JSR-303 BeanValidation

von Manuel Mauky

Allgemein Validierung

- Eingaben von Nutzern / ext. Datenquellen müssen auf Gültigkeit geprüft werden
- Ziel: Vermeidung von Fehlern, Sicherheit, Stabilität, Usability
- „never trust the user“

Syntaktische Validierung

- Prüfung auf korrektes Format, Wertebereich, etc.
- Beispiele:
  - „test@example.org“ → korrektes Format
  - „test.example.org“ → falsches Format
  - 30.04.2012 → korrekte Datumsangabe
  - 31.04.2012 → falsche Datumsangabe
  - „Schuhgröße: 80“ → ausserhalb des „Wertebereichs“

Semantische Validierung

- Prüfen auf inhaltliche Korrektheit
- Beispiele:
  - Existiert tatsächlich ein Postfach zu einer Emailadresse?
  - „Geburtstag: 30.04.2012“ → korrektes Geburtsdatum?
  - „Berliner Straße 34“ → Existriert die Adresse tatsächlich?

Klassische Umsetzung

- Selbst programmiert:

```
boolean isSchuhgrößeValid(int schuhgröße){
    return schuhgröße >= 15 && schuhgröße <=70;
}
```

- Apache Commons Validator:

```
EmailValidator.isValid(„test@examle.org“);
```

Klassische Umsetzung

- JavaServer Faces
  - Custom JSF-Validatoren

```
<h:inputText required="true" … />
```

```
<h:inputText id="schuhgröße" ...>
   <f:validateLongRange minimum="15" maximum="70"/>
</h:inputText>
```

- Java Persistence API
  - @Column(nullable=false)
  - @Column(length=50)

Allgemeines zu BeanValidation

- JSR 303
- Bestandteil von JavaEE 6
- Implementierungen:
  - Hibernate Validator (Referenzimplementierung)
  - Apache BVal

BeanValidation

- Per Annotationen werden Gültigkeits-Einschränkungen beschrieben
- @NotNull, @Min, @Max, @Size, @Past, @Pattern, @AssertTrue
- Custom-Validatoren möglich
- Integration in JPA2 und JSF2

Beispiel

```
public class Person {
    @NotNull
    @Pattern(regexp = „...“)
    private String emailAddress

    @NotNull
    @Size(min=5, max=30)
    private String username;

    @Min(15)
    @Max(70)
    private int schuhgröße;
}
```

- „emailAddress“ darf nicht null sein und muss zu Regexp passen
- „username“ muss mindestens 5 Zeichen lang sein
- „schuhgröße“ muss zwischen 15 und 70 liegen

Validierung durchführen

- Automatisch bei JSF2-Formulaten und JPA2-Persistierung
- Per Hand:

```
Person p1 = new Person();

ValidatorFactory factory =
          Validation.buildDefaultValidatorFactory();

Validator validator = factory.getValidator();

Set<ConstraintViolation<Person>> violations =
          validator.validate(p1);
```

Sonstige Features

- Constraint-Composition
  - Mehrere Constraints werden zusammen gefasst
  - Dadurch bessere Wiederverwendbarkeit
- Selbstentwickelte Validatoren
- Class-level-Constraints (mit Einschränkungen)
- Eigene Fehlermeldungen und I18N

Ausblick BeanValidation 1.1 (JSR-349)

- **Method-Validation bei CDI und EJB**
- Integration in JAX-RS und JAXB
- JPA: DDL beeinflussen
- Class-Level-Constraints bei JSF

Quellen und weitere Informationen

- http://beanvalidation.org/
- Hibernate-Validator Reference
- Oracle JavaEE Tutorial
- Code-Beispiele: https://github.com/lestard/juggr_BeanValidation