JIDE Decision Grid Developer Guide

Purpose of This Document

JIDE Decision Grid is a Java/Swing implementation of the decision table. This developer guide is designed for developers who want to learn how to use *JIDE Decision Grid* in their applications.

JIDE Decision Grid heavily depends on features and components provided by *JIDE Grids*. So if you never used *JIDE Grids* before, we strongly recommend you read *JIDE Grids Developer Guide* first or at least refer to it while reading this developer guide.

Decision Table Basic

A decision is a choice about a "course of action". A course of action may include many individual actions. A decision may be characterized on a continuum from unstructured to structured.

Unstructured Decisions

Unstructured decisions are generally one-time propositions taken in emergent situations, i.e. the set of conditions is unique and there are no fixed rules for the course of action to take based on the conditions. The possible courses of action need not be finite. Making an unstructured decision is therefore heuristic. Automating such decisions involves the use of Decision Support Systems, which attempt to obtain and organize as much relevant information as possible for presentation to the decision maker. The decision maker then applies whatever heuristics he considers appropriate to come up with a course of action.

Structured Decisions

Structured decisions are predictable, i.e. given a particular set of conditions, the course of action to be taken is clear and definable. The choice is which actions to take among a predefined, finite collection of actions. Making a structured decision is therefore algorithmic. There are three common methods of expressing these algorithms. Note that all three of these methods are capable of dealing with multivariate, not merely binary conditions.

A decision table enables you to take what seems to be an indecipherable mass of facts and extract any trends and patterns buried in the data. You can organize and summarize your data, perform comparisons, and extract meaningful information that can be invaluable to you and your organization.

Decision Tables

A decision table is a two-dimensional matrix with one column for each possible action and one column for each relevant condition and one row for each combination of condition states. A decision table can very concisely and rigorously show complex conditions and their resulting actions while remaining comprehensible to a human reader.

Overview

In order to add a decision table to your applications, there are several steps you need to follow. Here is the flow. We will cover each step in details.

1. Define DecisionField
2. Define DecisionRule
3. Define DecisionDataModel
4. Create DecisionTablePane

DecisionField

DecisionField is a named object. The name should be unique within the same decision table. However you can define different display name or the title. The title doesn't have to be unique although it'd better to be. *DecisionField* also defines a type such as int.class, Date.class or boolean.class.

DecisionField could be used as either a condition or an action but you can use methods such as *setAllowedAsActionField*, *setAllowedAsConditionField* to limit its usage.

If your decision table requires a list of known conditions and actions, you could define a list of DecisionFields beforehand. But in most cases, you will let your user define DecisionField through drag-n-drop, dialog input box etc.

DecisionRule

DecisionRule has a list of DecisionEntry(s). You can view DecisionEntry as DecisionField + value. To define DecisionRule, you just use addCondition and addAction method to add the conditions and actions.

```
DecisionRule r1 = new DecisionRule();
r1.addCondition(new DecisionEntry(ageField, 30))
      .addCondition(new DecisionEntry(scoreField, 700))
      .addAction(new DecisionEntry(approvedField, true));
```

Of course, you only need to do this if you predefine the rule. If you always let users create rules, you don't need to bother these APIs.

DecisionDataModel

Once you defined the DecisionField(s) and DecisionRule(s), you can create a DecisionDataModel. DecisionDataModel has a list of DecisionRule(s).