

Configuration (embedded & server)

This document describes the contents of a Match4J configuration file.

The configuration file is an XML document which complies to the DTD as defined in “*documents/config/match4j-config.dtd*” with the following structure:

```
<matchengine>
  <datamanager>...</datamanager>
  <matchmodel>...</matchmodel>
</matchengine>
```

The root XML tag is “*matchengine*”. It contains a “*datamanager*” tag, which configures the origin of the match data, and a “*matchmodel*” tag, which configures how matching should occur.

Data manager

The data manager consists of a single tag, named “*cache*”, which configures the data cache. There is a separate manual that describes the data caches and their configuration (*match4j_datacaches.pdf*).

The following XML code shows an example configuration of the database data cache (included in the Match4J package):

```
<cache>
  <classname>
    com.match4j.data.cache.DatabaseReplicationCache
  </classname>
  <parameter>
    <key>config</key>
    <value>config/databaseReplicationCache.conf</value>
  </parameter>
</cache>
```

The “*cache*” tag contains a “*classname*” tag and zero or more “*parameter*” tags. The “*classname*” tag specifies the fully qualified class name of the desired data cache implementation. The “*parameter*” tags serve as configuration parameters for the data cache.

Match model

The match model consists of a definition of match data (“*entity type*”) and match rules (“*match rule list*”) on how the match data should be compared.

```
<matchmodel>
  <entitytype>...</entitytype>          (1 or more)
  <matchrulelist>...</matchrulelist>    (1 or more)
</matchmodel>
```

Configuration (embedded & server)

Defining entities

Match4J matches “*entities*”. An entity is, for example, a car or a person. An entity is defined by an “*entity type*”. An entity type consists of an id, a description and one or more attributes. An attribute has a name and a type (which can be a Java class that can be constructed from a string, for example *java.lang.Integer*).

The following example defines a “*car*” entity type having a model and a price as its attributes.

```
<entitytype>
  <id>Car</id>
  <description>Represents cars</description>
  <attribute>
    <attributeid>brand</attributeid>
    <description>Car brand</description>
    <classtype>java.lang.String</classtype>
  </attribute>
  <attribute>
    <attributeid>price</attributeid>
    <description>Price in euro</description>
    <classtype>java.lang.Float</classtype>
  </attribute>
</entitytype>
```

Defining match rules

Match rules determine the way in which entities are matched.

The match rules are grouped in a “*match rule list*”. It is possible to define multiple match rule lists. This makes it possible to define various configurations (groups) of match rules for various strategies of matching entity types. For example, a match rule list with two match rules for simple matching and match rule list with ten match rules for more advanced matching.

You can also use the match rule lists to define multiple match entity type mappings. For example, a match rule list that defines match rules that can match a car entity and another car entity or a match rule list that defines match rules that can match a buyer entity and a car entity.

```
<matchrulelist>
  <id>Car2Car</id>
  <description>Matches cars to cars</description>
  <requestentitytypeid>Car</requestentitytypeid>
  <resultentitytypeid>Car</resultentitytypeid>

  <matchrule>...</matchrule>      (1 or more)
</matchrulelist>
```

The “*id*” gives a unique name to the list and the “*description*” tag a short description. The “*requestentitytypeid*” determines the entity type that is delivered in the

Configuration (embedded & server)

match assignments. The “*resultentitytypeid*” determines the entity of the answer. Furthermore, the “*matchrulelist*” tag contains of one or more “*matchrule*” tags. The following XML example shows a match rule:

```
<matchrule>
  <id>CarSunroofMatchRule</id>
  <description>
    Matches the sunroof option of the car,
    discrete matching
  </description>
  <weight>1.0</weight>
  <implementingclass>
    <classname>...</classname>
    <parameter>                                (0 or more)
      <key>...</key>
      <value>...</value>
    </parameter>
  </implementingclass>
</matchrule>
```

A match rule consists of:

- *id*: a unique name for the match rule
- *description*: a description of the match rule
- *weight*: weight of the match rule. Weight determines to what extent a match rule counts in the total match. Weight of match rules are relative to another. For example, a match rule with weight “2” counts twice as much as a match rule with weight “1” in the match result calculation. These values can also be for example “100” and “50”.

```
Example:
Match rule A, weight is 2, match score 50%
Match rule B weight is 1, match score is 80%
Match score calculation: (2*50 + 1*80) / (2+1) = 60%
```

- *implementingclass*: defines the Java class that implements the match rule. Match4J already includes a number of match rule implementations. However, it is also possible to create new implementations by implementing the “*com.match4j.IMatchRule*” interface.

The implementing class consists of a “*classname*” tag in which the fully qualified class name should be indicated. The match rule can be configured by zero or more “*parameter*” tags.

Match4J is supplied with several match rule implementations. These implementations are described in detail in a separate manual (match4j_matchrules.pdf).