CS544 EA

Applications

Validation: Annotations

Declaring Bean Constraints

- Constraints can be declared on:
  - Fields (validator framework will use reflection)
  - Properties (Class needs to adhere to JavaBean)
  - Constraint Inheritance (super class / interface)
  - Reference / creating a valid Object Graph
  - Class Level Constraints (always custom)
    - Useful for checking related properties
    - Eg. car.passengers <= car.seats

Provided Constraints 1/3

Annotation	Data Types	Description
@Null	Any	Check if it’s null (affects column)
@NotNull	Any	Check that it’s not null
@NotBlank	String	Not null, trimmed length > 0
@Valid	Any non-primitive	Go into the object and validate it
@AssertFalse	Boolean	Check that it’s false
@AssertTrue	Boolean	Check that it’s true
@Future	Date or Calendar	Check that it’s in the future
@Future OrPresent	Date or Calendar	Future or Preset
@Past	Date or Calendar	Check that it’s in the past
@PastOrPresent	Date or Calendar	Past or Present
@Size(min=,max=)	String / Collection	Check size is >= min and <= max, column length set to max
@Pattern(regex=,flag=)	String	Check that it matches the regex

Numeric Constraints (2/3)

Annotation	Data Types	Description
@Postitive	Numeric types	
@PositiveOrZero	Numeric types	
@Negative	Numeric types	
@NegativeOrZero	Numeric types	
@Min(value=)	Numeric types	Check that it’s not lower
@Max(value=)	Numeric types	Check that it’s not higher
@DecimalMin(value=,inclusive=)	Numeric types	Check that it’s not lower
@DecimalMax(value=,inclusive=)	Numeric types	Check that it’s not higher
@Digits(integer=,fraction=)	Numeric types	Checks if it has less digits / fractional points then given

@Min @Max and @Digits also affect DDL, adding constraints on the table column

@DecimalMin and @DecimalMax do not, but their min/max values can be specified as string which allows you to check beyond Long.MAX_VALUE / Long.MIN_VALUE

Additional Constraints

Annotation	Data Types	Description
@CreditCardNumber(..)	String	Credit Cards
@EAN	String	Barcode
@Email	String	Email address
@URL(…)	String	URL
@Length(min=,max=)	String	Column length set to max
@LuhnCheck(…)	String	Checksum (mod 10) CC
@Mod10Check(…)	String	Checksum (mod 10)
@Mod11Check(…)	String	Checksum (mod 11) (also used in ISBN)
@ISBN	String	Checks if valid ISBN number
@NotEmpty	String / Collection	Not null or empty
@Range(min=,max=)	Numeric	Checks >= min and <= max
@SafeHtml(…)	String	Requires jsoup, checks for <script> etc
@ScriptAssert(…)	Any Type	Executes JSR 233 script against target

Fields and Properties

Fields

```
public class Car {

  @NotNull
  private String manufacturer;

  @AssertTrue
  private boolean isRegistered;

  public Car(String manufacturer,
             boolean isRegistered) {
    this.manufacturer = manufacturer;
    this.isRegistered = isRegistered;
  }

  //getters and setters...
}
```

Properties

```
public class Car {
    private String manufacturer;
    private boolean isRegistered;
    public Car(String manufacturer, boolean
     isRegistered) {
        this.manufacturer = manufacturer;
        this.isRegistered = isRegistered;
    }

    @NotNull
    public String getManufacturer() {
        return manufacturer;
    }
    public void setManufacturer(String manufacturer)
     {
        this.manufacturer = manufacturer;
    }

    @AssertTrue
    public boolean isRegistered() {
        return isRegistered;
    }
    public void setRegistered(boolean isRegistered) {
        this.isRegistered = isRegistered;
    }
}
```

Container Types (Collections)

- Bean Validation 2.0 also adds support for:
  - Container constraints
  - Container cascades
  - Example:

```
private Map<@Valid @NotNull OrderCategory, List<@Valid @NotNull Order>> OrderByCategory
```

Inheritance

```java
public class Car {
    private String manufacturer;

    @NotNull
    public String getManufacturer() {
        return manufacturer;
    }

    // ...
}
```

```java
public class RentalCar extends Car {

    private String rentalStation;

    @NotNull
    public String getRentalStation() {
        return rentalStation;
    }

    //...
}
```

When validating RentalCar
both manufacturer and rentalStation
will be validated

Also works with interfaces

Object Graph

```
public class Car {

public class Car {

    @NotNull
    @Valid
    private Person driver;

    //...
}
```

```
public class Person {

    @NotNull
    private String name;

    //...
}
```

When validating Car
the @Valid makes the validator
cascade into Person and check
that its name is @NotNull

Class Level

```
@ValidPassengerCount
public class Car {

    private int seatCount;

    private List<Person> passengers;

    //...
}
```

You can make a custom class level annotations to check the relationship between properties

@ValidPassengerCount checks that:
passengers.size() <= seatCount
(see next slide)

Custom Constraint Annotation

```
@Target({ TYPE, ANNOTATION_TYPE })
@Retention(RUNTIME)
@Constraint(validatedBy = { ValidPassengerCountValidator.class })
@Documented
public @interface ValidPassengerCount {

    String message()
   default"{org.hibernate.validator.referenceguide.chapter06.classlevel." +
            "ValidPassengerCount.message}";

    Class<?>[] groups() default { };

    Class<? extends Payload>[] payload() default { };
}
```

Custom Validator

```
public class ValidPassengerCountValidator
  Implements ConstraintValidator<ValidPassengerCount, Car> {

  @Override
  public void initialize(ValidPassengerCount constraintAnnotation) {
  }

  @Override
  public boolean isValid(Car car, ConstraintValidatorContext context) {
    if (car == null) {
      return true;
    }
    return car.getPassengers().size() <= car.getSeatCount();
  }
}
```