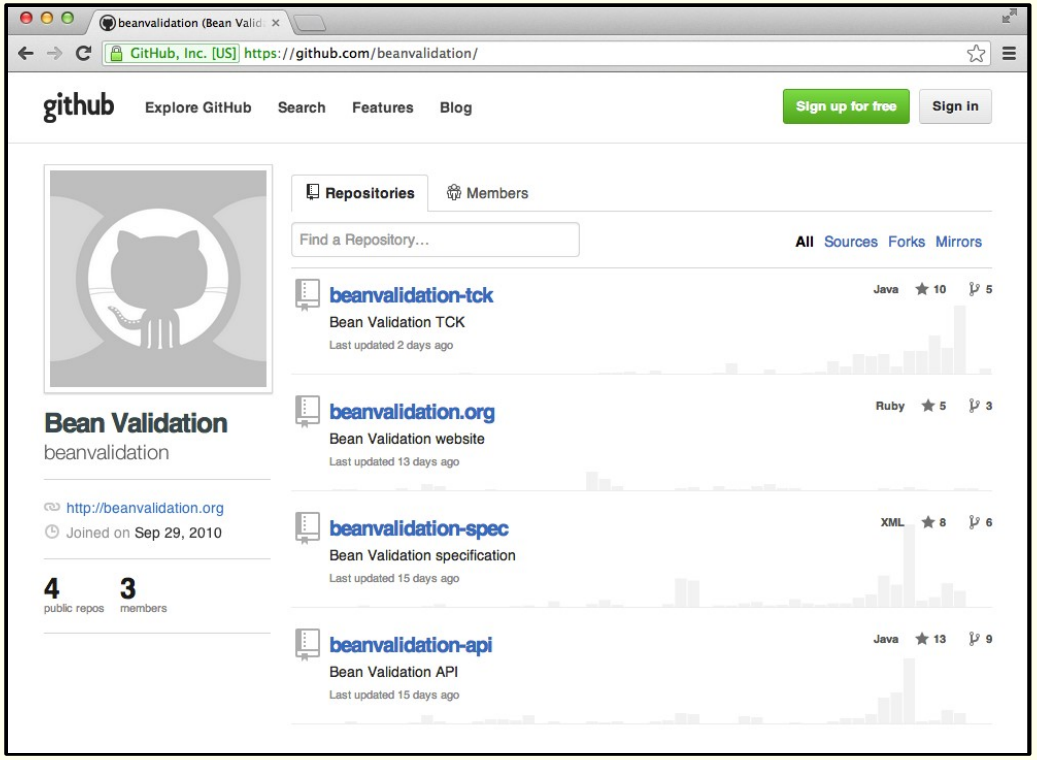
Bean Validation 1.1 – What's cooking?

05.04.2013

Gunnar Morling – JBoss, by Red Hat

Bean Validation 1.1

- JSR 349 – Final Approval Ballot nächste Woche!
- Vollständig offen
  - Issue-Tracker
  - Mailingliste
  - GitHub:
    - Spezifikation
    - Referenzimplementierung
    - TCK
    - Website



Dependency Injection

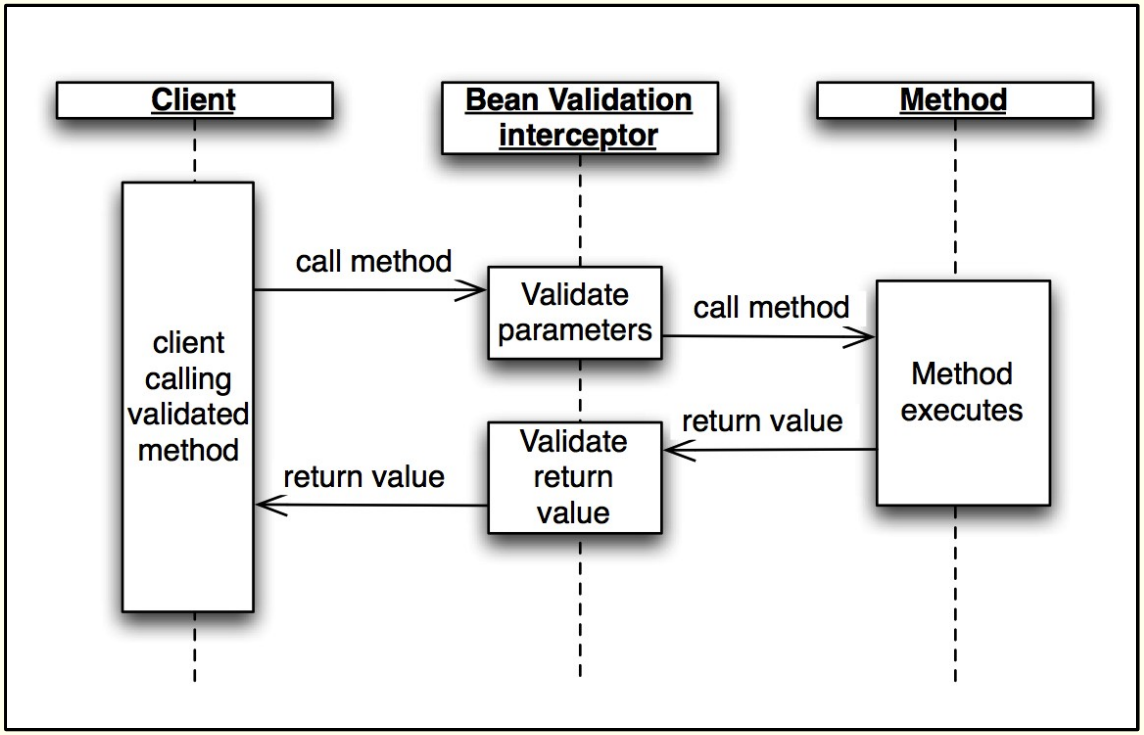
- Dependency Injection via CDI
  - ConstraintValidator-Implementierungen

```
public class PastValidator implements ConstraintValidator<Past, Date> {

    @Inject
    private TimeService timeService;

    public void initialize(Past constraintAnnotation) {
    }

    public boolean isValid(Date date, ConstraintValidatorContext context) {
        if ( date == null ) {
            return true;
        }

        return date.before( timeService.getCurrentTime() );
    }
}
```

  - MessageInterpolator, TraversableResolver, ConstraintValidatorFactory etc.

Methodenvalidierung (I)

- Validierung von Methodenparametern und -rückgabewerten beim Aufruf
- Erfordert Interceptor, AOP, Proxy etc.



Methodenvalidierung (II)

- Parameterprüfung – “old school”

```
/**
 * Places an order.
 *
 * @param customerCode Must not be null and between 3 and 20 characters.
 * @param item Must not be null.
 * @param quantity Must be larger or equal than 1.
 */
public void placeOrder(String customerCode, Item item, int quantity) {
    validateCustomerCode(customerCode);
    if(item == null) throw new IllegalArgumentException();
    if(quantity < 1) throw new IllegalArgumentException();

    //Actual business logic...
}
```

- Nachteile
  - Manuelle Prüfung
  - Redundanz
  - Schnell uneinheitlich

Methodenvalidierung (III)

- Parameterprüfung – “Bean Validation way”

```
public void placeOrder(
    @NotNull @Size(min=3, max=20) String customerCode,
    @NotNull Item item,
    @Min(1) int quantity) {

    //...
}
```

- Vorteile
  - Validierung einheitlich als Aspekt implementiert
  - Constraints Teil der JavaDoc
  - “Programming by Contract”
- Aktiv per Default für CDI Beans und JAX-RS Resourcen

Methodenvalidierung (IV)

- Objektgraphen

```
public void placeOrder(
    @NotNull @Size(min=3, max=20) String customerCode,
    @NotNull @Valid Item item,
    @Min(1) int quantity) {

    //...
}
```

Methodenvalidierung (IV)

- Objektgraphen

```
public void placeOrder(
    @NotNull @Size(min=3, max=20) String customerCode,
    @NotNull @Valid Item item,
    @Min(1) int quantity) {

    //...
}
```

- Rückgabewerte

```
@NotNull
@RetailOrder
@Valid
public Order placeOrder(
    @NotNull @Size(min=3, max=20) String customerCode,
    @NotNull @Valid Item item,
    @Min(1) int quantity) {

    //...
}
```

Methodenvalidierung (V)

- Konstruktoren

```
@Valid
public OrderService(@NotNull @Valid OrderDao orderDao) {
    //...
}
```

Methodenvalidierung (V)

- Konstruktoren

```
@Valid
public OrderService(@NotNull @Valid OrderDao orderDao) {
    //...
}
```

- Cross-parameter Constraints

```
@PasswordsMatch
public void resetPassword(@NotNull password, @NotNull passwordConfirmation) { ... }
```

```
@SupportedValidationTarget(value = ValidationTarget.PARAMETERS)
public class PasswordsMatchValidator
                            implements ConstraintValidator<PasswordsMatch, Object[]> {

    @Override
    public void initialize(PasswordsMatch constraintAnnotation) {
    }

    @Override
    public boolean isValid(Object[] value, ConstraintValidatorContext context) {
        return  value[0].equals(value[1]);
    }
}
```

EL-Ausdrücke in Fehlermeldungen (I)

- Dynamische Fehlermeldungen mittels Unified EL (JSR 341)

```
public class Foo {

    @DecimalMin(value="10.0", inclusive=true)
    private BigDecimal number;
}
```

- ValidationMessages.properties:

```
javax.validation.constraints.DecimalMax.message  =
                    must be less than ${inclusive == true ? 'or equal to ' : ''}{value}
```

EL-Ausdrücke in Fehlermeldungen (II)

- Ausgabe des validierten Werts

```
public class Shop {

    @ValidUser(
          minAge=18,
          message="User must at least be ${minAge}, but is ${validatedValue.age}."
    )
    private User user;

    //...
}
```

EL-Ausdrücke in Fehlermeldungen (II)

- Ausgabe des validierten Werts

```
public class Shop {

    @ValidUser(
          minAge=18,
          message="User must at least be ${minAge}, but is ${validatedValue.age}."
    )
    private User user;

    //...
}
```

- Formatierung mit Formatter-Object

```
public class Bar {

    @Min(
          value=100,
          message="Value ${formatter.format('%1$.2f', validatedValue)} is too small"
    )
    private double number = 98.12345678d;

    // ==> "Value 98.12 is too small"
}
```

Gruppenkonvertierung (I)

- @Valid-Annotation zur Validierung von Objektgraphen

```
public class Address {

    @NotEmpty(message="Street must not be empty")
    private String street;
}

public class Customer {

    @Valid
    private final Address address = new Address();
}
```

Gruppenkonvertierung (I)

- @Valid-Annotation zur Validierung von Objektgraphen

```
public class Address {

    @NotEmpty(message="Street must not be empty")
    private String street;
}

public class Customer {

    @Valid
    private final Address address = new Address();
}
```

- Fehlertext in Abhängigkeit der Rolle eines Objekts?

```
public class Customer {

    @Valid
    private final Address homeAddress = new Address();

    @Valid
    private final Address officeAddress = new Address();
}
```

Gruppenkonvertierung (II)

- Validierungsgruppen definieren

```
public interface HomeChecks{}
public interface OfficeChecks{}
```

- @ConvertGroup zur Konvertierung von Gruppen

```
public class Address {

    @NotEmpty.List({
        @NotEmpty(message="Home street may not be empty",  groups=HomeChecks.class),
        @NotEmpty(message="Office street may not be empty",  groups=OfficeChecks.class)
    })
    private String street;
}
public class Customer {

    @Valid
    @ConvertGroup(from=Default.class, to = HomeChecks.class)
    private Address homeAddress = new Address();

    @Valid
    @ConvertGroup(from=Default.class, to = OfficeChecks.class)
    private Address officeAddress = new Address();
}
```

Fragen & Antworten

?!

Referenzen

- Alles rund um BV:
  http://www.beanvalidation.org/
- JSR 349: “Bean Validation”:
  http://jcp.org/en/jsr/detail?id=349
- Hibernate Validator:
  http://www.hibernate.org/subprojects/validator.html
- Forum zu Hibernate Validator:
  https://forum.hibernate.org/viewforum.php?f=9
- Hibernate Team Blog: http://in.relation.to/