Intro to Servoy Concepts

In This Chapter

About Servoy

Servoy is unique in that it is both a Rapid Application Development (RAD) tool and a production deployment platform. Written entirely in Java, using industry standard components and protocols, Servoy offers the scalability of an enterprise-level computing platform, while at the same time providing unparalleled developer productivity.

Cross-Platform

As is the promise of Java, applications developed in Servoy can be deployed to all modern operating systems, including Windows, Mac, Linux, Solaris, etc. without the need to rewrite or recompile code.

Cross-Browser

Applications written in Servoy can also be deployed to any modern browser, including IE, Firefox, Safari, Opera, Chrome, etc. There is no need to rewrite or tweak an application to ensure a consistent experience across all browsers and no proprietary protocols or browser plug-ins are ever used.

Single Code Base

Servoy offers two main client options and any application developed in Servoy can be deployed to either client without the need for any rewrite.

The Smart Client provides a Client-Server setup, where applications assume the native look and feel of the client operating system.

The Web Client provides a dynamic browser experience enriched by AJAX technology, CSS and HTML. Developers are never required to author their own markup, client-side JavaScript or AJAX (Of course they can if they want) .

Cross-Database

Servoy is NOT a database. There are plenty great databases out there and Servoy can connect any modern RDBMS (and even some that aren't so modern).

N-Tier Architecture

Servoy provides a powerful Application Server which brokers connections between clients and back-end resources. This architecture is far more favorable to the 2-tier architecture used by most 4GL tools, where clients connect directly to back-end databases. An application server offers the following advantages over a 2-tier setup.

- Greatly reduced load on the back-end databases.
- Increased performance
- Increased Scalability
- Increased Security
- Ability to operate over a WAN (Wide Area Network)
- Data Broadcasting: The ability for all clients to see real-time data changes, a feature unique to Servoy.

Zero-Deployment

Servoy applications never have to be installed on the client machine.

Smart Client applications will download and install automatically using Java WebStart technology. In fact, developers can easily roll out new changes to users, with a single click to export a new build to the Application Server. The Application Server does NOT have to be restarted and connected clients do NOT have to be shutdown. Moreover, the Application Server supports versioning for roll-back, roll-forward capability.

Simply put: No Installation. No Maintenance.

Programming Concepts

Solutions

A *solution* is a single application containing forms, methods, variables, and other items built in Servoy Developer.

Solutions can stand alone, or they may also be a *module.* A module is a solution that is part of another solution. A solution can be specified to be a module, but it is not required to use a solution as a module.

Solution types

Solutions can be built for a specific purpose or a specific client technology. In Servoy there are 3 client technologies to choose from:

-NG client, which runs in a modern browser and uses angularjs and websockets

-Webclient, which runs in a browser and uses ajax (this client technology is available for backward compatability only)

-Smart client, which runs on computers using java runtime edition and uses an RMI connection

-Servoy mobile, which runs in a browser or embed in phonagap/Cordova, and works with local offline data

Choosing the right type depends on: requirement for online/offline and requirement for access to local hardware (on a computer)

Form

A *form* is the Servoy object that is used to edit, view, modify, insert, find and delete data. Forms can contain fields, labels, graphics, tabpanels, lines, buttons, portals and shapes - each of which has its own set of design time commands, events and properties.

Forms are normally based on a single database table, which means the data from that table is tied to the form. Forms can also not be linked to any table, meaning data will not be tied to the form.

Forms can also have methods and variables attributed to them, making form a unit of scope in Servoy programming. This means forms can have methods and variables that are within the scope of the form only. Inherited forms can also have methods and variables that are inherited from the parent form as well.

Form Element

A *form element* is a Servoy object that can be placed on a form for a purpose. Form elements include:

- Label - Text that can appear on a form; normally static but can be derived at runtime, cannot be changed or copied by the end user.
- Field - An element connected to a dataprovider; can be edited and selected by the end user and has many display types (text, radio buttons)
- Button - Special type of label preformatted to look like a clickable object on a form, normally tied to a method.
- Lines and shapes - various drawing objects that can be dropped on a form for aesthetics and UI design
- Portal - object that shows related data in a one to many relationship in fields.
- Tabpanel - an object that allows multiple forms to be displayed in a form. Additional forms can be added through a relationship (thereby showing related data) or as an unrelated form.
- Beans - a Java Bean can be added to a form, extending the functionality of the user interface in Servoy. A Java API is available to write beans that can be used in Servoy.

Dataproviders

A *dataprovider* manages information. Dataproviders are not only database columns, but also include calculations, aggregations, global variables, and form variables.

Methods

Servoy's scripting engine uses *methods* to identify code that belongs together. A Servoy method is essentially the same as a JavaScript function. It is a grouping of code in a single unit used to perform various tasks in Servoy. A method can call other methods, set runtime properties for Servoy form elements, and call functions available in the scripting libraries.

Functions

A *function* in Servoy is the same as a method. It is a JavaScript function and is seen as the first line in a method call as in the sample below.

```
function onActionAdd(event) {
   //code here
}
```

All of the methods written in Servoy will begin with the word function in the actual `.js` page they are written in.

Plugin

A *plugin* extends the functionality of Servoy by using the Java programming language. Written in Java, they can be accessed in a solution through scripting. Plugins can add additional functions available to the Servoy programmer and can link to resources outside of Servoy. An API is available to write plugins for Servoy.

Data Concepts

The following concepts are related to data management in Servoy.

Table

A *table* is a preset format of rows (records) and columns (fields) that defines a store of information in a relational database. Data from created or pre-existing tables can be used by one or more forms in a Servoy solution.

Column

A *column* is an attribute of a database table.

Record

A *record* is mapped to a single row in a SQL database table.

Foundset

A *foundset* is a Servoy object that represents a query from a single database table. Foundsets handle the caching of data for display, but do not need to requery the database to do so.

Any form based on a table in Servoy has a underlying foundset managing the data for it. Forms based on the same table will share a foundset unless they are directed differently through scripting or design time property.

Foundsets can be created, accessed, altered, controlled, and applied through scripting.

Relation

A `relation` is a Servoy object used to define the key relationship between two database tables. A standard relationship is defined by the primary key of one table being linked to a foreign key of another. Some features of a relation include:

- The tables can be linked by equality, but also by other operators (<,>,like, and so on).
- The tables can exist on two different database servers on different database platforms.
- The tables can be linked by more than one column.
- A global variable can be used on the left side of the link. This is useful for filtering, as a relation can be used to filter a set of records by setting the value of a global variable.
- Referential integrity can be enforced, as well as cascading deletes.

Dataset

A *dataset* is a Servoy object that can hold row and column data. Datasets can be created via scripting, or can be returned by a custom SQL query as well. Unlike foundsets, datasets do not represent a live link to a database resource.

Calculation

A *calculation* is a derived value that acts like another column in the database table. Calculation data is the result of a JavaScript function and is similar to a Servoy method. The main difference is what is available to the function; a calculation function cannot access form variables, values from form elements, or application functions as it is calculated in the data layer of the application. The function does have the current column's data and any related data available to it.

Calculations come in two variants: Stored Calculations and Unstored Calculations: Stored calculation are actual database columns, while unstored calculation only live in memory when the solution is running.

Calculations are calculated when their value is requested, for example when it is displayed in the UI or used in scripting or in another calculation and the value was not calculated yet or has been flagged as invalid. A calculation is flagged as invalidated when a change occurs in one of the dataproviders on which the calculation depends.

If there are UI elements that display the calculation, they will be notified to update themselves and the calculation is executed to calculate the new value.

If the calculation is a stored calculation, the record it is part of will be marked as changed (thus it'll be in the list of edited records) and will be part of next save action.

Aggregation

An *aggregation* is a value derived from a set of records in a table, they are computed based on a set of values in a column. Similar to calculations, aggregation metadata is stored with the solution, but are based on a table. Aggregations can count, sum, maximum, minimum, and average.

Variable

A *variable* can be defined and holds data at runtime, but is not tied to any database column. Variables can be used as a dataprovider on a form, and can be created in either form scope or global scope.

Quick Start Video