Einstieg in PatternTesting

Unter <u>http://oli.blogger.de/stories/1355070/</u> 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.

Aktuelle F	inanzrechner Base/ • Q+ Google	>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>				
Aktuelle Fi	nanzrechner		Cinsrechner	Betrag:	0.00 Erg	gel
Link zum Rechner	Beschreibung		Hypothekenrechner	Zinssatz:	0.00	
Zinsrechner	Zinsen berechnen		Autokosten-Rechner	Laufzeit:	0.00	
Hyporechner	Hypotheken berechnen				Berechne	
Autorechner	Autokosten berechnen					

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. <u>http://www.eclipse.org/ajdt/</u>).

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 <u>http://sourceforge.net/project/showfiles.php?group_id=48833</u> herunter und kopieren es ins WebRoot/WEB-INF/lib-Verzeichnis des JugsBase-Projekt.

- 1. patterntesting-rt.jar von <u>http://sourceforge.net/project/showfiles.php?group_id=48833</u> 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...)

0 0	Properties for JugsBase
type filter text Resource BeanInfo Path Builders Java Build Path Java Code Style Java Compiler Java Editor Java EE Module Javadoc Locatic	Java Build Path

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)

00	Run Configurations
reate, manage, and run configu	rations
Run a Java application	
Yppe filter text Type filter text Tomcat v6.0 Server Apache Tomcat Tomcat v6.0 Server Aspect] Load-Time Weat Aspect] Load-Time Weat Aspect] Load-Time Weat Eclipse Data Tools Generic Server Stava Application I ClasspathMonitor Ju Vorld Ju UbCTest Ju UdGenerator Test Ju JamonMonitor Tu JamonMonitor Tu JamonMonitor Tu JamonMonitor	Name: FinanzAnwendung
0	Close Run

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

Summan/ Ma	mony Threads	
Summary Me	mory meaus	Classes Midealis VM
Beans		
Tree	Attribute	operations Notifications Info
IMImplementation	Attribute	s operations motifications mo
iava.lang	Name	Value
iava util logging	BootClasspath	java.lang.String[7]
avaluation ogging avaluation ogging	Classpath	java.lang.String[31]
	ClasspathClasses	Java.lang.String[3632]
Classpatilionitor ProfileStatistic	Doublets	java.lang.String[2]
· Fromestatistic	LoadedClasses	java.lang.String[78]
	LoadedPackages	java.lang.String[168]
	UnusedClasses	gui.Hauptframe \$ BerechneAutoButtonListener gui.Hauptframe \$ BerechneHypoButtonListener gui.Hauptframe \$ BerechneZinsButtonListener init.ConfigServlet javax.annotation.Generated javax.annotation.PostConstruct javax.annotation.PreDestroy javax.annotation.Resource javax.annotation.Resource \$ AuthenticationType
	UnusedClasspath UsedClasspath	java.lang.String[25] java.lang.String[6]
		Refresh

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:

AspectJ Pro

0 0	Properties for JugsBa	ase	
type filter text ③	AspectJ Build	4	→ • ⇒ • •
Resource Aspectj Build Aspectj Compi BeanInfo Path Builders Iava Build Path		ath 📑 Aspect Path Ou	tput jar
	Aspects added to the aspect path are	applied to types in the project JugsBase/WebRoot/WEB-INF/li	roject:
► Java Code Style ▼			1
0		Cancel	ок

```
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.

	Summary Memory	Threads Classes MBeans VM
Beans		
 Tree JMmple java.lar java.lar java.lar apatern Clas Prof 	rmentation 19 Llogging testing-tumme.monitor spathMonitor	Attributes Operations Notifications Info void dumpStatistic () void logStatistic () void reset () Refresh

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.