The ObjectWeb Consortium

Specification

Perseus:
The Pool
–
Specification

Authors:

S Chassande-Barrioz (France Telecom R&D)

P Dechamboux (France Telecom R&D)

Released: January 27, 2004

Status: Draft

Version: 1.3

TABLE OF CONTENTS

TABLE OF FIGURES

1 INTRODUCTION

1.1 Overview

This document presents the definition (API and concepts) of a pool component.

1.2 Scope

1.3 Rationale

1.4 Goals

1.5 Document Convention

A Times Roman font is used for the default text.

```
A courier font is used for code fragments.
```

2 ARCHITECTURE

2.1 Overview

A pool is a simple component delivring reusable resources. It avoids the creation of a resource at each need. An unsused resource must be released in the pool after its use. Released instances will be reused later. The resource is not typed in order to pool all java objects. The figure below represents a Pool component.

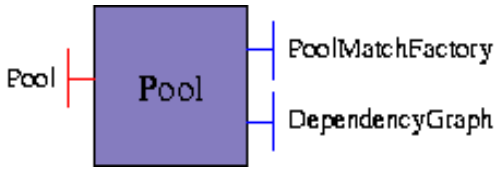

Figure 1: Pool overview

A pool requires a PoolMatchFactory able to create a resource, to select right resource instance and finally to destroy a resource. The dependency graph can be used to avoid dead lock.

2.2 The Pool interface

The Pool interface is very simple. It permits to fetch a resource through the getResource methods, and to release a resource through the releaseResource method. Here is the interface definition:

```
package org.objectweb.perseus.pool.api;
public interface Pool {
    Object getResource(Object hints) throws PoolException;
```

This method finds a free resource or create an new if the maximal size is not reached. The hints parameter of the getResource method permits to the PoolMatchFactory to choose a resource among the free resources.

```
    Object getResource(Object hints, Object user)
            throws PoolException, DeadLockException;
```

This method has the same principle that the previous. But the additional 'user' parameter is the identifier of the context requiring a resource. In case of no resource is availlable, the pool can waits that another user releases a resource. But this wait can create a dead lock. Then in order to avoid it, the pool implementation can use a dependency graph detecting cycles in the dependency graph of context. The DeadLockException is thrown in case of the wait of an availlable resource occurs an dead lock.

```
    int getSize();
```

This method retrieves the current size of the pool. This number is equals to the number of used resources plus the number of free resources.

```
    void releaseResource(Object resource) throws PoolException;
```

This method permits to release an instance into the pool. If the resource is not known or already released a PoolException is thrown.

```
}
```

2.3 The PoolAttributeController interface

This interface permits to configure four pool attributes. Here is the definition of the interface and a description of each attributes

```
package org.objectweb.perseus.pool.api;
public interface PoolAttributes extends AttributeController {
     int getMinSize();
     void setMinSize(int minsize) throws PoolException;
```

The minimal size attribute permits to define the number of resource which must already exist in the pool. The aim is always have availlable resources in order to loose time during the first uses.

```
     int getMaxSize();
     void setMaxSize(int maxsize) throws PoolException;
```

The maximal size attribute permits to fix maximal number of resource which can be allocated. The maximal pool size contains the number of used resources and the free resources. The -1 value means that there is no limit. The 0 value has any sense.

```
     long getTimeout();
     void setTimeout(long to);
```

The Timeout attribute is when the maximal pool size is reached. It indicates the time to wait a free resource. The time is declared in millisecond. The -1 value means that the pool must wait until a resource has been released. The 0 value means that the pool does not wait a free resource. Then in this case a PoolException is thrown immediately.

```
     long getTTL();
     void setTTL(long ttl);
```

The TTL attribute represents a time to live for free resources. When the TTL of an unused resources expires, the resource is destroyed. It is used to limit the life of an unused resource. This permits also to decrease the number of allocated resources to the minimum size when theey are not used.

```
}
```

2.4 The PoolMatchFactory interface

The aims of the PoolMatchFactory match the tree method declared in the following interface definition:

```
package org.objectweb.perseus.pool.api;
public interface PoolMatchFactory {
     Object createResource(Object hints) throws PoolException;
```

It creates a new resource. The hints parameter is the one specified in a getResouce method. This parameter can help the resource creation/initialisation.

```
     boolean matchResource(Object resource, Object hints);
```

This method permits to choose a instance which will be returned by a getResource method. The hints parameter is the one specified in a getResouce method. The factory can use the hints parameter to choose the right resource instance. If any free resouce match and the maximal pool size is not reached, the pool orders to the factory to create a new resource with the specified hints.

```
     void destroyResource(Object resource);
```

This method is called when a resource is going to be destroyed. It is a listener.

```
}
```