Markus Grandpre, 12/07/10

Installation und Konfiguration der Chatbot-Software mit Eclipse

Die ChatterBob-Webapplikation und die Admin-Applikationen wurden in den folgenden Schritten erstellt und zur Anwendung gebracht:

1. Vorausetzungen
2. Quellcode mit SVN auschecken
3. Quellcode in Eclipse IDE integrieren
4. Anlegen der notwendigen Schemata in MySQL
5. Anpassen der ChatterBob-Webapplikation
6. Test der ChatterBob-Webapplikation
7. Test der ChatterBob-Adminapplikationen

Somit konnte die ChatterBob-Software auf einem lokalen System mit Hilfe der Eclipse IDE zum Laufen gebracht und getestet werden.

Vorausetzungen

Das lokale System ist ein Debian-Linux-System mit der Kernel-Version 2.6.32-5 für einen AMD 64 Dual-Core Prozessor mit 4 GB RAM. Zur Anwendung der ChatterBob-Applikationen wurde die folgende Software benötigt:

- Subversion-Client 1.6.12
- Java Developer Kit und Java Runtime environment 1.6.0_22
- Eclipse IDE (Helios Release, Build id: 20100617-1415)
- Apache-Tomcat Servlet-Container 6.0.29
- MySQL DB Version 14.14, Distribution 5.1.49

Quellcode mit SVN auschecken

Um den Quellcode mit SVN auszuchecken, wurde der folgende SVN-Befehl verwendet:

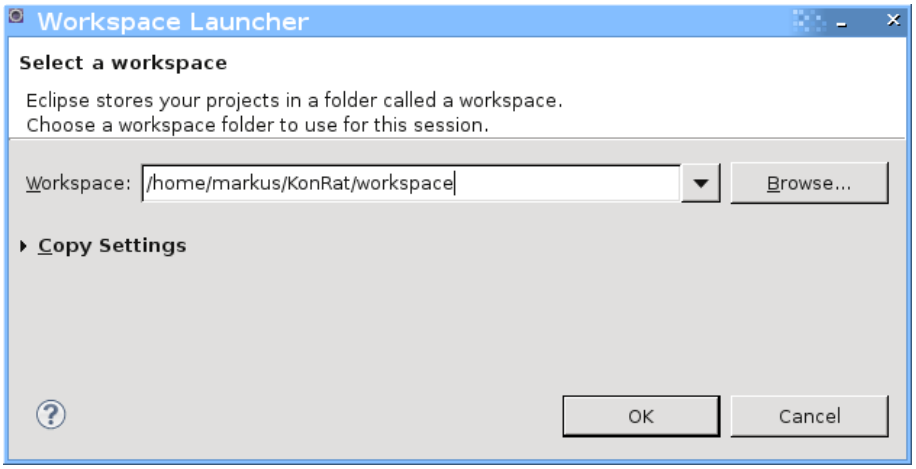
```
$ svn checkout http://chatterbot-bob.googlecode.com/svn/trunk/ \
    chatterbot-bob-read-only
```

Nach erfolgtem Checkout standen die Sourcen im Unterverzeichnis `chatterbot-bob-read-only/` zur Verfügung:

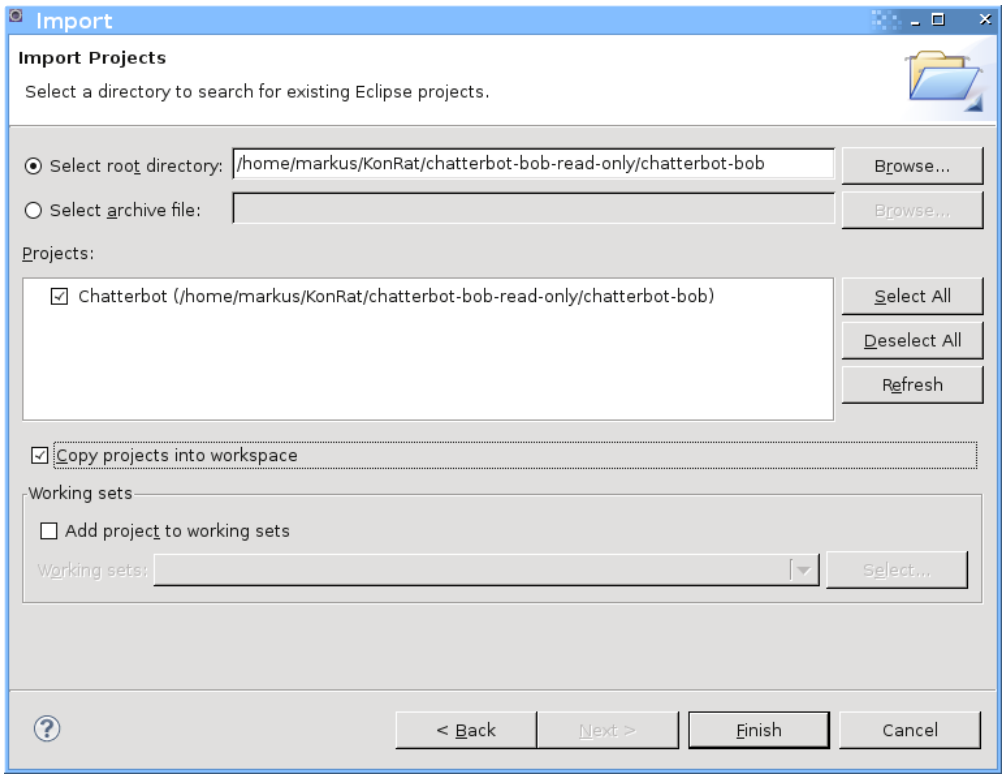
- *chatterbot-bob*: Web-Applikation
- *BCheck*: Administrations-Applikation
- *Macroparser:* Administrations-Applikation
- *QCheck*: Administrations-Applikation
- *TTCheck:* Administrations-Applikation
- *bob_knowledge_repository*: Dateien für die Web-Applikation
- *docs*: Dokumentation
- *libs:* JAR-Dateein zum Einbinden in den Build-Path unter Eclipse

Quellcode in Eclipse IDE integrieren

Um die Sourcen der ChatterBoB-Applikationen in Eclipse die IDE zu integrieren war es ratsam, einen eigenen Workspace (`$WORKSPACE`) anzulegen, um den Quellcode in die IDE zu importieren.



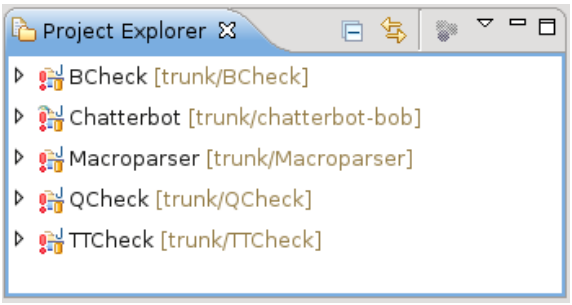
Da die ausgecheckten Sourcen bereits gültige Eclipse-Projekte waren, konnten sie leicht in den Workspace der IDE importiert werden (“*Existing Projects into Workspace*”):



Dabei mußten die folgenden Projekte:

- *chatterbot-bob* : Web-Applikation
- *BCheck:* Administrations-Applikation
- *Macroparser:* Administrations-Applikation
- *QCheck*: Administrations-Applikation
- *TTCheck*: Administrations-Applikation

einzeln importiert werden. Wichtig war, dass die Option "*Copy projects into workspace*" eingeschaltet war, sodaß sie Quellen nochmal in den eigentlichen Workspace kopiert wurden. Das hatte den Vorteil, dass Änderungen nun auf der Kopie ausgeführt wurden.



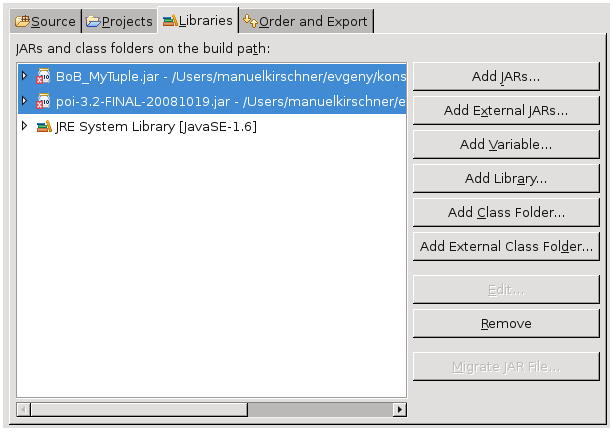
Das rote AusrufeZeichen deutete darauf hin, dass der sog. *Build-Path* (auch *Class-Path*) fehlerhaft war. Der nächste Schritt war demnach das Anpassen des Build-Paths der einzelnen Eclipse-Projekte im Workspace. Dazu wurden in jedem Projekt ein das Verzeichnis `lib/` angelegt:

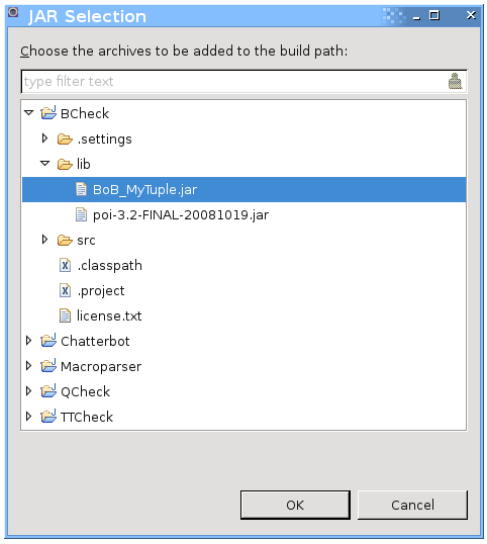
```
mkdir $WORKSPACE/BCheck/lib
mkdir $WORKSPACE/Chatterbot/lib
mkdir $WORKSPACE/Macroparser/lib
mkdir $WORKSPACE/QCheck/lib
mkdir $WORKSPACE/TTCheck/lib
```

um die folgenden, mit SVN heruntergeladenen, JAR-Files dort hinein zu kopieren:

- *BCheck*:
  - `BoB_MyTuple.jar`
  - `poi-3.2-FINAL-20081019.jar`
- *ChatterBot*:
  - `antlr.jar`
  - `BoB_Macroparser.jar`
  - `BoB_MyTuple.jar`
  - `jakarta-oro-2.0.8.jar`
  - `lingpipe-3.5.0.jar log4j-1.2.15.jar`
- *Marcoparser*:
  - `antlr.jar`
- *QCheck*:
  - `antlr.jar`
- *TTCheck*:
  - `antlr.jar`
  - `msv.jar`
  - `relaxngDatatype.jar`
  - `wstx-lgpl-3.2.9.jar`
  - `xsdlib-1.5.jar`

Dannach wurde der BuildPath für jedes einzelne Eclipse-Projekt angepaßt, indem die alten Referenzen auf die nicht mehr vorhandenen JAR-Dateien durch Referenzen auf die kopierten JAR-Dateien in den jeweiligen `lib/`-Verzeichnissen gesetzt wurden.

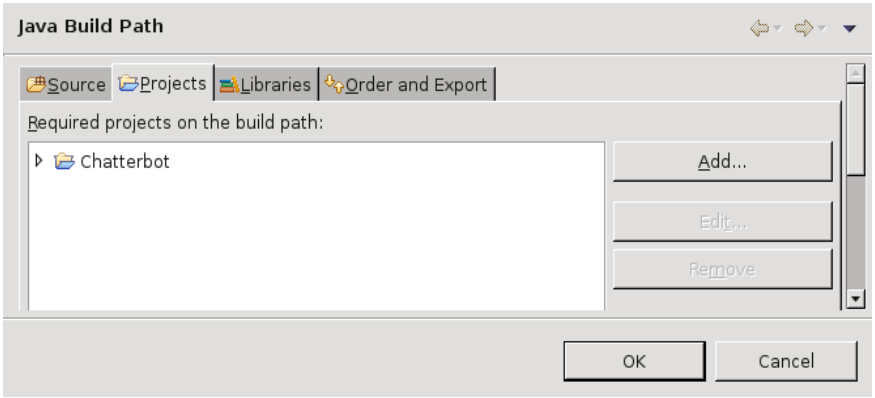




Zudem mußte darauf geachtet werden, daß die folgenden Projekte

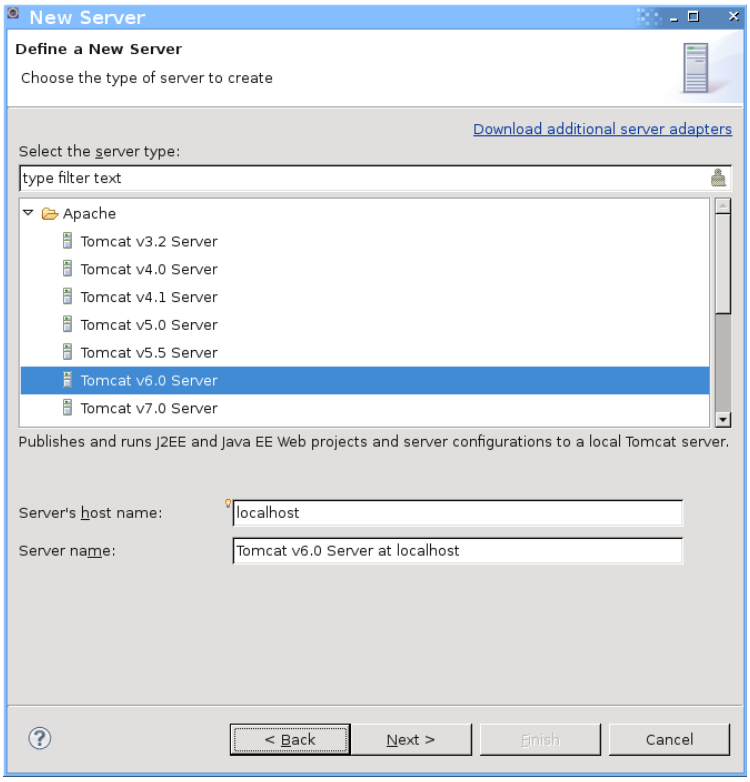
- *BCheck*
- *QCheck*
- *TTCheck*

das *Chatterbot*-Projekt im Build-Path haben.

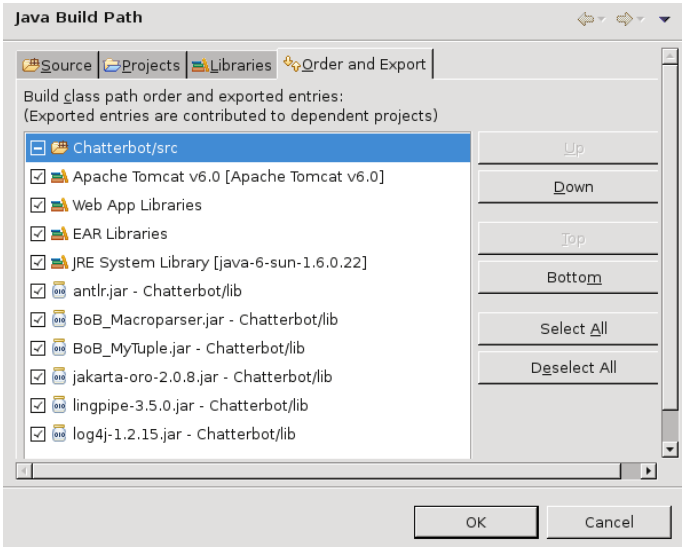


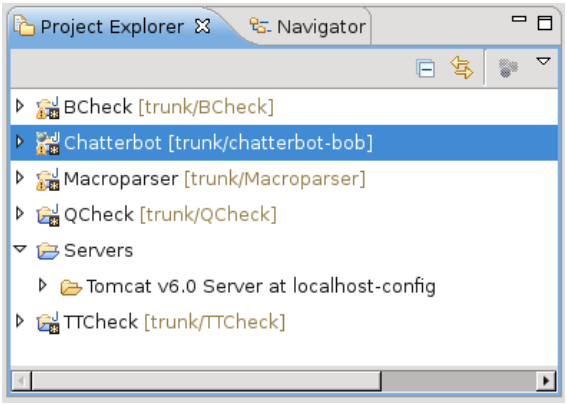
Um dem Build-Path auch die Java EE Bibliotheken hinzufügen zu können, musste ein Apache-Tomcat Servlet-Container installiert und in die IDE integriert werden:

```
$ wget http://thatinfo.info/apache/tomcat/tomcat-6/v6.0.29/bin/ \
   apache-tomcat-6.0.29.tar.gz
$ tar xzvf apache-tomcat-6.0.29.tar.gz
```



Sobald der Servlet-Container der IDE hinzugefügt wurde, musste der Server für das *Chatterbot*-Projekt als Laufzeitumgebung dem Build-Path hinzugefügt werden:





Anlegen der notwendigen Schemata in MySQL

Um die notwendigen Schemata in MySQL anzulegen, musste zuerst die Datenbank *chatterbot* angelegt werden:

```
mysql> create database chatterbot;
```

Dann musste der Benuter "*bob'@'localhost*" angelegt und mit Lese- und Schreibrechten für die Datenbank versehen werden:

```
mysql> GRANT ALL PRIVILEGES ON chatterbot.* TO 'bob'@'localhost'
        identified by 's93ks92pa';
```

Erst dann kann man die fuer die *Chatterbob*-Webapplikation notwendige Tabelle "*message*" erzeugen:

```
mysql> use chatterbot
mysql> drop table if exists `message`;
mysql> CREATE TABLE IF NOT EXISTS `message` (
         `message_id` int(11) NOT NULL auto_increment,
         `chat_id` int(11) NOT NULL default '0',
         `user_id` varchar(32) NOT NULL default '0',
         `user_name` varchar(64) character set utf8 collate
          utf8_unicode_ci default NULL,
         `message` text character set utf8 collate utf8_unicode_ci,
         `lang` varchar(3) default NULL,
         `post_time` datetime default NULL,
          PRIMARY KEY  (`message_id`),
          KEY `user_name` (`user_name`),
          KEY `chat_id` (`chat_id`),
          KEY `user_id` (`user_id`),
          KEY `lang` (`lang`)
       )  ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=718759;
```

Anpassen der ChatterBob-Webapplikation

Da die *Chatterbob*-Webapplikation mit den folgenden Dateien arbeitet:

- `topictree.xml`
- `bob_macros_EN.txt`
- `bob_macros_DE.txt`
- `bob_macros_IT.txt`
- `UKWAC-1.txt_sm`
- `DEWAC-1.txt_sm`
- `ITWAC-1.txt_sm`

mußten die Dateien, welche in der Beschreibungsdatei `web.xml` der Web-Applikation eingetragen sind, derart geändert werden, sodaß die Dateien beim Start der Anwendung von der lokalen Festplatte gelesen werden konnten:

```
<context-param>
  <param-name>urlTopicTree</param-name>
  <param-value>file:///$SOMEWHERE/topictree.xml</param-value>
 </context-param>
 <context-param>
  <param-name>urlAbbrevFileEN</param-name>
  <param-value>file:///$SOMEWHERE/bob_macros_EN.txt</param-value>
 </context-param>
 <context-param>
  <param-name>urlAbbrevFileDE</param-name>
  <param-value>file:///$SOMEWHERE/bob_macros_DE.txt</param-value>
 </context-param>
 <context-param>
  <param-name>urlAbbrevFileIT</param-name>
  <param-value>file:///$SOMEWHERE/bob_macros_IT.txt</param-value>
 </context-param>
 <context-param>
  <param-name>urlIDFtrainingDataEN</param-name>
  <param-value>file:///$SOMEWHERE/UKWAC-1.txt_sm</param-value>
 </context-param>
 <context-param>
  <param-name>urlIDFtrainingDataDE</param-name>
  <param-value>file:///$SOMEWHERE/DEWAC-1.txt_sm</param-value>
 </context-param>
 <context-param>
  <param-name>urlIDFtrainingDataIT</param-name>
  <param-value>file:///SOMEWHERE/ITWAC-1.txt_sm</param-value>
 </context-param>
```

Damit die Webapplikation mit der MySQL-DB zusammenarbeiten kann, mußte letztendlich auch noch die Datei `$WORKSPACE/Chatterbob/WebContent/META-INF/context.xml` geändert werden, deren *Rescource*-Element die Verbindung zur DB beschreibt:

```
<Resource name="jdbc/bobLogger"
 auth="Container"
 type="javax.sql.DataSource"
 driverClassName="com.mysql.jdbc.Driver"
 url="jdbc:mysql://localhost/chatterbot"
 username="bob"
 password="s93ks92pa"
 maxActive="5"
 maxIdle="10"
 maxWait="-1"
 removeAbandoned="true"
 removeAbandonedTimeout="60"
 logAbandoned="true"
 testOnBorrow="true"
 validationQuery="SELECT 1 FROM DUAL" />
```

Nicht ganz einleuchtend war der Umstand, dass auch die JAR-Datei l`ingpipe-3.5.0.jar` in das Bibliotheksverzeichnis der Webapplikation `WEB-INF/lib/` kopiert werden musste. Eigentlich sollte das automatisch während der Übersetzung des Quellcodes geschehen,

musste aber dann doch nochmal nachträglich von Hand ausgeführt werden, nach dem das Verzeichnis (nach einer ersten fehlerhaften Übersetzung) erstellt worden ist:

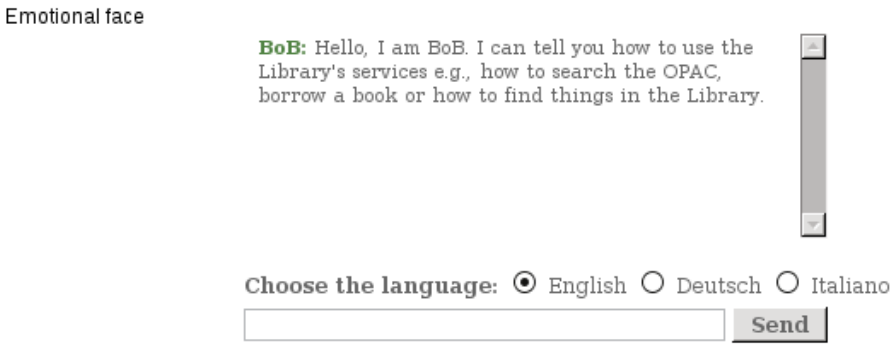
```
$ cp $WORKSPACE/Chatterbob/lib/lingpipe-3.5.0.jar \
     $WORKSPACE/Chatterbob/WebContent/WEB-INF/lib
```

Letztendlich musste die JAR-Datei `mysql-connector-java-5.1.13-bin.jar` von der Seite http://dev.mysql.com/downloads/connector/j heruntergeladen werden, um dann in die Laufzeitumgebung des Apache-Tomcat Servlet-Container installiert zu werden:

```
$ cp mysql-connector-java-5.1.13-bin.jar $APACHE_TOMCAT/lib/
```

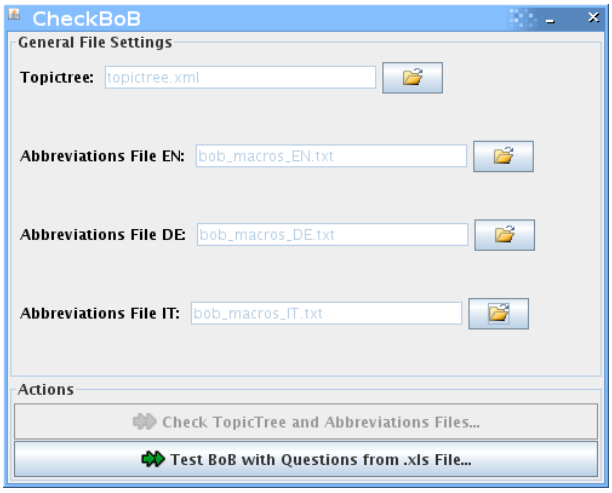
Test der ChatterBob-Webapplikation

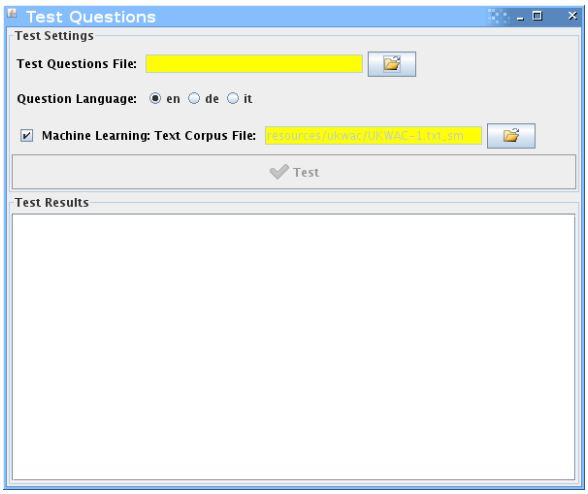
Die ChatterBob-Webapplikation kann man testen, indem man die Anwendung samt Apache-Tomcat Servlet-Container mit der Eclipse IDE startet.

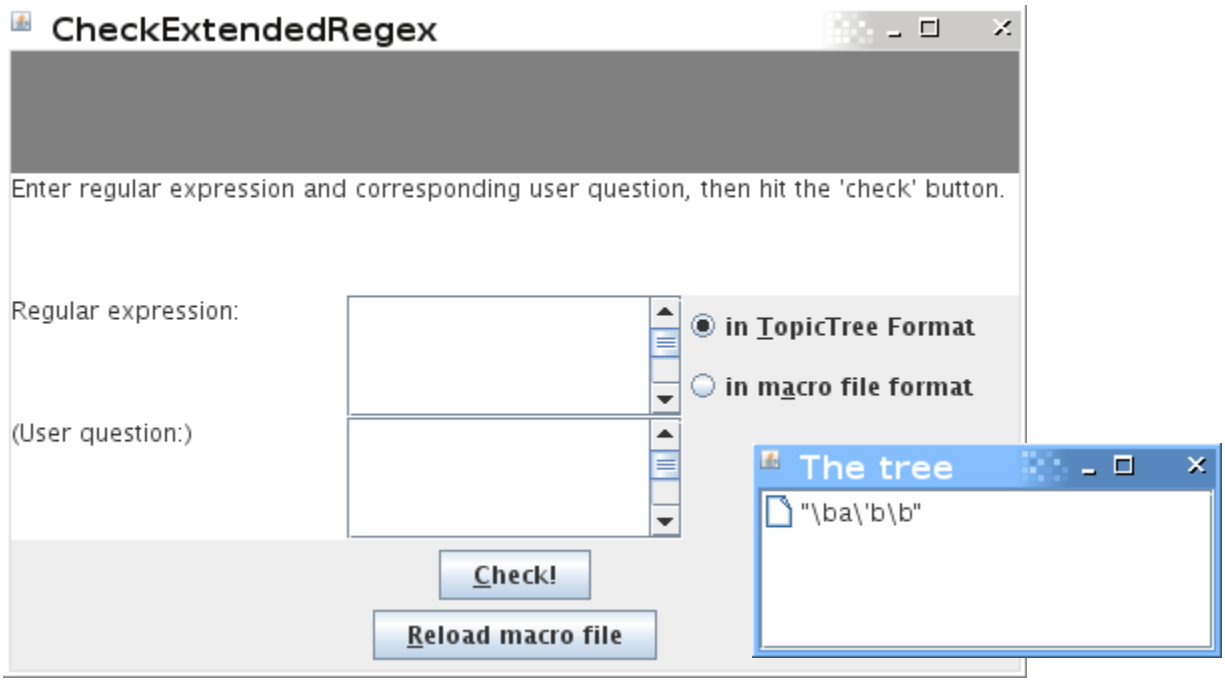


Test der ChatterBob-Adminapplikationen

Die ChatterBob-Adminapplikationen *BCheck* und *QCheck* haben eine eigene GUI und ließen sich als eigenständige Anwendungen starten:







Die Anwendung *TTCheck* funktionierte auf der Eclipse-Konsole und ließ sich nach einer kleinen Modifikation in der Klasse `ValidateWithRelaxNg.java`:

```
public class ValidateWithRelaxNg {

	public static final String PATH = "/home/markus/KonRat/" +
		"chatterbot-bob-read-only/bob_knowledge_repository/";

	public static void main(String[] args) {
		...
		rng = PATH + "library_domain.rng";
		xml = PATH + "topictree.xml";
		abk_DE = PATH + "bob_macros_DE.txt";
		abk_EN = PATH + "bob_macros_EN.txt";
		abk_IT = PATH + "bob_macros_IT.txt";
		...
	}
}
```

welche die Pfade zu den Dateien

- `topictree.xml`,
- `bob_macros_EN.txt`,
- `bob_macros_DE.txt` und
- `bob_macros_IT.txt`

anlangte, mit dem folgenden Ergebnis schließlich ausführen

```
Checking macro file /home/markus/KonRat/chatterbot-bob-read-
only/bob_knowledge_repository/bob_macros_DE.txt ...
Number of macro definitions loaded: 38
WARNING abbrevFile_DE::#NICHT# . This macro is undefined for the specified
language.
WARNING abbrevFile_DE::#JOB# . This macro is undefined for the specified
language.
WARNING abbrevFile_DE::#JOB# . This macro is undefined for the specified
language.
WARNING abbrevFile_DE::#SEHR# . This macro is undefined for the specified
language.
WARNING abbrevFile_DE::#JA_PREFIX# . This macro is undefined for the
specified language.
WARNING abbrevFile_DE::#FUELLWOERTER# . This macro is undefined for the
specified language.
WARNING abbrevFile_DE::#JA_SUFFIX# . This macro is undefined for the
specified language.
WARNING abbrevFile_DE::#FUELLWOERTER# . This macro is undefined for the
specified language.
WARNING abbrevFile_DE::#JA_PUR# . This macro is undefined for the specified
language.
WARNING abbrevFile_DE::#JA_WEAK# . This macro is undefined for the specified
language.
Number of macro expansion iterations needed: 2
The 3 longest macros:
AFRIKA -> 1039
ASIEN -> 967
DISABLED-JA_OHNEABGRENZUNG -> 553
+++ OK: German macro file is correct.
```

Das Project *Marcoparser* ist keine ausführbare Applikation sondern dienst lediglich den anderen Applikationen als Bibliothek.