

If you need data from a REST service, it's super easy to call the REST service with JVx. The JVx REST services are based on [RESTlet](#) and the library should be available in your application as well. To try it out, simply create a unit test:

ContactClient.java

```
import java.util.List;
import java.util.Map;

import org.junit.Test;
import org.restlet.data.ChallengeScheme;
import org.restlet.resource.ClientResource;

import com.sibvisions.rad.server.http.rest.JSONUtil;
import com.sibvisions.util.type.StringUtil;

public class ContactClient
{
    //~~~~~
    // Test methods
    //~~~~~

    @Test
    public void getContacts() throws Exception
    {
        ClientResource cres = new
ClientResource("https://cloud.sibvisions.com/demoerp/services/rest/Demo
ERP/Customers/data/customer/");
        cres.setChallengeResponse(ChallengeScheme.HTTP_BASIC,
"manager", "manager");

        List<Map<String, Object>> list = JSONUtil.getObject(cres.get(),
List.class);

        for (Map<String, Object> m : list)
        {
            //System.out.println(StringUtil.toString(m.keySet()));

            System.out.println(m.get("ID") + ": " + m.get("FIRSTNAME")
+ " " + m.get("SURNAME"));
        }
    }
}
```

The response is e.g.

```
[
  {
    "ID": 730,
    "COMPANY": "adsad",
```

```
"FIRSTNAME": "alasad",
"SURNAME": "asf test",
"EMAIL": "sadadsaa",
"ADDRESS": "1",
"ZIP": "1",
"CITY": "2",
"CUSTOMERSINCE": "2016-04-30T22:00:00.000+0000",
"TITL_ID": 1,
"TITL_TITLE": "Dr.",
"SALU_ID": 1,
"SALU_SALUTATION": "Male",
"CUSTOMER_NR": "CN - 730"
},
{
  "ID": 634,
  "COMPANY": "helDFF",
  "FIRSTNAME": "Peter",
  "SURNAME": "Fall",
  "EMAIL": "oooo",
  "ADDRESS": "zjr",
  "ZIP": "1235",
  "CITY": "www",
  "CUSTOMERSINCE": "2016-03-09T23:00:00.000+0000",
  "TITL_ID": 1,
  "TITL_TITLE": "Dr.",
  "SALU_ID": 2,
  "SALU_SALUTATION": "Female",
  "CUSTOMER_NR": "CN - 634"
},
{
  "ID": 2,
  "COMPANY": "Excompany",
  "FIRSTNAME": "dddd",
  "SURNAME": "eeee",
  "EMAIL": "susi.summ@excompany.com",
  "ADDRESS": "Street 2",
  "ZIP": "1100",
  "CITY": "Vienna",
  "CUSTOMERSINCE": "2012-12-01T23:00:00.000+0000",
  "TITL_ID": 1,
  "TITL_TITLE": "Dr.",
  "SALU_ID": 1,
  "SALU_SALUTATION": "Male",
  "CUSTOMER_NR": "CN - 2"
},
{
  "ID": 750,
  "COMPANY": "333",
  "FIRSTNAME": "abcefe",
  "SURNAME": "555",
  "EMAIL": "Schider.ngai@somehost.com",
```

```
"ADDRESS":"Far Far Away",
"ZIP":"55555",
"CITY":"zxcvasd",
"CUSTOMERSINCE":null,
"TITL_ID":1,
"TITL_TITLE":"Dr.",
"SALU_ID":1,
"SALU_SALUTATION":"Male",
"CUSTOMER_NR":"CN - 750"
}
]
```

You could integrate the REST call in your business logic, see [Calling a server-side action](#). If you need the result in your GUI, it's also possible to create a custom storage. We use a simple [statful storage](#) for our example, but you could also implement a stateless storage. Here's the storage:

ContactStorage.java

```
public class ContactStorage extends AbstractMemStorage
{
    //~~~~~
    // Initialization
    //~~~~~

    /**
     * Creates a new instance of <code>ContactStorage</code>.
     */
    public ContactStorage()
    {
    }

    //~~~~~
    // Abstract methods implementation
    //~~~~~

    @Override
    public RowDefinition getRowDefinition() throws ModelException
    {
        RowDefinition rdef = new RowDefinition();
        rdef.addColumnDefinition(new ColumnDefinition("ID", new
BigDecimalDataType()));
        rdef.addColumnDefinition(new ColumnDefinition("FIRST"));
        rdef.addColumnDefinition(new ColumnDefinition("LAST"));

        rdef.setPrimaryKeyColumnNames(new String[] {"ID"});

        rdef.setColumnView(null, new ColumnView(rdef));

        return rdef;
    }
}
```

```
@Override
public void loadData(MemDataBook pBook, ICondition pCondition)
throws ModelException
{
    pBook.close();
    pBook.open();

    ClientResource cres = new
ClientResource("https://cloud.sibvisions.com/demoerp/services/rest/Demo
ERP/Customers/data/customer/");
    cres.setChallengeResponse(ChallengeScheme.HTTP_BASIC,
"manager", "manager");

    try
    {
        List<Map<String, Object>> list =
JSONUtil.getObject(cres.get(), List.class);

        for (Map<String, Object> m : list)
        {
            pBook.insert(false);
            pBook.setValues(new String[] {"ID", "FIRST", "LAST"},
                new Object[] {new
BigDecimal((BigInteger)m.get("ID")), m.get("FIRSTNAME"),
m.get("SURNAME")} );
        }

        pBook.saveAllRows();
    }
    catch (IOException ioe)
    {
        throw new ModelException(ioe);
    }
}

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

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

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

```
}
```

Our storage doesn't support insert/update/delete/filtering/sort. Only fetching data is implemented. But if you use client filtering/sort, it will work well.

The storage is ready to use with life-cycle objects:

Contacts.java

```
public IStorage getContacts() throws Exception
{
    ContactStorage storage = (ContactStorage)get("contacts");

    if (storage == null)
    {
        storage = new ContactStorage();
        storage.open();

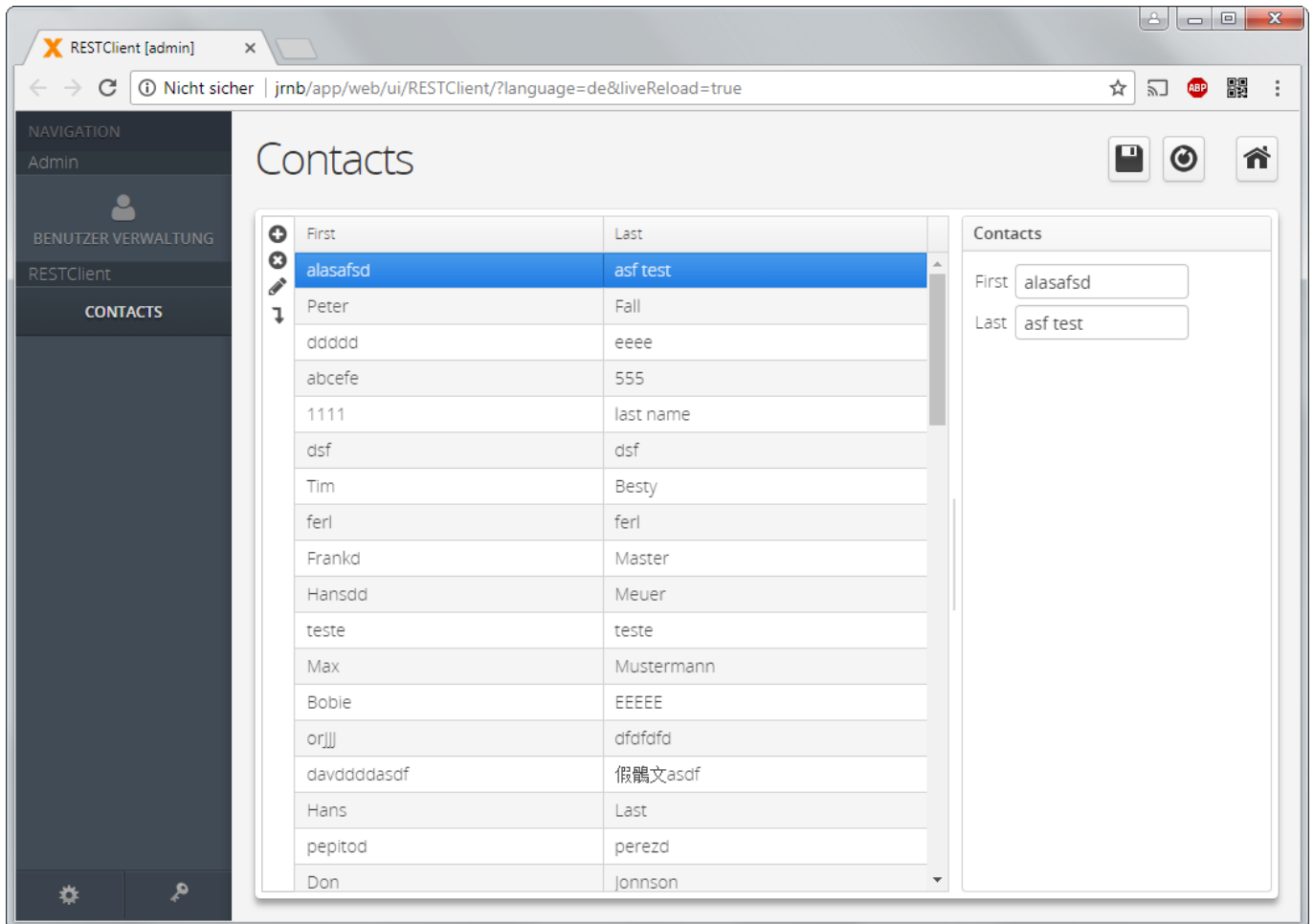
        put("contacts", storage);
    }

    return storage;
}
```

And on the client side, use a RemoteDataBook as usual:

```
RemoteDataBook rdbContacts = new RemoteDataBook();
rdbContacts.setName("contacts");
rdbContacts.setDataSource(getDataSource());
//because our storage doesn't support remote filtering/sort
rdbContacts.setMemFilter(true);
rdbContacts.setMemSort(true);
rdbContacts.open();
```

And here's an example application:



Note

It's also possible to use other REST libraries, like:

- <https://howtoprogram.xyz/java-technologies/java-rest-client-example/>
- <https://stackoverflow.com/questions/221442/rest-clients-for-java>

Using RESTlet may be an option because it's used from JvX and thanks to JSONUtil it's easy to access the JSON object from the Response.

From:
<http://doc.sibvisions.com/> - **Documentation**

Permanent link:
http://doc.sibvisions.com/jvx/rest_storage_client

Last update: **2018/03/09 12:52**

