

Meet the Lab

June 4th – 5th 2014

IBM Lab Böblingen



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.

URLs in WebSphere Portal

Dr. Carsten Leue
Portal Architect



Agenda

- URL Concepts in WebSphere Portal
 - Use cases
 - Navigational State
 - Rich URLs
- Types of URLs
 - Rich URLs
 - Friendly URLs
 - Vanity URLs
 - POC URLs
- Tips and Tricks for working with URLs

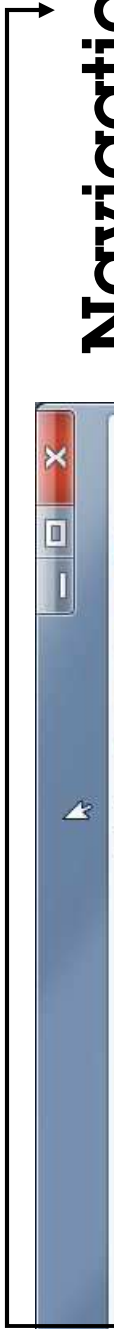


REST Programming Model



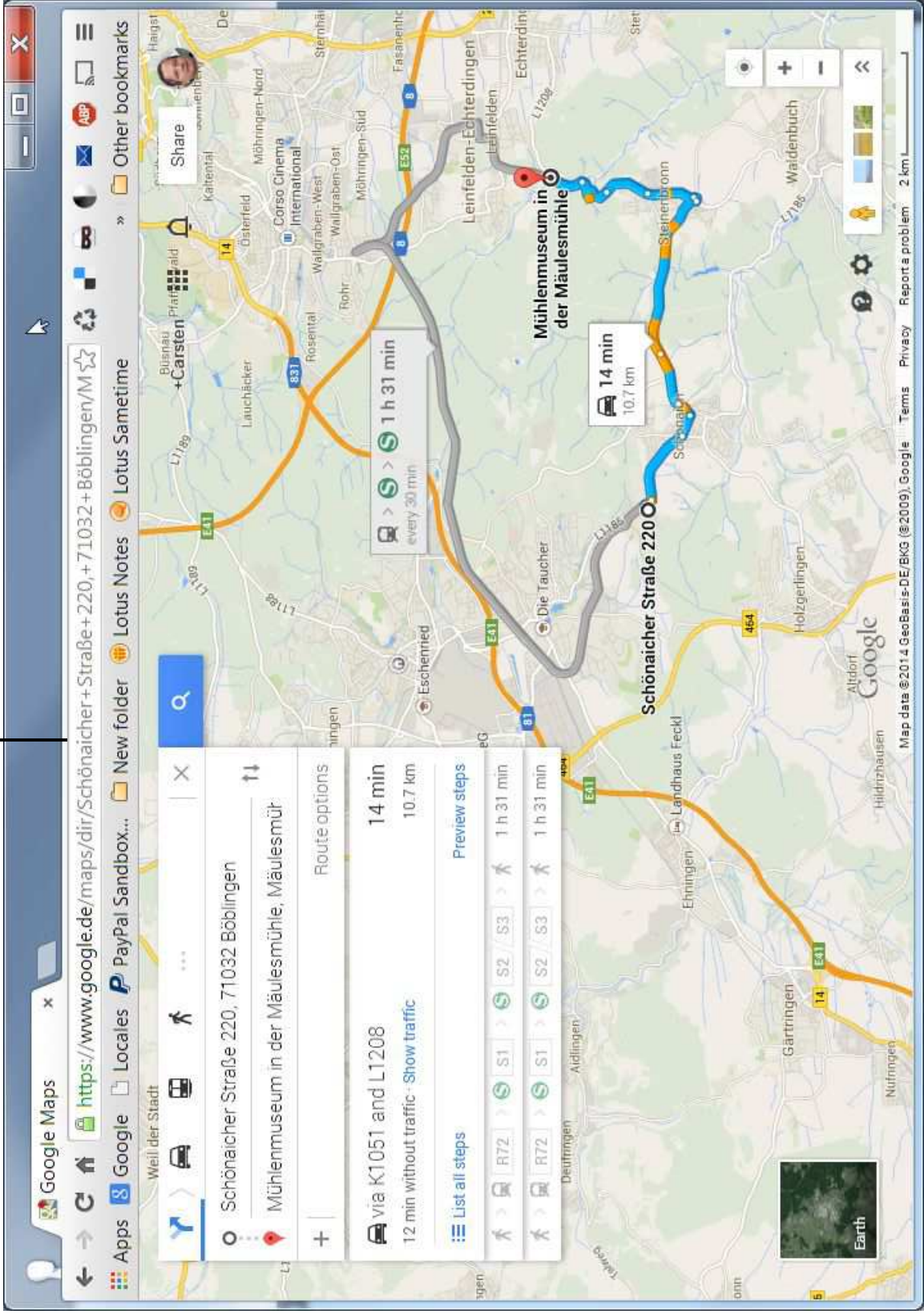
- REST is the foundation of the Web
 - **Resources are uniquely identified via URIs**
 - HTTP GET represents idempotent operations (*without side effects, aka "safe requests"*)
 - HTTP POST/PUT/DELETE represent modifications (*aka "unsafe requests"*)
- Why should I follow the REST pattern?
 - Performance: HTTP caching infrastructure assumes REST
 - Functionality: consistent behaviour of the back-button, full bookmarkability and crawlability
- WebSphere Portal makes it easy to follow the REST pattern
 - Portlet API is REST based

Example: Google Maps



Navigational State

<https://www.google.de/maps/dir/Schönaicher+Straße+220,+71032+Böblingen/M%C3%BChlenmuseum+in+der+M%C3%A4ulesm%C3%BChle,+70771+Leinfelden/@48.6878865,9.0005538,12z/data=!3m1!4b1!4m1!3!4m1!2!1m5!1m1!1s0x4799e0ca7199ee5d:0xb3f740a3990b05e9!2m2!1d9.0392321!2d48.6662016!1m5!1m1!1s0x4799e8088a181e0b:0xf69e09d2dab89fc2!2m2!1d9.128265!2d48.683495>

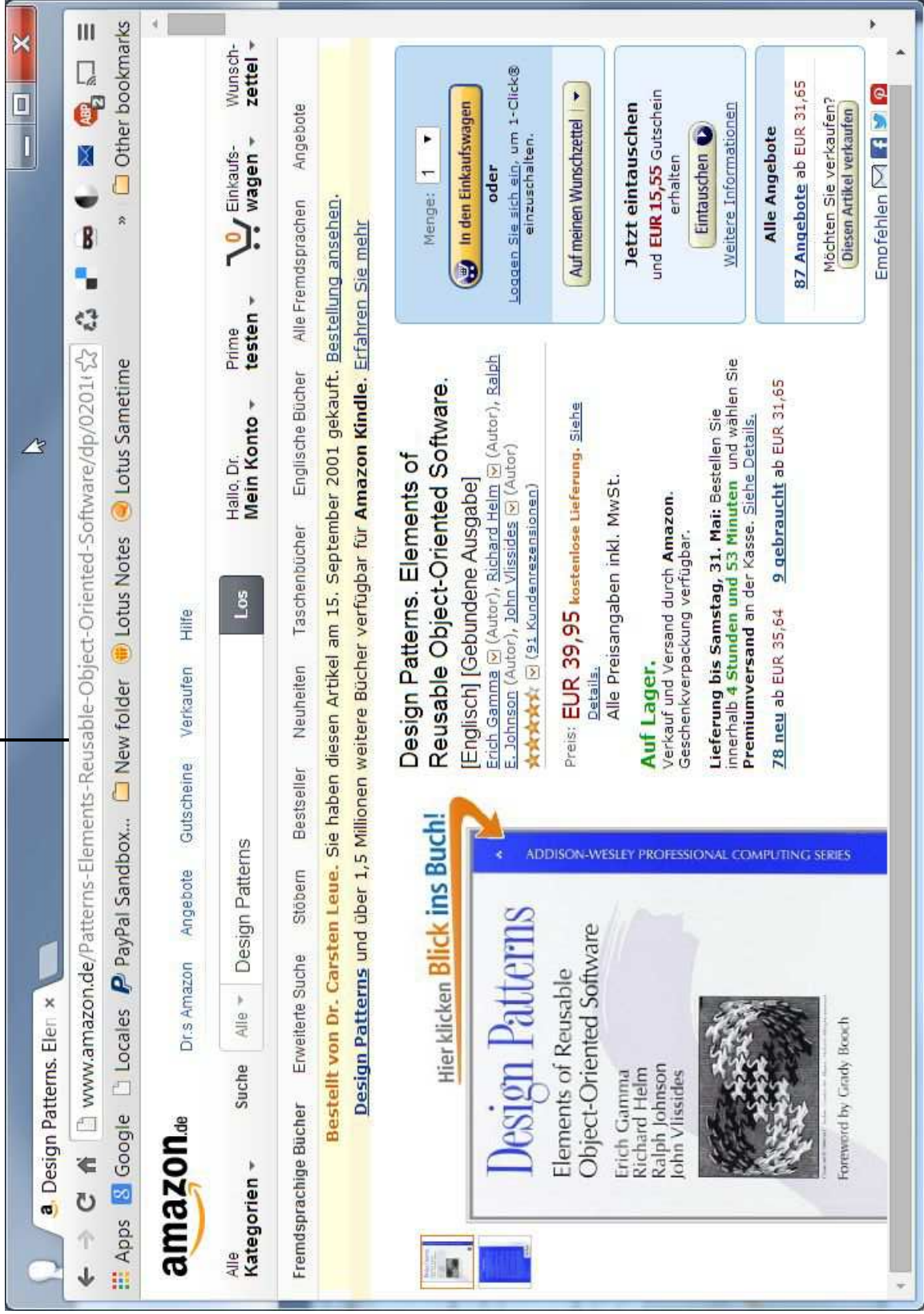


Example: Amazon



Navigational State

http://www.amazon.de/Patterns-Elements-Reusable-Object-Oriented-Software/dp/0201633612/ref=sr_1_1?ie=UTF8&qid=1401438971&sr=8-1&keywords=Design+Patterns





Navigation

亲，请登录 免费注册 手机逛淘宝

最新中国味

淘宝网女装
nz.taobao.com

更多

首页新品风格搭配

女人们市场 > 女装/女士精品 > 裙雪纺欧根纱 > 共 16.31万 件宝贝

女装/女士精品

连衣裙 中长款连衣裙 T恤 短袖连衣裙 雪纺裙 雪纺衫/蕾丝衫 短外套 半身裙 衬衫 大衣 打底裤 中长裤半身裙 长袖连衣裙 婚纱/礼服/旗袍

品牌 Jorya/卓雅 Max Mara Girdear/哥弟 秋水伊人 Ochirly/欧时力 Zara 韩国 Etam/艾格 Goetia/歌莉娅 INMAN/茵曼 DuoYi/朵以 2K12 淑女 AERI 金兰妃 ELF SACK/妖精... 三彩 冰激凌色 衬肤色 BF风 渐变色 马卡龙 巴洛克风 透视 垂感 海军风 连体衣 几何图案 塑身 性感 个性动物纹 动物纹 A字款 挂脖 露背 原宿风 蓬蓬

选购热点

更多选项 风格 元素 年份季节 适用年龄 是否商场同款

默认排序 销量 人气 信用 总价 - ¥ + ¥

☐ 天猫 ☐ 二手 ☐ 拍卖 ☒ 新品 ☒ 旺旺在线 ☒ 消费者保障 ☐ 尺码推荐 ☐ 7+退货保障 ☐ 折扣促销

共 100 页，将第 1 页宝贝按 默认排序 排序



Aggregation of the

REST state of the

applications into the

REST state of

WebSphere Portal



Base concept: Navigational State



- Navigational state is the information required to render the **view** of a portal page
- It is kept in each URL (REST pattern)
- In WebSphere Portal the complete navigational state is an **aggregation** of the navigational state of **portal** and the state of the **portlets**
 - Portal state: page selection, expansions, label mappings, action targets, ...
 - Portlet state: render parameters, window state, portlet mode
- Navigational state does not "**change**", instead each "render interaction" (HTTP GET) simply displays different views of portal
- New navigational state can be set by generating a **new URL** or as a **result of an action**
- Tip: you can think of navigational state as a "**clipboard**" on the URL

Key-Requirements for URL handling



- Back- and Forward-Button Support
 - The user can use the browser's back and forward buttons to switch between recent „views“
- Bookmarkability
 - The „current view“ can be saved into a client side bookmark
 - The bookmark can be accessed at any time (probably requires a logon first)
 - The bookmark must remain valid across portal versions
- Crawlability
 - Web-Crawlers (e.g. Googlebot) can crawl portal pages and index them
- Cacheability
 - Views of pages must be cacheable
- Non-Functional Requirements
 - Respect URL size limitations (2KB) as imposed by RFC 1738
 - Performance during URL generation
 - URL legibility

Navigational State



- Navigational state is all information required to define the „view“ of portal content
- Navigational state is „encoded“ into the URL
- The URL is the only place to store navigational state in → all URLs need to contain the complete navigational state
- Navigational State contains ...
 - Page selection
 - Label selections
 - Expansion states of navigation
 - Theme-Template
 - ...
 - **Render Parameters of all visited portlets**
 - Shared application context (aka public render parameters)
 - Selected locale
 - Context selection (projects, virtual portal, tenant, ...)

This is a lot of information that cannot be handled by the application independently

Different Navigational State results in different URLs.

(but the same Navigational State does not necessarily result in the same URLs)

Basic Approach to URL Encoding



- The navigational state is represented as a bean
- The bean is encoded into a URL
 - Uses only URL compatible characters (using a modified base64 encoding)
 - Applies compression to keep the URL short (modified GZIP encoding)
- Rich URL:
 - only **358** bytes URL vs. **1521** bytes document

```
/c1/1ZDRaoMwFlafpQ8wckxidJepbsno7Iwatd6IF6Mos26sdJ1Pv4QyBhYtyy
EXh4_v55yDKmTq0JzcfXNsh0PzhkpUsVqAAvIoMYDgDLClselpxwEAw3eszj
dOmscCO0DjB8MDiTv4Alr8yxZ0fbEJYwERAbP2HOXshk1v2IXddXl2y2HmcZ
j4frhdW_8eRzLC_hpP-
GQCnyxxu_Oip2iHKu_vuv5LQg13tQ59e1yMMlQCrDNufadP2bdmBXmpyP
JleeKh-
dNH32tVpMcodNLDjDXM607k6N_c4aCfx_VVc50Huc6R4khP8sENP_4lOyO
m4yHPpX9N5rXUL7lMb9ydbeoB-InAFV
```

Example State Document,
1521 bytes

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <state type="navigationnal">
    <selection selection-node="6_G0Q03FH200GA602523R7Q11000">
      <mapping src="6_VK1SVPG2104PE025CH2U0700Q3" dst="6_G0Q03FH200GA602523R7Q11000"/>
      <mapping src="6_VK1SVPG21G4BE025CH366C3GC6" dst="6_VK1SVPG21G4BE025CH366C3GA6"/>
      <mapping src="6_VK1SVPG21G4BE025CH366C3GC4" dst="6_VK1SVPG21G4BE025CH366C3GA6"/>
    </selection>
    <expansions>
      <node id="6_VK1SVPG2104PE025CH2U0700Q3"/>
      <node id="6_000000000000000000000000000000A0"/>
      <node id="6_VK1SVPG218DNB025C92MHM28B2"/>
      <node id="6_VK1SVPG21G4BE025CH366C3G83"/>
      <node id="6_VK1SVPG21G4BE025CH366C3GC6"/>
      <node id="6_VK1SVPG21G4BE025CH366C3GC4"/>
    </expansions>
    <portlet id="7_G0Q03FH208OR40255UUD810002">
      <parameters>
        <param name="0">
          <value>n</value>
          <value>3</value>
          <value>2</value>
        </param>
      </parameters>
    </portlet>
    <portlet id="7_G0Q03FH20GUS40255U065720G5">
      <parameters>
        <param name="0">
          <value>n</value>
          <value>3</value>
          <value>1</value>
        </param>
      </parameters>
    </portlet>
    <portlet id="7_G0Q03FH208OR40255UUD810001">
      <parameters>
        <param name="0">
          <value>n</value>
          <value>3</value>
          <value>3</value>
        </param>
      </parameters>
    </portlet>
    <theme-template>Home</theme-template>
  </state>
</root>
```

Why are Rich URLs illegible?



- URL length optimization
 - Rich URLs are compressed to ensure the shortest possible encoding. The resulting byte stream is not human readable.
 - URLs must only contain a restricted character set. The standard %HH encoding is inefficient for binary information.
- Compatibility
 - The default configurations of many security proxies block certain URL patterns. By using a base64 based encoding, WebSphere Portal can guarantee compatibility.
- Performance
 - URL generation performance is key to web site performance. Factoring our certain pieces of state in a human readable form in the URL is a computationally expensive operation.



Delta Encoding

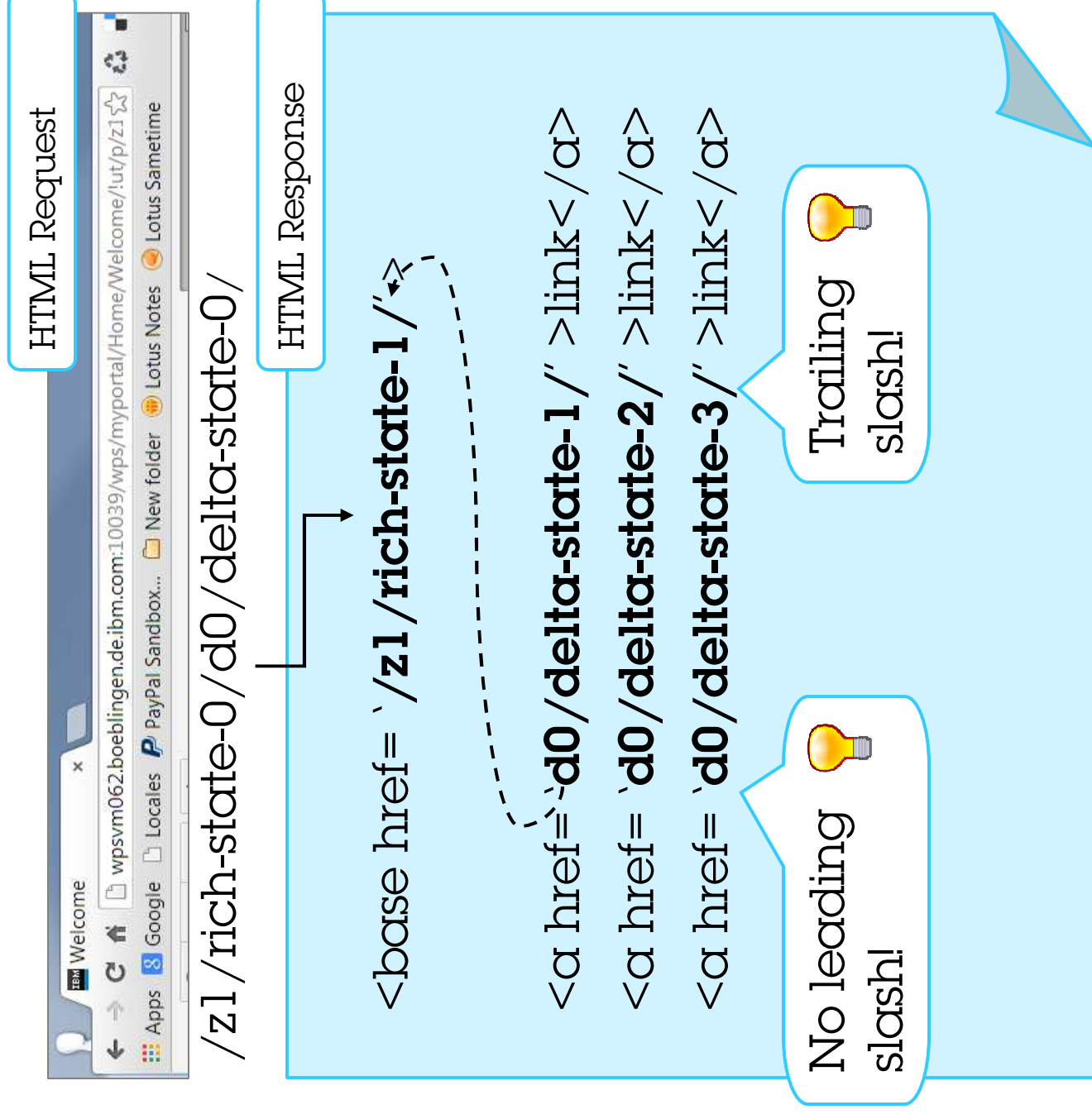
- Problem to Solve
 - Each URL has to encode the complete navigational state (REST pattern)
 - This can be time consuming and will in general result in very large HTML documents
- Solution
 - We assume that each URL only **slightly modifies** the base state
 - Only encode the modification in form of a relative URL

Codec identifier Codec identifier

/z1 / rich-state / d0 / delta-state /
Codec identifier Compressed base state Modification

The Ideal Situation

- The URL addresses navigational state via α delta URL
- The base tag normalizes this state, so the normalized state can be used as a new base for subsequent deltas
- Each URL only encodes the relative modification



Pros and Cons of Delta URLs



- Small Markup Size
 - Only the modifications are encoded
- High Performance
 - Since modifications are typically small, they do not have to be compressed, what saves CPU cycles

Trade-offs between these pros and cons can be configured.

- URL Aliases
 - The same state can be encoded via different URLs (analogy: $10 + 1 = 11$ and $9 + 2 = 11$)
 - Problematic for caching (see next slides)
 - Unacceptable for crawlers (see next slides)
- Relative URLs
 - Require a base tag which in turn requires an absolute URL. This can be problematic with reverse proxies
 - HTML and JS code can be written such, that it cannot deal with relative URLs

- URL length
 - Although the individual length of the relative URL is short, the combination of delta and base is longer than the resulting base




Delta URLs

- Per default delta URLs are **enabled**
- The generation of delta URLs can be disabled
 - No URL aliasing, resulting in better cache hits
 - Shorter overall URLs



com.ibm.wps.state.dom.delta.DeltaDocument=false

 We **DO NOT** recommend to disable delta URLs, the performance penalty at URL generation time outweighs the benefits.





Relative URLs

- Relative URLs are **disabled** per default
 - Mainly because it might break applications that use “strange” JS patterns
- Relative URLs can be enabled
 - Add a base tag to the theme

com.ibm.portal.theme.hasBaseUrl=true

- Declare the portlet application compatible with relative URLs (per application)

com.ibm.portal.state.url.allowrelative=true

- With disable relative URLs, delta URLs can still be used
 - Still good URL generation performance
 - However larger markup size

Constant prefix for all URLs (copy/paste)

``
link
``

`d0/delta-state-1/' >`
Modification per URL

Relative URLs – Application Development

- Relative URLs are defined in RFC 3986
- The browser handles the URL resolution
 - But only if the URL string appears in URL attributes in the markup!
- **Do NOT** encode URLs directly into script
 - In the general case this will NOT correctly resolve relative URLs


- **DO** always encode URLs in HTML elements and read the HTML attributes via script

- Browser magic will always return absolute URLs when attributes are read


 **DO NOT** do this!

```
var data = {url:'d0/delta'};  
<a onclick="doSth(`d0/delta`)">link</a>
```

 Reference the HTML element containing the URL

 instead **DO** this!

```
var data = {url-id:'id'};  
<a id='id'  
  href='d0/delta'  
  onclick="doSth(this.href)">link</a>
```

 Read the URL from the attribute (will be absolute!)



URL Aliases and Caching

- The same base state can be represented via different delta states
 - ``/z1/rich-state-1/d0/delta-state-1/`` → ``/z1/rich-state-3/``
 - ``/z1/rich-state-2/d0/delta-state-2/`` → ``/z1/rich-state-3/``
- HTTP Caches use the URL as the cache key
 - There exist more cache entries than theoretically required
 - No functional problem, but performance is not optimal



We recommend to accept this trade-off between URL generation performance and cacheability.

URL Aliases and Search Crawlers



- Search Crawlers follow and index all GET-URLs of a site
 - URL aliases would result in multiple entries in the search index
 - Lower ranking of the individual URL

- URL aliases are not acceptable for Crawlers

■ Solution: **URL normalization**

- WebSphere Portal detects crawler agents
- For crawlers all URLs will be full, **unique** rich URLs (performance of URL creation is not an issue)
- In addition the amount of navigational state in normalized URLs can be limited - to limit the size of the transitive closure of portal URLs.

- Use a Sitemap to seed the Crawl
 - A sitemap is a standardized XML document that lists all sites to be crawled
 - Providing a sitemap allows portal administrators to control the set of indexed pages



<http://www.ibm.com/developerworks/xml/library/x-sitemaps/index.html>

What else is Part of the URL?



- Navigational state is the most important part, but there is more
 - Since navigational state is application state, it can be cleared by the application
 - However some pieces of information MUST always be available, even if navigational state is cleared
- ➔ Contextual information
 - Information that must be preserved independent of the navigational state
- Virtual Portal
 - Independent of the navigational state the addressed virtual portal must be encoded into the URL
- Project
 - Projects act like workspaces and their context must always be preserved
- Multi Tenant Support
 - Similar to virtual portals, this feature represents scoping information

Format of Rich URLs

```
<protocol>://<host>[:<port>]/<context>/<servlet>  
[/$project/<projectname>][/$tenant/<tenantname>][/<virtual-portal>]/!ut/pl/<codec>/<payload>]*
```

Frequently Asked Questions



- Why is the navigational state not just a **Query Parameter**?
 - relative URLs do not work with query parameters
 - Portlets are free to add custom query parameters to URLs (JSR286, PLT.11.1.1.1). This would override query parameters added by the portal application
 - Query parameters run the risk of being blocked by security proxies
- Why is the state not just encoded in the **session** or a **cookie**?
 - Relying on the session renders the page uncacheable
 - No back button support
 - No bookmark support
 - No crawler support
 - Reduced performance because of server side resource consumption
 - Requires an action for every interaction



Frequently Asked Questions

- Will Rich URLs result in poor search ranking (SEO)?
 - We do not have any indication that legible URLs result in better search rankings.
 - What is important is that URLs are canonicalized which is the case (see URL normalization)
- Rich URLs are too bulky to be printed or published
 - True, we will discuss solutions on the next slides
- The illegibility of Rich URLs makes it hard to orient oneself in the web site
 - One the one hand side the URL should not be required to get along with a web site, the markup of the site should visualize this
 - Many mobile devices do not even display the URL
 - On the other hand some customers find a visual clue in the URL helpful (see next slides)

URL Handling Limitations



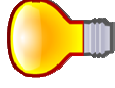
- Even though the state is compressed, the URL might become very long
 - In principle applications are free to encode as much information into the URL as they like
 - The JSF portal bridge e.g. encodes multiple kB of data as render parameters
- To prevent application breakage, WebSphere Portal offloads excessive state into the session
 - ⚠ – The application that excessively uses state is no longer crawlable or bookmarkable
 - Back and forward buttons still work
 - Increased resource consumption on the server because of session usage

- Configurable thresholds of state externalization



b1.keymanager.renderparameters.threshold
b1.keymanager.publicrenderparameters.threshold

■ Recommendation



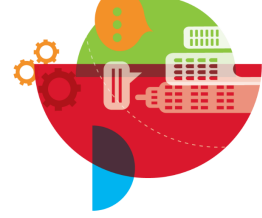
- Design your application to use short parameter names and only encode the information entropy
- For tips and tricks see my presentation “MTL2013 - Portlet Programming Model”

Legibility of WebSphere Portal URLs?

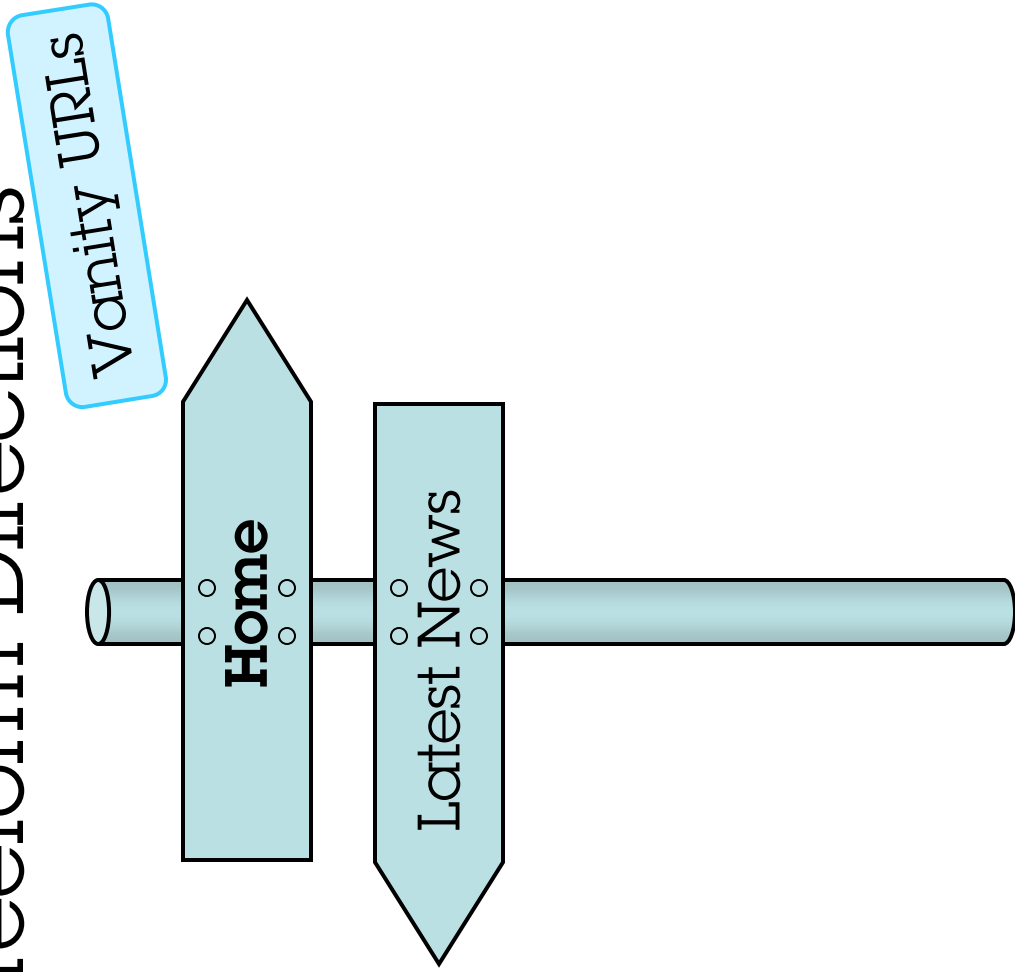


- URLs should be (partly) legible, because this increases the confidence level of the end user
 - By looking at the URL the end user gets an idea what is displayed on the site
- Legible keywords in the URL might improve search ranking
 - No evidence found, yet, still recommended by SEO specialists
- Legible URLs can be used in marketing campaigns or can be printed in brochures
 - Short, crispy URLs can be remembered by human users in cases the URL is not communicated electronically

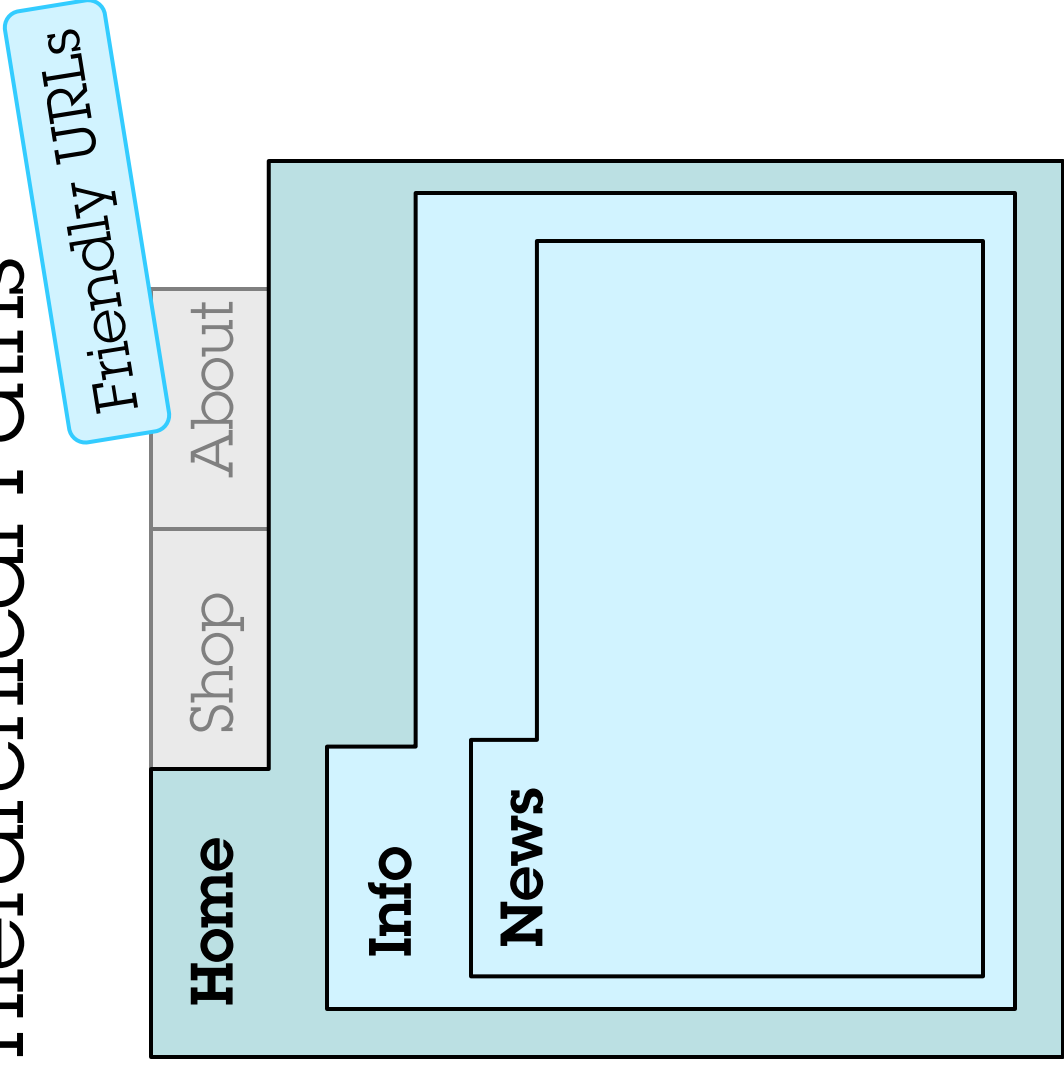
Two Way Approach to Legible URLs



Freeform Directions



Hierarchical Paths

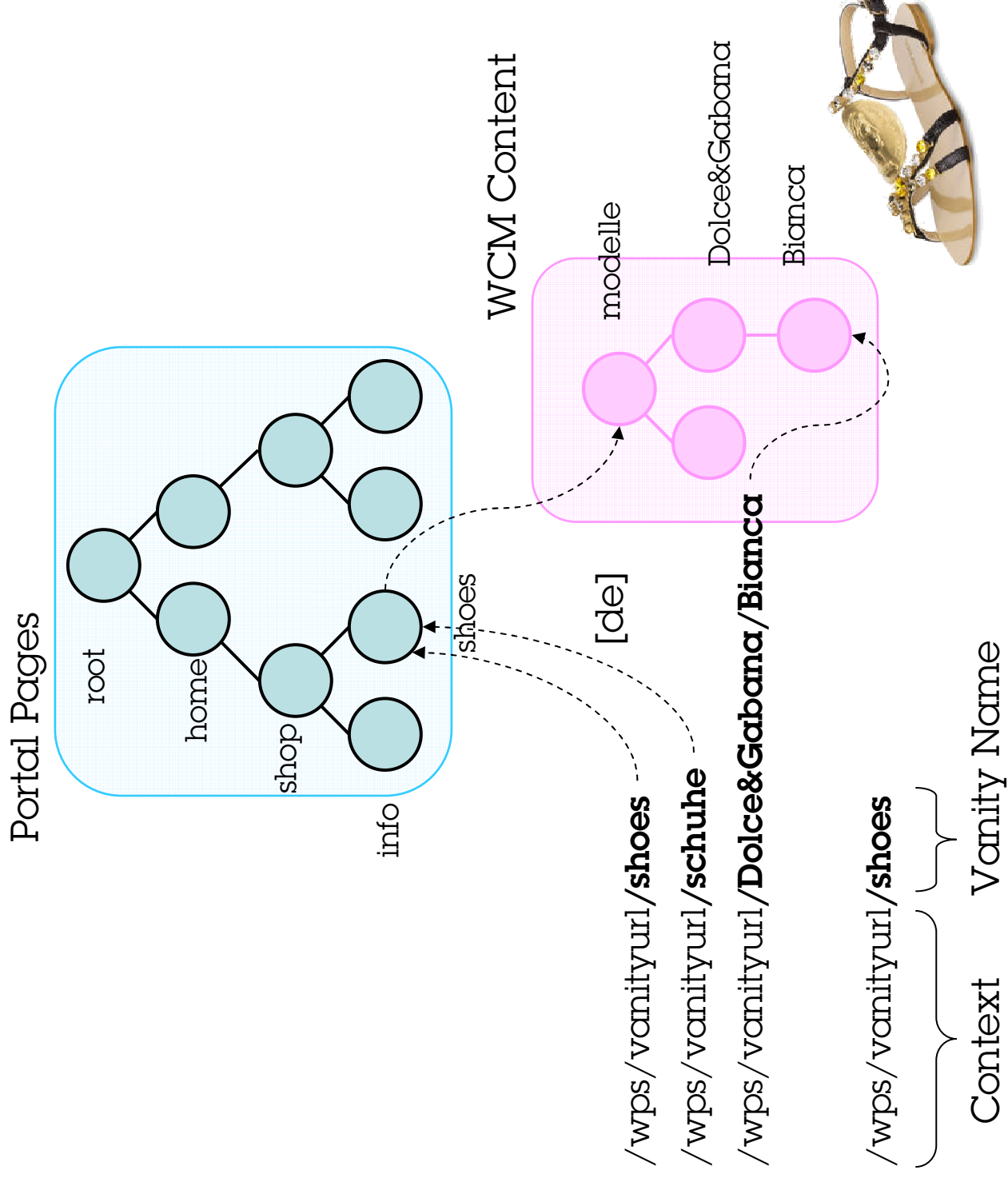


VS

/wps/portal/**home/info/news**

Vanity URLs

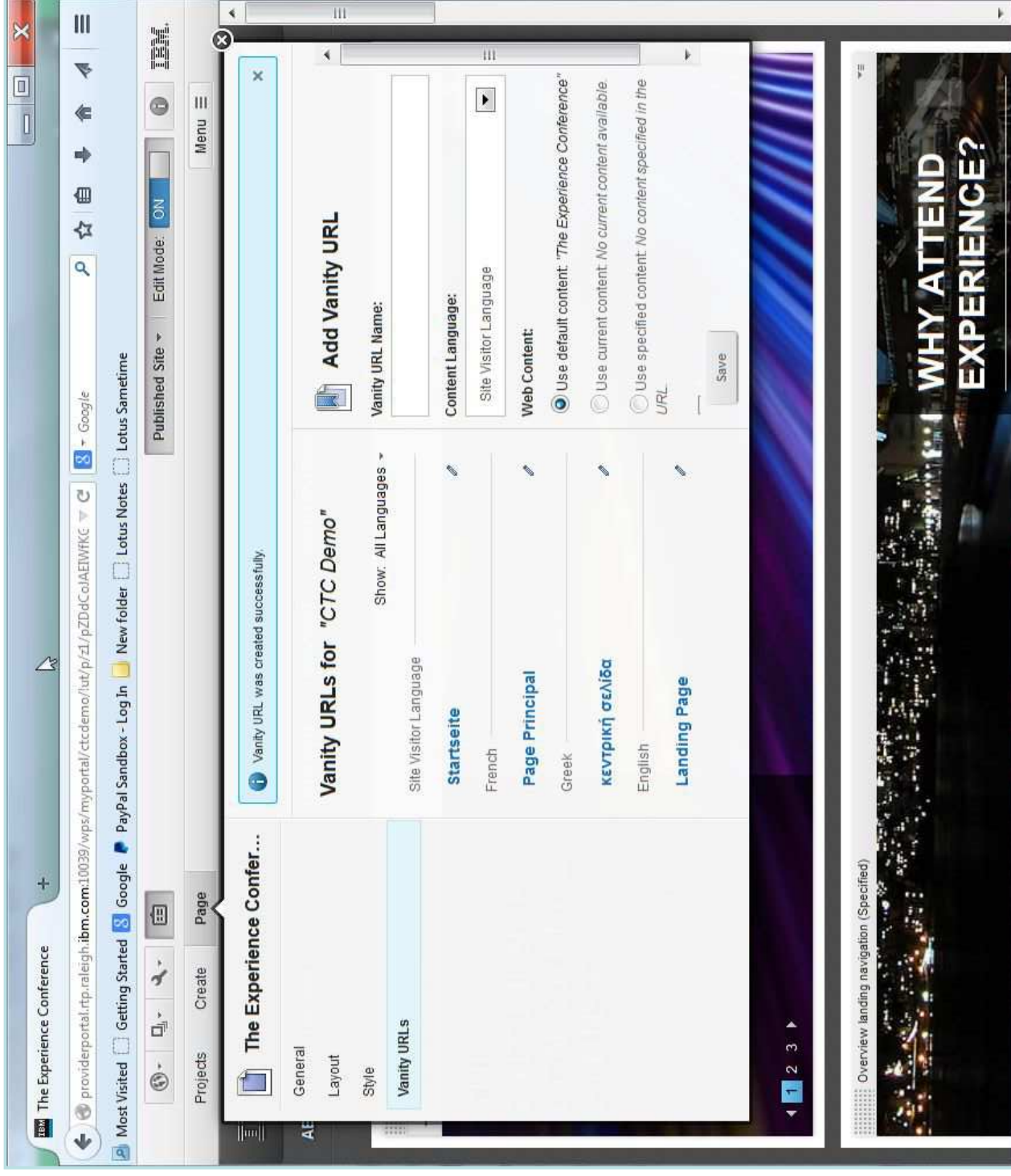
- Vanity URLs directly point to a portal page
 - Optionally include WCM content item referenced on that page
 - Optionally include an explicit locale
- The name of a Vanity URL is free form and not correlated to the site structure
- Full syndication and project support
- Meant to be used for print or marketing campaigns
- **NOT** preserved when navigating the site



Managing Vanity URLs



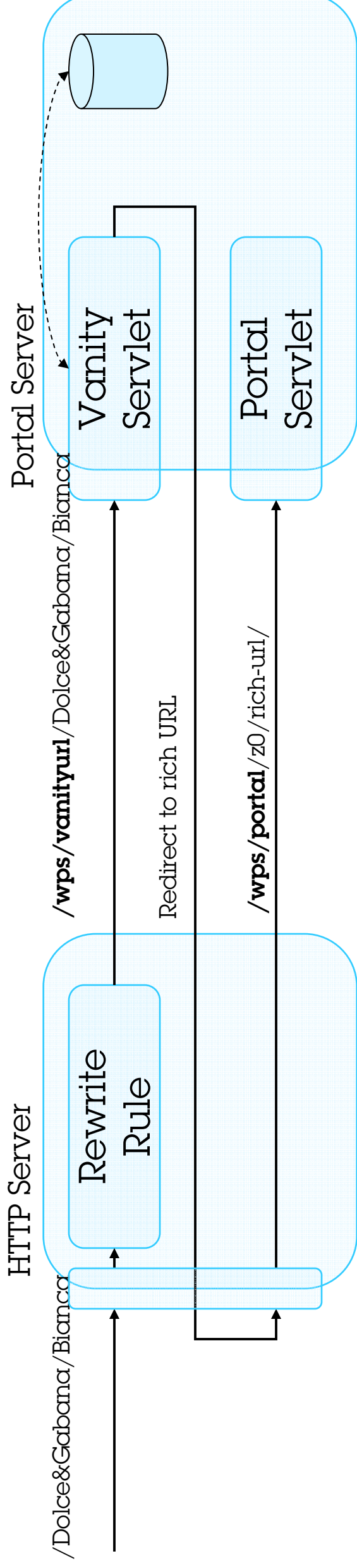
- Since Vanity URLs are tied to a page they can be managed in the page section of the toolbar
- Each portal page can have multiple vanity URLs
 - Different content
 - Different locale
- Vanity URLs are syndicated and cleaned up with the page they point to



Becoming even more Vain

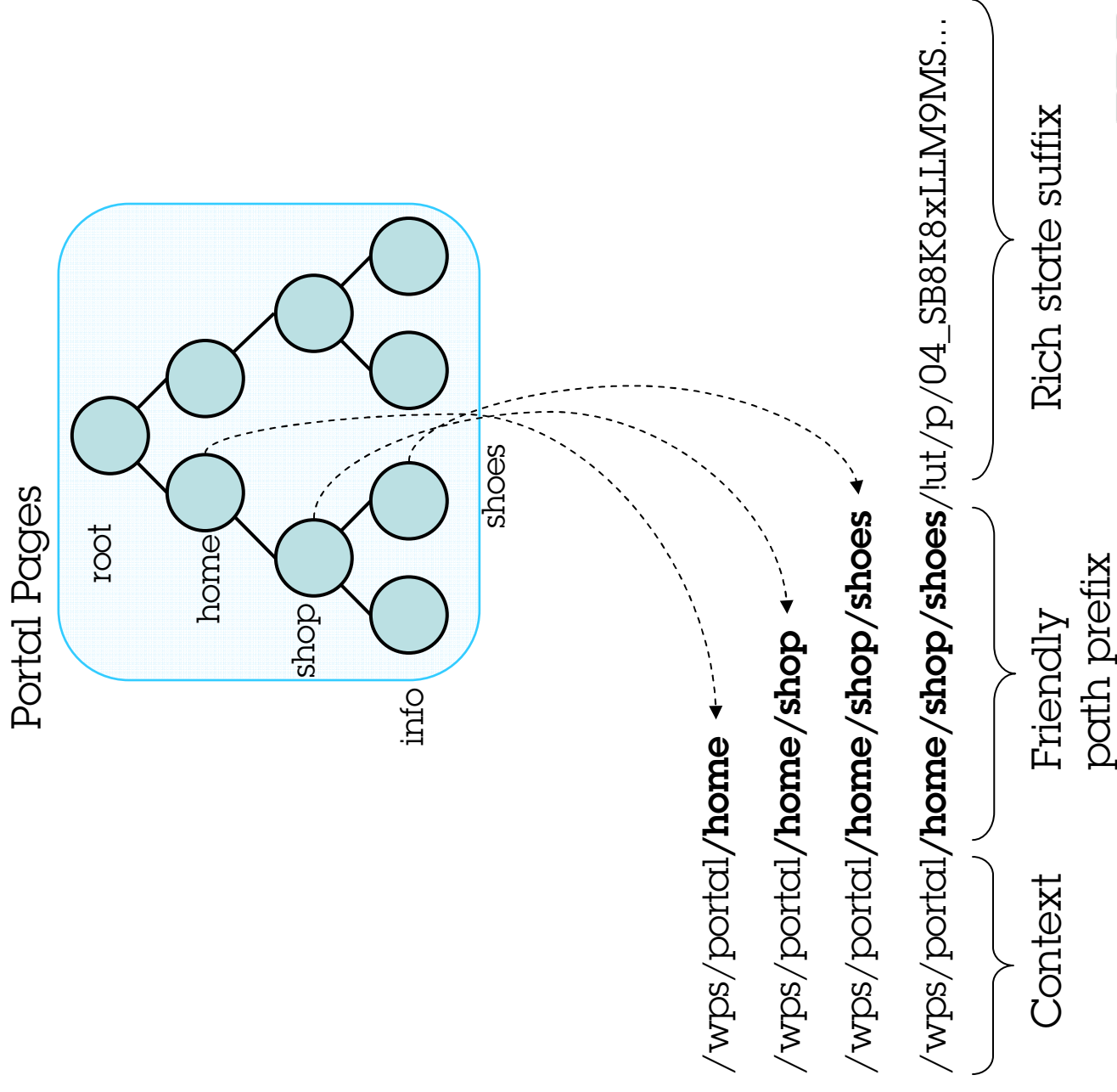


- Vanity URLs contain a context prefix
 - Realized via a vanity URL servlet (**/wps/vanityurl/shoes**)
 - The servlet requires a custom context root to be distinguishable from other applications
- Use HTTP server rules for even shorter URLs



Friendly Page URLs

- Friendly-URLs result in human readable, **hierarchical URL prefixes** that lead to portal pages
- Each portal page (content node) might have a **friendly name** assigned
- The friendly-URL is a hierarchical path constructed from these names based on the **content topology**
- Every generated URL will contain the friendly-path automatically



Friendly WCM Content URLs



- In WCM sites the portal page typically contains a WCM rendering portlet

- The portlet displays content from a WCM library
- The library has its own substructure that should be visualized in the URL

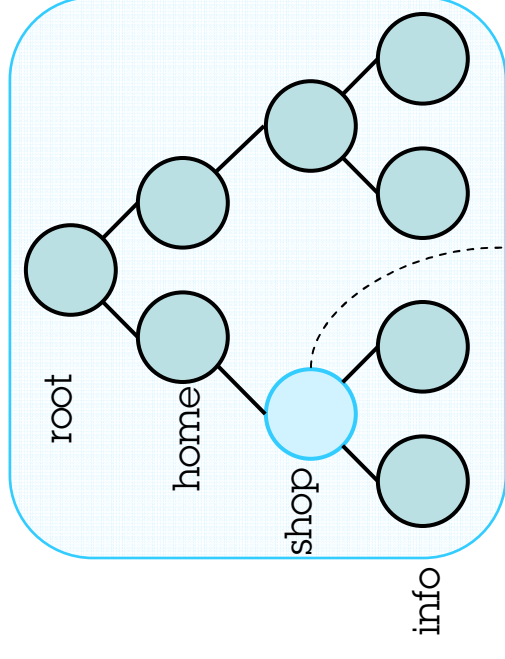
- In addition to the friendly path, a URL can also contain the content path

- Exposed as a public render parameter, so this feature can also be used by non-WCM applications

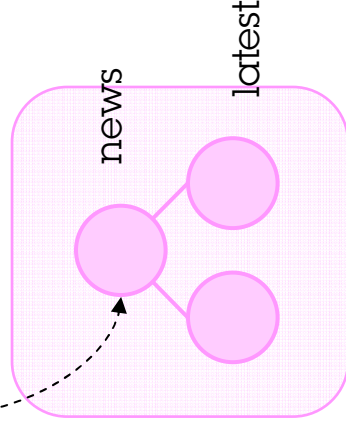
{http://www.ibm.com/xmlns/prod/websphere/portal/publicparams} **path-info**



Portal Pages



WCM Content



/wps/portal/**home/shop/news/latest**/lut/p/04_SB8K8xLLM9MS...

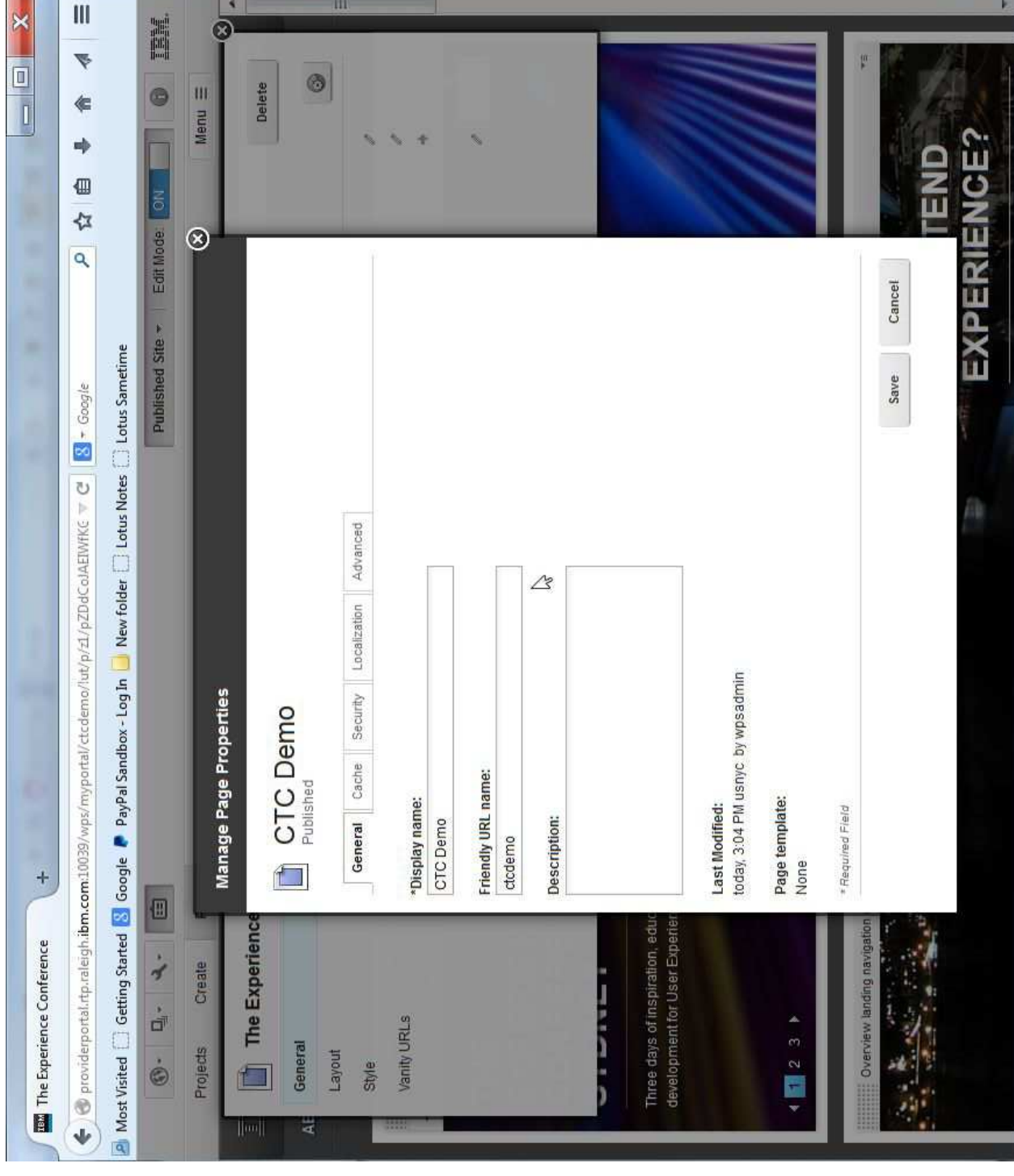
Context Page Content Rich state suffix
path prefix path prefix



Managing Friendly URLs



- Friendly URLs are composed of the **friendly names** of each page
- The friendly name is a property of the page, managed in the “page properties” dialog
- When creating pages, the friendly name is filled in for convenience
- A special “wildcard” name can be used to hide the page from the path
 - com.ibm.portal.friendly.wildcard



What Friendly URLs address



■ Site orientation

- Each portal page only contains a “name” that ends up as a path segment in the URL
- The URL path depends on the actual page hierarchy, so it exposes information of the location of the page

■ Trust

- The system can be configured such that every URL to a page will start with the friendly prefix
- Reverse proxies can use this feature to do page level access control

■ Noteworthy

- Friendly URL prefixes only encode a small part of the navigational state. The remainder is still encoded as a rich URL suffix.
- Friendly paths can be user specific, depending on access control and private pages
- The same friendly path can point to different pages. In this case the user is prompted with a disambiguation logic.

Challenges with Friendly URLs



- Guaranteed friendly URL prefix
 - URLs that result in different friendly prefixed can not longer be relative URLs
 - Computing the friendly URL for every URL on a page adds some small performance overhead
- Ambiguity of the friendly prefix
 - Multiple pages might have the same prefix (the administration UI tries to avoid this but cannot guarantee uniqueness)
 - Ambiguity between the path prefix that is constructed from the page name and the part that is constructed from the content hierarchy
- Configuration Options
 - **friendly.enabled**: enables or disables the feature
 - **friendly.redirect.enabled**: send a redirect if the friendly prefix does not match the addressed page
 - **friendly.pathinfo.enabled**: encode the content path into the URL
 - **friendly.pathinfo.invalid**: validate if the friendly path points to an existing page
 - **friendly.pathinfo.invalid.uri**: decide how to deal with an invalid path info
 - **friendly.force.public**: assume that the friendly path does not depend on the user
 - **friendly.cache.fulldependencies**: assume that friendly URLs are unique

Optimizations for Content Centric Sites



- A portal site (or part of it) might just render static web content without application logic
- Advanced state management mechanisms are not required for such sites
 - Trade-off between application functionality and legibility of URLs
- Configuration options for “stateless” URLs
 - Stateless page URLs
 - Stateless content URLs

Stateless Page URLs



- The navigational state is explicitly removed from the URLs in the navigation area
- The remaining friendly path contains enough information to address the page
- Interactions with applications on the page will still generate Rich URLs



- Configuration Steps
 - Avoid redirects by setting `friendly.redirect.enabled=false`
 - Update the theme JSP to compute stateless URLs for page changes

```
<portal-navigation:urlGeneration  
  contentTypeNode="{nodeID}"  
  keepNavigationalState="false">  
<a href="<%wpsURL.write(out);%>"  
  ...  
</a>  
</portal-navigation:urlGeneration>
```



<http://www-10.lotus.com/ldd/portalwiki.nsf/xpDocViewer.xsp>

Stateless Page URLs



- URLs to the pages will look like
 - /wps/myportal/home/location
 - /wps/myportal/home/blog
- Limitation: Switching pages loses the navigational state
 - Contextual information such as public render parameters
 - Explicitly selected locale
 - ...

Stateless Content URLs



- Used as an extension of stateless page URLs
- The URL generation mechanism of the **WCM rendering portlet** is overridden to construct friendly URLs with page and content prefix
- Interactions with other applications would still generate Rich URLs

■ Configuration Steps

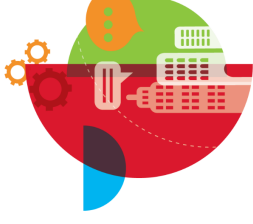
- Configure the system for “stateless page URLs”
- Enable the **base** tag in the theme
- Write a URL generation Java plugin (stubs are available on the Info Center)
- Deploy the plugin



/wps/myportal/mypage/**mycontent**



<http://www-10.lotus.com/ldd/portalwiki.nsf/xpDocViewer.xsp>



Stateless Content URLs

Stateless Content URL

IBM

IBM

IBM

IBM

Home

localhost:10039/wps/myportal/Home/Welcome/breaking news

Test Page

Breaking News

More info can be found in our [new release](#)

New Release

Social Business is done with a Social Portal

Latest Technology

http://localhost:10039/wps/portal/Home/Welcome/new+release

Stateless Page URL

IBM

IBM

IBM

IBM

Home

localhost:10039/wps/myportal/Home/Welcome

Getting Started

CTF JSR Test Page

CTF Legacy Test Page

Published Site

Log Out

Actions

WPSADMIN

Search

Latest Technology

More info can be found in our [social portal](#)

Please have a look at this [Getting Started](#) Portal Page

Breaking News

New Release

Social Business is done with a Social Portal

http://localhost:10039/wps/myportal/Home/Getting-Started

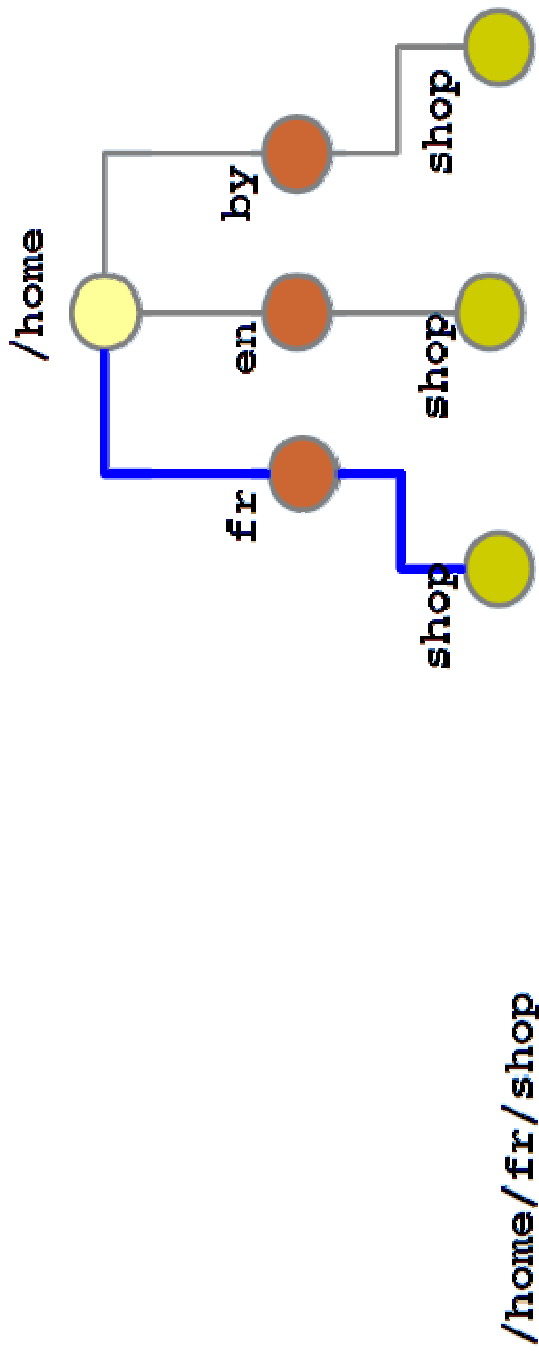


Region Identification - “Geo-targeting”



- Enable friendly URLs for Portal pages
- Structure the Portal site to reflect which groups of pages are targeted for a specific region or country
- Result: friendly URLs which carry the respective region information as part of the relative path information in the URL

<http://www.myco.com/wps/home/fr/shop>
<http://www.myco.com/wps/home/en/shop>
<http://www.myco.com/wps/home/by/shop>



Stateless Content URLs



- URLs to the pages will look like
 - /wps/myportal/home/location/**news**
 - /wps/myportal/home/blog/**blog+entry+1**
- Limitation
 - Same as for stateless page URLs
 - No WCM application state (e.g. paging)



URL Decoding

- The state contained in a Rich URL can be decoded via a REST service
 - For debugging or information purposes
 - For programmatic access to state information, e.g. for monitoring or auditing
- The URLs are **not encrypted**, state is freely accessible
 - So do not encode sensitive information such as passwords into the URL
- Sending GET to the REST service decodes a URL
- Sending a POST with the (modified) decoded document, constructs a new URL

Tips for working with URLs in Portlets



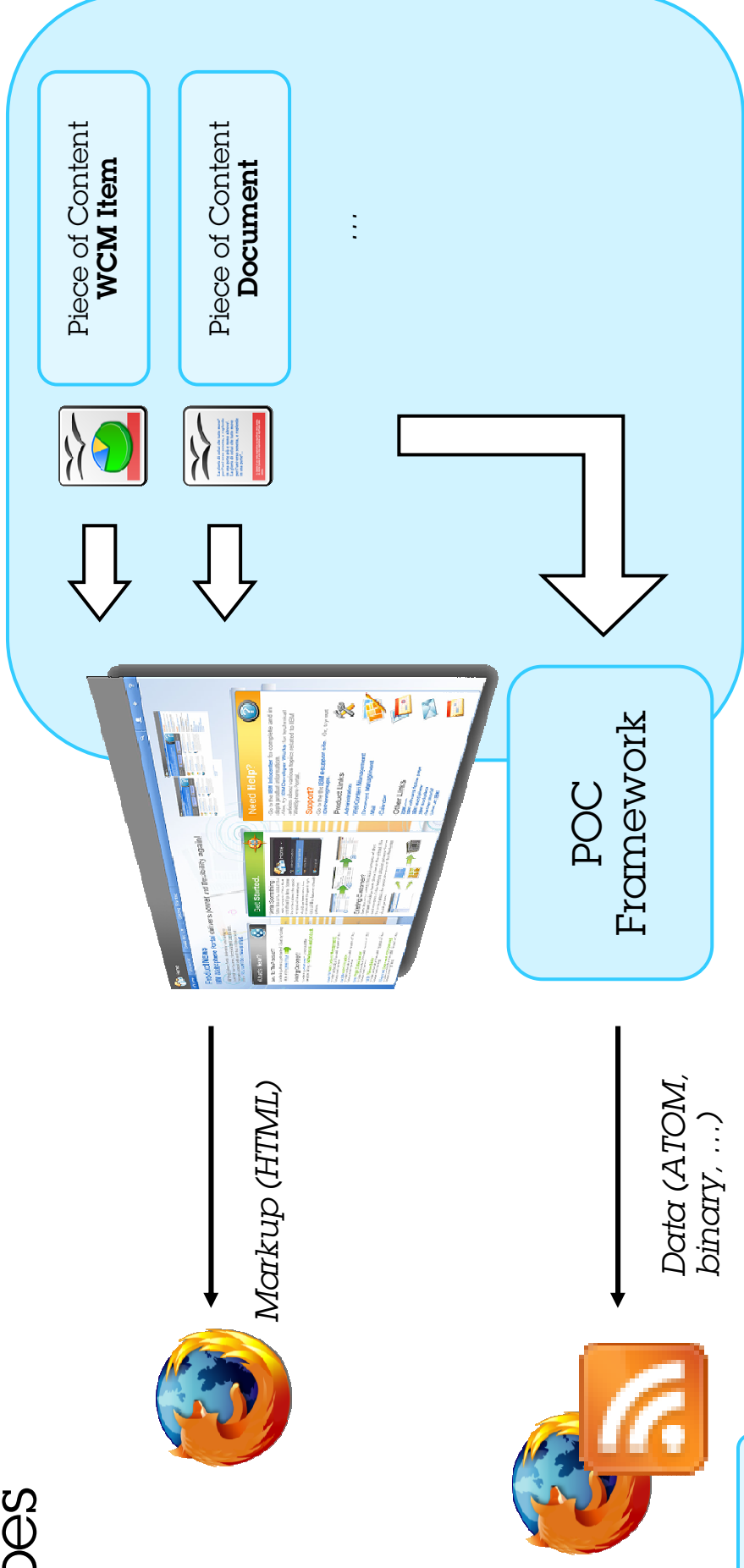
- Use the JSR286 portlet APIs for creating URLs in 99% of the cases
 - Best URL generation performance
 - Support for server side fragment caching
 - Support for WSRP
- Advanced APIs for the fringe case
 - State API
 - Friendly URL API
 - POC URL API
 - Vanity URL API

- Use the new set of “portal” **public render parameters** instead of the State API

– Page selection	
– Locale	
– Theme Template	
– Solo Mode	
– Device Class	
– Edit/Info Mode	<code>http://www.ibm.com/xmlns/prod/websphere/portal/publicparams</code>
– ...	

Resource Addressability

- The POC framework adds an **entry point** to WebSphere Portal that allows to access POCs in a variety of mime-types
- Customers can easily add their own POCs and leverage the POC Framework to serve them



POC = Piece of Content

WebSphere Portal



Piece of Content URI (POC)



■ When to use

- the actual portal context is not known a-priori
- to reference content in user-facing URLs
- passively share a context (HTTP GET semantics)
- context should become part of the navigational state (bookmarkable, browser navigation, browser caching)

■ When NOT to use

- active notifications of changed needed
- backend state changes based on the event (HTTP POST semantics)
- portlet must run on non-WebSphere servers

■ Advantages

- server-side concept that works with all JSR 286 and WSRP 2.0 implementations
- state stored in the URL, not on the server
- bookmarkable
- common programming model for private and shared navigational state
- out-of-the-box functionality

■ Disadvantages

- since the POC URI is encoded into the URL, the 2k limit may be reached
- requires server side roundtrip



How is a POC identified?

- A POC is identified by a URI (RFC 3986)

- **POCURI** = **<scheme>:<scheme-specific-part>**



*Interpreted by the
POC framework to
handle a POC*

Interpreted by the POC handler

- Components that expose POCs can define new schemes or reuse existing ones

- The POC URI is **NOT the URL** to the resource



Identity



Location



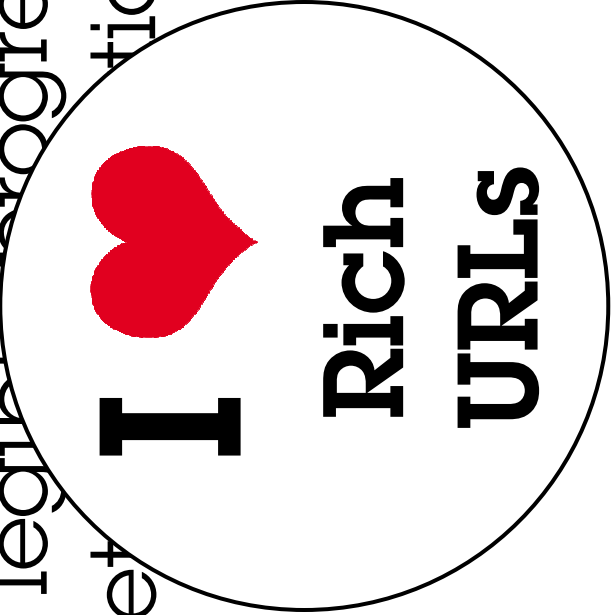
How is a POC addressed?

- The **URL** to a POC can be constructed via string-concatenation or via a Portal-API call
 - /wps/[my]poc[/<vp>]?**uri**=<**scheme**>:<**ssp**> (1)
 - /wps/[my]poc[/<vp>]/<**scheme**>/<**ssp**> (2)
 - Both URL flavours are equivalent, (1) is helpful for FORM submits, (2) is required if the resource contains relative URL references (e.g. images in CSS)
- POC URLs are fully **virtual portal aware**
- The **[mypoc]** servlet will establish a WAS security context, whereas **[poc]** is used for anonymous access
- Example: URL to a content page

/wps/mypoc/**nm**/**oid:ibm.portal.Home**

Summary

- URLs are the key aspect to state management and context sharing
- The design of Rich URLs offers full REST support
- Rich URLs are **compact**, considering their power
- URLs can be made legible progressively, allowing to find the best compromise between functionality and readability



I ❤️ Rich URLs





Questions

