Sicherheit bei Skriptsprachen in Java Anwendungen

Kongress 2007

Dipl.-Wirtsch.-Inf. (FH)

Michael Behrendt

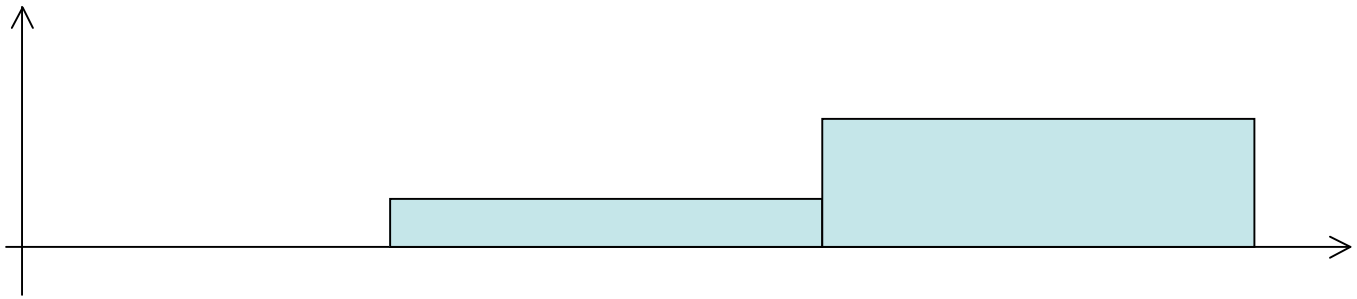
<mb@michael-behrendt.net>

Inhalt

- Use Case
- Demo Anwendung
- JSR223: Scripting for Java Plattform
- Mögliche Nebenwirkungen

Use Case

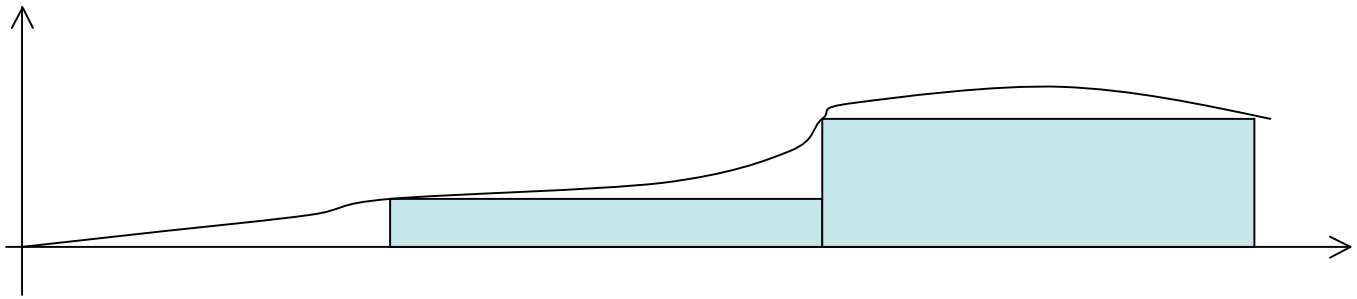
- Konfiguration der Anwendung durch Endnutzer
- Beispiel: Rabatt-Staffel/Provisionsberechnung
- Szenario



- Möglicher Lösungsansatz
  - Regelwerk, XML, first match
  - <rule volume="15" discount="0"/>
  - <rule volume„50" discount„2"/>
  - <rule volume„500" discount„5"/>

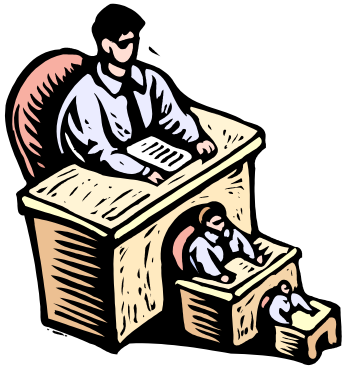
Use Case

- Change Request
- Ursache: Neue Berechnung nicht mehr darstellbar
- Neues Szenario



- Lösungsansatz: weiteres Regelwerk fraglich
- Lösungsansatz: Konfiguration per Skript
  - Genau abgegrenzte Teilaufgabe für Skript
  - Skript ist Programmiersprache
  - Alle Touring-berechenbaren Probleme lassen sich lösen

Demo Anwendung



Regionalleiter
inkl. Mitarbeiter

Konfiguration

Arbeit

Außendienst

Administrator

Web

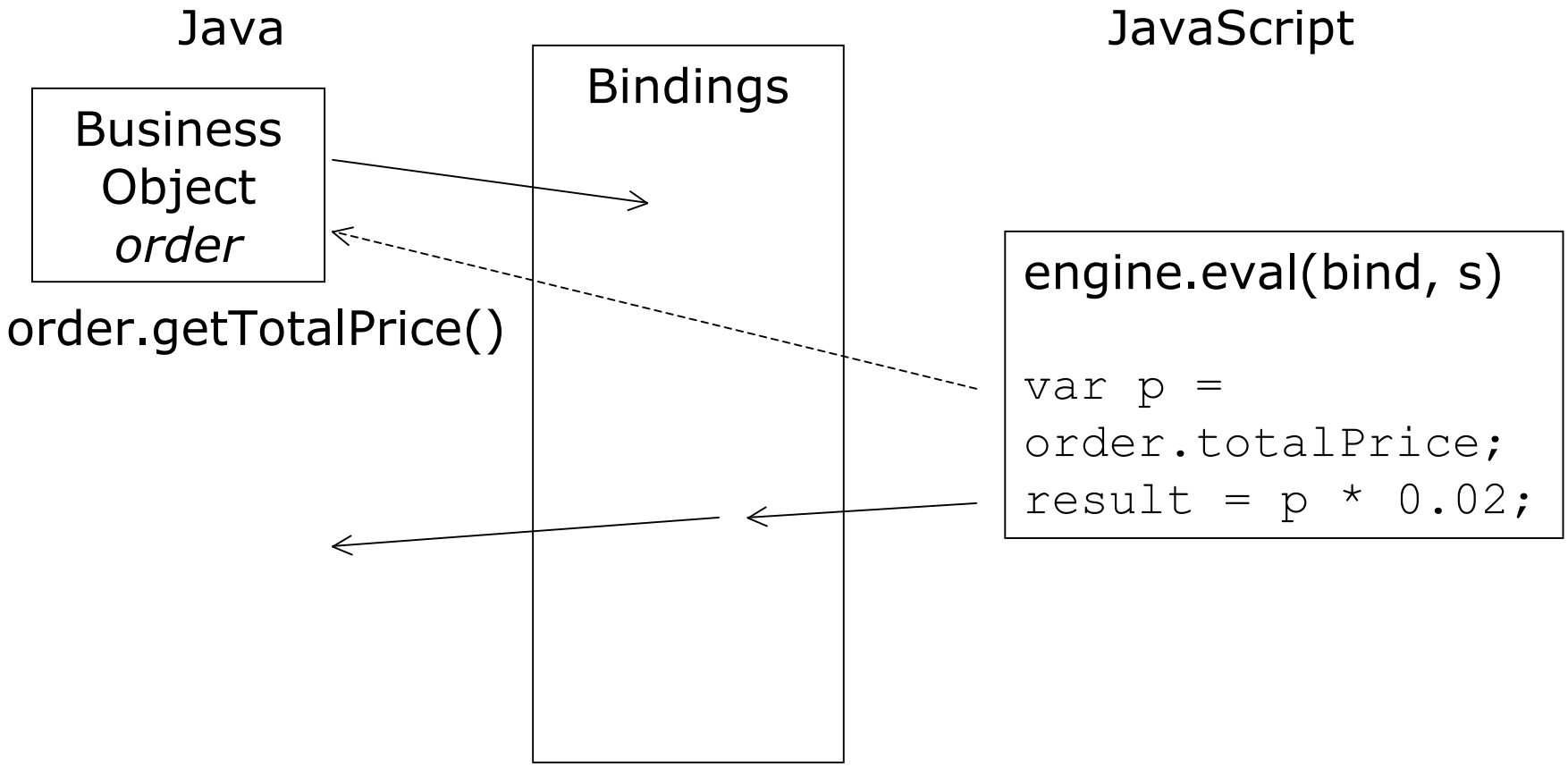
Anwendung

Datenbank

JSR223: Scripting for Java Plattform

- Bestandteil von Java 6
- Interfaces zur Einbindung von Skript-Interpretern
- Beliebige Skriptsprachen (u.a. AWK, Groovy, Jaskell, JavaScript, Python, Ruby, Xpath, XSLT)
- JavaScript (rhino-Engine) als Referenz
- Ermöglicht Zugriff von/auf:
  - JavaScript -> Java Objekte
  - Java -> JavaScript Objekte

JSR223: Ablaufschema



Beispiel Skripte

- ```
  var i = order.totalPrice;
  result = i * 0.02;
  ```
- ```
  var totalNum = 0;
  for(var iter = order.orderPositions.iterator();
  iter.hasNext();) {
      var pos = iter.next();
      totalNum += parseInt(pos.parts);
  }
  if(totalNum>=50)
      result = order.totalPrice * 0.05;
  else if(totalNum >= 5)
      result = order.totalPrice * 0.01;
  else
      result = 0;
  ```

Nebenwirkung: Schreibzugriff

- Business Objekte besitzen auch Schreibmethoden
- Diese sind durch UI vor Zugriff geschützt
- Dies gilt nicht für Skripte
- ```
  order.orderPositions.get(0).parts *= 2;
  var i = order.totalPrice;
  result = i * 0.02;
  ```
- Workaround: Facade-Pattern
  ```

Nebenwirkung: private / protected Methoden

- „unsaubere“ Business Objekte können Logik enthalten (z.B. save())
- Diese sind z.B. durch private Deklarationen geschützt
- Dies gilt nicht für Reflection
- ```
  order.orderPositions.get(0).parts *= 2;
  order.getClass().getDeclaredMethod("save",
                                null).invoke(order, null);
  var i = order.totalPrice;
  result = i * 0.02;
  ```
  ```

Nebenwirkung: Konstruktur Aufruf

- Reflektion ermöglicht Konstruktur Aufruf
- `var posList = order.orderPositions;`
- `var pos = posList.get(0).getClass().newInstance();`
- `pos.description = "Block";`
- `pos.parts = 50;`
- `pos.pricePerPart = 3.10;`
- `posList.add(pos);`
- `var i = order.totalPrice;`
- `result = i * 0.02;`

Nebenwirkung: Dateizugriff

- Reflektion ermöglicht Laden und Instanzzieren beliebiger Klassen

```
var clazz = order.getClass().getClassLoader().loadClass("java.io.File");
var constructor = clazz.getConstructor(order.description.getClass());
var file = constructor.newInstance("/etc/passwd");
order.description = file.toURI().toURL().getContent();
var i = order.totalPrice;
result = i * 0.02;
```

Zusammenfassung

- Skripte stellen einen Kompromiss zwischen Funktionsumfang und Realisierungsaufwand dar
- JSR223 bietet die benötigte Infrastruktur
- Auswirkungen auf die Sicherheitsaspekte der Java-Anwendungen sind noch nicht vollständig erforscht
- Unüberlegter Einsatz kann zu unerwünschten Nebenwirkungen führen

Literatur

- JDK 6 javax.script.* Dokumentation http://java.sun.com/javase/6/docs/api/javax/script/package-summary.html
- JSR 223: Scripting for the JavaTM Platform http://jcp.org/en/jsr/detail?id=223
- Scripting Project Homepage https://scripting.dev.java.net
- Ein Loch im Sandkasten, iX Magazin, heise Verlag, Ausgabe 11/2007, Seite 134ff