个人资料

Mr_xiaoM

关注 发私信

访问： 12738次

积分： 306

等级： BLOG 2

排名： 千里之外

原创： 12篇 转载： 40篇

译文： 0篇 评论： 0条

文章搜索

文章分类

java (32)

php (8)

算法 (7)

前端 (3)

web service (4)

工具 (6)

其他 (8)

阅读排行

使用Jersey框架创建RESTful ... (614)

使用 jersey构建RESTful的We... (540)

存储到数据库的文章如何保留... (478)

Netty In Action中文版 - 第八... (470)

Netty In Action中文版 - 第一... (461)

Netty In Action中文版 - 第十... (427)

Netty In Action中文版 - 第十... (397)

Netty In Action中文版 - 第五... (396)

Netty In Action中文版 - 第七... (361)

Netty In Action中文版 - 第十... (350)

推荐文章

* Android 反编译初探 应用是如何被注入广告的
* 凭兴趣求职80%会失败，为什么
* 安卓微信自动抢红包插件优化和实现

Netty In Action中文版 - 第十四章：实现自定义的编码解码器

标签： java netty nio

2014-09-15 21:26 350人阅读 评论

分类： java (31)

目录(?) [+]

本章讲述Netty中如何轻松实现定制的编解码器，由于Netty架构的灵活性，这些编解码器易于重用和测试。为了更容 Memcached作为协议例子是因为它更方便我们实现。

Memcached是免费开源、高性能、分布式的内存对象缓存系统，其目的是加速动态Web应用程序的响应，减轻数 实际上是一个以key-value存储任意数据的内存小块。可能有人会问“为什么使用Memcached？”，因为Memcachec 解。

14.1 编解码器的范围

我们将只实现Memcached协议的一个子集，这足够我们进行添加、检索、删除对象；在Memcached中是通过执 来实现的。Memcached支持很多其他的命令，但我们只使用其中三个命令，简单的东西，我们才会理解的更清楚。

Memcached有一个二进制和纯文本协议，它们都可以用来与Memcached服务器通信，使用什么类型的协议取决 本章主要关注实现二进制协议，因为二进制在网络编程中最常用。

14.2 实现Memcached的编解码器

当想要实现一个给定协议的编解码器，我们应该花一些事件来了解它的运作原理。通常情况下，协议本身都有一些 会发现多少细节？幸运的是Memcached的二进制协议可以很好的扩展。

在RFC中有相应的规范，并提供了Memcached二进制协议下载地 址：http://code.google.com/p/memcached/wiki/BinaryProtocolRevamped。我们不会执行Memcached的所有命 作：SET,GET和DELETE。这样做事为了让事情变得简单。

14.3 了解Memcached二进制协议

可以在http://code.google.com/p/memcached/wiki/BinaryProtocolRevamped上详细了解Memcached二进制 站如果不翻墙的话好像访问不了。

14.4 Netty编码器和解码器

14.4.1 实现Memcached编码器

先定义memcached操作码(Opcode)和响应状态码(Status)：

[java] view plain copy

```java
package netty.in.action.mem;

/**
 * memcached operation codes
 * @author c.king
 *
 */
public class Opcode {

    public static final byte GET = 0x00;
    public static final byte SET = 0x01;
    public static final byte DELETE = 0x04;
```

* 【游戏设计模式】之四 《游戏编程模式》全书内容提炼总结
* 带你开发一款给Apk中自动注入代码工具iodetools(完善篇)


```
13.
14. }
```

[java] view plain copy

```java
01. package netty.in.action.mem;
02.
03. /**
04.  * memcached response statuses
05.  * @author c.king
06.  *
07.  */
08. public class Status {
09.
10.     public static final short NO_ERROR = 0x0000;
11.     public static final short KEY_NOT_FOUND = 0x0001;
12.     public static final short KEY_EXISTS = 0x0002;
13.     public static final short VALUE_TOO_LARGE = 0x0003;
14.     public static final short INVALID_ARGUMENTS = 0x0004;
15.     public static final short ITEM_NOT_STORED = 0x0005;
16.     public static final short INC_DEC_NON_NUM_VAL = 0x0006;
17.
18. }
```

继续编写memcached请求消息体：

[java] view plain copy

```java
01. package netty.in.action.mem;
02.
03. import java.util.Random;
04.
05. /**
06.  * memcached request message object
07.  * @author c.king
08.  *
09.  */
10. public class MemcachedRequest {
11.
12.     private static final Random rand = new Random();
13.     private int magic = 0x80;// fixed so hard coded
14.     private byte opCode; // the operation e.g. set or get
15.     private String key; // the key to delete, get or set
16.     private int flags = 0xdeadbeef; // random
17.     private int expires; // 0 = item never expires
18.     private String body; // if opCode is set, the value
19.     private int id = rand.nextInt(); // Opaque
20.     private long cas; // data version check...not used
21.     private boolean hasExtras; // not all ops have extras
22.
23.     public MemcachedRequest(byte opcode, String key, String value) {
24.         this.opCode = opcode;
25.         this.key = key;
26.         this.body = value == null ? "" : value;
27.         // only set command has extras in our example
28.         hasExtras = opcode == Opcode.SET;
29.     }
30.
31.     public MemcachedRequest(byte opCode, String key) {
32.         this(opCode, key, null);
33.     }
34.
35.     public int getMagic() {
36.         return magic;
37.     }
38.
39.     public byte getOpCode() {
40.         return opCode;
41.     }
42.
43.     public String getKey() {
44.         return key;
45.     }
46.
47.     public int getFlags() {
48.         return flags;
49.     }
50.
51.     public int getExpires() {
52.         return expires;
53.     }
54.
55.     public String getBody() {
56.         return body;
57.     }
58.
```

```
    public int getId() {
        return id;
    }

    public long getCas() {
        return cas;
    }

    public boolean isHasExtras() {
        return hasExtras;
    }

}
```

最后编写memcached请求编码器：

[java] view plain copy

```
package netty.in.action.mem;

import io.netty.buffer.ByteBuf;
import io.netty.channel.ChannelHandlerContext;
import io.netty.handler.codec.MessageToByteEncoder;
import io.netty.util.CharsetUtil;

/**
 * memcached request encoder
 * @author c.king
 *
 */
public class MemcachedRequestEncoder extends MessageToByteEncoder<MemcachedRequest> {

    @Override
    protected void encode(ChannelHandlerContext ctx, MemcachedRequest msg, ByteBuf out)
            throws Exception {
        // convert key and body to bytes array
        byte[] key = msg.getKey().getBytes(CharsetUtil.UTF_8);
        byte[] body = msg.getBody().getBytes(CharsetUtil.UTF_8);
        // total size of body = key size + body size + extras size
        int bodySize = key.length + body.length + (msg.isHasExtras() ? 8 : 0);
        // write magic int
        out.writeInt(msg.getMagic());
        // write opcode byte
        out.writeByte(msg.getOpCode());
        // write key length (2 byte) i.e a Java short
        out.writeShort(key.length);
        // write extras length (1 byte)
        int extraSize = msg.isHasExtras() ? 0x08 : 0x0;
        out.writeByte(extraSize);
        // byte is the data type, not currently implemented in Memcached
        // but required
        out.writeByte(0);
        // next two bytes are reserved, not currently implemented
        // but are required
        out.writeShort(0);
        // write total body length ( 4 bytes - 32 bit int)
        out.writeInt(bodySize);
        // write opaque ( 4 bytes) - a 32 bit int that is returned
        // in the response
        out.writeInt(msg.getId());
        // write CAS ( 8 bytes)
        // 24 byte header finishes with the CAS
        out.writeLong(msg.getCas());
        if(msg.isHasExtras()){
            // write extras
            // (flags and expiry, 4 bytes each), 8 bytes total
            out.writeInt(msg.getFlags());
            out.writeInt(msg.getExpires());
        }
        //write key
        out.writeBytes(key);
        //write value
        out.writeBytes(body);
    }

}
```

14.4.2 实现Memcached解码器

编写memcached响应消息体：

[java] view plain copy

```
package netty.in.action.mem;

```

```java
/**
 * memcached response message object
 * @author c.king
 *
 */
public class MemcachedResponse {

    private byte magic;
    private byte opCode;
    private byte dataType;
    private short status;
    private int id;
    private long cas;
    private int flags;
    private int expires;
    private String key;
    private String data;

    public MemcachedResponse(byte magic, byte opCode, byte dataType, short status,
            int id, long cas, int flags, int expires, String key, String data) {
        this.magic = magic;
        this.opCode = opCode;
        this.dataType = dataType;
        this.status = status;
        this.id = id;
        this.cas = cas;
        this.flags = flags;
        this.expires = expires;
        this.key = key;
        this.data = data;
    }

    public byte getMagic() {
        return magic;
    }

    public byte getOpCode() {
        return opCode;
    }

    public byte getDataType() {
        return dataType;
    }

    public short getStatus() {
        return status;
    }

    public int getId() {
        return id;
    }

    public long getCas() {
        return cas;
    }

    public int getFlags() {
        return flags;
    }

    public int getExpires() {
        return expires;
    }

    public String getKey() {
        return key;
    }

    public String getData() {
        return data;
    }

}
```

编写memcached响应解码器：

```java
[java] view plain copy
package netty.in.action.mem;

import io.netty.buffer.ByteBuf;
import io.netty.channel.ChannelHandlerContext;
import io.netty.handler.codec.ByteToMessageDecoder;
import io.netty.util.CharsetUtil;

import java.util.List;
```

```

public class MemcachedResponseDecoder extends ByteToMessageDecoder {

    private enum State {
        Header, Body
    }

    private State state = State.Header;
    private int totalBodySize;
    private byte magic;
    private byte opCode;
    private short keyLength;
    private byte extraLength;
    private byte dataType;
    private short status;
    private int id;
    private long cas;

    @Override
    protected void decode(ChannelHandlerContext ctx, ByteBuf in, List<Object> out)
            throws Exception {
        switch (state) {
        case Header:
            // response header is 24 bytes
            if (in.readableBytes() < 24) {
                return;
            }
            // read header
            magic = in.readByte();
            opCode = in.readByte();
            keyLength = in.readShort();
            extraLength = in.readByte();
            dataType = in.readByte();
            status = in.readShort();
            totalBodySize = in.readInt();
            id = in.readInt();
            cas = in.readLong();
            state = State.Body;
            break;
        case Body:
            if (in.readableBytes() < totalBodySize) {
                return;
            }
            int flags = 0;
            int expires = 0;
            int actualBodySize = totalBodySize;
            if (extraLength > 0) {
                flags = in.readInt();
                actualBodySize -= 4;
            }
            if (extraLength > 4) {
                expires = in.readInt();
                actualBodySize -= 4;
            }
            String key = "";
            if (keyLength > 0) {
                ByteBuf keyBytes = in.readBytes(keyLength);
                key = keyBytes.toString(CharsetUtil.UTF_8);
                actualBodySize -= keyLength;
            }
            ByteBuf body = in.readBytes(actualBodySize);
            String data = body.toString(CharsetUtil.UTF_8);
            out.add(new MemcachedResponse(magic, opCode, dataType, status,
                    id, cas, flags, expires, key, data));
            state = State.Header;
            break;
        default:
            break;
        }
    }

}
```

14.5 测试编解码器

基于netty的编解码器都写完了，下面我们来写一个测试它的类：

[java] view plain copy

```

package netty.in.action.mem;

import io.netty.buffer.ByteBuf;
import io.netty.channel.embedded.EmbeddedChannel;
import io.netty.util.CharsetUtil;

import org.junit.Assert;
```

```java
import org.junit.Test;

/**
 * test memcached encoder
 * @author c.king
 *
 */
public class MemcachedRequestEncoderTest {

    @Test
    public void testMemcachedRequestEncoder() {
        MemcachedRequest request = new MemcachedRequest(Opcode.SET, "k1", "v1");
        EmbeddedChannel channel = new EmbeddedChannel(
                new MemcachedRequestEncoder());
        Assert.assertTrue(channel.writeOutbound(request));
        ByteBuf encoded = (ByteBuf) channel.readOutbound();
        Assert.assertNotNull(encoded);
        Assert.assertEquals(request.getMagic(), encoded.readInt());
        Assert.assertEquals(request.getOpCode(), encoded.readByte());
        Assert.assertEquals(2, encoded.readShort());
        Assert.assertEquals((byte) 0x08, encoded.readByte());
        Assert.assertEquals((byte) 0, encoded.readByte());
        Assert.assertEquals(0, encoded.readShort());
        Assert.assertEquals(2 + 2 + 8, encoded.readInt());
        Assert.assertEquals(request.getId(), encoded.readInt());
        Assert.assertEquals(request.getCas(), encoded.readLong());
        Assert.assertEquals(request.getFlags(), encoded.readInt());
        Assert.assertEquals(request.getExpires(), encoded.readInt());
        byte[] data = new byte[encoded.readableBytes()];
        encoded.readBytes(data);
        Assert.assertArrayEquals((request.getKey() + request.getBody())
                .getBytes(CharsetUtil.UTF_8), data);
        Assert.assertFalse(encoded.isReadable());
        Assert.assertFalse(channel.finish());
        Assert.assertNull(channel.readInbound());
    }

}
```

[java] view plain copy

```java
package netty.in.action.mem;

import io.netty.buffer.ByteBuf;
import io.netty.buffer.Unpooled;
import io.netty.channel.embedded.EmbeddedChannel;
import io.netty.util.CharsetUtil;

import org.junit.Assert;
import org.junit.Test;

/**
 * test memcached decoder
 *
 * @author c.king
 *
 */
public class MemcachedResponseDecoderTest {

    @Test
    public void testMemcachedResponseDecoder() {
        EmbeddedChannel channel = new EmbeddedChannel(
                new MemcachedResponseDecoder());
        byte magic = 1;
        byte opCode = Opcode.SET;
        byte dataType = 0;
        byte[] key = "Key1".getBytes(CharsetUtil.UTF_8);
        byte[] body = "Value".getBytes(CharsetUtil.UTF_8);
        int id = (int) System.currentTimeMillis();
        long cas = System.currentTimeMillis();
        ByteBuf buffer = Unpooled.buffer();
        buffer.writeByte(magic);
        buffer.writeByte(opCode);
        buffer.writeShort(key.length);
        buffer.writeByte(0);
        buffer.writeByte(dataType);
        buffer.writeShort(Status.KEY_EXISTS);
        buffer.writeInt(body.length + key.length);
        buffer.writeInt(id);
        buffer.writeLong(cas);
        buffer.writeBytes(key);
        buffer.writeBytes(body);
        Assert.assertTrue(channel.writeInbound(buffer));
        MemcachedResponse response = (MemcachedResponse) channel.readInbound();
        assertResponse(response, magic, opCode, dataType, Status.KEY_EXISTS, 0,
                0, id, cas, key, body);
    }
```

```
47.
48.     private static void assertResponse(MemcachedResponse response, byte magic,
49.             byte opCode, byte dataType, short status, int expires, int flags,
50.             int id, long cas, byte[] key, byte[] body) {
51.         Assert.assertEquals(magic, response.getMagic());
52.         Assert.assertArrayEquals(key,
53.                 response.getKey().getBytes(CharsetUtil.UTF_8));
54.         Assert.assertEquals(opCode, response.getOpCode());
55.         Assert.assertEquals(dataType, response.getDataType());
56.         Assert.assertEquals(status, response.getStatus());
57.         Assert.assertEquals(cas, response.getCas());
58.         Assert.assertEquals(expires, response.getExpires());
59.         Assert.assertEquals(flags, response.getFlags());
60.         Assert.assertArrayEquals(body,
61.                 response.getData().getBytes(CharsetUtil.UTF_8));
62.         Assert.assertEquals(id, response.getId());
63.     }
64.
65. }
```

14.6 Summary

本章主要是使用netty写了个模拟memcached二进制协议的处理。至于memcached二进制协议具体是个啥玩意，没有详细说明。

- 上一篇　Netty In Action中文版 - 第十六章：从EventLoop取消注册和重新注册
- 下一篇　Netty In Action中文版 - 第十五章：选择正确的线程模型


人事管理系统


发短信平台


猎头公司


文档管理系统


订单管理系统

查看评论

暂无评论

发表评论

用 户 名： gavin2lee7

评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

全部主题 Hadoop AWS 移动游戏 Java Android iOS Swift 智能硬件 Docker OpenStack VP

IE10 Eclipse CRM JavaScript 数据库 Ubuntu NFC WAP jQuery BI HTML5 Spring Apa

HTML SDK IIS Fedora XML LBS Unity Splashtop UML components Windows Mobile Ra

Cassandra CloudStack FTC coremail OPhone CouchBase 云计算 iOS6 Rackspace Web App S

Compuware 大数据 aptech Perl Tornado Ruby Hibernate ThinkPHP HBase Pure Solr An

Cloud Foundry Redis Scala Django Bootstrap