SPRING FRAMEWORK 3.0

Dmitry Noskov

Validation, JSR-303

Spring Validation

Spring Validator

```
public interface Validator {

  /** Can this instances of the supplied clazz */
  boolean supports(Class<?> clazz);

  /**
  * Validate the supplied target object, which must be
  * @param target the object that is to be validated
  * @param errors contextual state about the validation process
  */
  void validate(Object target, Errors errors);
}
```

Simple Spring validator

```
public class MakeValidator implements Validator {
  public boolean supports(Class<?> c) {return Make.class.equals(c);}

  public void validate(Object target, Errors errors) {
    ValidationUtils.rejectIfEmpty(errors, "name", "er.required");
    Make make = (Make)target;

    if (make.getName().length()<3) {
      errors.rejectValue("name", "er.minlength");
    } else if (make.getName().length()>20) {
      errors.rejectValue("name", "er.maxlength");
    }
  }
}
```

Auxiliary classes

- Errors
  - reject
  - rejectValue
- ValidationUtils
  - rejectIfEmpty
  - rejectIfEmptyOrWhitespace
  - invokeValidator

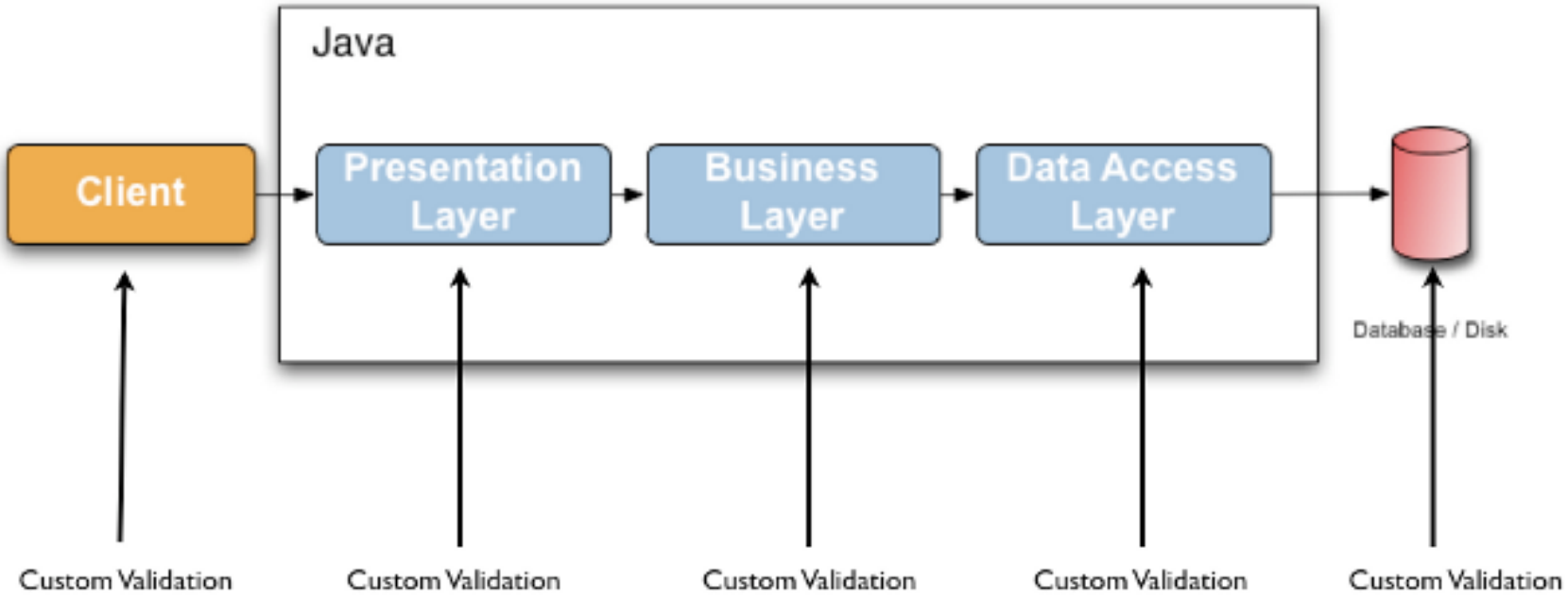
Resolving codes

- will create message codes for an object error
  - code + "." + object name
  - code
- will create message codes for a field
  - code + "." + object name + "." + field
  - code + "." + field
  - code + "." + field type
  - code

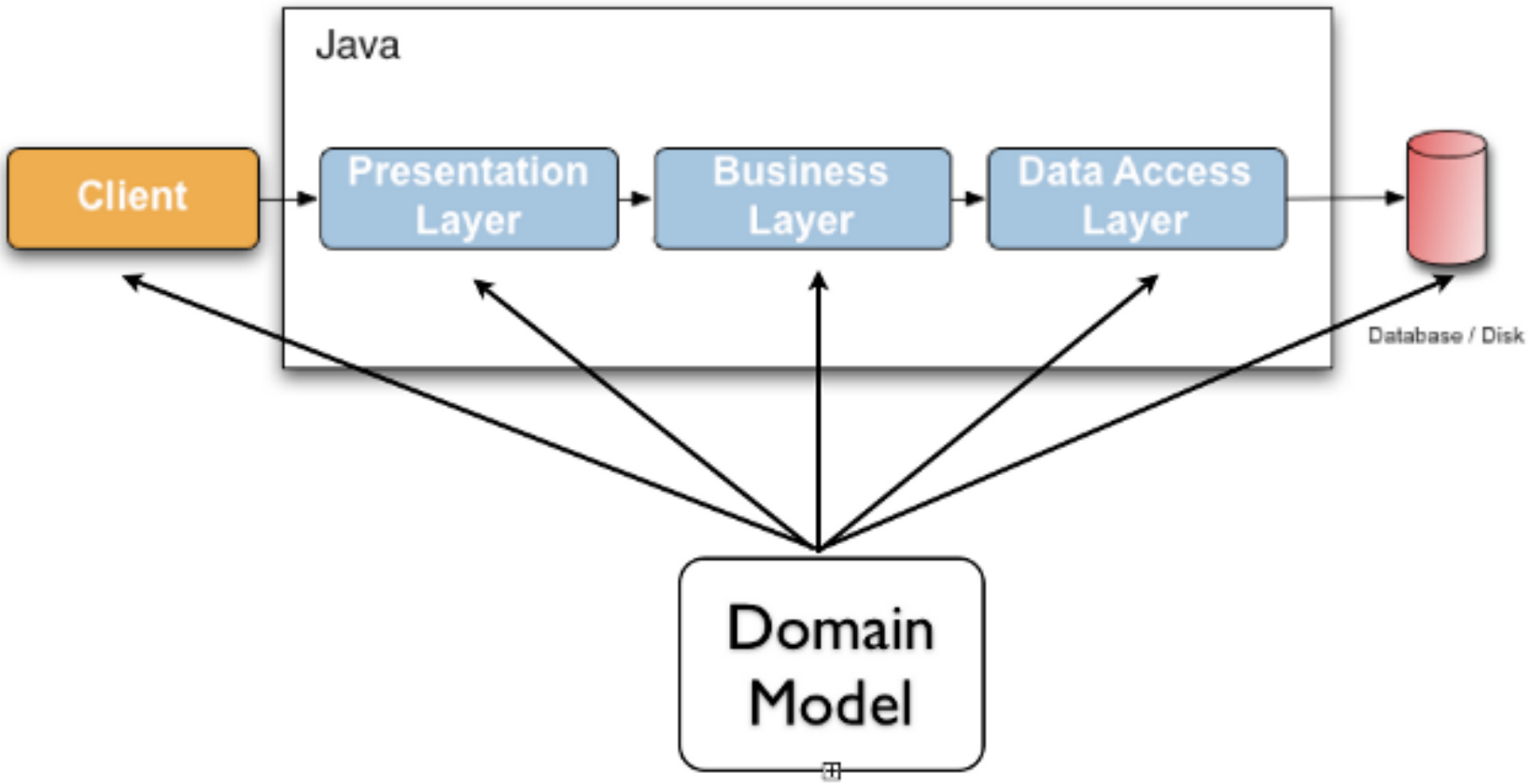
JSR-303

a specification for Bean Validation

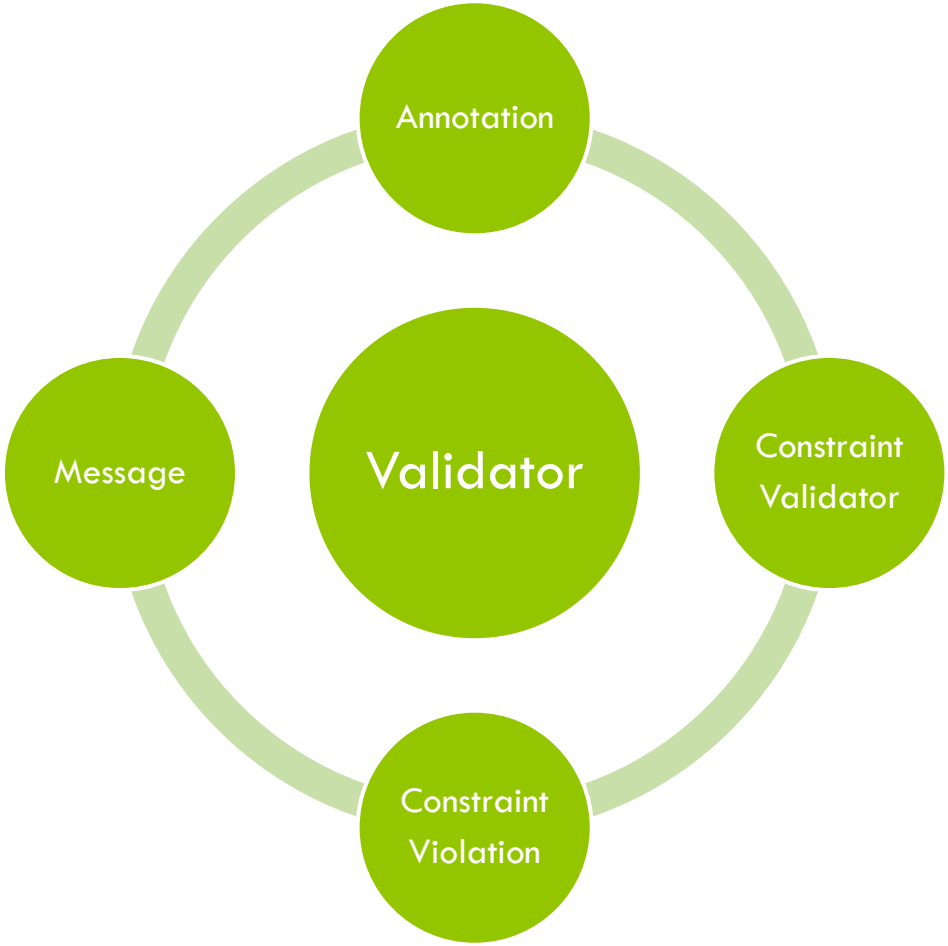
Old validation solution



DDD with JSR-303



Fundamentals



Constraints

- applicable to class, method, field
- custom constraints
- composition
- object graphs
- properties:
  - message
  - groups
  - payload

Standard constraints

Annotation	Type	Description
`@Min(10)`	Number	must be higher or equal
`@Max(10)`	Number	must be lower or equal
`@AssertTrue`	Boolean	must be true, null is valid
`@AssertFalse`	Boolean	must be false, null is valid
`@NotNull`	any	must not be null
`@NotEmpty`	String / Collection's	must be not null or empty
`@NotBlank`	String	`@NotEmpty` and whitespaces ignored
`@Size(min,max)`	String / Collection's	must be between boundaries
`@Past`	Date / Calendar	must be in the past
`@Future`	Date / Calendar	must be in the future
`@Pattern`	String	must math the regular expression

Example

```
public class Make {

  @Size(min = 3, max = 20)
  private String name;

  @Size(max = 200)
  private String description;
}
```

Validator methods

```
public interface Validator {
  /** Validates all constraints on object. */
  validate(T object, Class<?>... groups)

  /** Validates all constraints placed on the property of object
*/
  validateProperty(T object, String pName, Class<?>... groups)

  /** Validates all constraints placed on the property
  * of the class beanType would the property value */
  validateValue(Class<T> type, String pName, Object val,
Class<?>…)
}
```

ConstraintViolation

- exposes constraint violation context
- core methods
  - getMessage
  - getRootBean
  - getLeafBean
  - getPropertyPath
  - getInvalidValue

Validating groups

- separate validating
- simple interfaces for grouping
- inheritance by standard java inheritance
- composition
- combining by `@GroupSequence`

Grouping(1)

- grouping interface

```
public interface MandatoryFieldCheck {
}
```

- using

```
public class Car {
  @Size(min = 3, max = 20, groups = MandatoryFieldCheck.class)
  private String name;

  @Size(max = 20)
  private String color;
}
```

Grouping(2)

- grouping sequence

```
@GroupSequence(Default.class, MandatoryFieldCheck.class)
public interface CarChecks {
}
```

- using

```
javax.validation.Validator validator;
validator.validate(make, CarChecks.class);
```

Composition

- annotation

```
@NotNull
@CapitalLetter
@Size(min = 2, max = 14)
@Documented
@Retention(RetentionPolicy.RUNTIME)
@Target({ METHOD, FIELD, ANNOTATION_TYPE })
public @interface CarNameConstraint {
}
```

- using

```
@CarNameConstraint
private String name;
```

Custom constraint

- create annotation

```
@Constraint(validatedBy=CapitalLetterValidator.class)
public @interface CapitalLetter {
  String message() default "{carbase.error.capital}";
```

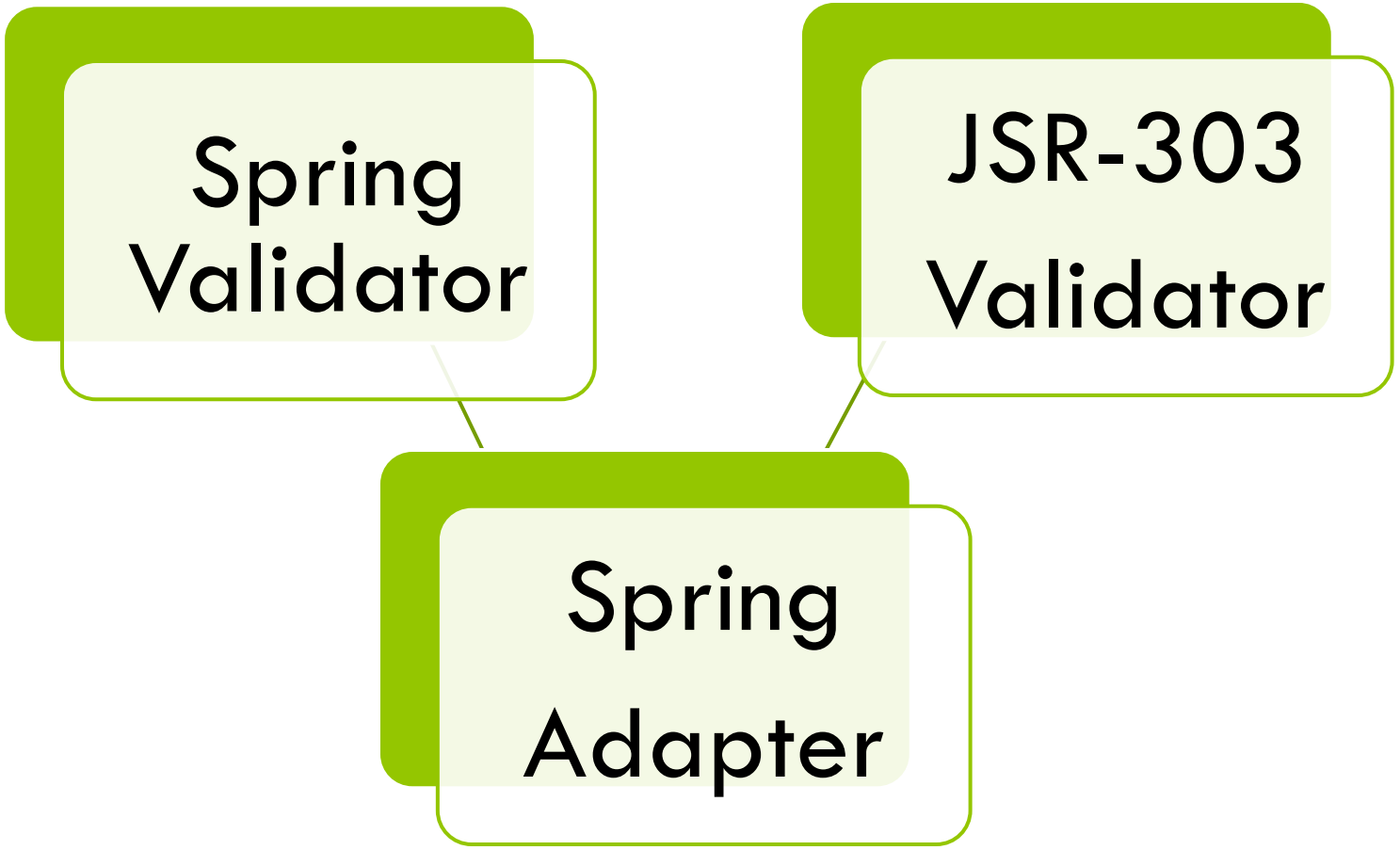
- implement constraint validator

```
public class CapitalLetterValidator implements
             ConstraintValidator<CapitalLetter, String> {
```

- define a default error message

```
carbase.error.capital=The name must begin with a capital letter
```

LocalValidatorFactoryBean



Configuration

- define bean

```
<bean id="validator"

class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean"/>
```

or

```
<mvc:annotation-driven/>
```

- injecting

```
@Autowired
private javax.validation.Validator validator;
```

or

```
@Autowired
private org.springframework.validation.Validator validator;
```

Information

- JSR-303 reference
  - http://docs.jboss.org/hibernate/validator/4.2/reference/en-US/html/
  - http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/validation.html
- samples
  - http://src.springsource.org/svn/spring-samples/mvc-showcase
- blog
  - http://blog.springsource.com/category/web/
- forum
  - http://forum.springsource.org/forumdisplay.php?f=25

Questions

The end

noskov.d@Gmail.com

my LinkedIn profile
http://www.linkedin.com/in/noskovd

slideshare
http://www.slideshare.net/analizator/presentations