Min dIntProg-browser

Martin Geisler

Uge 49, 2001

Resumé

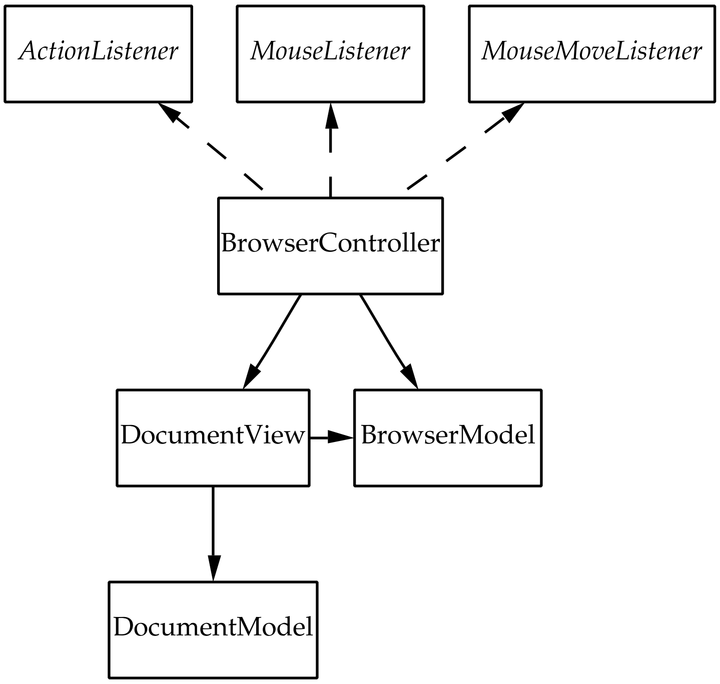
Dette dokument beskriver tankerne bag min dIntProg-browser, en browser skrevet i Java der skal kunne fortolke en mindre delmængde af HTML 4, kaldet HTML_0. Der gives forklaring af designet og af hvordan de forskellige klasser arbejder sammen. Eventuelle fremtidige udvidelsesmuligheder diskuteres også.

Indhold

Introduktion

At skulle skrive en webbrowser viste sig at være en utrolig spændende og meget udfordrende opgave. Jeg har beskæftiget mig med HTML i flere år, så det var meget interessant selv at skulle designe det program, der skal vise det.

Jeg har lagt mest vægt på selve den måde, hvorpå siden bliver tegnet, da jeg synes, at det er det vigtigste ved en browser. Jeg ville gerne have et system, der var fleksibelt, og som gjorde det rimelig nemt at lave "avancerede" ting som f.eks. tabeller. Derfor valgte jeg at lave min egen rekursive datastruktur til opbevaring af informationerne fra siden. Det viste sig, at det vigtige element i denne datastruktur var en *boks*.



Figur 1: Klassehierarkiet for hovedklasserne i browseren.

1 Klasserne

Klassernes funktioner er allerede beskrevet i API dokumentationen. Jeg vil her blot angive klassehierarkiet og skitsere deres måde at arbejde sammen på. Klassehierarkiet ses på figur 1.

BrowserController er kontrolløren der sørger for at vedligeholde modellen i form af klassen BrowserModel og displayet i form af et objekt af typen BrowserView.

Kontrolløren udvider klassen JFrame, hvilket måske umiddelbart ikke passer helt ind i Model-View-Controller mønsteret. Men da det er kontrollørens opgave at lytte efter hændelser, og reagere på disse, synes jeg at det er mest naturligt at lade denne lave det vindue hvor hændelserne bliver genereret.

I klassen BrowserModel bliver der holdt styr på listerne med besøgte URL'er og den slags.

Klassen BrowserView udvider JComponent og kan derfor indsættes i f.eks. en JScrollPane, sådan som det bliver gjort i browseren. Komponenten bliver får en DocumentModel med når den bliver skabt — det er denne model den viser. Modellen bliver lavet af klassen Parser når man kalder parse(URL) i denne.

2 Bokse

En boks er et abstrakt begreb der dækker over et rektangulært område. Et sådant område har en højde og en bredde — det er i princippet alt hvad man har brug for at vide om en boks.

Boksene kommer i spil når HTML-dokumentet bliver parset. Dokumentet bliver da delt op i en lang række bokse, hvoraf nogle bokse indeholder andre bokse. En konkret boks vil nedarve fra AbstractBox, der implementerer Box.

Alle formateringsalgoritmerne arbejder med disse abstrakte bokse. De ved altså ikke noget om hvad der er inden i en given boks, de kender kun boksens højde og bredde. Det gør, at tingene bare virker når man tilføjer nye bokse til hierarkiet. Klassehierarkiet for boksene kan ses på figur 2 på næste side.[1]

En abstrakt boks er som sagt kendetegnet ved at den har en højde og bredde. Den har desuden også en minimumsbredde og en bedste bredde. Det er idéen at man skal undgå at lave bokse mindre end deres minimumsbredde, og at man skal forsøge at lave dem så bredde som deres bedste bredde. Hvis man alligevel laver en boks mindre end dens minimumsbredde, så kan man forvente at boksen vil blive ved med at tegne sig selv med en bredde af dens minimumsbredde.

En boks har nemlig ansvaret for at tegne sig selv, når drawAt(int, int, Graphics) bliver kaldt på den. Man må dog ikke bede en boks om at tegne sig selv, før man har kaldt dens doLayout(Graphics, JComponent, int) metode.

Boksen får en x og y koordinat og en Graphics kontekst med, når den skal tegne sig selv. Punktet (x, y) bliver boksens øverste venstre hjørne på DocumentView. Boksen kan så frit disponere over rektanglet der strækker sig fra (x, y) til $(x + \mathsf{getWidth}(), y + \mathsf{getHeight}())$, men den skal ikke tegne uden for dette område.

De generelle bokse bliver delt op i to grupper: *fleksible* og *stive* bokse.

2.1 Fleksible bokse

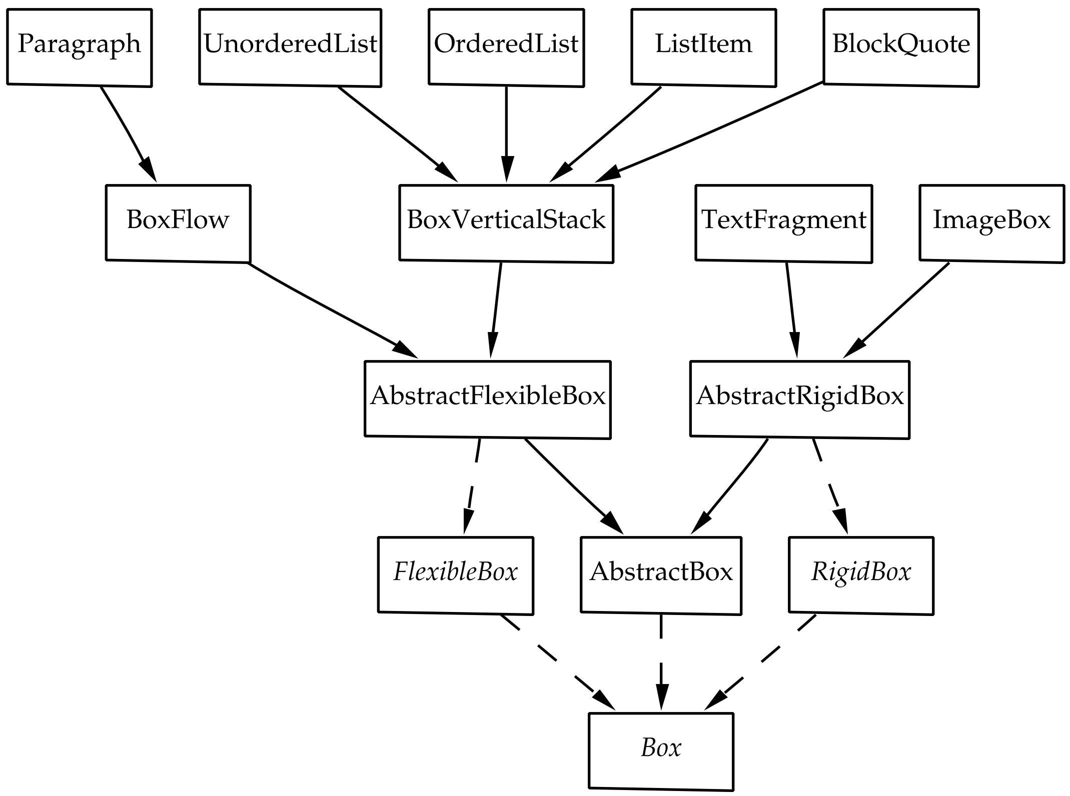
Fleksible bokse er bokse der kan indeholde andre bokse, og som desuden kan ændre deres størrelse trinløst. Konkrete fleksible bokse vil nedarve fra AbstractFlexibleBox, der implementerer FlexibleBox.

Hele designet bygger på, at man har bokse inden i bokse. Når siden er parset, skal man ende med netop én boks der indeholder alle andre bokse på siden. Denne boks vil være en boks, der kan stable andre bokse oven på hinanden — en VerticalBoxStack. Der er en tilsvarende HorizontalBoxStack, der placerer bokse på linie.[2]

En fleksibel boks har ansvaret for, at bede de bokse den indeholder om at tegne sig selv. En tabel er f.eks. en fleksibel boks. Den beder

[1] Det er ikke alle klasser der nedstammer fra Box der er med på diagrammet, men kun dem jeg endte med at bruge. Jeg lavede flere klasser undervejs, men fik ikke brug for dem alle sammen til sidst.

[2] Denne klasse bliver dog ikke brugt i øjeblikket.



Figur 2: Klassehierarkiet for de klasser der nedarver fra Box.

sine bokse om at tegne sig selv, sådan at de kommer til at stå i række og kolonne. Den har forud for dette regnet på deres højde og bredde, efter at have kaldt deres doLayout(Graphics,JComponent,int) metode.

2.2 Stive bokse

Stive/faste bokse er bokse der implementerer RigigBox. Disse bokse kan ikke ændre deres højde og bredde trinløst sådan som en fleksibel boks kan. I stedet må man bede en stiv boks om at dele sig.

Da man ikke kan sætte højde og bredde, må en fast boks være i stand til at finde sine mål, så snart den har fået adgang til et `Graphics` objekt og/eller en `JComponent` objekt.

Hvis man skal placere en fast boks, og man ved at den er for stor, kan man spørge om den kan deles ved et kald til splitIsPossible(int). Den heltals parameter man giver med, skal være det antal pixels fra boksens forkant hvor man ønsker at delingen skal ske. Hvis metoden returnerer sandt, kan man gå videre og kalde splitHead(int) og splitTail(int) for at få fat i den forreste og bagerste del af den splittede boks. Det vil være sådan, at den forreste del er mindre end den parameter man gav til splitHead(int), mens den bagerste del vil udgøre resten af den oprindelige boks. Der vil blive konstrueret nye bokse ved disse kald.

3 Mine udvidelser

Jeg har udvidet browseren på flere områder, bl.a. ved at tilføje support for nye tags:

⟨table⟩ Der er support for tabeller. Tabellerne består af en række celler af typen TableData, der selv er en subtype af VerticalBoxStack. Det betyder at man kan have alle slags bokse i tabeller, også tabeller inden i andre tabeller.

Søjlerne i en tabel får til at starte med en bredde der mindst er lig deres minimumsbredde. En søjles minimumsbredde er lig den største minimumsbredde af cellerne i den pågældende søjle, hvilket betyder at det er de største celler der bestemmer fordelingen af pladsen.

Hvis der stadig er ekstra bredde, bliver den overskydende bredde fordelt mellem søjlerne, sådan at den der mangler mest også får mest. Det giver nogle rimelig pæne tabeller.

⟨blockquote⟩ En ⟨blockquote⟩ rykker marginen ind med et lille stykke i både venstre og højre siden. En ⟨blockquote⟩ er ikke en subklasse af Paragraph, men derimod af VerticalBoxStack der også bruges til selve siden, så man kan have alle slags fleksible bokse i et objekt af typen BlockQuote.

⟨ol⟩ Der er lavet support for ordnede lister, dvs. lister hvor hvert punkt er nummereret. Der kan selvfølgelig være flere af disse lister inden i hinanden — lister på et ulige niveau vil blive nummereret med "1.", "2." og så videre, mens lister på lige niveauer vil blive nummereret med "(a)", "(b)" og så videre.

Overskrifter Der er support for ⟨h1⟩, ⟨h2⟩ og ⟨h3⟩. Disse tags giver selvfølgelig overskrifter i forskellig størrelse, fra størst til mindst. Det ville være nemt at udvide systemet til flere niveauer af overskrifter, men i praksis har man sjældent brug for mere end tre niveauer af overskrifter.

Skrivemaskine skrift Parseren understøtter tre tags, der alle giver tekst skrevet med en `Monospaced` font: ⟨code⟩, ⟨tt⟩ og ⟨pre⟩. Det sidste tag er lidt specielt, da det egentlig er meningen, at det også skulle bevare mellemrum. Det gør det dog ikke, da parseren ikke leverer mellemrum enkeltvis, og heller ikke skelner mellem mellemrum og linieskift.

Nye entities Jeg har tilføjet support for nogle HTML entities jeg manglede, se tabel 1.

Entity	Tegn
®	ⓡ
©	ⓒ
á	á
à	à
Á	Á
À	À
é	é
è	è
É	É
È	È

Tabel 1: Nye HTML entities.

Kommandolinieparametre Browseren accepterer op til tre parametre på kommandolinien. Man kan skrive en URL, der så vil blive brugt som hjemmeside.

Man kan også skrive `-d` for at slå debugging til. Browseren vil så skrive hvad den gør via System.err. Man vil desuden se nogle bokse bliver tegnet op i selve browseren. Hvert TextFragment bliver markeret med et orange rektangel og hver BoxFlox markeres med et rødt rektangel. Det gør det nemt at se hvordan boksene bliver brudt.

Endelig kan man skrive `-a` for at slå antialiasing fra. Det kræver en del ydelse at lave antialiasing, så derfor kan man slå det fra, sådan at gentegningen går hurtigere. Man vil især mærke det når man scroller op og ned på lange sider.

4 Fremtidige udvidelsesmuligheder

Der er en række udvidelsesmuligheder i browseren. For det første kunne man forbedre brugergrænsefladen — jeg har som sagt lagt mest vægt på selve det at tegne siden, og ikke så meget på indpakningen.

Det ville f.eks. være rart/smart med en menu hvor man kunne vælge at slå debugging og antialiasing til og fra. Man kunne også lave en menu hvor man kan vælge en basis fontstørrelse hvis man er utilfreds med den der bruges som standard.

Man kunne tilføje support for ⟨pre⟩ hvis man ændrede parseren, sådan at den ikke ignorerer flere på hinanden følgende mellemrum. Som det er nu, bliver flere mellemrum blot returneret som ét enkelt mellemrum. Parseren skulle endvidere også skelne mellem mellemrum og linieskift.

Det ville være rart hvis browseren understøttede links til ankre i teksten, og ikke kun til hele sider. Det ville sikkert være muligt at implementere dette uden de helt store vanskeligheder, da vi allerede har koordinaterne af alle rektanglerne.

Man kunne også give alle bokse en margin. Som det er nu, er det op til den enkelte boks at lave luft omkring sig — det burde generaliseres ved at bruge klassen Insets fra Javas standard biblioteksklasser.

Endelig kunne man udvide understøttelsen af tabeller, sådan tabeller med celler der bruger `colspan` eller `rowspan` også kan vises. Som det er nu bliver disse tabeller ignoreret, da antallet af celler ikke er lig produktet af antal søjler og antal rækker. Jeg tror dog at det vil være vanskeligt at kode dette.

5 Fejl og mangler

Der er så vidt jeg ved ikke nogen mangler i browseren — den kan fortolke HTML$_0$ og lidt til. Der skulle heller ikke umiddelbart være nogen fejl.

Der er dog en ting der stadig driller: Når boksene bliver sat sammen til linier i BoxFlow laver den linieskift mellem to bokse. Det går også fint normalt, men det er et problem hvis man har en side med HTML-kode som dette:

⟨p⟩Nogle ⟨b⟩fede⟨/b⟩, ⟨i⟩kursive⟨/i⟩ og normale ord.⟨/p⟩

Her vil der blive lavet fem fragmenter, markeret med en lodret streg:

|Nogle |**Fede**|, |*kursive*| og normale ord.|

Problemet er, at man kan risikere at linien bliver delt lige efter ordet "fede", men før kommaet. Resultatet bliver dette:

|Nogle |**Fede**|
|, |*kursive*| og|
|normale ord.|

Læg mærke til at det sidste fragment er blevet splittet op i to, da der lige var plads til ordet "og" på næstsidste linie.

Løsningen på dette problem kunne være, at man samler den slags kritiske bokse i større grupper. Det er hvad RigidBoxGroup er til for. Hvis man putter nogle bokse ind i en RigidBoxGroup, så vil de blive sammen, da en RigidBoxGroup nægter at lade sig dele. Det lykkedes mig dog aldrig at få parseren til at dele fragmenterne rigtigt og gruppere dem, så klassen ligger ubrugt hen.

Jeg har dog oplevet nogle problemer når jeg prøver at parse "rigtige" sider, altså sider skrevet i HTML 4. Men som regel går det ikke værre end at man kan trykke på tilbage-knappen.