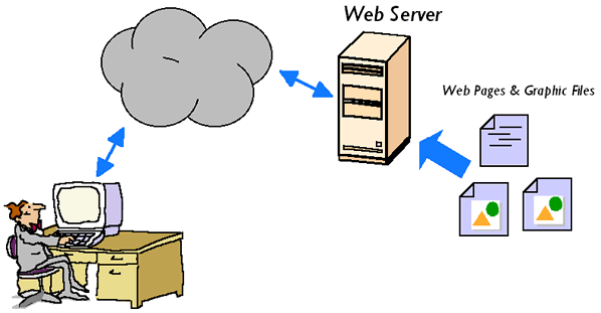
Programvaruteknik DVG302 vt2016

Inlämningsuppgift 4

Servlet

Bakgrund

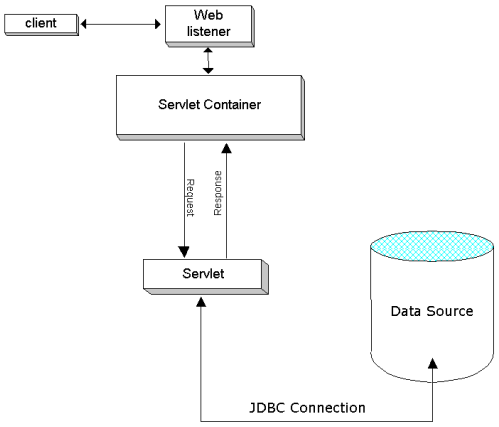
De första hemsidorna var endast statiska textdokument (som innehöll html-kod) lagrade på en webserver. När man ”gick in på en hemsida” så innebar det att man via webservern hämtade ett sådant textdokument (med filändelsen .html), sparade den temporärt på sin egen dator och lästes av (avkodades) i en webläsare så att text, bilder och annat visades som en ”hemsida”.



Detta fungerar inte så bra på hemsidor som hela tiden vill ändra sitt innehåll. Ta en nyhetssida till exempel, där har man ett skelett som är konstant (huvudrubriker och annat ligger hela tiden på samma ställe) men vad som visas under varje huvudrubrik kan ändras från den ena minuten till den andra beroende på vad som tidningen vill meddela sina läsare. Även utrymmen för loggor och reklam ligger ofta fast men innehållet byts ut. Därför är sådana sidor inte statiska html-dokument utan innehållet ligger på andra ställen (oftast i en eller flera databaser) och när man vill hämta en sådan sida, på samma sätt som tidigare, så är det aktiv kod som svarar på anropet och som ”bygger” den kodsträng (i princip samma typ av kod som står i ett statiskt html-dokument). På samma sätt som ni skulle kunna skriva en Javaapplikation som skriver ut ”Hello World” på konsolen så kan en sådan här applikation ”skriva” html-kod (kom ihåg att websidor bara är text i form av html-kod som tolkas i en webläsare) på out-stream från webservern (dvs som om man laddat ned en html-fil). Detta sköter webservern om, det enda man själv måste bry sig om är att skapa denna aktiva kod som knackar ihop html-text. Den här tekniken är alltså en *server-side* teknik för aktiva hemsidor och har funnits länge i olika versioner (ASP, CGI osv..) och finns hos många olika språk (bl.a. PHP, Pearl och andra). I princip kan man skriva sådan här aktiv kod i vilket språk som helst, det gäller bara att det finns någon typ av webserver som kan ”köra den” och ta hand om kopplingen mellan en klientbegäran (någon som försöker hämta en hemsida) och den aktiva koden som snickrar ihop html-koden.

Uppgift

Inom Javavärlden så kan dessa aktiva hemsidor skapas via en *Tomcat* server (en typ av webserver med nått som kallas *Servlet Container*) och den aktiva koden i Java kallas då *servlet*. Ni har nu som uppgift att skapa en *servlet* som via *Tomcat* (eller annan serverteknik) svarar på ett anrop. Resultatet skall vara det som ni skapade i en tidigare inlämningsuppgift, nämligen att hämta data om fotbollsresultat från allsvenskan och köra det mot fejkat data från sinusfunktionsklassen. Resultaten skall returneras som en *Json* formaterad sträng som skrivs ut i webläsaren.



För att öva på att kunna ta emot argument till en *servlet* så skall ni även kunna skicka med en parameter så att ut-datat formateras så att det blir mer läsbart.

Ex:

```
http://localhost/statistik
```

Skall alltså ge en Json formaterad sträng med resultatet från körningen av de två datakällorna (en enda lång sträng utan whitespace eller return).

```
{"2014-03-01":{"x":0.966428374821654,"y":12.333333333333334},"2014-04-
01":{"x":0.7976843827047911, osv..

http:/localhost/statistik?pretty=true
```

Ge en *Json* formaterad sträng med resultatet från körningen av de två datakällorna men som har rader och indenteringar (ungefär som vanlig källkod i era projekt). Detta gör att ni mycket enkelt kan kontrollera att det blir korrekt.

```
{
   "2014-03-01":{
      "x":0.966428374821654,
      "y":12.333333333333334
   },
   "2014-04-01":{
      "x":0.7976843827047911,
 osv..
```

TIPS: Använd Rasmus kod ”JasonFormatter” som ligger som bilaga sist i detta dokument.

För övrigt så kan ni för att titta på Json strängar ladda ned en plugin till er webläsare så att det visas på ett mer läsbart sätt. Eller, om ni kör Crome så finns det inbyggt (googla på det).

JUnit:

Inget krav i denna inlupp men testa gärna i alla fall.

Produktionskod:

Koden blir i form av en servletklass som ni kör på en *Tomcat* server (lokal). Testar gör ni genom att köra det via en Browser (webläsare). Observera att det är bara *Json* strängen som skall skrivas ut i webläsaren (rå eller formaterad enligt ovanstående).

Rapport

Självklart skall ni skriva en rapport, men ni behöver ju inte ordbajsa bara för att få till mycket textvolym, när nu uppgiften är så pass liten. Men alla delar skall finnas med korrekt format och språk. Som resultat går det alldeles utmärkt att visa exempelkörning (skärmdumpar) utöver ren beskrivande text. Formalia enligt tidigare inlämningsuppgift. D.v.s. använd mallen och var noga med referenser och annat.

Kod lämnas endast in som bilaga i rapporten, ingen extra zip-fil.

Bilaga1. Rasmus Östbergs JsonFormatter

```
/**
 * Is used to format strings to a JSON format that is easy to read.
 *
 * @author Rasmus Östberg
 *
 */
public class JsonFormatter {
     /**
      * Formats the given string to a readable JSON format
      *
      * @param unformatedString
      * @return String the formated String.
      */
     public String format(String unformatedString) {
          String base = "";
          int depth = 0;

          for (int i = 0; i < unformatedString.length(); i++) {
               char c = unformatedString.charAt(i);
               if (isBracket(unformatedString.charAt(i))) {
                    if (isleftBracket(c)) {
                         depth++;
                         base += c;
                         base += "\n";
                         base += getTabs(depth);
                    } else {
                         depth--;
                         base += "\n";
                         base += getTabs(depth);
                         base += c;
                    }
               } else if (isComma(c)) {
                    base += c;
                    base += "\n";
                    base += getTabs(depth);
               } else {
                    base += c;
               }
          }
          return base;

     }

     protected boolean isleftBracket(char c) {
          return (c == '[' || c == '{');
     }

     protected boolean isRightBracket(char c) {
          return (c == ']' || c == '}');
     }

     protected boolean isBracket(char c) {
          return isleftBracket(c) || isRightBracket(c);
     }

     protected String getTabs(int tabs) {
          String base = "";
          for (int i = 0; i < tabs; i++) {
               base += "\t";
          }
          return base;
     }

     protected boolean isComma(char c) {
          return c == ',';
     }
}
```