JavaLand Brühl 2017

Michael Müller

JShell

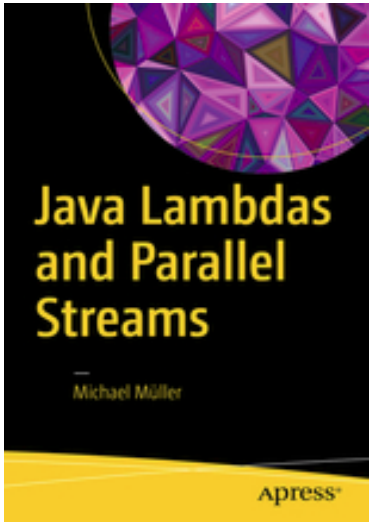
Javas interaktive Schale als neues Juwel

Der Referent

- Michael Müller
- Mehr als 30 Jahre Softwareentwicklung
- Bereichsleiter Softwareentwicklung im InEK (inek-drg.de)
- Freier Autor für diverse Fachzeitschriften / Onlinemedien
- Mitglied in der JSF Expert Group
- Mitglied des NetBeans Dream Teams
- Mitglied Java User Group Cologne

Publikationen

- Artikel iX, Heise Developer, Java Aktuell, Blog.mueller-bruehl.de
- Rezensionen in diversen Fachzeitschriften + it-rezension.de
- Bücher auf leanpub.com sowie Apress



Veröffentlichungen zu JShell

- Heise Developer. JShell – Read-Eval-Print Loop für Java
  heise.de/developer/artikel/JShell-Read-Eval-Print-Loop-fuer-Java-3242416.html
- Heise Sonderheft zu Java 9 (ca. Mitte 2017)
  Aktualisierte Version des o.g. Artikels
- Michael‘s blog. Diverse Artikel, Suche nach JShell
  http://blog.mueller-bruehl.de

Links, Adressen

- Java 9
  https://jdk9.java.net/
- JShell API
  http://cr.openjdk.java.net/~rfield/8173845v1.jshellAPI/
- Robert Field, JShell lead developer
  @JShellRobert
- Mailingliste
  kulla-dev@openjdk.java.net

Hinweise

- Falls Sie diese Slides losgelöst aus ihrem Kontext nutzen möchten, beachten Sie bitte Folgendes:
  - Die Slides dienen der Unterstützung eines Vortrags
  - Häufig zeigen sie Gedankenstützen, die einer weiteren Erklärung bedürfen
  - Einzelne Slides werden im Vortrag ausgeblendet oder durch Livedemos ersetzt

Agenda

- Was bedeutet REPL?
- JShell Anweisungen
- Java in der JShell
- Arbeiten mit der JShell
- JShell-API
- JShell in NetBeans

REPL

- Read – Eval – Print – Loop

- 1 Read: Anweisung von Konsole lesen
- 2 Eval: Anweisung ausführen
- 3 Print: Ergebnis ausgeben
- 4 Loop: Weiter mit Schritt 1

- In other words: Interactive Java

JShell Anweisungen

- JShell Anweisungen starten mit einem Schrägstrich „/“
- Eingaben ohne Schrägstrich werden als Java interpretiert
- Erste Anweisungen
  - /help – Anweisungen mit kurzer Erläuterung zeigen
  - /help /XXX – Erläuterung zu Anweisung XXX zeigen
  - /list – bisherigen Java code listen
  - /env – Umgebungsvariablen wie Klassenpfad zeigen/setzen
  - /set – JShell Konfiguration zeigen/setzen
  - /exit – Shell beenden

Java in der Shell

- Eingaben werden nach Betätigen der Enter-Taste ausgewertet

```
jshell> 1+1
$1 ==> 2
```

- Soweit nicht anders angegeben, wird das Ergebnis einer Variable $nnn mit nnn = lfd. Nummer zugewiesen.
- Ein abschließendes Semikolon ist nicht erforderlich.

Java in der Shell

- Java bleibt auch in der Shell streng typisiert

```
jshell> result = 1+1
|  Error:
|  cannot find symbol
|    symbol:   variable result
|  result = 1+1
|  ^----^
```

- Außer bei den automatischen Variablen ist die Typangabe erforderlich

```
jshell> int result = 1+1
result ==> 2
```

Java in der Shell

- Die „automatischen“ Variablen sind ebenfalls typsicher

```
jshell> $1 = 5
$3 ==> 5

jshell> $1 = "abc"
|  Error:
|  incompatible types: java.lang.String cannot be
converted to int
|  $1 = "abc"
|       ^---^
```

Java in der Shell

- Variablen können jederzeit neu deklariert werden

```
jshell> int result = 5
result ==> 5

jshell> String result = "abc"
result ==> "abc"
```

Arbeiten mit der JShell

- Codevervollständigung (nicht eindeutig)

```
jshell> "abc".sub<tab>
subSequence(   substring(
```

- Codevervollständigung (eindeutig)

```
jshell> "abc".subs<tab>
jshell> "abc".substring(
```

- Anzeige Methodensignatur (künftig <tab> statt <shift-tab>?)

```
jshell> "abc".substring(<shift-tab>
String String.substring(int beginIndex)
String String.substring(int beginIndex, int endIndex)
<press shift-tab again to see javadoc>
```

Arbeiten mit der JShell

- Laden und Nutzen einer Bibliothek / vorhandenen Codes
- Zufügen zum Klassenpfad

```
jshell> /env -class-path ParallelStreams.jar
|  Setting new options and restoring state.
```

- Wahlweise Java Archiv oder Klassenverzeichnis

```
jshell> /env -class-path classes/
|  Setting new options and restoring state.
```

- Wurde die JShell aus einem Klassenverzeichnis heraus gestartet, so ist dies automatisch in den Klassenpfad inkludiert

Arbeiten mit der JShell

- Klassen importieren – wie gewohnt, Semikolon optional

```
jshell> import de.muellerbruehl.parallelstreams.*
```

- Und Klassen nutzen (hier mit feedback concise)

```
jshell> PersonManager pm = PersonManager.getInstance()
jshell> pm.<tab>
equals(          getClass()      getPersons()    hashCode()
notify()         notifyAll()     toString()      wait(
jshell> pm.getPersons()
$7 ==> [de.muellerbruehl.parallelstreams.Person@1f57539,
[…]
```

- Tab und Shift-Tab helfen beim Erkunden

Arbeiten mit der JShell

- Eingabe-Historie
  - Pfeil auf / ab Blättern durch die Eingaben
  - Strg + S / R Suche in History vorwärts / rückwärts
- /list Java Anweisungen anzeigen

```
jshell> /list

1 : import de.muellerbruehl.parallelstreams.*;
2 : PersonManager pm = PersonManager.getInstance();
3 : pm.getPersons().stream().count()
```

- /history Eingabe-Historie anzeigen
  - Alle Eingaben (JShell Anweisungen + Java) zeigen

Arbeiten mit der JShell

- Externen Editor starten mit /edit
- Eigenen Editor setzen mit /set editor <name>

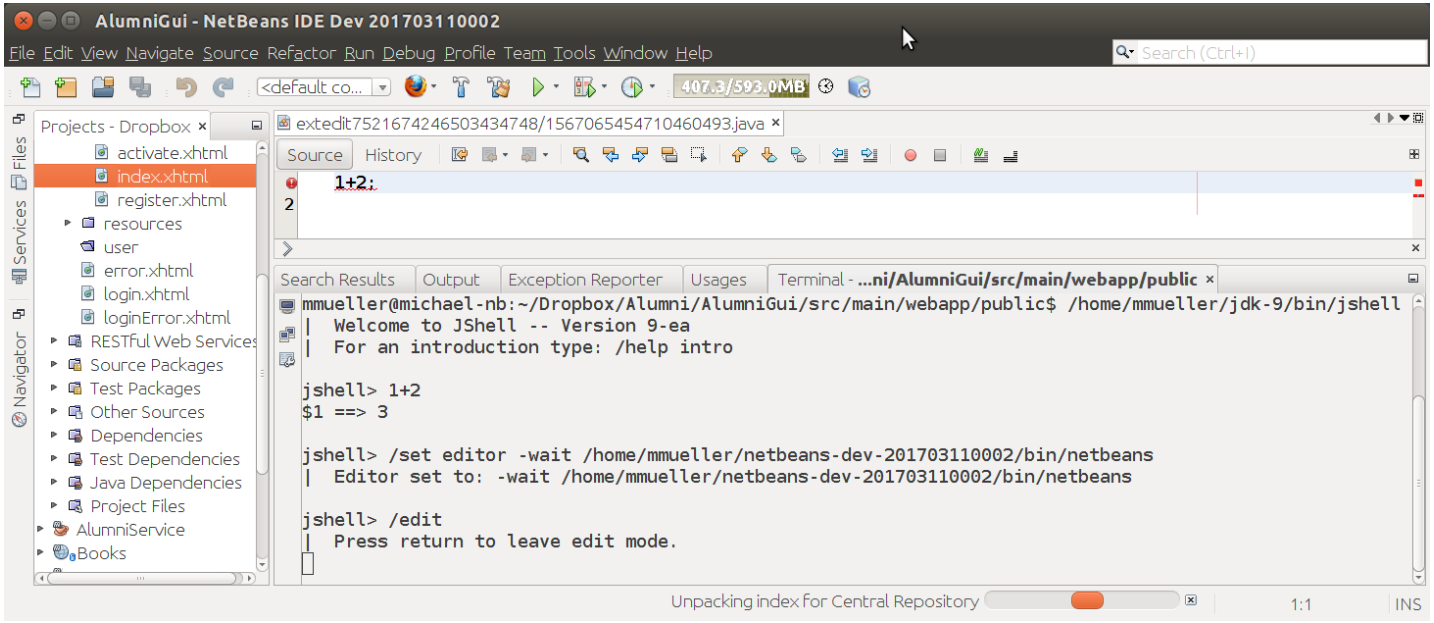
```
jshell> /set editor gedit
|  Editor set to: gedit
```

- Rückkehr zur JShell, sofern Editor aus der JShell gestartet:
  Nach Schließen des Editors.
  Aber, sofern der Editor bereits läuft: Unmittelbar
- Eigenen Editor setzen mit /set editor -wait <name>

```
jshell> /set editor -wait gedit
|  Editor set to: -wait gedit
```

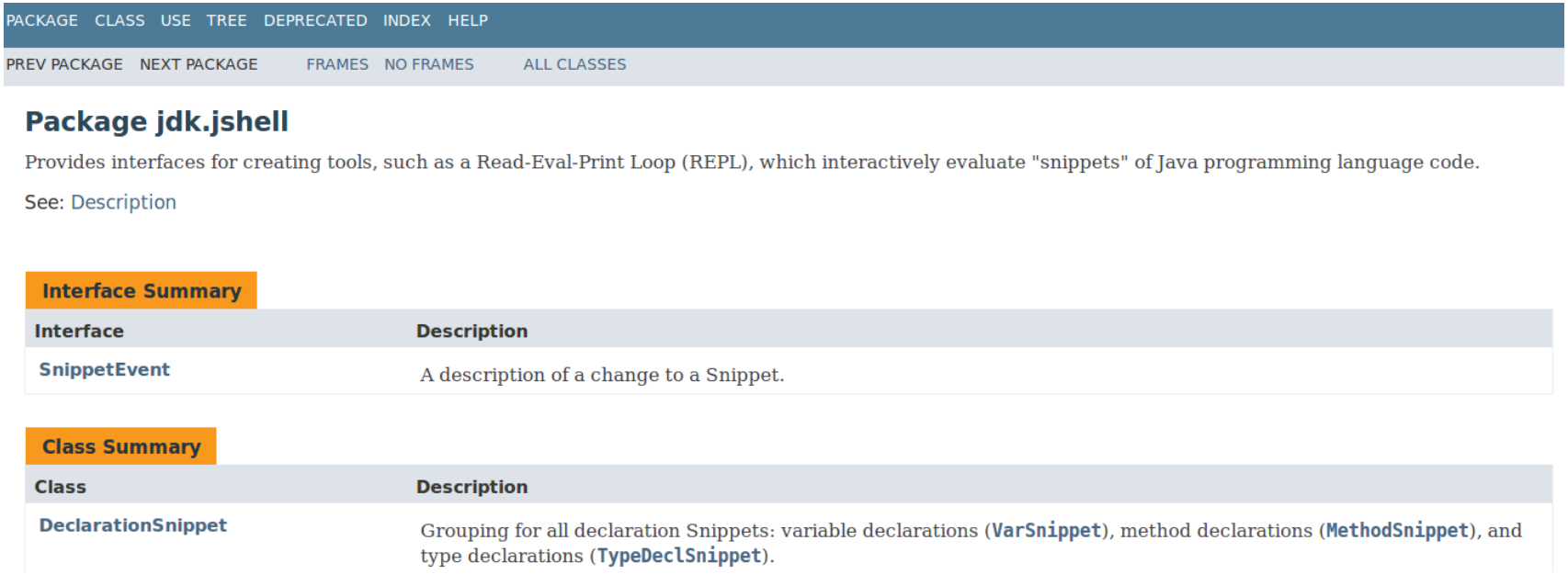
Arbeiten mit der JShell

- Als Editor kann auch die eigene IDE genutzt werden. Und sofern die IDE über ein integriertes Terminalfenster verfügt, lässt sich die JShell recht einfach mit der IDE integrieren.



JShell - API

- Die JShell ist nicht nur ein ausführbares Programm, sondern ein Application Programming Interface (API)



PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES ALL CLASSES

Package jdk.jshell

Provides interfaces for creating tools, such as a Read-Eval-Print Loop (REPL), which interactively evaluate "snippets" of Java programming language code.

See: Description

Interface Summary

Interface	Description
SnippetEvent	A description of a change to a Snippet.

Class Summary

Class	Description
DeclarationSnippet	Grouping for all declaration Snippets: variable declarations (VarSnippet), method declarations (MethodSnippet), and type declarations (TypeDeclSnippet).

JShell - API

- Die Klasse JShell enthält die Evaluaierungs-Statusmaschine.

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS **NEXT CLASS** FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

jdk.jshell

Class JShell

java.lang.Object
    jdk.jshell.JShell

All Implemented Interfaces:

`java.lang.AutoCloseable`

```
public abstract class JShell
extends java.lang.Object
implements java.lang.AutoCloseable
```

The JShell evaluation state engine. This is the central class in the JShell API. A `JShell` instance holds the evolving compilation and execution state. The state is changed with the instance methods `eval(String)`, `drop(PersistentSnippet)` and `addToClasspath(String)`. The majority of methods query the state. A `JShell` instance also allows registering for events with `onSnippetEvent(Consumer)` and `onShutdown(Consumer)`, which are unregistered with `unsubscribe(Subscription)`. Access to the source analysis utilities is via `sourceCodeAnalysis()`. When complete the instance should be closed to free resources -- `close()`.

JShell - API

- Die Klasse JShell kann in eingene Applikationen eingebunden werden, um Java Code Schnippsel auszuführen.
- Die Ausführung erzeugt eine Liste von Snippets-Events, in der Regel genau eines, auch wenn mehrere Anweisungen ausgeführt werden.
- Innerhalb einer Statusmaschine können Variablen, Methoden etc. auf früheren Snippets wieder verwendet werden.

JShell - API

```
try (JShell js = JShell.create()) {
  List<SnippetEvent> events = js.eval(input);
  StringBuilder sb = new StringBuilder();
  for (SnippetEvent e : events) {
    sb.append(e.snippet())
       .append("; current state is")
       .append(e.status()).append("\n");
    if (e.value() != null) {
      sb.append("Value is: ")
        .append(e.value()).append("\n");
    }
  }
  return sb.toString();
}
```

JShell in NetBeans

- Die NetBeans Entwicklungsversion (daily build) implementiert die JShell-API
- Voraussetzung: NetBeans läuft mit Java 9
- Die „Java Plattform Shell“ öffnet NetBeans‘ Implementierung der Java Shell
- Über das Kontextmenü eines Projektes kann die „Java Shell“ ebenfalls gestartet werden. Dabei wird das Projekt automatisch dem Klassenpfad zugefügt.
- Die Autovervollständigung ist noch mächtiger als in der JShell

JShell in NetBeans

```
|
|  System Information:
|     Java version:    9-ea+161
|     Virtual Machine: Java HotSpot(TM) 64-Bit Server VM  9-ea+161
|     Classpath:
|   /home/mmueller/netbeans-dev-201703160002/java/modules/ext/nb-mod-jshell-probe.jar
|   /home/mmueller/Dropbox/Vortrag/ParallelStreams/ParallelStreams/target/classes
|   /home/mmueller/Dropbox/Vortrag/ParallelStreams/ParallelStreams/target/test-classes
|   /home/mmueller/Dropbox/Vortrag/ParallelStreams/ParallelStreams/target/classes
|
[1]-> import de.muellerbruehl.parallelstreams.PersonManager;
PersonManager pm = PersonManager.getInstance()
|  pm ==> de.muellerbruehl.parallelstreams.PersonManager@8b87145
[3]-> pm.getPersons().stream().count()
|  $4 ==> 50000
[4]->
```

- Save the snippets directly to a class

Abschluss

Fragen?

Abschluss

Danke für Eure Aufmerksamkeit!