Einstieg in PatternTesting

Unter http://oli.blogger.de/stories/1355070/ wird erklärt, wie man PatternTesting in die Beispiel-Anwendung JugsBase der Stuttgart Test-Tage einbindet. Allerdings wurde dort die Bibliothek patterntesting-check-0.9.0.jar verwendet, die inzwischen in

- patterntesting-check-ct.jar und
- patterntesting-check-rt.jar

aufgeteilt wurde. Daher fangen wir noch einmal von vorne an.

Die Beispiel-Anwendung

Als Beispiel-Anwendung wird JugsBase verwendet, die sowohl einen einfache Web-Anwendung, als auch eine Swing-Oberfläche mitbringt.

Näheres zur Installation als Webanwendung finden Sie unter http://oli.blogger.de/stories/1309639/

Wenn Sie die Swing-Oberfläche ausprobieren wollen, starten Sie einfach gui.FinanzAnwendung als Java-Anwendung.

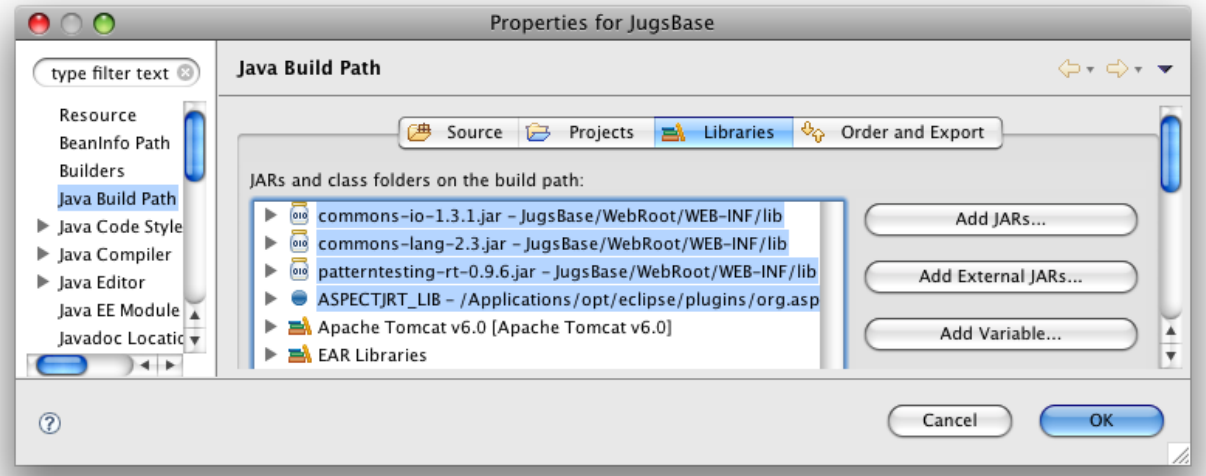
Zutaten

Zur Vorbereitung installieren wir noch das AJDT-Plugin für Eclipse (s. http://www.eclipse.org/ajdt/).

Einbindung als Java-Bibliothek

Alle PatternTesting-Bibliotheken benötigen die Runtime-Komponente als Grundlage. Daher werden wir im ersten Schritt patterntesting-rt.jar als normale Java-Bibliothek einbinden. Dazu laden wir patterntesting-rt von http://sourceforge.net/project/showfiles.php?group_id=48833 herunter und kopieren es ins `WebRoot/WEB-INF/lib`-Verzeichnis des JugsBase-Projekt.

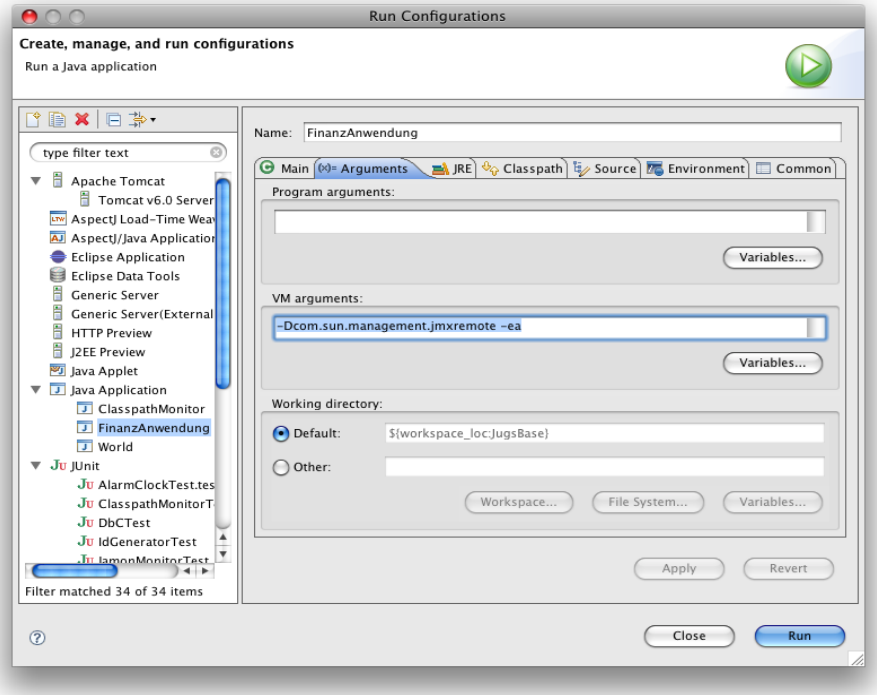
1. patterntesting-rt.jar von http://sourceforge.net/project/showfiles.php?group_id=48833 herunterladen
2. commons-lang-2.3.jar von http://commons.apache.org/lang/ herunterladen
3. commons-io-1.3.1.jar von http://commons.apache.org/io/ herunterladen
4. alle Jar-Dateien ins `WebRoot/WEB-INF/lib`-Verzeichnis des JugsBase-Projekt kopieren
5. in den Build-Path aufnehmen (Project > Properties > Java Build Path)
6. zusätzlich: ASPECTJRT_LIB als Variable hinzufügen (Add Variable...)



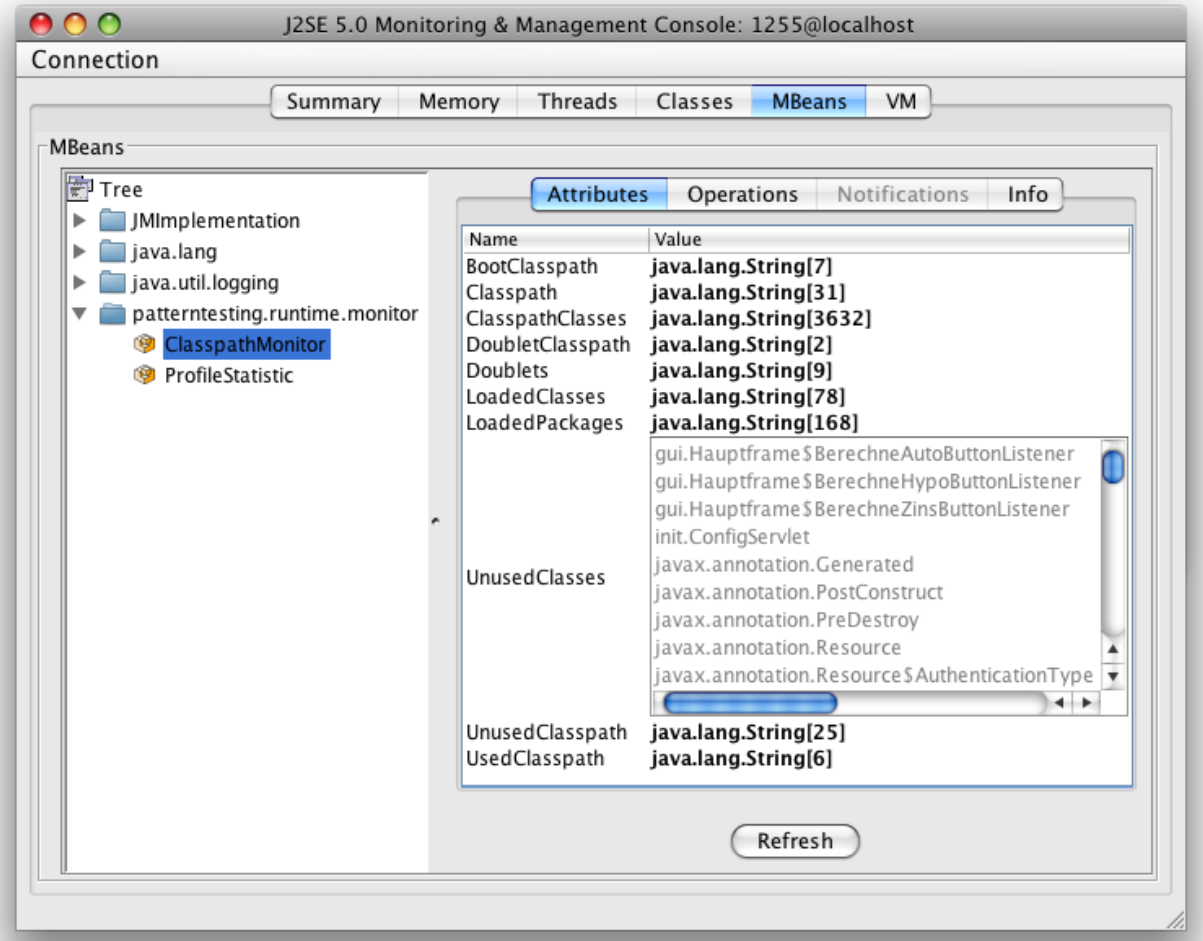
7. gui.FinanzAnwendung um die Anweisung `ClasspathMonitor.registerAsMBean();` erweitern:

```
package gui;

import patterntesting.runtime.monitor.ClasspathMonitor;

public class FinanzAnwendung {

	/**
	 * Startet die GUI fuer die Finanzanwendung mit
	 * Zinsrechner
	 * Hypothekenrechner
	 * Autokostenrechner
	 *
	 * Keine Parameter erforderlich
	 * @param args
	 */
	public static void main(String[] args) {
		ClasspathMonitor.registerAsMBean();
		new Hauptframe();
	}
}
```

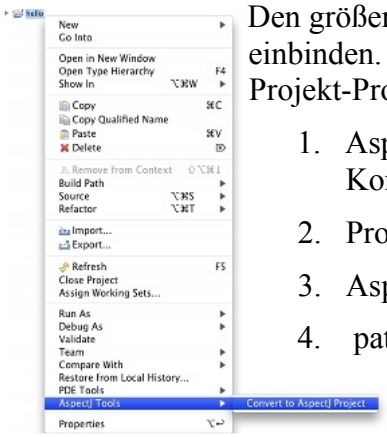
8. gui.FinanzAnwendung mit den Optionen `-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.local.only=false -ea` starten (wird für den Zugriff über die JConsole benötigt)



9. jconsole aufrufen, patterntesting.runtime.monitor.ClasspathMonitor (unter MBeans zu finden) öffnen und sich die UnusedClasses anschauen

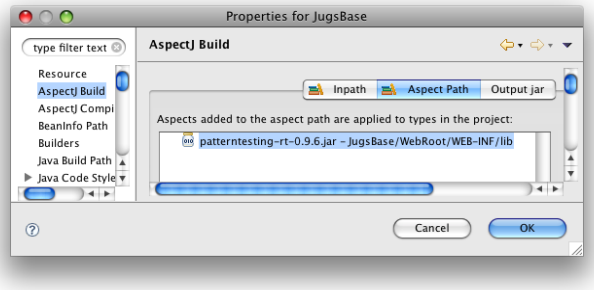


Einbindung als AspectJ-Bibliothek



Den größeren Nutzen bietet PatternTesting, wenn wir es als AspectJ-Bibliothek einbinden. Dazu müssen wir die Beispiel-Anwendung aber erst in ein AspectJ-Projekt-Projekt umwandeln:

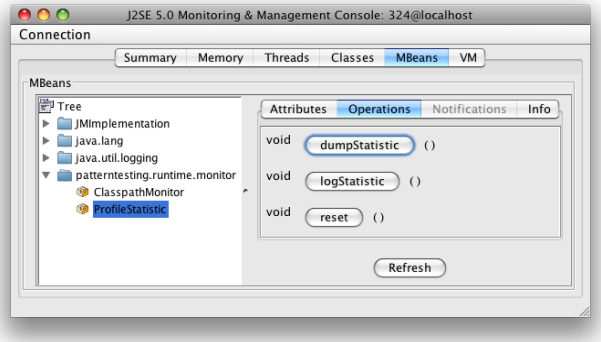
1. AspectJ Tools > Convert to AspectJ Project (Projekt selektieren und Kontext-Menü aufrufen)
2. Project > Properties
3. Aspect Build > Aspect Path auswählen
4. patterntesting-rt-0.9.6.jar eintragen
5. fertig



Jetzt können die Annotationen verwendet werden, die von PatternTesting zur Verfügung gestellte werden. Ausprobieren wollen wir es mit @ProfileMe, das wir der HypoZins-Klasse voranstellen:

```
package onlinerechner;

import patterntesting.runtime.annotation.ProfileMe;

/**
 * Berechnet aus den Eingaben eine Hypothek
 * @author iusft
 */
@ProfileMe
public class HypoZinsen {
    ...
```

Wenn wir jetzt die JConsole starten und die ProfileMontior-MBean unter patterntesting.runtime.monitor auswählen, so finden wir unter „Operations“ u.a. die Operation „dumStatistic“, die eine CSV-Datei erzeugt und im Temp-Verzeichnis ablegt.



Den Namen der erzeugten Datei wird im Log abgelegt. Schauen Sie in Eclipse auf der Console nach, wie die Datei heißt und öffnen Sie diese Datei mit OpenOffice oder Excel. Folgende Einstellungen müssen Sie dabei beachten:

- Feldtrenner: Strichpunkt

- Sprache: US-Englisch

Ohne diese Einstellung kann es passieren, dass manche Zahlung als Datum interpretiert werden. Als Ergebnis erhalten Sie z. B. folgende Tabelle (etwas gekürzt)[1]:

Label	Unit	Total	Avg	Hits	Max	Min
onlinerechner.HypoZinsen.berechneRestbetrag()	ms	0,31	0,15	2	0,17	0,14
new onlinerechner.HypoZinsen()	ms	0,03	0,01	2	0,02	0,01
onlinerechner.HypoZinsen.berechneMonatsrate()	ms	0,02	0,01	2	0,01	0,01
onlinerechner.HypoZinsen.setBetrag(double)	ms	0,01	0	2	0,01	0
onlinerechner.HypoZinsen.setLaufzeit(double)	ms	0,01	0	2	0	0
onlinerechner.HypoZinsen.setZinssatz(double)	ms	0,01	0	2	0	0
onlinerechner.HypoZinsen.setTilgungssatz(double)	ms	0,01	0	2	0	0
onlinerechner.HypoZinsen.getBetrag()	ms	0	NaN	0	0	1,#INF
onlinerechner.HypoZinsen.getLaufzeit()	ms	0	NaN	0	0	1,#INF
onlinerechner.HypoZinsen.getZinssatz()	ms	0	NaN	0	0	1,#INF
onlinerechner.HypoZinsen.getRestbetrag()	ms	0	NaN	0	0	1,#INF
onlinerechner.HypoZinsen.getMonatsRate()	ms	0	NaN	0	0	1,#INF
onlinerechner.HypoZinsen.getTilgungssatz()	ms	0	NaN	0	0	1,#INF

Zwei Dinge fallen dabei auf:

- manche Methoden wurden (noch) nicht aufgerufen (z.B. HypoZinsen.getBetrag()). Dies ist an der Spalte „Hits“ erkennbar
- die Auflösung beträgt ca. 0,01 ms – dies gilt allerdings nur für MacOS-X, Linux und vermutlich auch diverse Unix-Derivate. Unter Windows ist die Auflösung ca. 15 ms.

So, nun viel Spaß beim Ausprobieren der anderen Annotations (wie z.B. @NotYetImplemented) und der anderen PatternTesting-Bibliotheken (z.B. patterntesting-check-rt.0.9.6.jar).

Happy PatternTesting...

1 Die Zeiten der ClasspathMonitor-Klassen wurden hier weggelassen.