Muhammad Sadiq

DEMO HARNESS

Service Curation Layer v2.5

(Recommendation Builder)

25 DEC 2016

Table of Contents

1. Overview
2. Recommendation Builder
   1. Introduction
   2. Capabilities
   3. Use Case Scenarios
3. Overview

Recommendation Builder, generates recommendations through reasoning on the user proﬁle and life-log data and the knowledge rules developed in a specific format. RB provided recommendations are considered as initial recommendation because of the fact that the recommendations are yet to be interpreted from the user’s contextual perspective. The initial recommendation may be forwarded as-is or transforming it to a more applicable form.

1. Recommendation Builder
   1. Capabilities

Recommendation Builder generates recommendation based on the input rules and facts.

Recommendation builder uses forward chaining algorithm to provide recommendations for the given situation. Forward chaining algorithm is implemented inside *PatternMatcher* class, the PatternMatcher may fire more than one recommendation rules for a single situation, now to resolve this conflict we need a kind of conflict resolution mechanism for this purpose we have provided *ConflictResolver* class which uses *maximum specificity* mechanism to select most appropriate recommendation from the set of recommendations provided by the PatternMatcher. You can find PatternMatcher and ConflictResolver classes inside *org.uclab.scl.framework.recbuilder*

* 1. Use Case Scenario’s
* Fire Rule for Specific Situation

*fireRule(event, rules)* method of PatternMatcher class is used to fire rule/s for specific event or situation. This method requires two parameters

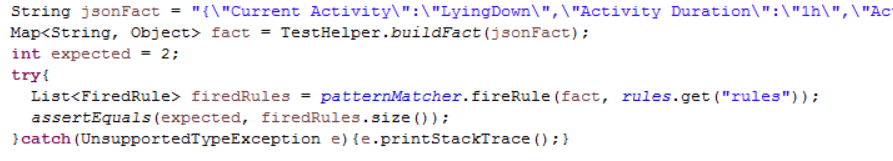
1. *Event – Map<String, Object>*
2. *Rules - List<Map<String, Object>>*

Calling this method with appropriate parameters will return list of FiredRule Objects e.g.

*List<FiredRule> firedRules = patternMatcher.fireRule(fact, rules.get("rules"));*

For more detail usage look over unit test cases in the following package:

*org.uclab.scl.framework.recbuilder*



* Conflict Resolution

As we mentioned above the pattern matcher may fire more than one rule for specific event or situation, so in that case we need to provide the fired rules to the conflict resolver so that it can resolve conflict among the fired rules and select the most applicable rule. You can use conflict resolver class as below:

*resolvedRules = conflictResolver.resolveConflict(firedRules)* for more details refer to unit test cases in the following package:

*org.uclab.scl.framework.recbuilder*