Finite State Machien Plugin - Reference Documentation

Puneet Behl

Version 1.0.0.BUILD-SNAPSHOT

Table of Contents

Chapter 1. Introduction

This plugin allow definition of simple workflows attached to domain classes, including states, events, transitions and conditions. Current workflow’s state will be held in domain class' property that must be defined. Multiple workflows can be defined on every domain class.

Chapter 2. Installation

In order to install the plugin, add the following dependency in your project `build.gradle` file as:

```
compile "org.grails.plugins:fsm:1.0.0.BUILD-SNAPSHOT"
```

Chapter 3. Configuration

The flow definition will be in a static block called `fsm_def`, which can hold several flow definition, each of them tied to one property in the class, such as:

```
static fsm_def = [
    <flow-property> : [
        <initial-value> : { <flow-definition-closure> }
        ],
    <another-flow> : [
        <initial-value-2> : { <flow-definition-closure> }
        ]
    ]
```

Every closure will use the `on`, `from`, `to`, `when` and `act` clauses to define the desired behavior. The structure of the flow definition closure is as follows:

```
{ flow ->
  flow.on (<event>) {
    from (<source>) [
        .when ( { <condition to allow the transition, must return boolean> })
    ].to (<destiny>) [
        .act ({ <code to run if success> })
    ]
  }
}
```

Chapter 4. Usage

Let's say you have the following domain class where you want to use Finite State Machine plugin functionality:

```
class SampleDomain {

    String name
    String status

    /**
     * fsm_def will override this value with 'none' in this example
     * and a Warning will be traced.
     */
    String mood = "ignored"

    /**
     * A number > 0 will allow the status flow to start
     */
    Integer amount
    /**
     * Sample method to call from conditions!
     */
    Boolean isSomethingPending() {
        return true
    }

    void doSomethingInteresting() {

    }

}
```

Following is the example `fsm_def` value for `SampleDomain`:

```
static fsm_def = [
        mood  : [ ①
                  none: { flow ->
                      flow.on('up') {
                          from('none').when({
                              status == 'running'  // Depends on second flow !
                          }).to('high')
                          from('low').to('high').act({
                              doSomethingInteresting()
                          })
                          from('high').to('high')
                          from('none').to('none') // Order is CRITICAL !
                      }
                      flow.on('down') {
                          from('none').when({
                              status == 'running'
                          }).to('low')
                          from('low').to('low')
                          from('high').to('low')
                          from('none').to('none') // Order is CRITICAL !
                      }
                  }
        ],
        status: [ ②
                  initial: { flow ->
                      flow.on('launch') {
                          from('initial').when({
                              isSomethingPending() && amount > 0
                          }).to('running')
                          from('initial').to('initial')
                      }
                      flow.on('stop') {
                          from('running').to('stopped')
                      }
                      flow.on('continue') {
                          from('stopped').to('running')
                      }
                      flow.on('finish') {
                          from('stopped').to('finished')
                          from('running').to('finished')
                      }
                  }
        ]
]
```

① The flow for property `status`.

② The flow for property `mood`.

The keys for `fsm_def` block should have corresponding properties in the domain class. Otherwise, the plugin will throw an error `Can't fire on flow 'nonflow' which is not defined in 'class testapp.SampleDomain'`.

4.1. Initial Values

When you create an instance of `SampleDomain` the properties `status` and `mood` will be initialzed with the values specificied in `fsm_def` block. In the above case, it will be `initial` and `none`. For example:

```
void "test initial values"() {

    when: "create a domain object"
    SampleDomain instance = new SampleDomain()

    then: "fsm fields will get initial values"
    instance.status == "initial"
    instance.mood == "none"
}
```

Please note that the default initial value (such as `String status = "test"`) of the field will be overriden by FSM

Chapter 5. Firing Events

Every domain that has FSM definition will get set of methods defined: