

New in this release

In This Chapter

- [Highlights](#)
 - [First Experience:](#)
 - [Tools for the Developer in Servoy Developer:](#)
 - [Extensions & updated to the Solution Design feature set:](#)
 - [Deployment Features:](#)
- [Servoy Developer](#)
 - [First Launch Experience](#)
 - [Servoy Developer UI](#)
 - [Form Editor](#)
 - [Script Editor](#)
 - [Search support](#)
 - [CSS Editor](#)
 - [Solution Explorer view](#)
 - [Properties view](#)
 - [Profiler view](#)
 - [New Command Console](#)
 - [Servoy Developer script access](#)
 - [Debug Perspective](#)
 - [Build process](#)
 - [Miscellaneous](#)
- [Solution development](#)
 - [Behavior changes](#)
 - [Form Design](#)
 - [Drag 'n' Drop](#)
 - [Solution Model](#)
 - [Client Design mode](#)
 - [Designtime API](#)
 - [Runtime API](#)
 - [Web Client specific](#)
 - [Plugins & Beans](#)
 - [Miscellaneous](#)
- [Deployment](#)
 - [Public Java API](#)
- [Miscellaneous](#)

Highlights

In this release, in which we focused on Developer Productivity and First Experience, these are the main new features:

First Experience:

- Cleaned up the User Interface of Servoy Developer
- New design for the Welcome screen that is displayed on first launch after installation
- New Start Page when opening Servoy Developer with the latest news & useful links
- Completely overhauled All-in-One installer

Tools for the Developer in Servoy Developer:

- Brand new Form Designer
- Brand new CSS Editor
- Completely new JavaScript build system, based on extended JSDoc support
- JavaScript Refactoring support

Extensions & updated to the Solution Design feature set:

- TableView row styling through CSS StyleSheets
- Conditional styling of Forms & Elements through a new onRender event
- New Windowing API
- Extended Inheritance model
- Improved Drag 'n Drop event model

Deployment Features:

- Completely overhauled All-In-One installer
- Added support for installing the Servoy Application Server as a service for all platforms

Servoy Developer

First Launch Experience

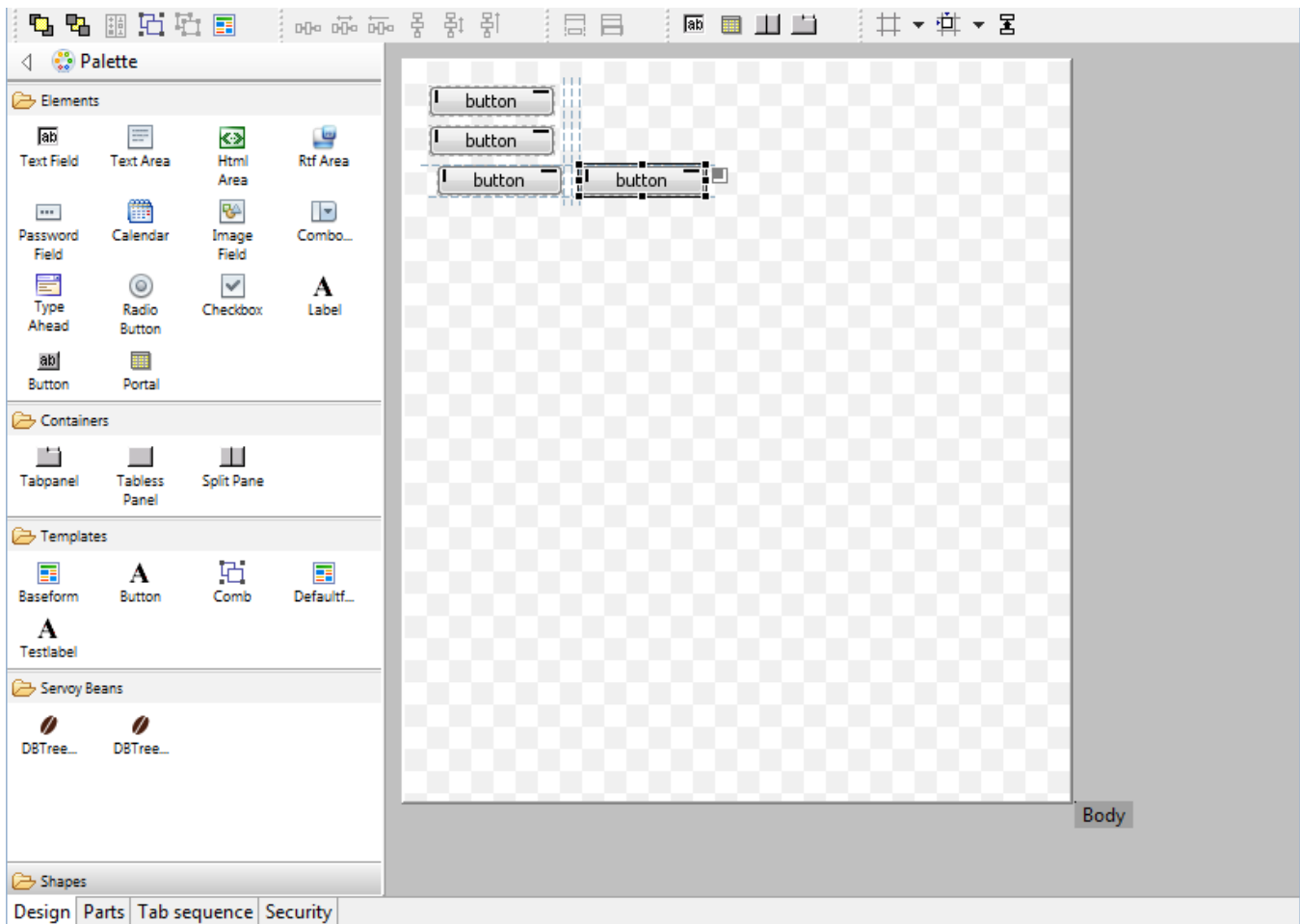
- Servoy Developer will launch maximized on first launch

Servoy Developer UI

- CVS team support has been removed (can be added again when needed through standard Eclipse update mechanism)
- Java related features of Eclipse have been hidden from view
- Toolbars related to Form editing have been inlined into the Form Editor
- Added Color/font chooser Toolbar
- Reordered default toolbar positions to achieve a more steady ToolBar when switching between different types of Editors
- Preferences:
 - Made then "*Window > Preferences > Servoy > Team settings*" all selected by default
 - Removed "*Window > Preferences > Servoy > Services*" as they are not relevant within Servoy Developer
 - Removed the obsolete "*Window > Preferences > JavaScript > Console*" preference page
- Hidden several not usefull Menultems, like File > Convert line Delimiters
- Moved Menultems from Elements menu under the Actions Menu on the MenuBar
- Moved the Debug Client Launch Menultems from the Actions menu to the Run menu
- Removed custom "New Form" option in the main toolbar and moved the New options into the default Eclipse mechanism for "New ..." operations (File > New or New button on toolbar or Alt-Shift-N keyboard shortcut)
- Removed Classic Search view: Search can be triggered through keyboard shortcuts as well as though buttons in the main toolbar area and the menubar
- Renamed "Profiler View" to just "Profiler"
- Hidden the Profiler view by default (Can be added through Window > Show view > Other > Servoy > Profiler)
- Hidden Console view by default: gets added automatically when starting a Debug Client
- Moved Tasks view next to Bookmarks view
- Improved enabling/disabling of Solution Explorer nodes, menultems & ToolBar buttons based on the current context, for example the enabled state of the Debug Client launch buttons
- Better visual feedback on which nodes are enabled/disabled in the Solution Explorer
- Cleaned up and aligned the Context menu's of the Form and Method Editors, to show only relevant entries and to follow the same menuitem order
- Removed the obsolete Script Debug Log view

Form Editor

The Form Editor in Servoy 6 has gotten a major overhaul. The main new features are a configurable Palette, Alignment Guides for placing, moving & resizing elements and many keyboard shortcuts for new and existing functions. Also the Form editing toolbars have been inlined into the editor, instead of being part of the main toolbar area.



Palette

- Supports Drag 'n' Drop of elements
- Supports "click select - click place" way of placing elements
- Supports placing the same element multiple times by holding down the Control-key while 'click select - click place' items from the Palette
- Servoy-aware beans automatically show up in a "Servoy Beans drawer"
- Java Swing components (JXxxx-named beans) are in a separate drawer, hidden by default
- Templates automatically show up in the Templates drawer
- Templates can be applied to existing elements while dragging by dropping them while hovering over existing elements in the Form Editor. Use *Control* modifier to do a place, instead of an apply
- The content, display and behavior of Palette is fully customizable:
 - New drawers can be created from Templates
 - Order of drawers and elements can be altered,
 - Drawers and elements can be made visible or hidden.
- The Palette can be inlined in the Form Editor (by default) or can be added to the Perspective as a regular view. In the latter case, the Palette will be automatically removed from the Form Editor. When the view is removed from the Perspective, the Palette will appear in the Form Editor again automatically.
- Newly created elements have better defaults settings for their properties
 - Containers (Tabpanel, Tableless Panel & Split Pane) are transparent by default (so the transparency setting of the form controls the behavior)
 - Checkbox & RadioButton fields transparent by default
 - Default size of TextArea HTMLArea & RTFArea
 - Media field disabled by default
 - HTMLArea and RTFArea not editable by default

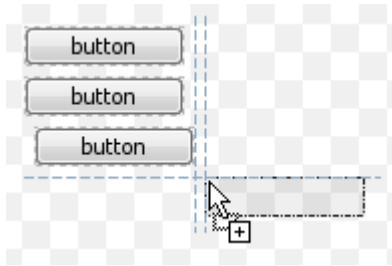
Alignment Guides

The Form Editor in Servoy 6 provides a sophisticated mechanism for aligning newly placed, moved or resized elements. The snapping to grid and the display of the grid have been disabled by default in favor of the new alignment guide functionality.

The new alignment snapping guides are like the grid snapping guides, but the snapping is not based on a fixed grid, but on proximity to existing elements of the Form and/or the Form edges.

The snapping guides offer three configurable offsets (which both work vertically and horizontally) based on proximity to other elements and Form edges and one configurable indentation size based on proximity to another element.

Optionally, Smart Anchoring can be turned on: Smart Anchoring will automatically set the anchoring properly when aligning an element to one of more edges of the Form



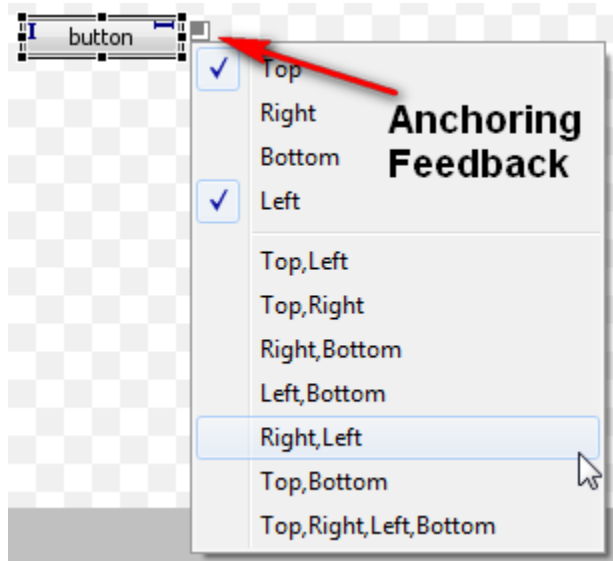
The color of the indicators can be configured through the Form Editor preferences (Window > Preferences > Servoy > Form Editor)

Custom Guides

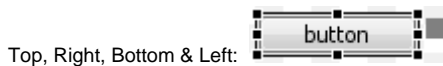
It is also possible to create custom snapping guides by right-clicking one of the rulers (The rulers must be showing). These custom guides are temporary and only for the Form Editor on which they were created. A custom guide can be removed by selecting it in the Ruler and pressing Delete.

Anchoring Feedback

Also added is a click-able Anchoring feedback area on selected element. Clicking the feedback icon pops up a menu with options to quickly set the anchoring options



The feedback are on the selected element provides visual feedback on how the element is anchored. Examples:



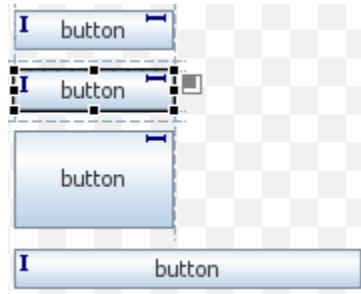
Same Width/Same Height functionality

New functionality has been added to make multiple elements have the same width and/or height. When two or more elements are selected, the Same Width and Same height functionality gets enabled:

- Available as buttons on the toolbar
- Available as entries in the context menu of the Form Editor
- Available as keyboard shortcuts (Shift-W & Shift-H)

When invoked, all elements get the same height and/or width as the element that was selected first.

When an element gets selected, all elements on the form that have the same width/height as the selected element also get a decorator to show that they have the same width/height as the selected element:



The feedback indicators can be enabled or disabled, see Misc. The color of the indicators can be configured through the Form Editor preferences (Window > Preferences > Servoy > Form Editor)

Keyboard shortcuts

The following keyboard shortcuts have been added:

- Send to Back/Bring to Front (resp. Control+] & Control+[beta 😊)
- Send backwards/Bring forward (resp.] and [)
- Group/Ungroup (resp. Control+G & Control+U)
- Same Height/Same Width (resp. Shift+H & Shift+W)
- Anchoring Top - Right - Bottom - Left (resp. Shift+., Shift+*, Shift++ & Shift+/-)
- Big step move (Control+Alt+Arrow keys)
- Big step resize (Shift+Alt+Arrow keys)

New Form wizard

- Previously used template will be remembered
- The Templates list will be filtered on Templates that contain form definitions (thus Templates created based on only a selection of Elements will be excluded)
- If a Template is selected and the StyleSheet property of the Form on which the Template was based is set, the StyleSheet property will be set

Misc.

- Created dedicated Group & Layout toolbars
- Improved visual feedback while resizing/moving elements using the keyboard
- When placing fields through the "Place Field wizard" button in the Toolbar of the Form Editor, the fields are placed in the order in which the dataproviders were selected in the "Select Data Providers" dialog
- Added a setting to sort the dataproviders that are columns from a Table in the order in which they are defined in the database (Window > Preferences > Servoy > Wizard options)
- Better placement of horizontally placed fields through the "Place Field Wizard" button in the Toolbar of the Form Editor
- The value of the Alignment Guide > Small Offset preference is used for the distance between the elements in the "Leftward" and "Upward" distribution actions
- Added option to toggle the display of the Rulers (default: on)
- Added option to toggle the display of "Page break" lines (default: off)
- Added option to toggle the visibility of inherited elements that don't have any overridden properties (default: show)
- Added the drawing of the Form border
- Offset the Form inside the editor viewport, for drawing the Form Border
- Visual feedback of Forms being transparent
- Elements marked as invisible are displayed semi-transparent
- Elements marked as invisible do not generate an "Element in form "...." is outside the bounds of form." builder marker when placed horizontally outside the bounds of the form
- Added "Element Placement Guide Options" shortcut on the toolbar to quickly toggle the Alignment mechanism to use for the current editor:
 - None
 - Grid Guides
 - Alignment Guides (default)
- Added "Visual Feedback Options" shortcut on the Toolbar to toggle the Form Editor feedback preferences for the current editor:
 - Selected Element Anchoring Indicator (default = true)
 - Selected Element Alignment Indicator (default = true)
 - Selected Element Same Size Indicator (default = true)
 - Grid (default = false)
 - Page Breaks
 - Rulers
- Added support for quick pan of viewport using SpaceBar+MouseDown+MouseMove (allows for quick scrolling the viewport if the Form is larger than the Form Editor area)
- Added forced lasso select for layered elements through Shift modifier when starting the select using the mouse
- Added preference for the lasso select mechanism: the ability to toggle between the mode where only elements that are fully enclosed by the lasso are selected, or any element which is partly within the lasso is selected
- Form part height can be set by double-clicking the part label
- Form parts can be resized using the keyboard
- When resizing Form parts using the mouse, if the Control modifier is pressed while the resize takes place, anchored elements also are resized based on their anchoring settings
- Form (parts) can be resized using the mouse by dragging their edges

- Form and Form Part can be resized in both dimensions at the intersection of the Form's right edge and the Form part bottom edge
- While hovering over the Form part label, a vertical resize cursor is shown
- Added location (x & y and x1 & y1) and size feedback in the status bar of Servoy Developer for the selected element
- Added "Revert Form" option to Context menu of the Form Editor, to undo all changes since the last Save at once (No undo possible!)
- Added "Preferences" option in the Context menu of the Form Editor
- Reorganized entries in the Context menu, including grouping related action into submenu's
- When pressing the Tab key while focus is on an element in the Form Editor, the next element in the tab sequence is selected
- The preferred size of beans is used when placing beans on a Form
- Support for dragging Forms from the Solution Explorer directly onto a Form in the Form Editor. The dragged Form can be dropped on existing container elements (TabPanel, SplitPane or TablePanel) or straight onto the Form in the Form Editor, in which case a new TabPanel is created with the dropped Form as the first tab
- When placing a new Tabpanel through the new Tabpanel button on the toolbar, the Tabpanel is dimensioned to the dimensions of the largest form, also taking into account the Form's border and navigator settings
- Splitpane elements get a name prefixed with split_ when placed on a form
- Splitpane elements render like a Splitpane in the Form Editor, instead of like a Tabpanel
- Tableless panel elements get a name prefixed with tableless_ when placed on a form
- Tableless panel element renders like a empty container element, instead of like a Tabpanel
- Visual appearance of elements representing Tabs on a Tabpanel changed to let the element better resemble a tab, especially on transparent Tabpanels
- Drag & Dropping a method from the Solution Explorer onto a field in the Form Editor sets the method as the field's onAction event handler
- Improved inline editing of Labels & Buttons in the Form Editor. Proper support added for Control-X/C/V keyboard shortcuts
- Added more finegrained layering controls: besides the already existing "Send to back" and "Bring to front", which would send the selected elements all the way to the back or front, the new "Send backwards" and "Bring forwards" move the selected elements only one layer down or up.
- Labels placed through Place Fields wizard also get their name filled, if the "Set name" checkbox is checked
- Elements in the Outline View linked to the Form Editor are grouped by Type of element

Script Editor

For Servoy 6 the goal was to introduce automated refactoring support for JavaScript. In order to be able to do automated refactoring, the refactoring mechanism needs to be able to analyse all the JavaScript code and determine exactly what every bit of code is, what it's scope is and how it is related to other parts of the code.

In order to achieve this, the entire builder process for the JavaScript source has been rewritten, to better parse and analyse all the JavaScript source code and infer typing information from it.

The result of the effort is that:

1. Servoy now offers automated JavaScript refactoring support
2. At design-time all code is being validated and where needed builder markers are generated (which can be ignored or result in Errors or Warnings based on configuration, see [Build process](#))
3. Code-completion has improved drastically
4. Call Hierarchy support on methods
5. Search for References support
6. JavaScript Search support

In order for all this to work, the builder process needs to be able to infer typing information from the JavaScript source. With JavaScript being a weak-typed and dynamic language, this information can't always be automatically inferred from the source-code itself.

An example of such a code structure is a JSDataSet. A JSDataSet created based on a query, cannot be automatically analysed @designtime to see which columns it will contain, thus references to the columns will generate builder markers. The same goes for dynamically inside code creating XML/XMLList objects or JavaScript Objects.

Other scenario's are functions that return a generic type like JSRecord or JSFoundset, while the context in which they are called gives then additional properties (in the case of these examples all their dataproviders).

The typing information that cannot be inferred from the source-code itself can be supplied using JSDoc. This can be done for in JSDoc comments above the declaration of functions and variables. In order to supply all the relevant information the JSDoc support within Servoy Developer has been extended as well.

To help identify the area's where the code cannot be automatically analysed, the improved builder process will generate builder markers for all parts of the code where it cannot automatically analyse the code and find all the relevant information.

The majority of the scenario's where code cannot be fully analysed fall into the following categories:

- Calls to functions for which the parameters have not been defined at all (the arguments object is used to access the parameters).
- Calls to functions for which the JSDoc info for the parameters is missing or incomplete
- Calls to functions for which the return type is not specified through JSDoc
- Use of scripting API that returns a generic type (for example JSRecord or JSFoundset)
- Use of dynamically objects like XML, XMLList, Object of JSDataSet
- Dynamic object property access (for example forms['myFormName' + instanceId])

While most scenario's can be solved by providing the right information through JSDoc, some code construct cannot be resolved through JSDoc. In this case dynamic property access can be used to suppress the builder process to generate builder markers:

```

var id = 1;
var x = {};
x['property' + id] = 1;
id++;
x['property' + id] = 2;
var y = x['property1'] + x['property2']

```

In order to take full advantage of the new refactoring features in Servoy 6, it is vital to resolve the markers by providing the required information through JSDoc. However, through the preferences, the warnings can also be ignored (they are just warnings, which not necessarily indicate errors in code!), but the usefulness of the refactoring functionality will diminish as code refactors will not be complete.

Refactoring support

Added support for automated JavaScript source code refactoring. The following refactoring options are available:

- *Rename...*: Renames variables (global, form and local (inside functions)), functions and function parameters. The rename refactor will change the name both for the declaration of the object to be renamed as well as all references to the object.
- *Extract Local Variable...*: Extracts the selected value within a function's source to a local variable (variable inside a function declaration). All occurrences of the selected value within the function's source are replaced by a reference to the new variable
- *Change Method signature...*: Alters the signature of an existing method and updated all calls to the method to use the new signature. Newly introduced parameters can be given a default value, so all calls to the method can call the method with the default value for the new parameter.
- *Extract Method...*: Creates a new function based on the selected code within a function's source and replaces the selected code with a call to the newly created function.
- *Introduce Parameter...*: Takes a hard-coded value inside the source of a function, creates a new parameter for it, replacing the hard-coded value with the parameter and also update all calls to the method to send the previously hard-coded value as value for the newly created parameter
- *Inline...*: Replaced a method call with the code of the called method

The Refactor functionality can be accessed in the Script Editor through the context menu of the Script Editor (Refactor menu) or through the keyboard shortcut Alt-Shift-R. when performing a refactoring operation, a UI will be show that allows to specify the required information and to preview all the changes that will be made in all the code.

JSDoc support

- Code completion inside JSDoc for supported JSDoc tags and on Types (type '@' first, then Control-Space). Supported tags:
 - @AllowToRunInFind
 - @author {userName}
 - @constructor
 - @deprecated
 - @example
 - @param {Type} name mandatory name property
 - @param {Type} [beta:name] optional name property
 - @private
 - @protected
 - @return {Type}
 - @see
 - @since
 - @SuppressWarnings (warning)
 - @throws {Type}
 - @type {Type}
 - @version
- Support for automatically creating the basic JSDoc tags for an existing function: See context menu of the Script Editor > Source > Generate Element Comment (Alt-Shift-J)
- Support for inline JSDoc to provide type information for objects:

```

function demo(){
    /** @type {JSFoundset<db:udm/contacts>} */
    var x = databaseManager.getFoundset('udm', 'contacts');
    x. //Code-completion here will know that x is a FoundSet on table Contacts in the UDM database, thus
    will include all columns, calculations and aggregates
}

```

- Support for the @deprecated tag: When a variable or function is marked as deprecated any reference to the object will generate a builder marker
- Support for the @private tag: Hides the object for access from outside the current scope, including child scopes. Accessing the hidden object from outside the scope in which it is defined will generate a builder marker
- Support for the @protected tag: Hides the object for access from outside the current scope. Accessing the hidden object from outside the scope in which it is defined will generate a builder marker
- Support for typed Arrays:
 - @type {String[]}
 - @type {Array<String>}
 - @type {Array<Byte>}
- Support for types Objects:
 - @type {Object<String>}
 - @type {Object<Array<String>>}: equivalent of {[beta:'one', 'two', 'three'], [beta:'four', 'five', 'six'], [beta:'seven', 'eight', 'nine']}
- Support for typing JSFoundsets and JSRecord:

- @type {JSFoundset<datasource>}
- @type {JSRecord<datasource>}
(a datasource string is build up like this: "db:/{serverName}/{tableName}", for example "db:/udm/contacts")
- @type {JSFoundset<{column1:String,column2:Number}>}
- @type {JSRecord<{column1:String,column2:Number}>}
- Support for typing as a RecordType:
 - @type {JSDataSet<{name:String, age:Number}>}
 - @param { {name:String, age:Number} } person A JavaScript object with a name and age property representing a person
- Support for optional properties on RecordType. There are two different syntaxes allowed:
 - Square brackets around the property name: @param {{sDocID:String, [beta:sTemplateID]:String}} oArg
 - An equal-sign behind the type: @param {{sDocID:String, sTemplateID:String=}} oArg
- Support for OR typing of parameters: "@param {String|Number} name description"
- Support for typing parameters as Objects with certain properties:

```
/**
 * @param {Object} person
 * @param {String} person.name
 * @param {String} person.email
 * @param {Number} [person.age] optional "age" property
 */
function processPerson(person) {
    application.output(person.name);
    application.output(person.email);
}
```

- Support for rest parameters: Allows to indicate through JSDoc that a function can take unlimited trailing arguments of the specified type:

```
/**
 * @param {...String} someExtraStrings One or more additional String can be send into this function
 */
function methodWithRestParams(someExtraString){}

function test() {
    methodWithRestParams('one', 'two', 'three', 'four');
}
```

- Support for typing as an instance of a Form. Can also be used to indicate that the Form must extend the specified Form:
 - @type {Form<mySuperFormName>}
- Support for the so-called "AnyType"
 - @param * myParam A parameter that can contain any type of value.
- Beans are made available as types in CodeCompletion, to be used for type checking and JSDoc
- Support for @AllowToRunInFind on functions to indicate that the the function should be executed on event handlers while in FindMode: Eliminates the //controller.search() workaround. Note that that running JavaScript while in FindMode has it's limitations and not all behavior is 100% defined. More details will follow.
- Support for @constructor tag, to indicate that a function declaration is to be used as a Constructor. Functions marked as Constructor also display differently in the Script Outline view
- Automatic JSDoc @type setting on global and Form variables
- Ability to update the type of a variable by updating the JSdoc @type tag value AND the default value
- Added specific JSDoc context for Templates (Window > Preferences > JavaScript > Editor > Templates), besides the already existing JavaScript context.
- Added dedicated RuntimeXxxx types for all runtime element types and for Forms. For example RuntimeLabel, RuntimeTextField and RuntimeForm. These types can be used in JSDoc, but also in scripting to check if an object is and instance of a certain type, using the instanceof operator. Note that RuntimeForm and Form are the same, RuntimeForm was added for consistency. RuntimeComponent is the base type for all default elements that Servoy Ships with
- Removed Strict Mode option from the JavaScript Preferences: the additional warnings that where generated with Strict Mode enabled are now part of the extended JavaScript Error/Warnings preferences and thus can be enabled/disabled on individual basis

Code completion

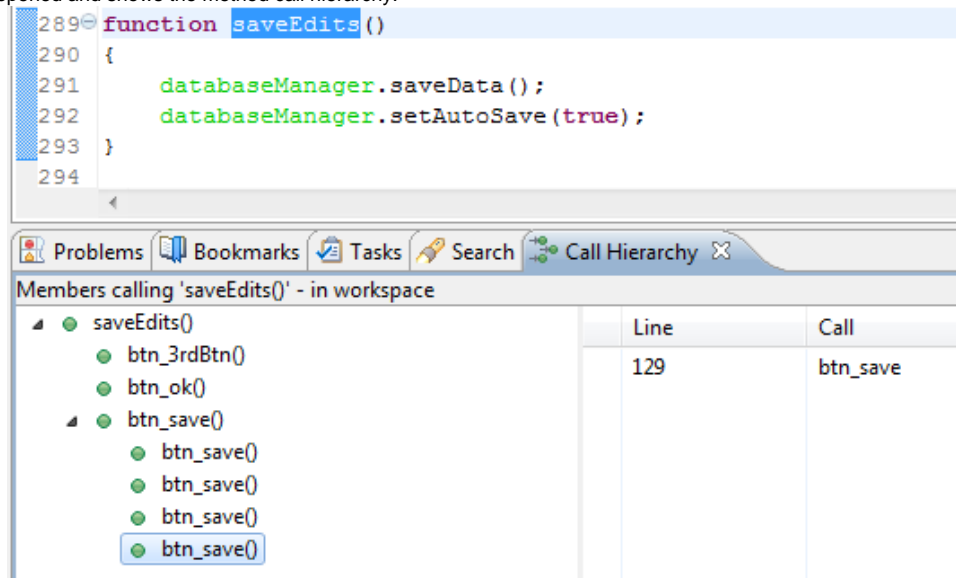
- Overall much better context aware code-completion, due to improved type inference and extended JSDoc support
- Support for code-completion on inline Java: The support is on Class level, not Package level:
 - Packages.java. //Code-completion here won't yield anything
 - Packages.java.lang.System. //Code-completion here works
- Deprecation decorator on deprecated objects:



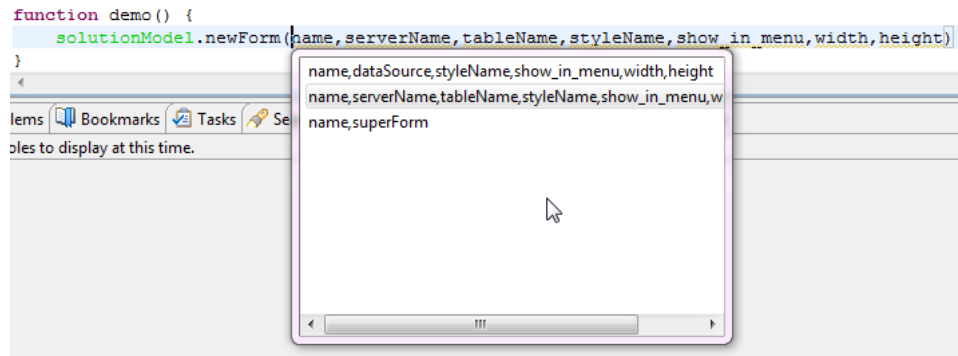
- Type Icons in code-completion popup
- Method return type info in code-completion popup
- Originating Scope info in in code-completion popup
- Scripting overloading support

Misc.

- Support added for "Mark Occurrences": highlights all occurrences of a the selected work in the Script Editor. Also see Window > Preferences > JavaScript > Editor > Mark Occurrences
- Support for Call Hierarchy on Methods: When invoked on a function declaration or reference (through keyboard shortcut Control-Alt-H), the Call Hierarchy view is opened and shows the method call hierarchy.



- Better support for Code completion on advanced JavaScript code structures
- Added support for Content Assist - Parameter Hints (Control-Shift-Space): Shows a tooltip with the function's parameters with their types, when invoked withing the parameter area of the function declaration "...". When the function is overloaded, first a popup is shown to select a specific variant:



```
function demo() {
    solutionModel.newForm(String name,String serverName,String tableName,String styleName,Boolean show_in_menu,Number width,Number height
    name,serverName,tableName,styleName,show_in_menu,w
    name,superForm
})
```

- Fixed expand/collapse behavior when saving the Script Editor after adding new variables or functions
- Made it possible to collapse/expand the multi-line content of variables (multi-line Strings, XML, Functions and Objects)
- Added support for putting comments in script files (.js files) outside functions and outside JSDoc tags linked to variables and functions. Note that the comments are lost in the following scenario's:
 - When using Servoy Repository for team development: The Servoy Repository stores individual variables and functions, so anything else in the .js file is lost. Solution: use SVN instead which is file-based
 - When exporting the Solution to a .servoy file: just like the Servoy Repository, the .servoy export format only contains the individual variables and functions. When the .servoy file is used to move a Solution from development into test/Q&A/production, there should be no issue loosing the comments. However, if the .servoy export is going to be used to transport the source into another workspace /development environment, the loss of the extra information could be unwanted. In this case the solution would be:
 1. Share sourcecode between different developers by allowing (anonymous) checkout from the source repository used (for example SVN).
 2. Use the native Eclipse Archive file export/import functionality to export and later import an entire Eclipse project (a.k.a. Servoy Solution)
- Fixed indentation of code when pasting code into the Script editor
- Fixed indentation of code when moving code using the "Move Sample" and "Move Code" buttons in the Solution Explorer view
- Prevented Typing info to get inserted by "Move Code"
- Show Parameter Hints after "Move Code"
- Improved JSDoc tooltip hover behavior over functions: moving the mouse from the function to the tooltip doesn't hide the tooltip, but instead gives it focus
- Removed the "Paste to JavaScript console" option in the Context menu of the Script Editor
- Hidden non-functional menu items "Open Type Hierarchy" and "Quick Type Hierarchy" from the context menu of the Script Editor
- Added Eclipse Spelling support inside the Script Editor on comments (singleline comments, multiline comments and JSDoc comments): See Window > Preferences > General > Editors > Text Editors > Spelling
- Save Actions for the JavaScript Editor: allows automatic triggering of formatting JavaScript sourcecode when the editor is saved. Currently, the Save actions can only be accessed through the Properties of a Solution: Accessible through the context menu or keyboard shortcut Alt-Enter on the Active solution node of Module nodes in the Solution Explorer. See JavaScript > Save Actions
- Removed the non-functioning "Run As" and "Debug As" menu items from the context menu

Search support

JavaScript source code only

- Added JavaScript support in Search (Control-H)
- In Search the Enclosing Projects now works properly: it searches within the selected solution/module and it's child modules.
- Added support for Search > References/Declarations in the context menu of the Script Editor

General

- Added "Search for References" on many of the objects in the Solution Explorer:
 - Database Servers
 - Tables
 - Methods
 - Variables
 - Forms
 - Relations
 - ValueLists
 - Plugins

CSS Editor

A new CSS StyleSheet editor has been incorporated into Servoy Developer, with the following features:

- Syntax coloring
- Code Completion, based on supported CSS subset that Servoy supports
- Outline view for the StyleSheet
- Properties view for the selected StyleClass
- Font & Color chooser available in main toolbar area

Others

- Added support for CSS hex color shorthand notation: #FFF instead of #FFFFFF
- Included a new and improved Sample StyleSheet in the New StyleSheet wizard

Solution Explorer view

- Activate a Solution by double-clicking the node for the solution to activate under the "All solutions" node
- Added support to rename the Active Solution (added item in the Context menu of the Active Solution node)
- More contrast between enabled and disabled nodes (for example when editing Calculations)
- Hide 'selectedrecord' & 'relations' node under the Form node of Forms that are not bound to a table
- Different icons under the "All solutions" node for solutions of type module or solution, with special decorators for Web Client or Smart Client only
- Warning & Error decorators on the Icons of each node if there are any builder markers (Errors and/or Warnings) on the object the Node represents or on any of its children
- Subnodes now ordered alphabetically
- Automatic expand of the Resources > Database Servers node on startup if there are invalid servers
- Added "Search for references" to the context menu of many of the object nodes (see [Search support](#)):
"Search for references" tried to find all references to the given object, both inside the JavaScript code as well as the definitions of objects like Forms, Relations, ValueLists etc. Due to the dynamic nature of the JavaScript language, the matches within the JavaScript code might not be complete and/or not valid in the case of code that uses dynamic code constructions
- Added preference to define the double-click behavior on Form nodes and on the globals node. See Window > Preferences > Servoy > Solution Explorer
- Improved "Move Code" support:
 - Properly indent the moved code
 - Removed parameter type info from the moved code
 - Show Parameter Hints over the moved code
- Renamed menuitem "New server" to "Connect to existing database" in the Context menu of the Database Server node of the Solution Explorer
- Moved "Create new Database" option from the Context menu of existing individual Database Servers definitions to the Database Servers node in the Solution Explorer
 - As before, the Create new database is only supported on PostgreSQL and Sybase databases
 - The option's are disabled when there are no Database Servers defined yet for PostgreSQL or Sybase DB's
 - When there are Database Server connections to multiple database servers, a popup is shown first to select on which database server to create the new database
- Improved Solution Explorer preferences page (Window > Preferences > Servoy > Solution Explorer)
 - Included Alignment and List Viewer options
 - Improved display of globals and Form Node double click options
- On forms that are not linked to a datasource, the selectedrecord and relations nodes are removed
- Moved the ServoyException node under the Application node: ServoyException is not a top level scriptable object, it is only obtained through the solutions onErrorHandler. Therefore, it is moved under the Application node, where the other similar types are already located
- Added "Properties" menu item to context menu of active solution and its modules, to gain access to the Project properties. Useful for Project specific Preference settings
- Added support for drag 'n' dropping files from outside Servoy Developer directly into the media library
- Added support for drag 'n' dropping media objects onto:
 - Form Editor: supports images, dimensions of the image is used when placing the image
 - Script & CSS Editor: the media url gets inserted
- Hidden the options "Go Home", "Go Back", "Go Into", "Refresh", "Expand" and "Collapse tree" from the Context menu in the Solution Explorer by default, can be turned on through the Solution Explorer options menu and preferences again
- Improved "Link with Editor" support i.e.w. the Form Editor: clicking a named element in the Form Editor will now select the element in the Solution Explorer
- Different icons for private and protected method in the Solution Explorer

Properties view

- Color properties show the HEX value, instead of the RGB value
- Color properties can be directly edited. Supported values are Hex values, RGB values and named colors (i.e. black, red etc)
- Font properties can be directly edited
- When setting a property to an invalid value, an Error dialog pops up, instead of a message in the status bar and a silent revert to the old value
- Dataprovider properties that are unresolved now show the dataprovider name that cannot be resolved
- StyleName and StyleClass property now have a button to open the StyleSheet/jump to the StyleClass declaration
- The tabSeq property has a button to open the "Tab sequence" tab of the Form Editor
- The Method Selection dialog used to assign methods to Commands and Events has an "OK & Show" button to assign the selected method to the Command/Event and open the assigned method in the Script Editor in one go
- Ability to select -none- as dataprovider when editing a dataprovider property
- Added option to select standard tags over a relation (where applicable) in the Text property editor
- Overrides of inherited property values show -OVERRIDE- next to the value
- Options to create either a private, protected or public method when creating a new method for an event/command handler. In case of the private or protected option, resp. the @private or @protected JSDoc tag is set in the JSDoc of the newly created method.
- Better context aware entries in the dropdown of labelFor property: For labels in the body part of Forms in TableView mode, the dropdown will include all named elements that are also in the body part and that don't have the labelFor property set yet.

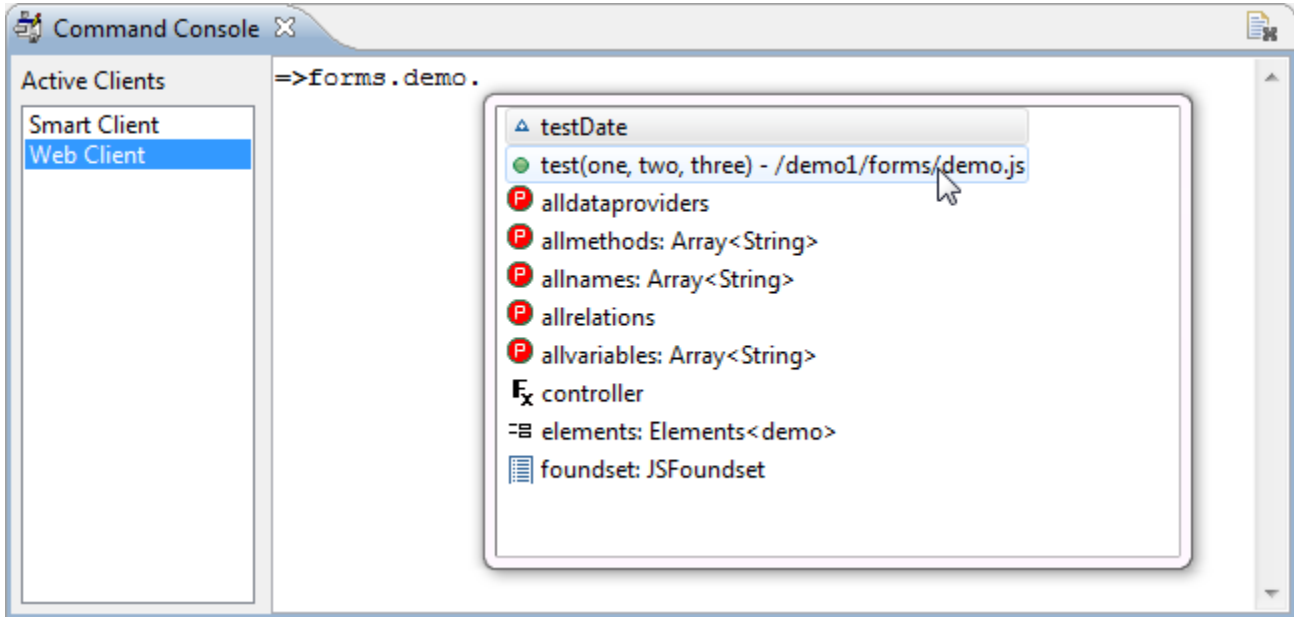
Profiler view

- Hidden from the Servoy Design perspective to improve the initial experience
- Made customized column width settings by the user persistent between session
- Added Query execution timing info to the Profiler view
- Reduced the info in the "Source File" column to only show the name of the .js file and the Form name where applicable
- Added more scope related info inside the view, like the solution or form name for methods

New Command Console

Servoy 6 supports a new Command Console view. The command console view is like the Interactive Console view, but with the following differences:

- The Command Console lists all the running Debug Clients (Smart Client, Web Client and Headless Client) and the user can choose one of the running Debug clients to interact with
- The interaction with the Debug Client can start as soon as the Debug Client is started. No need to set a breakpoint first and hit the breakpoint like in the Interactive Console
- The scripting scope in the Command Console is a self-contained solution-level scope, contrary to the Interactive Console, which has the current method call stack as scope



Servoy Developer script access

In Servoy 6 it is possible to interact with Servoy Developer from both Debug Clients and the new Command Console. The available methods are:

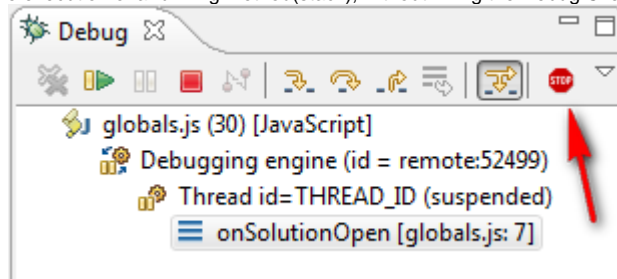
- `servoyDeveloper.openForm(Object)`: opens the specified Form (JSForm object or Form name) in a Form Editor in Servoy Developer
- `servoyDeveloper.save()`: persists all changes made through the solutionModel in a Debug Client or the Command Console into the project in the local workspace.
- `servoyDeveloper.save(Object)`: same as above, but then only persist the specified Form (JSForm object or Form name)

The `.save(...)` methods save the changes to the workspace and does the same as editing a Form in a Form Editor in Servoy Developer and saving the changes.

Needles to say, this functionality is available only inside Servoy Developer.

Debug Perspective

- Added a "Stop" button, to terminate the execution of a running method(stack), without killing the Debug Client:



- Ability to terminate the execution of a running method(stack) from within the Interactive Console by executing the code `!stop_current_script!`

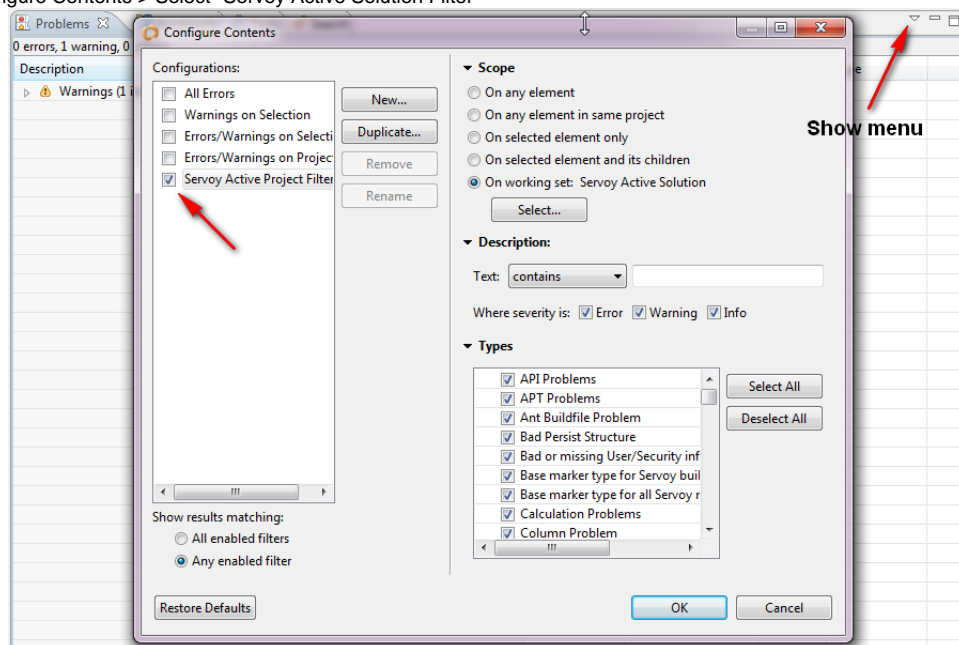
Build process

- Made all possible builder markers for JavaScript file configurable:
See Window > Preferences > JavaScript > Error/Warnings: Each possible builder marker on JavaScript files is listed and can be set to be ignored or to generate either an Error, Warning or Info marker
- Added QuickFix to create the unresolved method on builder markers for unresolved methods attached to events and commands
- Added "Element in form "...." is outside the bounds of form." builder marker for elements placed below the last Form part. Note: elements placed below the last form part on a form are not instantiated at runtime, thus are not in the elements array or visible on the Form

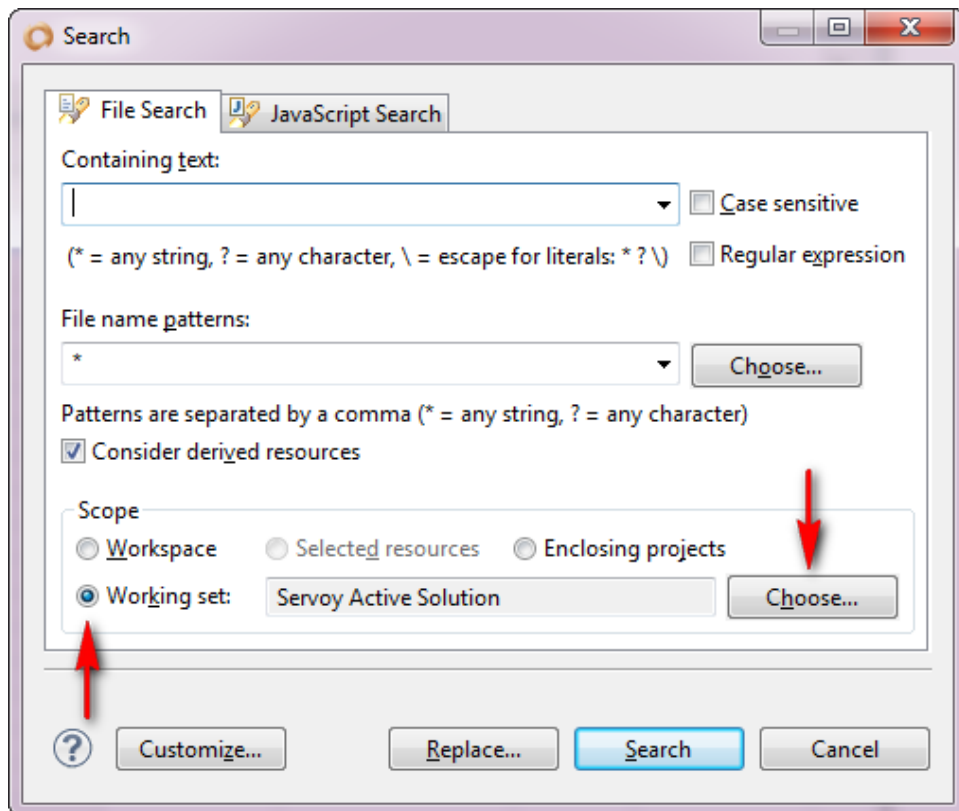
- Added builder marker to warn about dataprovider of type text with a small length that are attached to HTMLArea's or RTFArea's, as these type of fields generate more markup in the dataprovider than the user actually sees
- Added builder marker to warn about fields with a format setting that isn't appropriate for the type of the data provider: 'Element in form "..." has incompatible format with its data type (...) : "..."
- Added builder markers for use of deprecated scripting API
- Added builder markers for use of deprecated JavaScript variables and functions (see [Script Editor](#) > JSDoc support)
- Added builder markers for access to not declared objects or properties of objects
- Added builder markers for calling functions (global methods, form methods and scripting API) with the wrong number and/or type of parameters
- Added builder markers for variables and parameters that hide data providers, variables or parameters at a higher scope
- Added builder marker for labels with the labelFor property set to a non-existing element
- Excluded .js files contained in the Media library from the builder process
- Added builder markers for Database Server definitions of which the JDBC Driver class cannot be found
- Added @SuppressWarnings (warningIdentifiers) to suppress individual warnings from being shown: current supported types are "deprecated", "hides", "wrongparameters" and "undeclared". The @SuppressWarnings tag can only be applied in the JSDoc of functions
- Added support for modifying the BuildPath include/exclude settings on solutions (Solution Explorer > Solution's Context Menu > Properties > JavaScript > Build Path > Source): by default all .js files in the solution's directory are included, except .js files stored in the Media library. Modifying the excludes can be usefull when using the JSDoc plugin from Servoyforge, which creates a docs/ directory containing .js files as well. When not excluded from the Build Path, the builder process will include those .js files as well and validate them and generate builder markers (Error/Warning marker) for them.

Miscellaneous

- Added a Servoy Active Solution Working set: the workset functionality of Eclipse allows to limit the scope of operations like searching or display of builder markers. The Servoy Active Solution workset is a pre-defined workset that contains the Active Solution and all it's modules. The Servoy Active Solution working set can be used in:
 - Problems view: The contents of the Problems view can be filtered to only show the builder markers for the active solution. Click "View menu" > Configure Contents > Select "Servoy Active Solution Filter"



- Searching: In the Search dialog, the scope of the Search can also be restricted to the Servoy Active Solution workset:



- Servoy MarketPlace integration: Servoy MarketPlace can be opened in Servoy Developer, through Help > Servoy MarketPlace
- Added Servoy Admin page shortcut in the Help menu of Servoy Developer
- Added option to show the firstForm of the solution that is being activated (see Window > Preferences > Servoy > Form Editor)
- First form created in a Solution is automatically set as the Main form in the Solution properties
- Improved output to console of Javascript objects: {name: someValue} instead of [beta:object Object]
- Added predefined update url for JSDoc plugin (<https://www.servoyforge.net/projects/jsdoc>)
- Updated Servoy Developer to Eclipse Helios (3.6)
 - Option to export/import Preferences via generic export/import mechanism
- Reduced memory footprint of solutions in Servoy Developer by about 40%
- Added command line Solution Exporter (more info will follow)
- Added JUnit testclass for running JUnit tests contained in a solution within the JUnit framework (more info will follow)
- Made update of Calculations and Aggregates in the Replace Table action optional
- In the Debug Smart Client, a call to application.showUrl(...) will open the URL in a browser within Servoy Developer
- In Table Editor, when creating a new column containing the text "email" or "url" in the columnName, automatically set the type to "Text" and the length to resp. 254 or 2048 (based on the specifications for email addresses and urls)
- Added Copy action in columns rows in Table Editor, to quickly get the column name into the clipboard
- Added "Maximum prepared statements idle" property in Database Server Editor (was already exposed on the Admin page)
- Made "Link with Editor" work for Styles, Tables and Database Servers as well
- Added "FoxPro DBF" New server template
- Included the FoxPro DBF JDBC driver into the distribution
- Added ability to also externalize hard-coded Strings inside JavaScript code (See Externalize Messages dialog)
- Moved all Database Server management into Servoy Developer and disabled the functionality on the Servoy Admin page when running Servoy Developer
- Removed option to turn off Enhanced Security from the UI. enhanced Security is now the default mode. Can only be turned on by manually adding the relevant property to the servoy.properties file (see [Deployment](#))
- Images in the Media library are now opened with the Viewer which is registered for the image type. By default this will be the Image viewer Servoy supplies, but can now be overruled by the user, for example by installing another Image Viewer plugin, for example the QuickImage plugin for Eclipse: [<http://psnet.nu/eclipse/>]/[<http://psnet.nu/eclipse/>]
- Table Editor: support Control-Click to deselect the selected row and hide the Column Details
- Database Server Editor: Highlighted the JDBC Driver ClassName if it cannot be resolved, meaning that the specified class cannot be found in the available JDBC Drivers
- Added more options to synchronize I18N keys between the workspace and database in Servoy Developer
 - Read from DB: Option to sync the keys in the workspace (target) with the keys in the I18N table in the database (source)
 - Write to DB: Option to sync the keys in the I18N table (target) in the database with the keys in the workspace (source)
 - Both option provide the option to remove keys that are in the target, but not in the source
 - Both options are available through the context menu of the Resources > I18N files node of the Solution Explorer

Solution development

Behavior changes

- If a font in html_area is expressed in "pt" instead of "px" -> it will get smaller.

- Non-modal Dialogs in WC rendered inside the main window, instead of as new browser windows due to JSWindow implementation, allowing multiple modal and non modal in the same window
- In Smart Clients running under Java 6, Modal Dialogs (JSWindow of type MODAL_DIALOG and the dialogs displayed using the dialogs plugin) will be modal for their parent window (hierarchy). This means that when having multiple windows (JSWindow of type WINDOW) open, a Modal dialog opened in one window will only block that window, not the other Windows. This aligns the behavior with the Web Client behavior. Unfortunately, the ability to set the scope of dialog modality is only possible from Java 6 onwards. When running Smart Clients on Java 5 modal dialogs will block all windows.
- application.getTimestamp() in the Web client will return the timestamp based on the timezone of the Client and the time of the Server: Before it would just return the timestamp based on the time and timezone of the server.
- In the solutionModel when asking for a specific element on a form, if the element is not on the form, the super scope will be inspected and return the element if the element is inherited. For the plural getElements() type of functions, an optional paramter has been added to include the inherited elements in the return value or not.
- controller.recreateUI() will keep the existing x and y scrollPosition of the form being recreated.
- When adding a tableFilter on a column that already has a tablefilter on it, the newly applied filter will not override the existign filter, but the new filter will be appended, so both the existing and newly added tableFilter are in effect. To implement the old behavior, the TableFilters need to be created with a name and then when applying a new TableFilter the old one needs to be removed through scripting.
- When using labels with labelFor link to other elements in TableViews to control the TableView header rendering, the first label will now determine the height of the header
- The client that gets started as Authenticator during the login process isn't consuming a license any more
- Enhanced Security is now the default mode. The UI in Servoy Developer and in the Application Server to disable it has been removed. It is only possible to disable Enhance Security through setting the relevant property in the servoy.properties file: "servoy.application_server.enhancedSecurity=false"
- plugins.rawQuery.executeStoredProcudure() now returns the last resultset returned by the stored procedure, instead of the first
- The position and dimensions of JSWindow's (Windows and Dialogs) are now persisted over client sessions
- Default alignment for text in TableView & Portal headers brought inline (Centered)
- Aligned default border and alignment for all elements between Smart Client and Web client
- Due to added support for vertical and horizontal alignment on labels and buttons in the Web Client, any custom HTML displayed in a non-editable HTMLArea in the Web Client will be offset from the top by 50% of the height of the element, if the vertical alignment is left on Default, with means center. Set the vertical alignment to Top to get the HTML to display correctly again.
- Due to the added support for an editable HTMLArea in the Web Client, end users will now see an HTML Editor in the Web Client if the HTMLArea is left enabled in the Solution Design, whereas in previous versions it would show the raw content of the dataprovider.
- BugFix: A bug was fixed that allowed making elements visible in scripting on which there was a security constraint that made the element invisible. The new behavior is that security takes preference over scripting for "visibility", like was already the case for "enabled"
- Tooltips in the Web Client are not shown immediately, but after a short delay, to synchronize the behavior between the Smart and Web Client. Note that through the UIProperties APP_UI_PROPERTY.TOOLTIP_INITIAL_DELAY & APP_UI_PROPERTY.TOOLTIP_DISMISS_DELAY the initial show delay and the dismiss delay are now controllable by the developer in both the Smart and Web Client
- DeepLink arguments are now also applied to the onOpen method of the login solution. As a result of that, the application.getStartupArguments() method is deprecated, as it's no longer needed
- Related null-searches will no longer return records without related records (for example in find mode: orders_to_details.detail_date='^=' will only return orders that have one or more detail records with null-detail_date and not orders without detail records), this change can be suppressed with setting servoy.client.relatedNullSearchAddPkCondition=false in the servoy.properties file.
- BugFix: When passing JavaScript Arrays into the Scripting API (Servoy Scripting API or plugins), previously empty Array positions were removed, not they are reserved
- Bugfix: JSFoundset.getRecordIndex(JSRecord) now returns -1 instead of 0 when the record is not found, to be inline with other functions that return -1 to indicate "not found"
- The passing of deepLink arguments to the onOpen methods of both the Login solution (if applicable) and the main solution has been normalized:
 - Proper supports for deeplinks that specify multiple values for both the predefined "argument" parameter as well as custom named parameters.
 - The difference between deeplinks using the full syntax ("arguments") or short syntax ("a") for specifying arguments has been hidden: short syntax will be automatically converted to full syntax.
 - The query arguments in the URL that represent the solutionName or the methodName are filtered out, so only the real arguments are send into the onOpen method

Examples:

```
http://domain:port/servoy-webclient/ss/s/{solutionName}/argument/1/m/{globalMethodName}/a/2/a/3/b/1/b/2/b/3 or
http://domain:port/servoy-webclient/ss?s={solutionName}&a=1&method={globalMethodName}&a=2&a=3&b=1&b=2&b=3 or
http://domain:port/servoy-client/{solutionName}.jnlp?a=1&a=3&b=1&a=2&b=2&b=3
```

all results in the following arguments applied to the onOpen method

```
arguments[0]: 1
arguments[1]: {argument: [1,2,3], b: [1,2,3]}
```

Form Design

CSS support for grid row styling

The rowBGColorCalculation property of forms is deprecated and instead of this it's now possible to provide odd/even/selected row styling information through StyleSheets:


```

/* Generic Grid styling, styles the following:
 * - ListView row background
 * - TableView grid cell background, foreground & font
 * - Portal grid cell background, foreground & font
 *
 * The row styling classes can be used in combination with form and portal classes:
 * - form odd, form even, form selected, form.xxx odd, form.xxx even, form.xxx selected
 * - portal odd, portal even, portal selected, portal.xxx odd, portal.xxx even, portal.xxx selected
 */

/* Grid row styling: Applies to: Grid of forms in TableView and Portals */
odd, even, selected {
    background-color: #FFFFFF;
    color: #242424;
}

odd {
    background-color: #F5F7F9;
}

even {
}

selected {
    background-color: #d4d4d4;
    color: #FFFFFF;
}

```

The CSS grid row styling is applied to all elements in the grid rows. The deprecated `rowBGColorCalculation` property supported individual cell styling as well: this functionality is replaced conditional formatting functionality though a new `onRender` event on form and Element level. See "Conditional formatting" below for more information.

Conditional formatting

Starting from Servoy 6 there is a new `onRender` event added to Forms, Portals and Elements. The aim of this event is to allow changing display properties of supporting components just before they are shown or updated. As such it can be used for conditional formatting for example and it complements the static styling options (setting properties on elements of through StyleSheets, semi conditional Styling (Row styling, see previous item) and the runtime API of all elements.

On Form and Portals the event is fired not only for the Form/Portal itself, but also for all the standard Servoy Elements on the Form/Portal, if the element does not have its own `onRender` event handler. The Form/Portal level `onRender` event will not fire for beans.

The `onRender` event handler is called with a parameter of type `JSRenderEvent`, that provides the following functions:

- **getRecord():** the record of the rendered object
- **getRecordIndex():** the index of the rendered object
- **getRenderable():** the rendered object
The returned object is of type `Renderable`. A `Renderable` object can be an instance of a `RuntimeForm`, `RuntimePortal` or any of the other `RuntimeXxxx` elements.
The `Renderable` class exposes all the properties that can be set in the `onRender` event and also utility functions to get the rendering element type and its `dataProvider`.
The `Renderable` class is a generic class and some of the properties and methods are not applicable on the actual object being rendered. For example, if the object being rendered is an instance of `RuntimeForm`, the property `toolTipText` or the method `getDataProviderID` are irrelevant. When these are set/called anyway, they will fail silently
- **hasFocus():** whether or not the rendered object has the focus
- **isRecordSelected():** whether or not the record of the rendering object is selected


Any updates made in the `onRender` event to the rendering object are persistent within the Client session until changed through the runtime API of the element or in a consecutive `onRender` event. This means that in the `onRender` logic, both states of a property need to be handled. This means that if the `onRender` is used to set the `fgcolor` of a field depending if the `dataProvider`'s value is negative or not, the `fgcolor` needs to be explicitly set for both negative and positive numbers. When the same foreground property is also set in scripting and should overrule the `onRender`, the developer needs to take care of this inside the `onRender` logic

Example use: Making negative values in a column red and zero values orange


```

/**
 * Called before the form component is rendered.
 *
 * @param {JSRenderEvent} event the render event
 */
function onRender(event) {
  /** @type {JSRecord<db:/udm/orders>} */
  var rec = event.getRecord()
  if (rec.amt_tax == 0) {
    event.getRenderable().fgcolor = '#ff6600';
  } else if (rec.amt_tax < 0) {
    event.getRenderable().fgcolor = '#ff0000';
  } else {
    event.getRenderable().fgcolor = '#000000';
  }
}

```

 About performance: The onRender event will be fired often. It's therefore advised to keep the logic inside the onRender event handler method as optimized as possible, for quick execution. It's advised to refrain from calling any code outside the API of the JSRenderEvent or the Renderable class.

Inheritance model

The Inheritance model of Servoy has been enhanced to allow:

- Overriding all inherited Form properties
- Overriding all properties of inherited elements
- Overriding all properties of inherited Form parts



Behavior Change

Due to the extension of the Inheritance model a behavior change was introduced. When asking for elements and/or parts of forms that extend other forms, the inherited elements/parts are also returned in Servoy 6, whereas prior to Servoy 6, only the non-inherited elements would be returned and the (hierarchy of) super forms(s) had to be inspected

Misc.

- Added rolloverImageMedia support in TableViews
- Improved Sorting indicator on Grid columns behavior: the Sorting indicator is now always shown if a FoundSet is sorted and automatically follows the actual foundset sort. Columns that display a dataprovider on which the foundset is sorted get the sorting indicator. This means that if the Foundset is sorted on dataproviders that are not linked to columns, there will be no sorting indicator. Also when the foundset is sorted in-memory, there will be no sorting indicators
- Added multi-column sort behavior on grids (TableView's and Portals): by Shift-clicking a column header, the foundset sort is extended with a sort on the dataprovider of the clicked column. The multi-column sort is also reflected in the Sorting Indicators on TableView headers
- Right-click support on TableView headers through label linked to field using labelFor relation
- Ability to lock one or more columns on the left and/or right of a TableView in place (meaning they cannot be rearranged), by anchoring the element representing the column cell top and bottom. Can be combined with left/right anchoring for resizability of the column
- Support for number length formatting on number fields
- Support to hide the Form title from the Title bar in the Smart Client or the Tab name of the browser in case of the Web Client, by specifying "none" as value for the forms titleText property
- Added support for displaying HTML contained in the titleText property of non-editable HTMLArea's. Useful to show (multiple) href links in the UI. Same functionality was already available through Labels, but on Labels the entire label acts as click-able area (showing a HAND cursor), whereas when done with an HTMLArea only the links are click-able. The "displayTags" property on the HTMLArea needs to be set to true in order to process the titleText property.
- Added encapsulation support on Forms: the ability to mark Forms as Private, Module Private and/or to prevent access to the Form's elements, controller, foundset and/or dataproviders
- Improved support for separators within Combobox dropdowns:
 - Better styling
 - Non-selectable
 - Web Client support
- Added support for single RadioButtons:
 - Fields of type Radio with a ValueList attached containing only 1 value now render a single RadioButton
 - Multiple single Radio button type of fields can act as a RadioButton group, by sharing the same dataprovider
- Added option to control the automatic dropdown behavior of Comboboxes on focus gained through the APP_UI_PROPERTY. COMBOBOX_SHOW_POPUP_ON_FOCUS_GAIN property. Note that this property only affects the Smart Client, as this behavior is not controllable in browsers)
- Added ability to control the margin of text in TableView headers through the LabelFor relation
- Added support to control the delay intervals for showing and hiding ToolTips through the UIProperties APP_UI_PROPERTY. TOOLTIP_INITIAL_DELAY & APP_UI_PROPERTY.TOOLTIP_DISMISS_DELAY
- Grouped Elements now extend the RuntimeComponent, giving the Group object an extended designTime and Runtime API, providing options to set the location, size, visibility etc.

- Added 'empty' option to the namedFoundSet property on Forms: the 'empty' option determines if the foundset automatically loaded with records when it gets initialized. As such, this 'empty' value is a form level equivalent of the databaseManager.setCreateEmptyFormFoundsets() function, which acts globally. Note that the 'empty' setting is only looked at if the form foundset needs to be initialized: if the foundset is shared between with other forms and the through the other forms the foundset is already initialized, the 'empty' has no effect, as the foundset is already initialized

Drag 'n' Drop

- Added a custom Drag 'n' Drop event type JSDNDEvent with additional API:
 - dataMimetype property
 - getDragResult(): gives the result of the DND operation (DRAGNDROP.MOVE/DRAGNDROP.COPY/DRAGNDROP.NONE). Only relevant if the event type is JSDNDEvent.ONDRAGEND
 - getRecord(): gives the record on which the event occurred
- Added onDragEnd event to forms and portals: This event gets fired on the form on which the DND operation started (the form on which the onDrag event fired), regardless if a drop occurred or not
- Fix: onDragOver doesn't fire anymore for elements that are layered below other elements
- Fix: all DND related events don't fire anymore on disabled elements/forms

Web Client specific

- DND operations can be canceled by the Escape key (Behavior already existing in the Smart Client)
- Fix: when onDrag returns JSDNDEvent.NONE the cursor is reverted to normal immediately
- Fix: DND related event handlers only work for the Form on which they are assigned, not for additional forms contained in containers (Tabpanel, Tabless Panel & Splitpane) on the form that has the DND related event handlers

Smart Client specific

- Support for DND operations to originate or end outside Servoy

Beans

- Public Java API has been extended to make it possible to hook into the DND model of Servoy. These new features have been incorporated into the DBTreeView & DBTreeTableView bean.

Solution Model

- Added API to interact with existing Calculations or create new Calculations: Calculations added through the SolutionModel at runtime automatically become stored calculations if created with the same name as one of the dataproviders in the datasource on which the calculation is created
- Added API to interact with existing Beans and add new Beans to Forms: In order to add new beans to a Form using the Solution Model, the full class name of the bean is required. The bean vendor needs to supply this.
- Added factory methods to create the border, font and pageformat strings required in the solutionModel
- Added method on all objects to retrieve the object's UUID

See more details below under Runtime API > SolutionModel API

Client Design mode

- Added the ability to specify on element level which resize handles are enabled: See CLIENTDESIGN.HANDLES constant under Application in the Solution Explorer
- Added the ability to specify on element level if the element can be designed: See CLIENTDESIGN.SELECTABLE constant under Application in the Solution Explorer

Designtime API

- New onRender event on Forms and all elements
- New onDragEnd event on Forms and Portals
- New visible property on all elements
- New enabled property on all elements
- New encapsulation property on Forms
- Renamed "resizable" property of Portals to "resizable"
- Renamed "onTabChange" event of Tabpanels to "onChange"
- Made onChange event on SplitPanels fire when the dividerLocation is changed (through user action, window resize or scripting)
- Renamed "border" property to "borderType" on DBTreeView bean, to be inline with naming convention
- Replaced "opaque" property with "transparent" property on DBTreeView bean, to be inline with naming convention

Runtime API

Windowing API

- New controller.getWindow(...)
- New application.isFormInWindow(...)
- New application.getWindow(...)
- New application.createWindow(...)
- New application.closeAllWindows(...)
- New JSWindow class
- Deprecated controller.getContainerName(...)
- Deprecated application.closeForm(...)
- Deprecated application.getWindowHeight(...)

- Deprecated application.getWindowWidth(...)
- Deprecated application.getWindowX(...)
- Deprecated application.getWindowY(...)
- Deprecated application.isFormInWindow(): instead use the more powerfull controller.getWindow()
- Deprecated application.setWindowLocation(...)
- Deprecated application.setWindowSize(...)
- Deprecated application.showFormInDialog(...)
- Deprecated application.showFormInWindow(...)
- Deprecated constant UICONSTANTS.FULL_SCREEN

SolutionModel API

- New enabled & visible property on all JSComponent instances
- New getUUID() method on all relevant Solution Model objects
- New onRender event access on all objects that implement the onRender event
- New solutionModel.wrapMethodWithArguments(...)
- New solutionModel.newCalculation(...)
- New solutionModel.getCalculations(...)
- New solutionModel.getCalculation(...)
- New solutionModel.createTitledBorder(...)
- New solutionModel.createSpecialMatteBorder(...)
- New solutionModel.createPageFormat(...)
- New solutionModel.createMatteBorder(...)
- New solutionModel.createLineBorder(...)
- New solutionModel.createFont(...)
- New solutionModel.createEtchedBorder(...)
- New solutionModel.createEmptyBorder(...)
- New solutionModel.createBevelBorder(...)
- New solutionModel.newForm(String, JSForm)
- New solutionModel.removeRelation(...)
- New class JSCalculation(...)
- New class JSBean(...)
- New JSTabPanel.removeTab(...)
- Deprecated JSTabPanel.onTabChange
- New JSTabPanel.onChange
- Deprecated JSPortal.resizeble
- New JSPortal.resizable
- New JSPortal.onDragEnd
- New JSRelation.removeRelationItem(...)
- New JSMethod.getArguments(...)
- New JSForm.getBeans(...)
- New JSForm.getBean(...)
- New JSForm.removeBean(...)
- New JSForm.newBean(...)
- New JSForm.onDragEnd
- New JSForm.encapsulation
- JSForm.getBeans/getButtons/getComponents/getFields/getFormMethods/getFormVariables/getLabels/getParts/getPortals/getTabPanels: new optional parameter "returnInheritedElements"
- New constant classes
 - UNITS
 - TITLEPOSITION
 - TITLEJUSTIFICATION
 - PAGEORIENTATION
 - FONTSTYLE
 - BEVELTYPE

ServoyException

- New DataException.getStackTrace()
- New ServoyException.getStackTrace()

databaseManager

- getFailedRecords: optional JSFoundset parameter
- getEditedRecords: optional JSFoundset parameter
- New addTrackingInfo(...)
- New databaseManager.dataSourceExists()
- New createDataSourceByQuery(String name, String server_name, String sql_query, Object[] arguments, int max_returned_rows): Creates a datasource directly from an SQL statement, instead of having to create a JSDataSet based on SQL and then create the DataSource based on the JSDataSet
- Deprecated getFoundSetDataProviderAsArray() in favor of databaseManager.convertToDataSet(foundset, [beta:'columnName']). getColumnAsArray()

Miscellaneous

- Deprecated rowBGColorCalculation property on Forms, in favor of CSS support for odd/even/selected styling and onRender event for finegrained styling control
- Deprecated application.getStartupArguments(): the deeplink arguments are now also applied to the onOpen method of the login solution (were already applied to the onOpen method of the main solution), which makes the getStartupArguments() method obsolete
- Deprecated i18n.setDefaultTimezone() in favor of i18n.setTimezone()

- New class JSRenderEvent: eventType passed to onRender event handler methods
- New class Renderable: used by the JSRenderEvent
- New class JSWindow; see Windowing API
- New class JSDNDEvent: eventType passed to Drag 'n' Drop event handlers
- New class CLIENTDESIGN containing constants for ClientDesign
- New constants
 - UICONSTANTS.USE_SYSTEM_PRINT_DIALOG
 - ELEMENT_TYPES.TYPE_AHEAD
 - ELEMENT_TYPES.SPLITPANE
 - ELEMENT_TYPES.RTF_AREA
 - ELEMENT_TYPES.RECTANGLE
 - ELEMENT_TYPES.HTML_AREA
 - ELEMENT_TYPES.COMBOBOX
 - DRAGNDROP.MIME_TYPE_SERVOY_RECORD
 - DRAGNDROP.MIME_TYPE_SERVOY
 - JSEvent.ONDRAGEND
 - SQL_ACTION_TYPES.UPDATE_ACTION
 - SQL_ACTION_TYPES.SELECT_ACTION
 - SQL_ACTION_TYPES.NO_ACTION
 - SQL_ACTION_TYPES.INSERT_ACTION
 - SQL_ACTION_TYPES.DELETE_ACTION
- New security.isUserMemberOfGroup(...)
- New JSTable.getDataSource(...)
- New JSRecord.getDataSource(...)
- New JSDataset.sort(Function)
- New JSFoundset.sort(Function, [beta:defer])
- New setRollOverImageURL(...) on Labels and Buttons
- New getImageURL() on Labels
- Deprecated .getTitleText() on fields in favor of a titleText property that also allows setting the titleText at runtime
- Made Date.setYear deprecated (is deprecated in ECMAScript)
- Made relation parameter optional for setLeft/RightForm(...) functions for Splitpane instances
- Added support for multiple fallback filter values in i18n.setI18NMessagefilter, by supplying an Array of Strings for the value parameter
- New i18n.getCountries()
- New i18n.setDefaultTimeZone()
- Add support to utils.stringReplaceTags() to accept as input not only a JSFoundset or JSRecord but a RuntimeForm as well
- New application.removeClientInfo()
- Added UIConstant that determines whether readOnly fields remain readOnly in findMode or not: APP_UI_PROPERTY.LEAVE_FIELDS_READONLY_IN_FIND_MODE
- Support for IN operator support in scripted Find/Search. The advantage over the OR ('||') option are:
 - Unlimited size: the Or statement is limited to what the database supports
 - The values don't need to be concatenated into a String value, making it easier to work with values containing '||' or working with Dates
- ```
if (foundset.find()) {
 foundset.name = ['John', 'Paul', 'Ringo', 'George'];
 foundset.search();
}
```

## Web Client specific

- Support added to Icons on Tabs of a Tabpanel
- Support added for percentage values when setting the divider position of Splitpanes (values between 0 and 1 are considered percentage values e. g. 0.5 means 50%)
- Support added for .replaceSelectedText(...) on fields
- Support added for rollOverImageMedia
- Support for "media:///..." in HTML inside tooltips
- Support for "media:///..." for the "scr" attribute of Script and Style tags included in non-editable HTMLArea's and HTML contained in the text property of Labels
- Added mnemonic support in Web Client
- Added horizontal & vertical alignment support to labels/buttons with images in the Web Client
- Link tags contained in non-editable HTMLArea's are now inserted into the HEAD part of the Markup
- Support for HTML on nodes of the DBTree(Table)View beans
- Support for separators within ComboBox dropdown
- Partial Dialogs plugin support (see [Plugins & Beans](#))
- Added support for editable HTMLArea: editable HTMLArea's in the Web Client will now display a full WYSIWYG HTML editor. due to the nature of the Web Client, the HTML Editor displays all it's buttons inside the HTMLArea, whereas in the Smart Client, the buttons are displayed in a ToolBar in the main Toolbar area
- Added anchoring support for non-Servoy-aware beans
- Performance improvement of the Web Client due to compression enabling by default
- Moved all styling of Tabpanels to the servoy\_default\_webclient\_stylesheets.css
- Moved default form background-color into servoy\_default\_webclient\_stylesheets.css
- Exposed an 'indicator' styleClass in servoy\_default\_webclient\_stylesheets.css, to allow customization of the Loading indicator
- Ability to control the text of the "Loading..." indicator by specifying an i18n key with the following name: servoy.general.loading
- Disabled tooltips on touch devices
- Upgraded several used libraries:
  - Upgraded to jQuery 1.5
  - Upgraded to YUI 2.8.2r1
- Support for retrieving browser-side variables in JavaScript callbacks inside non-editable HTMLArea's by using a prefix

```
<html>
<body>
<button onclick=" javascript:globals.myCallBackMethod(browser:browserSideVariableName1, browser:
browserSideVariableName1) ">test</button>
</body>
</html>
```

## Plugins & Beans

### New OpenID plugin:

To be described

### Dialogs plugin

- The method `showWarningDialog(...)` can now be used in combination with the Web Client and present the user with the native browser "alert" dialog. Note that the dialog will only be show to the user after the method is finished.
- In Smart Clients running under Java 6 all dialogs popped up through the plugin only block their parent window hierargy: In Java 5 a dialog popped up through this plugin locks all existing windows

### DBTree(Table)View beans

- Drag 'n' Drop support
- CSS Styling Support
- Support for HTML as text for nodes of the tree in Web Client
- Added Double & RightClick support
- Added support to set the columnWidths

### File plugin

- Deprecated `getRemoteList()`
- Deprecated `plugins.file.getHomedirectory()` in favor of `plugins.file.getHomeFolder()`
- New `file.getRemoteFolderContents(...)`
- New `file.getDefaultUploadLocation(...)`
- New `JSFile.setBytes(...)`
- New class `JSProgressMonitor`

### Maintenance plugin

- New `JSCClientInformation.getLoginTime(...)`
- New `JSCClientInformation.getIdleTime(...)`
- New `JSTableObject.getDataSource(...)`

### RESTful Web Services plugin

- Added support to `ws_read` & `ws_create` method to optionally return an XML object instead of a JavaScript Object. (see JS Lib > XML)
- Added support for retrieving the Query Parameters of the Request: they will be passed in an Array as the last argument into the `ws_` call.

```
/**
 * @param {String...} pathArguments
 * @param {Object<Array<String>>} requestParams
 */
function ws_read(pathArgument, requestParams) {
 var _requestParams;
 if (arguments.length > 0 && arguments[arguments.length -1] instanceof Array) {
 _requestParams = arguments[arguments.length - 1];
 }
}
```

- Added support for returning specific HTTP Status codes from the `ws_*` methods, by throwing an exception with a number as value:

```
throw 404; //404 is the HTTP status code for Not Found, for example. (see http://www.w3.org/Protocols/rfc2616
/rfc2616-sec10.html)
```

- Ability to implement custom authentication, by implementing a `ws_authenticate` method, similar to the other `ws_` methods:

```
/**
 * @param {String} userName
 * @param {String} password
 * @returns {Boolean}
```

```

*/
function ws_authenticate(userName, password) {
 //Implement custom authentication logic here
 return true;
}

```

- Ability for clients to check through an HTTP Options Request which operations (PUT, GET, DELETE and UPDATE) are supported i.e. which ws\_\* methods are implemented

### SpellCheck plugin

- Added Google SpellChecker support  
In the Smart Client under Edit > Preferences > SpellCheck the a setting is exposed for the User to select the Spelling Engine to use. This can be either Rapid (default) or Google.

### HTTP plugin

Http plugin enhancements:

- Normalized programming API:
  - Instead of creating a Client and referencing the client based on a name string, now a Client object is returned, with it's own API
  - Each RequestType has it's own Class with the relevant methods
  - The execution of a Request returns a Response class which again has it's own API
- Support for all requestTypes: Post, Put, Delete, Get, Options, Trace
- API on Requests to set Headers, Parameters and BodyContent
- API on Response to get Headers, Parameters and BodyContent
- Included constants for all Http Status codes
- Kerberos & NTLM authentication support
- New .setTimeout(...) method on the HttpClient class

### PDF plugin

- Added support for working with XFA forms

### PDF Output plugin

- The PDF output plugin is now registered as Printer, to allow printing to PDF from the Print Preview in the Smart Client

### Window plugin

- Deprecated .validate() on Toolbar and MenuBar class, as they are no longer required. Changes to the Toolbar or MenuBar are visible immediately
- Deprecated plugins.window.setToolBarVisible() in favor of plugins.window.setToolBarAreaVisible(), to align the name with what the method actually does
- Added support to create MenuBars on (Modal) Dialogs as well (Windows was already possible)
- Added .putClientProperty() to the UI classes of the windows plugin
- New setBackgroundColor/setForegroundColor methods on RadioButton/Checkbox/MenuItem classes of window plugin
- Added maximize() method, to maximize a JSWindow of type Window in the Smart Client

### Misc.

- Added plugins.textport.textExport(...) to script the export
- Added plugins.excelport.excelExport(...) to script the export
- New plugins.mail.isValidEmailAddress(...)
- The preferred size of Beans is now honored when placing beans onto a form in the Form Editor
- Added anchoring support for non-servoy-aware beans in the Web Client
- Added setting on the Admin page to store the StyleClass name set on an element as a UIProperty on the element. The setting on the Admin page, "servoy.smartclient.componentStyleClassDelegatePropertyKey" allows to specify the name for the UIProperty. This information can then be read and used by, for example, Look and Feels.
- For Beans that supply an Icon through a BeanInfo class, the Icon is now used in the display of the bean in the Form Editor Palette
- Support for multi-value property editors on beans (see Servoy 6 Public Java API for more details: [<http://www.servoy.com/docs/public-api/6xx/com/servoy/j2db/dataui/PropertyEditorHint.html>] and [<http://www.servoy.com/docs/public-api/6xx/com/servoy/j2db/dataui/PropertyEditorOption.html>])

### Miscellaneous

- Dedicated SolutionType values for pre and post import hooks, instead of relying on a naming convention
- Ability to use a global method as PK generator, through selecting a global method under Auto Enter > Lookup Value on the Column in the Table editor
- Support for using variables of type Boolean inside relations, mapping on integer columns
- Added ability to use separators in ValueLists
- Made "Left Outer Join" the default value for new relations
- Ability to use global variables of type Boolean in Relations. Can map to integer columns containing 0 or 1 in the database
- Support i18n filter fallbacks for i18n.setI18NMessagesFilter (see samplecode)
- Support to mark functions through the JSDoc @AllowToRunInFind tag to be executed as event handlers while in FindMode:  
Previously methods attached to event handlers would only execute in FindMode if the contained the code the execute a search (controller.search() or foundset.search()). As of Servoy 6 any method that contains the @AllowToRunInFind tag in it's JSDoc will be executed in FindMode when

attached to an event. Note that there are limitations to what is possible in scripting while in FindMode and that certain behavior is not well defined. More information on this will follow.

- Support to mark variables and functions as private through JSDoc using the `@deprecated` JSDoc tag
- Support for dependent ValueLists when in FindMode
- Added extra parameter to globals method ValueList that indicated if the FoundSet is in FindMode
- Optimized databaseManager.recalculate() to only trigger the recalculation of stored calculations
- Added weeknumber to Date Field calendar popup in Smart Client, to bring it inline with the Web Client

## Deployment

### General

- Compression is enabled by default on the Servoy Application Server for serving HTML, CSS and JavaScript files, which improves the performance of the Web Client
- Lazy loading of the datamodel of database server clones, resulting in faster Servoy Application Server startup and no unneeded overhead for database server clones that aren't going to be used
- Enhanced Security is now the default mode, can only be disabled by manually setting the relevant property in the `servoy.properties` file: `"servoy.application_server.enhancedSecurity=false"`
- The default value for the MaxHeapSize memory setting (`-Xmx`) for the Application Server has been changed to 1Gb (both in `servoy_server.bat` as well as in the `wrapper.conf` for the Service component)
- Display Community Edition splash screen content over HTTPS, to prevent mixed content warnings when running the community edition over HTTPS
- In the Preferences of the Smart Client, the Services Preference tab is hidden if the Smart Client is not connected to the Application Server through Direct connection of TwoWay socket
- BugFix/BehaviorChange: pre-import hook now runs before updating the datamodel and importing sample data
- Improved Web Client source generation: The generated source will always use short, dynamic ID's, to lower the amount of data that needs to be sent back and forth between the browser and the server.  
The option on the Servoy Admin page to use Local ID's (`servoy.webclient.templates.use_local_ids`) has been removed, as it's no longer needed. The HTML/CSS templates contain the long UUID and Servoy will automatically map the UUID's in customized templates at runtime to the internal ID's that end up in the Web Client's page markup

### Admin page

- Ability to update the "Maximum connections active" and "Maximum connections idle" without requiring a restart
- Added ability to specify the user directory for Smart Clients on the Admin page
- Ability on the Admin page to set on Solution level if the the solution is to use Anchors in the Web Client or not
- Ability to "compact" a Solution: this removes all but the latest release of the solution
- Added Styles node, to allow the deletion of StyleSheets from the Repository
- Added a grid-like Database Server overview on the Servoy Admin page for better overview of the status of all Database Server connections, with links to get to the details of a specific Database Server. Database Server clones will be grouped under their master.
- Moved the Log Server setting away from the individual Database Server details to the Database Server overview, as there can only be one Log server defined on the Application Server
- Added Solution import option to auto update Servoy Sequences after import
- Added Database Server name to the Performance Data page of the Admin page
- Added log entry when clearing the log on the Server Log page of the Admin page
- Moved the link to import Solutions to the top of the Solutions page on the Admin page
- Improved type tagging of queries in the Performance Data page on the Admin page. For example, FoundSet queries are now properly tagged as FoundSet queries.
- Added option to push a Font to clients as well (To complement the already existing Look & Feel and Theme push options)
- Removed `SocketFactory.useTwoWaySocket` and `SocketFactory.rmiServerFactory` on the Network section of the Admin page and replaced them with a `SocketFactory.tunnelConnectionMode` property that is displayed as a dropdown with the available options.
- Added option to disable SSL encryption when using the HTTP Tunnel if the HTTP Tunnel runs over HTTPS (to prevent double encryption)
- Added option to set the min and max Java Version for Smart Clients on the Admin page: Setting the minimum Java version to Java 1.6 will force the Smart Client to only start when Java 6 is available on the Client machine.
  - `servoy.smartclient.min_java_version`
  - `servoy.smartclient.max_java_version`
- Relocated Smart Client related memory settings under the "Smart Client Settings", instead of the "Memory/Performance settings"
- Removed `"servoy.webclient.templates.use_local_ids"` preference, see "Improved Web Client source generation" item under General

### All-In-One installer

- Split the workflow in the installer into two separate flows: Servoy Developer installation and Servoy Application Server installation
- Added the option to download & install the EnterpriseDB Standard Server instead of installing the bundled PostgreSQL database engine
- Added the option to connect to an existing database instead of installing the bundled PostgreSQL database engine
- Due to the two items above, the installation will always contain a `servoy.properties` file after installation, contrary to Servoy 5.2 where it was missing after deselecting the Database option in the installer
- Added the ability to specify the database server and data directories and the database server port when installing the bundled PostgreSQL database
- Added the PostgreSQL upgrade module to the bundled PostgreSQL
- Made installation of Clustering configuration optional for Application Server installations and removed it from the Developer installation
- Split the installation of the Service component into two parts: installing the component and setting up the actual service
- Changed License Agreements text to styled text for better readability
- Added options to set the HTTP and RMI port in the installer
- Added warning when trying to install into Program files on Windows
- Changed the options panel into a tree structure to better resemble dependencies
- When installing the Application Server, the option `"servoy.application_server.startRepositoryAsTeamProvider"` is set to false by default

### Service component

- Added support for the Service Component on all platforms (Windows, OSX, \*nix)
- Config has been added to automatically restart the Application Server when OutOfMemory exception occur: the rationale behind the automated restart is that an OutOfMemory exception can severely compromise the stability of the Application Server and the clients it hosts.
- The default value for the MaxHeapSize has been set to 1Gb
- On Windows, the startType of the Service has been changed to "AUTO\_START"
- Added install\_service/uninstall\_service.bat/.sh files to install and uninstall the service on all platforms (on Windows the existing install\_windows\_service.bat files was removed)

## Public Java API

- Added API to hook into Servoy's Drag 'n' Drop mechanism
- Added API to interact with Servoy StyleSheets
- For Beans that supply an Icon through a BeanInfo class, the Icon is now used in the display of the bean in the Form Editor Palette
- Support for multi-value property editors on beans (see Servoy 6 Public Java API for more details <http://www.servoy.com/docs/public-api/6xx/com/servoy/j2db/dataui/PropertyEditorHint.html> and <http://www.servoy.com/docs/public-api/6xx/com/servoy/j2db/dataui/PropertyEditorOption.html>)
- Deprecated Window getCurrentWindow() in favor of IRuntimeWindow getCurrentRuntimeWindow()

## Miscellaneous

- Upgrades several libraries (.jar files)
  - metadataextractor.jar updated to version 2.4.0b1
  - quartz.jar updated to 1.8.3
  - Apache HTTP Client updated to 4.0
  - Eclipse updated to 3.6.2 (Helios release)
  - DLTk updated
  - GEF updated
  - jQuery updated to version 1.5
  - YUI updated to 2.8.2r1
  - Tomcat upgraded to 6.0.32
- Removed several libraries (.jar files) from the distribution as they are no longer needed
  - jdbc2\_0-stdext.jar: standard as of 1.4
  - jdbc2\_0-stdext.jar
  - jmx.jar
  - jndi.jarjmx.jar: standard as of 1.5
  - jndi.jar: standard as of 1.3
- Included several new libraries (.jar files)
  - openid4java.jar
  - guice.jar (version 2.0)