Java Stored Procedures in FirstSQL

This is a note giving a real-world example of using Java Stored Procedures in FirstSQL/J. For an example, we will use the Money class (Java code is included below). The Money class is for storage of monetary amounts in mixed currencies. It includes a static method - convert(), that is called as a stored procedure. convert() converts a monetary amount from one currency to another.

The Money class is placed in the FirstSQL/J catalog with CREATE CLASS:

```
CREATE CLASS Money FROM 'Money'
```

An object of the Money class has two values - a decimal amount and a currency type string.

The convert() method receives 2 parameters - a Money object and a currency type string. It returns a new Money object for the specified currency with the amount converted to the new currency:

```
public static Money convert(Money source, String currency)
```

The convert() method uses a database table - money, to do the conversion. The money table contains a row for each currency type. Each row has the value in US Dollars of a unit in the specified currency. The SQL commands for creating the money table and adding some example rows are included below.

The SQL command for calling the convert() method as a stored procedure is:

```
{?=call Money.convert(?,?)}
```

The Java code to use the convert() stored procedure:

```
// value to be converted
Money source = new Money(1445.36, "USD");

PreparedStatement prep = conn.prepareStatement("{?=call Money.convert(?,?)}");
// register return value
prep.registerOutParameter(1, Types.OTHER);
// set arguments to convert()
prep.setObject(2, source);
prep.setString(3, "EUR");
// execute call
prep.execute();
// retrieve result
Money target = (Money) prep.getObject(1);
```

```
import java.math.*;
import java.io.*;
import java.sql.*;
import COM.FirstSQL.Dbcp.Database;

// Money.java - FirstSQL/J Example Database Class

public class Money implements Serializable
{
  protected String currency;
  protected BigDecimal amount;
  public Money(BigDecimal a, String c)
  {
    amount = a;
    currency = c;
  }
  public Money(double a, String c)
  {
    this(new BigDecimal(a), c);
  }
  public Money(String a, String c)
  {
    this(new BigDecimal(a), c);
  }
  public String getCurrency()
  {
    return currency;
  }
  public double doubleValue()
  {
    return amount.doubleValue();
  }
  public BigDecimal decValue()
  {
    return amount;
  }

  public String toString()
  {
    return amount.setScale(2, BigDecimal.ROUND_HALF_UP).toString();
  }

  // change a Money object to a new currency
  public static Money convert(Money source, String currency)
  {
    Money target = null;
```

```
    PreparedStatement stnt = null;
    ResultSet rs = null;
    try
    {

      stnt = conn.prepareStatement("SELECT " +
            "? * (SELECT usdollar FROM money WHERE code = ?) / usdollar " +
            "FROM money WHERE code = ?");

      stnt.setBigDecimal(1, source.decValue());
      stnt.setString(2, source.getCurrency());
      stnt.setString(3, currency);
      rs = stnt.executeQuery();
      if (rs.next())
        target = new Money(rs.getBigDecimal(1), currency);
    }
    catch (SQLException ex)
    {
    }
    finally
    {
      if (stnt != null)
        try
        {
          if (rs != null)
            try
            {
              rs.close();
            }
            catch (SQLException ex)
            {
            }
          stnt.close();
        }
        catch (SQLException ex)
        {
        }
    }
   }
   return target;
  }
}
```

==========

money.sql:

```
drop table money
```

```
  country varchar(32),
  name varchar(16),
  usdollar double
)
go
insert into money Values('USD', 'USA', 'Dollar', 1.00)
go
insert into money Values('GBP', 'UK', 'Pound', .68)
go
insert into money Values('EUR', 'European Union', 'Dollar', 1.05)
go
insert into money Values('CAD', 'Canada', 'Dollar', 1.53)
go
insert into money Values('AUD', 'Australia', 'Dollar', 1.75)
go
insert into money Values('JPY', 'Japan', 'Yen', 125.31)
go
```