The Servoy Beginner's Handbook

by

Adrian McGilly Information Systems Consultant

McGilly Information Systems, Inc.
Servoy Training & Consulting
San Francisco, CA, USA

Website: www.mcgilly.com
E-mail: info@mcgilly.com
Tel.: 510-428-1035

The Servoy Beginner's Handbook

by

Adrian McGilly Information Systems Consultant

McGilly Information Systems, Inc.
Servoy Training & Consulting
San Francisco, CA, USA

Website: www.mcgilly.com
E-mail: info@mcgilly.com
Tel.: 510-428-1035

Edition: 1.5.2.2

The Servoy Beginner's Handbook

by

Adrian McGilly Information Systems Consultant

Servoy Training & Consulting
San Francisco, CA, USA
Website: www.mcgilly.com
E-mail: info@mcgilly.com

v 1.5.2.2, last updated October 2009

This document is a preview of the Servoy Beginner's Handbook. Below is the full Table of Contents, followed by the first 7 chapters. The complete handbook is 168 pages long. You can purchase and download it from the McGilly Information Systems website for $49 US. Purchase entitles you to free updates to the handbook for 18 months.

Table of Contents

Introduction

Servoy is a brilliant development environment that raises developer productivity to new heights. It leverages today's most powerful technology standards (Java, JavaScript, JDBC, RMI, AJAX, SQL Databases, application server technology, browser technology, CSS, XML, etc.) elegantly and efficiently in a way that empowers developers like never before.

My name is Adrian McGilly and I am a Servoy consultant and trainer, and a member of the Servoy Alliance Network (SAN). I have a bachelor's degree in Computer Science and Mathematics and I taught myself Servoy in 2006 after a decade of developing database applications, primarily using Omnis and SQL databases. I provide Servoy development, training and one-on-one coaching. For information about online training classes and other services please visit my website.

As I was teaching myself Servoy, I noticed that there were topics that the existing Servoy documentation didn't address fully, and that is why I wrote this handbook. This handbook is a collection of 'how-to' lessons and discussions for developers who want to learn Servoy. It is not a replacement for the Servoy documentation but a complement, one that will help round out your understanding of Servoy and get you up to speed faster.

Feel free to email me anytime at feedback@mcgilly.com with questions, comments, criticism or corrections regarding any aspect of the handbook.

Everything mentioned in this handbook applies to Servoy v. 4.x, unless otherwise noted.

My thanks to John Allen, Marc Norman, Jan Aleman and Bob Cusick who all contributed ideas, corrections and encouragement towards the creation of this handbook. Also a big thank you to all the Servoy developers who ask and answer questions on the Servoy Forum, as those exchanges help me identify what topics to include in this handbook.

by Adrian McGilly, Servoy Consultant & Trainer

How to Learn Servoy

by Adrian McGilly

Available Resources

Because Servoy leverages many different languages & technologies (SQL, SQL Databases, Java, Javascript, HTML, etc.) becoming successful with Servoy requires that you become somewhat familiar in all of these areas.

If you haven't yet worked with a truly relational SQL database, then you'll need to read up on that a bit so that you become familiar with relational database concerpts and know how to design an efficient, normalized datamodel.

- If you don't know HTML at all then read up on it a bit.
- If you don't know what a Java Virtual Machine is then you need to educate yourself a bit about Java.
- If you're new to JavaScript, look at the JavaScript Primer in the *Servoy Advanced Programming Guide for FileMaker Developers*. You don't need to know much JavaScript to get started in Servoy so just get a feel for the basics at first - (grammar, syntax, variable declaration & typing, flow control). I also recommend you buy O'Reilly's "JavaScript, The Definitive Guide" (it's the one with a rhinoceros on it), as the O'Reilly books have always impressed me as being, clear, concise and well-organized manuals.
- If you're new to SQL, take a quick look at the SQL primer in the *Servoy Advanced Programming Guide for FileMaker Developers*. Depending on the complexity of your project, you may never need to use SQL, but it's a good idea to have a basic understanding of what it is and how it is used.

When it comes to learning Servoy itself, there are tons of resources available to help you. Below is a list of all the ones I know of at this time:

- Documentation from Servoy, available for free download as a PDF from the PDF Documentation page of the Servoy website or for purchase in book format from the Servoy Store. This includes:

    Servoy Developer User's Guide

Servoy Developer Reference Guide

Advanced Programming Guide for FileMaker Developers

Servoy Server Administrator's Guide

- The Servoy Help system built into Servoy Developer contains everything that you would find in the Servoy Developer Reference Guide.
- There is some documentation built into the Servoy Script Editor itself. See this section for more information.
- The Servoy Magazine, an online, interactive magazine full of great tips, techniques and how-to's (plus the odd laugh).
- Online Tutorials on the Servoy website
- Virtual Users Group meetings and Webinars put on periodically by Servoy – watch the Events page of the Servoy website
- Online Tutorials on Scott Butler's (The Servoy Guy) website
- Various resources on Greg Pierce's (Agile Tortoise) website
- The Servoy Gallery - a section of the Servoy website that shows screenshots and descriptions of over a dozen Servoy solutions in production. Looking at these screenshots is a great way to see the full range of capabilities of Servoy' GUI.
- Demo solutions which you can download from the Servoy website, explore and use as templates for your own solutions.
- This handbook (*The Servoy Beginner's Handbook* by Adrian McGilly).
- Servoy Training classes, offered online and in person by myself, Servoy, ClickWare Inc. and others
- The Servoy Forum - an online discussion group where close to 1000 developers are asking questions, sharing ideas and getting answers. Development staff at Servoy regularly respond to questions on this forum.

- ServoyWorld, the annual conference hosted by Servoy, is a great place to meet developers and Servoy staff, learn what others are doing with Servoy and find out what's on the horizon from Servoy's engineering team.

- And by far the most important learning tool of all: *Servoy itself*. Play with it. Mess with it. Don't worry – it won't bite. The best way to learn is by doing.

Suggested Learning Path

Here is my suggested learning path for newcomers to Servoy.

1. Get Servoy up and running on your computer, using the Sybase SQL Anywhere database as your database. If you have trouble getting connected to the Sybase SQL Anywhere database read the section about connecting to the database.

2. Watch the Online Tutorials on the Servoy website.

3. Read the Developer User's Guide cover to cover. It's OK to skim parts that don't feel terribly relevant to your particular needs but you should spend some time in every section of this book so you understand the full range of Servoy's capabilities. You should also 'play along' and try things in Servoy as you read.

4. Skim the Developer Reference Guide, but don't try to understand everything in there at first. That will come with time.

5. Open up the svyCRM Demo solution which is provided with the Servoy install and which is available for download from the Download Demo Solutions section of the Servoy Website. Poke around in this solution, both in run-time and design modes. Look at how the forms are put together. Look at some methods. Explore the tables. Don't worry about breaking the demo - you can always reinstall it.

6. If you're a FileMaker Developer, you will find other parts of the *Servoy Advanced Programming Guide for FileMaker Developers* very useful too.

7. You probably have a project in mind that you're aching to try in Servoy. Think of some simple part of that project - an interface for entering and/or browsing Client records, for example, and start building it. As you stub your toe (and brain) against obstacles, consult this handbook - hopefully a good

many of your questions will be answered here. If not, run a search in the *Servoy Magazine* to see if any articles have been written on the topic you're stuck on, and check the *Servoy Forum*. The next section discusses how to get the most out of the *Servoy Forum*.

8. Lastly, at some point remember to take a look at the *Servoy Server Administrator's Guide*. Even if you're still a ways off from doing any server administration, it will help you to understand Servoy if you understand some of the server admin capabilities.

Do I Have To Know SQL To Use Servoy?

It's useful to know a little SQL, but in most cases it's not necessary, and certainly not when you're just getting started with Servoy.

Servoy provides its own set of straightforward commands for retrieving data, but Servoy also gives you the option to query your db using SQL. For example you can pass an SQL query to a function called loadRecords() to retrieve rows from a database and display it on a form exactly as if you'd used Servoy's built-in search commands. This is a convenient option for those who are comfortable writing SQL queries, but you can almost always get by without it.

Certain complex, multi-join queries are beyond the capabilities of Servoy's built-in search functions and can only be achieved using SQL. With a mature, complicated database, directly querying the database using SQL will often improve the performance significantly on some queries.

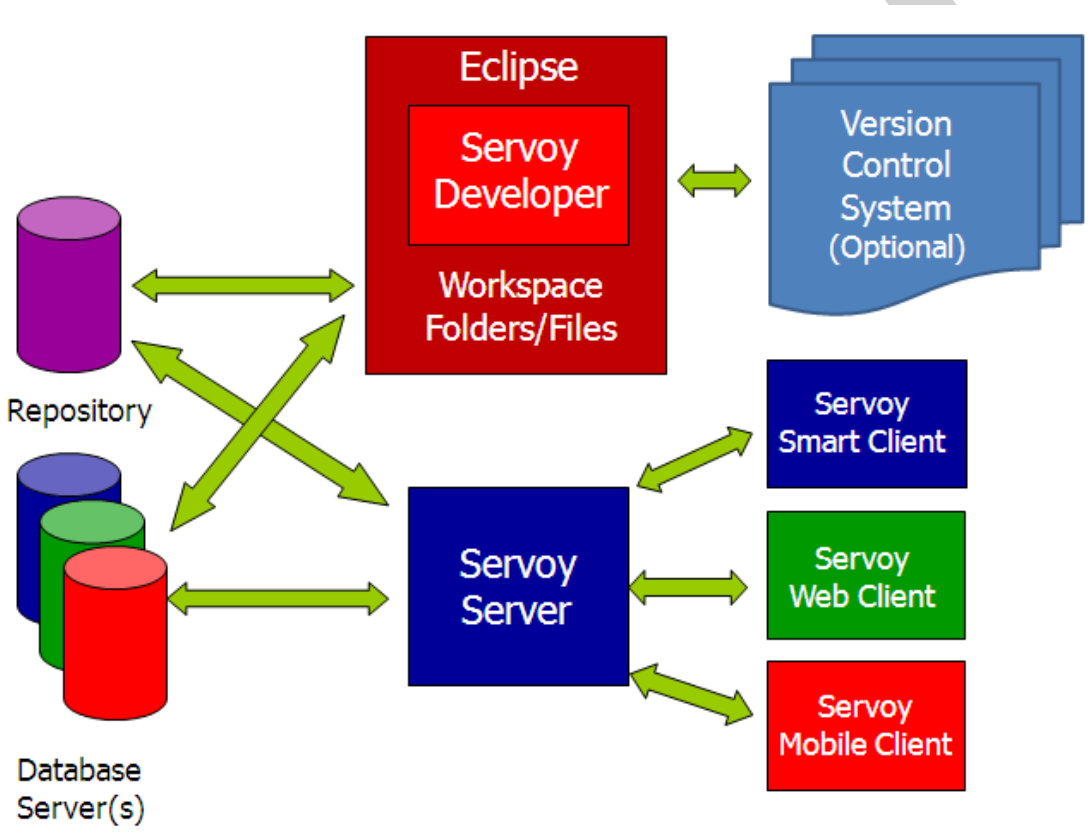
Technically, you can even use the rawSQL plugin to modify your database using SQL update, insert and delete commands, but this is only for expert Servoy users who understand the risks of doing this. Rarely is this going to be necessary.

by Adrian McGilly, Servoy Consultant & Trainer

Overview of Servoy's Architecture

by Adrian McGilly

It's important to understand the architecture of Servoy. Below is a picture of the Servoy Architecture (provided by Servoy)



Applications written in Servoy are called *solutions*.

The first thing to understand is that your solution and your business data are two separate entities in the Servoy architecture (as they should be!). Your data is held in one or more SQL databases managed by one or more database servers. Your deployed solution is also stored in a SQL database called the Servoy Repository. One of the great feats of Servoy is that it stores and manages your entire solution, including your GUI and all your business logic, in an SQL database. In fact, it can do this in any standard, JDBC-compliant SQL database.

Servoy Developer

When you are designing a solution and making changes to forms and methods, Servoy is saving your work in various files in your Servoy Workspace, which is simply a folder in your operating system's file system, typically on your local hard drive (find out more about the Servoy workspace here) .

If you are a developer working alone, this Workspace is all you need to manage your applications under development, but you will be limited in terms of version control. If you want to periodically save revisions of your solution (such that you can return to an earlier version if necessary), you can submit each revision of your solution to a *servoy_repository* db to which you will connect from Servoy Developer.

If you are part of a team of developers you should use the Servoy Team Provider feature to merge your changes into a version control system such as SVN or CVS. That features is beyond the scope of this document.

When you are ready to deploy your solution, you export it from Servoy Developer and import it into Servoy Server. This import process places your solution in a db called *servoy_repository* to which Servoy Server is connected.

It is technically possible for Developer and Server to share the same *servoy_repository* db, making it possible for you to deploy your solution by submitting it straight from Developer into Server, however from an operations standpoint this is not recommended.

Servoy Server

When Servoy Server is running and a user opens up a solution in a Servoy Client session, Servoy Server reads the solution from the *servoy_repository* (using SQL) and pushes it down to the Client. When it's time to populate forms with data, Servoy Server reads the data from whatever database(s) the solution calls for and pushes the data down to the Client. The business data managed in one solution (and indeed in one form) can span multiple databases and multiple database vendors. The repository, however, is always contained entirely in a single database. Servoy Server is written in Java and runs on top of the Tomcat application server.

Servoy Client

The piece that users interact with directly is the Servoy Client. This can be either the Java client or the Web client. More on this in a moment.

The Java Client (sometimes called the Smart Client or the Rich Client) is a Java application that runs on top of a Java Virtual Machine on the end-user's computer, providing users with a "rich UI experience".

Under the Web Client, users access a servoy solution via a web browser. The Servoy Server renders the forms using HTML, JavaScript, AJAX and other standard browser protocols. Due to limitations of browser technology, there are some UI features that are available to the Smart Client that aren't available to the Web Client.

In addition to these major components of the architecture, Java beans and Servoy Plugins can be used to extend Servoy's functionality. These need to be placed in the appropriate folders on the Servoy Server. Servoy Server takes care of downloading them to the client machines as necessary (including updates when new versions of the beans and plugins are installed) so that all plugin and bean administration takes places on the server, not on the client.

Batch Processes

Servoy Server also supports Batch Processes. These are Servoy solutions that don't have any user interface and which you want running continuously on the server. One use for batch processes is to schedule various jobs to fire at regular intervals using Servoy's built-in Scheduler functions.

by Adrian McGilly, Servoy Consultant & Trainer

Servoy Developer & Eclipse

by Adrian McGilly

What is Eclipse?

Servoy Developer is an Eclipse plugin. Eclipse is a standard software development platform which is included in your Servoy installation. Started by IBM and now an open source project supported by the Eclipse Foundation, Eclipse supports development in many programming languages and environments, not just Servoy. The Servoy plugin for Eclipse provides a full-featured Integrated Development Environment (IDE) for Servoy development. By working within Eclipse, Servoy makes itself more attractive to a huge number of enterprise developers who are already using Eclipse, thus promoting greater acceptance in the marketplace.

Eclipse is full of features that will make a Servoy developer's life easier, but there are a great many features in Eclipse which are of little or no use to a Servoy developer, so don't be intimidated by its depth and the huge number of commands and options in its various menus. Over time you'll figure out which ones are useful and which ones to ignore.

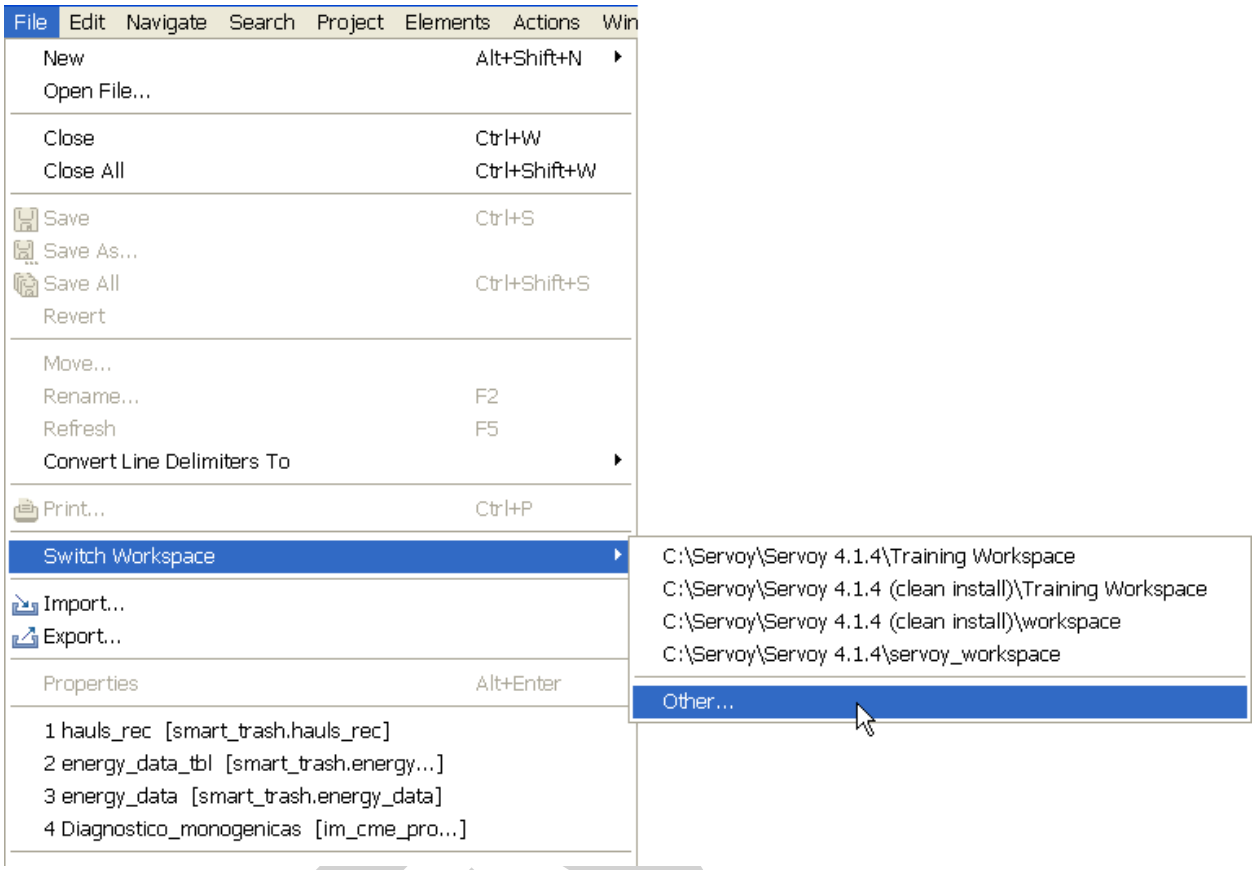
As you work with Servoy 4.x, sometimes you will be using Eclipse features, and sometimes you will be using Servoy features, and it won't always be apparent which is which. In this document, I won't attempt to make that distinction – I'll just say "Servoy does X" even if in fact it's Eclipse doing X for Servoy.

Here is a posting on the Servoy Forum describing useful features of Eclipse.

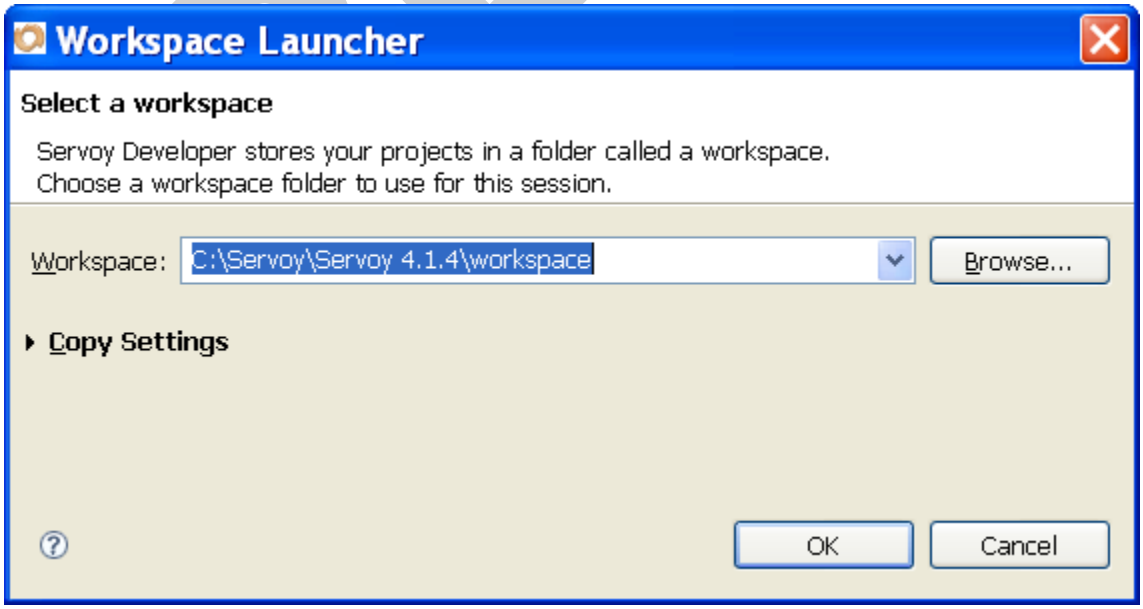
Workspace vs. Repository

Before you can begin working on a solution you have to get it into your Eclipse Workspace. The Workspace is a folder, usually on your local hard-drive, where Servoy stores your solution while you are modifying it. Certain preferences and settings for the Servoy IDE are also stored in the Workspace.

You can find out where your Workspace folder is by selecting File>>Switch Workspace>>Other…



This will display a dialog showing you what your current workspace folder is, and allowing you to change to a different workspace folder if you like:

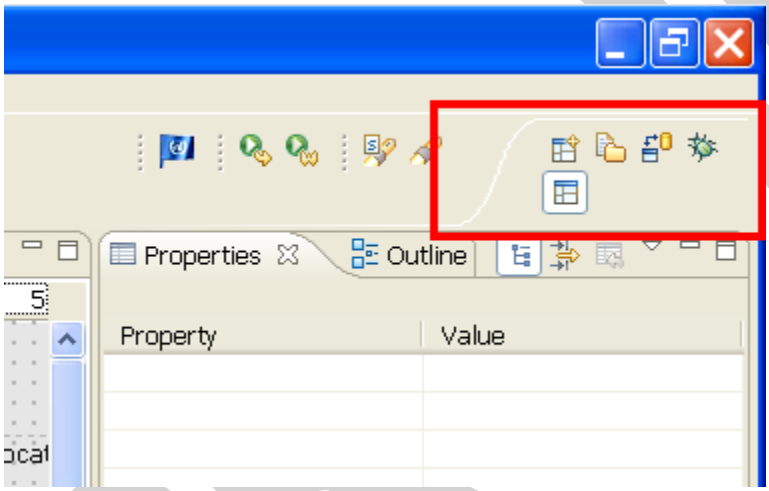


When you first start up Servoy, it will create a Workspace folder for you called servoy_workspace. Where it puts this folder varies by operating system. I personally like my Workspace folder to be inside my Servoy folder. This way I know that if I make a backup of my Servoy folder, I know I'm backing up everything, including any solutions I'm working on.

Eclipse Perspectives

In Eclipse, a "Perspective" is a set of panes that together support a certain type of work such as Form Design, Debugging or Synchronization. Servoy has already created some perspectives for you, which you can see if you select Window>>Open Perspective>>Other…

You can also switch among any perspectives you have visited recently using the Perspectives toolbar :



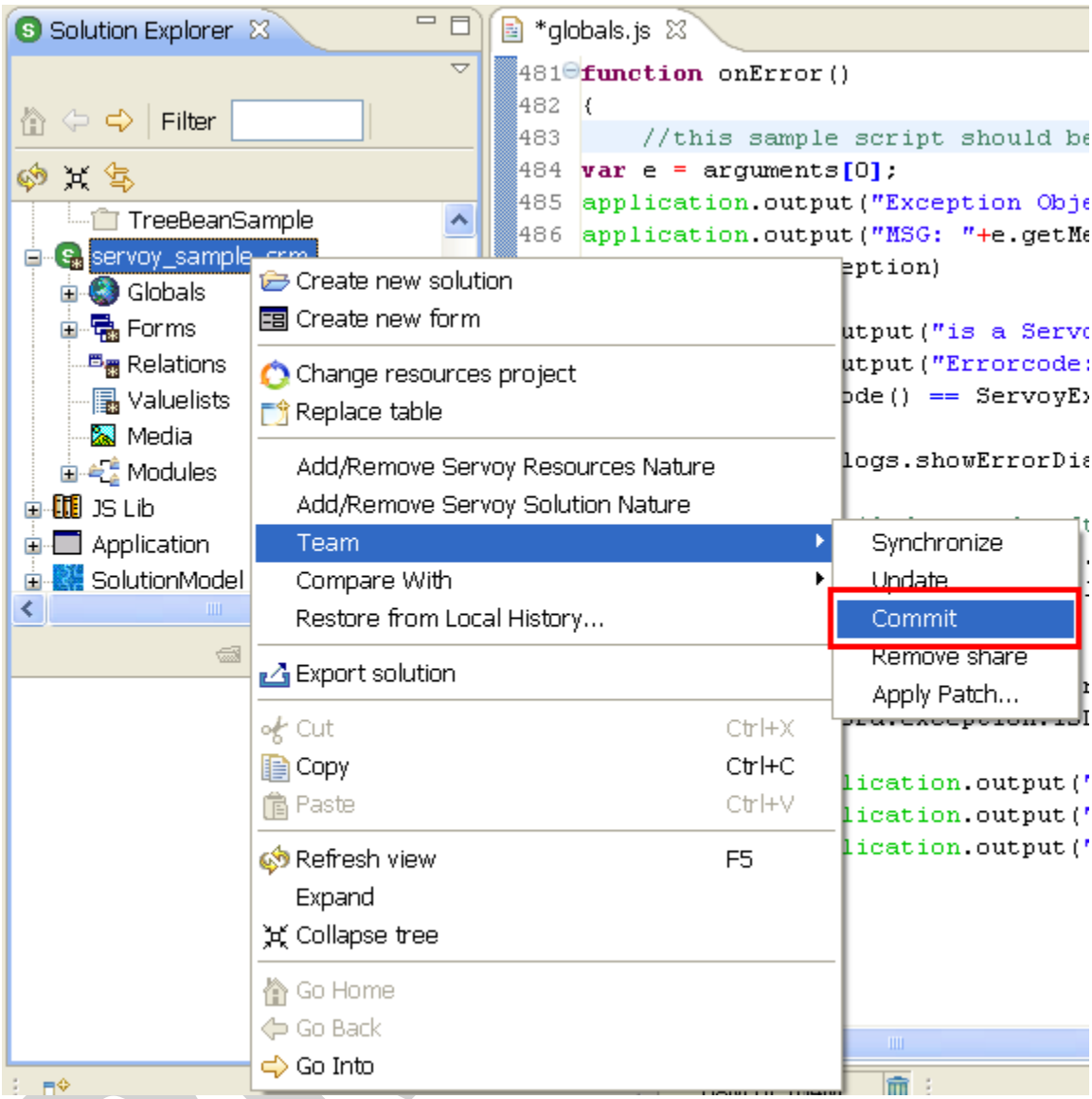
How do I save my changes to the Repository?

To save your solution in your local repository:

1. The solution needs to be the active solution. If it isn't already, activate it by right-clicking and selecting 'Activate solution'.
2. Right-click on the solution and select Team>>Commit. If the 'Commit' command isn't available then select 'Share Project…' and instruct Servoy to

share the project using the servoy_repository db. Now the Team>>Commit command should be available for you to select.

3. If you are lucky, your solution will get committed without requiring any further steps. If you are working as part of a team of developers sharing the same repository then there may be conflicts requiring resolution before your changes can be committed. To help you resolve these conflicts Eclipse has a 'Synchronization' perspective which you will be thrown into if conflicts are detected. I have found that even as a lone developer with sole access to my repository, Servoy often perceives conflicts between my new version (the one I'm submitting to the repository) and the latest version in the repository. Obviously this makes no sense seeing I'm the only one working on the solution and I can't be in conflict with myself. This is something the engineers at Servoy are aware of and have promised to fix in a future release. So if this happens to you, here are some workarounds to this problem.
    a. Go back to the Form Design perspective and try committing a second time. Sometimes the second time it works without finding any conflicts.
    b. If that doesn't work the from the Synchronization perspective, look in the list of conflicts for any bearing a red flag. Right click on each such item and select 'Mark as merged'. Now return to the Form Design perspective and try committing your solution again.



Here is an article on the Servoy Forum which describes the normal lifecycle for submitting code to the repository. A lot more could be written about this topic but it's beyond the scope of this document.

Connecting to SQL Anywhere Databases

by Adrian McGilly

About SQL Anywhere

SQL Anywhere is just one of many dbs that Servoy can connect to, but seeing it's the one that's bundled with Servoy, it's the one I'll assume you're using throughout this handbook.

SQL Anywhere has many names. It is sometimes called ASA for Adaptive Server Anywhere, and it is sold by a subsidiary of Sybase called iAnywhere and is sometimes called iAnywhere for that reason.

The license agreement Servoy has with Sybase allows you to deploy solutions using the Sybase SQL Anywhere database without having to pay any license fees to Sybase, provided no other software is connecting to the SQL Anywhere db. Given the depth, robustness and scalability of the SQL Anywhere db, this is a great value.

Before getting too far into Servoy, you will want to download a copy of the database management toolset for SQL Anywhere, called "SQL Anywhere Developer Edition" the main component of which is called Sybase Central. Sybase Central lets you manage your database schema, tweak column attributes like NOT NULL, INDEXED and UNIQUE, browse your raw data, perform adhoc SQL queries, create new databases, etc.

To read about SQL Anywhere, visit this link.

To download a free copy of SQL Anywhere, visit the Download Sybase Central page on the Servoy Website or go here: http://www.ianywhere.com/downloads/ and click on SQL Anywhere Developer Edition

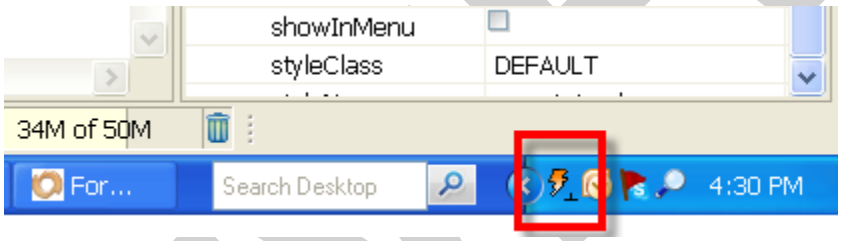
Connecting To SQL Anywhere Databases

There are a few details that need to be precisely in place for Servoy to successfully connect to SQL Anywhere databases. Generally, after installing Servoy you should

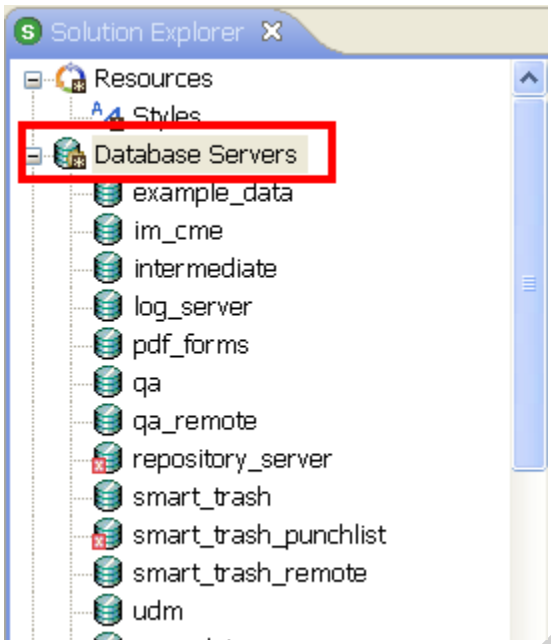
be able to connect without any problems, but there have been installers in the past where this wasn't the case, so here are some pointers.

When Servoy starts up, it starts the SQL Anywhere database engine, which is contained in the servoy/application_server/sybase_db folder. SQL Anywhere then looks in the file called sybase.config for a list of databases it expects to find and it tries to open each one in the list. If ANY of these don't open successfully, Sybase quits (however Servoy sill opens – it just won't be able to connect to any databases and therefore won't work very well).

If you're developing on a Windows machine you can tell if SQL Anywhere is running by whether its icon appears in the icon tray (see below):



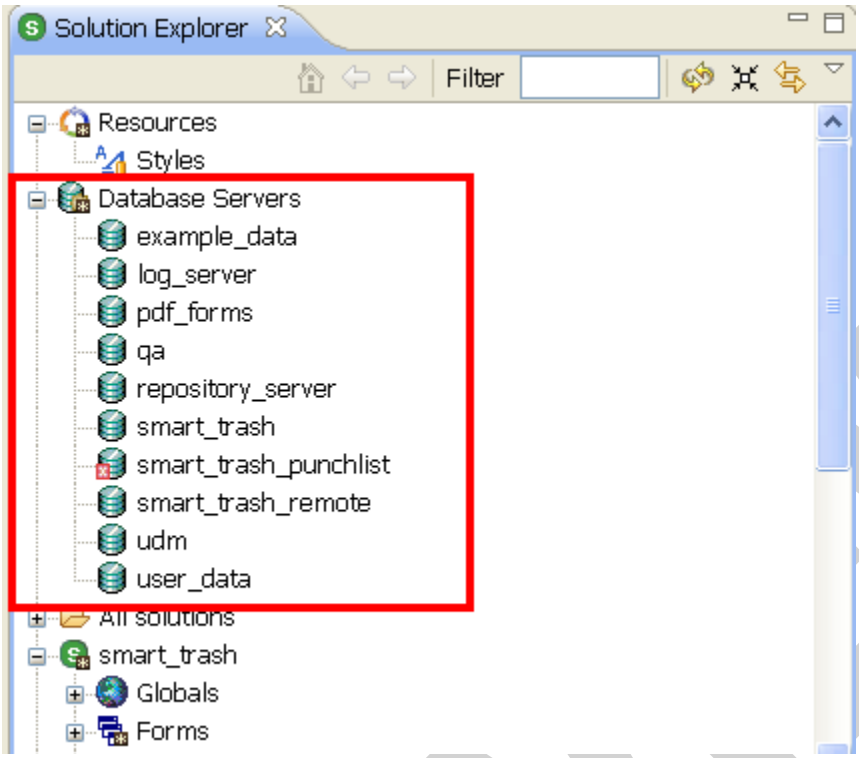
On a Mac look in the Activity Monitor for a process called dbsrvXX where XX is the version of Sybase you are running.



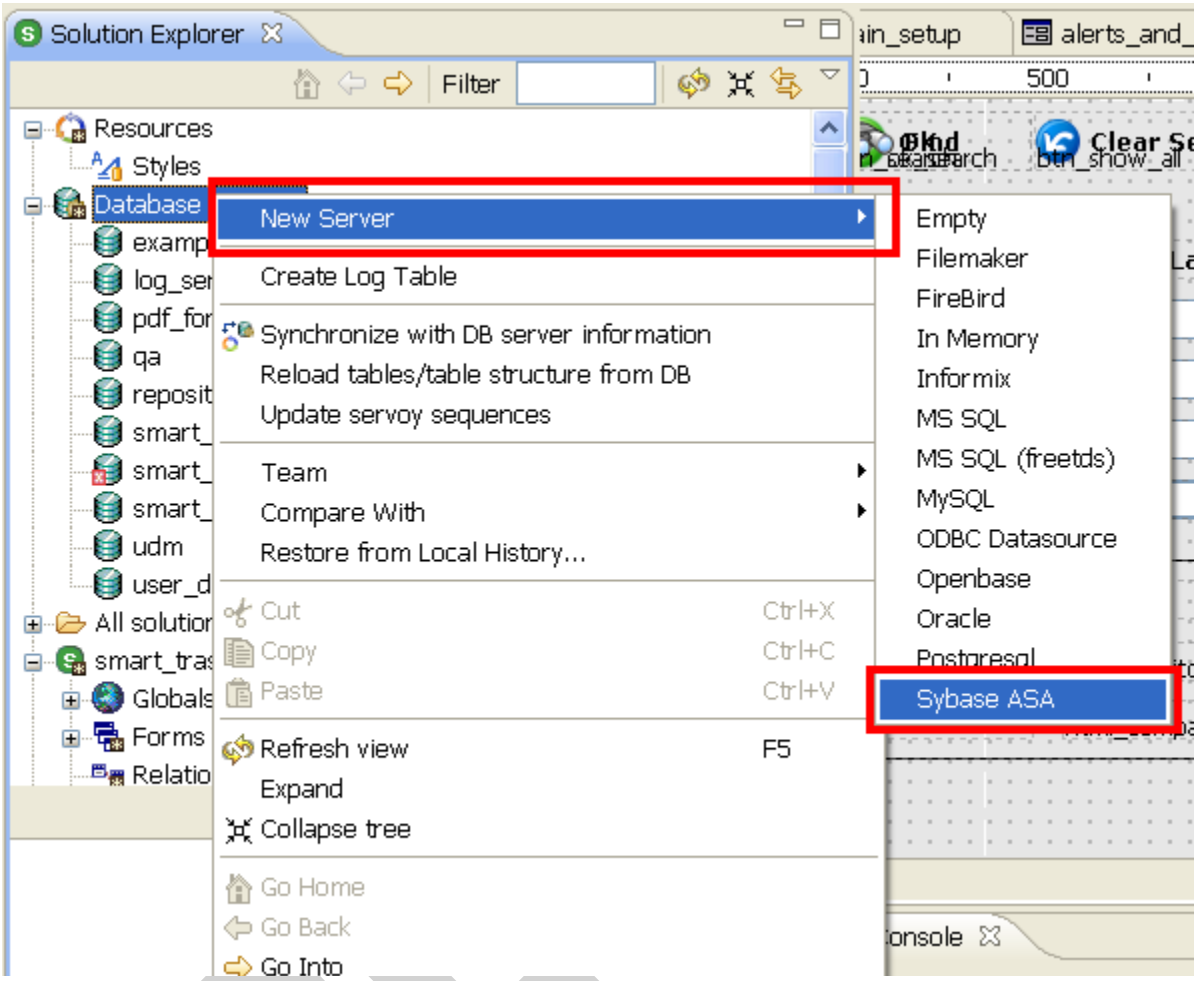
If you don't see any database servers listed under the Database Servers node of your Solution Explorer, then something went wrong with Sybase. To resolve this, quit Servoy, open the sybase_log.txt file in the sybase_db folder and scroll to the bottom. Usually you will see there a message telling you what's wrong, such as "Could not open/read file: database/mydatabase.db".

Note: it's OK if there are databases in the database folder OTHER than the ones specified in the sybase.config file. What's not OK is if there is a database specified in the sybase.config file that is NOT in the database folder.

Once Sybase has successfully opened a database, Servoy needs to be told what databases to connect to and how. These database connections are simply called "Database Servers" in the Servoy IDE, and they appear in the Solution Explorer under the Database Servers node (see below).



In order for Servoy to connect to a database it must have a server connection defined in this list. To create a new connection right-click on Database Servers and select New Server, then select the brand of db you are connecting to:

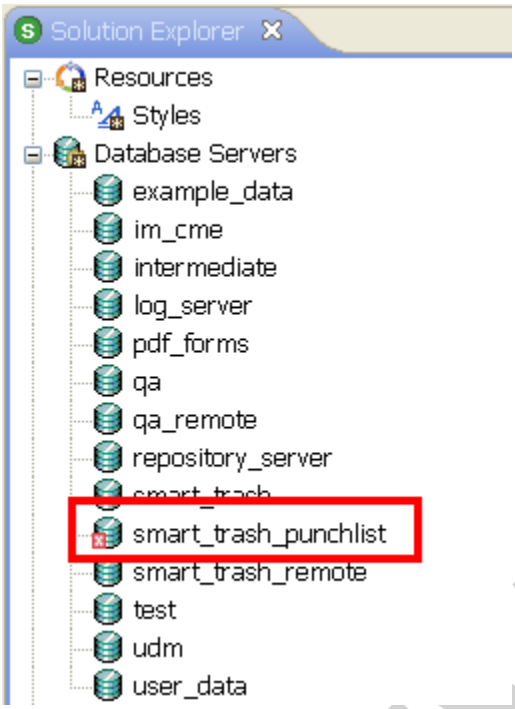


Now fill in the settings for the new db connection. Note that if your Sybase database is called crm.db, then you will put 'crm' (no quotes) as your database name (not 'crm.db'). Servoy will only let you save this new connection if it can successfully connect right then.

More details on how to create these connections are in the Servoy Developer User's Guide.

These connection settings are all stored in your servoy.properties file, which you will find in the application_server folder of your Servoy folder. It can happen that you have more connections defined than you have databases opened. In such cases, in the list of Servers under Database Servers, there will be a red X next to

any database connection which has failed to connect to its designated database (see example below).



Troubleshooting Connecting To SQL Anywhere:

Corrupt or out-of-sync .log file, corrupt database

Each database file should have a .log file by the same name (e.g. the log file for crm.db is crm.log). It can happen that the log file gets out of sync with the db file. In such cases, you can try deleting the log file and SQL Anywhere will create a new one the next time it successfully opens the database.

If that doesn't work, try forcing Sybase to start up the database without a .log file by using the '-f' option.

1. On Windows, the procedure is:
2. Shut down Servoy and any running Sybase servers.

3. Run the dbsrv10.exe application in the servoy\application_server\sybase_db folder

4. Specify the .db database you are trying to open, and put -f in the options field

5. Click OK.

On a Mac, the procedure is:

Open up a new window using /Applications/Terminal and enter these commands (assuming your Servoy directory is located in /Applications/Servoy):

```
cd /Applications/Servoy/application_server/database

export
DYLD_LIBRARY_PATH=/Applications/Servoy/application_server/sybase_db

export
PATH=$PATH:/Applications/Servoy/application_server/sybase_db

dbsrv10 -f servoy_repository.db
```

SQL Anywhere Version Mismatch

I have seen it happen that the error reported in the sybase_log.txt file is something like "capability 32 missing" or "capability 35 missing". These usually have to do with version mismatches between the db files and the SQL Anywhere db engine. This should never happen on a fresh install of Servoy, but it can happen later down the road if for instance you open a db in Sybase Central using a more recent version of the dbsrv10.exe database engine than the one that came with Servoy. If this happens, replace Sybase files in the Servoy install with those from the Sybase Central install and that should resolve the mismatch.

Firewalls

Firewalls can interfere with Servoy connecting to SQL Anywhere. If you can turn off your firewall at least while you're troubleshooting getting

connected that will help. If you need to use a firewall, look in your firewall settings for any rules blocking access to/from dbsrv10.exe and remove these rules.

Corrupt Database

If any of the dbs listed in the sybase.config file are corrupt that will cause problems. You can often decorrupt these by running the dbsrv10.exe app and using the –f parameter to "force" the database.

Need to restart SQL Anywhere and Servoy

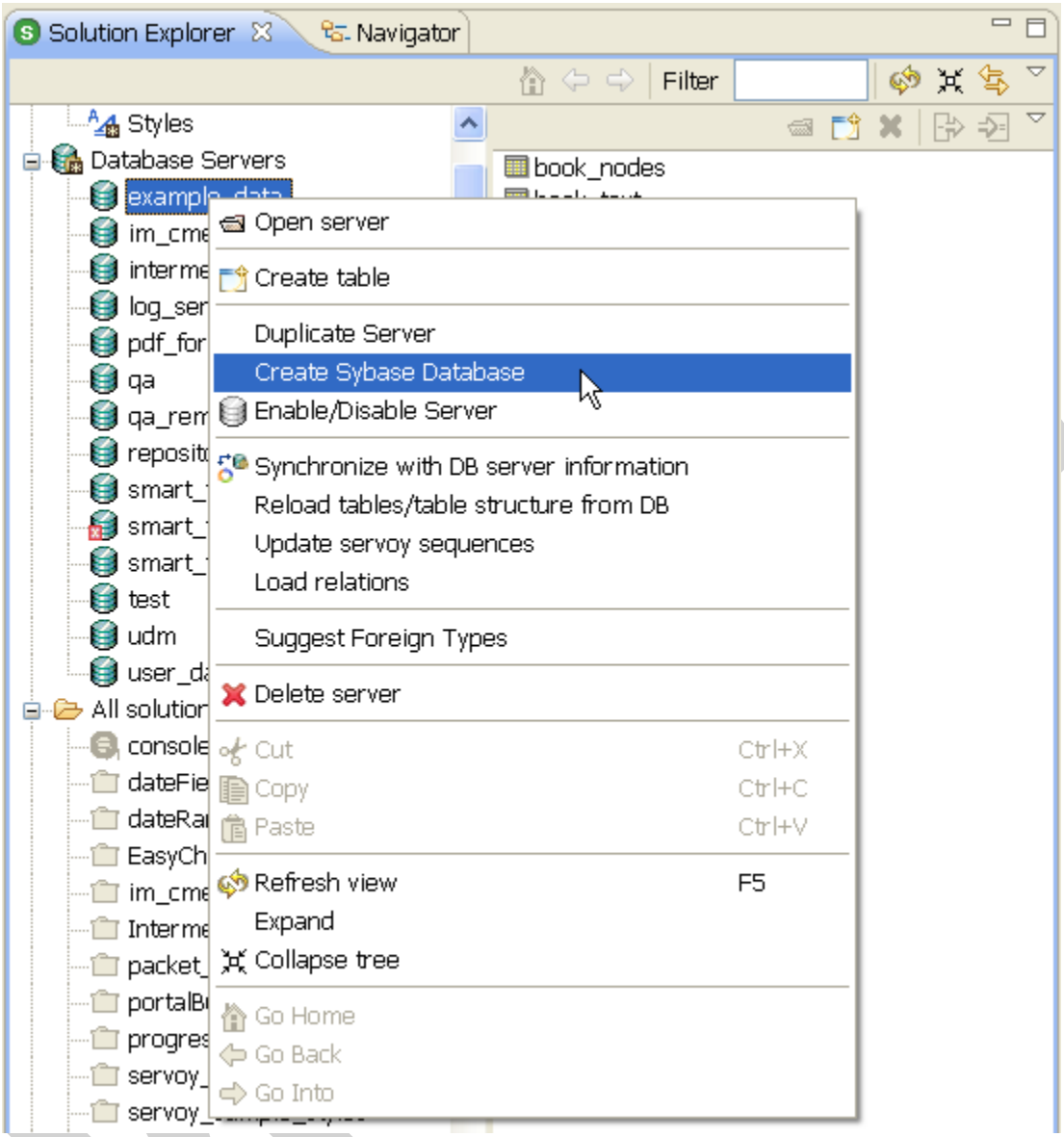
If you change settings in the sybase.config file, they will only take effect the next time SQL Anywhere starts up. If SQL Anywhere is already running and you start up Servoy, your changes to sybase.config will be ignored. So after making changes to this file, make sure you exit SQL Anywhere before you start up Servoy. To shut down SQL Anywhere in Windows, find its icon in the icon tray, right-click and select exit or shutdown (or on a Mac, select 'Quit' after selecting 'dbsrv10' using the Activity Monitor).

Similarly, if you think you've got everything set up right but Servoy is still refusing to connect, always try quitting out of both Servoy and SQL Anywhere and then restarting Servoy before giving up – Servoy uses a "lazy load" approach to starting up and this can mean that certain changes don't take effect until you restart.

Creating A New Database

Once you have Servoy connecting successfully to SQL Anywhere, you will want to create a new solution and start playing around. What database should you use for that?

You can use one of the existing sample databases that come with the initial Servoy installation (at this time they include example_data.db, user_data.db, crm.db and udm.db) or you may want to create a new database. To do this, right-click on one of the existing databases and select 'Create Sybase Database' and follow the instructions.

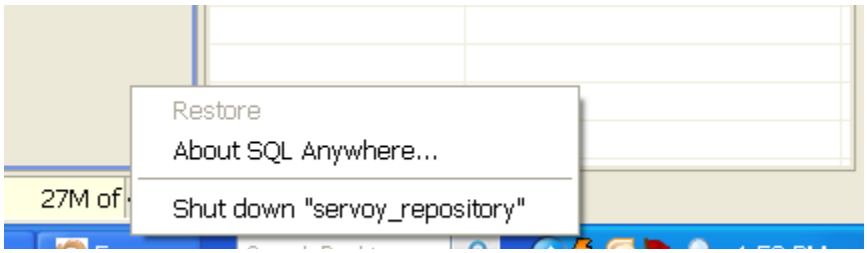


The default username and password for your new SQL Anywhere database are:

username=dba
password=sql.

Shutting Down Sybase Server

To shut down Sybase on Windows, right-click on the Sybase SQL Anywhere icon in the system tray and select Shut Down "servoy_repository":



On a Mac, locate the dbsrvXX process in the list of active processes in the Activity Monitor and hit the Quit button.

Be patient - it can sometimes take Sybase a while to quit as it cleans up its transaction log. DO NOT do a force quit of this process unless you are prepared to lose data.

How Can I See My Raw Data Outside of Servoy?

One way to see your raw data is to use Sybase Central. Once you connect to a db from within Sybase Central you can drill down to a particular table, select the 'data' tab and you'll be looking at raw rows and columns of data. Be aware that if you make changes to the data in Servoy, you will have to hit 'refresh current folder' in Sybase Central to see the changes reflected there.

There is also an interactive SQL Query editor built into Sybase Central that can be useful (though it's not hard to build such a tool in Servoy - there's one built into the svyCRM demo solution, called AdHoc Query under the Admin tab).

Eclipse also supports plug-ins for viewing SQL data right in Eclipse. Sybase and other vendors offers a number of Eclipse-based database design and administration tools which may be of interest to you.

by Adrian McGilly, Servoy Consultant & Trainer

Where is my database schema stored?

by Adrian McGilly

Tables And Columns

Servoy lets you do a small amount of database management from within Servoy itself: you can create and delete tables and columns. For any other changes to the schema (e.g. modifying attributes of columns, renaming or deleting columns, renaming tables, etc.) you need to use Sybase Central.

When a Servoy solution is opened, it loads all db schema info from the db. If you go into Sybase Central and modify the db schema (e.g. add a column, modify a column's attributes, add a table) you will see your changes in Servoy's dataprovider window the next time you start up Servoy. (It's best to shut down SQL Anywhere and restart Servoy after making such changes, although in some cases it's not necessary.)

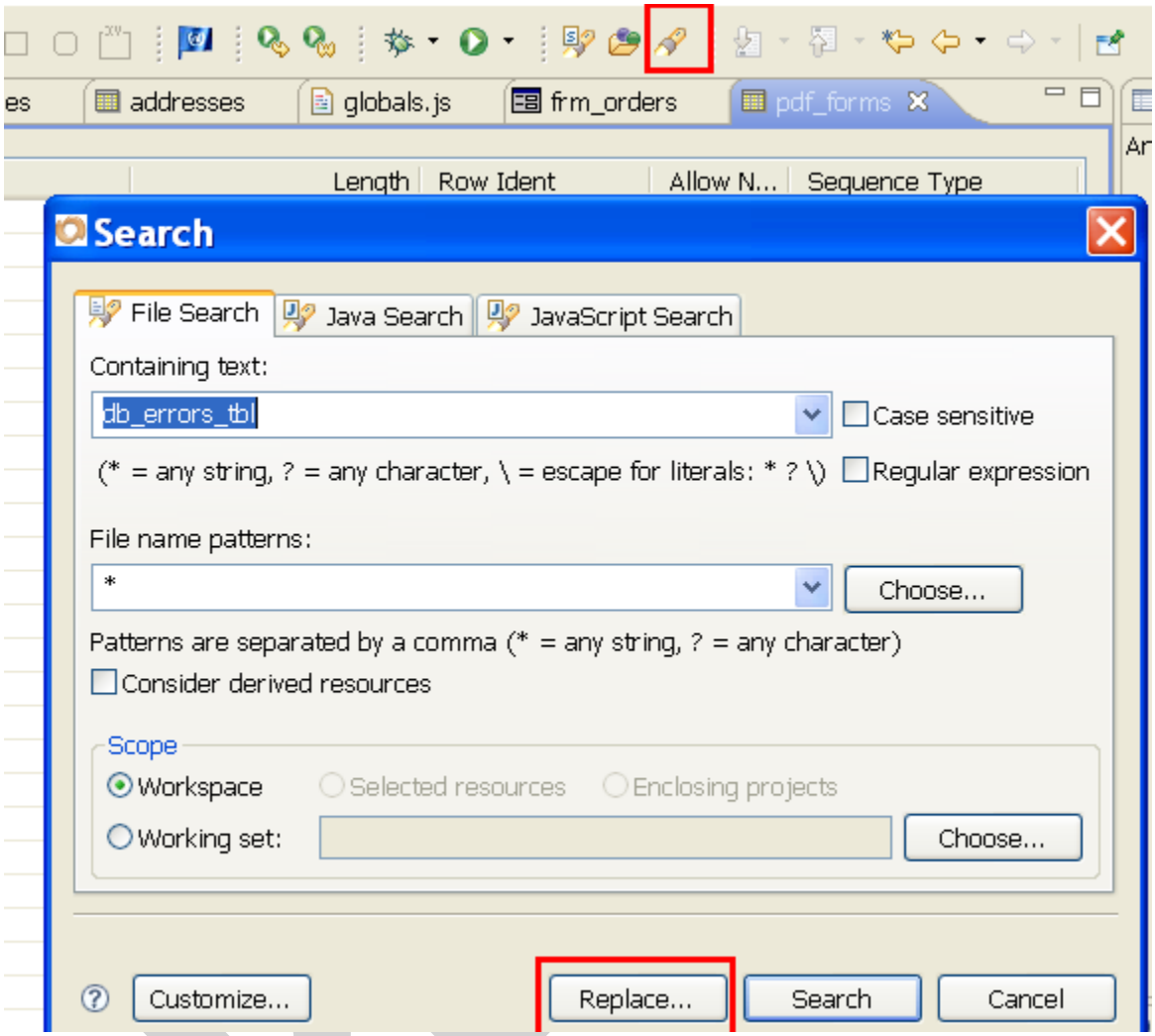
You need to be mindful when you do this so as not to break any code that refers to tables and columns.

Some examples:

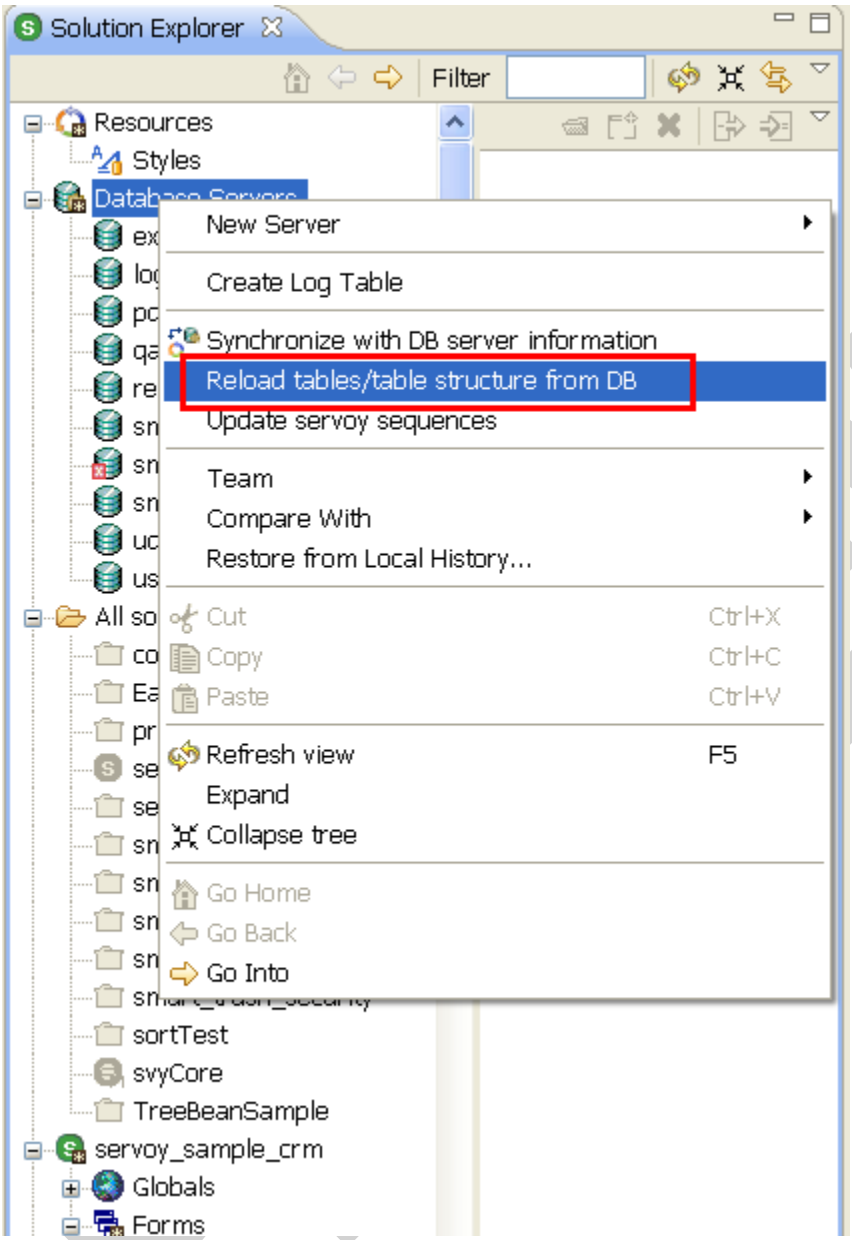
Change To Backend DB	**Effect on Servoy**
Rename a column	The new name will appear in the table editor in Servoy the next time you start up Servoy but any fields on forms that referred to that column by the old name will display as empty – they remember the old name – and methods that refer to the old name will cause errors, so be careful.
Delete a column	The change will be reflected in the table editor in Servoy the next time you start up Servoy; any fields that referred to the deleted column will display as empty;

	methods that refer to the deleted column will cause an error.
Modifying a column's attributes	The change will be reflected in the table editor in Servoy the next time you start up Servoy; adverse affects to fields displaying that column or methods referring to that column are possible, depending on the kind of change you made.
Detele or rename a table	If you delete or rename a table and that table is referred to in your solution you will get an error the next time you open your solution, either at startup or the first time the solution refers to the table.

Note that there is a global find & replace command in the Servoy Editor that will let you make global changes to your JavaScript methods:



Also, if you've made changes to the db schema outside of Servoy while Servoy Developer is running, you can load those schema changes into Servoy without restarting Servoy by right-clicking on the Database Servers node in the Solution Explorer and selecting 'Reload tables/table structure from DB:



Sequences

You can define sequences from within Servoy. One type of sequence, called a "Servoy Sequence" enlists Servoy Server to manage the generation of sequence numbers (as opposed to a "database sequence" where you are telling the database server to generate the sequence numbers). In the case of Servoy Sequences, Servoy stores information ABOUT each sequence (such as the "increment by" amount and

the next number in the sequence) in the repository, but the sequence numbers themselves remain in the database.

by Adrian McGilly, Servoy Consultant & Trainer

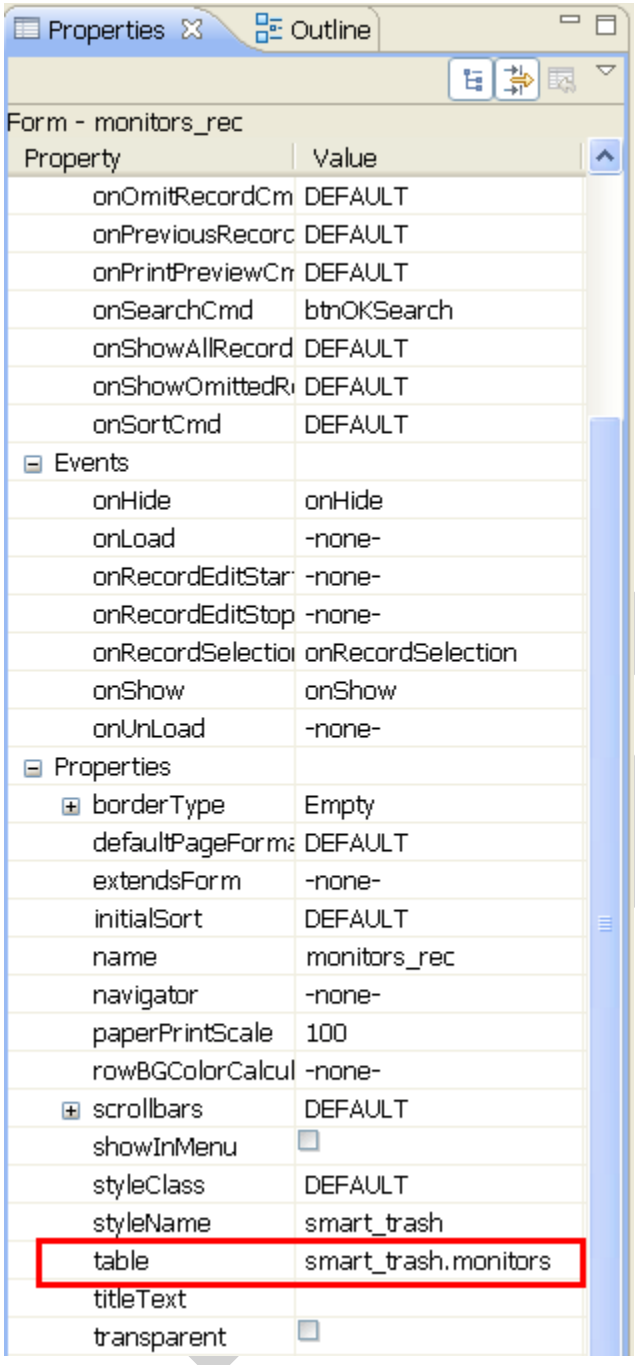
Foundsets

by Adrian McGilly

What Is A Foundset?

Each time a Servoy solution requests data from the database, a Foundset is created containing the results of the query. We'll look at the details of how and where this is done a bit later, but for now just think of a Foundset as a collection of rows that have been retrieved from the db, and which the solution can now 'play with'. Once rows of data are contained in a foundset, Servoy offers many ways to display, sort, edit, delete, print and generally process that data. Foundsets are you friends.

When you create a form in Servoy you tell it what db table the form is based on. This setting is stored in the form's "table" property (shown below) and can be seen in the properties panel when viewing the properties for that form.



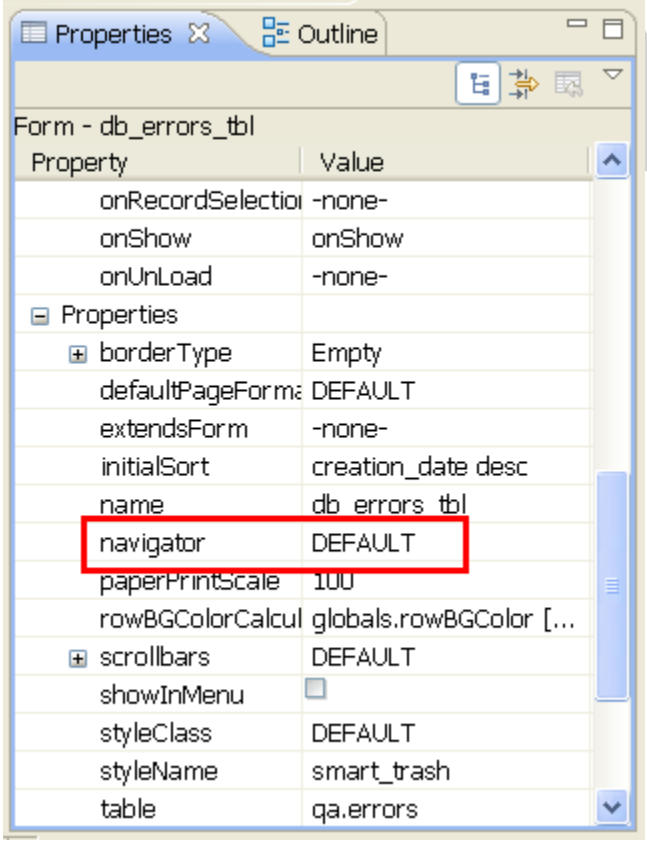
This doesn't mean you are limited to working with that table's columns on the form, it just serves as a starting point. (As you will see, you will use Relations and Tabpanels to pull in data from other tables, but more on that later.) By default, all forms that are based on the same database table will share the same foundset. As a result, two different forms that are based on the same table will by default display

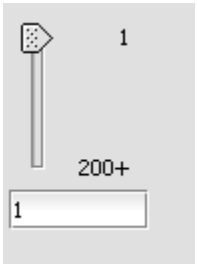
the same records when opened, and furthermore the exact same record will be "selected" or "current" in both forms.

In case that sounds a bit limiting, you'll be glad to know there's a property on each form called *UseSeparateFoundset* which, if set to true, lets each form have its *own* foundset.

A good way to get a feel for how foundsets behave is to experiment with a table that has at least 500 rows of data in it. Fortunately, the example_data sample database that comes with the installer contains several tables with 500+ rows in them (e.g. the orders and order_details tables).

Once you have the 500+ row table and a form based on that table, start fooling around with searches, inserts, saving data, list views, record views, table views, etc. Be sure the 'navigator' property of the form is set to 'DEFAULT' (as illustrated below) so that you get the default slider control (also illustrated below) for navigating records in your foundset. The numbers on this slider are quite informative.





Let's look at an example. Let's say you have a Companies table with 500 records in it. You create a customerForm form based on that table and you open the form up in runtime mode.

When you open a form in Servoy, by default it will create a foundset comprising all 500 rows in the table sorted by primary key (PK), but at first it will only retrieve the first 200 rows. Let's say you perform a search that comprises 450 rows. Your *foundset* now contains 450 rows, but again Servoy will start by retrieving just the first 200 rows in the foundset. Let's call those 200 rows the *partial foundset.*

The moment you select the 200th row of your *partial foundset*, Servoy automatically goes out and retrieves the next 200 records, so you now have a *partial foundset* of 400 out of a total foundset of 450 rows from a table that has 500 rows.

The moment you select the 400th record in your *partial foundset*, Servoy will go out and retrieve the remaining 50 records, making your *partial foundset* equal to your *foundset.*

How Do Foundsets Work?

What's really going on behind the scenes here?

To answer that question we need to look again at the architecture of Servoy. Remember that in a production situation, there are three tiers working in concert: the database server, the Servoy Server and the Servoy Client. (In development mode, there is no Application Server – the Servoy development environment is in fact a local instance of Servoy Server.)

Each time you create a foundset in Servoy, you are creating an object in Servoy Server's memory that keeps track of a whole bunch of things. For starters, it retrieves a list of primary keys (PKs) for all rows that match the search, and it holds that list in memory. It also keeps track of:

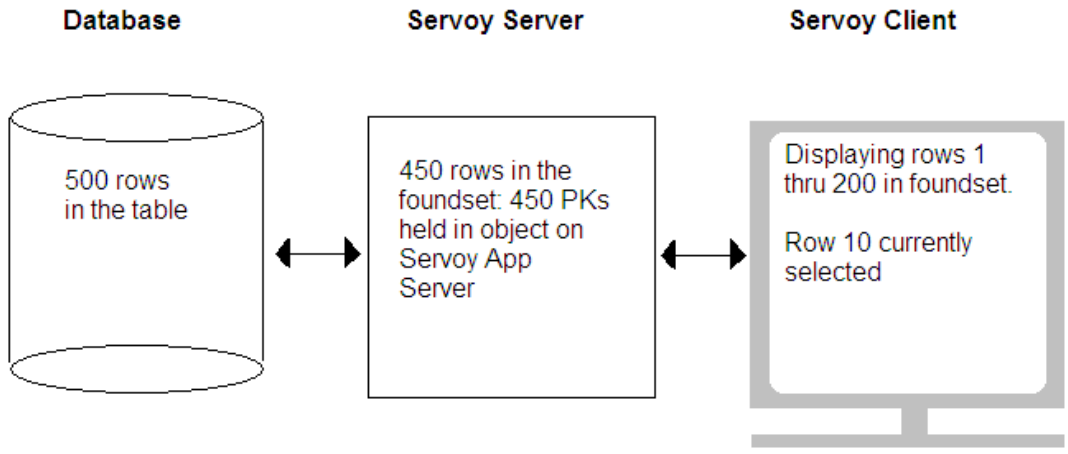
- the "query" that created the foundset,
- the total number of rows in the foundset,

- what rows are currently *displayed*, and on what form(s)
- which row of the foundset is currently *selected*.
- which rows you've modified or deleted since the foundset was created
- what other running Servoy clients are looking at those rows (i.e. in a multi-user context)
- etc.

So when you perform a search that returns 450 rows, Servoy is building a list ON THE SERVER (NOT in client memory) of the 450 PKs in the foundset. It feeds the first 200 PKs to the client, as well as all the data columns for just those rows that need to be displayed on the form.

As you navigate through the foundset, the client knows what records need to be displayed and it requests their data columns from the Servoy Server on an as-needed basis, using the PKs provided by the server. When you hit a multiple of 200 in the foundset, the server sends the client another block of 200 PKs and the whole process continues.

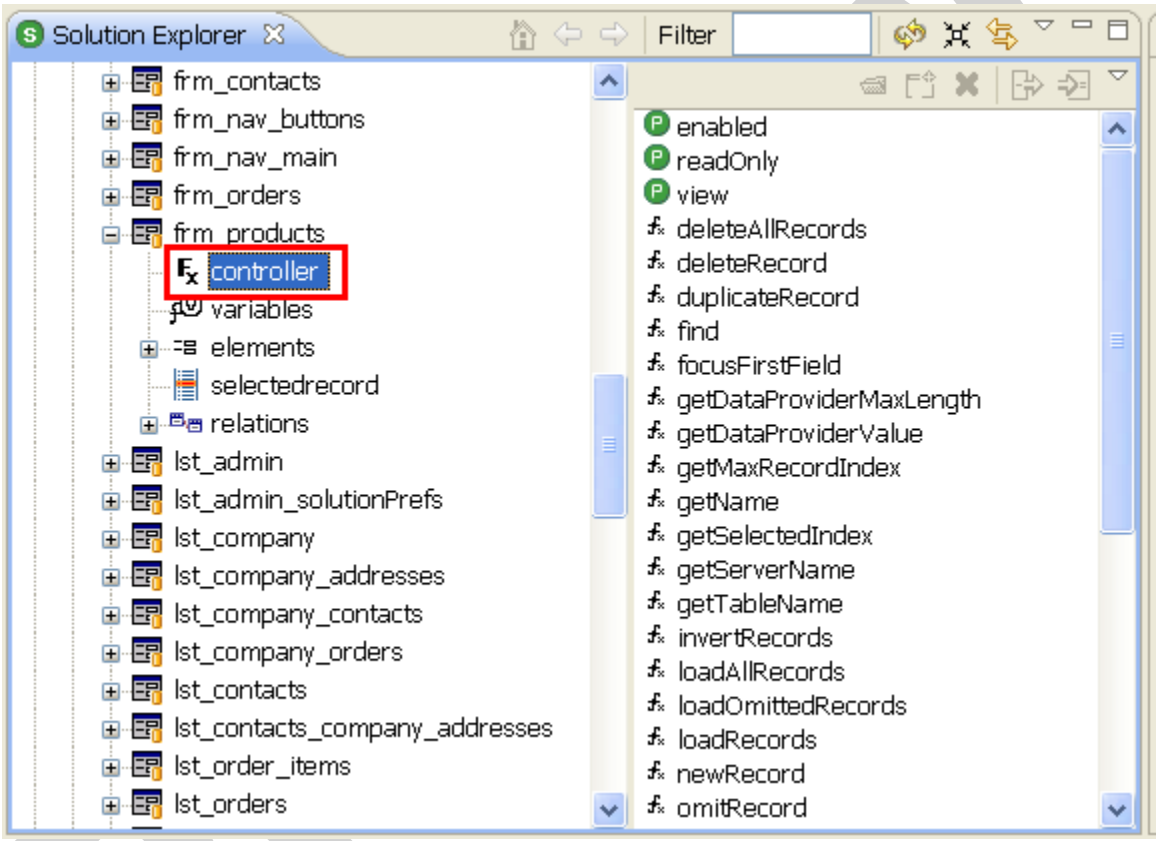
Returning to our example, let's say that Servoy has retrieved the first 200 records of your 450-record foundset and you have selected the 10th record. Here's an illustration of what's going on:



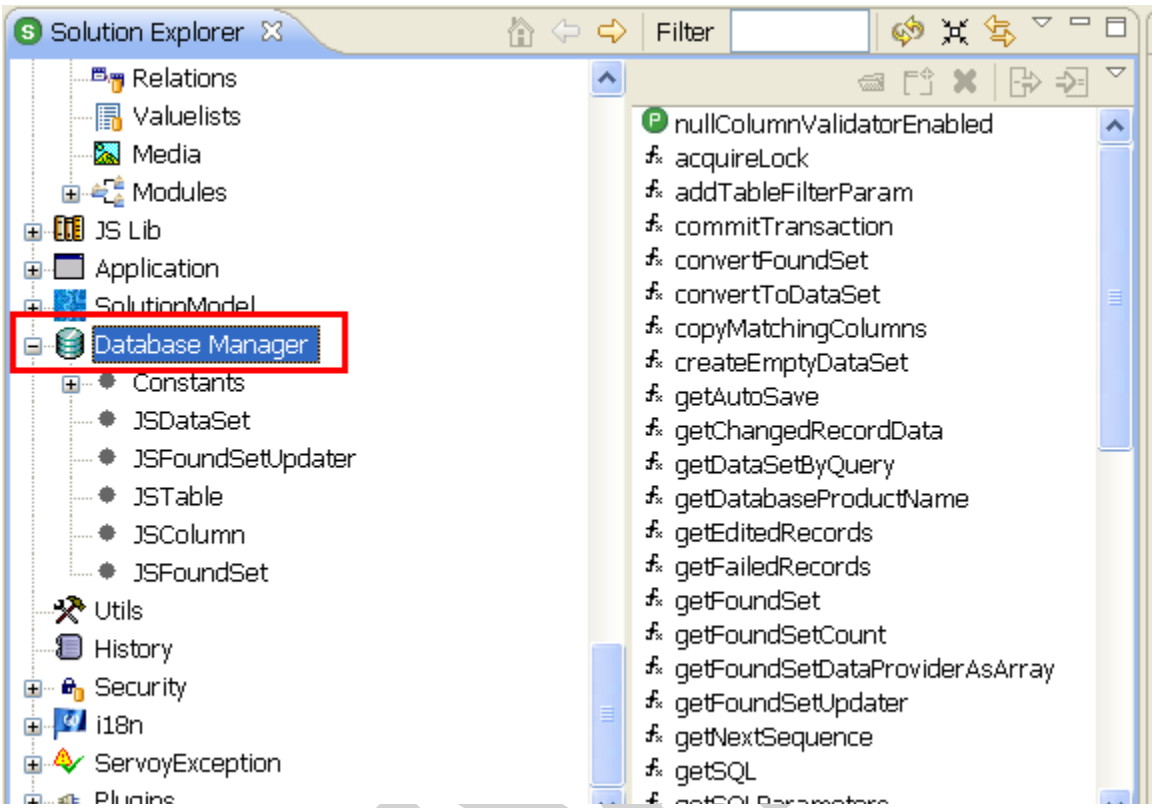
How To Programmatically Navigate A Form's Foundset

Now we know a bit about forms and their foundsets, how the foundsets are created and how they work. What tools are available to the programmer to navigate and process a form's foundsets?

There are many functions available to help you navigate a form's foundset. Most are under the form's controller node in the Servoy Editor (see below):



but a couple are in the databaseManager class, shown below:



For now, let's focus on a few key functions that give you most of what you need will need to navigate a form's foundset:

Function	**Description**
databasemanager. getFoundSetCount(foundset)	returns the total number of records in your foundset (not just how many have been returned so far to the client, but how many there are in all). In our example, that's 450.

Function	**Description**
controller.getMaxRecordIndex()	returns the number of records in your *partial* foundset (i.e. the number of rows for which the client has been fed the PKs so far). In our example that started out being 200, then jumped to 400 and then ended at 450. You can get the same result with foundset.getSize().
databasemanager. getTableCount(foundset)	returns the number of records in the table that the foundset is based on. In our example that's 500
controller.getSeletedIndex()	returns the number of the currently selected row in the foundset, which in our example is 10
controller.setSelectedIndex(n)	selects the nth record in the foundset (explained further below)

Perhaps the most important of these commands is the last one, controller.setSelectedIndex(n), which selects the nth record in the foundset. (Foundset rows are indexed starting at 1, so setSelectedIndex(5) selects the 5th row of the foundset).

What does it mean to 'select the 5th row of the foundset'? It means two things:

- Visually, that row will become displayed and selected on the form displaying the foundset.

- All values from all columns of that record will be loaded into the Servoy dataproviders for that table, making those data values available to your methods for reading and writing.
- If the row you've selected is a multiple of 200, it will trigger the retrieval of the next 200 records

Going back to our example, let's say you've just performed the search that retrieved the first 200 records in a 450-record foundset. If you did

```
controller.setSelectedIndex(199)
```

you would be selecting the 199th record in the foundset, loading its values into the dataproviders. If you did

```
controller.setSelectedIndex(200)
```

you would not only be selecting the 200th record but you would also trigger the retrieval of the next 200 records into the *partial foundset*. Note however that if you did

```
controller.setSelectedIndex(250)
```

when you only have the first 200 records, nothing would happen. You can't select beyond the end of the currently loaded foundset, and you must select the precise multiple of 200 to trigger the retrieval of the next 200 rows.

Looping Through A Foundset Containing 200+ Records

A question that often comes up is, how do can I cycle through ALL the records of a 200+ foundset if Servoy only retrieves 200 records at a time?

The easiest approach is to make use of databasemanager.getFoundSetCount() which returns the total number of records in the foundset. You can capture that value in a local var and then loop from 1 to that var. Here's the code:

```
//declares a local var and set it to the total number of
//records in the foundset
var max = databasemanager.getFoundSetCount(foundset)

for (var i = 1; i <= max; i++ )
{
     controller.setSelectedIndex(i);
```

```
	// do whatever processing you need to do here
}
```

Notice that we skipped the creating of the local var max and just done this:

```
for (var i = 1; i <= databasemanager.getFoundSetCount(foundset);
i++ )
```

But given Servoy's warnings (discussed above) about the potential expense of the getFoundSetCount() function it's best to avoid calling it at every iteration of a loop like that.

Here's another way to loop through all the records in a foundset:

```
var i = 1  //declares a local var and initializes it to 1

while ( i <= foundset.getSize() )
{
	//select row i and increments the value of i by 1
	controller.setSelectedIndex(i++);

	// do whatever processing you need to do here
}
```

or you could do this:

```
for (var i = 1; i <= foundset.getSize(); i++ )
{
	controller.setSelectedIndex(i);

	// do whatever processing you need to do here
}
```

In both of these cases, the moment the variable i hits a multiple of 200, the setSelectedIndex(i) call will trigger the retrieval of another batch of rows, and this will affect the value returned by getSize() in such a way that the loop will continue running until ALL records in the foundset have been selected.

How Adds, Edits & Deletes Affect Foundsets

When you add a new record using

```
controller.newRecord()
```

the record gets added to the current foundset immediately, but is not saved to the db until a save is performed. You can read more about saving data here.

Similarly, edits made to records are saved immediately to the foundset, but aren't saved to the db until a save is performed.

Deletes occur simultaneously in the foundset and the db.

By default, Servoy saves a record's changes to the db as soon as you leave the record, but you can modify this behavior in such a way that saving to the db is deferred until you explicitly request it. This means you might make multiple changes to multiple rows, (and possibly in multiple foundsets!) before any of these changes get saved to the db. The foundset in this case acts as a sort of staging area for making multiple changes to records before they are saved to the db.

Note: this behavior is not to be confused with db transactions, which give a separate level of control and are discussed here.

Caching & Data Broadcasting

Servoy Server does some very clever caching in order to minimize the number of times it has to communicate with the clients and the database, but it does this without compromising the freshness of the data. In fact, Servoy Server makes sure that any changes you make (and save) to any rows in your foundset are instantly reflected as necessary on any other forms in your solution and even *on the forms of other Servoy clients currently running the same solution!*

For example, if you change a unit price from $1.00 to $2.00, and that affects some calculated total appearing on a sub-form in a tabpanel, Servoy takes care of refreshing the subform automatically. Furthermore, if another user running the same solution is viewing a form that is affected by that change, Servoy Server will refresh their form automatically. Pretty amazing. (And as of version 3.5, data broadcasting works on the Web Client just as it does in the Java Client.)

by Adrian McGilly, Servoy Consultant & Trainer

This document is a preview of the Servoy Beginner's Handbook. The complete handbook is 168 pages long. You can purchase and download it from the McGilly Information Systems website for $49 US.