LINKÖPINGS UNIVERSITET

TDP007: Konstruktion av datorspråk

Utvecklarblogg
Seminarieuppgift 2

Författare

Daniel Huber, danhu849@student.liu.se
Theodore Nilsson, theni230@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Vårterminen 2021
Version 1.0

2021-01-28

Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Utvecklarblogg, Seminarie 2 TDP007	280121

Vilka fel och misstag har ni gjort under tiden?

Flera gånger har vi försökt använda metoder på en instans av ett nil-objekt som uppstått av att det mönster vi försökt matcha mot inte gett någon retur då den inte funnit en matchning. Detta hände både när vi använde regex och xpath. Vi tycker inte att debuggingen där ger så mycket.

När vi jobbade med tabeller och regex kunde detta åtgärdas genom att städa tabellerna ytterligare, och när vi jobbade med xpath berodde det mestadels på att vi inte var bekanta med hur de fungerade.

Andra fel som gjort att vi fastnat är att vi glömt att hela tabellen har datatypen stringtrots att den innehåller en mängd siffror. Vi har försökt hantera dessa som ”integer”utan att inse vad det är som gått snett.

På samma sätt glömde vi bort att början på en rad i tabellen kunde bestå av vita tecken och att split på detta genererade nil.

Vad finns det för nya konstruktioner i Ruby som ni inte sett förut?

- Att utropstecknet i sort_by! gör att sorteringen utförs i samma array och inte genererar en kopia.
- Att ett bindessträck efter en sort_by! gör att sorteringen inventeras:

```
1    self.sort_by! do |day| -
2
```

Finns det något som ni irriterar er på eller tycker om i Ruby?

Möjligheten att använda attribute accessors gör klasser mycket mer lättöverskådliga och minskar kodbasen markant.

Det har varit skönt att inte behöva bry sig så mycket om saker så som minneshantering och datatypssäkerhet. Det är en stor skillnad från C++ där ofta gäller att hålla reda på flera saker även för att göra någonting relativt enkelt som att skapa en ny funktion.

I den dokumentation vi hittat för ruby (ruby-doc.org) ger sökningar resultat för äldre versioner. Detta kan vara förvirrande om en inte är beredd på detta. Det skulle vara önskvärt att resultat som berör den senaste versionen skulle prioriteras.

Reflektioner kring de olika tekniker som ni stöter på:

Hur lätt/svårt är det att sätta sig in i?

Ruby är likt Python på så sätt att båda är lekstugor för programmerare. Även om det inte ser ut att det kommer fungera så gör det troligtvis det ändå. Exempelvis fungerar den vid ytan tveksamma koden nedan:

```
1        wanted_data = rows[5..-2].each do |row|
2             weather_array.push( Day_Temp.new( row.scan(/([0-9]+)/)[0..2].join(" ").split)
```

Har ni hittat alternativa källor för att ta reda på nya saker?

Vi har förutom ruby-doc.org och rubular använt w3schools för information rörande xpath syntax.

Dokumentation av hur ni har tänkt när ni arbetat fram era lösningar:

Vad arbetade ni med (i grova drag) vid varje labbpass?

Vår prioritet var att vid varje deluppgift först försöka skriva ut med puts vad varje del av koden åstadkom. Därefter modifierade vi delar av koden och skrev ut dem tills dess att vi uppnått det vi ville. Trial and error.

Hur har ni tolkat uppgiften?

Vi har tolkat det som att de endast de av uppgiften specificerade värdena ska presenteras. Alla andra värden, exempelvis alla andra värden än högsta och minsta temperatur i andra delen av första uppgiften kan kasseras. I andra uppgiften är vi medvetna om att utskriften av vevent objektet inte behövde vara strukturerad, men valde att göra det ändå.

Vad var svårt eller lätt med uppgiften?

Det var svårt att veta hur DOM objektet kunde traverseras då det initiellt var oklart hur strukturen av elementen i root var såg ut. Efter hjälp från labledare om hur man utgick från nuvarande plats i xml-trädet kom vi vidare.

Reflektera över hur mycket av koden från första delen som gick att återanvända inför andra delen av uppgiften.

I första uppgiften kunde ca 50% av koden återanvändas då initiala matchningen av datan var samma. Datan som önskades låg innanför <pre> taggarna. Andra uppgiften var svårare att dela upp då väderdatan var mer heterogen. I fotbollsfilen hade varje lag en punkt på femte raden exempelvis. Utöver detta ändrades bara namnet på klassen till databehållaren