**Spiro2**

The page view contains either one or two ‘main panes’ – each of which has identical capabilities.

Each pane has one of three possible ‘roots’:

* home (the list of main menus - replaces the services root in Spiro1)
* object
* query

Unlike Spiro one:

* Where an action method signature returns a single object, the resulting view will directly reference that object, not (as in Spiro 1) an ‘action result’.
* Where an action method signature returns a collection/queryable, the result will always be rendered as a collection showing a collection even if that collection has only one element, or no elements at all

Each of those three roots offers several options that determine the display, including sub-panes.

**Home**

The home root renders all of the Main Menus.

The user may select on any Main Menu, to see the actions for that menu.

The home view allows any one action to be opened as a dialog.

(Unlike Spiro 1 the result of an action is not rendered within the same view as a sub-pane. It either replaces the existing main pane, or is opened as the other main pane).

State to be held in the URL:

* Which main menu (if any) is selected
* Which action (if any) main-menu action is open as a dialog

**Object**

Each of the collections on that object may be rendered either as a summary, list or table. (In contrast to Spiro1, where only one of an object’s collections might be opened, and that was in a separate pane). (Note: Spiro2 also needs to honour the ‘Eager’ rendering specified in the domain model).

A single action dialog may be opened, representing an action on that object.

State to be held in the URL:

* Object identity
* Which collections are opened, and whether as List or Table in each case
* Whether the actions menu is open or not
* Which action (if any) object action is open as a dialog
* Whether the object is in edit mode or not

**Query**

The query result always consists of a collection, even if there are zero or one elements.

The result may be paged.

The result may be rendered as a List or a Table.

The result will be rendered with check-boxes and a menu of collection-contributed actions if appropriate.

Any one of the collection-contributed actions may be rendered as an open dialog.

(Change to Spiro1/Mvc behaviour: if the dialog is opened, then the number of objects selected should update as elements are selected/deselected, OR, should not be visible at all).

State to be held in the URL:

* The identity of the action, and the parameter values, to produce this collection
* Whether the collection is rendered as a List or a Table
* The page number if applicable
* Whether the collection-contributed ‘actions’ menu is open
* Which action collection-contributed action is open as a dialog

**URLs – scheme1**

First part of the url shows the root type for each pane e.g.:

For a single main-pane:   
/home  
/object  
/query

For two main-panes e.g.:  
/home/object  
/object/object  
/query /home

All other state information is held in the parameters. The following generic parameters are recognised, and have a suffix ‘1’ or ‘2’ to indicate which pane they apply to. Where there may be more than one value of the same form within a single pane, these have a further suffix e.g. 1\_1. Thus, table2\_3 indicates the name of a third collection to be rendered as a table within pane2.

Applies to all three roots:

* menu: the menu opened. For the home root, this will be the name of the menu. For the object or query root this will (unless the rendering has been customised) be ‘actions’.
* dialog: the name of the action (from the open menu) opened as a dialog

Applies to *object* root only:

* object: the identity of the Object root
* table: the name of the first collection to be opened as a table
* list: the name of the first collection to be opened as a list
* edit: value is ‘true’ if object is in edit mode

Applies to *query* root only:

* action: identity of action to produce the query
* param: prefix for each parameter values on the action invoked e.g. Param1\_1
* table: if the results are rendered as a table the value will be ‘true’. If the value is ‘false’ or the

param missing, then results are rendered as a list.

* page: page number of result

**Examples:**

The first story is confined to a single pane only:

1. Starting point:   
   /home
2. User selects on customers menu:   
   /home?menu1=customers
3. User selects FindByPPSN, opening dialog:   
   /home?menu1=customers&dialog1=FindByPPSN
4. Invoking action, customer object is returned:   
   /object?object1=[oid]
5. User selects edit button:
6. /object?object1=[oid]&edit1=true
7. User saves or cancels edit, then expands the Addresses collection as a list:

/object?object1=[oid]&list1\_1=Addresses

1. User expands the Notes collection as a list:

/object?object1=[oid]&list1\_1=Addresses&list1\_2=Notes

User changes Addresses expansion to table view:  
/object?object1=[oid] list1\_1=Notes&table1\_1=Addresses

1. User selects the Actions menu to open actions:   
   /object?object1=[oid] list1\_1=Notes&table1\_1=Addresses&menu1=actions
2. User selects action ‘Find Orders’ to open dialog:   
   /object?object1=[oid] list1\_1=Notes&table1\_1=Addresses  
   &menu1=actions?&dialog1=FindOrders
3. User completes params & executes dialog to return a query of Orders:  
   /query?action1=[id for action on object]&param1\_1=1234.56&param1\_2=Shipped
4. User renders the returned list as a table:
5. /query?action1=[id for action on object]&param1\_1=1234.56&param1\_2=Shipped&table=true

Now a scenario using split panes:

1. User selects FindByPPSN, opening dialog:   
   /home?menu1=customers&dialog1=FindByPPSN
2. User invokes action with right-click so customer object is returned in RH pane:   
   /home/object?menu1=customers&dialog1=FindByPPSN&object2=[oid]
3. User swaps left and right panes:

/object/home?object1=[oid]&menu2=customers&dialog2=FindByPPSN

1. User invokes the finder action again with right-click so new object opens on RH pane:  
   /object/object?object1=[oid]& object2=[oid]&
2. User expands LH pane to be full pane:   
   /object?object1=[oid]

**Some subtleties**

1. The query representation is of the results of an action with specified parameters. This action MUST be a query-only action (i.e. must return an IQueryable, or is marked [QueryOnly]). This is reflected in the RO representation by the http method i.e. GET. Non query-only actions returning collections *will not be recognised as actions by Spiro2*. If you really need this functionality then you should return a ViewModel that contains references to the returned objects in its keys.
2. Actions returning a single object e.g. Most Recent Order. The returned representation will be of the object NOT the action, so refreshing will always return that same object. If, for example, the user wants to guarantee to always see the latest order, they need to go back to the action/dialog and invoke it again. This way, combined with the point above, there is no danger of repeating a non-idempotent action just by hitting refresh.
3. Transient objects. In the long run we want to get rid of proto-persistent objects from the RO spec. ‘Transient’ objects would appear just like persistent objects and/or addressable view models. One solution is that a transient object will have a GUID Id, and the server maintains a short-lived cache of these objects, so that a transient object may be refreshed like a persistent object and will have the correct field values as were set up by the method that created it. If the cached version has vanished then the system fails gracefully, requiring the user to go back to the method that created the transient object.