

From straightforward to sophisticated: UI customization for IBM WebSphere Portal and WCM



Graham Harper

Application Architect, IBM Software Services for Collaboration
Graham_harper@uk.ibm.com



David Strachan

CTO, IBM Software Services for Collaboration
david.strachan@uk.ibm.com

Please Note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Abstract

- Delivering a great digital experience for your customers or employees depends on having a polished UI that meets your organization's unique needs.
- WebSphere Portal & WCM come with a full spectrum of UI customization options, from straightforward to sophisticated.
- In this session we will give you a flavour of that full range, starting with powerful customization options that can be applied more simply than you would imagine.
- At the other end of the scale, we'll explore the capabilities available if you need go much further in developing a custom UI, taking as an example a custom page-editing capability.
- We will look at the extension points and APIs provided by the product, the opportunities they present and some of the different possible design approaches to our page-editing case study.

Us

[Profile](#)
[Wikis](#)[Communities](#)
[Files](#)[Blogs](#)
[Bookmarks](#)[Forums](#)
[Activities](#)

Strachan, David



**CTO, ISSC. Team Lead, ISSC EMEA
Centre of Excellence.**
EDINBURGH Great Britain

44-7769-880109

david.strachan@uk.ibm.com

[Profile](#)
[Wikis](#)[Communities](#)
[Files](#)[Blogs](#)
[Bookmarks](#)[Forums](#)
[Activities](#)

[Send Email](#) | [More Actions](#) ▼



Harper, G (Graham)



**Application Architect, Consulting IT
Specialist**

FELTHAM Great Britain

Building: 3 | Floor: NA | Office: MOBILE

44-20 88678287

graham_harper@uk.ibm.com

[Send Email](#) | [More Actions](#) ▼

You

- **Theme customization is something I know my project has to do but:**
 - I don't really know how to go beyond the basics
 - I know my client wants more than basic styling changes
 - I feel constrained by theme customization
- **In this session you will:**
 - Learn about the tools Portal v8 offers for theme customization and extension
 - Understand the circumstances under which you might need to extend the out-of-the-box Portal v8 theme
 - Review the APIs provided by Portal v8 to manage the page layout & contents
 - See some solution examples that you can use as inspiration
- **But this is not:**
 - An HTML or Java development lesson
 - Theme Customization 101

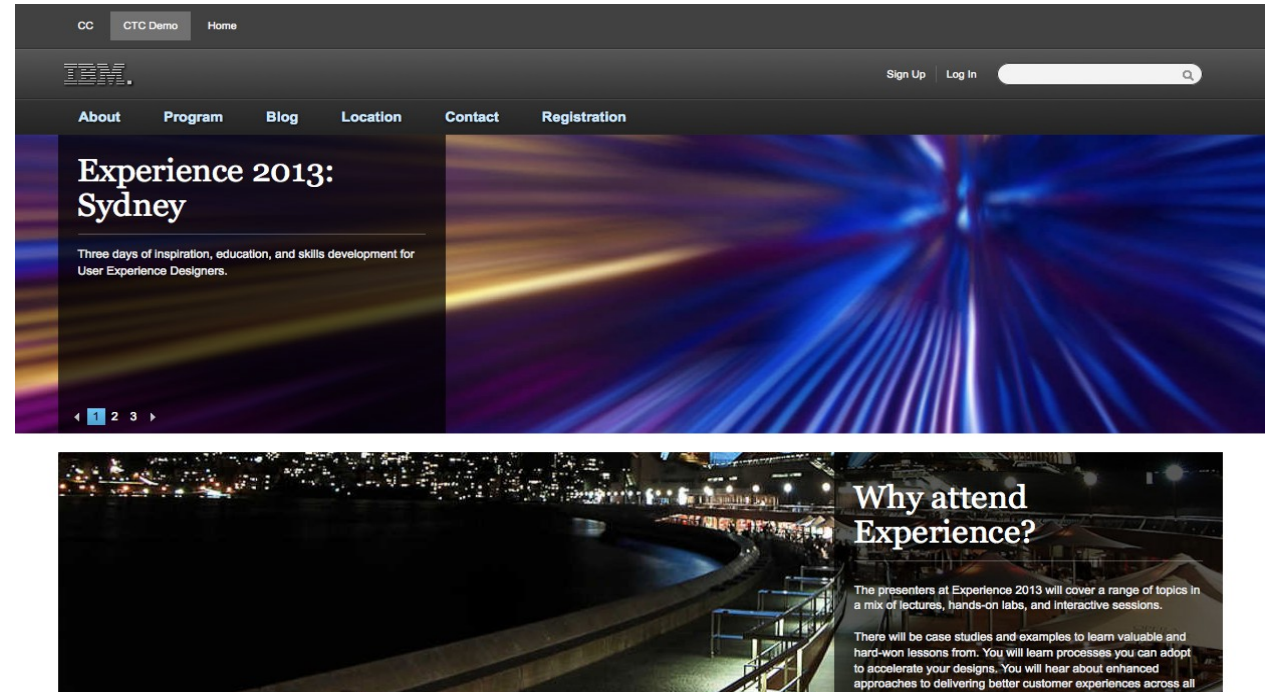
Agenda

- **Introduction**
- **What's the scope of what we're talking about**
- **Straightforward stuff**
 - View and edit modes
 - Easy theme customizations
 - The tools Portal gives you
- **Sophisticated stuff**
 - Client-side refresh of page changes
 - Enforcing a strict editing workflow
 - “Preview” mode for portlets on edited page
 - Editing portlet settings in constrained environment
- **Q&A**

Requirements from recent projects

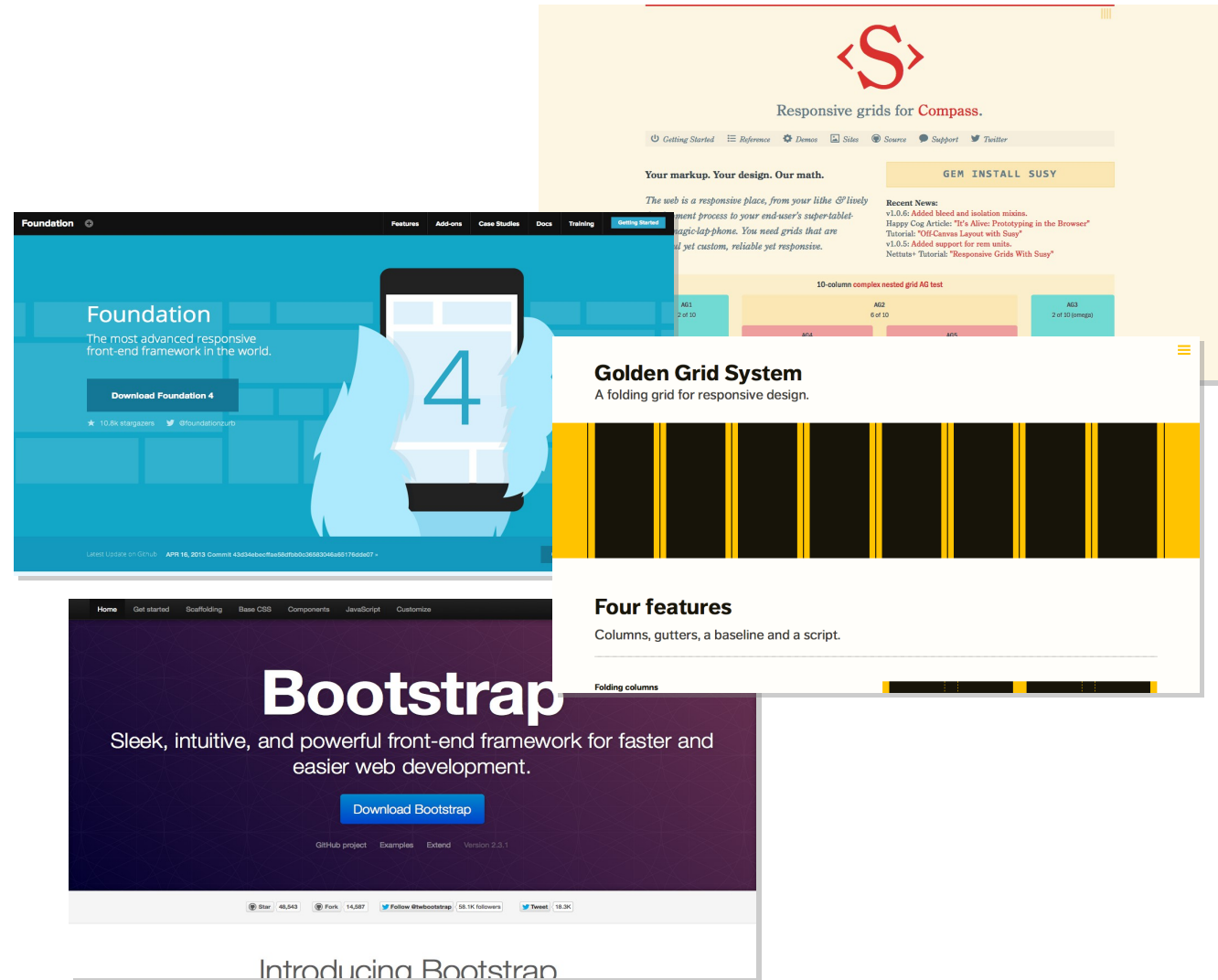
Everyone wants a custom theme

- Apply corporate branding
- Remove IBM “look”
- Deliver custom UI features



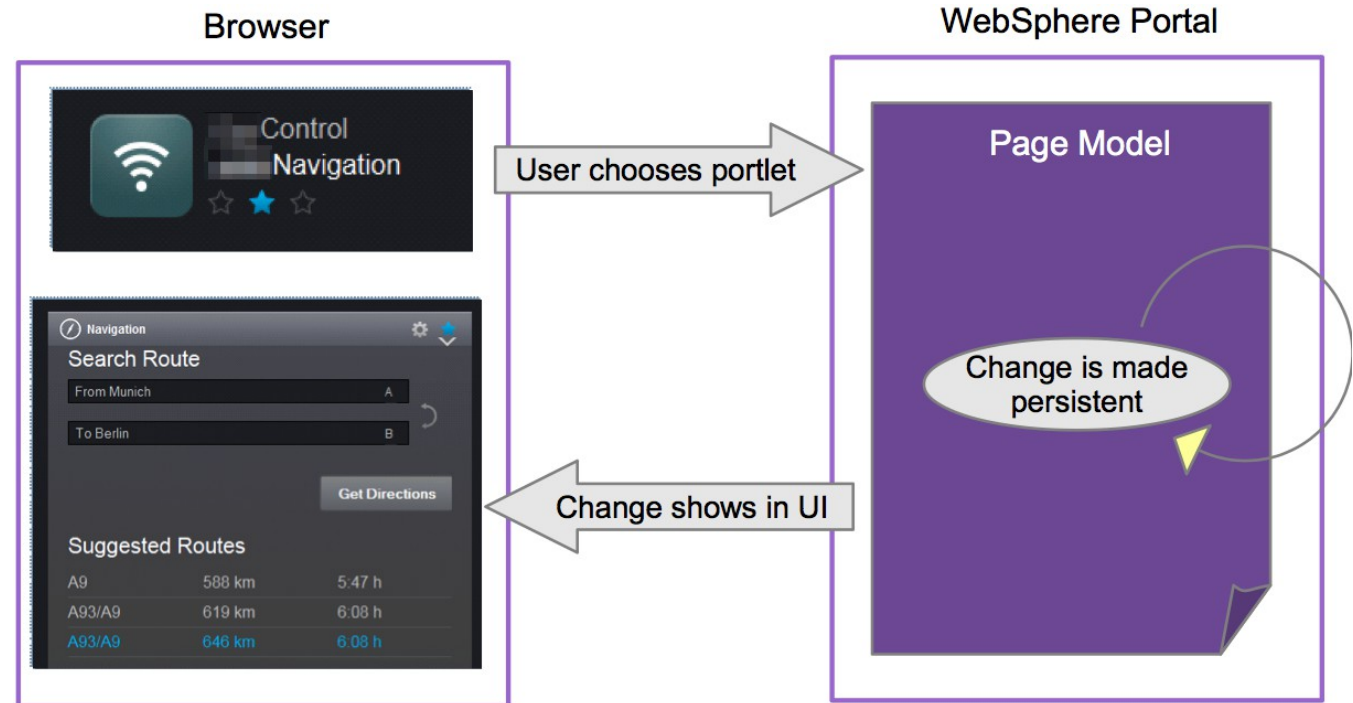
Customers ask for all sorts of things : easier stuff we've met

- Apply branding
- Use JQuery instead of Dojo
- Use their choice of responsive framework
- Optimize download size



Customers ask for all sorts of things : harder stuff we've met

- Custom page editing experience (e.g. strict editing workflow)
- Untemplated layout
- Client-side refresh of page changes
- “Preview” mode for portlets on edited page
- Editing portlet settings in constrained environment
- Mobile editing experience



IBM Software

Straightforward stuff

Portal 8.0 theme

- **One theme – called Theme 8.0**
 - Same theme modularization architecture from Portal 7.0.0.2
 - Replaces all previously shipped out of the box themes

- **Key features**
 - Modularization
 - Server Side Aggregation support
 - Portlet and iWidget support
 - Static html templates: theme, skin, layout with WebDAV editing
 - Dynamic-content provides means to inject server side logic into static templates without inserting code
 - Dojo 1.7

Portal page types

- **Dynamic Pages**

- “Traditional” pre-v7 portal pages, constructed of nested row and column containers added with the Manage Pages portlet
- Portlets are added and removed in the same way

- **Static Pages**

- Page layout completely described with HTML / CSS / images uploaded as a ZIP file
- Portlets embedded where desired by pseudo-CSS classes

- **“Page Builder” Style Static Pages**

- Page layout and portlet containers determined by a layout template
- Layout templates are HTML files, typically in the theme
- Portlets are added and removed using theme's page editing capabilities (or custom functionality)
- Layout template markup is “cached” in page until new layout selected or refresh forced by administrative action

Static templates

- **Static templates are stored in WebDAV**
- **Themes**
 - **theme.html** in /fs-type1/themes/<theme-name>/
- **Skins**
 - **skin.html** in /fs-type1/themes/Portal8.0/skins/<skin-name>/
- **Layouts**
 - **layout.html** in /fs-type1/themes/Portal8.0/layout-templates/<layout-name>/
- **When editing using OOB templates, remember they are localized!**
 - You must edit the localized template to see changes
 - Localized templates example:
 - /fs-type1/themes/Portal8.0/nls/theme_en.html

Dynamic content spots

▪ Dynamic spots

- Microformat defined in theme.html and other static files
- ``
- Parsed at runtime, href resolved and response streamed out
- Config in **WP_DynamicContentSpotMappings** resource environment provider

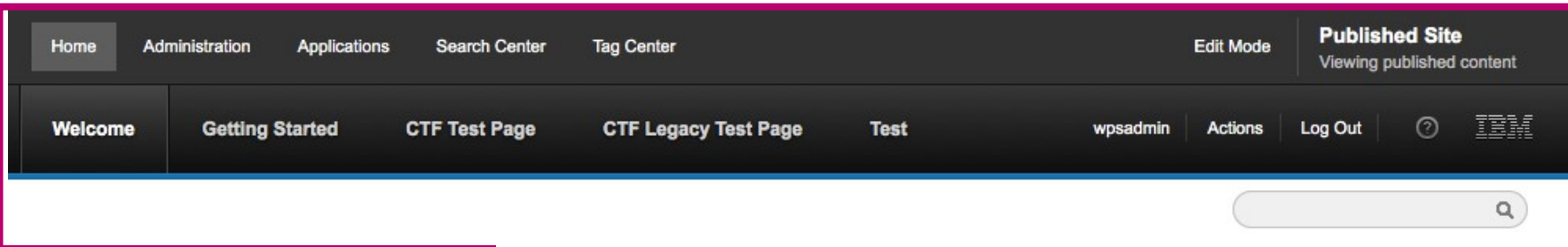
▪ 80theme_primaryNav

- `mvc:res:/wps/defaultTheme80/themes/html/dynamicSpots/navigation.jsp?rootClass=wpthemePrimaryNav`
`%2520wpthemeLeft&startLevel=1, smartphone@, tablet@`

▪ 80theme_mobileNav

- `mvc:smartphone/tablet@res:/wps/defaultTheme80/themes/html/dynamicSpots/navigation.jsp?rootClass=wpthemeMobileNav`
`%2520wpthemeLeft&startLevel=1&levelsDisplayed=2`

Theme.html



Welcome
We

```
<div class="wpthemeBanner">
  <div class="wpthemeBannerInner">
    <div class="wpthemeInner">
      <a rel="dynamic-content" href="dyn-cs:id:80theme_search"></a>
      <a rel="dynamic-content" href="dyn-cs:id:80theme_commonActions"></a>
      <a rel="dynamic-content" href="dyn-cs:id:80theme_mobileNav"></a>
    <div class="wpthemeClear"></div>
    </div>
  </div>
</div><!--end main banner-->

<div class="wpthemeBanner wpthemeBannerPrimaryNavigation">
  <div class="wpthemeBannerInner">
    <div class="wpthemeInner">
      <a rel="dynamic-content" href="dyn-cs:id:80theme_primaryNav"></a>
    <div class="wpthemeClear"></div>
    </div>
  </div>
</div><!--end primary nav banner-->
```

Skin.html

- ▼ Portal8.0
 - ▶ contributions
 - ▶ css
 - ▶ images
 - ▶ js
 - ▶ layout-templates
 - ▶ menuDefinitions
 - ▶ nls
 - Plain.html
 - preview_rtl.gif
 - preview.gif
 - preview.png
 - ▶ profiles
 - ▼ skins
 - ▼ Hidden
 - ▶ nls
 - skin.html
 - skin.js
 - ▶ NoSkin
 - ▶ Standard

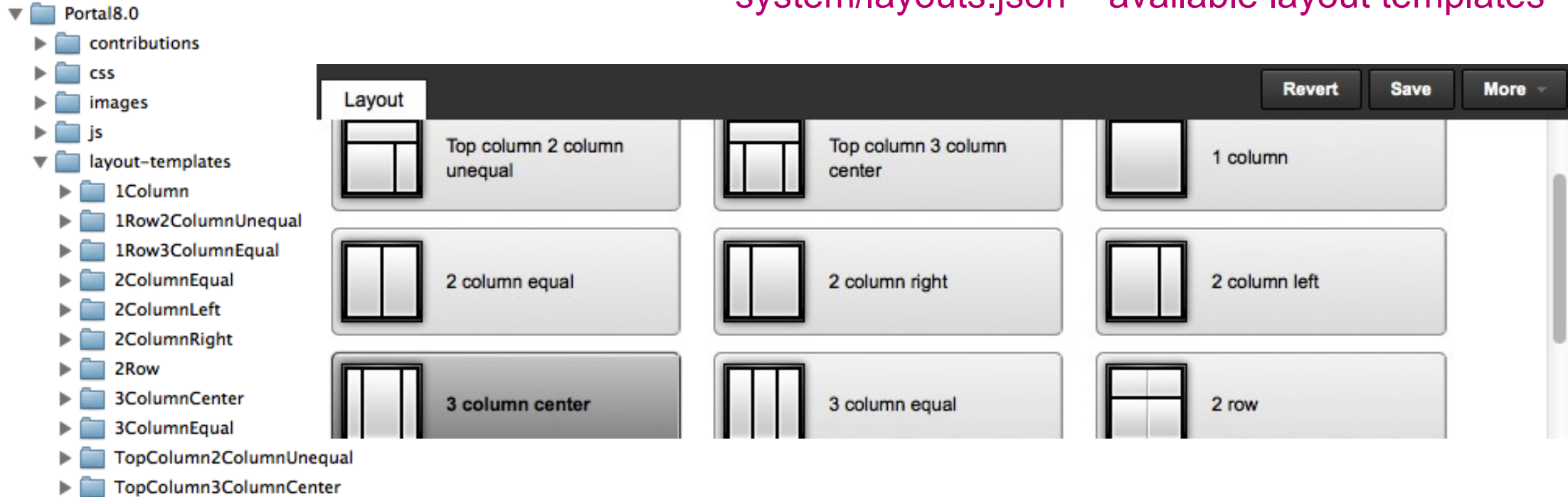
```
<div class="wpthemeControlBody wpthemeOverflowAuto wpthemeClear"> <
<!-- asa.overlay marks the node that the AsaOverlayWidget will be p
  <div style="position:relative">
    <div class="analytics.overlay" ></div>
  </div>
  <a rel="dynamic-content" href="lm:control"></a>
  <div class="wpthemeClear"></div>
</div>
```

Login

Welcome to the portal, wpsadmin

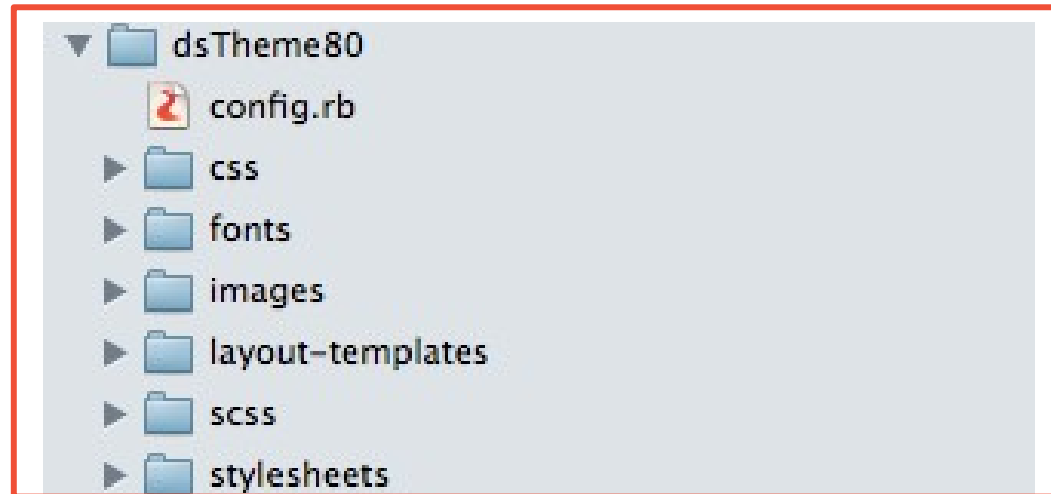
Layout.html

system/layouts.json – available layout templates



```
<div class="wptheme2Col wpthemeUnequal wptheme2ColLeft">
  <div class="component-container wpthemeCol wpthemePrimaryContainer ibmDndColumn wpthemeLeft
    wpthemeCol8of12 wpthemeWide" name="ibmMainContainer"></div>
  <div class="component-container wpthemeCol wpthemeSecondaryContainer ibmDndColumn wpthemeLeft
    wpthemeCol4of12 wpthemeNarrow" name="right"></div>
</div>
```

Place framework files

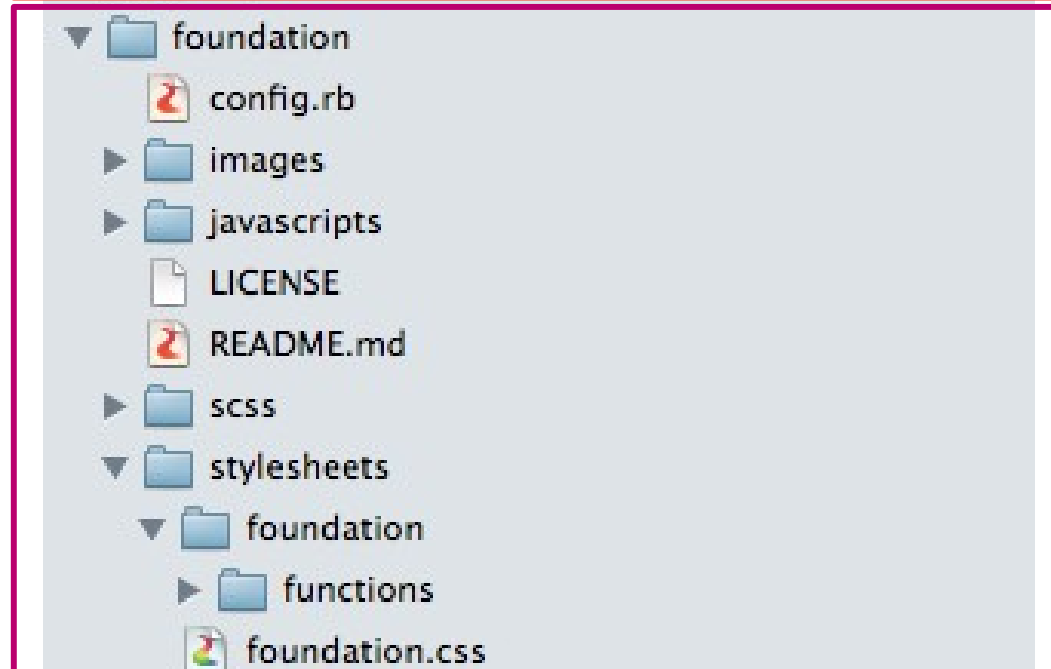


My custom files

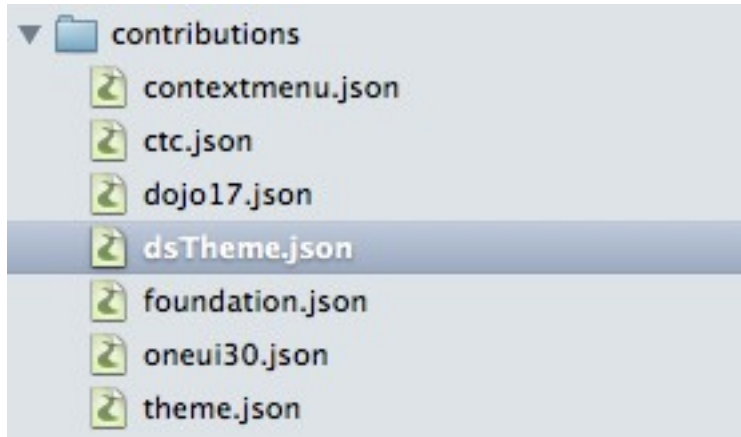
- Custom layout templates
- Custom styles
- Fonts

Framework CSS and JS

- In this example I'm using SCSS to manage stylesheets



Define theme contributions



```
{
  "modules": [{
    "id": "dsTheme",
    "prereqs": [{
      "id": "foundation"
    }],
    "contributions": [{
      "type": "head",
      "sub-contributions": [{
        "type": "css",
        "uris": [{
          "value": "/dsTheme80/stylesheets/theme-base.css"
        }]
      }],
      "type": "css",
      "uris": [{
        "value": "/dsTheme80/stylesheets/offcanvas.css"
      }]
    }, {
      "type": "css",
      "uris": [{
        "value": "/dsTheme80/stylesheets/layout-base.css"
      }]
    }
  ]
}]
}
```

3 Neat things with theme contributions

From contributions/CTC.json

```

    }, {
      "type": "css",
      "uris": [{
        "value": "/CTC/css/theme-mid-screen.css",
        "deviceClass": "smartphone"
      }]
    }
  ]
}

```

```

    }, {
      "type": "css",
      "uris": [{
        "value": "/CTC/css/theme-mid-screen.css"
      }, {
        "value": "/CTC/css/theme-mid-screenRTL.css",
        "type": "rtl"
      }]
    }
  ]
}

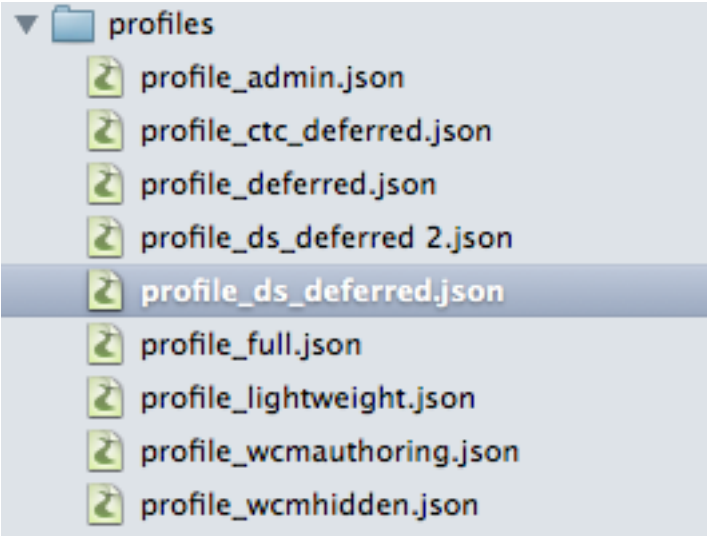
```

```

  "contributions": [{
    "type": "config",
    "sub-contributions": [{
      "type": "js",
      "uris": [{
        "value": "/CTC/js/nls/theme-contributions.js" },
        { "value": "/CTC/js/nls/theme-contributions_ar.js", "lang": "ar" },
        { "value": "/CTC/js/nls/theme-contributions_ca.js", "lang": "ca" },
        { "value": "/CTC/js/nls/theme-contributions_cs.js", "lang": "cs" },

```

Define a theme profile



```
{
  "moduleIDs": [
    "foundation",
    "dsTheme",
    "dsTheme_content"
  ],
  "deferredModuleIDs": [
    "wp_theme_widget",
    "wp_toolbar",
    "wp_project_menu_edit",
    "wp_preview_menu",
    "wp_analytics",
    "wp_content_mapping_picker",
    "wp_pagebuilder_controls",
    "wp_pagebuilder_dnd",
    "mm_new_page_dialog",
    "mm_builder_wiring",
    "mm_move_page",
    "mm_delete_page",
    "mm_delete_control",
    "mm_page_sharing_permission",
    "wcm_inplaceEdit",
    "wp_template_select_dialog",
    "wp_content_targeting_cam",
    "ctc_theme_contributions"
  ],
  "titles": [
    {
      "value": "DS Theme Content",
      "lang": "en"
    }
  ]
}
```


Define any dynamic content spots

▪ Dynamic spots

- Microformat defined in theme.html and other static files
- ``
- Parsed at runtime, href resolved and response streamed out
- Config in **WP_DynamicContentSpotMappings** resource environment provider

▪ dstheme_primaryNav

- `mvc:res:/wps/defaultTheme80/themes/html/dynamicSpots/navigation.jsp?rootClass=dsthemePrimaryNav%2520wpthemeLeft&startLevel=1, smartphone@, tablet@`

▪ dstheme_mobileNav

- `mvc:smartphone/tablet@res:/wps/defaultTheme80/themes/html/dynamicSpots/navigation.jsp?rootClass=dsthemeMobileNav%2520wpthemeLeft&startLevel=1&levelsDisplayed=2`

Build theme.html

```
<!DOCTYPE html>

<!-- paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/ -->
<!--[if IE 8]>    <html class="no-js lt-ie9" lang="en"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js" lang="en"> <!--<![endif]-->
<head>
    <meta charset="utf-8" />

    <!-- Set the viewport width to device width for mobile -->
    <meta name="viewport" content="width=device-width" />

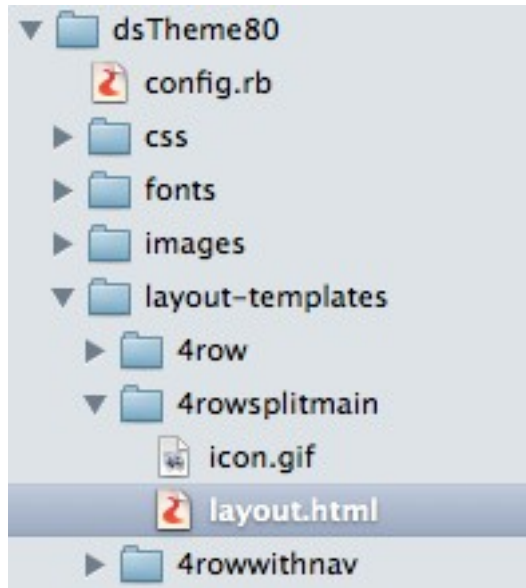
    <link rel="dynamic-content" href="co:head?optimized=false">
    <link rel="dynamic-content" href="dyn-cs:id:80theme_head">

</head>

<body id="page" class="off-canvas hide-extras lotusui30dojo tundra locale_en linen">

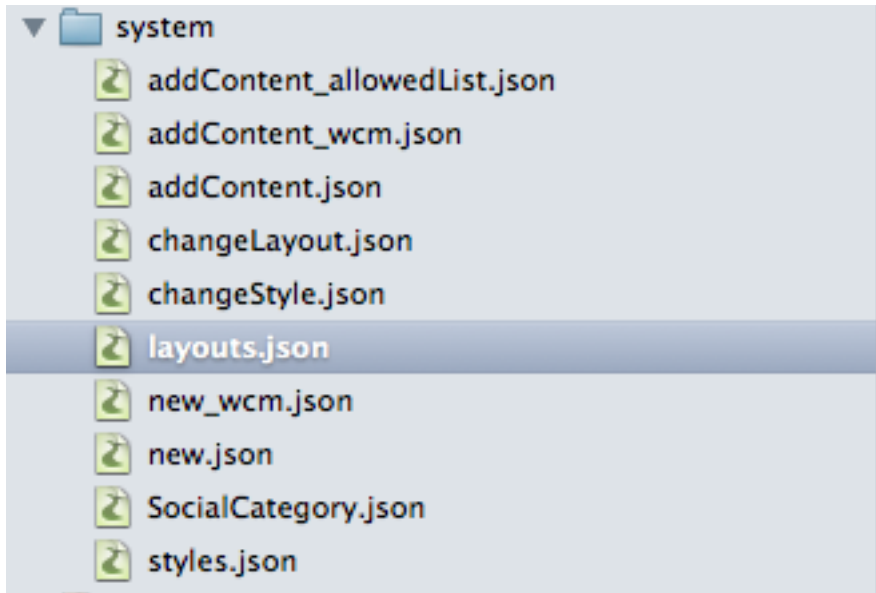
<div class="show-for-medium-down mobileNavigationContainer">
    <div class="wpthemeLeft">
        <a class='mobileNavigationToggle left' id="sidebarButton" href="#sidebar" ></a>
    </div>
    <div class="cornerCardLogo">
        
    </div>
    <div class="wpthemeRight">
        <a rel="dynamic-content" href="dyn-cs:id:80theme_commonActions"></a>
    </div>
</div>
```

Build layout templates



```
<div class="hiddenWidgetsDiv">
  <div class="component-container hiddenWidgetsContainer ibmDndColumn" name="ibmHiddenWidgets">
    <div style="clear:both"></div>
  </div>
  <div class="pre contentLeader">
    <div class="row">
      <div class="component-container wpthemeHeadlineContainer ibmDndColumn inner twelve columns"
        <div style="clear:both"></div>
      </div>
    </div>
    <div class="contentLeader">
      <div>
        <div class="component-container wpthemeHeadlineContainer ibmDndColumn inner twelve" name="l"
          <div style="clear:both"></div>
        </div>
      </div>
      <div class="contentMain">
        <div class="row">
          <div class="component-container ibmDndColumn eight columns" name="ibmMainContainer"></div>
          <div class="component-container rightContainer ibmDndColumn four columns" name="right"></di
          <div class="contentClear"></div>
        </div>
        <div class="contentClear"></div>
      </div>
    </div>
  </div>
</div>
```

Register layout templates



```
{
  'label': 'change_layout_4row',
  'url': ibmCfg.themeConfig.themeWebDAVBaseURI+'dsTheme80/layout-templates/4row/',
  'id': '4row',
  'thumbnail': ibmCfg.themeConfig.themeRootURI+'/dsTheme80/layout-templates/4row/icon.gif',
  'help': ''
},
```

IBM Software

Sophisticated stuff

Portal Model SPIs

- There are many model SPIs giving read access to various parts of the portal configuration, such as:
 - **(Admin)PortletModel** – portlets and their configuration
 - **ContentModel** – the page hierarchy and page data (title, description, etc.)
 - **ContentMetaDataModel** – metadata for nodes of the ContentModel
 - **LanguageList** – supported languages within WebSphere Portal
 - **LayoutModel** – the layout of a page
 - **MarkupList** – supported markup languages
 - **NavigationModel** – the navigation topology visible to a specific user
 - **NavigationSelectionModel** – the currently selected navigation nodes
 - **SkinList** – the list of skins
 - **ThemeList** – the list of themes

Model SPI usage examples

The screenshot shows the IBM WebSphere Portal interface. The top navigation bar includes the IBM logo, the text 'WebSphere Portal', and user controls for 'wpsadmin', 'Actions', 'Log Out', and a search bar. The left sidebar contains a menu with categories like 'Portal User Interface', 'Theme', 'Page Template', 'Sidenav Configuration', 'Controller Sample', 'Investigations', 'Portlet Management', 'Web Modules', 'Applications', 'Portlets', 'Web Services', 'Web Clipping', 'Virtual Web Application Manager', 'Access', and 'Users and Groups'. The main content area is titled 'Manage Users and Groups' and includes search filters for 'User groups' and 'All available', along with 'New Group' and 'New User' buttons. Below these are two tables: 'Users and Groups' and 'All Portal User Groups', both showing 'Page 1 of 1'. Annotations highlight four Model SPI usage examples: 'Selection Mode' (orange box pointing to the breadcrumb 'WebSphere Portal > Access > Users and Groups'), 'Content Model' (blue box pointing to the 'Portlet Management' menu item), 'Layout Model' (red box pointing to the main content area), and 'Portlet Model' (purple box pointing to the 'Manage Users and Groups' portlet).

Selection Mode

Content Model

Layout Model

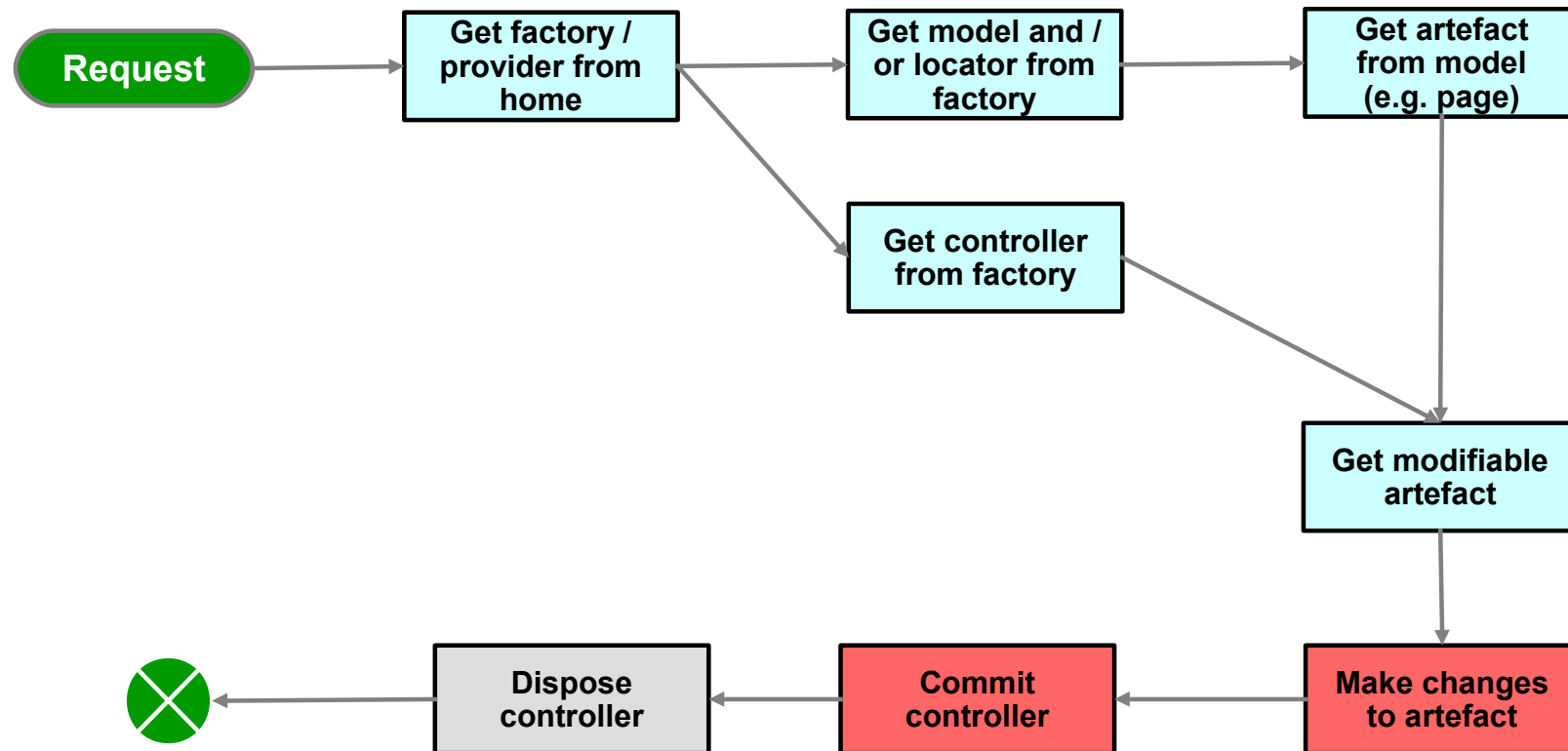
Portlet Model

Portal Model Controller SPIs

- **To make updates we use *controller* SPIs associated with the models, mainly:**
 - ContentModelController
 - LayoutModelController
 - PortletModelController

- **The basic steps of working with a controller are:**
 - 1) Obtain the controller from a factory
 - 2) Obtain a modifiable instance of an artefact from that controller
 - 3) Apply your changes to that instance
 - 4) Commit the controller to save the changes
 - 5) Dispose the controller, if necessary

Model Controller usage flow



Portal SPIs hints and tips

- **Controller factories are usually themselves obtained from a “home” interface, looked up via JNDI**
 - These “homes” can be cached for the lifetime of the JVM to save expensive JNDI lookups
- **Some (but not all) controllers have a “dispose()” method that frees up resources they have reserved**
 - Where present, these methods should always be called once a controller is no longer needed for a given request (e.g. in a “finally” block)
- **May need to “re-find” newly created resources (e.g. pages, portlet instances) before serializing their object IDs to text**

Page rendering via Portal URLs

- Portal will provide the markup for the “portlet area” of a page (i.e. without the markup generated by the theme) in response to a URL of the form:

`/wps/myportal?uri=lm:oid:<page-id>`

- Or the markup for a single portlet instance on page in answer to:

`/wps/myportal?uri=lm:oid:<portlet-instance-id>@oid:<page-id>`

Page updates via Portal URLs

- Portal provides a REST API for accessing and updating resources, using URLs such as:

`/wps/mycontenthandler?uri=<model>:<resource-id>`

where “model” is the appropriate model abbreviation, such as: *content model (cm)*, *navigation model (nm)*, *layout model (lm)* or *portlet model (pm)*

- So, an example call to get the content (page) information for the content root would be:

`/wps/mycontenthandler?uri=cm:oid:wps.content.root`

Portal REST / ATOM feeds

- **Scope is a subset of Java SPI**
- **ContentModel**
 - obtain and modify the content topology and the properties of content nodes such as pages, labels and content URLs.
- **NavigationModel**
 - obtain the navigation topology only
- **LayoutModel**
 - obtain and modify the layout of a page
- **PortletModel**
 - obtain, create, update and delete portlets
- http://www-10.lotus.com/ldd/portalwiki.nsf/dx/Remote_Model_SPI_REST_service_wp8

IBM Software

Solution example

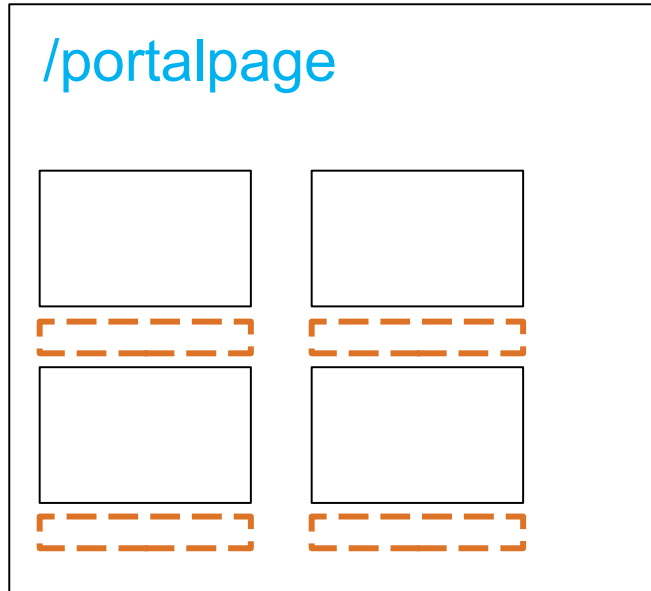
Solution example

- **Creating a page customizer:**
 - Client-side refresh of page changes
 - Enforcing a strict editing workflow
 - “Preview” mode for portlets on edited page
 - Editing portlet settings in constrained environment

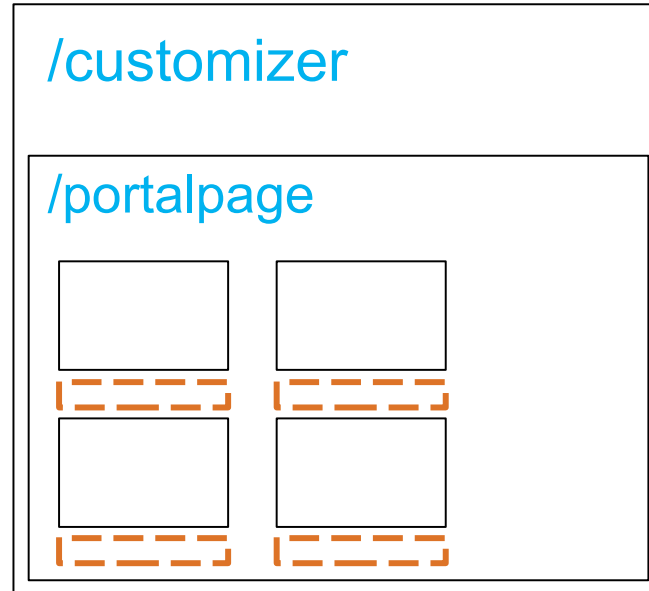
Page customizer design considerations

- Editing page “in situ” vs from another page
- iframes vs “in page” rendering
- Portlets vs theme components
- Enabling editing of portlet settings
- Client-side vs server-side update operations

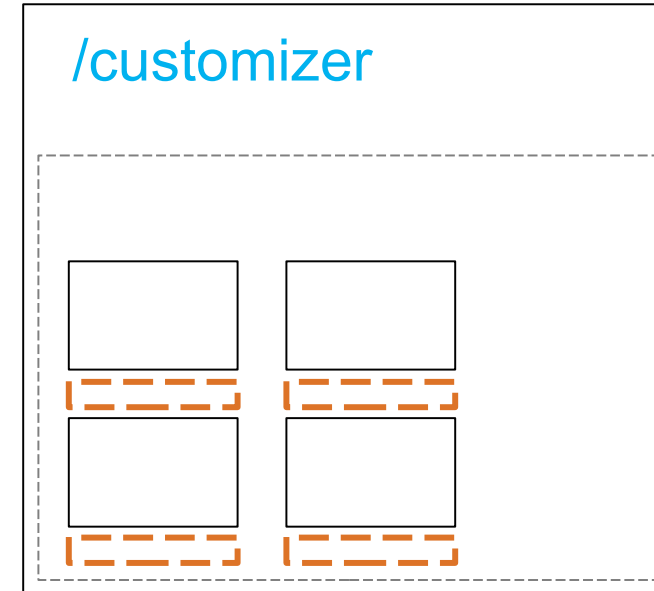
Three basic patterns



Edit page inline

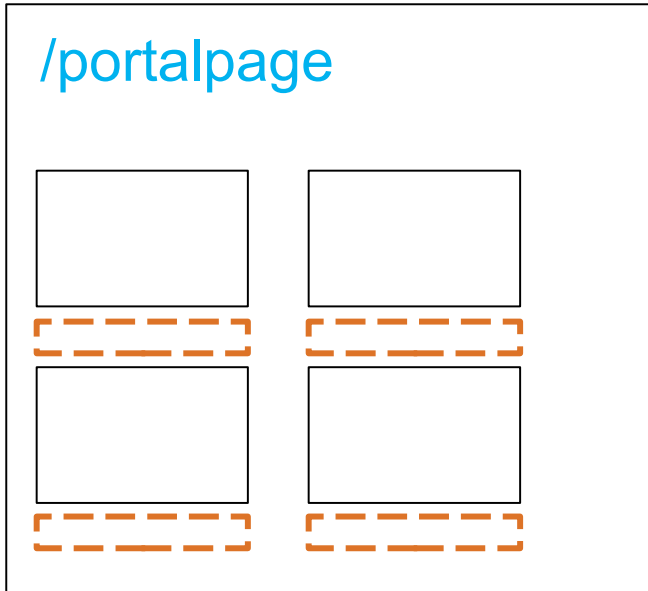


Dedicated customizer page.
Page preview in an
iframe.



Dedicated customizer page.
Page preview inline.

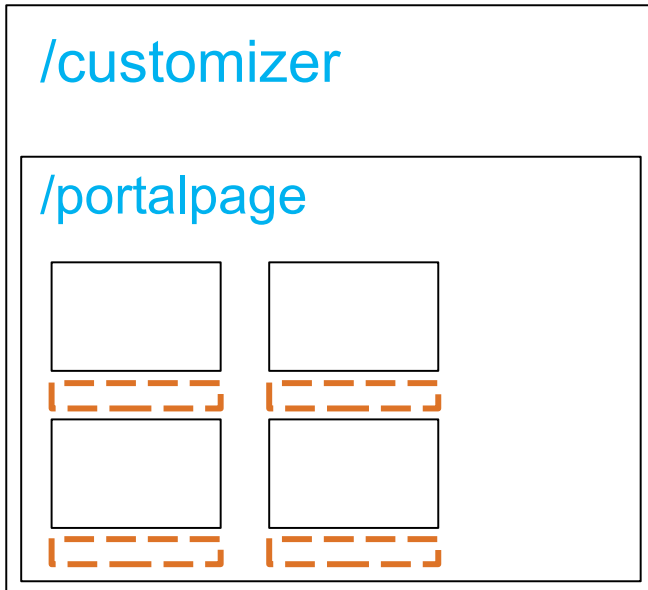
Inline customizer



Edit page inline

- **Editing page "in situ"**
 - Put the current page into "edit mode"
 - Theme components required to provide editing capabilities
 - Approach used in the 8.0 out-of-the-box functionality
 - Standard navigation components show that the user is still "on" the page
 - Saving and full page refresh usually required to see changes
- **Editing using theme components**
 - Tied to the theme
 - Less common and standardized programming model

Customizer page – preview in iframe



Dedicated customizer page.
Page preview in an iframe.

- **Editing page from a different page**

- Typically portlet(s) on the customizer page do the editing
- Good when no WYSIWYG editing interface required (but not only then as we will see!)
- Good control over editing workflow possible
- Allows other portlets to be present during the editing process, even though not on the edited page
- Standard navigation components show that user is on the customizer page, not the edited page

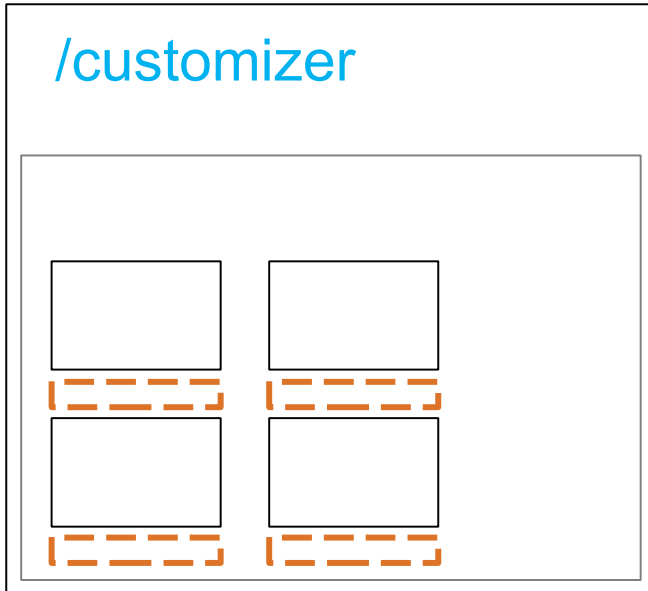
- **iframe use**

- Allows separation of “preview” of edited page from other components e.g. portlet list
- Therefore only the iframe content will usually need to be refreshed on changes
- Edited page in iframe will need to handle e.g. drop events, so custom theme components will probably be needed
- Links in portlets on edited page can be active and will affect only the iframe (e.g. switching a portlet into edit mode)

- **Editing using portlets**

- Well-known and standardized (JSR 286) programming model
- Can be deployed on pages along with other portlets using standard portal administration tools

Customizer page – preview inline



Dedicated customizer page.
Page preview inline.

- **Editing page from a different page**

- Typically portlet(s) on the customizer page do the editing
- Good when no WYSIWYG editing interface required (but not only then as we will see!)
- Good control over editing workflow possible
- Allows other portlets to be present during the editing process, even though not on the edited page
- Standard navigation components show that user is on the customizer page, not the edited page

- **No iframe use**

- No custom theme components are required when editing from another page - functionality can all be under the control of portlets
- Rendered page or portlet instance markup can be requested from portal and embedded
- However, will usually need to save page changes before requesting
- Links in portlets on the edited page should not be live or will navigate to that page and away from customizer page (so need a different solution for editing portlet settings)

- **Editing using portlets**

- Well-known and standardized (JSR 286) programming model
- Can be deployed on pages along with other portlets using standard portal administration tools

Enabling editing of portlet settings

- **Editing page “in situ”**
 - Can use standard links to edit settings, once page saved and portlets rendered
 - Would need to change links (i.e. update the skin) to accommodate requirements such as maximised edit mode or a change to the navigation display
- **Editing page from a different page, preview in an iframe**
 - Can use standard links to edit settings, once page saved and iframe refreshed
 - Link effects restricted to changing iframe content and constrained to size of iframe
 - Can't accommodate requirements such as change to navigation display
- **Editing page from a different page, preview rendered “in page”**
 - Standard links can't be used or will lose context
 - Can create new links to navigate as desired
 - e.g. to open the edited page with a particular portlet in edit mode and maximised and with a different theme navigation displayed
 - Need a way to get back, placing some requirements on the portlet's edit mode

Client-side vs server-side update operations

▪ Client-side updates

- Portal provides a REST API for page update operations
- REST API is a subset of the server-side SPI
- Potentially quite “chatty” in the number of requests and responses required

▪ Server-side updates

- Very powerful server-side model SPI provided by portal
- Can reduce the number of requests from client-side code by providing coarse-grained operations (e.g. through the “serveResource” method of a portlet)

Demo page customizer solution

▪ To show:

- Client-side refresh of page changes
- Enforcing a strict editing workflow
- “Preview” mode for portlets on edited page
- Editing portlet settings in constrained environment

▪ We have selected the following design options:

- Edit the page from a different page
- In-page rendering of edited page (no iframes)
 - Rendered portlets are “masked” to be read-only
- Encapsulate functionality in portlets, not theme components
- Perform updates server-side
 - Page customizer portlet provides a coarse-grained “API” to the client-side code

Demo

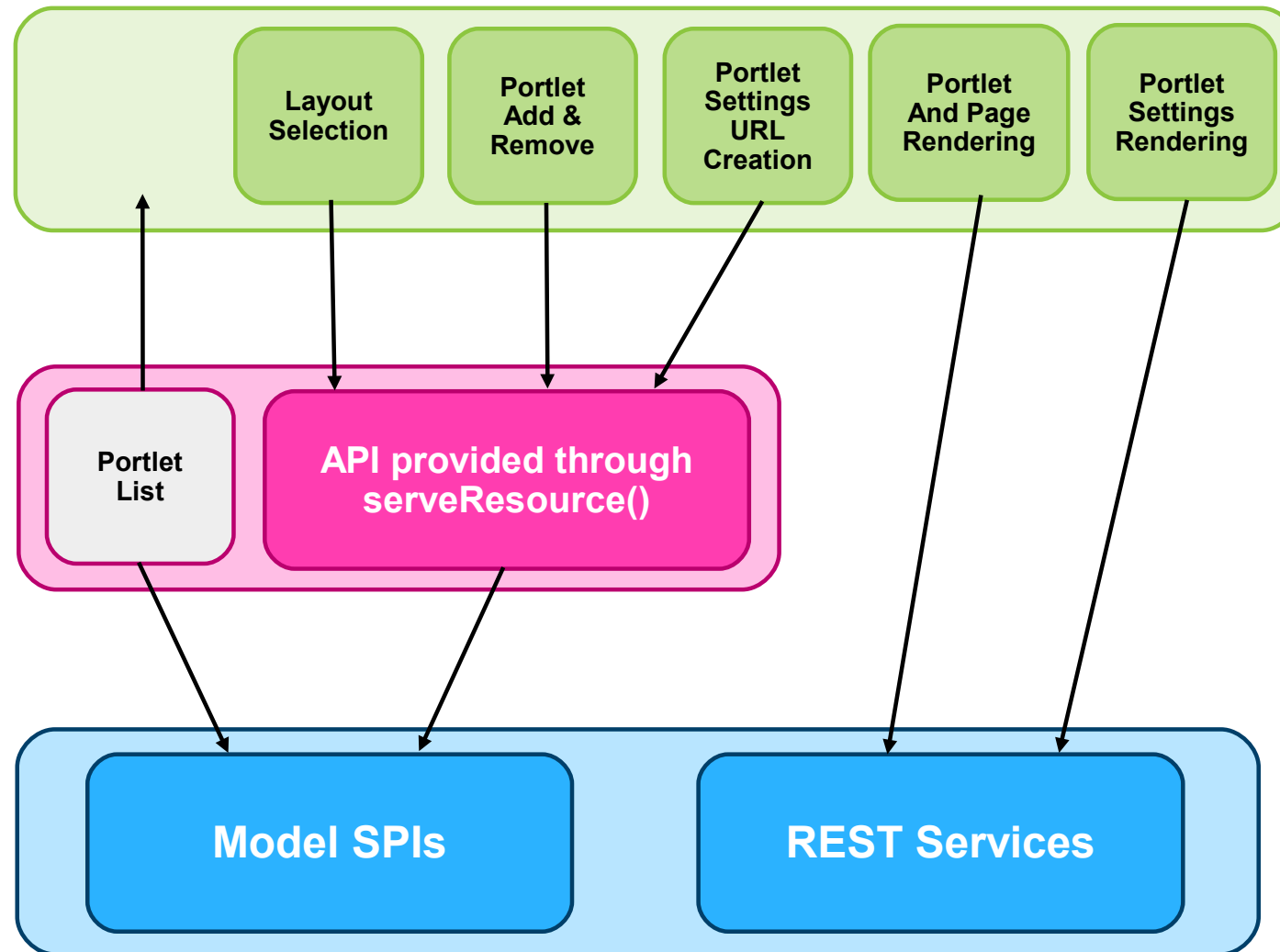
- Client-side refresh of page changes
- Enforcing a strict editing workflow
- “Preview” mode for portlets on edited page
- Editing portlet settings in constrained environment

Layers of the demo page customizer

**Page Customizer
Portlet
(Client Side)**

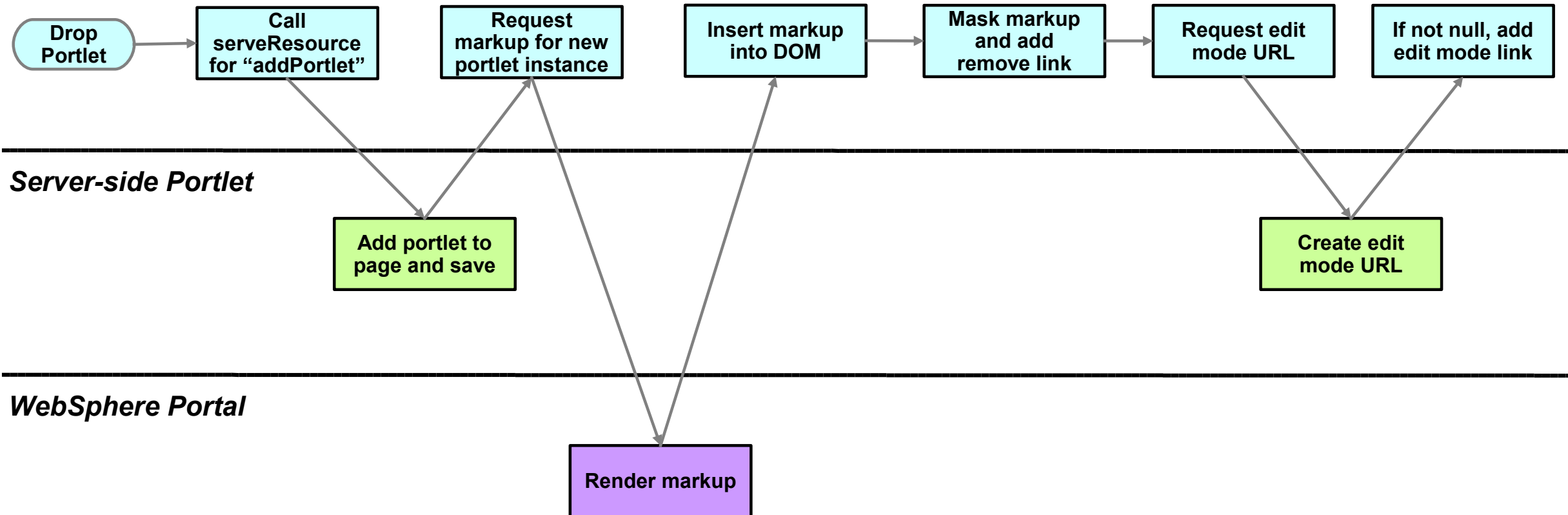
**Page Customizer
Portlet
(Server Side)**

**WebSphere
Portal**



Adding a portlet to a page

Client-side JavaScript



Questions

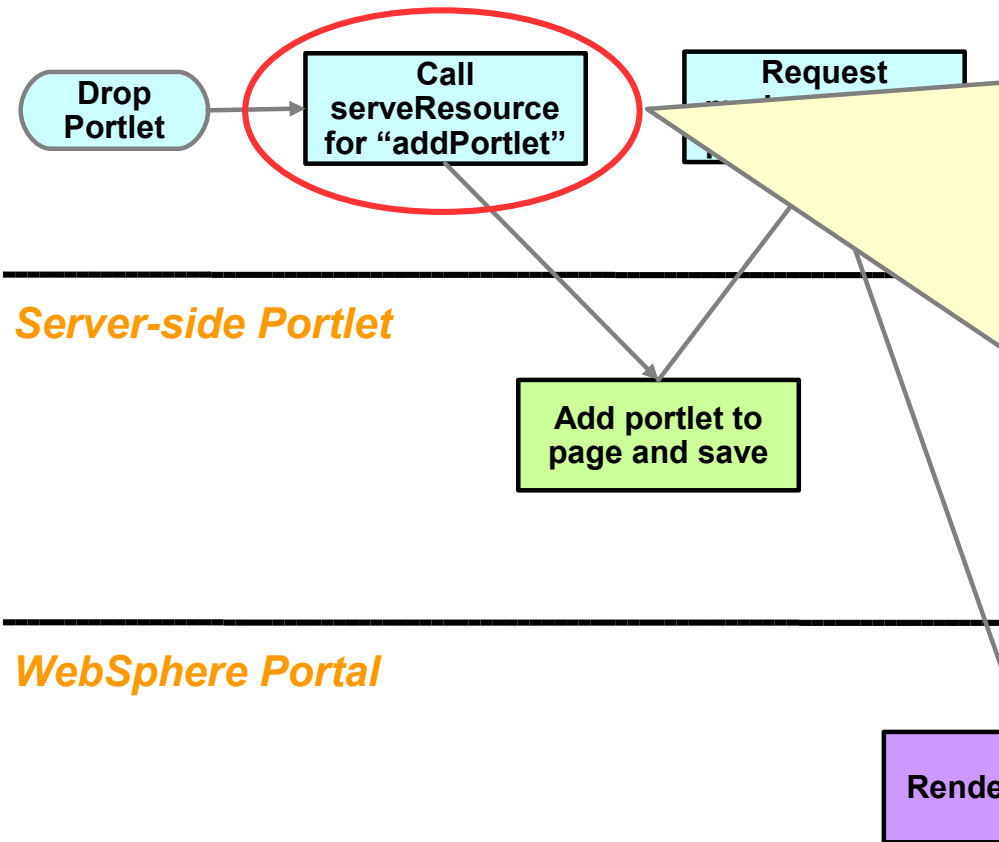


Reference slides

Adding a portlet to a page – calling `serveResource()`

IBM Software

Client-side JavaScript



Server-side Portlet

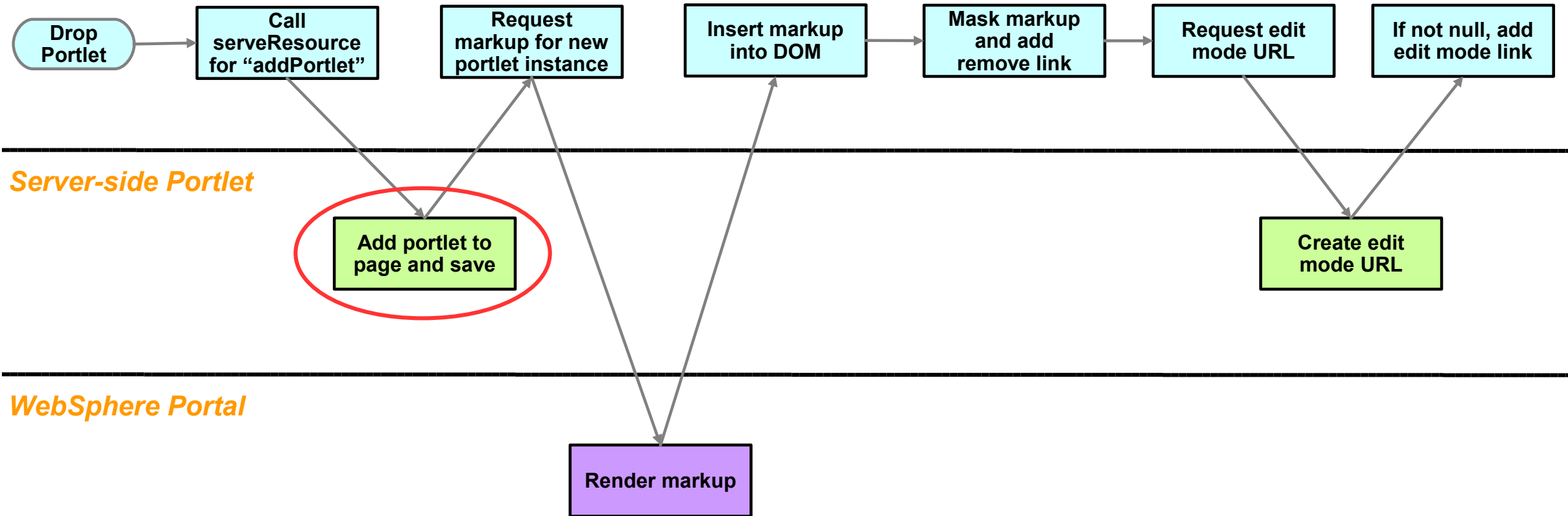
WebSphere Portal

```
$('.portlet-library').on('dropitem', function (evt, data) {  
    var portlet = data.$el;  
    portlet.css('min-height', 150);  
    // Add a container to the portlet node.  
    var portletContainer = $('<div class="portlet-container"></div>');  
    portlet.html(portletContainer);  
    // Look for a successor ID  
    var successor =  
        portlet.next('.portlet').attr('id');  
    // Make AJAX call to add the portlet to the page  
    $.get("#{pc_PortletSelection.addPortletURL}",  
        {portletId: data.id, targetContainerId:  
            portlet[0].parentNode.id, successor:  
            successor}, "json")  
        .done(function(jqxhr){  
            ...  
        })  
    );  
});
```

Adding a portlet to a page – *updating the model server-side*

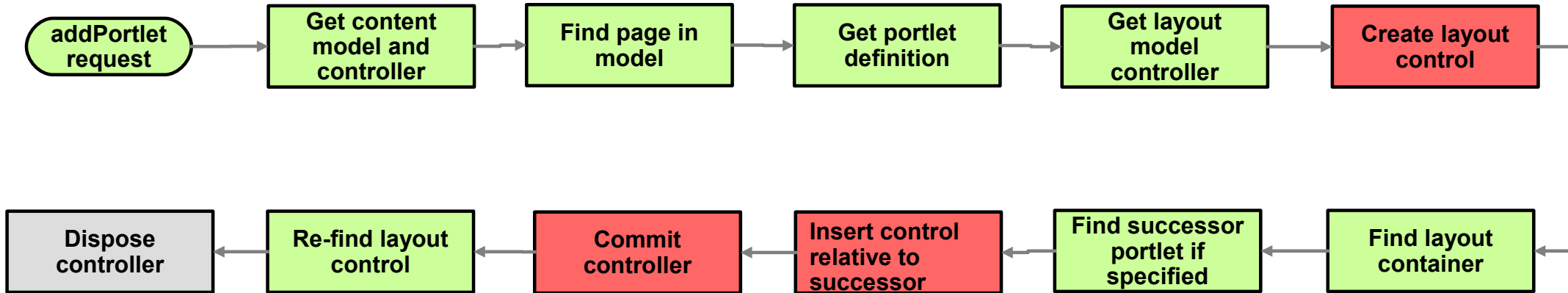
IBM Software

Client-side JavaScript



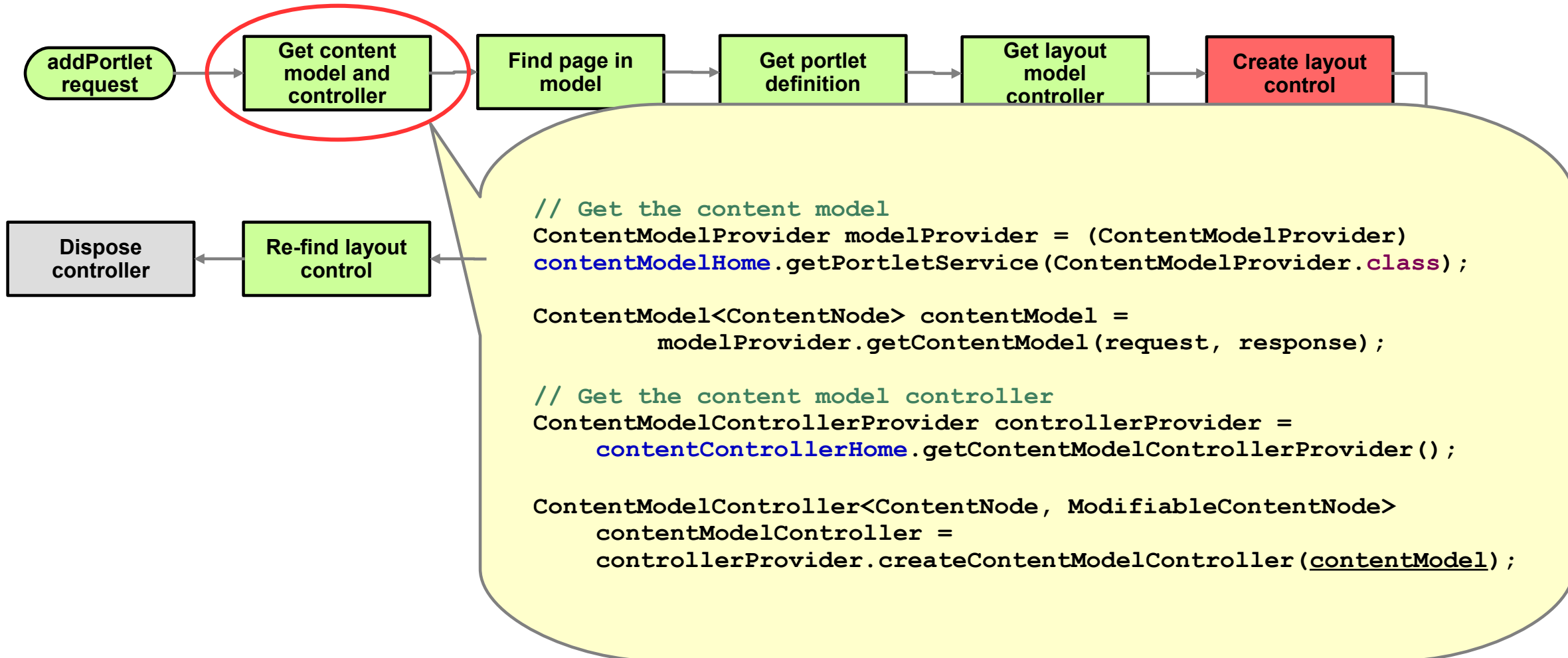
Adding a portlet to a page – *model update drill-down*

IBM Software



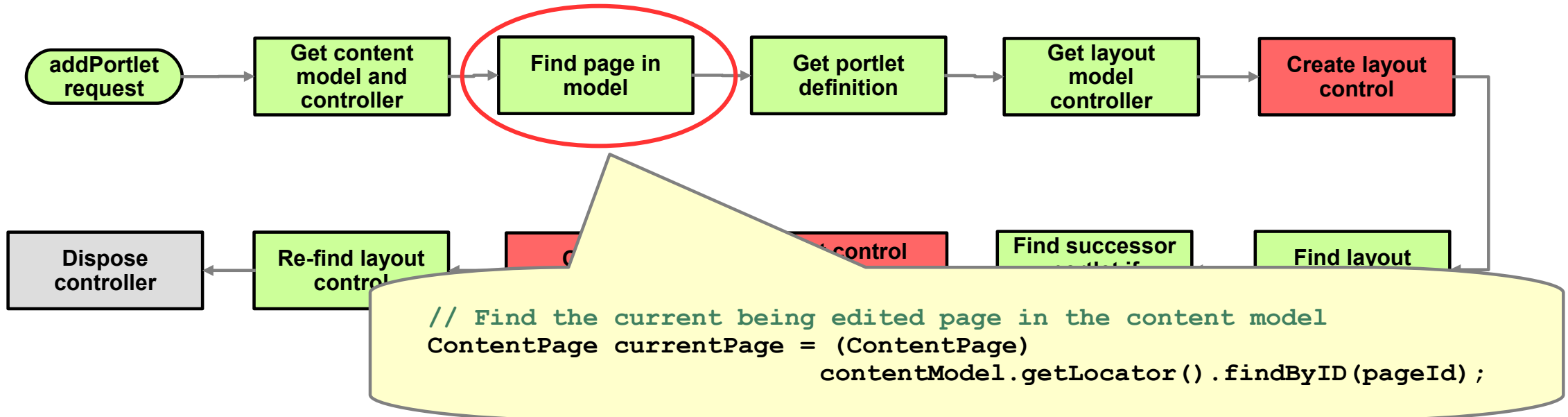
Adding a portlet to a page – *model update drill-down*

IBM Software



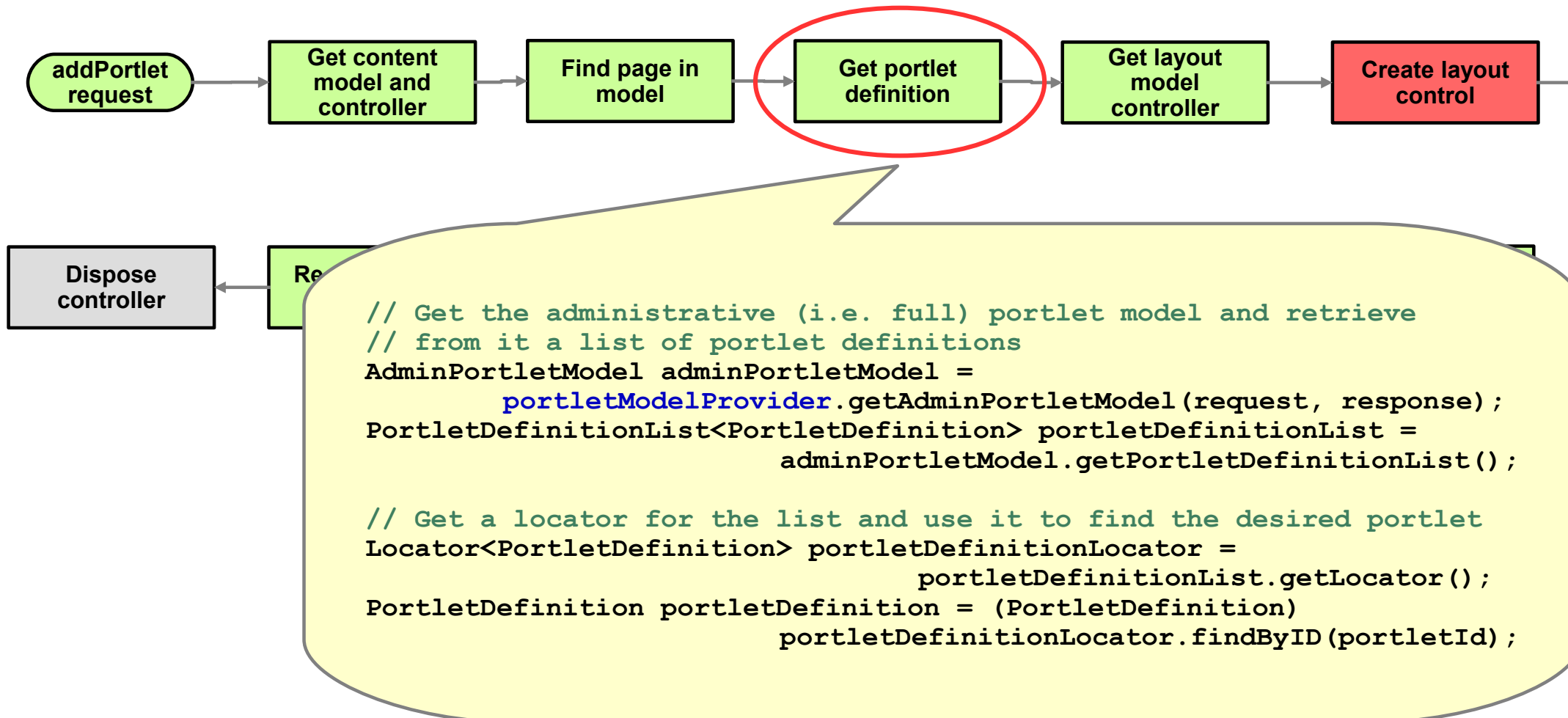
Adding a portlet to a page – *model update drill-down*

IBM Software



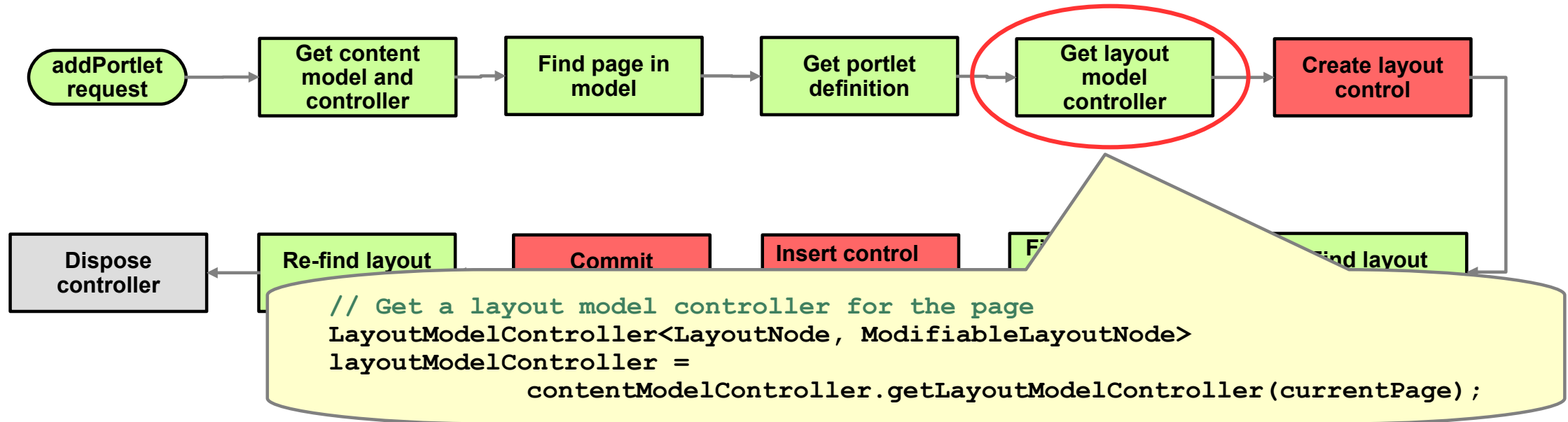
Adding a portlet to a page – *model update drill-down*

IBM Software



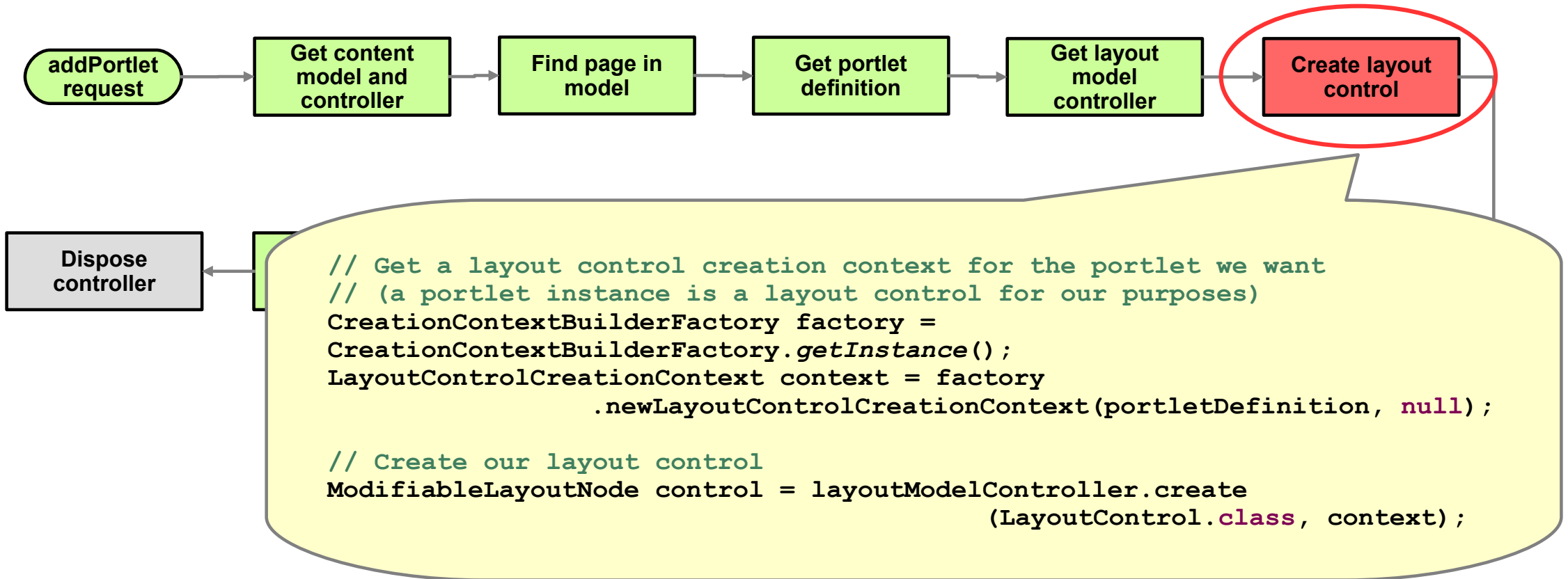
Adding a portlet to a page – *model update drill-down*

IBM Software



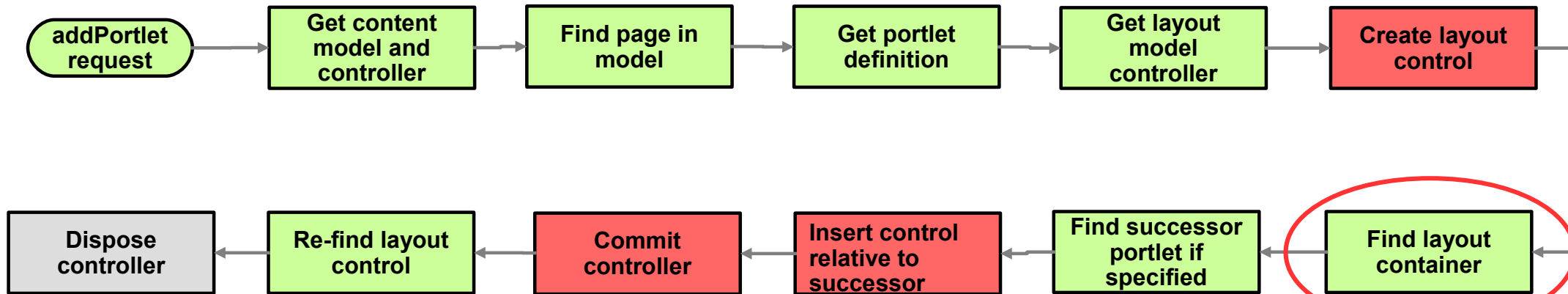
Adding a portlet to a page – *model update drill-down*

IBM Software



Adding a portlet to a page – *model update drill-down*

IBM Software

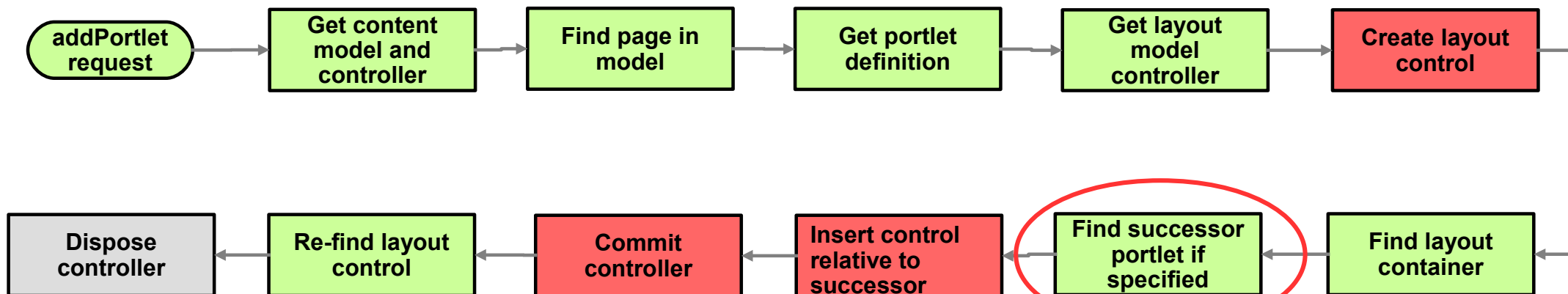


```
// Find the layout container we want to add to in the model
Locator<LayoutNode> locator = layoutModelController.getLocator();

LayoutContainer container = (LayoutContainer) locator.findByID(containerId);
```

Adding a portlet to a page – *model update drill-down*

IBM Software

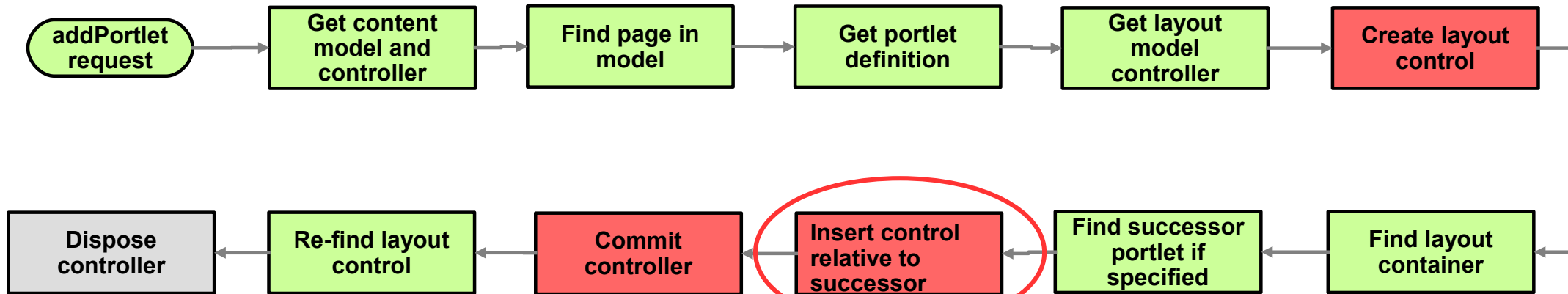


```
// If a next portlet instance is provided, find it in the layout
// model
LayoutNode nextPortletInstance = null;

if (nextPortletInstanceId != null) {
    nextPortletInstance = locator.findByID(nextPortletInstanceId);
}
```


Adding a portlet to a page – *model update drill-down*

IBM Software

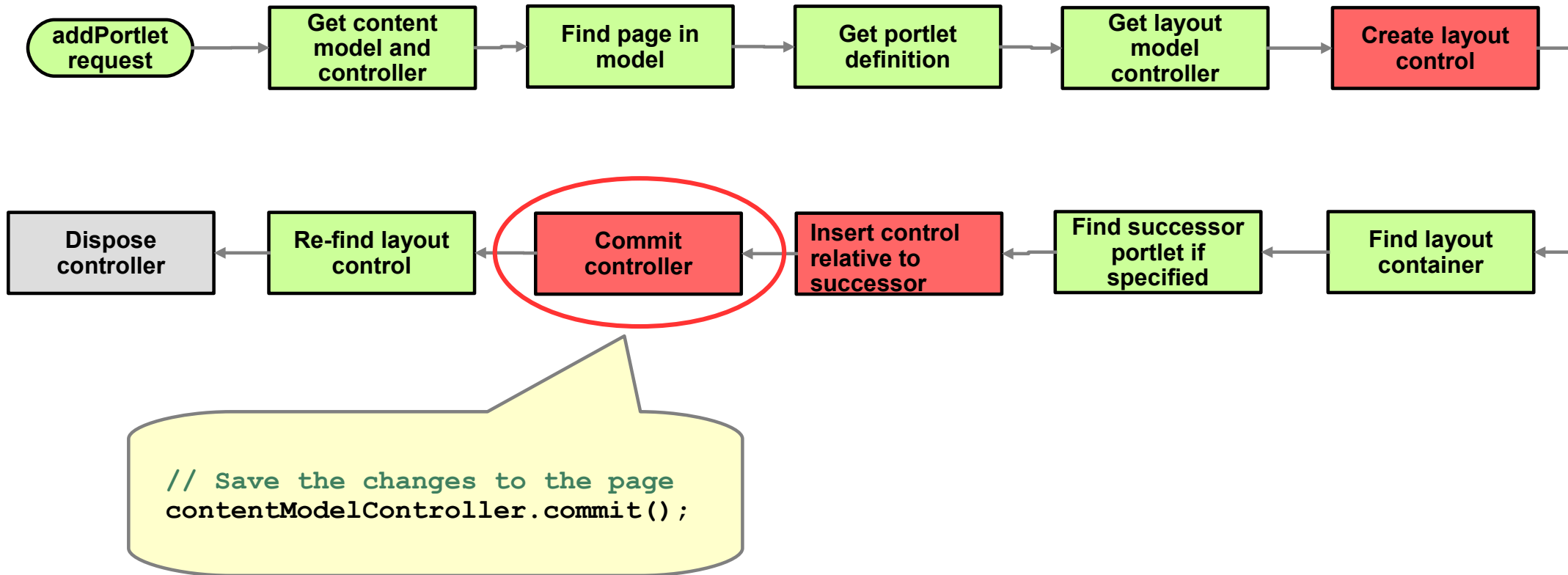


```
// Insert the new control into the layout model in the correct position  
// within the container
```

```
layoutModelController.insert(control, container, nextPortletInstance);
```

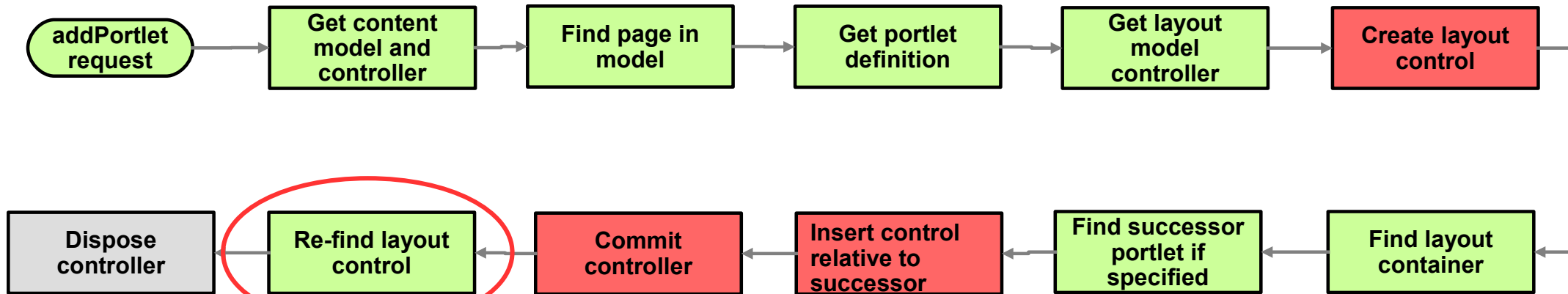
Adding a portlet to a page – *model update drill-down*

IBM Software



Adding a portlet to a page – *model update drill-down*

IBM Software

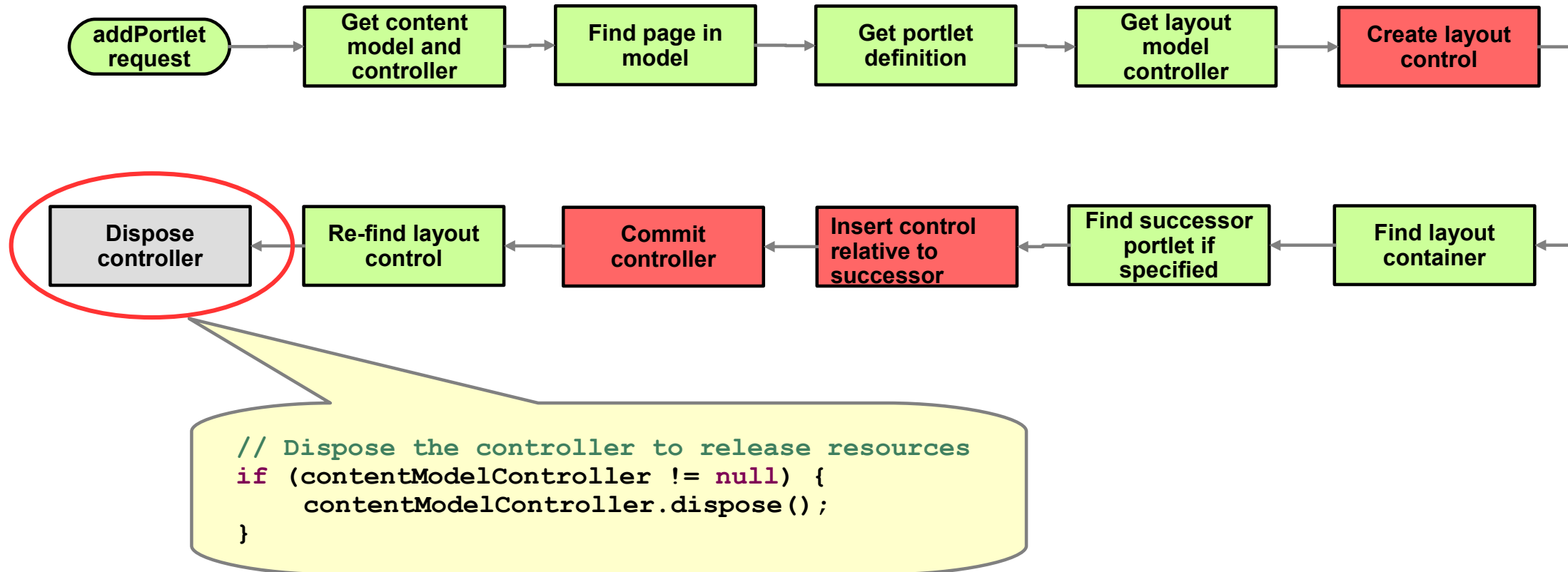


```
// Re-find the new portlet instance to get a valid ID
// Note that an error is generated converting the ID of a new
// portlet instance to text if you do not do this.
```

```
LayoutNode newNode = contentModel.getLayoutModel(currentPage)
                        .getLocator().findByID(control.getObjectID());
```

Adding a portlet to a page – *model update drill-down*

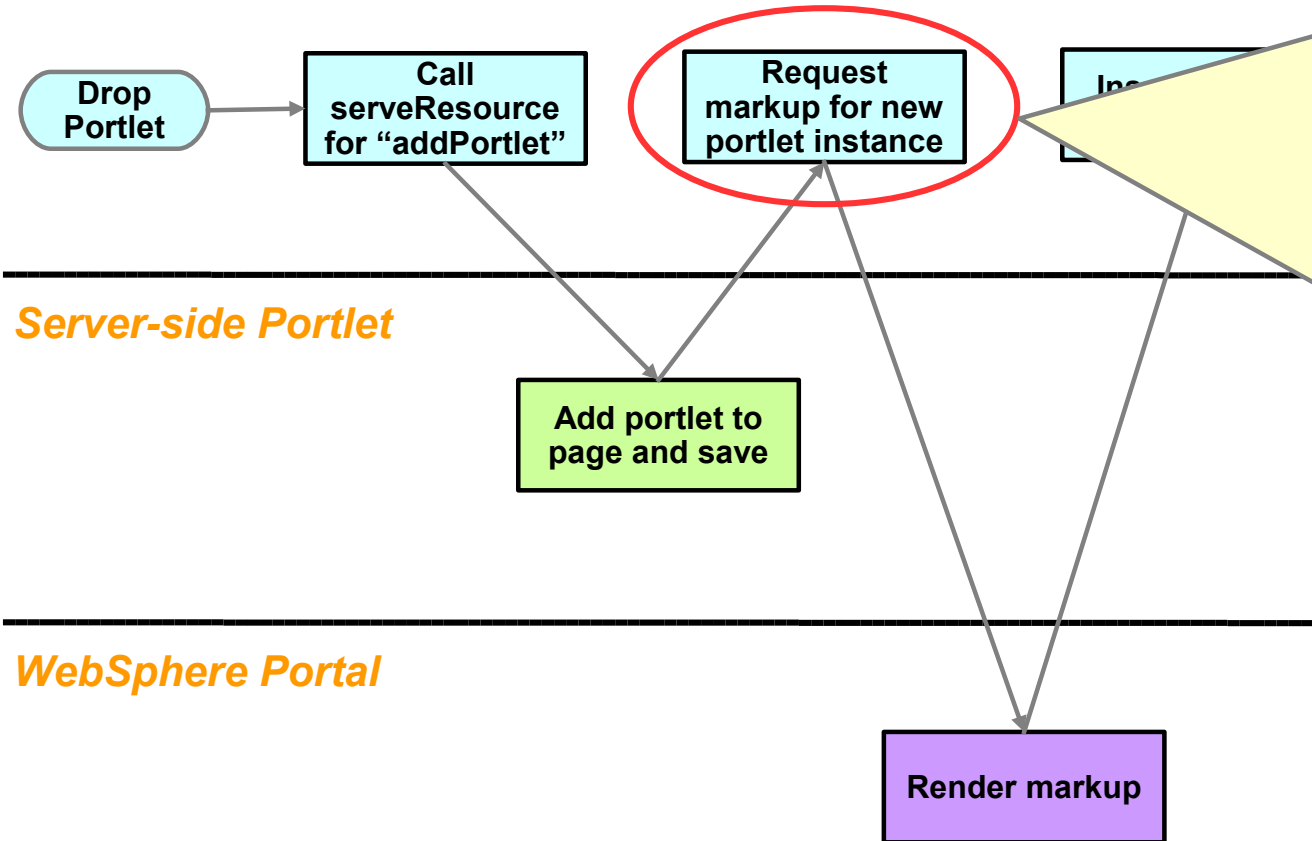
IBM Software



Adding a portlet to a page – *requesting portlet markup*

IBM Software

Client-side JavaScript

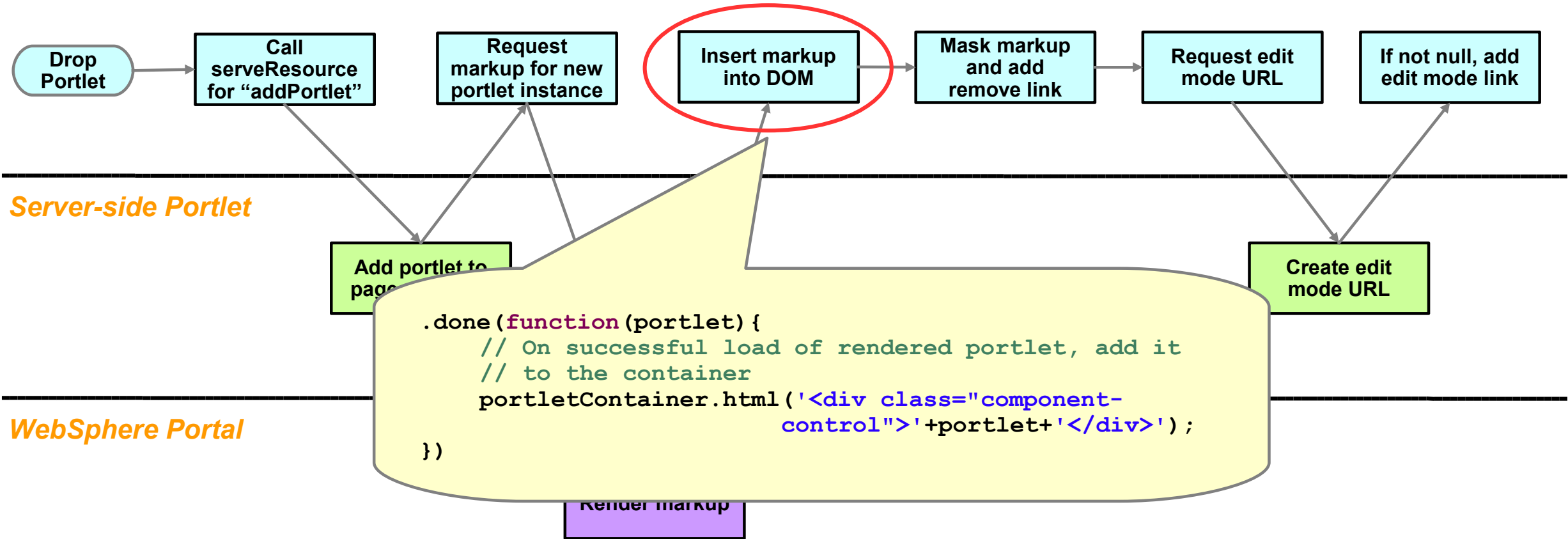


```
.done(function(jqxhr){
    // On successful addition, parse the
    // response to get access to the portlet
    // instance ID.
    var response =
        $.parseJSON(jqxhr.responseText||
                    jqxhr);
    // Update the ID of the portlet tag to
    // match the portlet instance ID in order
    // to support removal of portlets.
    portlet[0].id =
        response.portletInstanceId;
    // Make AJAX call to retrieve rendered
    // output of newly added portlet instance
    $.get('/wps/myportal?uri=lm:oid:' +
        response.portletInstanceId +
        '@oid:' + page.id)
        .done(function(portlet){
            ...
        })
    })
})
```

Adding a portlet to a page – *inserting the markup*

IBM Software

Client-side JavaScript



Adding a portlet to a page – *masking the portlet contents*

IBM Software

Client-side JavaScript

