PostgreSQL Primer

This PostgreSQL Primer is by no means a replacement for the excellent PostgreSQL documentation.

The goal of this primer is to provide a starting point for the most common actions for working with PostgreSQL, with pointers to more in-depth documentation where applicable.

In This Chapter

- Servoy & PostgreSQL
- Installing PostgreSQL
    - Embedded PostgreSQL database engine
    - EnterpriseDB PostgreSQL Server
    - Which installation to choose
- Upgrading PostgreSQL
    - Embedded PostgreSQL database engine
    - EnterpriseDB installation
- Starting and stopping PostgreSQL manually
- Troubleshooting
- Running PostgreSQL as a Service
    - Windows
    - OSX
    - *nix
- Performing backups & restores
    - Manual backups
- PostgreSQL Database Admin tool
- PostgreSQL SQL Tips 'n' Tricks
    - No automatic type casting
    - Quoting strings and aliases
- Migrating to PostgreSQL

Servoy & PostgreSQL

As of Servoy 5.2 Servoy comes bundled with PostgreSQL. While Servoy comes bundled with PostgreSQL, Servoy is 100% database agnostic. The PostgreSQL database can therefor be replaced by any type of database.

Bundled PostgreSQL version

Which version of PostgreSQL is shipped with a specific version of Servoy can be found in the Servoy documentation for the specific Servoy version. For the latest version, see the Servoy Stack Info here.

Installing PostgreSQL

As of Servoy 5.2 Servoy comes bundles with PostgreSQL. When selecting the *Database* option in the All-In-One installer, PostgreSQL is automatically installed and Servoy is setup to connect to it.

As of Servoy 6.0 the Servoy All-In-One installer provides 2 different options to install PostgreSQL:

- Install the embedded PostgreSQL installation
- Download and install the PostgreSQL Server installer

When using Servoy 5.2, the PostgreSQL Server installer can be manually downloaded from the EnterpriseDB website here.

As of Servoy 2019.06 Servoy comes only as an archive installer which has Java and PostgreSQL already installed.

Embedded PostgreSQL database engine

Compared to the EnterpriseDB PostgreSQL Server the embedded PostgreSQL database engine is a no frills PostgreSQL installation, just just the bare minimal options installed to run, backup and upgrade the PostgreSQL database.

The PostgreSQL database engine is installed in the *../application_server/postgres_db/* directory. The actual databases and configuration files are stored in the *../application_server/database/* directory. As of Servoy 6.0, the location of the database engine and database files can be configured during installation.

As of Servoy 2019.06 Servoy comes only as archive installers which has PostgreSQL already installed for you.

EnterpriseDB PostgreSQL Server

The PostgreSQL Server installer of EnterpriseDB adds to enterprise level tools to PostgreSQL, like replication. Besides that the Standard Server installer is more sophisticated than the Servoy installer. It will for example install the PostgreSQL database under it's own user.

As of PostgreSQL 11, EnterpriseDB doesn't provide Linux installers anymore. The PostgreSQL community maintains repositories for several Linux flavours:

Debian and Ubuntu: https://wiki.postgresql.org/wiki/Apt

Centos, RHEL, Fedora and Oracle Enterprise Linux: https://wiki.postgresql.org/wiki/YUM_Installation

NOTE: EnterpriseDB is a commercial party with its own (commercial) version of PostgreSQL, they also create and host the community (FOSS) installers of PostgreSQL. In this document we are always referring to the community installers.

Which installation to choose

For production purposes, it's advised to use the PostgreSQL Server installer of EnterpriseDB as these come with all the security best practices out of the box. For development and test purposes, both the embedded PostgreSQL database engine and the EnterpriseDB option would be sufficient.

Upgrading PostgreSQL

Embedded PostgreSQL database engine

To be described

EnterpriseDB installation

Updating an PostgreSQL Server installation from EDB to the next maintenance release, let's say 11.1 to 11.2, can be done by running the 11.2 installer. It will automatically find the 11.1 installation and update it. Make sure to stop Servoy (and any other client connecting to PostgreSQL) performing the upgrade.

Upgrading major versions, let's say from 11.x to 12.x is not possible in the same manner. Running the 12.x installer will install 12.x next to the 11.x installation. The installer automatically set the port of this version to one that doesn't collide with the one already installed. So the 2 versions can run at the same time.

To get the databases over to the new major version a backup can be made of the entire database cluster using the command-line *pg_dumpall* backup utility, after which the backup can be loaded back into the new instance using the *psql* utility. This operation an also be performed using *PgAdmin*. It is recommended to use the tools from the newer version to backup the databases from the older version.

See the backup section on this page for more information on how to do backups and restores.

Starting and stopping PostgreSQL manually

Through the pg_ctl tool that comes with PostgreSQL, it is easy to start and stop a PostgreSQL database. In a default installation the pg_ctl tool is located in

```
../application_server/postgres_db/bin/
```

When you have used the community installer from EnterpriseDB (EDB) then all of PostgreSQL (binary and database cluster) is installed in the following directory:

OSX

```
/Library/PostgreSQL/12.0/
```

Windows

```
C:\Program Files\PostgreSQL\12.0\
```

Linux

```
/opt/PostgreSQL/12.0/
```

Where the subdirectory *bin* holds the command-line tools and other binaries and the subdirectory *data* holds the database cluster. Also PostgreSQL will be running under the system user *postgres* and the *data* directory is only accesible by that user. All this is for security.

On linux however you can also use ready made packages specially made for the Linux distribution and have their own places of storing the various parts of PostgreSQL.

For example on Centos/RHEL you will find the database cluster in

```
/var/lib/pgsql/12/data/
```

and the binary in

```
/usr/pgsql-12/bin/
```

Starting PostgreSQL

Using the bundled PostgreSQL (assuming you run this from within the application_server directory):

```
postgres_db\bin\pg_ctl start -D database -l postgres_db\postgres_log.txt
```

When using the EDB installed version of PostgreSQL then the command is the following:

```
sudo -u postgres /Library/PostgreSQL/12.0/bin/pg_ctl start -D /Library/PostgreSQL/12.0/data
```

The EnterpriseDB installed version already writes logs in the *data/pg_log/* directory so no need to add the -l argument.

Stopping PostgreSQL

Using the bundled PostgreSQL (assuming you run this from within the application_server directory):

```
postgres_db\bin\pg_ctl stop -D database
```

When using the EnterpriseDB installed version of PostgreSQL then the command is the following:

```
 sudo -u postgres /Library/PostgreSQL/12.0/bin/pg_ctl stop -D /Library/PostgreSQL/12.0/data
```

You need to provide the location of the database cluster because you could have instances of PostgreSQL running using different data directories.

Troubleshooting

After you installed Servoy and PostgreSQL you might find that PostgreSQL is not running.

This usually could mean 2 things:

- PostgreSQL wasn't yet initialised. In other words there is no cluster created yet to host the databases. This means that the *database* or *data* directory is empty.
- There is another PostgreSQL instance running on port 5432 (PostgreSQL's standard port)

Initialise the database cluster

Using the bundled PostgreSQL (assuming you run this from within the application_server directory):

```
postgres_db\bin\pg_ctl init -D database
```

When using the EnterpriseDB installed version of PostgreSQL then the command is the following:

```
 sudo -u postgres /Library/PostgreSQL/12.0/bin/pg_ctl init -D /Library/PostgreSQL/12.0/data
```

Changing the port number

The port number is setup in the postgresql.conf file. This file is located in PostgreSQL's data directory.

Using the bundled PostgreSQL you'll find it here

```
application_server/database/postgresql.conf
```

When using the EnterpriseDB installed version of PostgreSQL you will find it here:

```
/Library/PostgreSQL/12.0/data/postgresql.conf
```

You can edit it with your favourite text editor and change the following line (by default it's a commented line)

```
#port = 5432
```

to this

```
port = 5433
```

IMPORTANT: After you changed the port number make sure your database connections use this port number as well.

Running PostgreSQL as a Service

Windows

The pg_ctl utility that comes with PostgreSQL has a built-in command to register the PostgreSQL database as a Windows Service

Registering the Windows Service:

```
postgres_db\bin\pg_ctl register -N PostgreSQL -D database
```

Unregistering the Windows Service:

```
postgres_db\bin\pg_ctl unregister -N PostgreSQL
```

The commands above assume execution from the *../application_server/* directory. When executing from a different location, make sure to update the paths to the database (-D database).

The *-N* option specifies the name for the Windows Service. Make sure the name is unique.

When also running the Servoy Application Server using the Service Component, a dependency can be setup between the Windows Service for the database and the Application Server. See Running the Server As a Service for more information.

OSX

See https://www.postgresql.org/docs/current/server-start.html

*nix

See https://www.postgresql.org/docs/current/server-start.html

Performing backups & restores

PostgreSQL stores it's data in a 'database cluster' which is a collection of files and directories that together make up the entire database. It's not recommended to backup these files, but instead use the backup utilities that come with PostgreSQL.

Manual backups

PostgreSQL provides a pg_dump and a pg_dumpall utility to make a backup file of a running database. In a default installation of Servoy, the utilities are located in *../application_server/postgres_db/bin/*. These utilities can be used command-line, but also through the PgAdmin utility.

Backup a specific database

```
postgres_db\bin\pg_dump -U dba -Fc servoy_repository > servoy_repository.dump
```

Restore dump into a newly created database

```
postgres_db\bin\pg_restore -U dba -d new_servoy_repository  servoy_repository.dump
```

Backup all databases

```
postgres_db\bin\pg_dumpall -U dba -f db.out
```

Reload all database dump

```
postgres_db\bin\psql -U dba -f db.out postgres
```

Note that the examples above are very minimalistic. The utilities provided by PostgreSQL offer many options. Check the PostgreSQL documentation on pg_dump, pg_dumpall, pg_restore and pg_sql for detailed information.

Also be aware that if you have PostgreSQL running on a non-default port (default is 5432) then you need to add *-p <portnr>* argument.

All the PostgreSQL command-line tools have an extensive help build in. You can access it by using the *--help* argument.

The samples above assume execution from the *../application_server/* directory.

PostgreSQL Database Admin tool

For PostgreSQL there are multiple admin tools available, from commercial to open source. The most used admin tool for PostgreSQL is the open source, cross-platform PgAdmin.

PgAdmin can be downloaded here and the excellent documentation is available here.

The EnterpriseDB installer for PostgreSQL has the PgAdmin tool bundled.

PostgreSQL SQL Tips 'n' Tricks

No automatic type casting

PostgreSQL doesn't do automatic casting of types. This means that when sending in parameters into a prepared statement, it is up to the developer to make sure to send in the right type.

For example:

```
var value = 1;
var query = "SELECT * FROM table WHERE textcolumn = ?"
var args = new Array();
args[0] = value;
var dataset = databaseManager.getDataSetByQuery(controller.getServerName(), query, args, maxReturnedRows);
```

Has to become:

```
var value = 1;
var query = "SELECT * FROM table WHERE textcolumn = ?"
var args = new Array();
args[0] = value + ""; //forcing the integer value to become a string
var dataset = databaseManager.getDataSetByQuery(controller.getServerName(), query, args, maxReturnedRows);
```

Quoting strings and aliases

In PostgreSQL, the use of quotes and double quotes is more strict than in some other databases.

Quoting for column Aliasses:

```
SELECT some_column as "A nice name" FROM table
```

Quoting literal strings:

```
SELECT * FROM table where text_column = 'someValue'
```

Migrating to PostgreSQL

A good starting point to data migration towards PostgreSQL is the PostgreSQL wiki, which has a lot of tutorials migrating data into postgreSQL from many different source databases.

When migration an existing solution developed in Servoy from one type of database to PostgreSQL, Servoy's built-in mechanism to generate the required data model in a new database and to move sample data can also be used to migrate to PostgreSQL (or any other type of database). There are however some restrictions. Such a migration will not move views, stored procedures and functions, database level triggers and sequences. If the amount of records to be moved is very large, memory issues can occur due to the sheer amount of data. To make sure that PK-FK integrity stays in tact in import of the sample data, either Servoy Sequences must be used or the database level generation of sequence values (especially for PK sequences) must be temporarily disabled.

When the amount of data to be moved to the new database is massive or required pre-processing, a better solution is to dump the data to a generic format and import it directly into the new database.