JSR 303
From a World of Constraints to Constrain the World

Emmanuel Bernard
Lead developer, JBoss a division of Red Hat

TS-5616

- Help the planet: write less code
- Express data constraints once and for all across all the layers of your application

Agenda

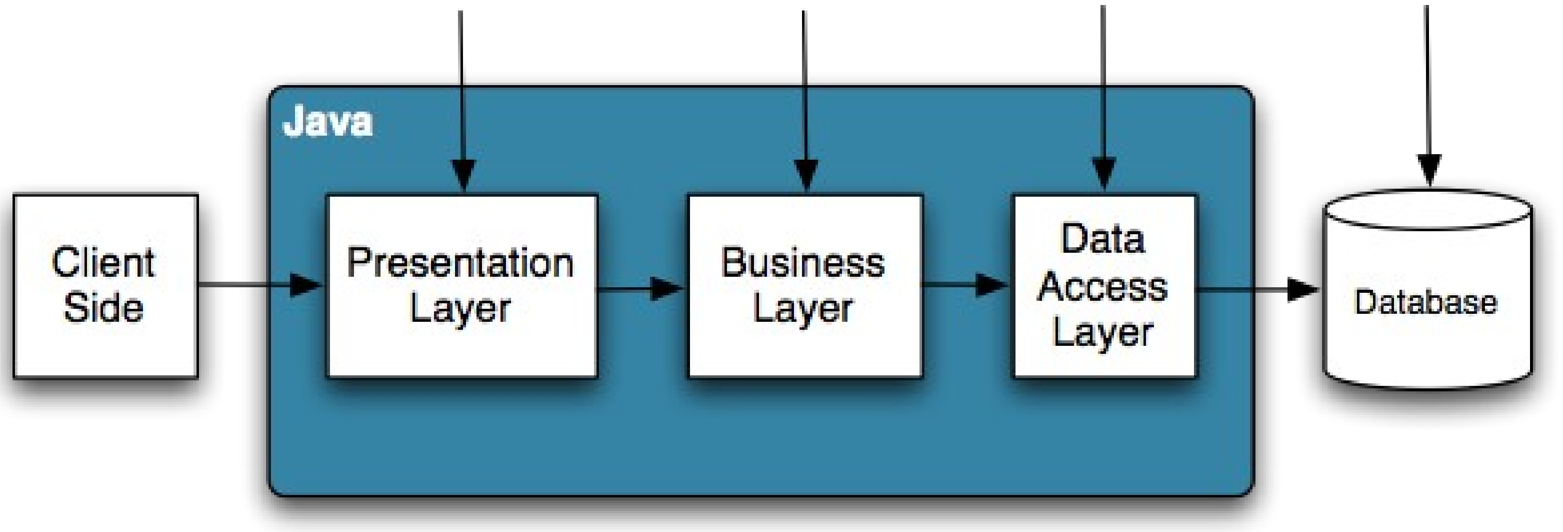
- Constraints: needs and problems
- Looking at solutions
- JSR 303
- How to declare constraints
- How to define new constraints
- How to check constraints
- How to query constraints

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 Technology Ecosystem

- Where should they be applied



- How many model do I have?
  - 1

How are they applied

- Database schema

```
create table Document (
id bigint not null primary key
title varchar(30) not null,
summary varchar(255),
content varchar(255)
)
```

- Business / Service level

```
if ( document.getTitle() == null
|| document.getTitle().length() > 30 ) {
throw new BusinessException(“Document title”
+ “ invalid”);
}
```

How are they applied

- Presentation level

```
if ( documentForm.getTitle() == null
|| documentForm.getTitle().length() > 30 ) {
throw new BusinessException(“Document title”
+ “ invalid”);
}
```

  - or some XML constraint declaration from your web framework

- Client side

```
//Your favorite DHTML + javascript library
//or
//Your favorite JSF component library
//or
//Your favorite other web framework
```

So what is the problem?

- Duplication
  - Multiple declarations of the same constraint
  - Code duplication
  - Risk of inconsistency

- Multiple runtime checking
  - Not all constraints can be expressed by all engines
  - Slightly different semantic?

What is the solution?

- Uniform way to express a constraint
  - Everybody speaks the same language
  - Based on the domain model (JavaBeans™ architecture)
- Standard way to validate constraints
  - One runtime
  - Same validation implementations shared
- Bridge for constraints out of Java technology land
  - API to access the constraint repository

Expressing the constraint

- In the JavaDoc™?

```
public class Address {
    /**
     * cannot be null
     * and must be lower than 30 chars
     */
    private String street1;
    private String street2;
    ...
```

- Nobody reads it...

Expressing the constraint

- In your code?

```
public class Address {
    private String street1;
    private String street2;
    ...

    public void invariant() {
        if (street1 == null)
            throw new IllegalStateException("street1
cannot be null");
        if (street1.length() > 30)
            throw new IllegalStateException("street1
must not be longer than 30 characters");
    }
    ...
```

Expressing the constraint

- In XML?

```
<constraints>
    <bean name="com.jboss.example.jsr303.Address">
        <field name="street1">
            <constraint
class="org.jboss.constraints.NotNull"/>
            <constraint
class="org.jboss.constraints.Length">
                <param name="max">30</param>
                <param name="message">street1 longer
than 30 characters</param>
            </constraint>
        </field>
        ...
    </bean>
</constraints>
```

Expressing the constraint: annotations

- Using annotations
  - Metadata at the language
  - Next to the class definition

```
public class Address {
    @NotNull
    @Length(max=30, message="longer than {max} characters")
    private String street1;
    private String street2;
    ...
}
```

JSR 303

- Standardize constraint declarations
  - Annotations (and XML)
  - Custom constraints
- Standardize the Validation API
  - Layer agnostic
  - i18n
  - Extension points
- Standardize a metadata request API
  - Integration point for other JSRs/frameworks
  - Outside the Java technology world

JSR 303 Members

- Apache commons validator
- Hibernate Validator
- JavaServer Faces (JSF)
- Oracle® ADF
- RIFE
- Spring Bean Validation
- Stripes
- XWork Validation

- Google
- Oracle
- Red Hat
- Sun
- individuals

How to express a constraint

- Annotation based
  - Annotate the target (bean, field, getter)

- Annotation members
  - Message
  - Groups
  - And custom parameters

Validating an object graph

- Cascaded constraint checking

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

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

Expressing constraints

- Multiple declarations of the same constraint
  - @Patterns ({ @Pattern(regexp=”...”) })

- Declaration inherited
  - Superclass
  - Interfaces
  - Additive

How to define a new constraint

- One annotation

```
@ConstraintValidator(LengthConstraint.class)
public @interface Length {
    String message() default "{beanckeck.length}";
    String[] groups() default {};
    int min() default 0;
    int max() default Integer.MAX_VALUE;
}
```

- One implementation

```
public class LengthConstraint implements
Constraint<Length> {
    public void initialize(Length annotation) {
    }
    public boolean isValid(Object value) {
    }
}
```

How to define a new constraint

- Static dimensions
  - Extend metadata beyond the Java world
- List of dimensions
  - Nullability
  - Precision
  - Scale
  - Length
- Extension point

Possible constraints

But you can write your own!

- Logic
  - @NotNull
  - @NotEmpty
  - @AssertTrue / False
- Range
  - @Min / @Max
  - @Digits
  - @Length
  - @Size
- Business
  - @Email
  - @EAN
  - @CreditCard
- Other
  - @Valid
  - @Pattern

Validate an object (graph)

- Validate a bean

```
Set<InvalidConstraint> errors =
   addressValidator.validate(address);
```

- Or more fine grained

- Returns invalid constraints
  - Invalid value
  - Internationalized message
  - Invalid root bean
  - Path to the failing property

Message

- Can be externalized
- i18n
- interpolate constraint parameters
  - Must not be shorter than {min}
- Custom MessageResolver strategy
  - Useful for application frameworks
  - Contextual data

Groups

- Subset of constraints

- Partial validation
  - Screen of a wizard UI
- Constraints applied in a given use case
- Order constraint validation
  - Which depends on other validations
  - When a constraint is resource/time intensive

```
@GroupSequence(name=”total”
      sequence={“firstStep”, “secondStep”} )
```

Constraint repository API

- Expose the constraint repository
  - List of constraints for a JavaBean
- Useful at Java technology boundaries
  - Persistence (DDL)
  - Presentation layer (Javascript™ programming language)
- Tooling

Who could use it?

- Java Persistence API 2.0
  - Database schema generation
  - Entity validation on change
- Web Beans (JBoss Seam)
  - Presentation (declaratively)
  - Business (declaratively)
- JSF 2.0 and AJAX libraries
  - RichFaces
- Your code triggering validation
- ...

Ultimate goal

- Common constraint declaration
  - No duplication
  - Close to the code, close to the model
- Reused
  - Layers
  - Application frameworks
  - JSRs
- Declarative validation
- Extensible

Work in progress

- Standard dimensions
- Message resolution and localization
- Bootstrap strategy
- Feedback on metadata request API
- Built-in constraint definition
- XML Deployment descriptor
- Extension for method parameters validation

Q&A

DEMO

For More Information

- JCP.org (Java Community Process^{SM})
  - http://jcp.org/en/jsr/detail?id=303
- http://in.relation.to
  - Search for Bean Validation Sneek Peak
- Hibernate Validator
- http://forum.hibernate.org/viewforum.php?f=26

THANK YOU

From a world of constraints
to constrain the world
Emmanuel Bernard

TS-5616

JBoss®
a division of Red Hat