Verteilte Systeme FS 13
Übung 3

Thomas Baumann

08. April 2013

1 Beschreibung

Für die Übung 3 habe ich XML-RPC ausgewählt. Dazu habe ich ein flaches Interface *FlatBank* eingeführt, welches die Schnittstelle zwischen Client und Server darstellt und die Interfaces Bank und Konto vereint. Die Methoden der Konten enthalten einen zusätzlichen Parameter mit der Kontonummer.

Wie bereits die vorherigen Übungen, ist diese Lösung nicht Thread-Safe.

2 Code

Listing 1: Interface für XML-RPC

```
1  package bank.xmlrpc;
2
3  import java.io.IOException;
4
5  import bank.InactiveException;
6  import bank.OverdrawException;
7
8  public interface FlatBank {
9
10     String createAccount(String owner) throws IOException;
11
12     boolean closeAccount(String number) throws IOException;
13
14     Object[] getAccountNumbers() throws IOException;
15
16     boolean getAccount(String number) throws IOException;
17
18     Object transfer(String fromNumber, String toNumber, double amount)
19             throws IOException, IllegalArgumentException, OverdrawException,
20             InactiveException;
21
22     String getOwner(String number) throws IOException;
23
24     boolean isActive(String number) throws IOException;
25
26     Object deposit(String number, double amount) throws IOException,
27             IllegalArgumentException, InactiveException;
28
29     Object withdraw(String number, double amount) throws IOException,
30             IllegalArgumentException, OverdrawException, InactiveException;
31
32     double getBalance(String number) throws IOException;
33 }
```

Listing 2: Server Driver

```
1  package bank.xmlrpc;
2
3  import java.io.IOException;
4
5  import org.apache.xmlrpc.XmlRpcException;
6  import org.apache.xmlrpc.server.PropertyHandlerMapping;
7  import org.apache.xmlrpc.server.XmlRpcServer;
8  import org.apache.xmlrpc.server.XmlRpcServerConfigImpl;
9  import org.apache.xmlrpc.webserver.WebServer;
10
11 import bank.IServerDriver;
12 import bank.StartClient;
13
14 public class ServerDriver implements IServerDriver {
15
16     @Override
17     public void start(String[] args) throws IOException {
18         if (args.length < 1) {
19             System.out.println("Usage: java " + StartClient.class.getName() + " "
20                     + ServerDriver.class.getName() + " <portnumber>");
21             System.exit(1);
22         }
23         int port = 0;
24         try {
25             port = Integer.parseInt(args[0]);
26         } catch (NumberFormatException e) {
27             System.out.println("Port must be a number");
28             System.exit(1);
29         }
```

```
30          WebServer webServer = new WebServer(port);
31
32          XmlRpcServer xmlRpcServer = webServer.getXmlRpcServer();
33
34          PropertyHandlerMapping phm = new PropertyHandlerMapping();
35          try {
36              phm.addHandler(bank.xmlrpc.FlatBank.class.getName(),
37                      bank.xmlrpc.ServerBank.class);
38              xmlRpcServer.setHandlerMapping(phm);
39
40              XmlRpcServerConfigImpl serverConfig = (XmlRpcServerConfigImpl) xmlRpcServer
41                      .getConfig();
42              serverConfig.setEnabledForExtensions(true);
43              serverConfig.setEnabledForExceptions(true);
44              serverConfig.setContentLengthOptional(false);
45
46              webServer.start();
47              System.out.println("Server started at port: " + port);
48          } catch (XmlRpcException e) {
49              e.printStackTrace();
50              System.exit(1);
51          }
52      }
53  }
```

Listing 3: Server Bank

```
1  package bank.xmlrpc;
2
3  import java.io.IOException;
4
5  import bank.IAccount;
6  import bank.IBank;
7  import bank.InactiveException;
8  import bank.OverdrawException;
9
10 public class ServerBank implements FlatBank {
11
12     private static final IBank bank = new bank.local.Bank();
13
14     @Override
15     public String createAccount(String owner) throws IOException {
16         return ServerBank.bank.createAccount(owner);
17     }
18
19     @Override
20     public boolean closeAccount(String number) throws IOException {
21         return ServerBank.bank.closeAccount(number);
22     }
23
24     @Override
25     public Object[] getAccountNumbers() throws IOException {
26         return ServerBank.bank.getAccountNumbers().toArray();
27     }
28
29     @Override
30     public boolean getAccount(String number) throws IOException {
31         return ServerBank.bank.getAccount(number) != null;
32     }
33
34     @Override
35     public Object transfer(String fromNumber, String toNumber, double amount)
36             throws IOException, IllegalArgumentException, OverdrawException,
37             InactiveException {
38         IAccount from = ServerBank.bank.getAccount(fromNumber);
39         IAccount to = ServerBank.bank.getAccount(toNumber);
40         ServerBank.bank.transfer(from, to, amount);
41         return null;
42     }
43
```

```
44      @Override
45      public String getOwner(String number) throws IOException {
46          IAccount acc = ServerBank.bank.getAccount(number);
47          if (acc == null) {
48              throw new IOException();
49          } else {
50              return acc.getOwner();
51          }
52      }
53
54      @Override
55      public boolean isActive(String number) throws IOException {
56          IAccount acc = ServerBank.bank.getAccount(number);
57          if (acc == null) {
58              throw new IOException();
59          } else {
60              return acc.isActive();
61          }
62      }
63
64      @Override
65      public Object deposit(String number, double amount) throws IOException,
66              IllegalArgumentException, InactiveException {
67          IAccount acc = ServerBank.bank.getAccount(number);
68          if (acc == null) {
69              throw new IOException();
70          } else {
71              acc.deposit(amount);
72          }
73          return null;
74      }
75
76      @Override
77      public Object withdraw(String number, double amount) throws IOException,
78              IllegalArgumentException, OverdrawException, InactiveException {
79          IAccount acc = ServerBank.bank.getAccount(number);
80          if (acc == null) {
81              throw new IOException();
82          } else {
83              acc.withdraw(amount);
84          }
85          return null;
86      }
87
88      @Override
89      public double getBalance(String number) throws IOException {
90          IAccount acc = ServerBank.bank.getAccount(number);
91          if (acc == null) {
92              throw new IOException();
93          } else {
94              return acc.getBalance();
95          }
96      }
97  }
```

Listing 4: Client Driver

```
1  package bank.xmlrpc;
2
3  import java.io.IOException;
4  import java.net.URL;
5  import java.util.HashSet;
6  import java.util.Set;
7
8  import org.apache.xmlrpc.client.XmlRpcClient;
9  import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;
10 import org.apache.xmlrpc.client.util.ClientFactory;
11
12 import bank.IAccount;
13 import bank.IBank;
```

```java
import bank.IBankDriver;
import bank.InactiveException;
import bank.OverdrawException;
import bank.StartClient;

public class ClientDriver implements IBankDriver {

    private IBank bank;
    private FlatBank flatBank;

    @Override
    public void connect(String[] args) throws IOException {
        if (args.length < 1) {
            System.out.println("Usage: java " + StartClient.class.getName() + " "
                    + ClientDriver.class.getName() + " <server>");
            System.exit(1);
        }
        XmlRpcClientConfigImpl config = new XmlRpcClientConfigImpl();
        config.setServerURL(new URL(args[0]));
        config.setEnabledForExtensions(true);
        config.setEnabledForExceptions(true);
        config.setContentLengthOptional(true);

        XmlRpcClient client = new XmlRpcClient();

        client.setConfig(config);

        ClientFactory factory = new ClientFactory(client);
        this.bank = new Bank();
        this.flatBank = (FlatBank) factory.newInstance(FlatBank.class);
    }

    @Override
    public void disconnect() throws IOException {
        this.bank = null;
        this.flatBank = null;
    }

    @Override
    public IBank getBank() {
        return this.bank;
    }

    public class Bank implements IBank {

        @Override
        public String createAccount(String owner) throws IOException {
            return ClientDriver.this.flatBank.createAccount(owner);
        }

        @Override
        public boolean closeAccount(String number) throws IOException {
            return ClientDriver.this.flatBank.closeAccount(number);
        }

        @Override
        public Set<String> getAccountNumbers() throws IOException {
            Object[] obj = ClientDriver.this.flatBank.getAccountNumbers();
            Set<String> set = new HashSet<>(obj.length);
            for (Object object : obj) {
                set.add((String) object);
            }
            return set;
        }

        @Override
        public IAccount getAccount(String number) throws IOException {
            if (ClientDriver.this.flatBank.getAccount(number)) {
                return new Account(number);
            }
            return null;
        }
```

```
86
87          @Override
88          public void transfer(IAccount a, IAccount b, double amount) throws IOException,
89                  IllegalArgumentException, OverdrawException, InactiveException {
90              ClientDriver.this.flatBank.transfer(a.getNumber(), b.getNumber(), amount);
91          }
92
93          public class Account implements IAccount {
94
95              private String number;
96
97              public Account(String number) {
98                  this.number = number;
99              }
100
101             @Override
102             public String getNumber() throws IOException {
103                 return this.number;
104             }
105
106             @Override
107             public String getOwner() throws IOException {
108                 return ClientDriver.this.flatBank.getOwner(this.number);
109             }
110
111             @Override
112             public boolean isActive() throws IOException {
113                 return ClientDriver.this.flatBank.isActive(this.number);
114             }
115
116             @Override
117             public void deposit(double amount) throws IOException,
118                     IllegalArgumentException, InactiveException {
119                 ClientDriver.this.flatBank.deposit(this.number, amount);
120             }
121
122             @Override
123             public void withdraw(double amount) throws IOException,
124                     IllegalArgumentException, OverdrawException, InactiveException {
125                 ClientDriver.this.flatBank.withdraw(this.number, amount);
126             }
127
128             @Override
129             public double getBalance() throws IOException {
130                 return ClientDriver.this.flatBank.getBalance(this.number);
131             }
132         }
133     }
134 }
```