Vertica Knowledge Base Article

Developing UDxs in Java Tutorial Part 1: Introduction and Setting Up

Document Release Date: 10/25/2018

MICRO FOCUS | VERTICA

Legal Notices

Warranty

The only warranties for Micro Focus International plc products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from Micro Focus required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Copyright Notice

Trademark Notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

Contents

Developing UDxs in Java Tutorial Part 1: Introduction and Setting Up

Welcome to the first part of our tutorial series on developing User Defined Extensions (UDxs) with the Java programming language. In this part, you'll learn what UDxs are and why you would want to develop one. We'll also show you how to set up a development environment to make developing your UDxs easier.

Download this article in PDF format.

What is a UDx?

User Defined Extensions let you add your own features to Vertica. They help you analyze and transform your database's data in ways that are difficult or impossible to do using SQL alone.

UDxs are broken down into two broad categories:

- User Defined Functions (UDFs) provide functions that you can call from within your SQL statements (usually, a SELECT statement).
- User Defined Loads (UDLs) let you replace one or more steps in the data load process.

Each category has several types of UDxs. Each type of UDx fills a specific role in processing your data. The future installments of this tutorial series explain how you develop and use each type of UDx.

Why Create a UDx?

If you've ever found yourself thinking "if I could just do *X* with my data…" then you may want to create a UDx. (Well, for many values of *X*, at least.) For example:

- Load data from a data source other than a flat file or stream. For example, load data from a web API or extract it from PDF documents. If you can write code to extract the data, you can load it into Vertica.
- Load data compressed using an unsupported compression format.
- Load data encoded with an unsupported character encoding. (Finally! A way to process all of that data in EBCDIC format you have lying around on punch cards!)
- Transform data in one table into a completely different format. For example, extract words from a VARCHAR column and turn them into a word cloud based on their frequency.
- Use a programming library that performs the data analytics you want to run on your data stored in Vertica.

How Do You Create a UDx?

You develop UDxs in one of three programming languages: C++, Java, or R. Vertica provides a Software Development Kit (SDK) for each of these languages. It reduces the complexity of creating a distributed data processing or analytic function. Instead of worrying about organizing and processing the data, all your code does is process blocks of data that Vertica sends to it and return the results.

What Do These Tutorials Cover?

These tutorials explain how to develop UDxs using Java. The topics include:

- Developing each type of UDx supported in the Vertica Java SDK:
  - User Defined Scalar Functions
  - User Defined Transforms
  - User Defined Sources
  - User Defined Filters
  - User Defined Parsers
- Handling different numbers of arguments
- Handling parameters
- Debugging your Java UDx
- Additional features being introduced in Vertica 7.2

What Do You Need to Know?

These tutorials assume:

- You know basics of using Vertica . For example: how to create tables and run queries. If you are familiar with other SQL databases, you should be fine.
- You know how to program in Java.
- You are familiar with Eclipse. These tutorials show you how to use the Vertica Java SDK plugin for Eclipse to save you time and effort when creating UDxs. It creates templates for your UDx code.

Setting Up Your Development Environment

Before you start developing UDxs, you need to set up a development environment. You can develop UDxs on any platform that supports Java. However, developing your UDx on a Linux system that has Vertica installed makes testing your UDx easier. If you choose to develop on a non-Linux system, you must transfer your compiled UDx to an Vertica database in order to test it. These tutorials assume you are developing UDxs on a Linux system on which you have also installed a single-node Vertica database.

> Important Never use your production Vertica database as a development platform for UDxs. Because Java UDxs run in a sandbox, they cannot corrupt or crash your database. However, bugs in your UDx can still consume enough RAM and CPU to cause

> performance issues in your Vertica cluster.

If you do not have a physical Linux system to use as a development system, you can create a virtual development system using virtualization software such as VMWare. The demonstration videos in this tutorial series were recorded on a virtual Ubuntu Linux system running on a Windows host. If you would like to set up a similar system, you could start with the pre-configured CentOS-based Vertica virtual machine. It is available for download at https://www.vertica.com/download/vertica/community-edition/ free registration required). You can also create your own VM and then install Vertica using a community license.

Whether you choose to develop on a physical or virtual system, your development environment must meet the following requirements:

- It must run a version of Linux supported by Vertica . See the Vertica Supported Platforms guide for details.
- It must have sufficient resources to run both Vertica and Eclipse. Your development system should have at least 8GB of memory and two processor cores.
- It must have a supported Java Development Kit (JDK) installed. Vertica supports Oracle Java Platform Standard Edition and OpenJDK. Be sure to install version 6 or 7.
- You must configure your single-node Vertica database to enable Java UDxs. See Installing Java on Hosts in the Vertica documentation to learn how to configure Vertica to enable Java UDxs.

In addition to your development system, you should also have a test cluster. Before you deploy your UDx to a production database, always perform further testing on a multi-node test cluster. If you do not have that hardware available for a physical test cluster, you can create a virtual three-node cluster using an Vertica Community license.

To make developing even easier, consider developing your UDxs while logged into your development Linux system as the dbadmin user. This ensures that you have the permissions necessary to deploy your UDx. Another option is to add a user to your test Vertica database whose name matches your Linux user name. Then grant the Vertica user superuser privileges.

About Eclipse and the Vertica Eclipse Plugin

These tutorials assume you are using the the Eclipse Foundation's Eclipse IDE to develop Java UDxs. To help you develop UDxs, Vertica has developed the Vertica Java SDK Plugin for Eclipse. This plugin:

- Automates the setup of UDx projects. It links the project to the Vertica SDK. You can start coding immediately.
- Contains the Vertica Java SDK documentation. As you edit your UDx code, Eclipse shows you context-sensitive help and offers to autocomplete class and method names for you.
- Has wizards to generate the basic code for your UDx function. It automates a lot of the tedious boilerplate code for you, so you can start coding your UDx immediately.

The Vertica Java SDK Plugin for Eclipse supports Eclipse version 4.4.0 (“Juno”).

> Note Many Linux distributions let you install Eclipse using their package management system. However, some of them (such as Ubuntu) install an out-of-date version of Eclipse. To ensure your version of Eclipse is compatible with the Vertica Java SDK Plugin for Eclipse, consider directly downloading and installing Eclipse from the Eclipse website.

Installing the Vertica Java SDK Plugin for Eclipse

Several versions of the Vertica Java SDK Plugin are attached to this wiki article. Each plugin version corresponds to a version of Vertica . The plugins contain the Vertica Java SDK library and documentation that support a single version of Vertica.

To install the plugin:

1. Download the version of the plugin for the version of Vertica you are running.
2. Exit Eclipse if it is currently running.
3. Extract the .zip file into your Eclipse directory’s `dropins` directory. If your copy of Eclipse is installed system-wide (such as in `/opt/eclipse`) you must have root privileges to extract the files in the directory.
4. Start Eclipse.
5. To verify that the plugin works, on the **File** menu, point to **New** and click **Other…**.
6. Verify that the **Wizards** list contains an entry named Vertica ***x.x.x*** **Java SDK** (where *x.x.x* is your version of Vertica).

Once you have installed the plugin, you are ready to develop UDxs!

What's Next?

In the next tutorial, we'll learn how to develop a User-Defined Scalar Function.

Developing UDxs in Java Tutorial Part 1: Introduction and Setting Up