

JSON Rules

Adrian Giurca¹ and Emilian Pascalau¹

¹Institute of Informatics
Brandenburg University of Technology



4th Workshop on Knowledge Engineering
and Software Engineering, 2008,
Kaiserslautern, Germany

Motivation

- Nowadays, a large scale of Internet Applications are Rich Internet Applications
- However, much more sophisticated approaches are already experimented by main players (see for example IGoogle)
- Traditional RIA's development does not always meet the requirements of novel applications
- User created content and user-enabled UI's are on the front of current interest
- One possible solution is to enrich RIAs with rule-based reasoning.

Motivation

- Nowadays, a large scale of Internet Applications are Rich Internet Applications
- However, much more sophisticated approaches are already experimented by main players (see for example IGoogle)
- Traditional RIA's development does not always meet the requirements of novel applications
- User created content and user-enabled UI's are on the front of current interest
- One possible solution is to enrich RIAs with rule-based reasoning.

Motivation

- Nowadays, a large scale of Internet Applications are Rich Internet Applications
- However, much more sophisticated approaches are already experimented by main players (see for example IGoogle)
- Traditional RIA's development does not always meet the requirements of novel applications
- User created content and user-enabled UI's are on the front of current interest
- One possible solution is to enrich RIAs with rule-based reasoning.

Motivation

- Nowadays, a large scale of Internet Applications are Rich Internet Applications
- However, much more sophisticated approaches are already experimented by main players (see for example IGoogle)
- Traditional RIA's development does not always meet the requirements of novel applications
- User created content and user-enabled UI's are on the front of current interest
- One possible solution is to enrich RIAs with rule-based reasoning.

Motivation

- Nowadays, a large scale of Internet Applications are Rich Internet Applications
- However, much more sophisticated approaches are already experimented by main players (see for example IGoogle)
- Traditional RIA's development does not always meet the requirements of novel applications
- User created content and user-enabled UI's are on the front of current interest
- One possible solution is to enrich RIAs with rule-based reasoning.

Motivation

- Nowadays, a large scale of Internet Applications are Rich Internet Applications
- However, much more sophisticated approaches are already experimented by main players (see for example IGoogle)
- Traditional RIA's development does not always meet the requirements of novel applications
- User created content and user-enabled UI's are on the front of current interest
- One possible solution is to enrich RIAs with rule-based reasoning.

Goal

Let user modify (rule-based) the browser view of its preferred pages. For example, I can customize by myself the appearance of Yahoo portal.

1. A JSON-based rule language
2. A JavaScript-based rule engine

Goal

Let user modify (rule-based) the browser view of its preferred pages. For example, I can customize by myself the appearance of Yahoo portal.

1. A JSON-based rule language
2. A JavaScript-based rule engine

Goal

Let user modify (rule-based) the browser view of its preferred pages. For example, I can customize by myself the appearance of Yahoo portal.

1. A JSON-based rule language
2. A JavaScript-based rule engine

A JSON Rule

For all elements of class `note` having as first child a `ul` change the first child background color to `blue`.

```
{ "id": "rule101",
  "appliesTo": [
    "http://www.yahoo.com",
    "http://www.google.com/"
  ],
  "condition":
    "$X:Element( class=='note',
                  $Y:firstChild
                  )
    &&
    ($Y.nodeName == 'ul')",
  "actions": [
    "changeBackground($Y, 'blue')"
  ]
}
```

Overall Language Features

- Based on JSON (any rule is a JSON object)
- Uses a Drools-like condition language
- Uses JavaScript function calls as actions
- Supports both Production Rules and Event-Condition-Action Rules

Overall Language Features

- **Based on JSON (any rule is a JSON object)**
- Uses a Drools-like condition language
- Uses JavaScript function calls as actions
- Supports both Production Rules and Event-Condition-Action Rules

Overall Language Features

- Based on JSON (any rule is a JSON object)
- Uses a Drools-like condition language
- Uses JavaScript function calls as actions
- Supports both Production Rules and Event-Condition-Action Rules

Overall Language Features

- Based on JSON (any rule is a JSON object)
- Uses a Drools-like condition language
- Uses JavaScript function calls as actions
- Supports both Production Rules and Event-Condition-Action Rules

Overall Language Features

- Based on JSON (any rule is a JSON object)
- Uses a Drools-like condition language
- Uses JavaScript function calls as actions
- Supports both Production Rules and Event-Condition-Action Rules

UML model of JSON Rule

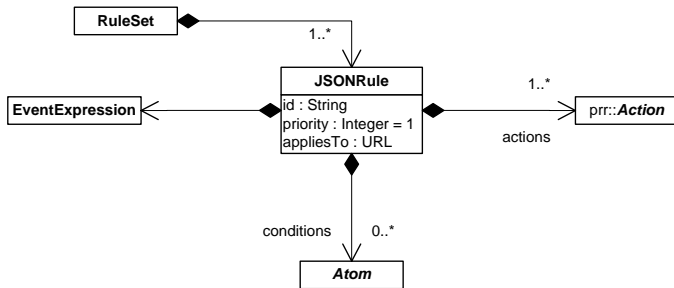


Figure: Rules and Rulesets

Condition Language

A rule condition is always a conjunction of atoms. Empty conditions are interpreted as true conditions.

- `JavaScriptBooleanCondition` the simplest conditional atom. Any JavaScript boolean expression is allowed.
- `Description`: related to the syntax of Drools pattern conditional.
- `XPathCondition`

Condition Language

A rule condition is always a conjunction of atoms. Empty conditions are interpreted as true conditions.

- `JavaScriptBooleanCondition` the simplest conditional atom. Any JavaScript boolean expression is allowed.
- `Description`: related to the syntax of Drools pattern conditional.
- `XPathCondition`

Condition Language

A rule condition is always a conjunction of atoms. Empty conditions are interpreted as true conditions.

- `JavaScriptBooleanCondition` the simplest conditional atom. Any JavaScript boolean expression is allowed.
- **Description:** related to the syntax of Drools pattern conditional.
- `XPathCondition`

Condition Language

A rule condition is always a conjunction of atoms. Empty conditions are interpreted as true conditions.

- `JavaScriptBooleanCondition` the simplest conditional atom. Any JavaScript boolean expression is allowed.
- `Description`: related to the syntax of Drools pattern conditional.
- `XPathCondition`

Atom types (Conditions)

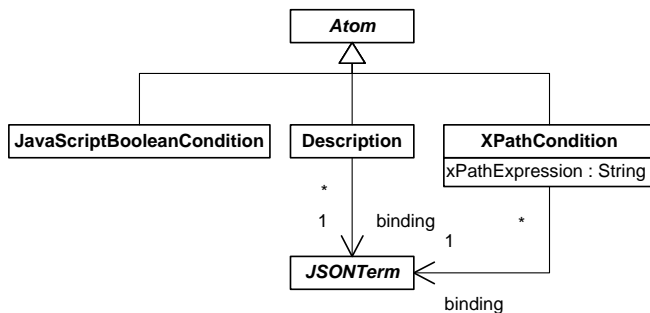


Figure: The language atoms

JavaScriptBooleanCondition

- the simplest conditional atom;
- **any** JavaScript boolean expression is allowed.

Examples:

```
window.find('rule')
```

```
document.getElementById('id').value==10
```

Descriptions

- related to the syntax of Drools pattern conditional;
- offers two types of constraints:
 1. `PropertyRestriction`: describes a set of value restrictions to properties of the `JSONTerm` that are bound to it.
 2. `PropertyBinding`: performs a variable binding of a property belonging to the related `JSONTerm`.

Descriptions - UML Model

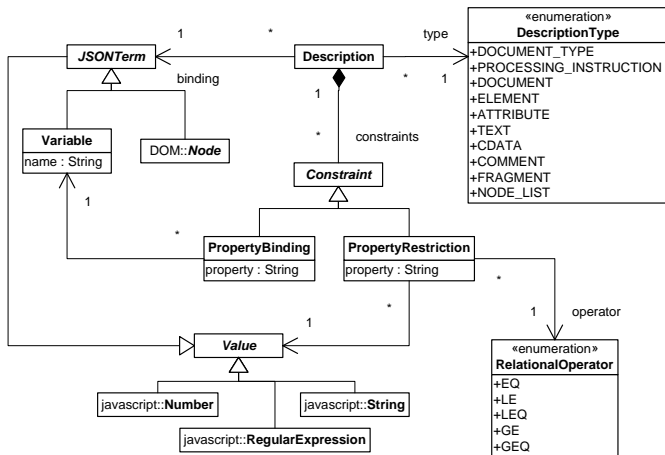


Figure: Descriptions

XPathCondition

Consider a HTML page that contains two tables:

T1:row 1, cell 1	T1:row 1, cell 2
T1:row 2, cell 1	T1:row 2, cell 2
T2:row 1, cell 1	T2:row 1, cell 2
T2:row 2, cell 1	T2:row 2, cell 2

A condition checking that the content of the first row of the second table is "T2:row1, cell 1", might be:

```
{ "nodeName": "tr",  
  "firstChild": { "nodeName": "td",  
                  "textContent": "T2:row1, cell 1"  
                }  
} in "html//table//tr"
```

XPathCondition - UML Model



Figure: The XPath Condition

Actions

- **standard actions** from OMG Production Rule Representation (PRR);
- **any** user-defined JavaScript functions can be called in the rule actions part.

PRR Standard Actions	JSON Rules
AssignExp	change properties of an element
InvokeExp	any JavaScript function call
AssertExp	insert a DOM node
RetractExp	remove a DOM node
UpdateExp	update a DOM node

Table: PRR Standard Actions and their representation

EventExpression

- JSON event expressions are related to the Event interface specification in *W3C DOM Level 2 Events*
- the properties of these expressions have the same meaning as in the W3C specification
- properties of event expressions are matched against the incoming DOM events at the runtime. Their values can be processed in the rule conditions and actions.

EventExpression - Example

When `click` event is raised, if the event comes from an `a` element, and if the `href` property matches a specific regular expression¹ then call `append()` with the message subject as parameter.

```
{ "id": "rule102",
  "appliesTo": ["http://mail.yahoo.com/"],
  "eventExpression": { "eventType": "click",
                      "eventTarget": "$X"
                    },
  "condition": " ($X.nodeName == 'a',
                $X.href=='match(showMessage\?fid=Inbox)') ",
  "actions": ["append($X.textContent) " ]
}
```

¹rudimentary check that the link is an inbox Yahoo message link

EventExpression - UML Model

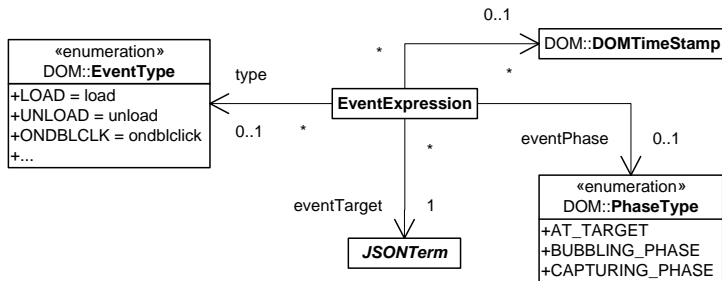


Figure: Event Expressions

Main Characteristics

- Tailored to execute rules related **only** to the page Document Object Model (DOM).
- DOM events are processed as atomic events (i.e. no duration).
- The working memory is the document DOM itself. Conditions are matched against DOM entities (such as *elements processing instructions, attributes* an so on).
- A forward chaining implementation;
- The engine executes rulesets²
- Rules are stored *locally* or *remote* or **both**.
- Rule serialization both RIF and R2ML.

²a ruleset is the set of all rules referring to a specific URL

Main Characteristics

- Tailored to execute rules related **only** to the page Document Object Model (DOM).
- DOM events are processed as atomic events (i.e. no duration).
- The working memory is the document DOM itself. Conditions are matched against DOM entities (such as *elements processing instructions, attributes* an so on).
- A forward chaining implementation;
- The engine executes rulesets²
- Rules are stored *locally* or *remote* or **both**.
- Rule serialization both RIF and R2ML.

²a ruleset is the set of all rules referring to a specific URL

Main Characteristics

- Tailored to execute rules related **only** to the page Document Object Model (DOM).
- DOM events are processed as atomic events (i.e. no duration).
- The working memory is the document DOM itself. Conditions are matched against DOM entities (such as *elements processing instructions, attributes* an so on).
- A forward chaining implementation;
- The engine executes rulesets²
- Rules are stored *locally* or *remote* or **both**.
- Rule serialization both RIF and R2ML.

²a ruleset is the set of all rules referring to a specific URL 

Main Characteristics

- Tailored to execute rules related **only** to the page Document Object Model (DOM).
- DOM events are processed as atomic events (i.e. no duration).
- The working memory is the document DOM itself. Conditions are matched against DOM entities (such as *elements processing instructions, attributes* an so on).
- A forward chaining implementation;
- The engine executes rulesets²
- Rules are stored *locally* or *remote* or **both**.
- Rule serialization both RIF and R2ML.

²a ruleset is the set of all rules referring to a specific URL 

Main Characteristics

- Tailored to execute rules related **only** to the page Document Object Model (DOM).
- DOM events are processed as atomic events (i.e. no duration).
- The working memory is the document DOM itself. Conditions are matched against DOM entities (such as *elements processing instructions, attributes* an so on).
- A forward chaining implementation;
 - The engine executes rulesets²
 - Rules are stored *locally* or *remote* or **both**.
 - Rule serialization both RIF and R2ML.

²a ruleset is the set of all rules referring to a specific URL 

Main Characteristics

- Tailored to execute rules related **only** to the page Document Object Model (DOM).
- DOM events are processed as atomic events (i.e. no duration).
- The working memory is the document DOM itself. Conditions are matched against DOM entities (such as *elements processing instructions, attributes* an so on).
- A forward chaining implementation;
- The engine executes rulesets²
 - Rules are stored *locally or remote* or **both**.
 - Rule serialization both RIF and R2ML.

²a ruleset is the set of all rules referring to a specific URL

Main Characteristics

- Tailored to execute rules related **only** to the page Document Object Model (DOM).
- DOM events are processed as atomic events (i.e. no duration).
- The working memory is the document DOM itself. Conditions are matched against DOM entities (such as *elements processing instructions, attributes* an so on).
- A forward chaining implementation;
- The engine executes rulesets²
- Rules are stored *locally* or *remote* or **both**.
- Rule serialization both RIF and R2ML.

²a ruleset is the set of all rules referring to a specific URL

Main Characteristics

- Tailored to execute rules related **only** to the page Document Object Model (DOM).
- DOM events are processed as atomic events (i.e. no duration).
- The working memory is the document DOM itself. Conditions are matched against DOM entities (such as *elements processing instructions, attributes* an so on).
- A forward chaining implementation;
- The engine executes rulesets²
- Rules are stored *locally* or *remote* or **both**.
- Rule serialization both RIF and R2ML.

²a ruleset is the set of all rules referring to a specific URL.

Implementation

- A prototype Firefox Add-On (Web Rules 0.3.0) by Matthias Tylkowski
<https://addons.mozilla.org/en-US/firefox/addon/8098>

Future Work

- Engine Implementation (JavaScript)
- Use cases implementations
- Rule Registry Implementation (Web Service, SOAP/REST)