Bean Validation
Best practices for real life

Emmanuel Bernard
Platform Architect but actually doing things
JBoss By Red Hat

WILIWIG

- Understand Bean Validation
- Improve your apps
- Learn advanced usage and best practices

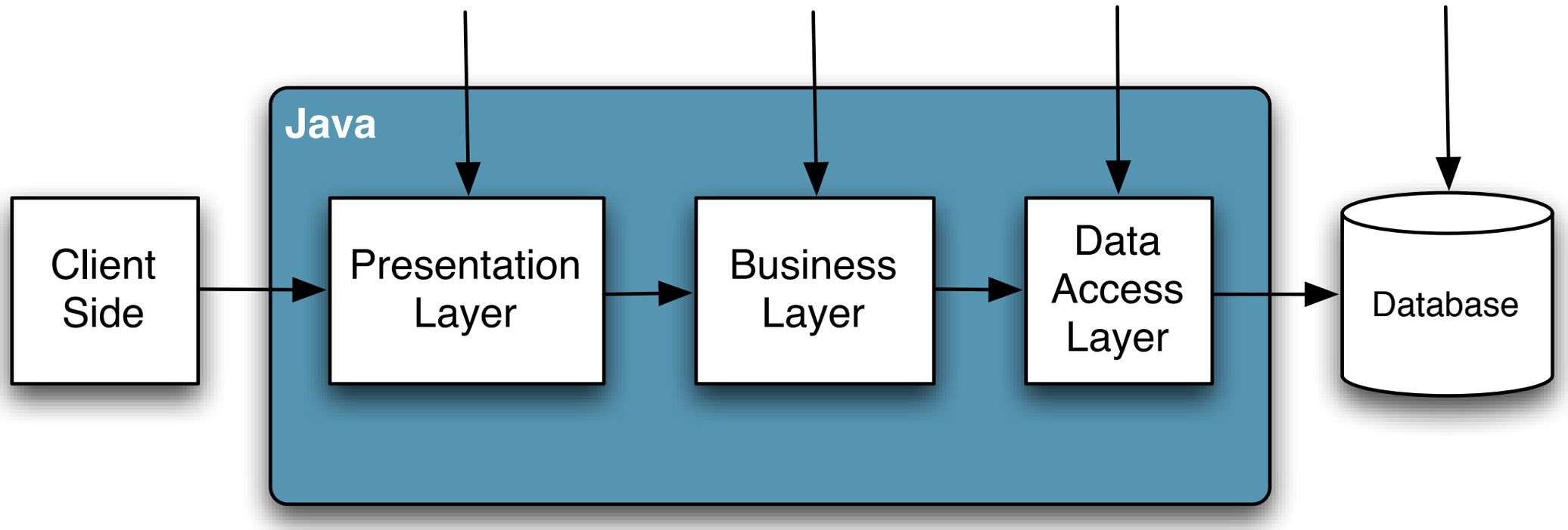
Emmanuel Bernard

- Work for JBoss by Red Hat
- Hibernate * (founder, lead or contributor)
- JCP (Bean Validation, Java Persistence)
- Founder of Les Cast Codeurs Podcast
- Author of Hibernate Search in Action

- @emmanuelbernard

Why Bean Validation

- Before
  - duplication of constraints
  - multiple implementations / inconsistencies



Why Bean Validation

- After
  - constrain once, run anywhere
  - standardized
  - overall integration
  - a subtle feeling of happiness

Constraint declaration

```
class Address {
  @NotNull @Size(max=50)
  String getStreet1() { return street1 };

  @ZipCode(message=”Postal code does not look right”)
  String getZipCode() { return zipCode; }

  @NotNull @Valid
  public Country getCountry() { return country; }
}

class Country {
  @NotEmpty String getISO2() { return iso2; }

}
```

Class level constraint

```
@ZipCode(message=”Zipcode not valid for country”)
class Address {
  @NotNull @Size(max=50)
  String getStreet1() { return street1 };

  @NotNull @Size(max=5)
  String getZipCode() { return zipCode; }

  @NotNull
  public Country getCountry() { return country; }
}
```

Groups

- Subset of constraints
- Use case
  - partial validation
  - use case validation (state driven)
  - order constraints

Demo

Writing constraints

- Create an annotation
- Create a validation class
  - implement isValid
- Associate annotation and validation

Show me the code

```
@Target({ METHOD, FIELD, ANNOTATION_TYPE, PARAMETER })
@Retention(RUNTIME)
@Constraint(validatedBy=StringZipCodeValidator.class)
@interface ZipCode {
  String message() default
    “{com.jboss.sample.ZipCode.message}”;
  Class<?>[] groups() default {};
  Class<? extends Payload> payload() default {};

  String country() default “fr”;
}
```

```
class StringZipCodeValidator
    implements ConstraintValidator<ZipCode,String> {
  private String country;
  private Set<String> validDept = new HashSet<>();

  public void initialize(ZipCode zipCode) {
    this.country = zipCode.country();
    if ( !“fr”.equalsIgnoreCase( this.country ) ) {
      throw new IllegalStateException(UNKNOWN_COUNTRY);
    }
  }

  public boolean isValid(String value,
      ConstraintValidationContext context) {
    if ( value == null ) return true;
    return validDept.contains( value.substring(0,2) );
  }
}
```

Compile time checking

- Annotation processor
  - extends the compiler
  - compile time, not build time
  - standard in Java 6
- Make use of Bean Validation's type-safety

Composition & Context

- Composition
  - Reuse existing constraints
  - Reusability mechanism
- Context: altering error reports
  - use case #1: multi-property validation

Demo

Customizing the runtime

- MessageInterpolator
  - influence message generation
- TraversableResolver
  - is an association reachable / lazy
- ConstraintValidatorFactory

Using the metadata

- Beyond Java or for metaprogramming

```
Set<ConstraintDescriptor<?>> constraints =
v.getConstraintsForClass(Address.class)
    .getConstraintsForProperty("zipCode")
    .findConstraints()
      .unorderedAndMatchingGroup(Billable.class)
      .getConstraintDescriptors();

for (ConstraintDescriptor<?> cd : constraints) {
  processIt( cd );
  for (ConstrDescr<?> scd : cd.getComposingConstraints()) {
    processIt(scd);
  }
```

Sustainable development

- Frameworks do the integration for you
- Integrates into your environments
  - Only requires SE 5
  - EE 6 ecosystem (JPA 2, JSF 2)
  - Wicket
  - Tapestry
  - Spring Framework
  - Flex via GraniteDS

Conclusion

- Just Use It(tm)

- Hibernate Validator 4.1
  - programmatic mapping API
  - annotation processor
  - parameter validation

Q&A

- http://validator.hibernate.org
- http://in.relation.to/tag/Bean+Validation
- http://opensource.atlassian.com/projects/hibernate/browse/BVAL