Apache Commons Pool

共享池提供一个对象池化API和一些对象池实现。

The Apache Commons Pool open source software library provides an object-pooling API and a number of object pool implementations. Version 2 of Apache Commons Pool contains a completely re-written pooling implementation compared to the 1.x series. In addition to performance and scalability improvements, version 2 includes robust instance tracking and pool monitoring. Version 2 requires JDK level 1.6 or above.

Releases

共享池的第2版包含一个完全重写的对象池化实现。
除了性能和扩展性的改进，版本2还包括强大的实例跟踪和池监控。
第2版需要JDK 1.6或以上

See the downloads page for information on obtaining releases.

Features

The org.apache.commons.pool2 package defines a handful of pooling interfaces and some base classes that may be useful when creating new pool implementations.

pool2包定义了少量的池化接口和一些基类，这些基类可能在创建新的池实现时是很有用的。

PooledObjectFactory

`PooledObjectFactory` provides a generic interface for managing the lifecycle of a pooled object:

提供了一个"管理一个池对象的生命周期"的泛型接口。

```
public interface PooledObjectFactory<T> {
    PooledObject<T> makeObject();
    void activateObject(PooledObject<T> obj);
    void passivateObject(PooledObject<T> obj);
    boolean validateObject(PooledObject<T> obj);
    void destroyObject(PooledObject<T> obj);
}
```

Users of 1.x versions of Commons Pool will notice that while the `PoolableObjectFactory` s used by 1.x pools create and manage pooled objects directly, version 2 `PooledObjectFactory` s create and manage `PooledObject` s. These object wrappers maintain object pooling state, enabling `PooledObjectFactory` methods to have access to data such as instance creation time or time of last use. A `DefaultPooledObject` is provided, with natural implementations for pooling state methods. The simplest way to implement a `PoolableObjectFactory` is to have it extend `BasePooledObjectFactory` . This factory provides a `makeObject()` that returns `wrap(create())` where `create` and `wrap` are abstract. You provide an implementation of `create` to create the underlying objects that you want to manage in the pool and `wrap` to wrap created instances in `PooledObject` s. To use `DefaultPooledObject` wrappers, use

`PooledObjectFactory`s负责创建和管理`PooledObject`s
`DefaultPooledObject`是`PooledObject`的默认实现
实现一个`PooledObjectFactory`的最简单方式是继承自`BasePooledObjectFactory`

```
@Override
public PooledObject<Foo> wrap(Foo foo) {
    return new DefaultPooledObject<Foo>(foo);
}
```

where `Foo` is the type of the objects being pooled (the return type of `create()`).

Another important difference between 1.x and version 2 pools is that the implementations provided maintain references to all objects under management by the pool. Correct behavior depends on underlying instances being discernable by equals - i.e., if

所提供的"维护引用到所有由池管理的对象"的实现，正确的行为依赖于后端的实例是有迹可寻的。

A and B are two different instances being managed by the pool, A.equals(B) should return false.

`KeyedPooledObjectFactory` defines a similar interface for `KeyedObjectPool`s:

```
public interface KeyedPoolableObjectFactory<K,V> {
    PooledObject<V> makeObject(K key);
    void activateObject(K key, PooledObject<V> obj);
    void passivateObject(K key, PooledObject<V> obj);
    boolean validateObject(K key, PooledObject<V> obj);
    void destroyObject(K key, PooledObject<V> obj);
}
```

`BaseKeyedPooledObjectFactory` provides an abstract base implementation of `KeyedPooledObjectFactory`.

The org.apache.commons.pool2.impl package provides some *Pool* implementations.

GenericObjectPool

提供各种各样的配置选项

`GenericObjectPool` provides a wide variety of configuration options, including the ability to cap the number of idle or active instances, to evict instances as they sit idle in the pool, etc. As of version 2, `GenericObjectPool` also provides abandoned instance tracking and removal.

提供废弃的实例跟踪和移除

`GenericKeyedObjectPool` offers the same behavior for keyed pools.

SoftReferenceObjectPool

`SoftReferenceObjectPool` can grow as needed, but allows the garbage collector to evict idle instances from the pool as needed.

允许垃圾收集器在必要情况下驱逐池中的空闲实例。

Migrating from Pool 2.x to Pool 2.y

不需要代码更改

Client code that uses a Pool 2.x release should require no code changes to work with a later Pool 2.x release.

New Pool 2.x releases may include support for new configuration attributes. These will be listed in the change log. Note that the MBean interfaces (those with names ending in MXBean or MBean) such as `DefaultPooledObjectInfoMBean`, `GenericKeyedObjectPoolMXBean` or `GenericKeyedObjectPoolMXBean` may change from one release to the next to support these new attributes. These interfaces should, therefore, not be implemented by client as the changes will not be backwards compatible.

Migrating from Pool 1.x to Pool 2.x

The migration from Apache Commons Pool 1.x to 2.x will require some code changes. The most significant changes are the changes in package name from `org.apache.commons.pool` to `org.apache.commons.pool2` and the change in the implementation classes to use `PooledObjectFactory`s, as described above.

The key implementation classes (`GenericObjectPool` and `GenericKeyedObjectPool`) have retained their names so no changes should be required there although a number of attributes have been renamed to improve consistency and ensure attributes with the same name in different pools have the same meaning. It is likely that some changes will be required to use the new attribute names.