Table of Contents

Sometimes, you don't have a database accessible, but you need a storage for temporary or session relevant data. In that case, you could use an AbstractMemStorage. This storage stores data in an internal MemDataBook. It fully implements IStorage interface and allows clients to access data via databooks.

Usage

The class is an abstract class and you should extend if if you need it. It is also possible to create an anonymous class. The following example shows usage in the session life cycle object of an application:

```
public IStorage getContacts() throws Exception
{
    AbstractMemStorage amsContacts = (AbstractMemStorage)get("contacts");

    if (amsContacts == null)
    {
        amsContacts = new AbstractMemStorage()
        {
            private int iId = 0;

            @Override
            public RowDefinition getRowDefinition() throws ModelException
            {
                RowDefinition rowdef = new RowDefinition();
                rowdef.addColumnDefinition(new ColumnDefinition("ID",
                                           new BigDecimalDataType()));
                rowdef.addColumnDefinition(new
ColumnDefinition("FIRST_NAME"));
                rowdef.addColumnDefinition(new
ColumnDefinition("LAST_NAME"));
                rowdef.addColumnDefinition(new ColumnDefinition("DOB",
                                           new TimestampDataType()));

                //important, otherwise our client does not receive "hidden"
                //columns like ID
                rowdef.setColumnView(null, new ColumnView("ID",
"FIRST_NAME",
                                                          "LAST_NAME",
"DOB"));

                //important, otherwise our client does not write-back
                rowdef.setPrimaryKeyColumnNames(new String[] {"ID"});

                return rowdef;
            }

            @Override
            public void update(DataBookEvent pEvent) throws ModelException
            {
```

```
            }

            @Override
            public void loadData(MemDataBook pBook, ICondition pFilter)
throws
ModelException
            {
            }

            @Override
            public void insert(DataBookEvent pEvent) throws ModelException
            {
                pEvent.getChangedDataBook().setValue("ID",
BigDecimal.valueOf(iId++));
            }

            @Override
            public void delete(DataBookEvent pEvent) throws ModelException
            {
            }
        };

        amsContacts.open();

        put("contacts", amsContacts);
    }

    return amsContacts;
}
```