

# Zeiterfassungs Tool

...

SS 2019

# Anforderung & Aufgabe

## Verwaltung von

- ✓ Bereichen
- ✓ Projekten
- ✓ Unterprojekten
- ✓ Aufgaben
- ✓ Rollen
- ✓ Auftraggebern

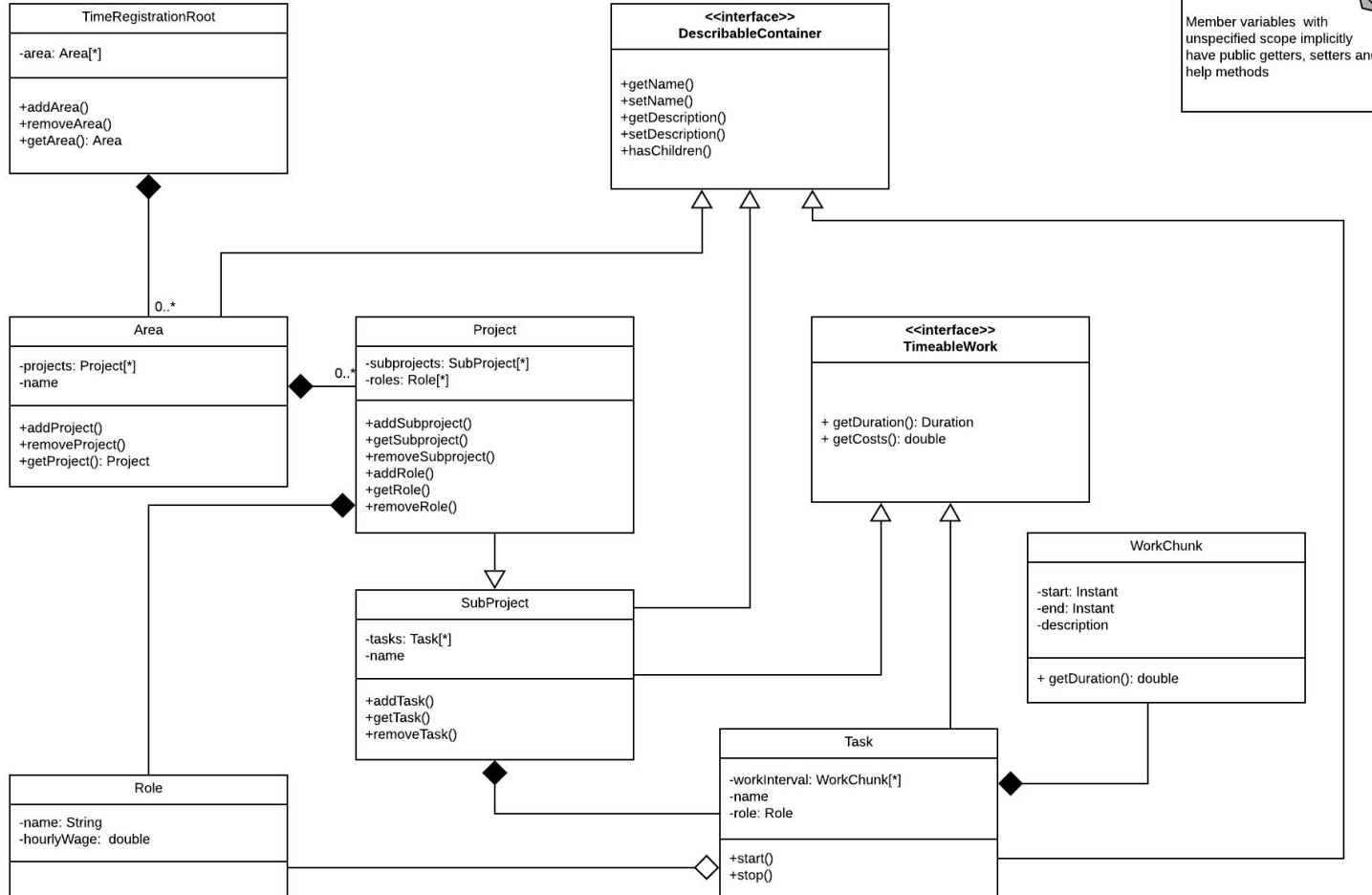
## Funktionalität

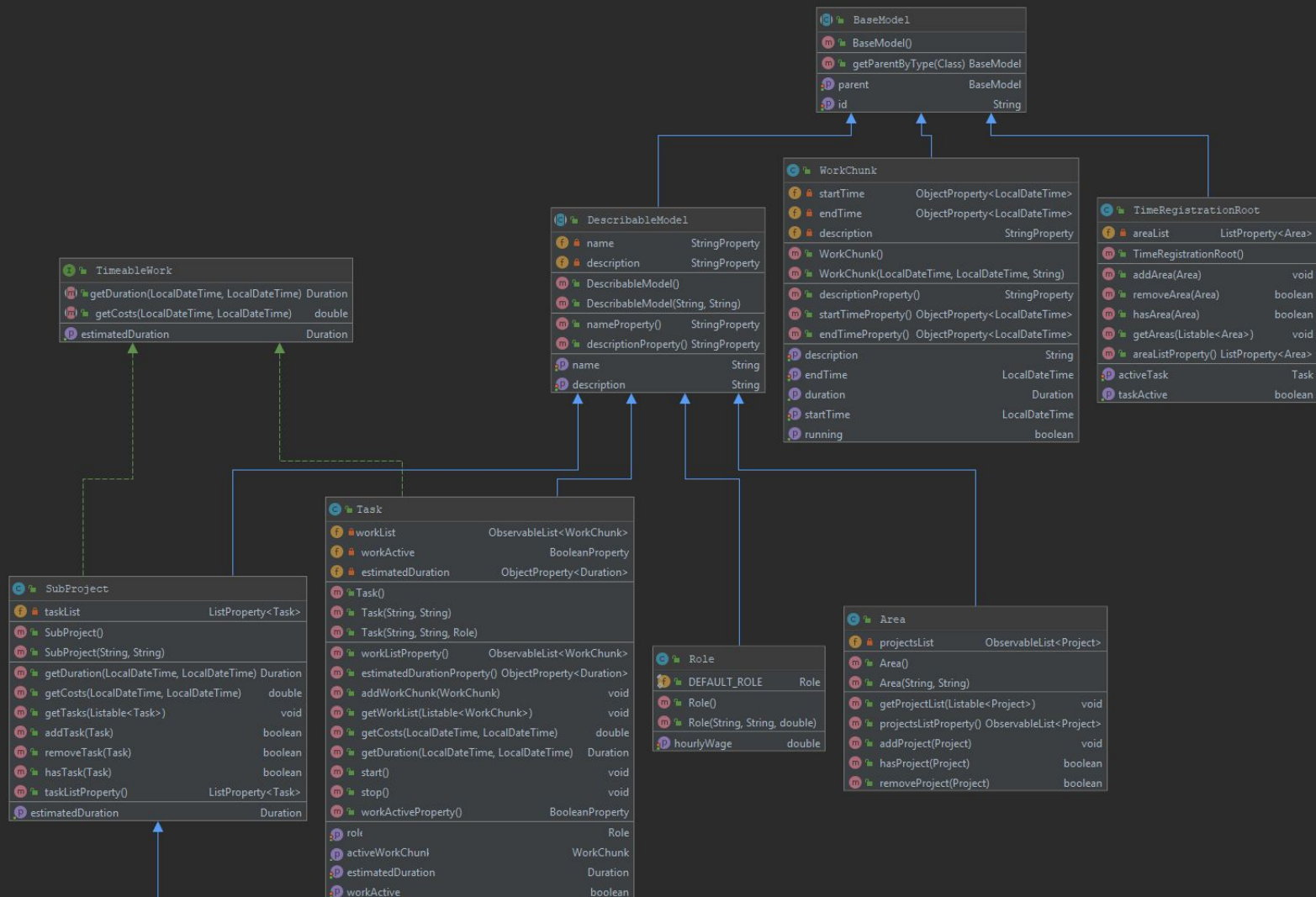
- ✓ Zeit messen
- ✓ Zeit auswerten
- ✓ Zeit schätzen
- ✓ Rechnung erstellen
- ✓ Stundenzettel erstellen












# Produktvorstellung























# UML Problembereichsmodell (Klassendiagramm) - Zeiterfassungstool







































	taskList	ListProperty<Task>
	SubProject()	
	SubProject(String, String)	
	getDuration(LocalDateTime, LocalDateTime)	Duration
	getCosts(LocalDateTime, LocalDateTime)	double
	getTasks(Listable<Task>)	void
	addTask(Task)	boolean
	removeTask(Task)	boolean
	hasTask(Task)	boolean
	taskListProperty()	ListProperty<Task>
	estimatedDuration	Duration

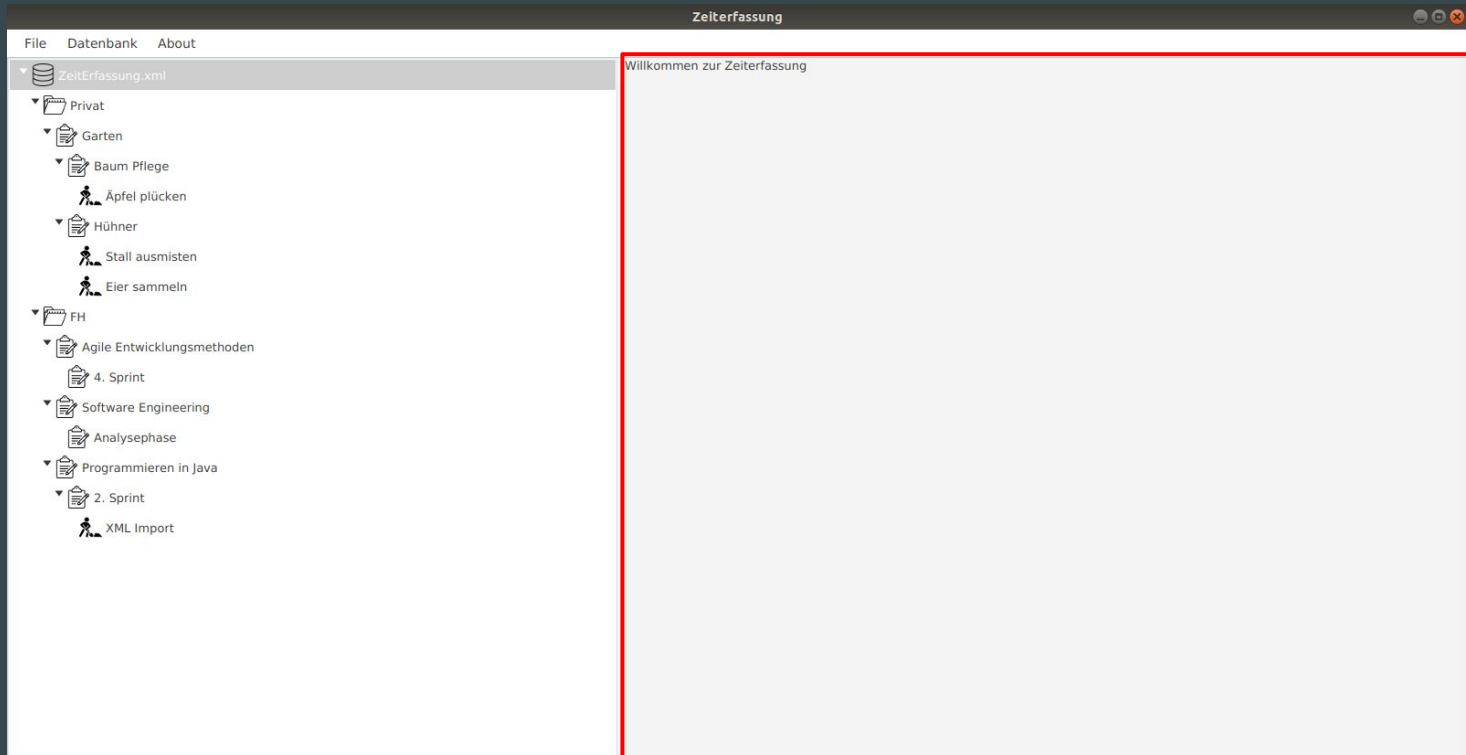
	Task(String, String)	
	Task(String, String, Role)	
	workListProperty()	ObservableList<WorkChunk>
	estimatedDurationProperty()	ObjectProperty<Duration>
	addWorkChunk(WorkChunk)	void
	getWorkList(Listable<WorkChunk>)	void
	getCosts(LocalDateTime, LocalDateTime)	double
	getDuration(LocalDateTime, LocalDateTime)	Duration
	start()	void
	stop()	void
	workActiveProperty()	BooleanProperty
	rolt	Role
	activeWorkChunk	WorkChunk
	estimatedDuration	Duration
	workActive	boolean

	Role	
	DEFAULT_ROLE	Role
	Role()	
	Role(String, String, double)	
	hourlyWage	double

	Area	
	projectsList	ObservableList<Project>
	Area()	
	Area(String, String)	
	getProjectList(Listable<Project>)	void
	projectsListProperty()	ObservableList<Project>
	addProject(Project)	void
	hasProject(Project)	boolean
	removeProject(Project)	boolean

	Project	
	customer	StringProperty
	roleList	ObservableList<Role>
	subProjectList	ListProperty<SubProject>
	Project()	
	Project(String, String)	
	getSubProjects(Listable<SubProject>)	void
	roleListProperty()	ObservableList<Role>
	subProjectListProperty()	ListProperty<SubProject>
	addSubProject(SubProject)	boolean
	removeSubProject(SubProject)	boolean
	hasSubProject(SubProject)	boolean
	getRoles(Listable<Role>)	void
	getRoleByName(String)	Role
	addRole(Role)	boolean
	removeRole(Role)	boolean
	hasRole(Role)	boolean
	customerProperty()	StringProperty
	getDuration(LocalDateTime, LocalDateTime)	Duration
	getCosts(LocalDateTime, LocalDateTime)	double
	defaultRole	Role
	rolesSize	int
	customer	String
	allTasks	ArrayList<Task>
	estimatedDuration	Duration

# JavaFX Base



# JavaFX Reactivity - Models um Properties erweitert

```
9  import javax.xml.bind.annotation.XmlType;
10
11  @XmlAccessorType(XmlAccessType.NONE)
12  public abstract class Dog {
13      private String name;
14
15      public Dog() {
16          super();
17      }
18
19      public Dog(String name) {
20          super();
21          setName(name);
22      }
23
24      @XmlElement(name = "name")
25      public String getName() {
26          return name;
27      }
28
29      public void setName(String name) {
30          this.name = name;
31      }
32  }
```

```
9  import javax.xml.bind.annotation.XmlType;
10
11  @XmlAccessorType(XmlAccessType.NONE)
12  public abstract class Dog {
13      private StringProperty name = new SimpleStringProperty();
14
15      public Dog() {
16          super();
17      }
18
19      public Dog(String name) {
20          super();
21          setName(name);
22      }
23
24      @XmlElement(name = "name")
25      public String getName() {
26          return name.get();
27      }
28
29      public void setName(String name) {
30          this.name.set(name);
31      }
32
33      public StringProperty nameProperty() {
34          return name;
35      }
36  }
```



# JavaFX Property Bindings

```
20
21     @FXML
22     private TextField name;
23
24     @FXML
25     private TextArea description;
26
27     @FXML
28     private Text timeEstimated;
29
30     @FXML
31     private Text timeSpent;
32
33     @FXML
34     private Text costs;
35
36     @FXML
37     private ProgressBar time;
38
39     /**
40      * Initializes the Controller with a SubProject
41      *
42      * @param subProject This SubProject is handled by the Controller
43      */
44     public void setSubProject(SubProject subProject) {
45         this.subProject = subProject;
46         name.textProperty().bindBidirectional(subProject.nameProperty());
47         description.textProperty().bindBidirectional(subProject.descriptionProperty());
```

# XML Serialisierung

## Warum XML?

- > Einfach zu verwenden (größtenteils nur Annotations setzen)
- > Sofort ersichtliche Hierarchien
- > Über Annotieren von getter-Methoden auch für z.B. Stringproperties geeignet

# XML Serialisierung

Nur die Elemente serialisieren, die annotiert sind und festlegen als XML Rotelement

U.a. Namensgebung für Element in XML Datei

```
21 @XmlAccessorType(XmlAccessType.NONE)
22 @XmlRootElement
23 public class Project extends SubProject {
24     /**
25      * The customer for to generate the invoice
26      */
27     private StringProperty customer = new SimpleStringProperty();
28
29     /**
30      * A List of Roles which can be assigned to the tasks
31      */
32     @XmlElement(name = "Role")
33     private ObservableList<Role> roleList = FXCollections.observableArrayList();
34
35     /**
36      * Containing subproject
37      */
38     @XmlElement(name = "SubProject")
39     private ListProperty<SubProject> subProjectList = new SimpleListProperty(FXCollections.observableArrayList());
40
41     @XmlElement(name = "Customer")
42     public String getCustomer() { return customer.getValue(); }
43
44
45 }
```

# XML Serialisierung

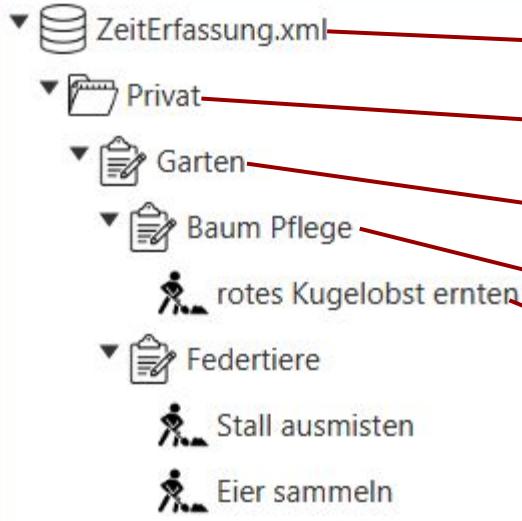
Für nicht-primitive Datentypen: Schreiben einer Adapterklasse

```
9      public class LocalDateTimeAdapter extends XmlAdapter<String, LocalDateTime> {  
10  
11          @Override  
12      public LocalDateTime unmarshal(String v) throws Exception {  
13          return LocalDateTime.parse(v);  
14      }  
15  
16          @Override  
17      public String marshal(LocalDateTime v) throws Exception {  
18          return v.toString();  
19      }  
20  }
```

Setzen der Adapter Annotation in der Klasse, die nicht-primitives Element enthält

```
52      @XmlJavaTypeAdapter(LocalDateTimeAdapter.class)  
53      @XmlElement(name = "start")  
54      public LocalDateTime getStartTime() {  
55          return startTime.getValue();  
56      }  
57
```

# XML Serialisierung (Ergebnis)



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Root id="0f9c119b-a1c6-4b82-9047-4346ab9bbbad">
  <Area id="7d41bd4e-2093-41e0-ae6c-0447e4018eca">
    <parent>0f9c119b-a1c6-4b82-9047-4346ab9bbbad</parent>
    <name>Privat</name>
    <description>Hier findet man alle meine privaten Projekte</description>
    <Project id="a7a79044-23b9-4872-aa59-4745b98e3d43">
      <parent>7d41bd4e-2093-41e0-ae6c-0447e4018eca</parent>
      <name>Garten</name>
      <description>Dieses Projekt ist für meine Garten Aufgaben</description>
      <SubProject id="bac7c862-06db-492d-8699-0681df4ad413">
        <parent>a7a79044-23b9-4872-aa59-4745b98e3d43</parent>
        <name>Baum Pflege</name>
        <description>Keine</description>
        <Task id="1e61712e-d277-4612-875a-1413c3905ace">
          <parent>bac7c862-06db-492d-8699-0681df4ad413</parent>
          <name>rotes Kugelobst ernten</name>
```

# Tests (Codebeispiel)

```
122      @Test
123      public void testAddRole() {
124          assertFalse(project.hasRole(role));
125          project.addRole(role);
126          assertTrue(project.hasRole(role));
127      }
```

Prüfen, dass das Projekt noch nicht die Rolle hat  
Rolle hinzufügen  
Prüfen, dass das Projekt die hinzugefügte Rolle hat

```
@Test
public void testGetCosts() {
    int duration = 42;
    WorkChunk workChunk = new WorkChunk(LocalDate.now(), LocalDateTime.now().plusHours(duration), description: "Testing");
    long wage = 15;
    Role role = new Role(name: "Name", description: "Beschreibung", wage);
    task.setRole(role);
    task.addWorkChunk(workChunk);
    long hours = 10;
    task.setEstimatedDuration(Duration.ofHours(hours));
    subProject.addTask(task);
    assertEquals(expected: (double) duration*wage, subProject.getCosts(LocalDate.MIN, LocalDateTime.MAX));
}
```

Fiktive Arbeit  
erstellen (Dauer:  
42h)

Neue Rolle mit  
Lohn=15€/h  
erstellen

Prüfe, ob erwarteter  
Wert (42h \* 15€/h =  
630€) eintritt

# Javadoc

```
/**
 * Represents a project. A Project is part of an area and contains tasks and subprojects.
 */
@XmlAccessorType(XmlAccessType.NONE)
@XmlRootElement
public class Project extends SubProject {
    /**
     * The customer for to generate the invoice
     */
    private StringProperty customer = new SimpleStringProperty();

    /**
     * A List of Roles which can be assigned to the tasks
     */
    @XmlElement(name = "Role")
    private ObservableList<Role> roleList = FXCollections.observableArrayList();

    /**
     * Containing subproject
     */
    @XmlElement(name = "SubProject")
```

OVERVIEW PACKAGE **CLASS** TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

zeiterfassung.models

## Class Project

java.lang.Object  
zeiterfassung.models.BaseModel  
zeiterfassung.models.DescribableModel  
zeiterfassung.models.SubProject  
zeiterfassung.models.Project

### All Implemented Interfaces:

TimeableWork

public class Project  
extends SubProject

Represents a project. A Project is part of an area and contains tasks and subprojects.

### Property Summary

All Methods	Instance Methods	Concrete Methods
Type	Property and Description	
javafx.beans.property.StringProperty	customer	The customer for to generate the invoice
javafx.collections.ObservableList<Role>	roleList	A List of Roles which can be assigned to the tasks
javafx.beans.property.ListProperty<SubProject>	subProjectList	Containing subproject

### Properties inherited from class zeiterfassung.models.SubProject

taskListProperty



# Stundenzettel Garten Juni 2017

## Garten

- Baum Pflege
  - Äpfel plücken

Start	Ende	Beschreibung
2017-06-05T17:32:38	2017-06-05T22:24:11	back to work for the 0 time
2017-06-06T02:45:17	2017-06-06T02:58:45	back to work for the 1 time
2017-06-06T12:11:09	2017-06-06T15:04:10	back to work for the 2 time
2017-06-06T19:22:08	2017-06-07T04:33:17	back to work for the 3 time
2017-06-07T13:36:24	2017-06-07T22:15:57	back to work for the 4 time
2017-06-07T23:07:21	2017-06-08T08:41:47	back to work for the 5 time
2017-06-08T15:40:59	2017-06-08T19:51:43	back to work for the 6 time

Aufgabe Kosten Gesamt: 1367,57€

Aufgabe Zeit Gesamt: 79h 3m

Unterprojekt Kosten Gesamt: 1367,57€

Unterprojekt Zeit Gesamt: 79h 3m

- Hühner
  - Stall ausmisten

Start	Ende	Beschreibung
2017-06-12T13:45:53	2017-06-12T18:06:15	back to work for the 15 time
2017-06-18T05:35:20	2017-06-18T10:24:51	back to work for the 31 time
2017-06-18T12:47:09	2017-06-18T21:49:35	back to work for the 32 time
2017-06-19T02:59:18	2017-06-19T12:49:52	back to work for the 33 time
2017-06-19T17:46:50	2017-06-19T23:22:34	back to work for the 34 time

Aufgabe Kosten Gesamt: 1668,30€

Aufgabe Zeit Gesamt: 96h 26m

Unterprojekt Kosten Gesamt: 3267,39€

Unterprojekt Zeit Gesamt: 188h 52m

Projekt Kosten Gesamt: 4634,96€

Projekt Zeit Gesamt: 267h 55m

## HTML Stundenzettel - Beispiel ProjectContent

```
Project project;
LocalDateTime start;
LocalDateTime stop;

@Override
public HTMLElement getHtmlNode() {

    HtmlTagElement root = SPAN.build().addProperty("style", "color: green");

    // Caption
    root.addElement(H4.build().addText(project.getName()), BR.build());

    HtmlTagElement ul = UL.build();

    // All Tasks
    project.getTasks(list -> {
        for (Task iter: list) {
            ul.addElement(LI.build().addElement(new TaskContent(iter, start, stop).getHtmlNode()));
        }
    });

    // All Subprojects
    project.getSubProjects(list ->{
        for (SubProject iter: list){
            ul.addElement(LI.build().addElement(new SubProjectContent(iter, start, stop).getHtmlNode()));
        }
    });

    root.addElement(ul).addElement(BR.build());

    root.addText("Projekt Kosten Gesamt: " + Utils.formatCosts(project.getCosts(start, stop)))
        .addElement(BR.build())
        .addText("Projekt Zeit Gesamt: " + Utils.formatDuration(project.getDuration(start, stop)));

    return root;
}
```

HTML Enum Factory

Kaskadierung



# Fazit

- Viel Spaß
- Viel Stress
- Erfolgreicher Projektabschluss
- Hoher Wissenszuwachs
- Vielen Dank!