Memcached flush_all 命令 Windows 下安装 Memcached

Java 连接 Memcached 服务

使用 Java 程序连接 Memcached，需要在你的 classpath 中添加 Memcached jar 包。

本站 jar 包下载地址：spymemcached-2.10.3.jar。

Google Code jar 包下载地址：spymemcached-2.10.3.jar（需要科学上网）。

以下程序假定 Memcached 服务的主机为 127.0.0.1，端口为 11211。

连接实例

Java 连接 Memcached

MemcachedJava.java 文件：

```
import net.spy.memcached.MemcachedClient;
import java.net.*;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 本地连接 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 关闭连接
mcc.shutdown();
}catch(Exception ex){
System.out.println( ex.getMessage() );
}
}
}
```

该程序中我们使用 InetSocketAddress 连接 IP 为 127.0.0.1 端口 为 11211 的 memcached 服务。

执行以上代码，如果连接成功会输出以下信息：

```
Connection to server successful.
```

set 操作实例

以下使用 java.util.concurrent.Future 来存储数据

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
```

```
// 存储数据
Future fo = mcc.set("runoob", 900, "Free Education");
// 查看存储状态
System.out.println("set status:" + fo.get());
// 输出值
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 关闭连接
mcc.shutdown();
}catch(Exception ex){
System.out.println( ex.getMessage() );
}
}
}
```

执行程序，输出结果为：

```
Connection to server successful.
set status:true
runoob value in cache - Free Education
```

add 操作实例

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 添加数据
Future fo = mcc.set("runoob", 900, "Free Education");
// 打印状态
System.out.println("set status:" + fo.get());
// 输出
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 添加
fo = mcc.add("runoob", 900, "memcached");
// 打印状态
System.out.println("add status:" + fo.get());
// 添加新key
fo = mcc.add("codingground", 900, "All Free Compilers");
// 打印状态
System.out.println("add status:" + fo.get());
// 输出
System.out.println("codingground value in cache - " + mcc.get("codingground"));
// 关闭连接
mcc.shutdown();
}catch(Exception ex){
System.out.println(ex.getMessage());
```

```
}
}
}
```

replace 操作实例

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try {
//连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 添加第一个 key=》value 对
Future fo = mcc.set("runoob", 900, "Free Education");
// 输出执行 add 方法后的状态
System.out.println("add status:" + fo.get());
// 获取键对应的值
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 添加新的 key
fo = mcc.replace("runoob", 900, "Largest Tutorials' Library");
// 输出执行 set 方法后的状态
System.out.println("replace status:" + fo.get());
// 获取键对应的值
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 关闭连接
mcc.shutdown();
}catch(Exception ex){
System.out.println( ex.getMessage() );
}
}
}
```

append 操作实例

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 添加数据
Future fo = mcc.set("runoob", 900, "Free Education");
// 输出执行 set 方法后的状态
System.out.println("set status:" + fo.get());
```

```
// 获取键对应的值
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 对存在的key进行数据添加操作
fo = mcc.append("runoob", 900, " for All");
// 输出执行 set 方法后的状态
System.out.println("append status:" + fo.get());
// 获取键对应的值
System.out.println("runoob value in cache - " + mcc.get("codingground"));
// 关闭连接
mcc.shutdown();
}catch(Exception ex) {
System.out.println(ex.getMessage());
]
}
}
```

prepend 操作实例

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 添加数据
Future fo = mcc.set("runoob", 900, "Education for All");
// 输出执行 set 方法后的状态
System.out.println("set status:" + fo.get());
// 获取键对应的值
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 对存在的key进行数据添加操作
fo = mcc.prepend("runoob", 900, "Free ");
// 输出执行 set 方法后的状态
System.out.println("prepend status:" + fo.get());
// 获取键对应的值
System.out.println("runoob value in cache - " + mcc.get("codingground"));
// 关闭连接
mcc.shutdown();
}catch(Exception ex) {
System.out.println(ex.getMessage());
}
}
}
```

CAS 操作实例

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.CASValue;
import net.spy.memcached.CASResponse;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 添加数据
Future fo = mcc.set("runoob", 900, "Free Education");
// 输出执行 set 方法后的状态
System.out.println("set status:" + fo.get());
// 使用 get 方法获取数据
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 通过 gets 方法获取 CAS token（令牌）
CASValue casValue = mcc.gets("runoob");
// 输出 CAS token（令牌） 值
System.out.println("CAS token - " + casValue);
// 尝试使用cas方法来更新数据
CASResponse casresp = mcc.cas("runoob", casValue.getCas(), 900, "Largest Tutorials-Library");
// 输出 CAS 响应信息
System.out.println("CAS Response - " + casresp);
// 输出值
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 关闭连接
mcc.shutdown();
}catch(Exception ex) {
System.out.println(ex.getMessage());
}
}
}
```

get 操作实例

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 添加数据
Future fo = mcc.set("runoob", 900, "Free Education");
// 输出执行 set 方法后的状态
System.out.println("set status:" + fo.get());
// 使用 get 方法获取数据
System.out.println("runoob value in cache - " + mcc.get("runoob"));
```

```
// 关闭连接
mcc.shutdown();
}catch(Exception ex) {
System.out.println(ex.getMessage());
}
}
}
```

gets 操作实例、CAS

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.CASValue;
import net.spy.memcached.CASResponse;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 添加数据
Future fo = mcc.set("runoob", 900, "Free Education");
// 输出执行 set 方法后的状态
System.out.println("set status:" + fo.get());
// 从缓存中获取键为 runoob 的值
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 通过 gets 方法获取 CAS token（令牌）
CASValue casValue = mcc.gets("runoob");
// 输出 CAS token（令牌） 值
System.out.println("CAS value in cache - " + casValue);
// 关闭连接
mcc.shutdown();
}catch(Exception ex) {
System.out.println(ex.getMessage());
}
}
}
```

delete 操作实例

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
```

```
// 添加数据
Future fo = mcc.set("runoob", 900, "World's largest online tutorials library");
// 输出执行 set 方法后的状态
System.out.println("set status:" + fo.get());
// 获取键对应的值
System.out.println("runoob value in cache - " + mcc.get("runoob"));
// 对存在的key进行数据添加操作
fo = mcc.delete("runoob");
// 输出执行 delete 方法后的状态
System.out.println("delete status:" + fo.get());
// 获取键对应的值
System.out.println("runoob value in cache - " + mcc.get("codingground"));
// 关闭连接
mcc.shutdown();
}catch(Exception ex) {
System.out.println(ex.getMessage());
}
}
}
```

Incr/Decr 操作实例

MemcachedJava.java 文件：

```
import java.net.InetSocketAddress;
import java.util.concurrent.Future;
import net.spy.memcached.MemcachedClient;
public class MemcachedJava {
public static void main(String[] args) {
try{
// 连接本地的 Memcached 服务
MemcachedClient mcc = new MemcachedClient(new InetSocketAddress("127.0.0.1", 11211));
System.out.println("Connection to server sucessful.");
// 添加数字值
Future fo = mcc.set("number", 900, "1000");
// 输出执行 set 方法后的状态
System.out.println("set status:" + fo.get());
// 获取键对应的值
System.out.println("value in cache - " + mcc.get("number"));
// 自增并输出
System.out.println("value in cache after increment - " + mcc.incr("number", 111));
// 自减并输出
System.out.println("value in cache after decrement - " + mcc.decr("number", 112));
// 关闭连接
mcc.shutdown();
}catch(Exception ex) {
System.out.println(ex.getMessage());
}
}
}
```

← Memcached flush_all 命令 Windows 下安装 Memcached →

点我分享笔记