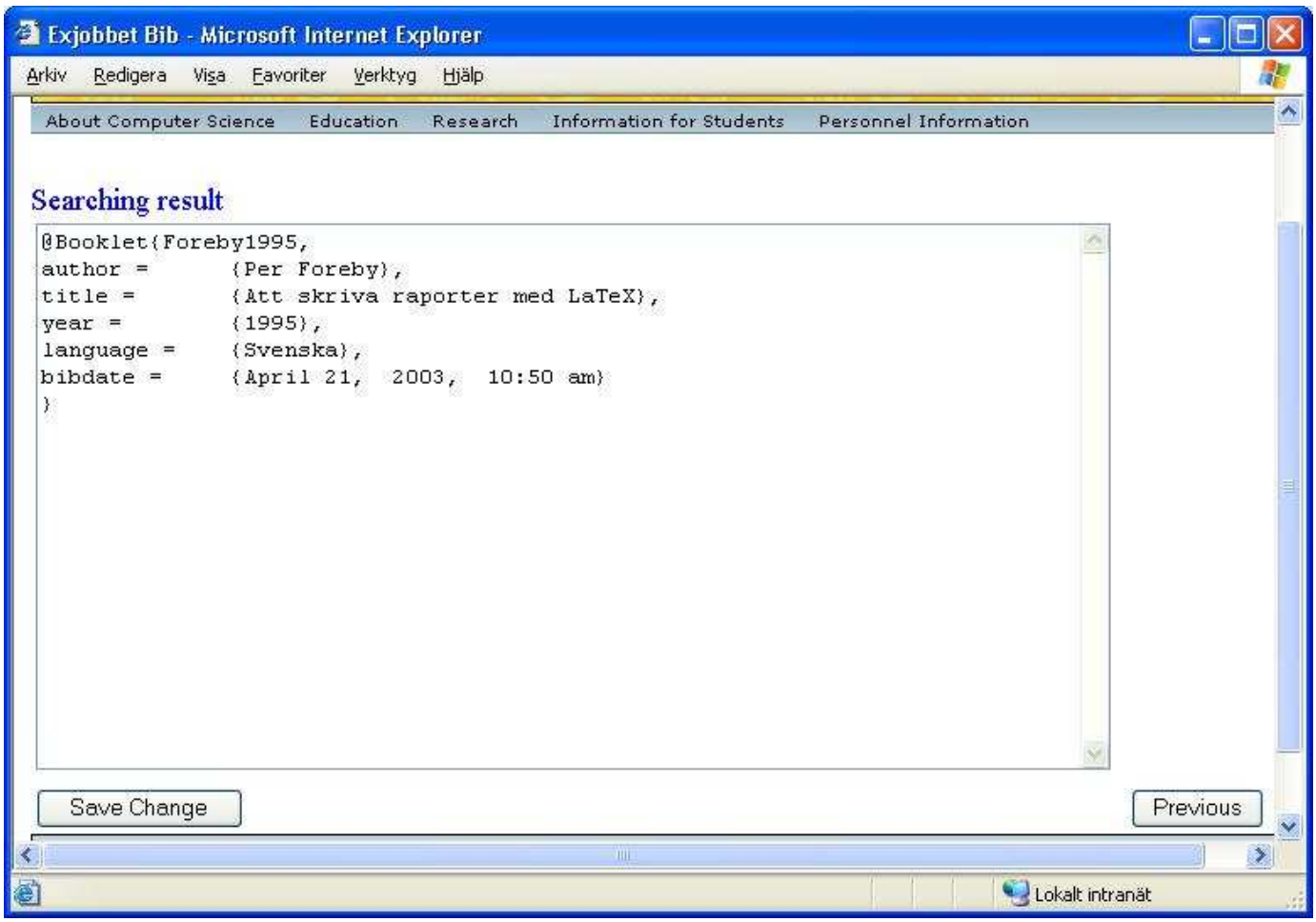


Figur C.2: Användargränssnittet för InsertMisc.php

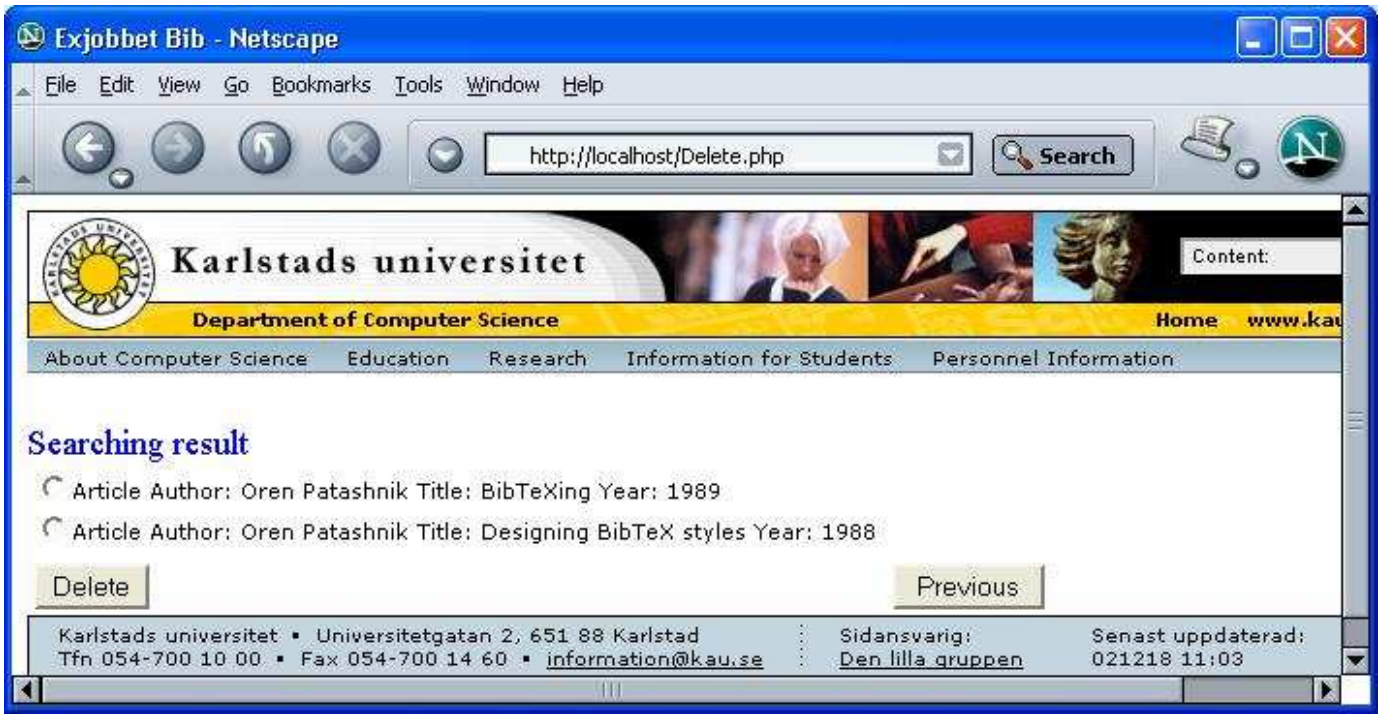


Figur C.3: Användargränssnittet för InsertBooklet.php

Figur C.4: Användargränssnittet för Search_change.php



Figur C.5: Användargränssnittet för Change.php



Figur C.6: Användargränssnittet för Delete.php