Utvecklarblogg Seminarium 2

Den här dokumentationen är för utvecklarblogg till seminarium 2 i TDP007, meningen med att skriva detta dokument är för att redovisa hur vi har tänkt/tolkat och löst uppgifterna.

- **UPPGIFT 1**

I denna uppgift använde vi oss av **REGEX** för att fånga den texten som vi är intresserade av. Vi tolkade uppgiften att man ska fånga antal gjorda *(F)* och insläppta *(A)* mål i *"football.txt"* samt lägsta *(MnT)* och högsta *(MxT)* dygnstemperaturen i *"weather.txt"*. Därefter ska man räkna den minsta skillnaden och man ska i slutet sortera de efter den minsta skillnaden i en *Array*.

Vi skannade hela filen och kunde lägga till alla matchningar i en *Array* genom att använda funktionen *football_file/weather_file"* Sedan räknade vi skillnaden i alla lagen/dygn och printade vi ut de i funktionen *goal_diff/weather_diff* där vi använda oss av *each loopar* och absolutbelopp funktionen för att skillnaden ska alltid vara positive. Vi kunde dock sortera listan efter den valda index genom att använda den inbyggda funktionen *sort_by* .

Sedan har vi skrivit funktion som heter *least_goal_diff/least_temp_diff* som i sin tur kan printa det minsta skillnaden genom att returnera det första valda index i den sorterade *Arrayen* .

Denna uppgift var både lätt och svårt, lätt eftersom vi hade arbetat förut med regex trots att vi hade tyvärr spenderat mycket tid för att hitta det rätta regex som passar till varje text fil. Det var lätt också att genomföra testerna eftersom det skiljer inte sig mycket av den första seminariet. Det var svårt att kunna ignorera av de mellanslag som ska inte följa med på matchningen samt på vilket sätt ska man läsa från de olika textfilerna eftersom det finns så mycket olika sätt.

- **UPPGIFT 2**

I denna uppgift fick vi lösa problemet genom trädparsning och XPath trots att vi först försökte strömparsa filen. Det skillnaden mellan trädparsning och strömparsning var att i strömparsning måste vi ha gått igenom hela texten tecken för tecken medans i trädparsning behövde vi att gå igenom hela texten på en gång. Anledningen till att denna uppgift tog längre tid var för att vi försökte skriva ett Ruby program för att strömparsa filen men lyckades vi inte till slut.

Sedan fattade vi beslutet att använda XPath för att trädparsa filen vilket var ganska lättare och koden blev betydligt kortare också. Med XPath kunde vi lätt hämta alla

elements och sedan hantera varje element som ett objekt i klassen ***MyCalendarClass***.

Enligt uppgift beskrivningen så behövde vi skriva ett Ruby program som kan läsa filen och skapa lämpliga Ruby objekt som innehåller alla kalenderhändelser som finns i en webbsida. Vi tolkade uppgiften så att vi måste skriva ett program som vi kan plocka alla händelser från webbsidan och kunna använda den som objekt också. Vi insåg också att det inte räcker med bara namnen på händelser då det var 2 olika händelser som vars skulle ske 4 gånger det vill säga både "**The Dark Carnival - 101.9FM**" och "**Sinister Sundays**". Så vi tänkte lägga till datumet med i vår program för att specificera varje händelse efter datumet.

Det andra problemet vi stött på med den här uppgiften var att skriva testfall för dem, vilket var lite problematisk och tidskrävande för oss. Men efter hjälp från assistenten så märkte vi att vi hade ett litet fel med tests funktions namnet, där måste man alltid kalla den funktionen som man ska testa i med *test_u2* och inte *u2_test* det vill säga att det måste alltid börja med ordet **test.**

Vi testade funktionen *find_all_info* genom att använda *assert_equal* samt *assert_not_equal* där vi testade till exempel att första event i calender vilket är *"The Dark Carnival - 101.9FM"* ligger i index 0 i Arrayen *events_list* .
De källorna vi fick hjälp av var b.l.a **föreläsningar** och **w3school.com.**