Beyond the Portlet API

(More) Advanced Topics in WebSphere Portal Development

Graham Harper
Application Architect
IBM Software Services for WebSphere

Ideas behind this session

- Broaden the discussion when considering what sort of solutions are possible using WebSphere Portal
- Introduce some of the lesser-known / lesser-used development facilities of the product
- Do this through some specific examples of non-standard solutions

Agenda

- Introductions
- Portal development – building the jigsaw
- Filtering all portlets
- Using portlets from another page
- Styling the theme for the current user
- Extending personalization
- Questions and discussion

Introductions

Introductions - I'll go first...

- Worked in IBM Software Group since the acquisition of Lotus in 1995
- Developed solutions for customers on WebSphere Portal for approximately 15 years
- Used many facilities of the product in that time

Introductions – your turn

- So, who here has:
  - Developed JSR 286 portlets in RAD?
  - Created their own themes?
  - Used the Portal APIs / SPIs?
    - e.g . PUMA, Login Service, Credential Vault, Selection Model

Portal development

Building the jigsaw

Standard
Portlets

JSR 286 Portlets

- Standardised contract
- Java and JSPs
- RAD or WEF tooling
- Can add frameworks like JSF to improve productivity

Script Portlet instances

- HTML, CSS & JavaScript
- Managed as content

Portlets and Portal together create applications

- Portal aggregates portlets into pages
- Portlets work together via events and public render parameters

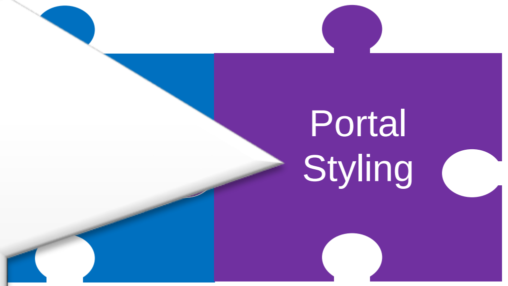
Standard Portlets
Portal Styling

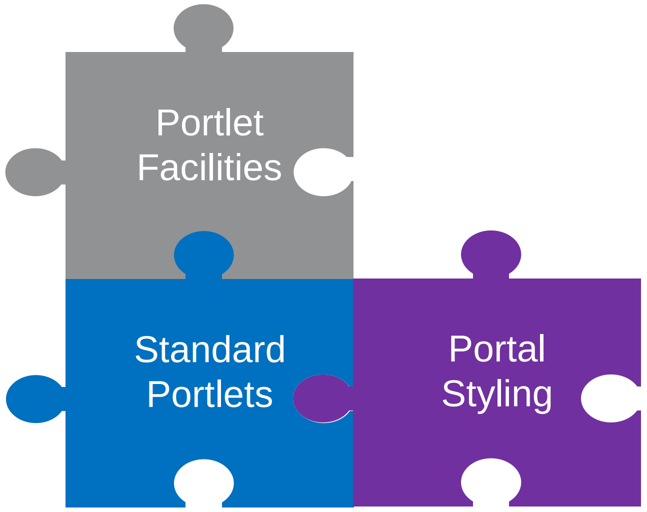
Themes, skins and layouts

- Overall structure of a page
- Consistent look and feel
- Organisation’s branding
- Multiple themes, profiles or palettes allow different looks for parts of the portal

JavaScript frameworks

- Provide rich client-side functionality
- Typically loaded by theme modules for efficiency – turned on and off by profiles
- Dojo used natively by Portal
- Easy to support others like jQuery, AngularJS and Bootstrap



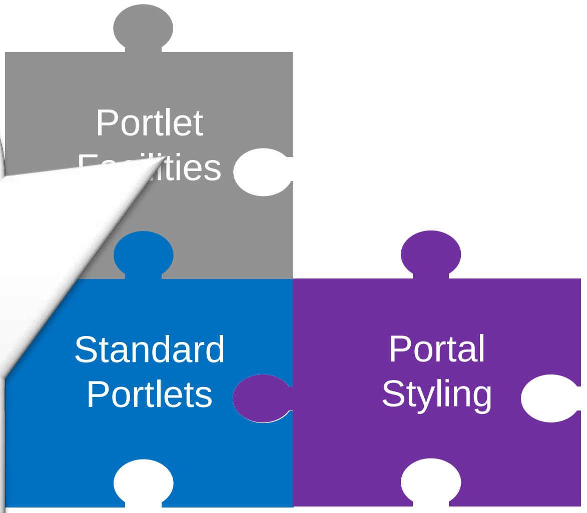

Portlet Facilities
Standard Portlets
Portal Styling

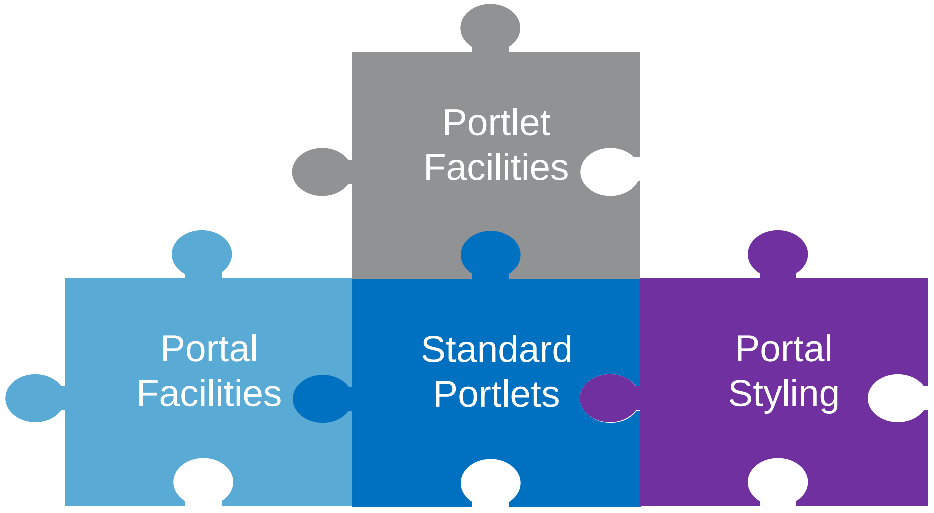
Advanced parts of JSR 286

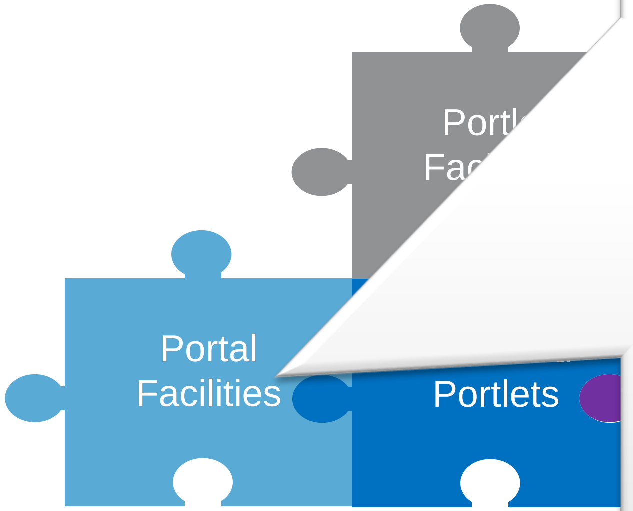
- Resource-serving for Ajax requests
- Complex event payloads with JAXB
- Portlet filters
- Injection of page headers

IBM-specific portlet services

- Credential Vault Service
- Ajax Proxy
- Portal User Management Architecture (PUMA) API




Portlet Facilities
Portal Facilities
Standard Portlets
Portal Styling



Web Content Management

- Integrate content with function
- Manage pages, script portlets etc. as content (via workflows)
- Integrate external data via Digital Data Connector (DDC)

Personalization

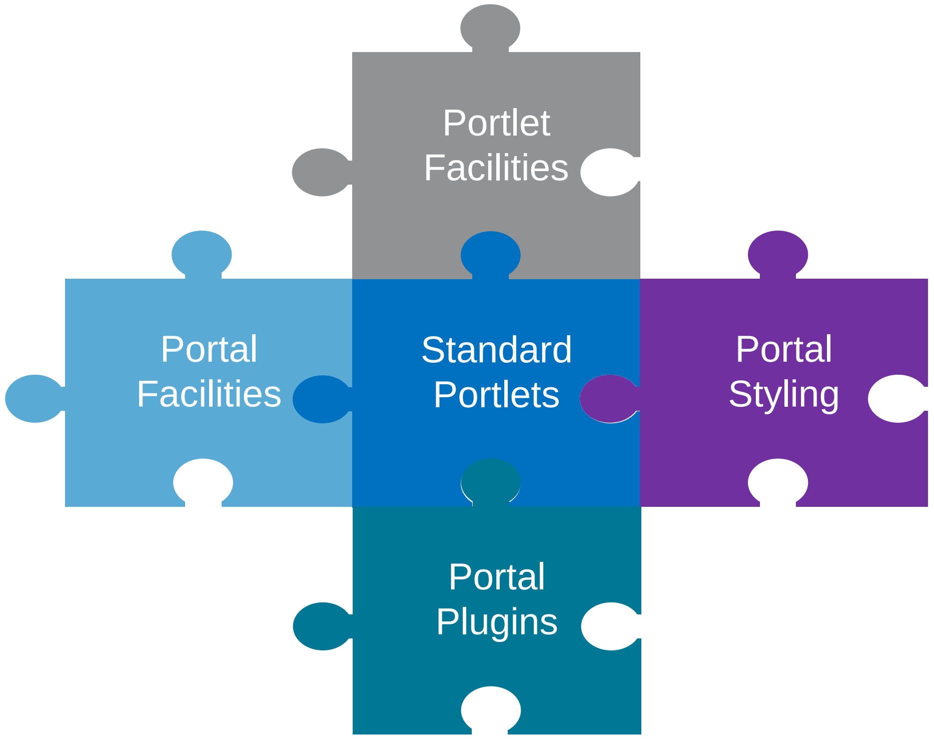
- Portlet and page visibility rules
- Targeted content
- Extend with application objects

UX Screen Flow Manager

- Control UI flow across pages and portlets

Portal APIs and SPIs

- Navigation & Selection models
- Dynamic UI
- Login
- Content Access
- Impersonation
- Tagging and Rating


Portlet Facilities
Portal Facilities
Standard Portlets
Portal Styling
Portal Plugins

Add Portal entry points

- Piece Of Content (POC) URI resolvers
- Data sinks

Enhance login and logout

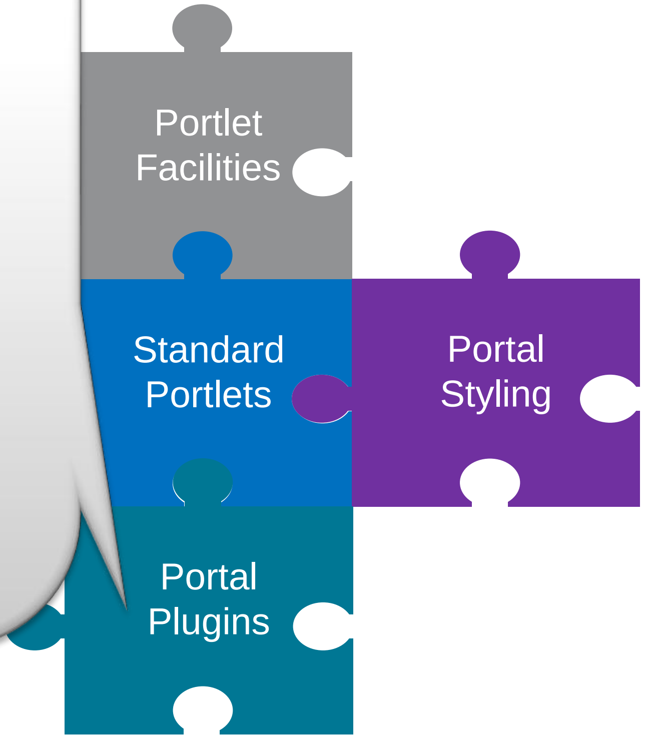
- Authentication Filters

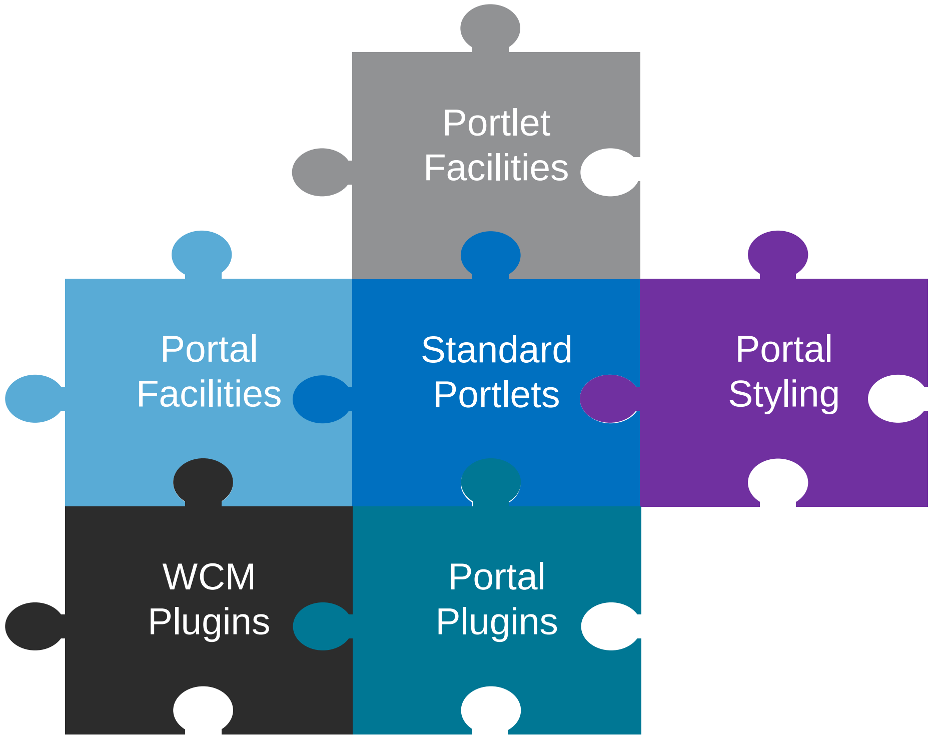
Intercept page transitions

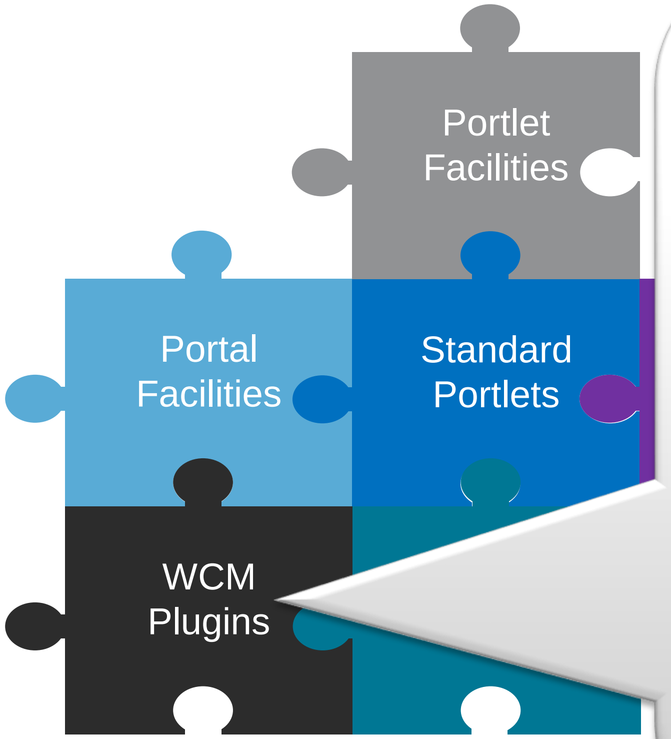
- State Preprocessors

Intercept portlet lifecycle

- Global portlet filters




Portlet Facilities
Portal Facilities
Standard Portlets
Portal Styling
WCM Plugins
Portal Plugins



Customise and extend the display of content

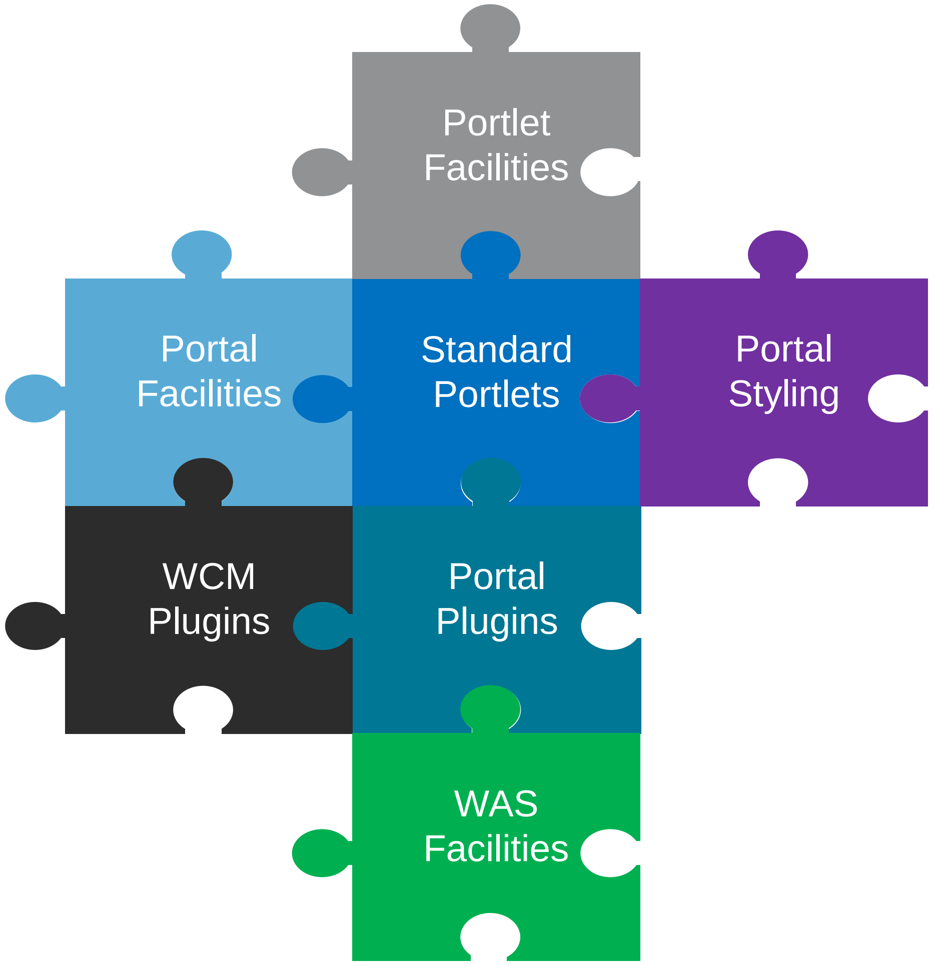
- New tags with Rendering Plugins
- Output control with custom JSPs

Take advantage of or extend content lifecycle

- Custom Workflow Actions

Control content displayed

- Context Processors
- Content Page Resolution Filters

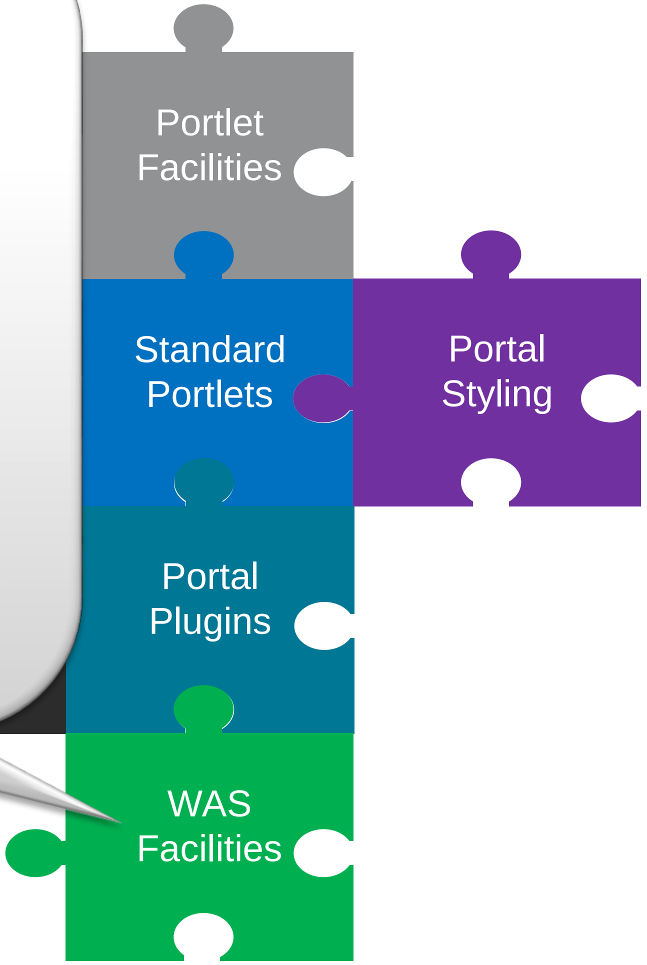

Portlet Facilities
Portal Facilities
Standard Portlets
Portal Styling
WCM Plugins
Portal Plugins
WAS Facilities

WAS facilities

- Standard web applications
- Schedulers
- Asynchronous Work Managers
- *Be careful of your licence agreement, however*

Security plugins

- Custom Repository Adapter
- Custom User Registry
- Trust Association Interceptor (TAI++)



Filtering all portlets

Business problem

- You need to intercept calls to and / or responses from all portlets in the portal

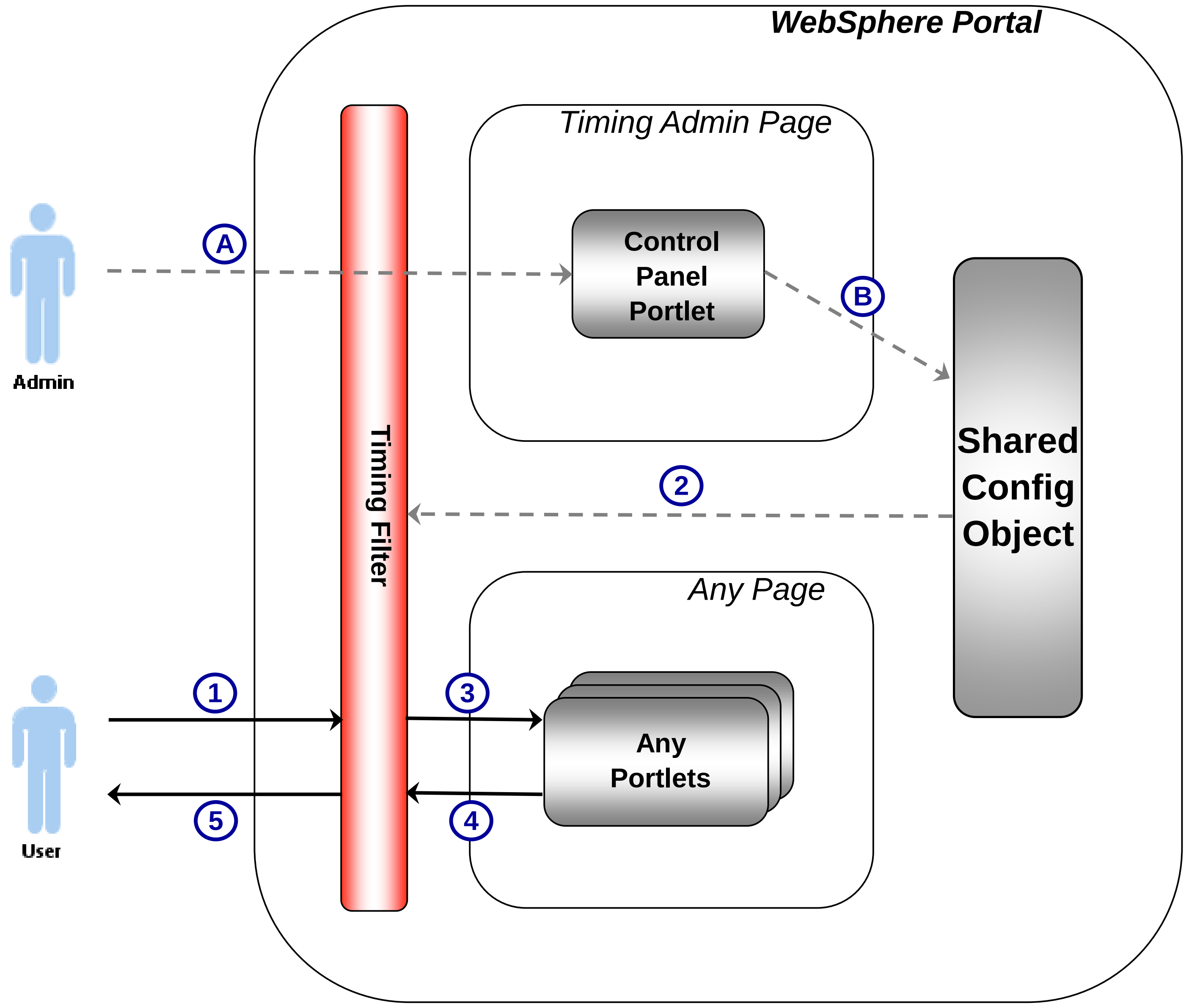
- For example you might need to:
  - Log all portlet invocations
  - Time how long it takes to invoke each portlet
  - Decorate what comes back from third-party portlets
  - Apply additional security checks

Featured solution

- Use a ***global portlet filter*** to:
  - Time all or selected portlet invocations
    - Including render, resource, action and event phases
  - Examine session usage and session attribute sizes for portlets
  - Log the results

- Use a portlet to allow the administrator to configure what the filter logs
  - Adds flexibility to reduce impact of timing and reduce clutter in the log

Solution flow



Solution flow

A. Administrator configures timing required via control panel portlet
B. Configuration saved into shared memory object

1. User requests page containing portlet(s)
2. Timing filter intercepts each portlet call and checks whether configured to time
3. If timing, filter records time before chaining call to portlet
4. If timing, then when portlet responds, filter logs duration of call and other required information
5. Portlet output is aggregated into page and returned to user

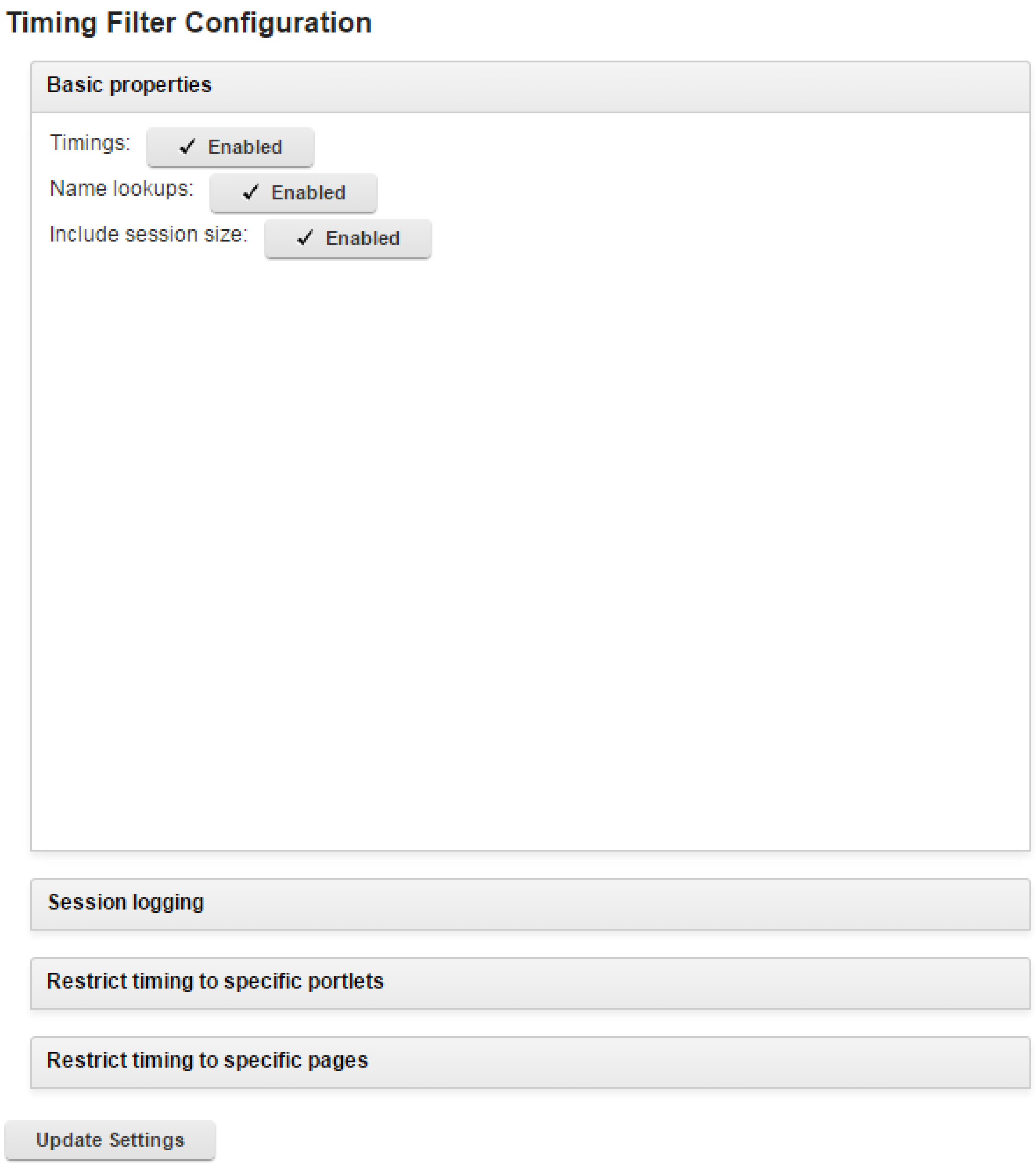
Demo

Log output for “Timing” page

```
[22/03/15 19:16:03:840 GMT] 00000146 PortletTiming I

TIMING: Portlet 'JS Clock' on page 'Timing'
(Z7_6O841KO0K89T10A62QAULK1082) RENDER phase took 47ms
```

Timing control panel (basic)



Timing Filter Configuration

Basic properties

Timings: ✓ Enabled

Name lookups: ✓ Enabled

Include session size: ✓ Enabled

Session logging

Restrict timing to specific portlets

Restrict timing to specific pages

Update Settings

Components of the example

- Global timing portlet filter
  - OSGi plug-in packaged in the same WAR as the portlet

- “Control panel” portlet added on a new page in the administration area of portal

Global filter plugin.xml

```xml
<plugin id="com.ibm.issc.portal.timing.filter"
  name="Portlet Timing Filter" provider-name="IBM" version="1.0.0">

    <extension point="com.ibm.ws.portletcontainer.portlet-filter-config">

        <portlet-filter-config
          class-name="com.ibm.issc.portal.timing.filter.PortletTimingFilter"
          order="99">

            <description>Portlet Timing Filter</description>
            <lifecycle>RENDER_PHASE</lifecycle>
            <lifecycle>RESOURCE_PHASE</lifecycle>
            <lifecycle>ACTION_PHASE</lifecycle>
            <lifecycle>EVENT_PHASE</lifecycle>
            …

        </portlet-filter-config>

    </extension>

</plugin>
```

Filter class

```
@Override
public void init(FilterConfig filterConfig) throws PortletException {
}

@Override
public void doFilter(RenderRequest request, RenderResponse response,
                FilterChain chain) throws IOException, PortletException {
    long millisBefore = System.currentTimeMillis();
    chain.doFilter(request, response);
    doTiming(RENDER_PHASE, request, response, millisBefore);
}

@Override
public void doFilter(EventRequest request, EventResponse response, …) {…}

@Override
public void doFilter(ActionRequest request, ActionResponse response, …) {…}

@Override
public void doFilter(ResourceRequest request, ResourceResponse response, …) {
    …
}
```

Deploying the filter

- Deploy and start the WAR file
  - In this case via the Portal administration area as we also have a portlet in it
  - For just a filter, use the WAS Integrated Solution Console

- Run an XmlAccess file
  - To create a page in the administration area containing the control panel portlet

Alternative solutions

- Non-global portlet filter
  - Need to register the filter in the portlet.xml of each portlet WAR
  - And make the code available in each WAR or a server library
  - Particularly undesirable in the case of third-party WARs

Possible future refinements for this solution

- Include alternative destinations for timing information
- Record timings below 1ms
- Set thresholds for alerts of slow performance
- Highlight badly performing portlets within the portal user interface (by manipulating portlet returned markup)

Other potential uses for global portlet filters

- Just log (rather than time) all portlet invocations

- Decorate what comes back from third-party portlets
  - Perhaps remove offensive language or injected malware

- Apply additional security checks
  - Enforce complex portlet entitlements

- Be cognisant of performance implications as this code will run for all portlets on all pages

Using portlets from another page

Business problem

- You need some functionality to appear on multiple pages, but not in the portlet area

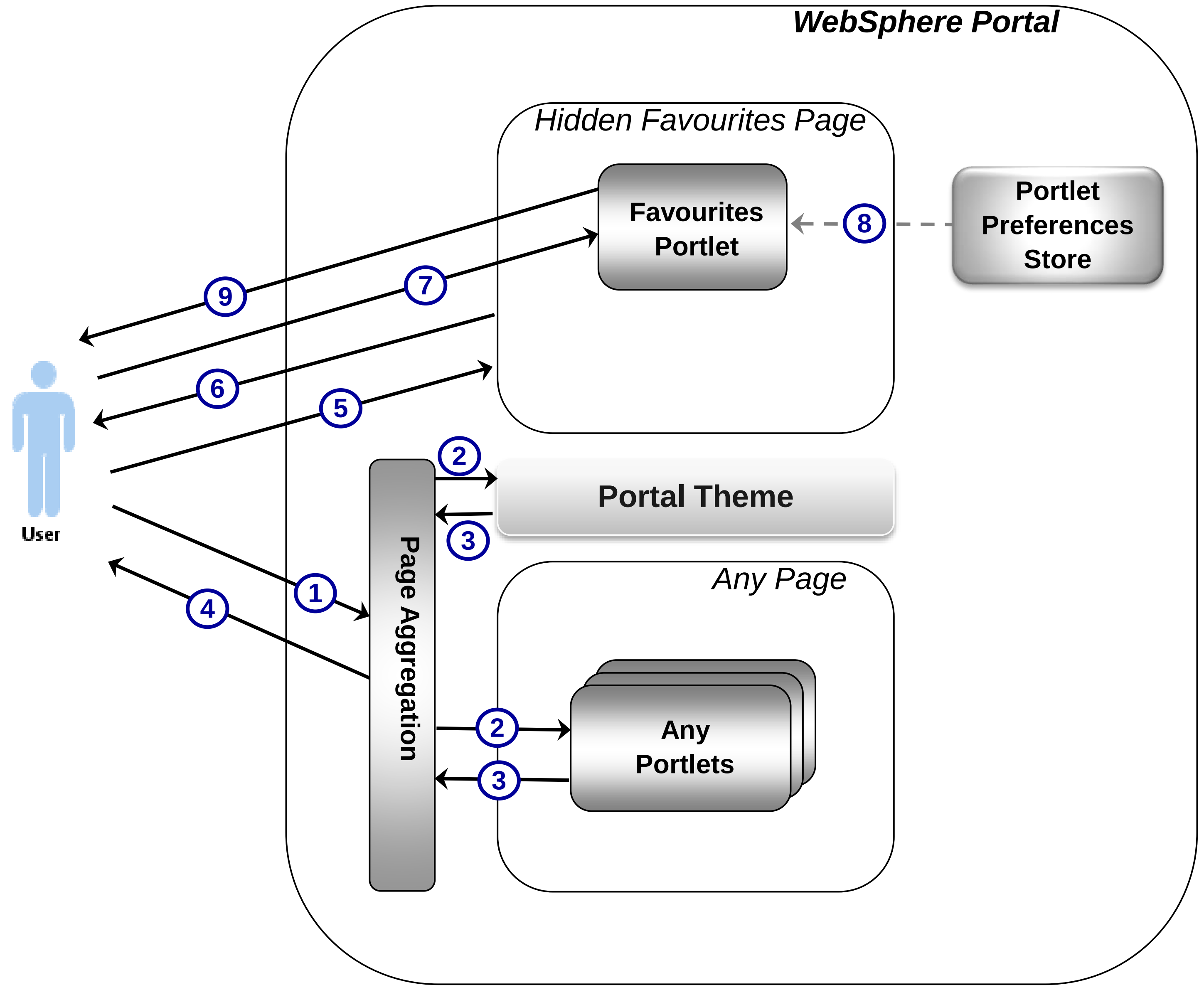
- For example, you might want:
  - Functionality in the header, footer or a sidebar
  - The ability to store per-user configuration for that functionality, with the configuration common across all pages

- Our specific example:
  - Simple “favourite pages” functionality in the banner of multiple pages
  - Favourites are stored per user, but the same list should appear on all pages for that user

Featured solution

- A portlet instance on a hidden page provides the functionality to:
  - Display a list of favourites and navigate to the one selected
  - Add the current page to the list
  - Clear the list
- The portlet’s preferences are used to store the favourites
- The theme includes the portlet by making an Ajax call to a portal rendering URL for the hidden page

Solution (render) flow



Solution (render) flow

1. User requests page containing some portlet(s)
2. Portal calls each portlet, plus the theme to render
3. Portlets and theme return their own markup fragments
4. Portal aggregation engine assembles responses into a single page and returns it to the user
5. Client-side code included in the theme makes Ajax request to Favourites Portlet on hidden page
6. Portal returns Favourites Portlet markup which is incorporated into banner area of page
7. Client-side code in Favourites Portlet markup makes Ajax request to portlet’s “serveResource()” method
8. Portlet retrieves current favourites for user from preferences store
9. Portlet returns favourites as JSON, which are then used to populate drop-down

Demo

Log output for “Page 2” page

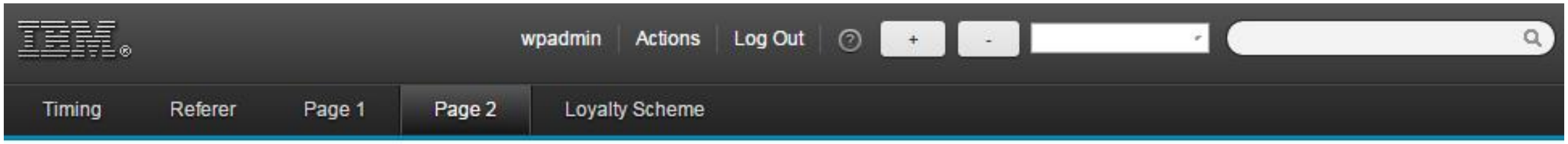
```
[22/03/15 19:22:01:299 GMT] 00000148 PortletTiming I   TIMING: Portlet 'JS
Clock' on page 'Page 2' (Z7_6O841KO0KGFQ60A6K7OTKA3067) RENDER phase took
16ms

[22/03/15 19:22:01:314 GMT] 00000148 PortletTiming I   TIMING: Portlet 'JS
Clock' on page 'Page 2' (Z7_6O841KO0KGFQ60A6K7OTKA30M4) RENDER phase took
15ms

[22/03/15 19:22:02:126 GMT] 00000148 PortletTiming I   TIMING: Portlet
'Favourite Pages' on page 'Favourites' (Z7_6O841KO0KGNG80A6K52FBB20O6) RENDER
phase took 0ms

[22/03/15 19:22:02:172 GMT] 00000148 PortletTiming I   TIMING: Portlet
'Favourite Pages' on page 'Favourites' (Z7_6O841KO0KGNG80A6K52FBB20O6)
RESOURCE phase took 0ms
```

Favourites portlet included in theme banner



Components of the example

- A Favourite Pages portlet:
  - Displays a list of favourites in a drop-down and navigates to the one selected
  - Has buttons to add the current page to the list and to clear the list
  - Portlet preferences used to store favourites list for the user
  - Buttons use Ajax to make changes to the portlet preferences and then refresh the options in the drop-down

- A single instance of the portlet exists on one hidden page:
  - As there is only one instance, then there is only one set of portlet preferences per user: so the same favourites list is shown on every page

- The theme includes the portlet by making an Ajax call to a portal rendering URL for the hidden page
  - The portlet markup can appear in theme-controlled areas of the page, such as the banner

Rendering URL for a specific portlet instance

- Format:

```
/wps/myportal?uri=lm:oid:<portlet_container_id>@oid:<page_id>
```

- Can get the IDs from an XmlAccess export
  - Or via the model APIs dynamically if you think they may change

commonActions.jsp in the theme (complete code)

```
<script>
     require([
         'dojo/domReady!'
     ], function () {
          var insertWithEval = function(target, text) {
               dojo.place(text, target, "only");
               dojo.query("script", target).forEach(function(scriptElement) {
                  var theScript = (scriptElement.text || scriptElement.textContent ||
                  scriptElement.innerHTML || "");
                  eval(theScript);
               });
          };
          var portletUrl =
             '/wps/myportal?uri=lm:oid:Z7_6O841KO0KGNG80A6K52FBB20O6@oid:Z6_6O841KO0KGNG80A6K52FBB20G5';
          var sendRequest = function (method, requestUrl) {
             var xhrArgs = {
                     url: requestUrl,
                load: function(data) {
                     if (data) {
                               insertWithEval('favouritesPortletContainer', data);
                     }
                },
                error: function(error) {
                      alert("An error occurred contacting the server: " + error);
                }
             };
             dojo.xhr(method, xhrArgs);
          };

          sendRequest("GET", portletUrl);
     });
</script>
<div id="favouritesPortletContainer"></div>
```

Deploying the solution

- Deploy the Favourite Pages portlet in a WAR through the Portal administration area
- Create a (hidden) page and add the portlet to it
- Set permissions on the page and portlet
- Export the page using XmlAccess to get the OIDs of the page and portlet container
- Update the theme with the IDs and deploy the theme as an EAR / WAR
- Set an appropriate profile on pages that will include the portlet in the theme
  - The example code requires a profile providing Dojo

Alternative solutions

- Add the portlet to each page
  - The portlet could only appear in the “portlet area” of the page
  - The list of favourites would be on a per-page basis

- Code the functionality into the theme
  - Less maintainable and reusable
  - Where would we store the favourites for each user?

Possible future refinements for this solution

- Perhaps we could add a dialog to allow the user to add external URLs to the favourites list?

Other uses for using portlets from another page

- A sidebar on every page
- A stock ticker in the header or footer

Styling the theme for the current user

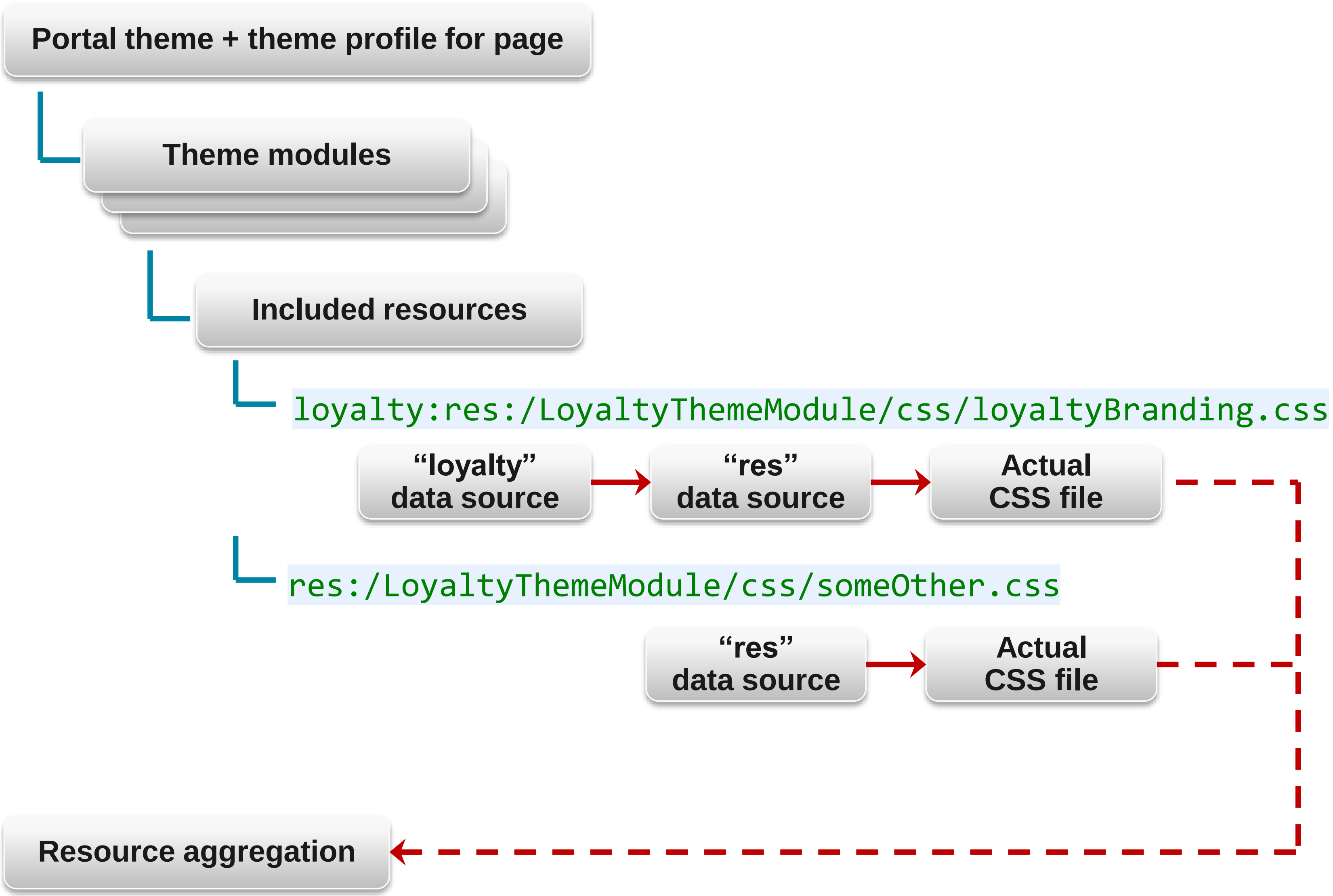
Business problem

- You need to style portal pages differently for different users
- Specifically, you need the page “branding” to be different, not the portlets or content displayed
- In our example, a loyalty scheme requires different colours in the banner for users at different “levels” in the scheme
  - Navigation tabs should be blue for most users
  - Tabs should be gold for users at that loyalty level

Featured solution

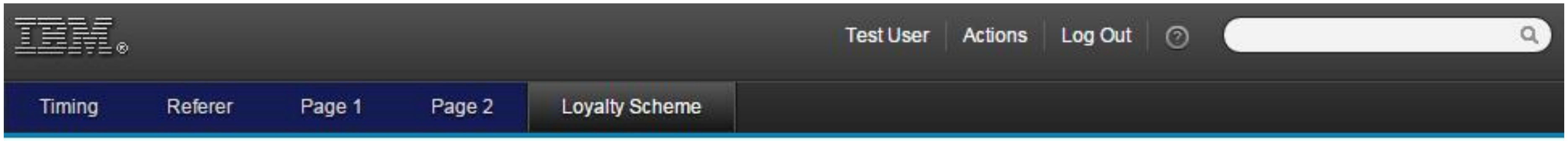
- Create a new “data source” plugin that registers a new URI scheme for including level-related CSS files
  - The data source updates CSS file paths dynamically based on the loyalty level of the current user
- Create a new theme module to add such a CSS file to the theme
- User loyalty level is stored as user attribute in their profile

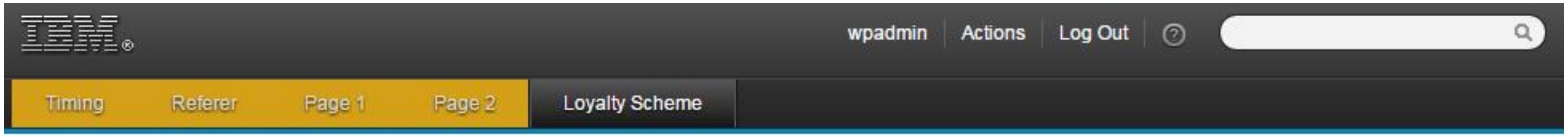
Solution flow



Demo

Different branding per user





Components of the example

- Loyalty data source
  - Processes URIs that have a “loyalty:” scheme
  - Looks up user’s loyalty level from a profile attribute via PUMA API
  - If value is “gold”, rewrites any CSS URI to load “_gold.css” suffixed version
  - Delegates remainder of URI to be processed by other data sources (important!)
- Loyalty theme module
  - Just loads a single CSS file that brands the navigation tab backgrounds
- Theme profile used on “Loyalty Scheme” page has new module added

Data sources use a factory pattern

- Register a factory in the plugin.xml
- Factory creates the actual data source
- In our example, most of the work is done in the factory

Data source plugin.xml

```
<plugin  id="com.ibm.issw.example.loyalty.plugin"
         name="Loyalty Branding Data Source Plugin"
         version="1.0.0"
         provider-name="IBM">

    <extension point="com.ibm.content.operations.registry.locationTypeContribution">
       <contentLocationType
           class="com.ibm.portal.resolver.helper.cor.DefaultContentLocationFactory"
           id="com.ibm.issw.example.loyalty.selector"
           match.uri.scheme="loyalty"
           title="Selects resources based on current user's loyalty level" />
    </extension>

    <extension point="com.ibm.content.operations.registry.locationServiceHandler">
       <serviceHandler
           class="com.ibm.issw.example.loyalty.datasource.LoyaltyDataSourceFactoryImpl"
           locationTypeId="com.ibm.issw.example.loyalty.selector"
           id="com.ibm.portal.resolver.data.DataSourceFactoryEx" />
    </extension>

</plugin>
```

Factory updates the scheme-specific part of the URI

```
@Override
public DataSource newSource(URI uri, String mode,
            Map<String, String[]> params, Context ctx) throws IOException {

    String schemeSpecificPart = uri.getSchemeSpecificPart();

    // Extract the user's loyalty level from their profile
    String loyaltyLevel;
    try {
        loyaltyLevel = getProfileAttribute(USER_ATTRIBUTE_LOYALTY_LEVEL);
    } catch (UserProfileException upe) {
        loyaltyLevel = null;
    }

    // Update the URL to use loyalty-level-specific CSS files, if appropriate
    schemeSpecificPart = updateRemainingURI(schemeSpecificPart, loyaltyLevel);

    …
```

Delegates the remainder of the URI to be parsed

```
// Delegate processing of the remaining URI to other data source(s)
try {
    URI delegateURI = new URI(schemeSpecificPart);
    InitialContext jndiCtx = new InitialContext();
    CorPocServiceHome corPocServiceHome = (CorPocServiceHome)
                        jndiCtx.lookup(CorPocServiceHome.JNDI_NAME);
    final DataSourceFactoryEx fac =
                            corPocServiceHome.getDataSourceFactory(false);

    final DataSource origDS = fac.newSource(delegateURI, mode, params, ctx);
    final CharDataSource delegateDs =
                    corPocServiceHome.convert(CharDataSource.class, origDS);

    return new LoyaltyDataSource(uri, mode, params, ctx, delegateDs);
} catch (URISyntaxException e) {
    …
} catch (NamingException e) {
    …
}
```

Actual data source is very simple

```
public LoyaltyDataSource(URI uri, String mode, Map<String, String[]> params,
                        Context ctx, CharDataSource delegateDs) {
    this.uri = uri;
    this.params = params;
    this.created = new Date();
    this.delegateDs = delegateDs;
}


@Override
public Writer write(Writer writer) throws IOException {
    delegateDs.write(writer);
    return writer;
}
```

Theme module plugin.xml

```xml
<plugin
    id="com.ibm.issw.example.loyalty.module.main"
    name="Loyalty artefacts module"
    version="1.0.0"
    provider-name="IBM">

    <extension point="com.ibm.portal.resourceaggregator.module"
               id="loyalty_main">
        <module id="loyalty" version="1.0.0">
            <prereq id="wp_portal"/>
            <capability id="loyalty" value="1.0.0"/>
            <contribution type="head">
                <sub-contribution type="css">
                <uri value="loyalty:res:/LoyaltyThemeModule
                            /css/loyaltyBranding.css"/>
                </sub-contribution>
            </contribution>
        </module>
    </extension>

</plugin>
```

Addition to theme profile JSON file

```
{
"moduleIDs": [
"getting_started_module",
"wp_theme_portal_85",
"testTheme_dynamicContentSpots_85",
"wp_toolbar_host_view",
"wp_portlet_css",
"wp_one_ui",
"wp_one_ui_dijit",
"wp_client_ext",
"wp_status_bar",
"wp_theme_menus",
"wp_theme_skin_region",
"wp_theme_high_contrast",
"wp_layout_windowstates",
"wp_portal",
"wp_analytics_aggregator",
"wp_oob_sample_styles",
"wp_ic4_wai_resources",
"wp_worklight_ext",
"wp_social_rendering_85",
"wp_sametime_proxy",
"loyalty"
],
"deferredModuleIDs": [
…
```

Deploying the solution

- Deploy and start the Loyalty Data Source as a WAR using the WAS Integrated Solution Console

- Deploy and start the Loyalty Theme Module as a WAR using the WAS Integrated Solution Console

- Update the JSON file(s) for the profile(s) that will use the theme module
  - Depending upon how your theme was developed, either redeploy the WAR / EAR or upload the file using WebDAV

Alternative solutions

- Create separate set of pages for each loyalty level
  - Configure each set to have different theme, profile or palette
  - Reasonable if functionality is also different for each loyalty level
  - Maintenance overhead if the only difference is branding
  - Virtual portals per level have similar pros and cons

- Add code to theme to dynamically add /remove links to CSS files based on loyalty level
  - Linked files will not be included in resource aggregation, so will cause additional HTTP requests to the server

Other uses for styling based on user

- Branding / white-labelling according to the business brand / subsidiary with which the user has a relationship

- Branding according to the domain via which the user is accessing the portal

Extending personalization

Business problem

- You need to personalize what the user sees based on data not currently available to the normal mechanisms determining visibility (e.g. security, visibility rules)

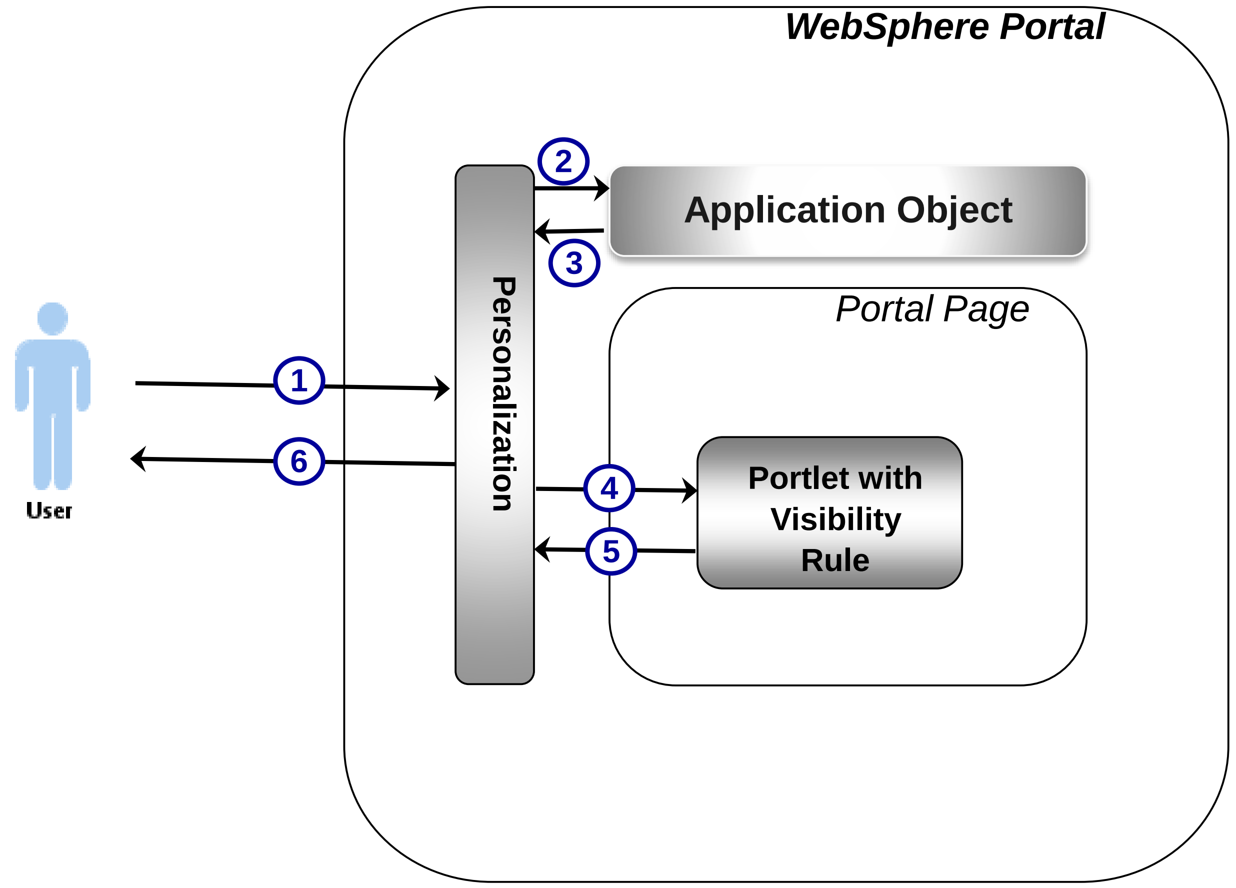
- For example, you want to base what the user sees on the value of a cookie

- In our specific example, we assume that a cookie has been set if the user arrives via a referral from another site
  - We would like to show additional content in this case

Featured solution

- Extend the data available to the personalization engine to include the cookie value, through the creation of an application object
- Create a business rule that uses the cookie value
- Apply the rule as a visibility rule on a portlet instance, so that it is only shown to referred users

Solution flow

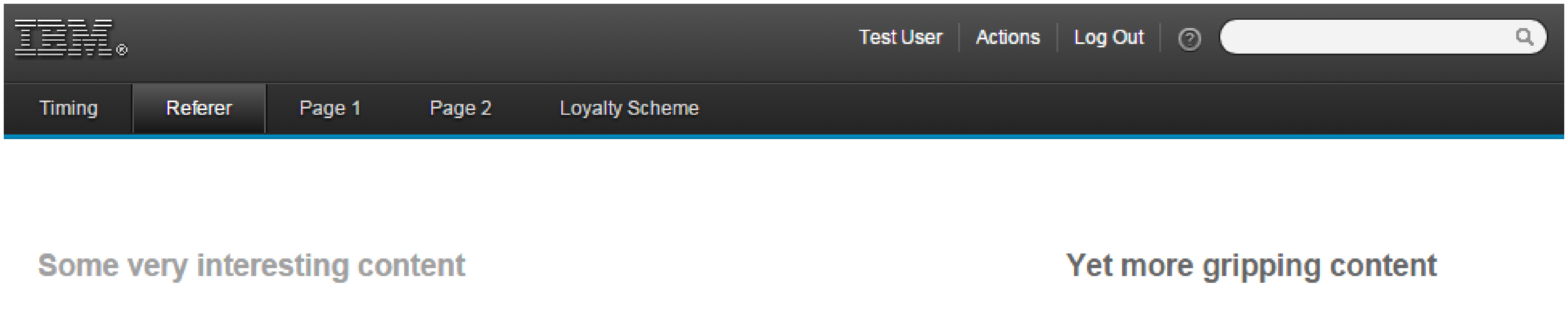


Solution flow

1. Authenticated user requests portal page
2. Personalization sees a portlet has a visibility rule and that rule references an application object, so instantiates the application object and asks it for the value
3. Application object retrieves the cookie value from the request and returns this to personalization
4. Personalization evaluates the rule and determines whether the portlet should be rendered; if so, the portlet is called
5. The portlet returns its markup, if asked
6. The aggregated page is returned to the user, with or without the portlet, depending upon the rule evaluation

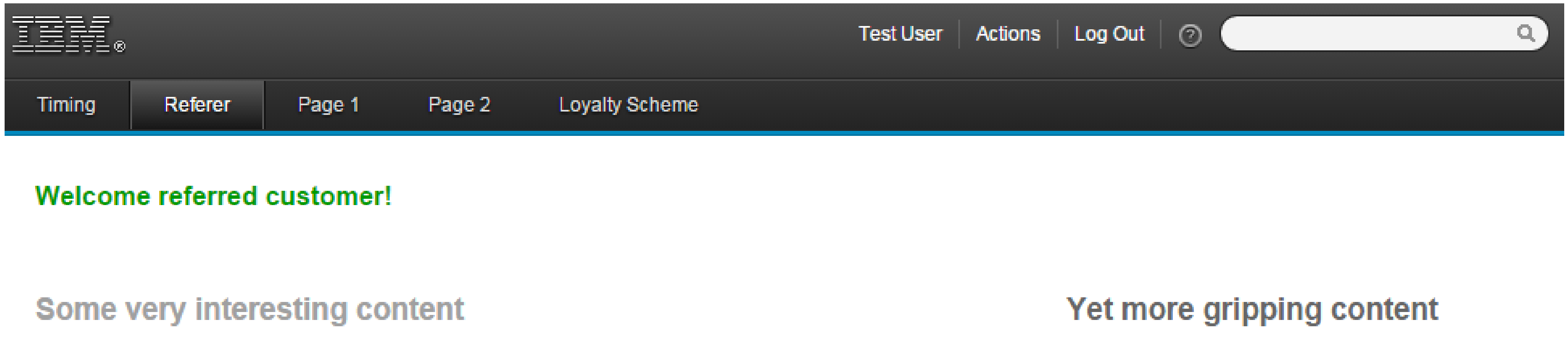
Demo

Referred users see an additional script portlet



Some very interesting content

Yet more gripping content



Welcome referred customer!

Some very interesting content

Yet more gripping content

Components of the example

- Application object to expose the cookie value
- Visibility rule to use the application object / cookie value
- Portlet instance on the “Referer” page which depends on the rule
- A browser plug-in to change the cookie value for testing!

Creating the application object

```
public class RefererCookieAO implements SelfInitializingApplicationObject {

    public static final String COOKIE_NAME = "refererCookie";
    private String value;

    @Override
    public void init(RequestContext context) {
        Cookie theCookie = context.getCookie(COOKIE_NAME);
        if (theCookie == null) {
            this.value = "";
        } else {
            this.value = theCookie.getValue();
        }
    }

    public String getValue() {
        return value;
    }

    public void setValue(String value) {
        this.value = value;
    }
}
```

Deploying the application object

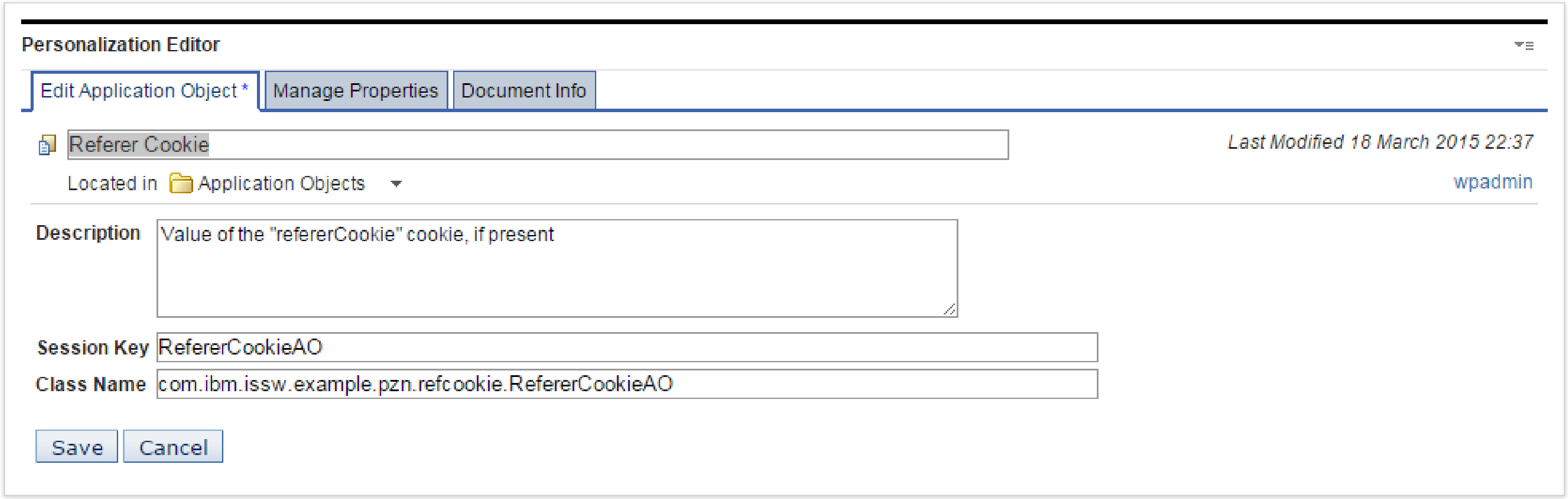
- Create a JAR file containing the class
- Deploy the JAR to:

```
        <portal server root>/pzn/prereq.pzn/collections
```

- Restart the server

Registering the application object

- After deployment you need to make personalization aware of the new object, e.g.
  - Go to “Applications -> Personalization -> Business Rules”
  - Select “New -> Application Object”
  - Enter a name, the fully-qualified class and a session key *(covered later)*



Personalization Editor

Edit Application Object * | Manage Properties | Document Info

Referer Cookie

Last Modified 18 March 2015 22:37

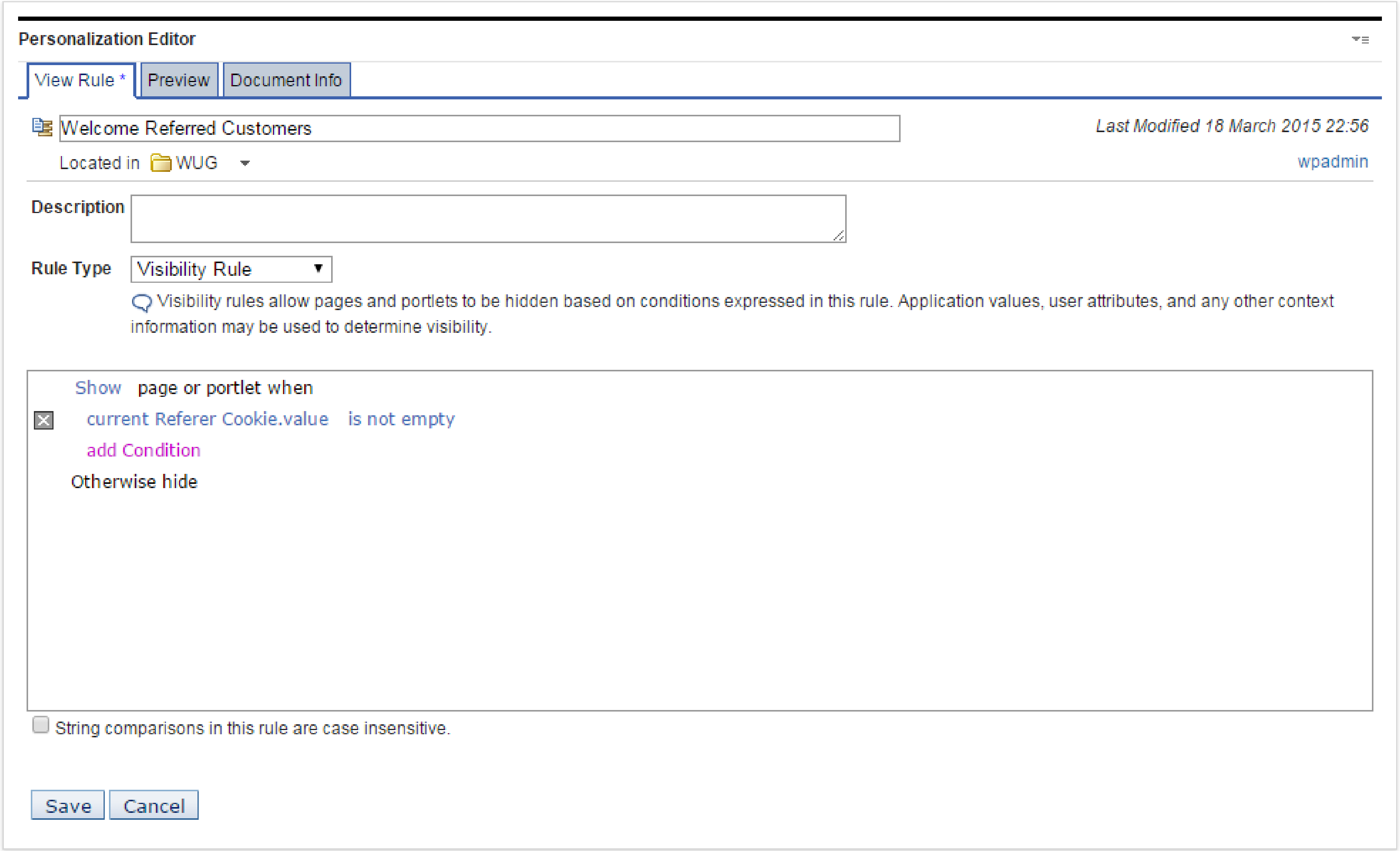
Located in Application Objects

wpadmin

Description	Value of the "refererCookie" cookie, if present
Session Key	RefererCookieAO
Class Name	com.ibm.issw.example.pzn.refcookie.RefererCookieAO

Save | Cancel

Creating a visibility rule



Personalization Editor

View Rule * | Preview | Document Info

Welcome Referred Customers

Last Modified 18 March 2015 22:56

Located in WUG

wpadmin

Description

Rule Type Visibility Rule

Visibility rules allow pages and portlets to be hidden based on conditions expressed in this rule. Application values, user attributes, and any other context information may be used to determine visibility.

Show page or portlet when
current Referer Cookie.value is not empty
add Condition
Otherwise hide

☐ String comparisons in this rule are case insensitive.

Save | Cancel

Caching considerations

- By default, a new instance of the application object is created whenever personalization needs the value
- To avoid this, you can put the following at the end of the "init()" method:

```
        context.setSessionAttribute("RefererCookieAO", this);
```

- Note that the name here is the same as that entered when registering the application object
- Unfortunately, only the "init()" method of the application object (and not the getters) receives a request context
  - So if this is needed to get the value (as it is in our example), then it will not be possible to get an updated value during the same session if you cache

Alternative solutions

- Put the data somewhere that personalization can already access
  - E.g. in the **portal** session
  - **BUT:**
    - Need to get the data in there somehow
    - May result in session “bloat”

- Determine portlet visibility programmatically, such as by:
  - Having one portlet with alternate views
  - Putting logic in portal skins
  - Using a global portlet filter
  - **BUT:**
    - More complex
    - Harder to maintain
    - Does not exploit the power of the rules engine

Other uses for extending personalization

- Any time you have information from additional sources that you'd like to use in rules

- However, note that visibility rules can only be used to change what is shown to ***authenticated*** users

Questions and discussion