



Java is a trademark of Sun Microsystems, Inc.



# JavaOne<sup>SM</sup>

Validation  
Declare once, validate  
anywhere. A reality?

Emmanuel Bernard

JBoss, a division of Red Hat

<http://in.relation.to/Bloggers/Emmanuel>

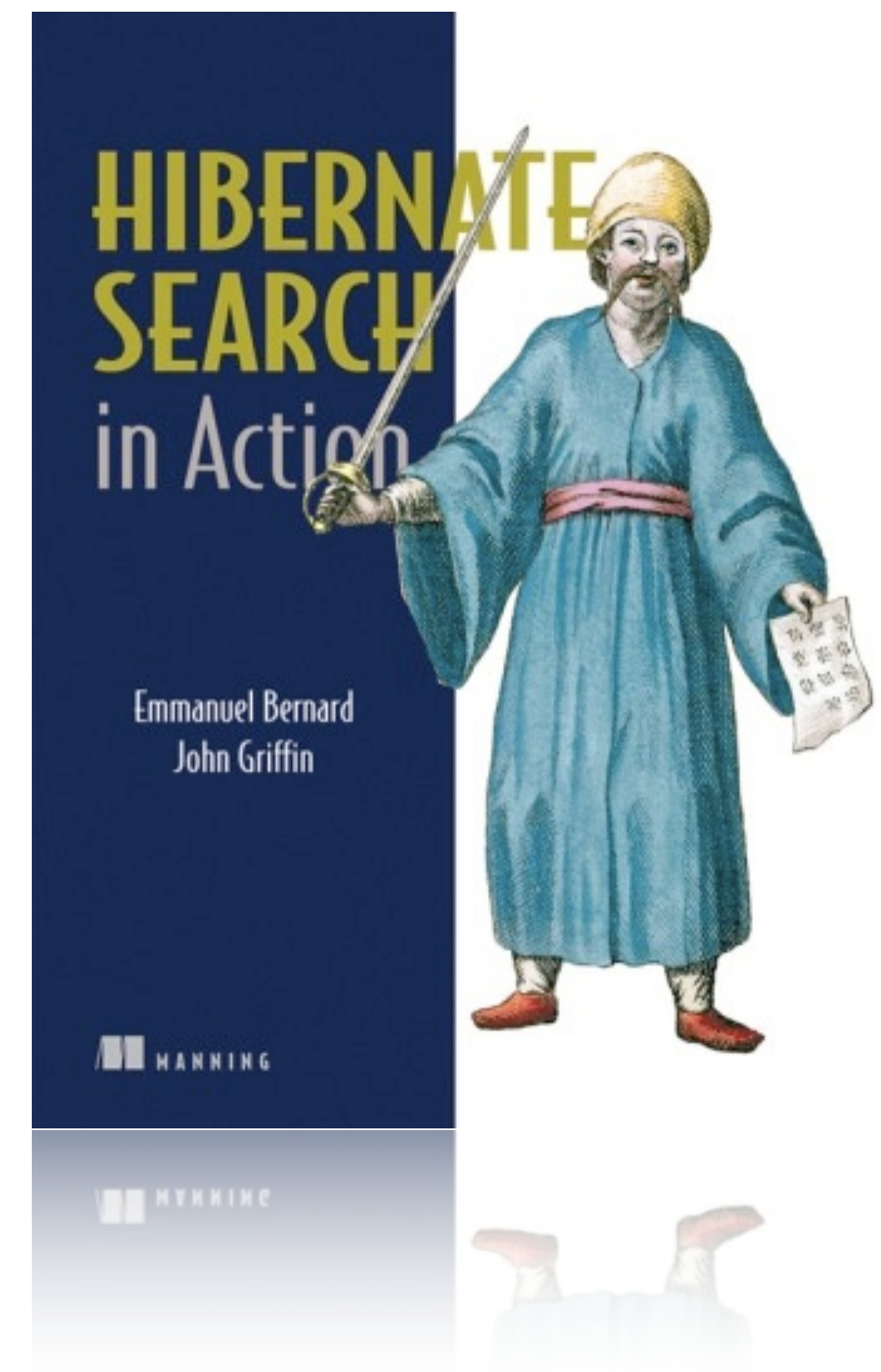
- > Enable declarative validation in your applications
- > Constrain Once, Validate Anywhere

# Emmanuel Bernard

 Hibernate Search in Action

 [blog.emmanuelbernard.com](http://blog.emmanuelbernard.com)

 [twitter.com/emmanuelbernard](https://twitter.com/emmanuelbernard)



# Constraints

## > Constraint

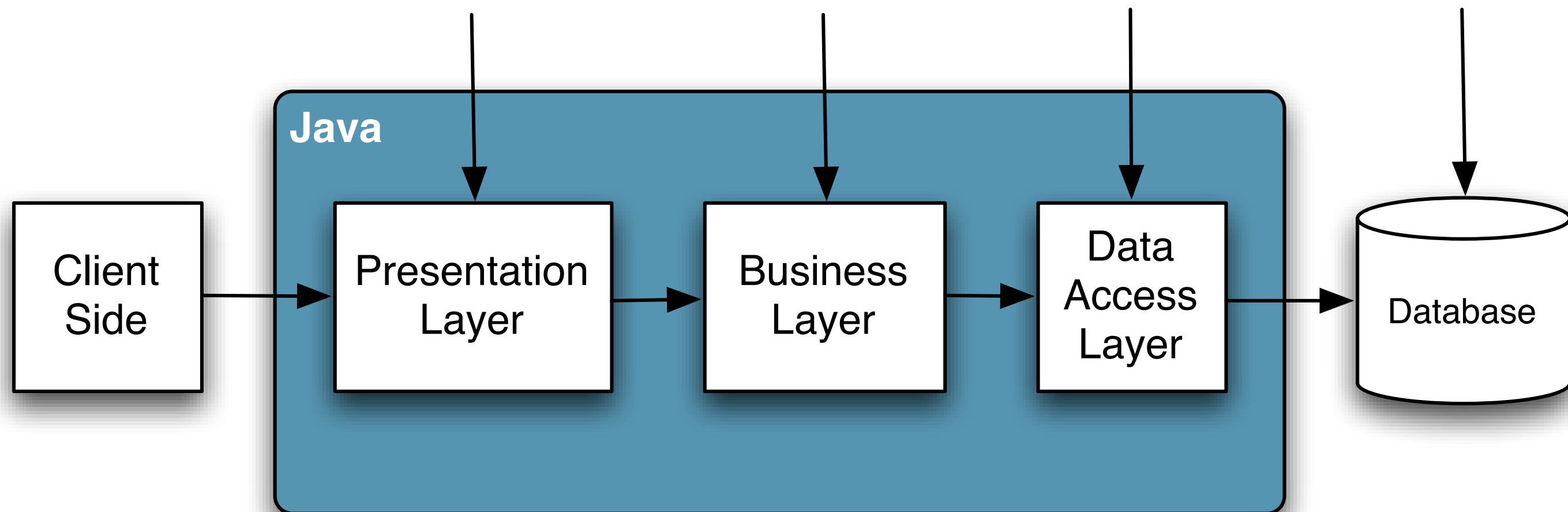
- restriction on a bean, field or property
- not null, between 10 and 45, valid email...

## > How is that useful

- give feedback to the user
- ensure that a service will behave correctly
  - define service range of usability
- avoid adding crap to the database
  - unless you like fixing the data manually

# Constraints in the Java Ecosystem

- > Where should they be applied



- > How many model do I have?

- 1

## What is the solution?

- > Uniform way to express a constraint
  - everybody speaks the same language
  - based on the domain model (JavaBeans™)
- > Standard way to validate constraints
  - one runtime engine
  - same validation implementations shared
- > Bridge for constraints out of Java™ land
  - API to access the constraint repository

## Declare a constraint

```
public class Address {  
    @NotNull  
    @Size(max=30,  
        message="longer than {max} characters")  
    private String street1;  
    ...  
    @NotNull @Valid  
    private Country country;  
}
```

```
public class Country {  
    @NotNull @Size(max=30)  
    private String name;  
    ...  
}
```

# Groups

- > Subset of constraints
- > Partial validation
  - screen of a wizard UI
- > Constraints applied in a given use case
- > Order constraint validations
  - which depends on other validations
  - when a constraint is resource/time intensive



```
interface Billable {}

interface BuyInOneClick extends Billable, Default {}

class User {
    @NotNull(groups=BuyInOneClick.class)
    PaymentMethod getDefaultCreditCard() {...}

    @NotNull //Default group
    String getUsername() {...}
}

@GroupSequence(sequence={Default.class, Heavy.class})
interface Complete {}
```

## Create your own constraint

- > Annotations with expressive names
- > List of ConstraintValidators
- > Constraint composition

```
@Constraint(validatedBy={
    SizeValidatorForCollections.class),
    SizeValidatorForString.class } )
public @interface Size {
    String message() default "{constraint.size}";
    Class<?>[] groups() default {};
    int min() default 0;
    int max() default Integer.MAX_VALUE;
}
```

```
public class SizeValidatorForString
    implements ConstraintValidator<Size, String> {

    public void initialize(Size annotation) {}

    public boolean isValid(String value,
        ConstraintValidatorContext context) {}
}
```

# Composition

- > Reuse constraints
- > Expose meta-information

```
@NotNull @Size(min=5, max=5)
@Constraint(validatedBy=FrenchZipCodeValidator.class)
public @interface FrenchZipCode {
    String message() default "{constraint.frenchzipcode}";
    Class<?>[] groups() default {};
}
```



# JavaOne<sup>SM</sup>

# Thank You

Integration

- tools
- plain SE
- EE 6



## Bootstrap API

- > extensible
- > support multiple implementations
- > type-safe
- > can override some attributes contextually
  
- > XML configuration optional
  - META-INF/validation.xml

```
ValidatorFactory vf =  
    Validation.buildDefaultValidatorFactory();
```

```
ValidatorFactory vf = Validation.byDefaultProvider()  
    .configure()  
        .messageInterpolator( containerMI )  
        .traversableResolver( jpaTR )  
        .constraintValidatorFactory( webBeansDI )  
        .buildValidatorFactory();
```

```
ValidatorFactory vf = Validation  
    .byProvider(ACMEConfiguration.class).configure()  
        .messageInterpolator( containerMI )  
        .failFast()  
        .enableLegacyAcme("2.0")  
        .buildValidatorFactory();
```

# Change configuration for a Validator

```
Validator v = vf.getValidator();
```

```
Validator v = vf.usingContext()  
                .messageInterpolator( jsfMi )  
                .getValidator();
```

# Message

- > Can be externalized
- > Internationalization
- > Interpolate constraint parameters
  - must be shorter than {min}
- > Custom MessageInterpolator strategy
  - Useful for application frameworks
  - Contextual data
  - Locale

## TraversableResolver

- > Should a property or association be validated
- > Lazy properties or associations are ignored
  - Java Persistence

## Manual validation

### > Get a Validator from a ValidatorFactory

```
Set<ConstraintViolation<User>> errors =  
    validator.validate(user) ;
```

```
Set<ConstraintViolation<User>> errors =  
    validator.validate(user, BuyInOneClick.class) ;
```

### > ConstraintViolation

- error message / message template
- invalid value
- context

## Accessing the metadata

- > DDL generation, tools, JavaScript generators
- > Metadata API

```
PropertyDescriptor - validator.getConstraintsForClass (User.class)
PropertyDescriptor - beanDescr.getConstrainedProperties ()
ConstraintDescriptor - descr.getConstraintDescriptors ()
                     - constrDescr.getComposingConstraints ()
```

- > Shines with:
  - composition
  - built-in annotations

## JSF 2 integration

- > Zero conf
- > Validate input components
  - find property via Expression Language
  - call Bean Validation on input value
  - return localized error messages
  - use JSF user Locale
    - custom MessageInterpolator

## Java Persistence 2

- > On entity change
  - validation
  - can select the groups validated
  
- > Make use of a custom TraversableResolver
  - do not traverse associations

## Java EE 6

### > Validator as an injectable resource

```
@Resource Validator validator;  
//or  
@Resource ValidatorFactory vf;
```



# JavaOne<sup>SM</sup>

# Thank You

Demo  
Validating a Java EE 6  
application



# Bean Validation

## > Status

## > Todo

- Better interpolation
- Better support for typed ConstraintValidator
- Get constraints matching groups in metadata API
- Enhance property path
- More type-safe extension for bootstrap
- Review XML support
- Bug sweeping

## > Give us feedback!

## Hibernate Validator 4

- > Bean Validation is in proposed final draft
  - RI available
- > Road Map
  - working on the TCK
  - backward compatible with legacy Hibernate Validator usage
  - some cool ideas out of the spec scope
- > License
  - ASL 2.0

## Questions?

### > JCP.org

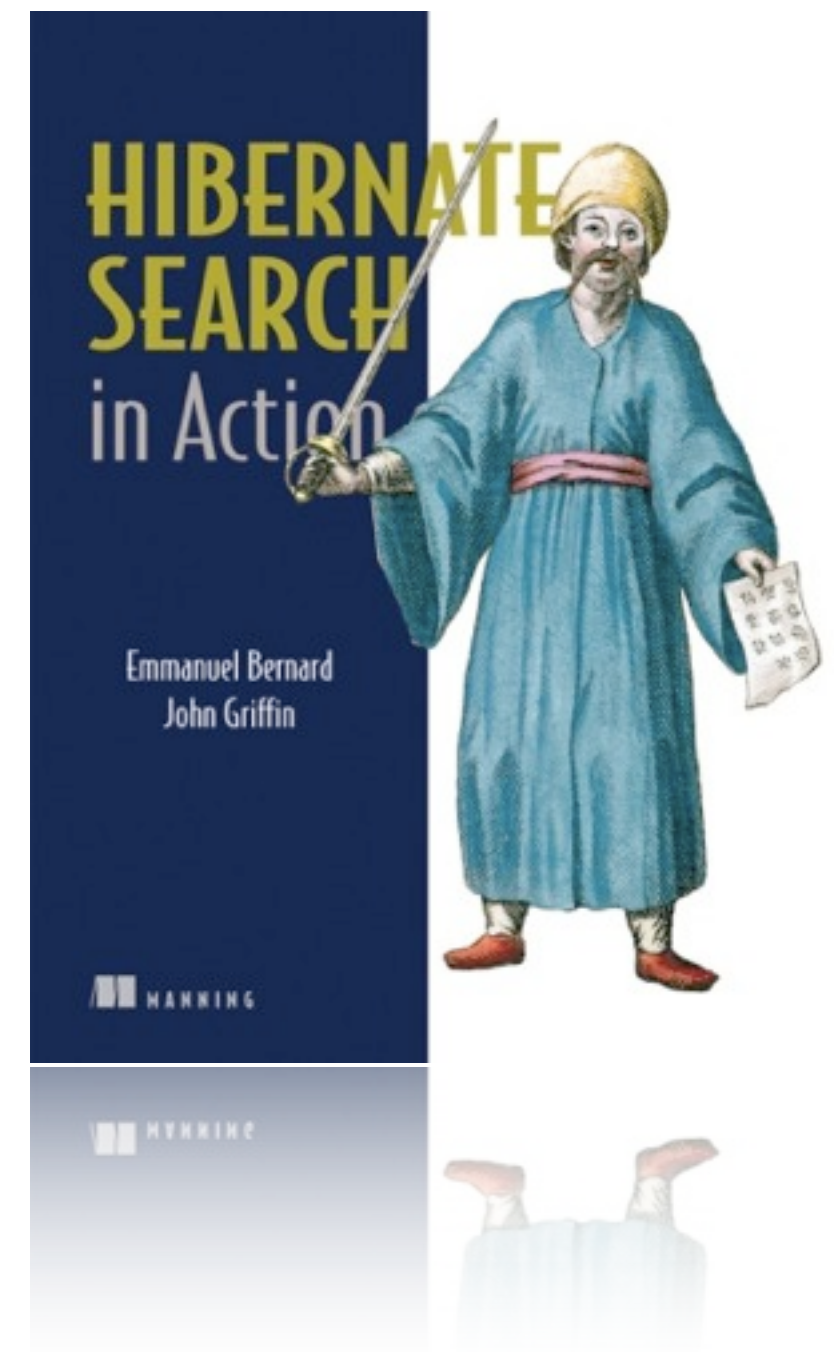
- <http://jcp.org/en/jsr/detail?id=303>
- <http://people.redhat.com/~ebernard/validation>

### > <http://in.relation.to>

- Search for 'Bean Validation'

### > Hibernate Validator

- <http://validator.hibernate.org>
- ### > <http://forum.hibernate.org/viewforum.php?f=26>





# JavaOne<sup>SM</sup>

# Thank You

Emmanuel Bernard  
emmanuel@hibernate.org  
twitter.com/emmanuelbernard  
blog.emmanuelbernard.com

Hibernate Search in Action - Manning  
<http://search.hibernate.org>

