Paper 31-27

Accessing the Version 9 OLAP Server from Java

Jeff Diamond, SAS Institute Inc., Cary, NC
Tammy Gagliano, SAS Institute Inc., Carmel, IN

ABSTRACT

Significant development resources have been put into the area of OLAP data access with performance and scalability being two of the most influential factors considered when designing the Version 9 architecture. The Version 9 SAS® OLAP Server uses the SAS Scalable Architecture to achieve these results. Along with a much improved back-end for creating, storing, and accessing your multidimensional data, a new set of Java-based model and viewer classes have also been developed to exploit this new functionality from both applet and JSP/servlet-based web applications. This paper will focus on those front-end components – the Java classes you can use to web enable your OLAP applications in Version 9. These Java components will be released as experimental with Version 3 of AppDev Studio™, and the details are subject to change.

A greatly expanded version of this paper will be made available on the web by SUGI. It will contain details on all the topics outlined here. You can find the final version of this paper under the Reference link on the AppDev Studio Developer's Site (http://www.sas.com/rnd/appdev).

INTRODUCTION

OLAP (Online Analytical Processing) continues to be an essential technology for exploring and discovering data. It allows the data to be viewed from every possible angle, exposing those golden nuggets that are key to intelligent decision making. The Version 9 SAS OLAP Server was written to make this technology faster and more scaleable. Also, it was written to more closely follow industry standards and emphasize open architecture. With these improvements comes a need to access the new OLAP Server across many different environments. This paper will focus on accessing the OLAP server from Java.

To this end, SAS has developed a new set of Java-based components that help describe, view, and navigate an enterprise's OLAP data. These components can be categorized into three groups, Model, View, and Controller, and together they form the three points of the popular MVC architecture.

The model describes the enterprise data. There are three major components that are categorized as model components, the data model, the metadata model, and the query model. The data model, which is called the OLAPDataSetInterface, is comparable to a JDBC RowSet. It encompasses the connection to the OLAP Server, the ability to execute a query on that server, and the ResultSet representing the slice of data returned from the server. The second component, the query model, extends the functionality of the data model in the sense that it builds the query. The data model uses MDX (Multidimensional Expressions) as its query language, and so this is what is built by the query component. MDX can be thought of as the OLAP equivalent to SQL for relational data. MDX can be complicated to write by hand, so it is not required that users be able to write their own MDX. The query model itself is not a visual component; however, there are a number of visual components called "selectors" that assist the user in building an MDX query. The data model coupled with the query model support the basic operations of reading data from the cube. The last model component is the Metadata object. This object allows the user to query the cube for its metadata such as the Dimensions, Hierarchies, Levels, Members, etc. These elements are distinctly different from the data model in that they represent the structure of the underlying cube, not the slice of data returned from that cube. The metadata elements may be used to visualize the organization of the data or to formulate a query.

The view renders the data in the model. Viewers can be written for any of the models. For example, the "selectors" mentioned above can be thought of as a view of the query model. However, the discussion of the view components will concentrate on those visual components that render the contents of the data model. Such viewers are available in both the applet and JSP/servlet-based environments, and the set of components available in each environment mirror one another. One common type of viewer is a table, which is called the OLAPTableView. There are also several graph viewers, for example: a bar chart and a pie chart. In addition to rendering the data these viewers also support various OLAP actions such as drilling or defining a Top N criteria via a selector. When interacting with the visual component through these actions such as drill or expand, the necessary information is passed along to the controller so that the action can be processed accordingly. In this case, the query model uses the information to regenerate an MDX query that represents the state of the data after the action has taken place. The new results are then displayed in the viewer.

The role of the controller in this architecture is two fold. It processes the user actions on the view and translates these to the appropriate methods for executing the command. It also manages the actions that are available on the view based on the context of the user's action. In this way the view remains focused on rendering the data. But what does this mean for you, the application writer. It means the actions are handled in a very general way and in a centralized location. This allows custom actions to be easily added without having to rewrite or extend numerous viewers. The standard OLAP controller supports a robust set of OLAP functionality, so to the more basic OLAP application all of this is virtually invisible. The group of components that surface this functionality is known as the Action Provider Framework. The core of this framework is not specific to OLAP; however, its classes are integral when navigating through OLAP data.

CONCLUSION

From canned reports to complex "what-if" analysis OLAP has become integral to any business intelligence solution. This suite of components, included in Version 3 of AppDev Studio, gives application developers and report writers the tools to include OLAP as part of their solution.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jeff Diamond
SAS Institute Inc.
Work Phone: (919) 531-6099
Email: Jeff.Diamond@sas.com

Tammy Gagliano
SAS Institute Inc.
Work Phone: (317) 569-9598
Email: Tammy.Gagliano@sas.com