Petriflow

Petriflow : Actions API

Created by Juraj Mažári, last modified on 04 Sep, 2018

Actions

make <Field f>,<Closure behaviour> on <Transition t> when <Closure<Boolean> condition>

Changes *behaviour* of given data field *f* on transition *t*, iff *condition* returns true. Behaviour can be one of:

- visible,
- editable,
- required,
- optional,
- hidden.

Example

```
garage_check: f.garage_check,
garage_cost: f.garage_cost,
garage: t.garage;
```

```
make garage_cost,visible on garage when {
        return garage_check.value == true;
}
```

change <Field> about <Closure>

Deprecated

See change value.

change <Field f> value <Closure calculation>

Sets new value to data field *f* returned by calculation closure. If the returned value is null, fields value is set to default value. If the returned value is *unchanged*, fields value is unchanged and actions with a trigger set on given field are not triggered.

Example

```
period: f.108001,
sum: f.308011;
change period value {
    def limit = 20.0;
        if (period.value == "polročná")
        limit = 40.0;
    if (period.value == "štvrťročná")
        limit = 80.0;
    if ((sum.value as Double) &lt; (limit as Double))
            return "ročná";
    return unchanged;
}
```

change <Field> choices <Closure choices>

Sets a new set of *choices* to data field *f*.

Example

```
other: f.410001,
field: f.this;
change field choices {
    if (other.value == "Nehnutelnost")
        return field.choices + ["rozostavaná stavba"];
    return field.choices;
}
```

generate <String method,Closure repeat> into <Field f>

Calls *method* and saves generated value into data field *f*. The field can be only of type *Text* or *File*. If repeat is equal to always new value is generated on each run of action. If repeat is equal to once new value is generated only if fields value is null.

Example

```
self: f.this;
generate "Insurance.offerPDF",always into self
```

changeCaseProperty <String property> about <Closure supplier>

Changes the *property* of the current case, the new value is generated by *the supplier*.

Example

```
trans: t.this;
```

```
changeCaseProperty "icon" about { trans.icon }
```

Case createCase(String identifier, String title = null, String color = "", User author = userService.ystem)

Creates a new instance of the newest version of net identified by the *identifier*. If the *title* is not specified, nets default case name is used. If the *colour* is null, the default colour is used (black at the moment).

Example

```
createCase("create_case_net","Create Case Case","green");
createCase("create_case_net","Create Case Case");
createCase("create_case_net");
```

Case createCase(PetriNet net, String title = net.defaultCaseName.defaultValue, String color = "", User author = userService.loggedOrSystem)

Creates a new instance of the given *net*. If the *title* is not specified, nets default case name is used. If the *colour* is null, the default colour is used (black at the moment).

Example

```
todo
```

List<Case> findCases(Closure<Predicate> predicate)

Finds all the cases that match the given *predicate*. The predicate is a groovy closure that accepts QCase object and returns QueryDSL Predicate.

Example

```
List<Case> cases = findCases( { it.title.eq("Case 1") } );
...
List<Case> cases = findCases( { it.dataSet.get("name").value.eq("Jozko") } );
```

Case findCase(Closure<Predicate> predicate)

Finds the first case that matches the given *predicate*. The predicate is a groovy closure that accepts QCase object and returns QueryDSL Predicate.

Example

```
Case useCase = findCase( { it.title.eq("Case 1") & it.processIdentifier.eq("insurance") } );
...
Case useCase = findCase( { it.dataSet.get("name").value.eq("Jozko") & it.processIdentifier.eq("in
```

List<Task> findTasks(Closure<Predicate> predicate)

Finds all tasks that match the given *predicate*. The predicate is a groovy closure that accepts QCase object and returns QueryDSL Predicate.

Example

```
def useCase = findCase(...)
Task task = findTask( { it.caseId.eq(useCase.stringId) & it.transitionId.eq("edit_limit") } );
```

Task findTask(Closure<Predicate> predicate)

Finds the first task that matches the given *predicate*. The predicate is a groovy closure that accepts QCase object and returns QueryDSL Predicate.

Example

```
List<Task> tasks = findTasks( { it.transitionId.eq("edit_limit") } )
...
def useCase = findCase(...)
List<Task> tasks = findTasks( { it.caseId.eq(useCase.stringId) } );
```

close <List<Transition>>

Deprecated
See cancel.

execute <String transitionId> where <Closure<Predicate>> with <Map>

Executes all fireable transitions identified by the *transitionId* in all case where the predicate returns true. For each task following actions are called:

1. assign to the system user
2. save new data values
3. finish.

The predicate is a list of Querydsl queries. Every case property can be used in a query. For more info see querydsl doc and QCase javadoc.

Example

```
field: f.field;
execute "synchronized" where ([
        "title eq Case 1"
] as List) with ([
        "field": [
        value: 128.0,
        type: "number"
        ]
] as Map)
```

Task assignTask(String transitionId, User user = userService.loggedOrSystem)

Assign the task in current case with given *transitionId*. Optional parameter *user* identifies actor who will perform assign.

Example

```
<action>
    <!-- @formatter:off -->
    selectedUser: f.select_controler,
    if (selectedUser.value) {
        def user = userService.findById(selectedUser.value.id, false)
        assignTask("control", user);
    }
    <!-- @formatter:on -->
</action>
```

Task assignTask(Task task, User user = userService.loggedOrSystem)

Assign the *task* to user. Optional parameter *user* identifies actor who will perform assign.

Example

```
<action>
```

```
    <!-- @formatter:off -->
    selectedUser: f.select_controler,
    if (selectedUser.value) {
                def usecase = findCase({ it.title("Some case") }).first()
                def task = findTask({ it.importId.eq("control") & it.caseId.eq(usecase.stringId)
        def user = userService.findById(selectedUser.value.id, false)
        assignTask(task, user);
    }
    <!-- @formatter:on -->
</action>
```

assignTasks(List<Task> tasks, User assignee = userService.loggedOrSystem)

cancelTask(String transitionId, User user = userService.loggedOrSystem)

Cancels the task in current case with given *transitionId*. Optional parameter *user* identifies actor who will perform cancel.

Example

```
def taskId = "work_task";
cancelTask(taskId);
```

cancelTask(Task task, User user = userService.loggedOrSystem)

cancelTasks(List<Task> tasks, User user = userService.loggedOrSystem)

finishTask(String transitionId, User assignee = userService.loggedOrSystem)

finishTask(Task task, User finisher = userService.loggedOrSystem)

finishTasks(List<Task> tasks, User finisher = userService.loggedOrSystem)

setData(Task task, Map dataSet)

Sets values of data fields on given *task*. Values are mapped to data fields in *dataSet* using data fields import Id as key.

Example

```
def usecase = findCase({ it.title.eq("Limits") }).first()
def task = findTask({ it.caseId.eq(usecase.stringId & it.transitionId.eq("edit_limit")) })
setData(task, [
    "new_limit": [
        "value": "10000",
        "type" : "number"
    ],
])
```

setData(Transition transition, Map dataSet)

Sets values of data fields on task of *transition* in current case. Values are mapped to data fields in *dataSet* using data fields import Id as key.

Example

```
transition: t.edit_limit;
setData(transition, [
    "new_limit": [
        "value": "10000",
        "type" : "number"
    ],
])
```

setData(String transitionId, Case useCase, Map dataSet)

Sets values of data fields on task identified by *transitionId* of given *case*. Values are mapped to data fields in *dataSet* using data fields import Id as key.

Example

```
def usecase = findCase({ it.title.eq("Limits") }).first()
setData("edit_limit", usecase, [
    "new_limit": [
        "value": "10000",
        "type" : "number"
    ],
])
```

Map<String, Field> getData(Task task)

Gets all data fields on given *task*, mapped by its import Id.

Example

```
actual_limit: f.actual_limit;
def usecase = findCase({ it.title.eq("Limits") }).first()
def task = findTask({ it.transitionId.eq("view_limit") & it.caseId.eq(usecase.stringId) })
def data = getData(task)
change actual_limit value {
    data["remote_limit"].value
}
```

Map<String, Field> getData(Transition transition)

Gets all data fields on the task of *transition* in the current case, mapped by its import Id.

Example

```
view_limit: t.view_limit;
actual_limit: f.actual_limit;
def data = getData(view_limit)
change actual_limit value {
    data["remote_limit"].value
}
```

Map<String, Field> getData(String transitionId, Case useCase)

Gets all data fields on the task defined by its *transitionId* in given *case*, mapped by its import Id.

Example

```
view_limit: t.view_limit;
def usecase = findCase({ it.title.eq("Limits") }).first()
def data = getData("view_limit", usecase)
change actual_limit value {
    data["remote_limit"].value
}
```

User assignRole(String roleId, User user = userService.loggedUser)

Assigns role identified by *roleId* to *user*. User is optional parameter, default value is currently logged user. Returns updated object of user.

Example

```
transition: t.task;
assignRole(transition.defaultRoleId);
```

Document generated by Confluence on 16 Oct, 2018 07:48