Pawlowsky

HuntingServlet Übung

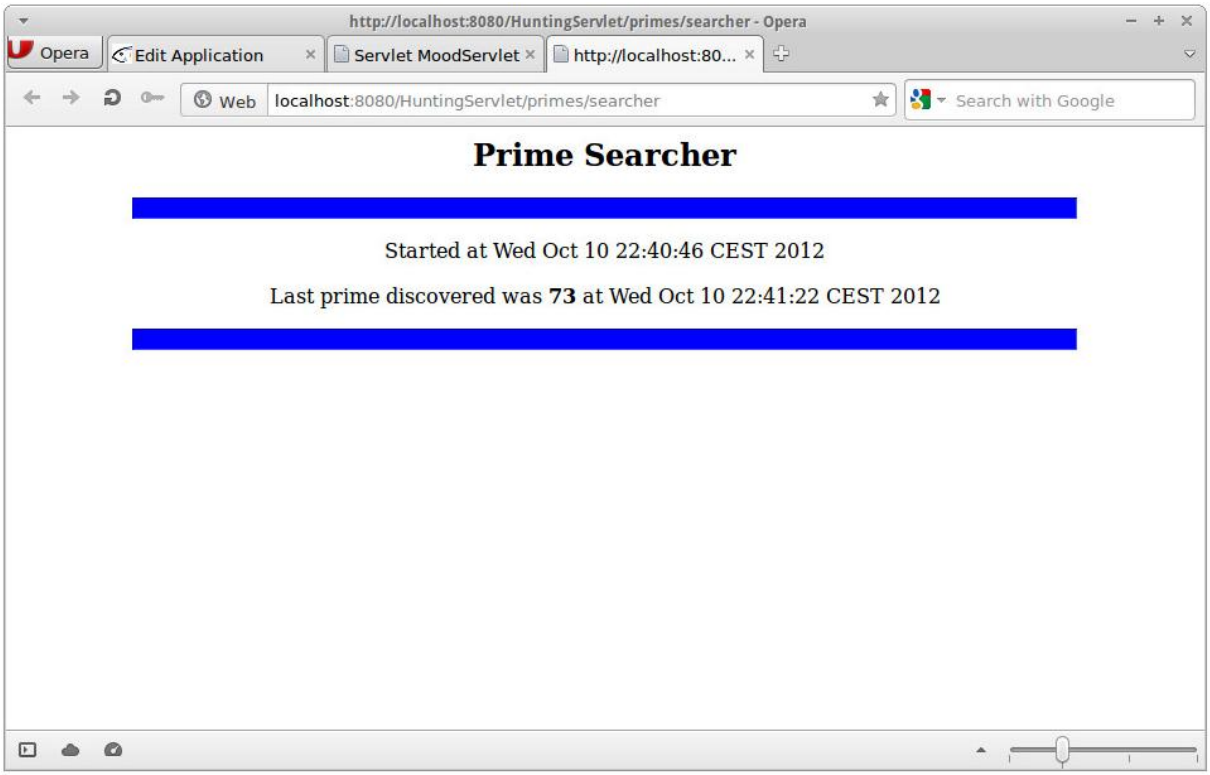
APR 5BHITS 14.10.2012

Pawlowsky

Inhalt

Aufgabenstellung

Ziel:



Aufgabenstellung:
Erstelle eine Web-Applikation welche nach Primzahlen forscht.

Die Applikation soll auf /primes deployed sein.
Das Servlet wird mittels /primes/searcher aufgerufen.
Richte eine Redirection mittels Servlet von /primes auf /primes/searcher ein
Solange das Servlet im Web-Container läuft soll weiter nach Primzahlen geforscht werden.
Falls das Servlet vom Container entfernt oder undeployed wurde, soll das Servlet die Möglichkeit erhalten die Suche sauber zu beenden.
Webbrowser-Clients können zu beliebigen Zeiten den aktuellen Stand abrufen!
Achte auf eine sparsame Nutzung der Ressourcen des Applikationsservers!
Achte besonders auf eine ausführliche Dokumentation der Sourcen und des web deployment descriptors (web.xml).

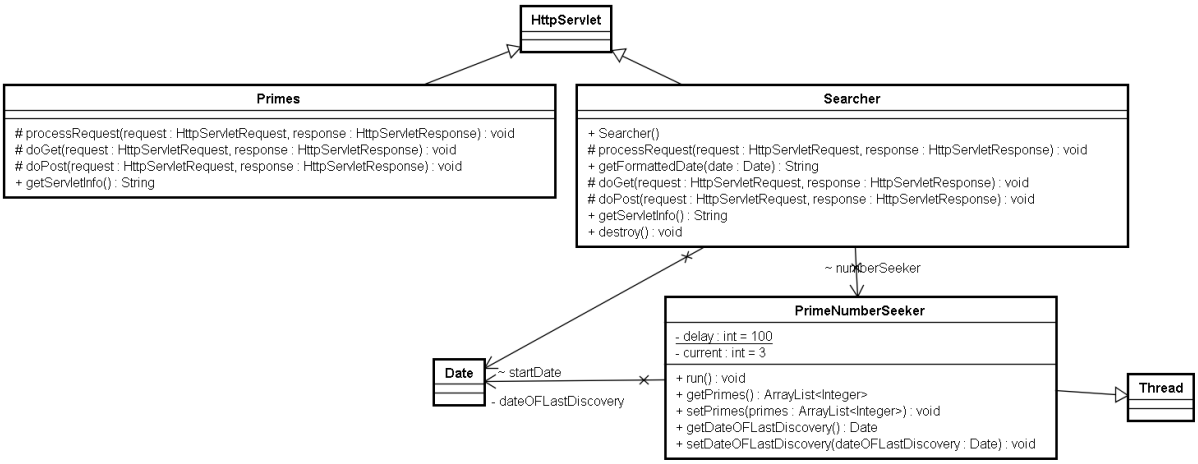
Abgabe:

Gesamtes Projekt inklusive Dokumentation und *.war

Punkte (16):
Ausführliche Sourcedokumentation (4), Ressourcennutzung (4), Undeployment (2), Deployment (4), Protokoll (2 .. -4)

Designüberlegung

Mein grundsätzliches Klassen-Design sieht in etwa so aus:



Ich habe den Thread zur Berechnung der Primzahlen, wie man sehen kann, auf eine eigene Klasse ausgelagert. Dadurch wird es einfacher ihn als Thread zu verwalten und etwaigen Komplikationen bezüglich mehrerer Funktionalitäten in einer Klasse können so leicht aus dem Weg gegangen werden.

Die Berechnung der Primzahlen habe ich wie folgt gelöst:

```
/**
 * This run method of the thread discoveres prime numbers by saving<br>
 * the old prime numbers and trying to divide the new prime number<br>
 * by any of the old. If it can be divided by one of them its not a<br>
 * prime number. Otherwise it is saved to the list of prime numbers.
 */
public void run() {
    primes = new ArrayList<Integer>();
    primes.add(2);
    while(true){
        //boolean indicating if the current number is a prime.
        boolean isPrime = true;
        //trying to divide it by the old prime numbers
        for(int i = 0; i < primes.size(); i++){
            if(current%primes.get(i) == 0){
                isPrime = false;
            }
        }
        //If it is still a prime after trying to divide it...
        if(isPrime){
            //add the prime number to the list
            primes.add(current);
            //Set the date of discovering the last prime number
            this.setDateOFLastDiscovery(new Date());
        }
        //Sleep for the specified delay.
        try {
            sleep(delay);
        } catch (InterruptedException ex) {
            Logger.getLogger(PrimeNumberSeeker.class.getName()).log(Level.SEVERE, null, ex);
        }
        //increment the currently investigated number
        current++;
    }
}
```

Zum Formatieren des Datums habe ich ebenfalls eine separate Methode geschrieben.

Außerdem wurde zum sauberen Undeployment des Servlets, die destroy()-Methode implementiert. Sie interrupted den Thread, sodass dieser noch maximal eine Primzahl berechnet und dann aufhört.

Das web.xml-File sieht so aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
    <servlet>
        <servlet-name>Primes</servlet-name>
        <servlet-class>primes.Primes</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Primes</servlet-name>
        <url-pattern>/Primes</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
</web-app>
```

Man erkennt das jedem Servlet ein Name gegeben wird und dann diesem Namen einen Pfad in der URL (localhost:8080/…) zugeteilt wird. Außerdem ist es ersichtlich, dass ein Timeout einer möglichen Websession einstellbar ist. In diesem Beispiel ist sie auf 30 gesetzt.

Aufwandsabschätzung

Protokoll:1,5h
Implementierung der Servlets: 2,5h
Gesamt: 4h

Arbeitsdurchführung

Pawlowsky:

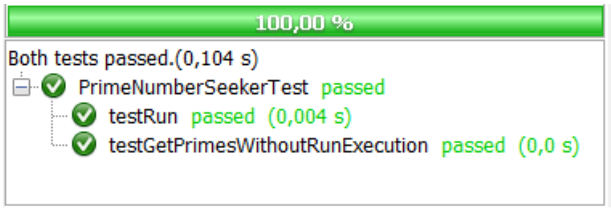
11.10.2012 Implementierung 2,5h
14.10.2012 Protokoll 1,5h
Gesamt: 4h

Resultate/Niederlagen

Richtige Niederlagen gab es nicht. Ich finde eigentlich, dass Servlets auf dieser Ebene ganz angenehm zu programmieren sind.

Tests

Hier die Bestätigung, dass alle Tests erfolgreich durchführbar waren:



Quellen

http://docs.oracle.com/javaee/
http://docs.oracle.com/javaee/6/tutorial/doc/
http://jcp.org/en/jsr/detail?id=315
http://www.oracle.com/technetwork/java/index-jsp-135475.html
http://www.oracle.com/technetwork/java/javaee/documentation/code-139018.html
http://java.net/downloads/javaeetutorial/javaeetutorial6.zip

http://stackoverflow.com/questions/2385909/most-elegant-way-to-write-isprime-in-java
http://openbook.galileocomputing.de/javainsel9/javainsel_15_007.htm (Tipp von Sochovsky)