FHIR Kochrezept

Inhalt

Relations

0..*

@OneToMany(cascade=CascadeType.ALL)

->FK ist immer auf der “n“ Seite

@JoinColumn(name=““, referencedColumn=““, nullable=true)

Private List<Item>

1…*

@NotEmpty

@OneToMany(cascade=CascadeType.ALL)

@JoinColumn(name=””, referencedColumn=””, nullable=false)

Private List<Item>

0…1

@Column(name=””)

1…1

@NotNull

@Column(name=””, nullable=false)

Boolean- boolean

0…1 boolean

Private Boolean

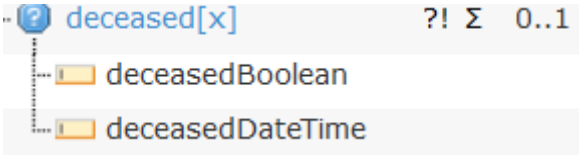
1…1 boolean

Private boolean

Zeichen

 nur eines folgenden eingerückten Felder darf befüllt sein

 nur eines darf befüllt sein



nur eines darf befüllt sein, wegen 0…1

dateTime

@PastOrPresent oder @PresentOrFuture

code

public enum Item { option1, option2}

@Enumerated(EnumType.STRING|ORDINAL)

@Column(name=“”)

private Item item;

Wenn ungültige Zeichen gefragt sind (_, -) dann so:

pubilc enum Item {

item1(„name“),

item2(„name2“);

private String value;

private Item (String value){

this.value = value;

}

Public String toString(){

Return this.value;

}

}

Neues Model erbt von Datentyp des ersten Feldes + Attribute eingerückt hinzufügen

Public class Item extends ItemAbove{}

Allgemein

@Entity

@Table(name=””)

@Setter

@Getter

@AllArgsConstructor

@NoArgsConstructor

@Builder

Keine Relation

@MappedSuperclass

@Getter

@Setter

@NoArgsConstructor

@AllArgsConstructor

Public abstract class Item extends Item1{}

Validators

positiveInt

@Min(1) oder @Positive

unsignedInt

@PositiveOrZero oder @Min(0)

OneToMany

@OneToMany(cascade=CascadeType.ALL, fetch=FetchType.EAGER)

->FK ist immer auf der “n“ Seite

@JoinColumn(name=““, referencedColumnName=““, nullable=false|true)

->referenziert auf PK der “1|0” Seite

Base64Binary

@Lob

Url,uri

Private String

Eigene Validatoren

Annotation Interface erstellen:

```
@Target({ElementType.TYPE}) //FIELD bei einem Feld
@Retention(RetentionPolicy.RUNTIME) //Wann ausgeführt? Während RUNTIME
@Constraint(validatedBy = {ItemValidator.class})
@Documented
public @interface ItemValid {

    //Errortext ausgeben
    String message() default "";

    //----- gehört einfach zur Annotation dazu---------
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
    //-------------------------------------------------
}
```

JavaClass:

```
public class ItemValidator implements ConstraintValidator<ItemValid, Item>
{
    @Override
    public void initialize(ItemValid constraintAnnotation){
    }
    @Override
    public boolean isValid(Item item, ConstraintValidatorContext context){
        //TODO: eigenen Validator schreiben
    }
}
```

Model Klasse Item:

@ItemValid

Controller:

Bei POST, PUT -> @Valid

Exception fangen:

```
@ExceptionHandler(ConstraintViolationException.class)
@ResponseStatus(HttpStatus.BAD_REQUEST)
@ResponseBody
public Map<String, String>
onConstraintValidationException(ConstraintViolationException e) {
    Map<String, String> errors = new HashMap<>();
    for (ConstraintViolation violation : e.getConstraintViolations()) {
        errors.put(violation.getPropertyPath().toString(),
violation.getMessage());
    }
    return errors;
}
```

```
@ExceptionHandler(MethodArgumentNotValidException.class)
@ResponseStatus(HttpStatus.BAD_REQUEST)
@ResponseBody
public Map<String, String>
onMethodArgumentNotValidException(MethodArgumentNotValidException e) {
    Map<String, String> errors = new HashMap<>();
    for (FieldError fieldError : e.getBindingResult().getFieldErrors()) {
        errors.put(fieldError.getField(), fieldError.getDefaultMessage());
    }
    return errors;
}
```