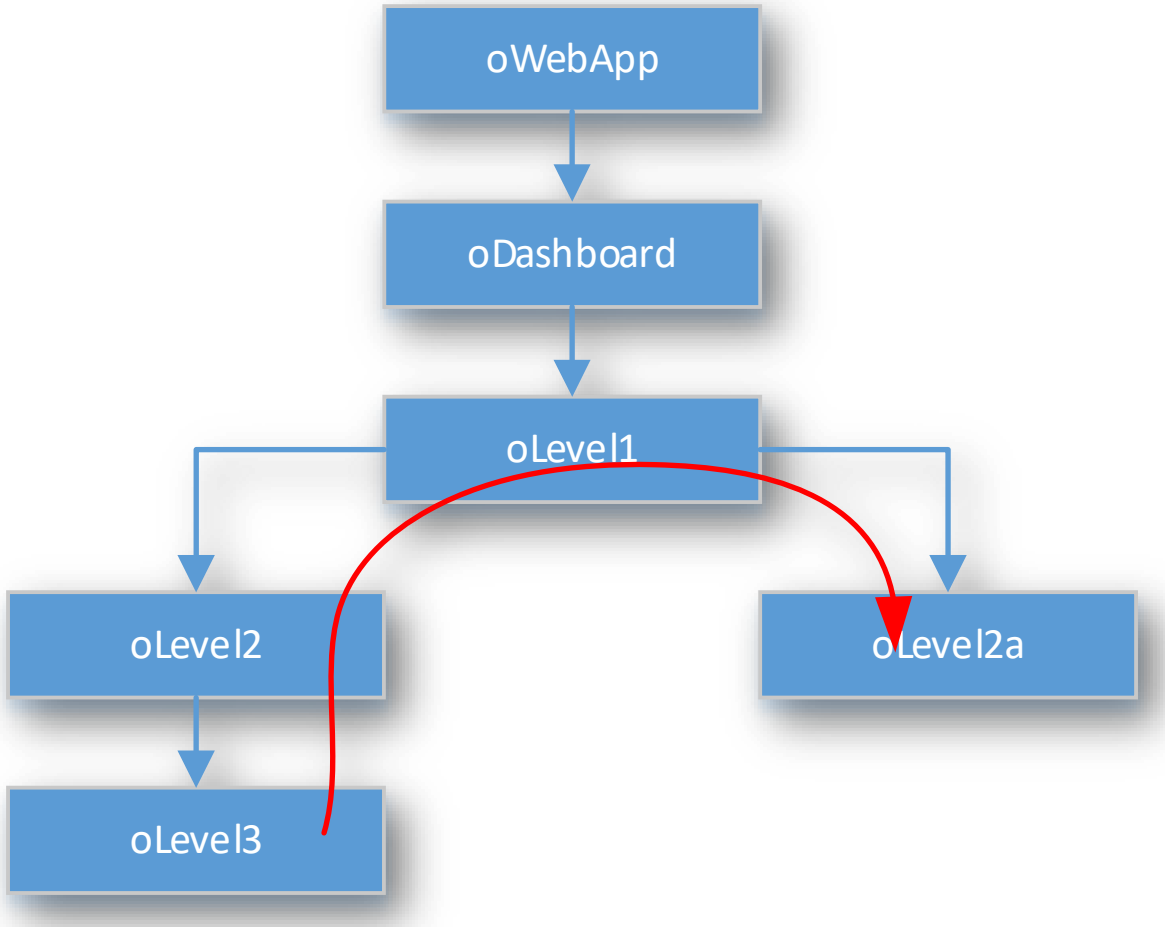
Advanced Navigation

Eddy Kleinjan, Data Access Europe

Drilldown Model

Navigation Drilldown Model

- Drilldown Model
- Follows a Path
- Get off the beaten Path?
- URL seeding
  - Load a page that his pre-loaded with data



WebOrderMobile

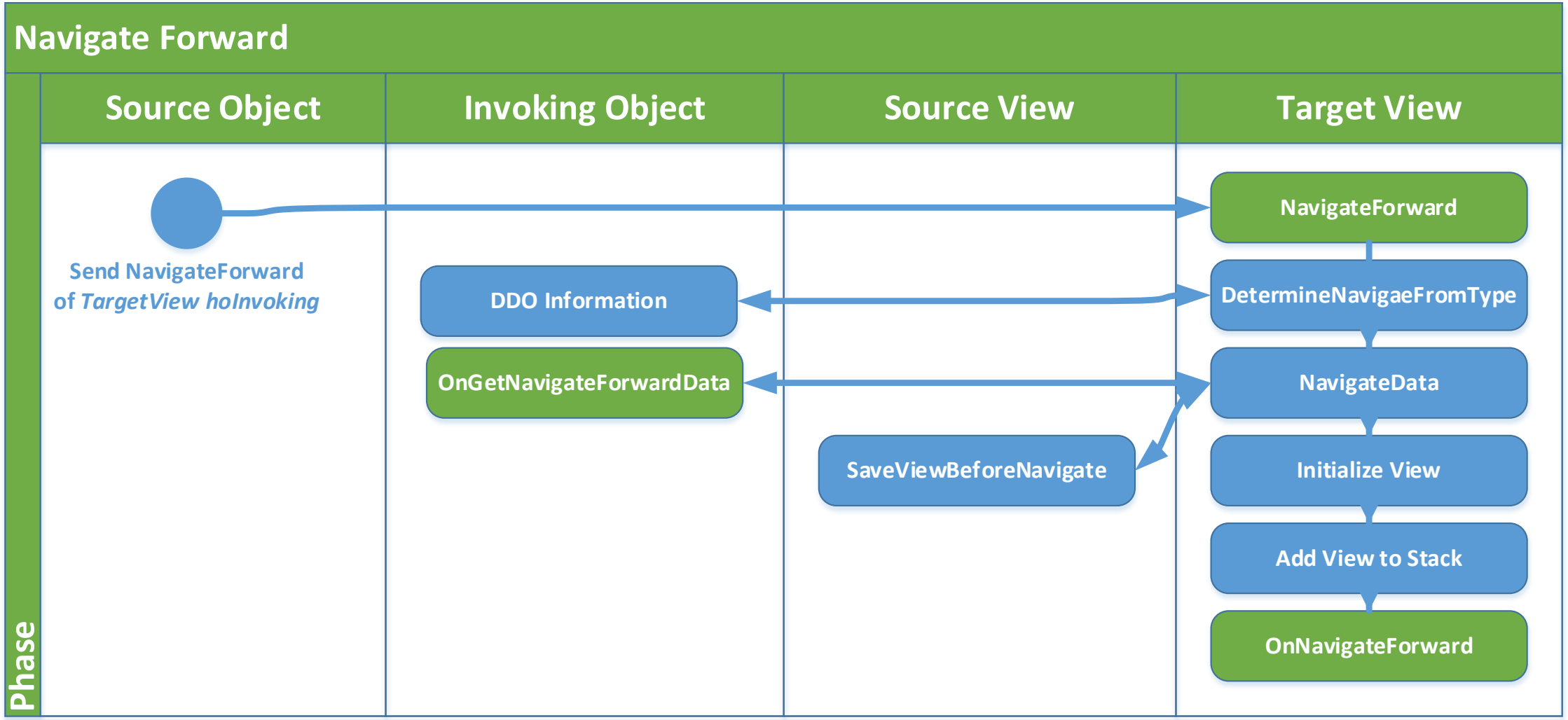
Navigation Methods

Navigation Methods

- NavigateForward / NavigateForwardCustom
- NavigateClose
- NavigateCancel
- NavigateCloseTo / NavigateCancelTo
- NavigateBegin

NavigateForward

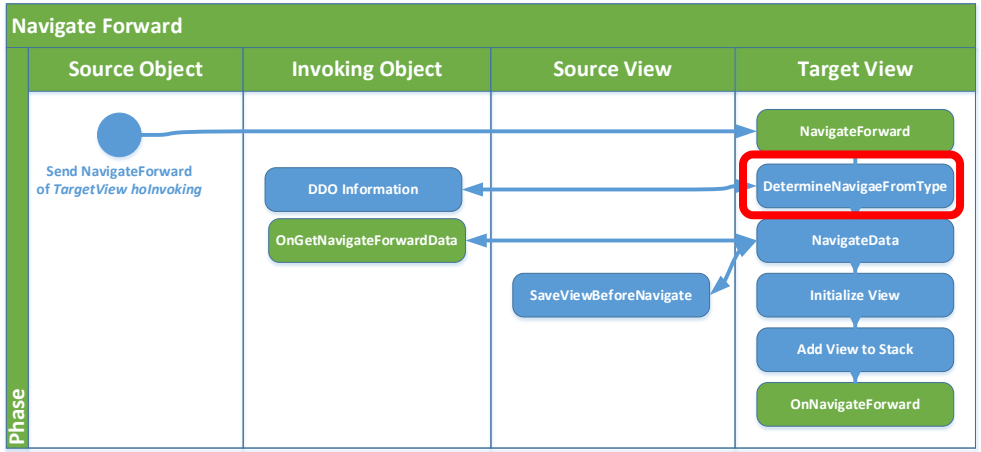
Navigate Forward



1 Determine Navigate-From type

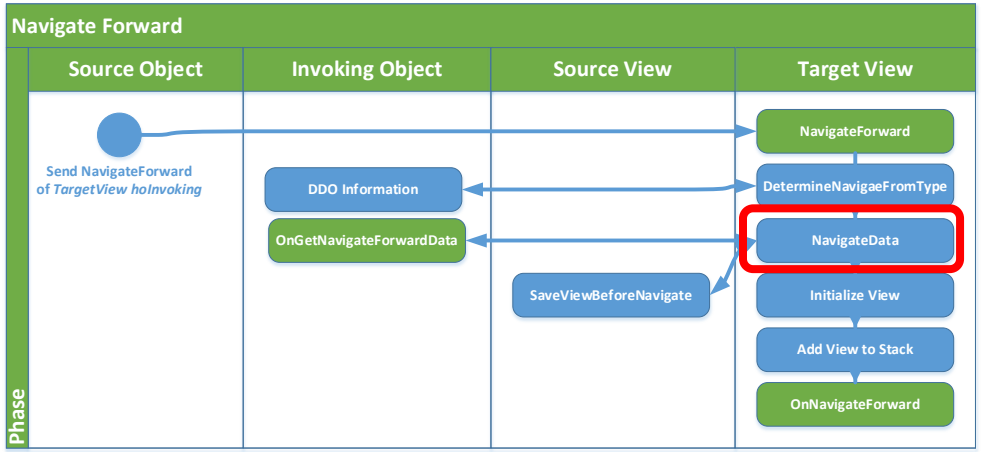
- Determines the Navigate-From type.

  The relationship between the invoking view/invoking object and the view being navigated to is used to determine this relationship.



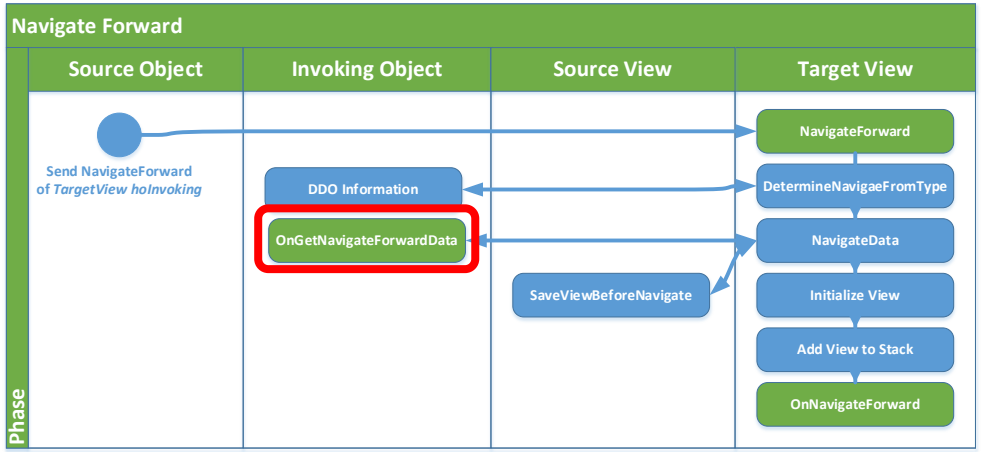
2 Create tWebNavigateData

- Create a tWebNavigateData variable and populate it with proper values. It does this by looking at and using eNavigateType, hoInvokingObject and the invoking view.



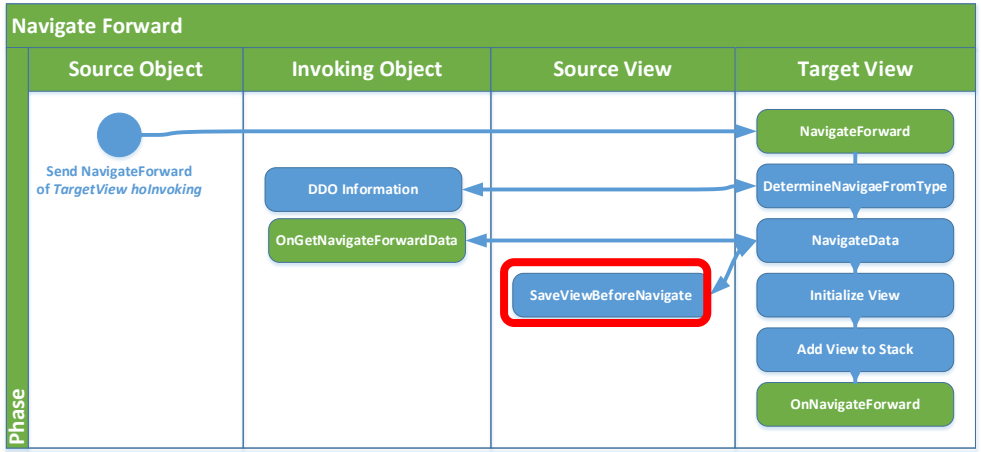
3 OnGetNavigateForwardData

- Sends OnGetNavigateForwardData to hoInvokingObject. This event passes the tWebNavigateData variable by reference, allowing the developer to customize this. Note that OnGetNavigateForwardData is sent to the invoking object, not the invoking view.



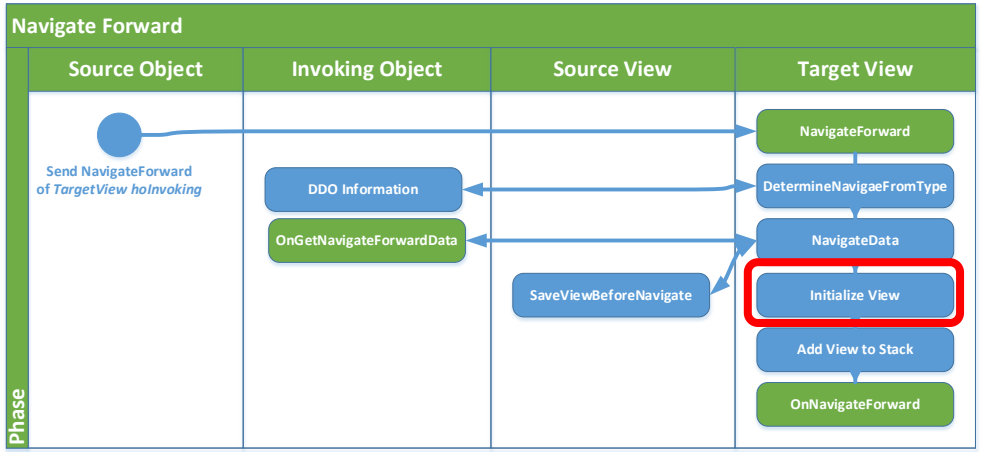
4 bSaveBeforeNavigate

- If the tWebNavigateData.bSaveBeforeNavigate member has been set True in OnGetNavigateForwardData, the invoking view will attempt to save any changed data. If the data cannot be saved or this is an empty record with nothing to save, the navigation will be halted.



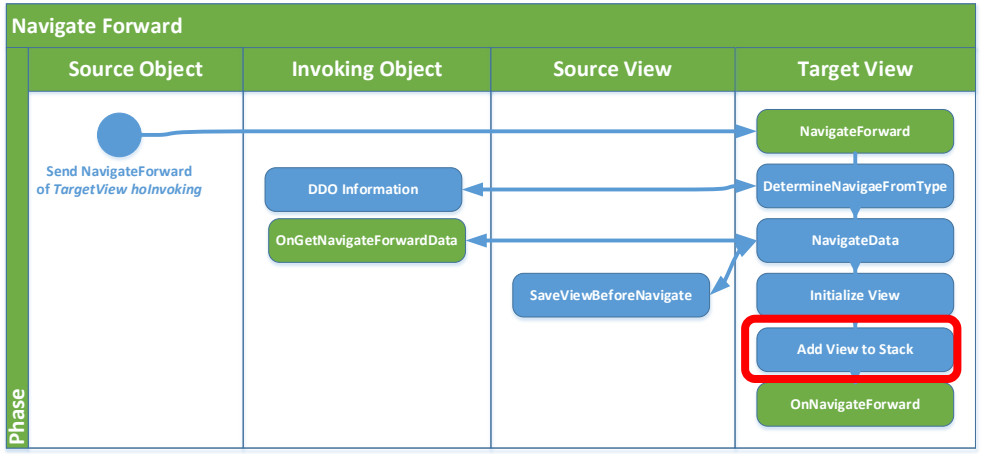
5 Init View

- Initialize the new view based on the contents of the tWebNavigateData variable. Depending on the Navigate-From type, the data in the invoking view and all of the data in the view stack, this will set relates-to constraints and find appropriate records for the new view.



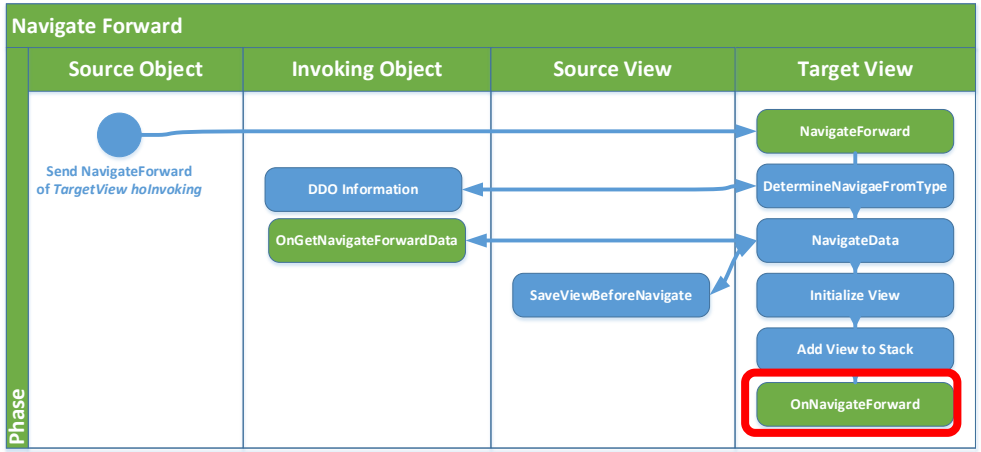
6 Add View to Stack

- Add the new view to the view stack, making it the new top view in the view stack.



7 OnNavigateForward

- Send OnNavigateForward to itself, passing the tWebNavigateData instance, a handle to the invoking view and a handle to the invoking object (hoInvokingObject). OnNavigateForward is an important event that the developer will use to further initialize and customize the view.



8 OnRefindRecordError

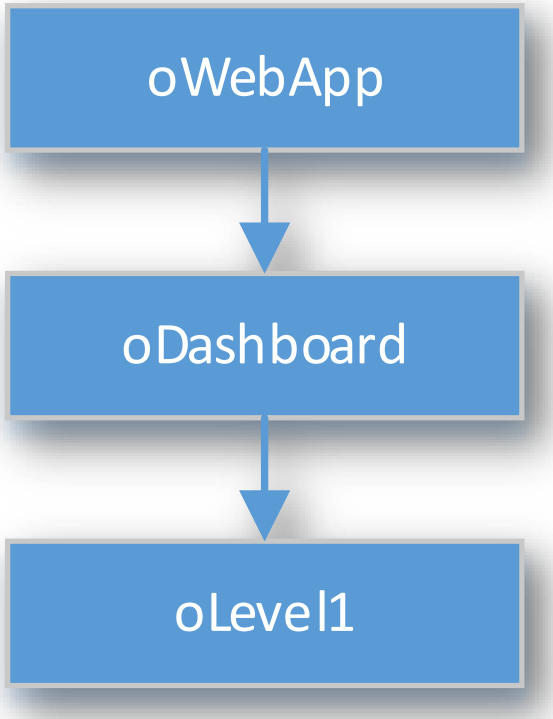
- A forward navigation can fail if DDO records cannot be refound. This can happen when data is changed in a multi-user environment or possibly by a programming error. When this happens the forward navigation is reversed. The event OnNavigateForwardRefindError is called which calls NavigateCancel to reverse the process. OnNavigateForward is not sent. See OnRefindRecordError for more on this.

NavigateForwardCustom

- Similar to NavigateForward
- Will not determine NavigateType; will always be nfUndefined
- Custom views; programmer responsible for moving values in and out using NavigateData.

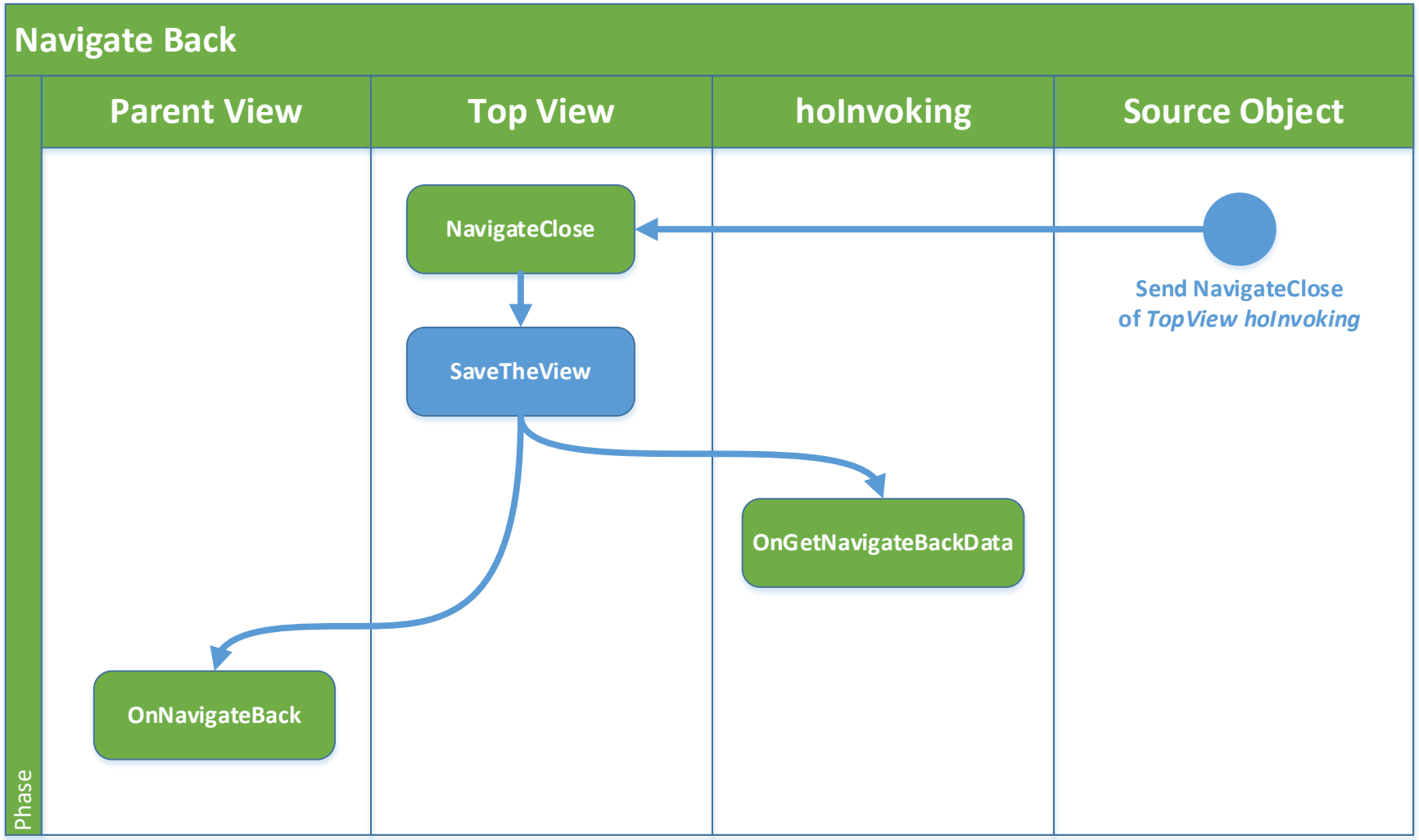
Level1

- Create a basic workspace
- Steps...
  - Create a New Workspace ‘NavTraining’
  - Create a ‘Mobil Web Project’
  - Set peLoginMode to lmLoginSupported
  - Create a new ‘Web Mobile Zoom’ view called ‘Level1’. Do not use the wizard for this.
  - Open the Dashboard.wo and add a button to activate the Level1 view. Put the button above the oTiles_Grp object.



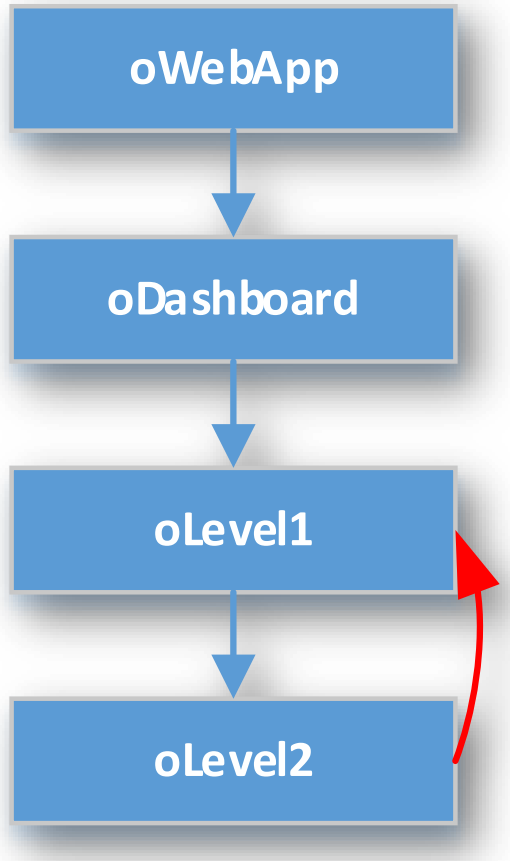
NavigateClose / NavigateCancel

NavigateClose



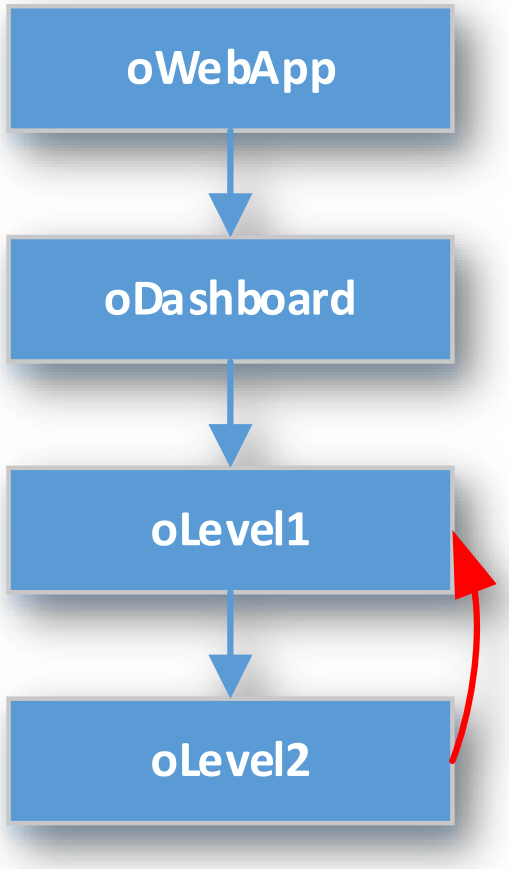
NavigateClose

- Procedure NavigateClose Handle hoCallback
- May involve Round-trip.
- OnGetNavigateBackData to Closing View callback Object
- OnNavigateBack to NavigateForward invoking Object



Level2

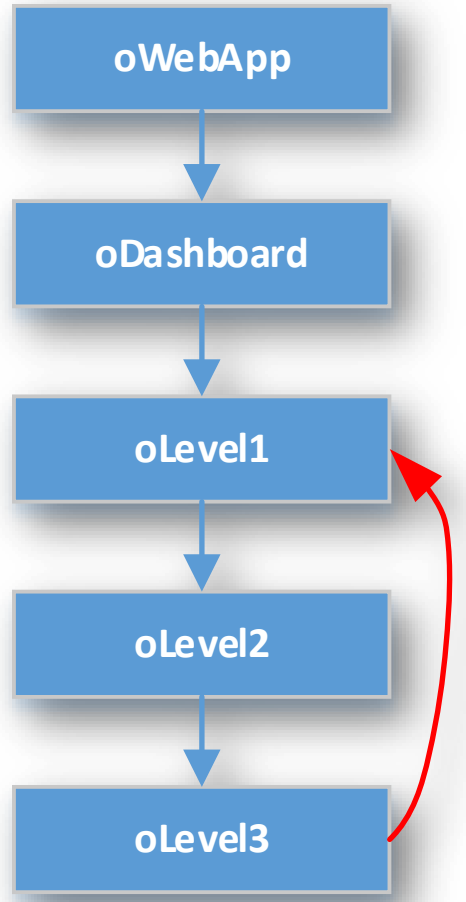
- Create a second level to drilldown
- Steps...
  - Copy Level1.wo to Level2.wo
  - Add a button to Level1 that will activate Level2
  - Add a button to Level2 that will close it.



NavigateCloseTo / NavigateCancelTo

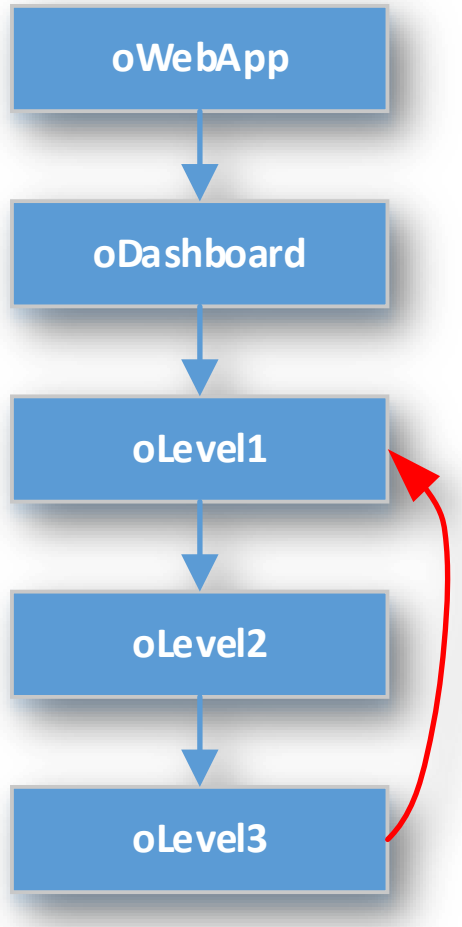
NavigateCloseTo

- Similar to NavigateClose
- NavigateClose to a view in the stack.
- Allows for passing back data as opposed to NavigateCancelTo
- NavigateType will be nfUndefined



Level3

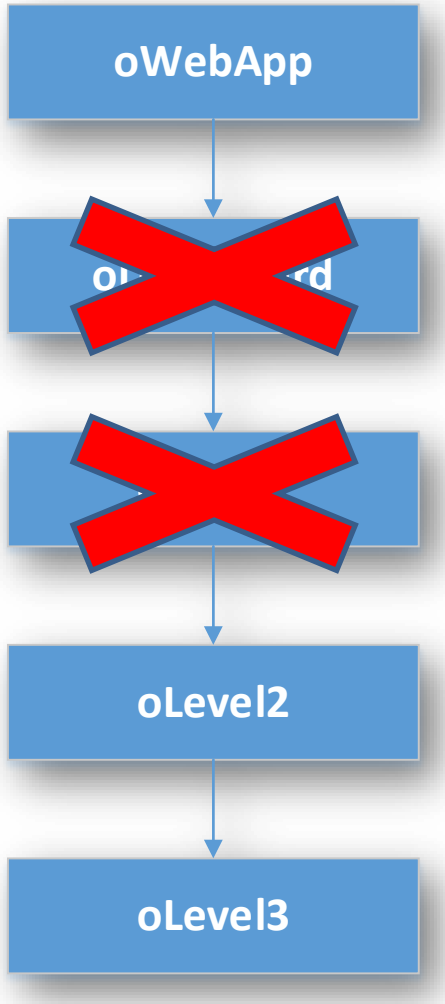
- Add a level 3 view to drilldown.
- Make it possible to directly close to level 1, bypassing level 2
- Steps...
  - Copy Level 2 to Level3
  - Change the button in Level2 to activate Level3
  - Add a button to Level3 to Close directly to Level 1



NavigateBegin

NavigateBegin

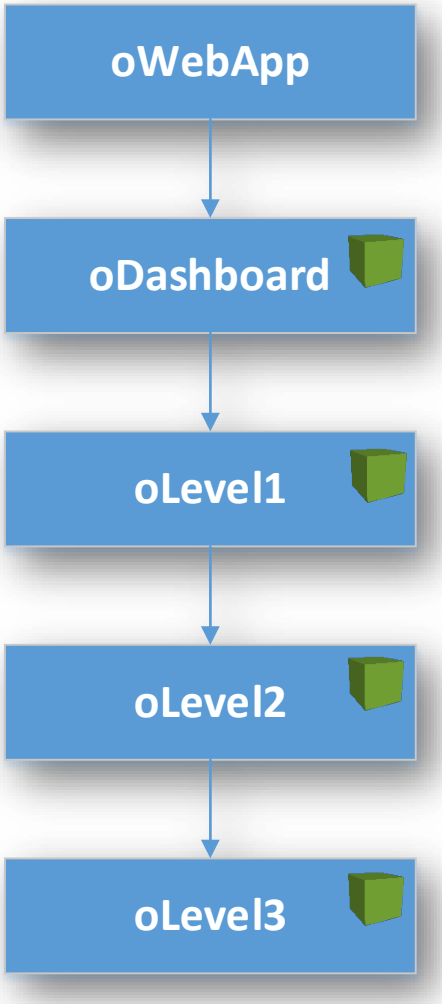
- Set New Stack Root
- NavigateBegin hoInvokingObject bUnconditional
- Send to View to become the first object in stack
- Clears stack and breadcrumb trail
- Adds itself by NavigateForward
- Dashboard has priority



NavigateData

NavigateData

- Structured type at view level
- Is a web property
- Each view has local instance of NavigateData
- Contains state of the view
- Stored on Client; don't store sensitive data



NavigateData 

```
Struct tWebNavigateData
    Integer eNavigateType // nfFromMain, nfFromChild, nfFromParent, nfUndefined
    Integer iTable        // Table number of the invoking object
    Integer iColumn       // Column number of the invoking object
    String sRowID         // Forward: current row id; Back: selected row id
    Boolean bNewRecord    // Start view in ‘new’ state
    Boolean bReadOnly     // Start view in readonly state
    Boolean bSaveBeforeNavigate  // A save is required before navigating from view
    tNameValuePair[] NamedValues // Your domain for custom data...
End_Struct
```

Working with NavigateData

- Get local copy using GetNavigateData
- Store using SetNavigateData
- Bypasses WebGet/Set limitations: Accessible from any child object of view
- Helper functions for managing NamedValues array:
  - NamedValueAdd
  - NamedValueGet
  - NamedValueRemove
  - NamedValueIndex

NamedValueAdd

```
Function NamedValueAdd NamedValues sName sValue returns NamedValues
```

Adds or updates a Name/Value pair to the NamedValues array

- NamesValues — Array holding NamedValues
- sName — Name to add; the name is case in-sensitive
- sValue — Value to add
- Returns — Updated NamedValues array

NamedValueGet

```
Function NamedValueGet NamedValues sName returns sValue
```

Gets stored value of NamedValues array by Name

- NamesValues — Array holding NamedValues
- sName — Name to of the Name/Value pair to get
- sValue — Stored value; empty if Name is not found

NamedValueIndex

```
Function NamedValueIndex NamedValues sName returns iIndex
```

Gets index of NamedValues array by Name

- NamesValues — Array holding NamedValues
- sName — Name to of the Name/Value pair to get
- iIndex — Index value of Name in NamedValues array
  -1 when not found

NamedValueRemove

```
Function NamedValueRemove NamedValues sName returns NamedValues
```

Removes stored value of NamedValues array by Name

- NamesValues — Array holding NamedValues
- sName — Name to of the Name/Value pair to remove
- Returns — Updated NamedValues array

Procedure OnGetNavigateForwardData

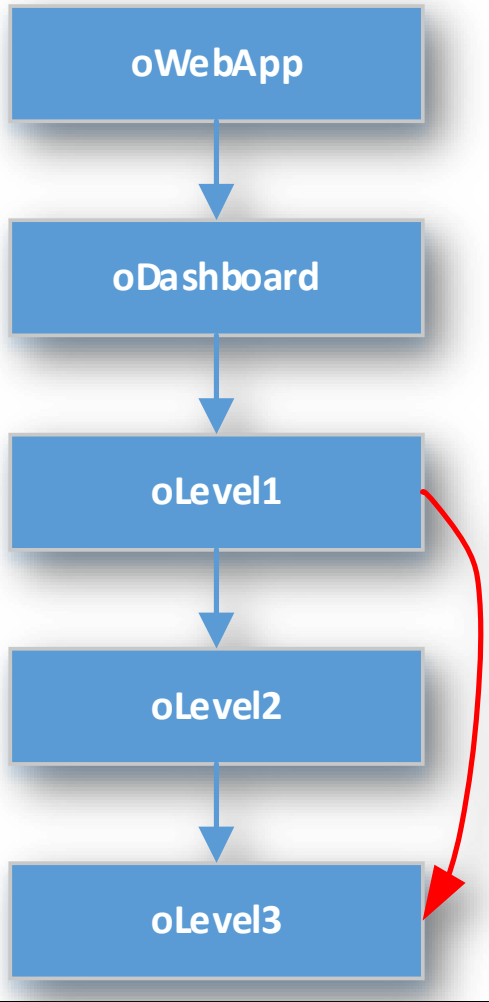
- Hook for providing Data to the view you NavigateForward to
- Data passed by reference

```
Procedure OnGetNavigateForwardData tWebNavigateData ByRef NavigateData ;
        Handle hoToView

Get NamedValueAdd NavigateData.NamedValues "Level2Data" sValue to NavigateData.NamedValues
Get NamedValueGet NavigateData.NamedValues "Level2Data" to sValue
```

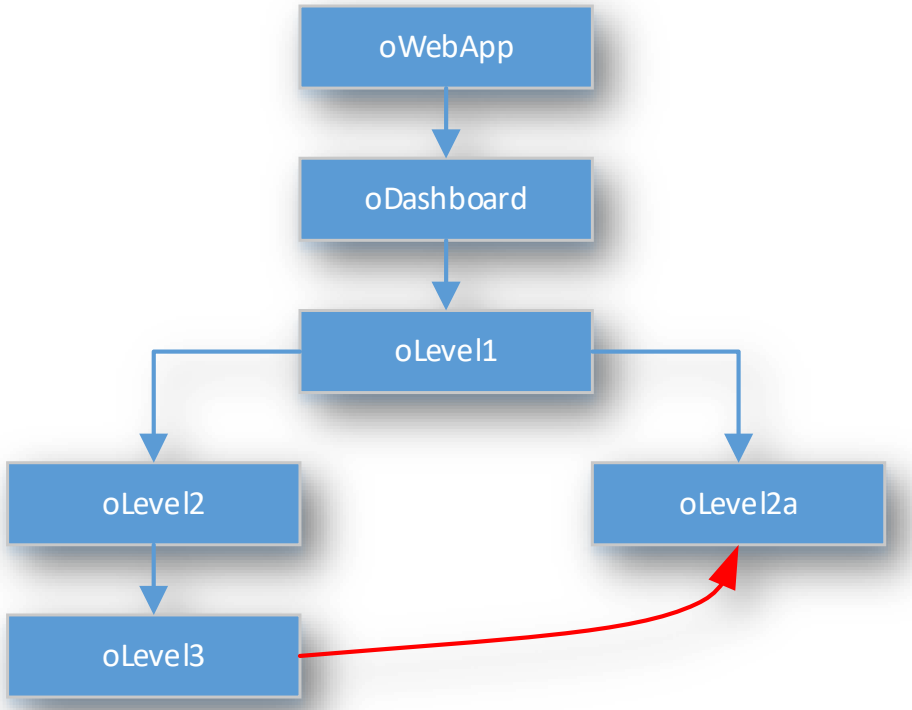
Navigate Forward 2 Levels

- Button on Level1 to go to Level 3 without bypassing Level2
- Use NamedValues of NavigateData to pass the objectname to forward to
- Use OnGetNavigateForwardData in Level1 to set the information
- Use OnNavigateForward in Level2 to get the information and decide where to go to



Navigate Forward Upside-down-U-Turn

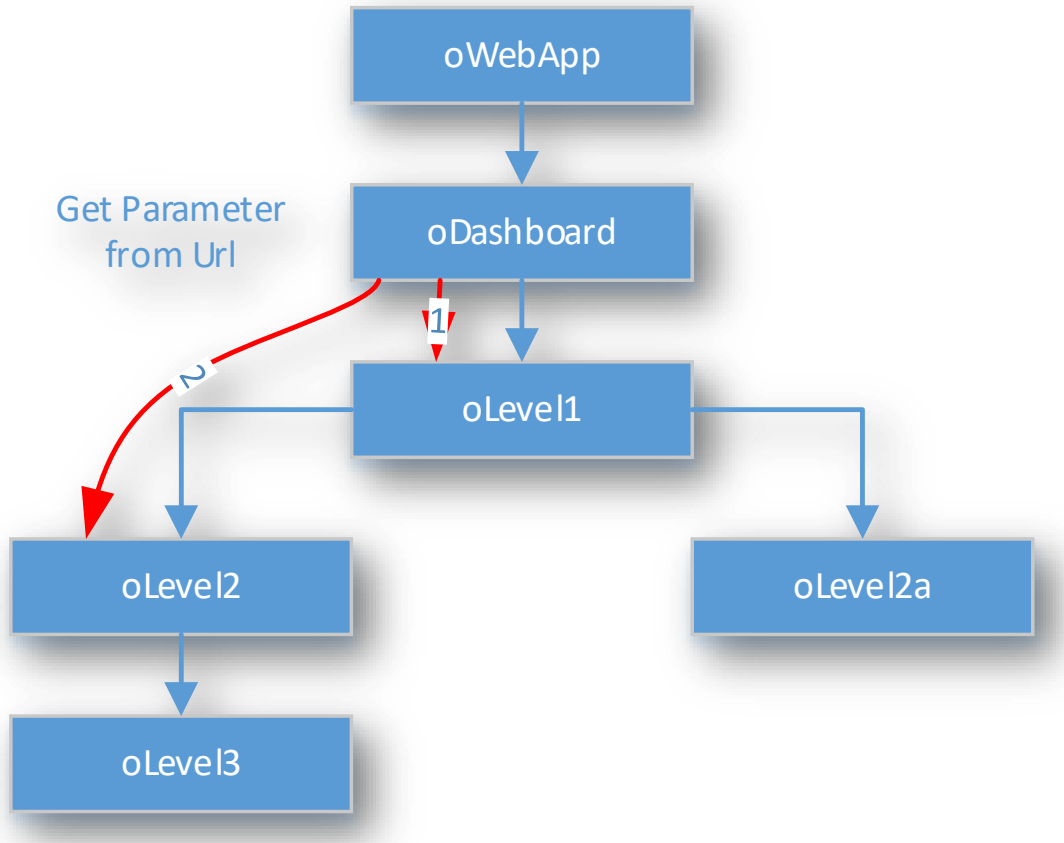
- Create sibling Level2 view called Level2a. Allow direct navigation from Level3 to Level2a. Requires closing to Level1 and forward to Level2a.
- Steps...
  - Copy Level2 to Level2a
  - Create a button in Level1 to activate Level2a
  - Remove the button in Level2a to activate Level3
  - Create a button in Level3 to directly activate Level2a
  - Going back and forward requires:
    - Close to the lowest common level (Level1 in this case)
    - Store target view in NavigateData (OnGetNavigateBackData)
    - OnNavigateBack to activate other view



Load App from URL

Activate on URL Parameter

- Pass URL parameters to activate specific view with path
- Steps...
  - Use Get UrlParameter in OnLoad of oDashboard
  - Store in NavigateData of oDashboard
  - Activate views in right order
  - Provide the URL parameters using OnGetNavigateForwardData of oDashboard
  - Show the passed parameters in a WebForm object
  - Remove parameters after to avoid reuse
  - Try...



Pop out example

- Launch another tab preloaded with record directly from WebApp
- http://localhost/WebOrderMobileAtlanta

Questions…

Thank you!