Abstract

In a modern world where web is becoming increasingly significant personally and professionally, more and more, web applications are replacing old-fashioned software installed on computer. Main benefits are an easy and a common way to access for end-users and the possibility to externalise your IT in dedicated infrastructure like cloud computing. With that in mind and focusing on study biological programming needs and how it could be possible to offer full services through web technologies, an important first question will have to be raised: What are the technical possibilities to render a SAS dataset in a web browser? Is that even possible?

The aim of this poster is to answer to this question, analyze different ways to retrieve information from your SAS datasets. We decided to use Java technology to read SAS datasets. Why Java? Because it is a polyvalent programming language, very popular for server side management when it comes to work on mobile applications development. Moreover, there are a lot of bridges between SAS and Java, like Java material provided by SAS (jar files) or SAS Java library exposed as web services.

Challenge is to be able to read SAS dataset structure and data through Java and store that in Java objects.

SASSHARE software is a data server that allows multiple users to gain simultaneous access to SAS files.

SAS/SHARE in your IT environment. Some use cases:

- SAS provides Java Database Connectivity (JDBC) drivers to connect and interacts with SAS datasets stored on SAS/SHARE servers.
- Using the com.sas.soa or SassyReader Java class, you can create a connection to your SAS/SHARE server and use SQL queries to retrieve information from your SAS datasets.
- With access to the complete features of JDBC, it is easy to create java programs to store SAS dataset data in Java objects.

Parso is a high-level parser library that is used to parse SAS dataset files. It offers a lot of functionalities to access to the SAS dataset properties like row and columns count, DS type, compression method, SAS release version, ... To read the rows, you have two main methods:
- Read all rows at the same time and store everything in Java objects
- Read each row one by one

In case you’re working on big datasets (more than 1 GB), you could have troubles with Java memory if you try to read all rows in one step.

We have successfully tested Parso with datasets generated from different DS and from several SAS Releases.

Pascal Verrecchia

pascal.verrecchia@sokaris.fr

Sokaris Informatique

Explore ways to render SAS datasets through Web

Parsy is a Java lightweight library to parse SAS dataset files.

It offers a lot of functionalities to access to the SAS dataset properties like row and columns count, DS type, compression methods, SAS Release version, ...

To read the rows, you have two main methods:
- Read all rows at the same time and store everything in Java objects
- Read each row one by one

In case you’re working on big datasets (more than 1 GB), you could have troubles with Java memory if you try to read all rows in one step.

We have successfully tested Parso with datasets generated from different DS and from several SAS Releases.

Example of Java code using Parsy:

```java
import org.parso.sas.SASReader;
import org.parso.sas.SASRow;
import org.parso.sas.SASData;
import org.parso.sas.SASColumn;
import org.parso.sas.SASDataset;
import org.parso.sas.SASException;

public class SASReaderExample {
    public static void main(String[] args) throws SASException {
        SASReader reader = new SASReader();
        SASDataset dataset = reader.readSASFile("data.sas7bdat");
        SASRow row = dataset.getRow(0);
        System.out.println("Dataset Name: " + dataset.getName());
        System.out.println("Rows: " + row.getNumberOfRows());
    }
}
```

SassyReader is an Open source Java library that simply offer a way of reading the data that is stored in SAS dataset files.

It is a portage of the SAS/HBIF project that has been implemented in Java - though not a unique interface, you can precisely define what your want to retrieve from the SAS dataset: structure, columns, type, raw data, ...

Unfortunately, SassyReader was not working correctly when we test it on multiple recent SAS datasets.

Example of Java code using SassyReader:

```java
import org.sassyreader.sas.SASReader;
import org.sassyreader.sas.SASRow;
import org.sassyreader.sas.SASData;
import org.sassyreader.sas.SASColumn;
import org.sassyreader.sas.SASDataset;
import org.sassyreader.sas.SASException;

public class SassyReaderExample {
    public static void main(String[] args) throws SASException {
        SASReader reader = new SASReader();
        SASDataset dataset = reader.readSASFile("data.sas7bdat");
        SASRow row = dataset.getRow(0);
        System.out.println("Dataset Name: " + dataset.getName());
        System.out.println("Rows: " + row.getNumberOfRows());
    }
}
```

JSON

When SAS datasets data are stored in JSON, it will be necessary to communicate with client side of your web application (mostly for front three data and display it). Most of the web application frameworks are new implementations using JSON as a data transport framework (like Angular). The most common way to do that is to convert your SAS data into JSON (JavaScript Object Notation) format, which is an open standard format pretty well adapted for JavaScript environment. There are multiple ways to convert your data from Java to JSON, like Jackson library.

Example of Java code using JDBC connection on SASSHARE:

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class SASJDBCExample {
    public static void main(String[] args) {
        try {
            Class.forName("com.sas.net.sharenet.ShareNetDriver");
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM my_table");
            while (rs.next()) {
                System.out.println("Column1: " + rs.getString(1) + ", Column2: " + rs.getString(2));
            }
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Example of Java code using JSON conversion:

```java
import com.fasterxml.jackson.databind.ObjectMapper;
import java.util.HashMap;
import java.util.Map;

public class SASJSONExample {
    public static void main(String[] args) {
        try {
            ObjectMapper mapper = new ObjectMapper();
            Map<String, Object> data = new HashMap<String, Object>();
            data.put("column1", "value1");
            data.put("column2", "value2");
            String json = mapper.writeValueAsString(data);
            System.out.println(json);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Example of Java code using Parsy reading operation:

```java
import org.parso.sas.SASReader;
import org.parso.sas.SASRow;
import org.parso.sas.SASData;
import org.parso.sas.SASColumn;
import org.parso.sas.SASDataset;
import org.parso.sas.SASException;

public class ParsyExample {
    public static void main(String[] args) throws SASException {
        SASReader reader = new SASReader();
        SASDataset dataset = reader.readSASFile("data.sas7bdat");
        SASRow row = dataset.getRow(0);
        System.out.println("Dataset Name: " + dataset.getName());
        System.out.println("Rows: " + row.getNumberOfRows());
    }
}
```

Example of Java code using SassyReader reading operation:

```java
import org.sassyreader.sas.SASReader;
import org.sassyreader.sas.SASRow;
import org.sassyreader.sas.SASData;
import org.sassyreader.sas.SASColumn;
import org.sassyreader.sas.SASDataset;
import org.sassyreader.sas.SASException;

public class SassyReaderExample {
    public static void main(String[] args) throws SASException {
        SASReader reader = new SASReader();
        SASDataset dataset = reader.readSASFile("data.sas7bdat");
        SASRow row = dataset.getRow(0);
        System.out.println("Dataset Name: " + dataset.getName());
        System.out.println("Rows: " + row.getNumberOfRows());
    }
}
```