# Flowable6.4 –

> Flowable6.4 –

UserTask UserTask

UserTask UserTask

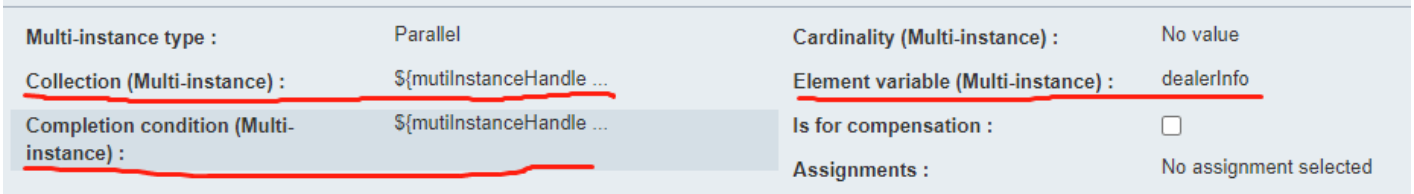
UserTask

```
/**
 *
 * @param execution
 * @return
 */
public List<DealerInfo> getList(DelegateExecution execution) {
    FlowElement flowElement = execution.getCurrentFlowElement();
    UserTask userTask = (UserTask) flowElement;
    UserTaskExtension userTaskExtension = FlowUtil.getUserTaskExtension(userTask);
    //DealerInfo
    //       UserTask
    //              UserTask
    return userTaskExtension.getDealers();
}

/**
 *
 * @param execution
 * @return
 */
public boolean getComplete(DelegateExecution execution) {
    Integer nrOfCompletedInstances = (Integer)
execution.getVariable("nrOfCompletedInstances");
    Integer nrOfInstances = (Integer) execution.getVariable("nrOfInstances");
    int agreeCount = 0, rejectCount = 0, abstainCount = 0;
    Map<String, Object> vars = execution.getVariables();
    for (String key : vars.keySet()) {
```

```
        //    SIGN_VOTE+TaskId
        //
        if (key.contains(FlowConst.SIGN_VOTE) && !key.equals(FlowConst.SIGN_VOTE_RESULT))
{
            Integer value = (Integer) vars.get(key);
            //
            //     ......
        }
    }
    //
    if (!nrOfCompletedInstances.equals(nrOfInstances)) {
        //
        return false;
    } else {
        //     ,
        if (rejectCount > 0) {
            //
            //  SIGN_VOTE+TaskId
            removeSignVars(execution);
            //
            execution.setVariable(FlowConst.SIGN_VOTE_RESULT, false);
            //
            return true;
        } else {
            //
            removeSignVars(execution);
            execution.setVariable(FlowConst.SIGN_VOTE_RESULT, true);
            return true;
        }
    }
}
```

UserTask




Colled `${mutiInstanceHandler.getList(execution)}`

Comple `${mutiInstanceHandler.getComplete(execution)}`

Element Var UserTask

## Flowable6.4 -

```
/**
 *
 */
@Override
protected void handleAssignments(TaskService taskService, String assignee, String owner,
List<String> candidateUsers, List<String> candidateGroups, TaskEntity task, ExpressionManager
expressionManager, DelegateExecution execution) {
    if (null != this.userTaskExtension) {
        //
        List<DealerInfo> dealerInfos = getDealerInfo();
        if (null == dealerInfos || dealerInfos.size() == 0) {
            throw new RuntimeException("       ");
        }
        if (hasMultiInstanceCharacteristics())
            //
            //        Element Var
            Object objDealer = execution.getVariable("dealerInfo");
            if (objDealer instanceof DealUserInfo) {
                DealUserInfo userInfo = (DealUserInfo) objDealer;
                assignee = userInfo.getId();
            } else if (objDealer instanceof DealRoleInfo) {
                DealRoleInfo roleInfo = (DealRoleInfo) objDealer;
                task.addGroupIdentityLink(roleInfo.getId(), "role");
            }
        }

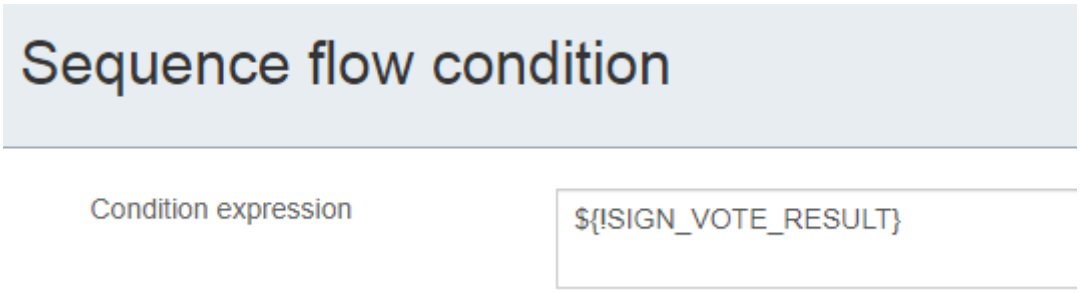
    }
    super.handleAssignments(taskService, assignee, owner, candidateUsers, candidateGroups,
task, expressionManager, execution);
}
```

Complete

```
//
```

```
variables.put(FlowConst.SIGN_VOTE + "_" + taskId, signVoteType.getCode());
```

Sequence Flow Condition




Revision #2
Created 30 July 2020 14:27:32 by
Updated 30 July 2020 14:38:17 by