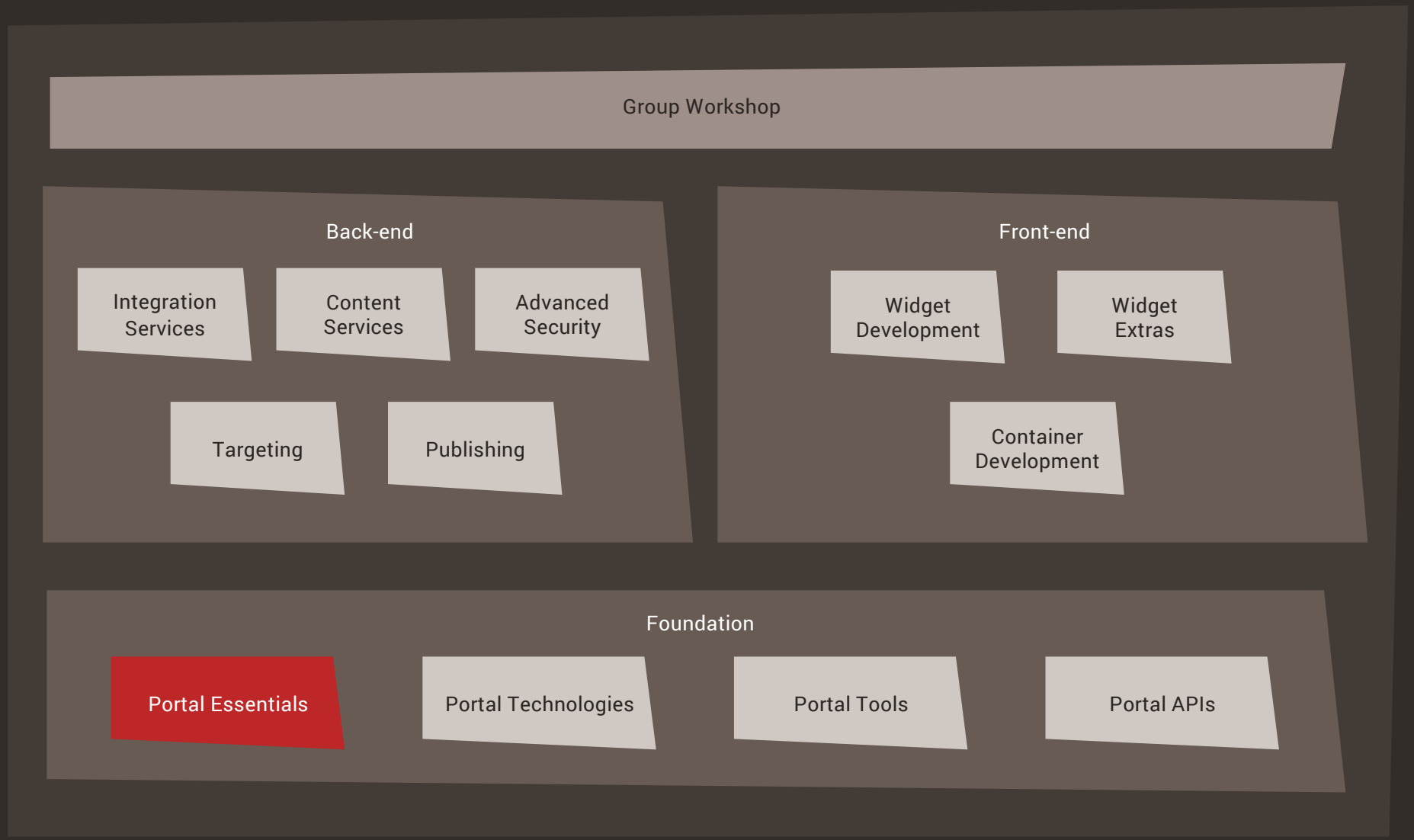
BACKBASE

Portal Essentials

Focus Area: Foundation

BACKBASE

Training Overview

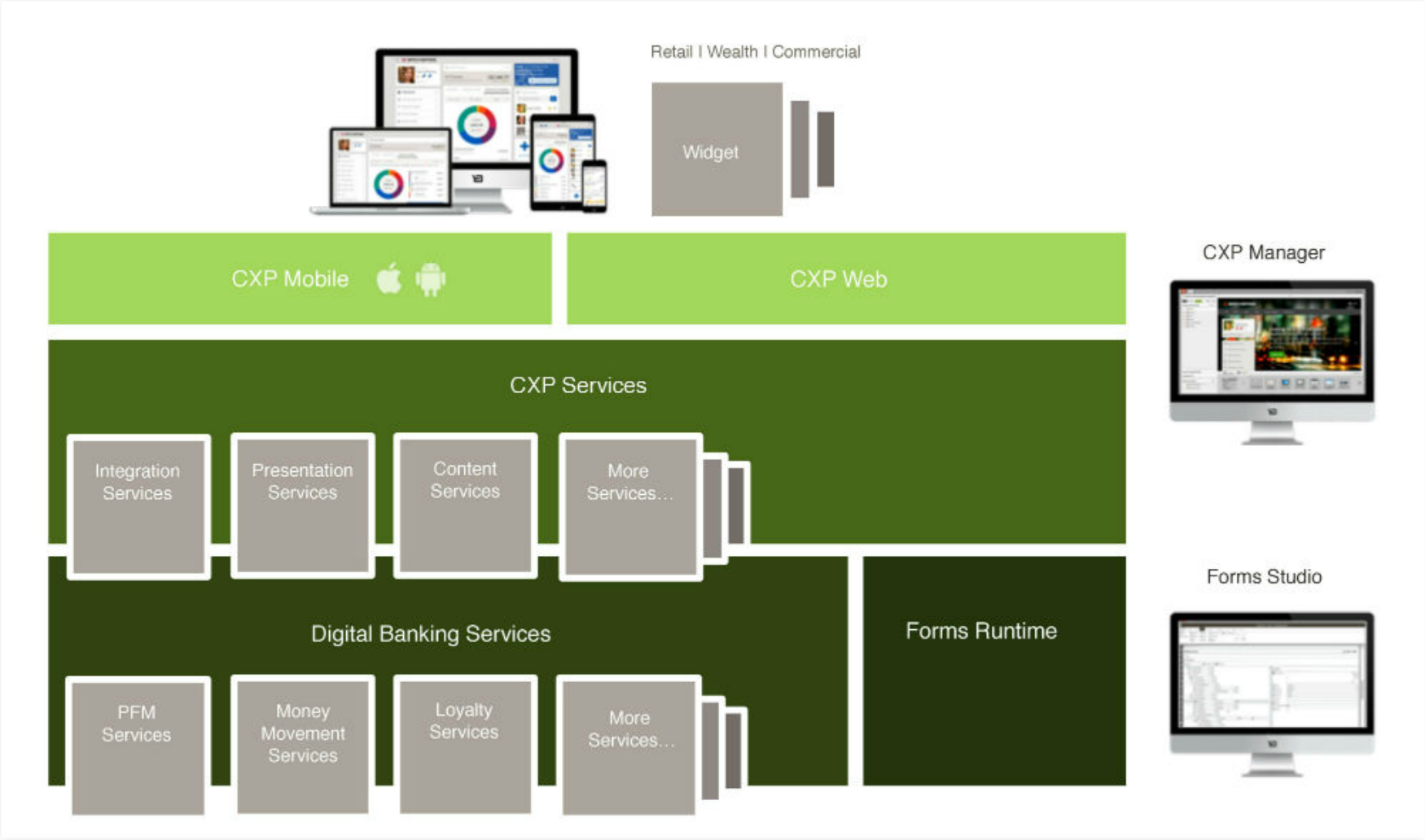


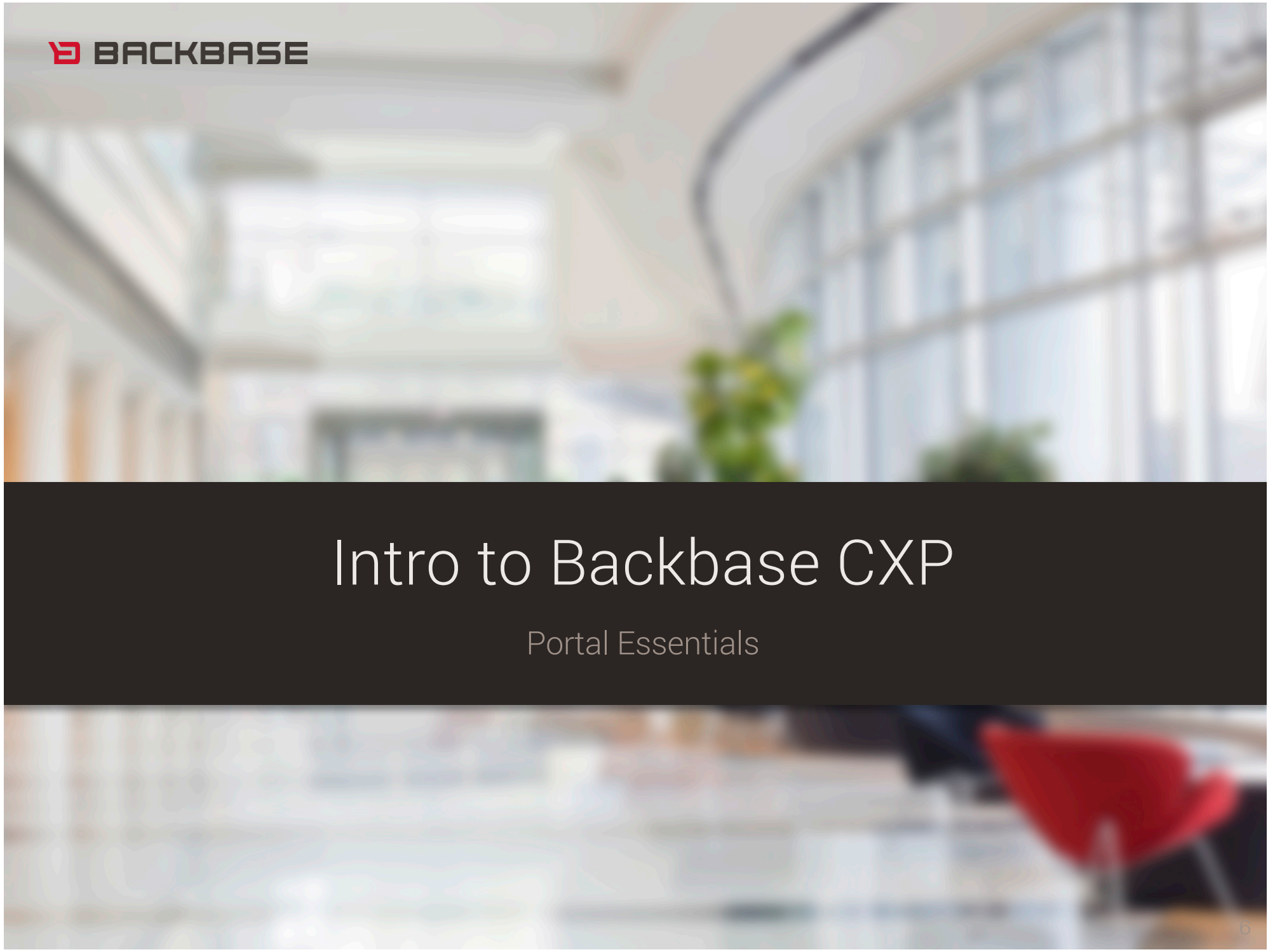
Objectives

1. Learn the foundation concepts of Backbase CXP
2. Overview of the Backbase CXP logical architecture
3. Understand the Portal Model
4. Understand how the Portal Model is rendered
5. Understand how the Portal Model is secured

Backbase Products

Portal Essentials

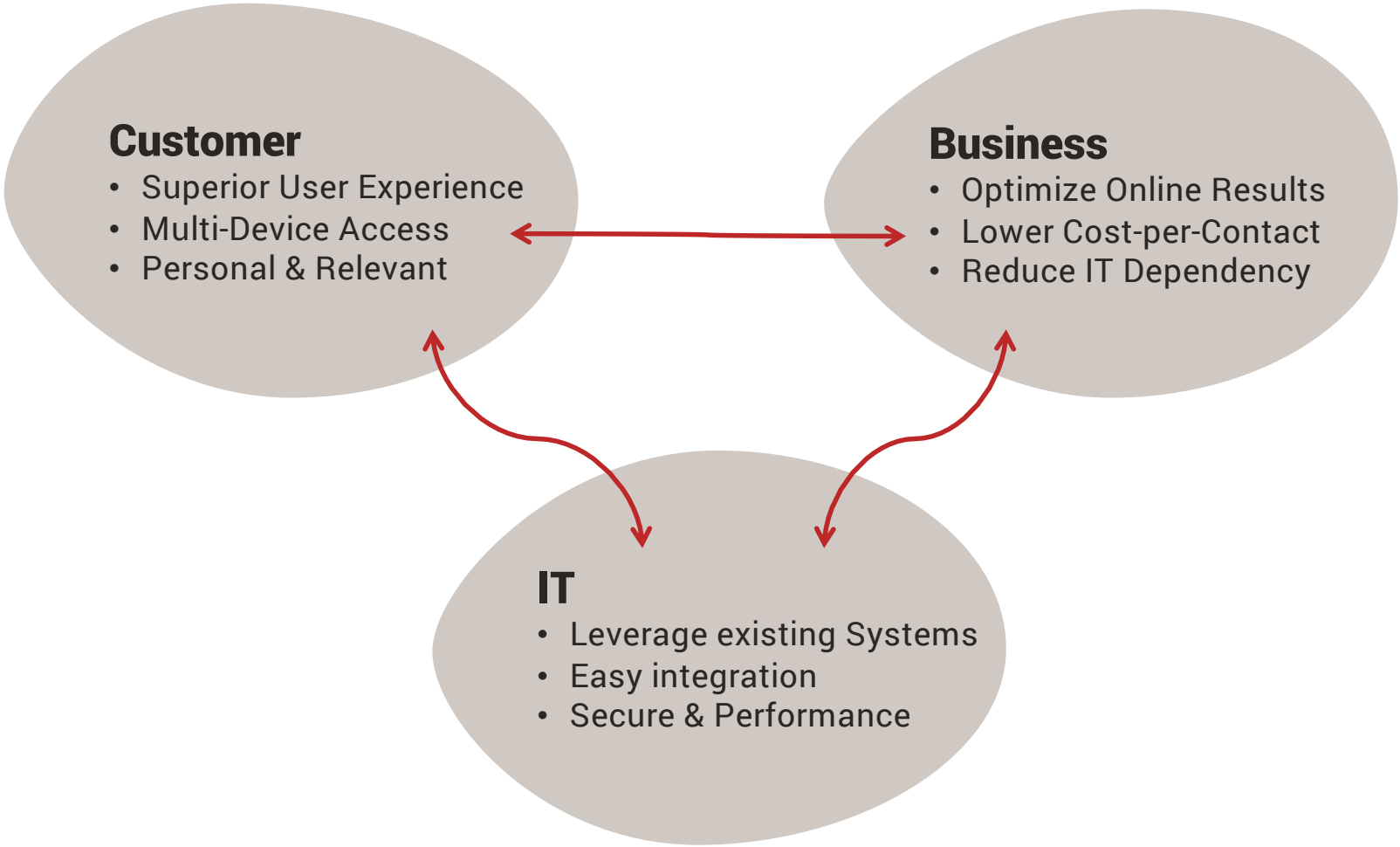

Retail I Wealth I Commercial
Widget
CXP Mobile
CXP Web
CXP Manager
CXP Services
Integration Services
Presentation Services
Content Services
More Services...
Digital Banking Services
PFM Services
Money Movement Services
Loyalty Services
More Services...
Forms Runtime
Forms Studio


BACKBASE
Intro to Backbase CXP
Portal Essentials

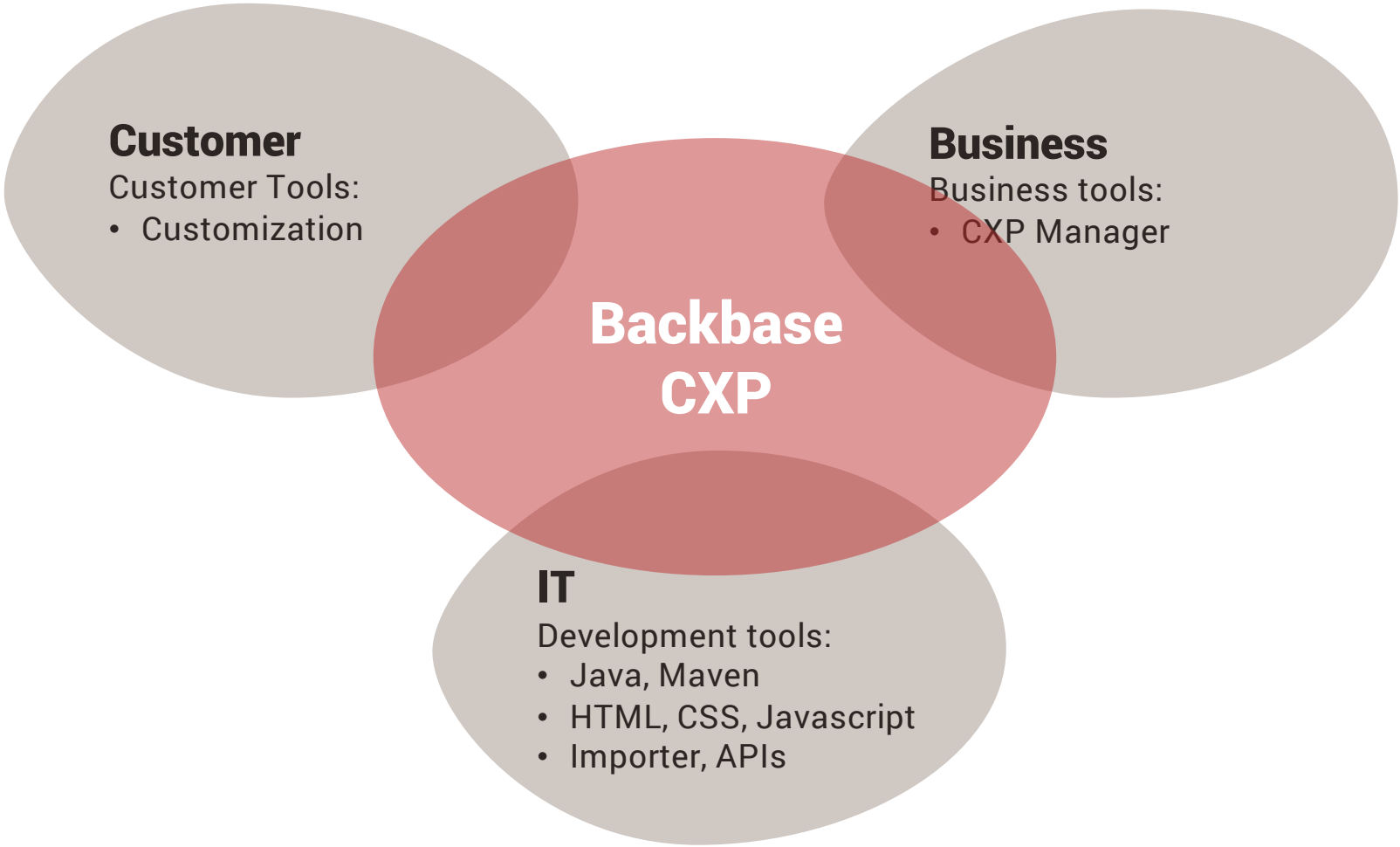
- Gartner defines portals as **personalized points of access to relevant information, business processes, and people**

- Brings value to **three major audiences**:
  - Provide **end-users** with a single, personalized point of access to relevant and authoritative information
  - Provide **business organizations** with a unified place to engage, support, learn from and respond to customers and other audiences
  - Provide **IT organizations** with an agile, scalable means to deliver Web applications, an environment to enable collaboration and a means to delegate responsibility to the business

Our Vision: Empower Key Stakeholders



Empower Key Stakeholders: Tools



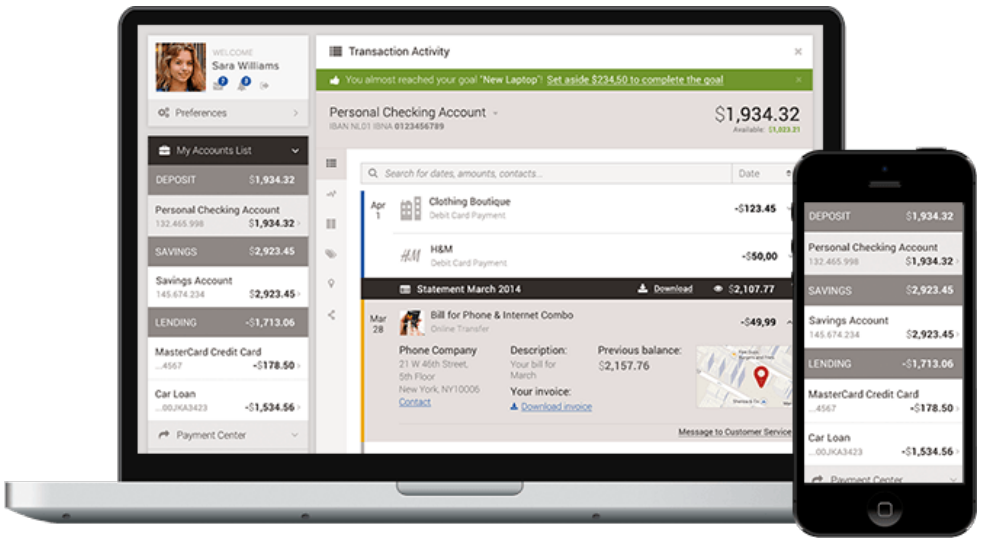
- Existing solutions in the portal market provide sophisticated and versatile products but come with **functional bloat** and **technical complexity**

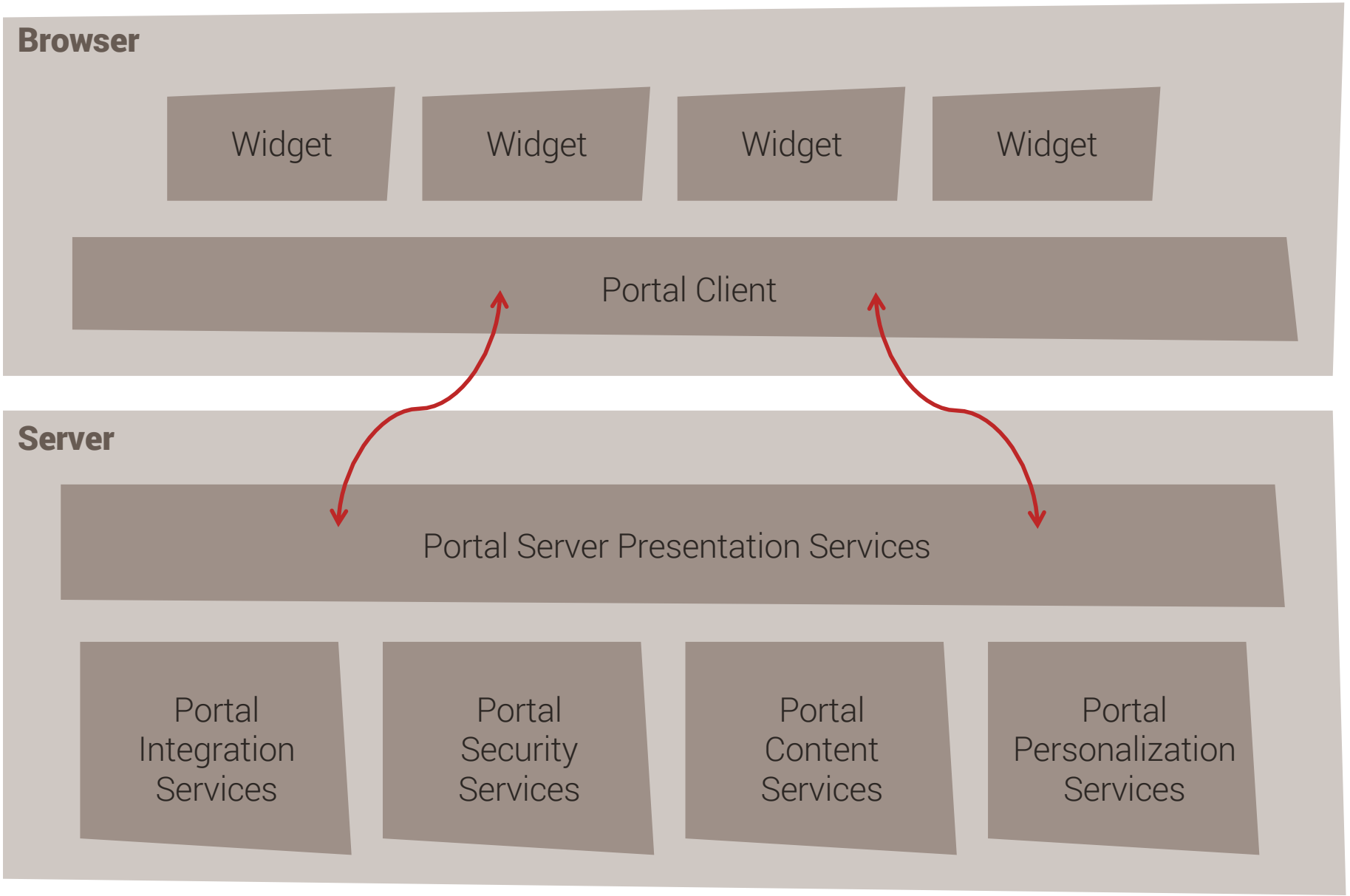
- Companies want lean portals: *simple, more cost-effective*
  - *focus on portal's primary proposition offering a personalized point of access to relevant information, business processes, and people*

- Rely on modern architectures and representational state transfer (REST)-oriented standards for interoperability
  - *bringing customers faster time to value, without violating existing software standards, such as content management and analytics*

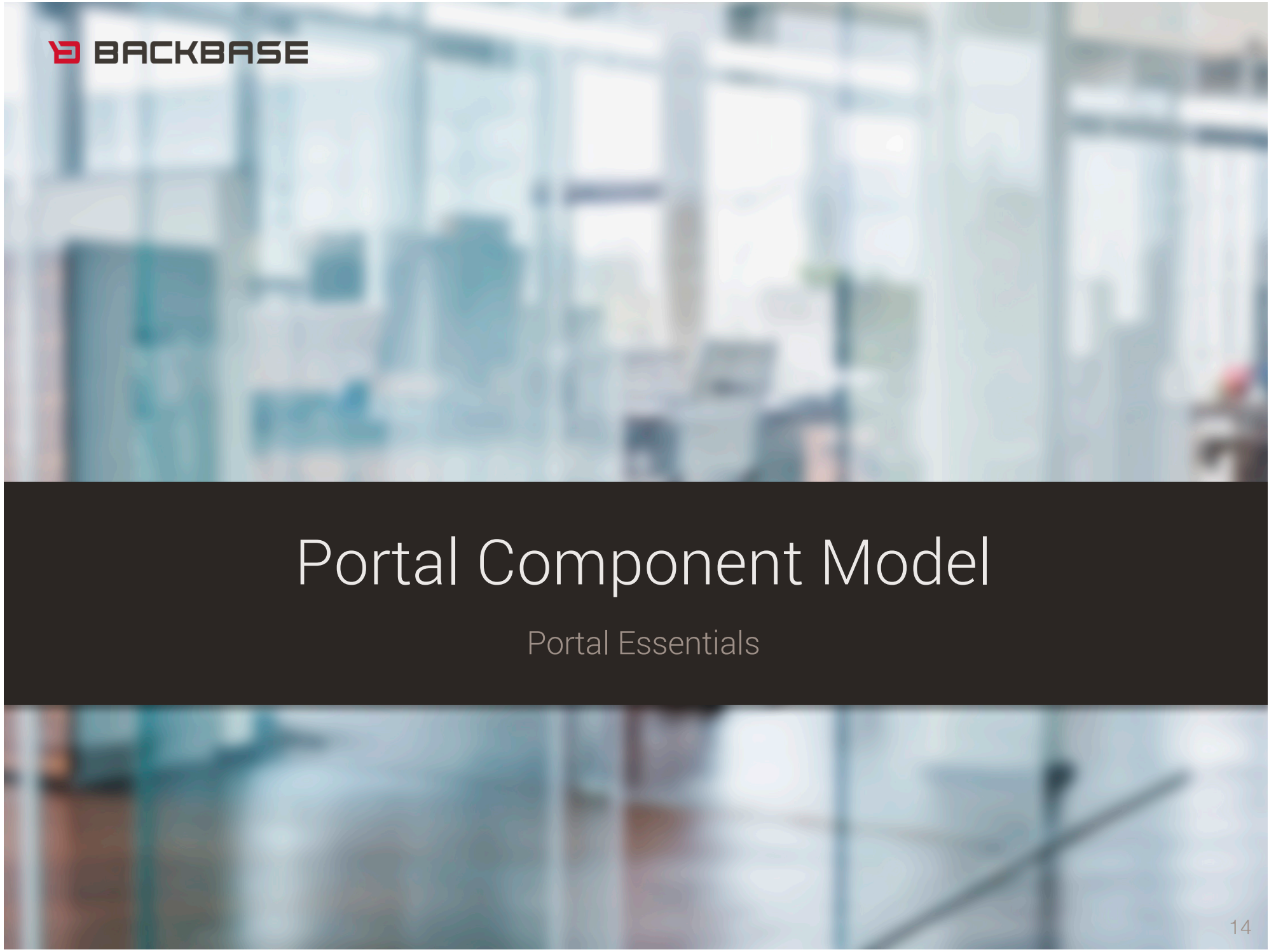
An enterprise web framework that empowers developers, business users, and end users

An aggregation platform with unified experience across these data and content sources




Browser
Widget
Widget
Widget
Widget
Portal Client
Server
Portal Server Presentation Services
Portal Integration Services
Portal Security Services
Portal Content Services
Portal Personalization Services

1. **A component model**
2. Flexible rendering and support for multi-device delivery
3. Personalization
4. The ability to integrate with a wide range of technologies
5. Content services
6. Security administration
7. Targeting
8. Publishing



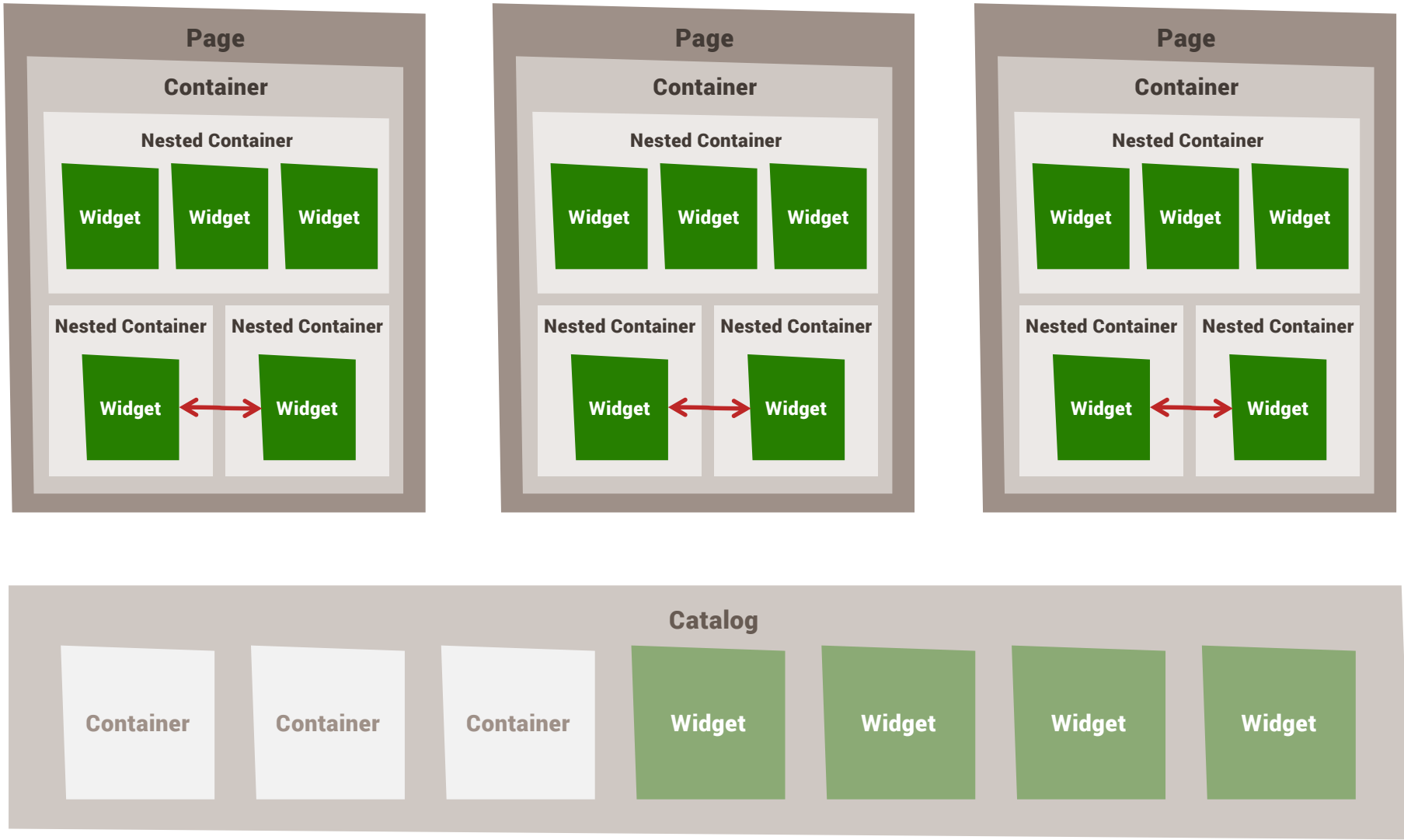
Portal Component Model

Portal Essentials

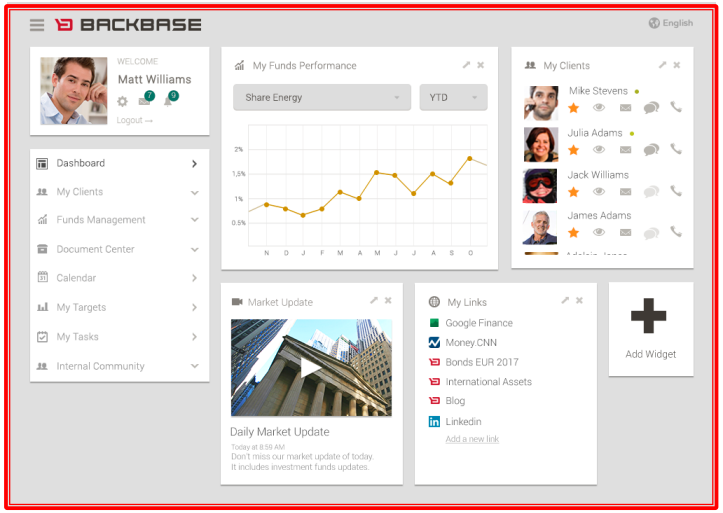
- Model **defines** the structure of the portal
  - Used by Portal to render web pages
  - More abstract way to conceptualize your application
  - Portal model is **stored** in portal database
- Portal model can be **modified**:
  - By **business user,** using **CXP Manager**
  - By **developers**:
    - Using Backbase command line tool
    - **Programmatically**, using APIs (REST, Java, JavaScript)
  - By **end-users** through **personalization**

- **Portals, pages, containers, and widgets** — collectively called items; form the structure of portal websites.
- **Users and groups** — describe authorization for each item.
- **Templates** — determine how pages, containers, and widgets are rendered.
- **Catalogs** — contain abstract items that can be used across the Backbase CXP installation, or in a particular portal.
- **Links** — contain pointers to items, other links, external URLs, or item states.
- **Tags** — store keywords assigned to portals, pages, containers, widgets, or templates.
- **Features** — make shared resources available to items.
- **Structured content types** — define the underlying structure of reusable content items that are stored in Content Services and displayed by the structured content widget.

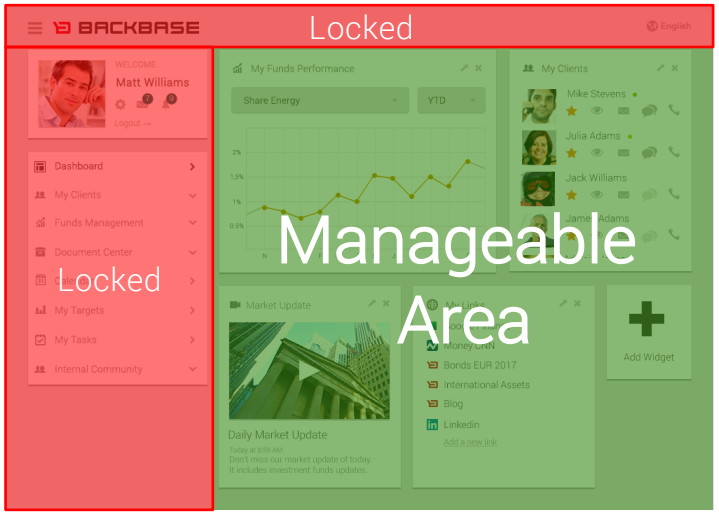
Concepts of CXP Object Model



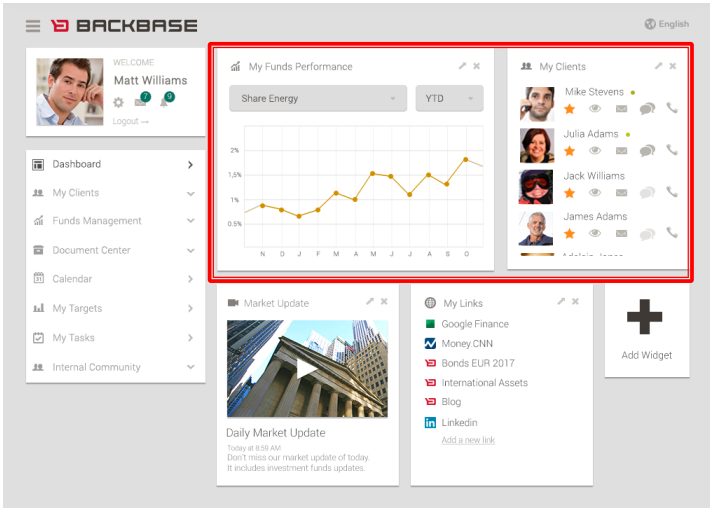
- A portal is composed of one or more pages
- Pages may directly include containers and widgets
- Pages can be of two types:
  - Master pages: blueprint for pages
  - Pages: extend master pages



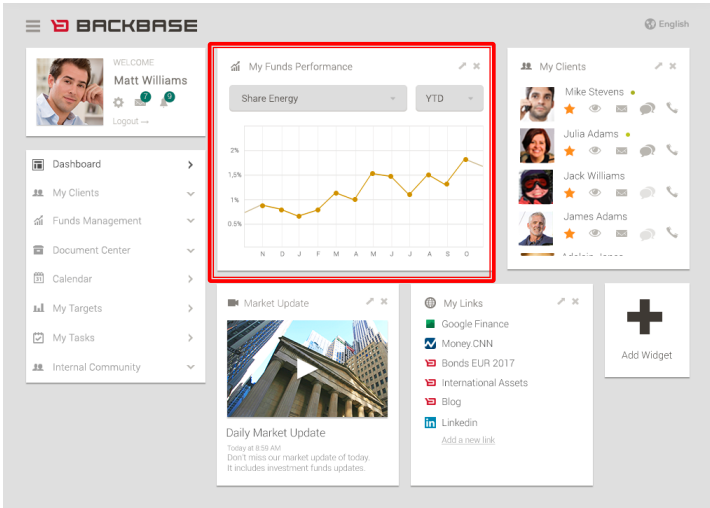
- **Master Page**
  - A blueprint for pages
  - Contains at least one manageable area
  - New pages are created based on a Master Page
    - Areas outside manageable areas are locked for editing
- **Manageable areas**:
  - Available for editing on regular pages.



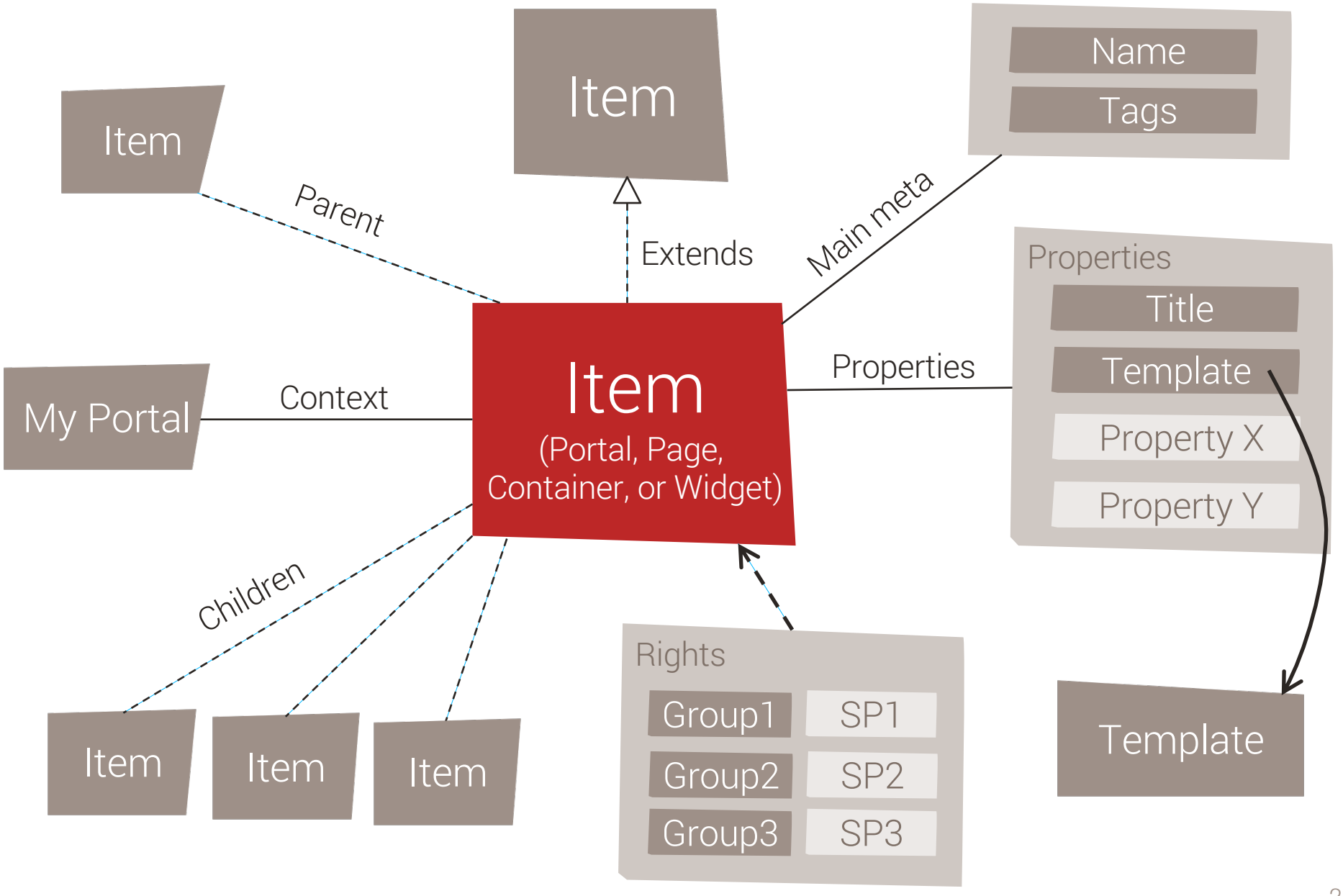
- Containers logically group other container and widgets
- Containers have a layout defining the visual representation of a grouping

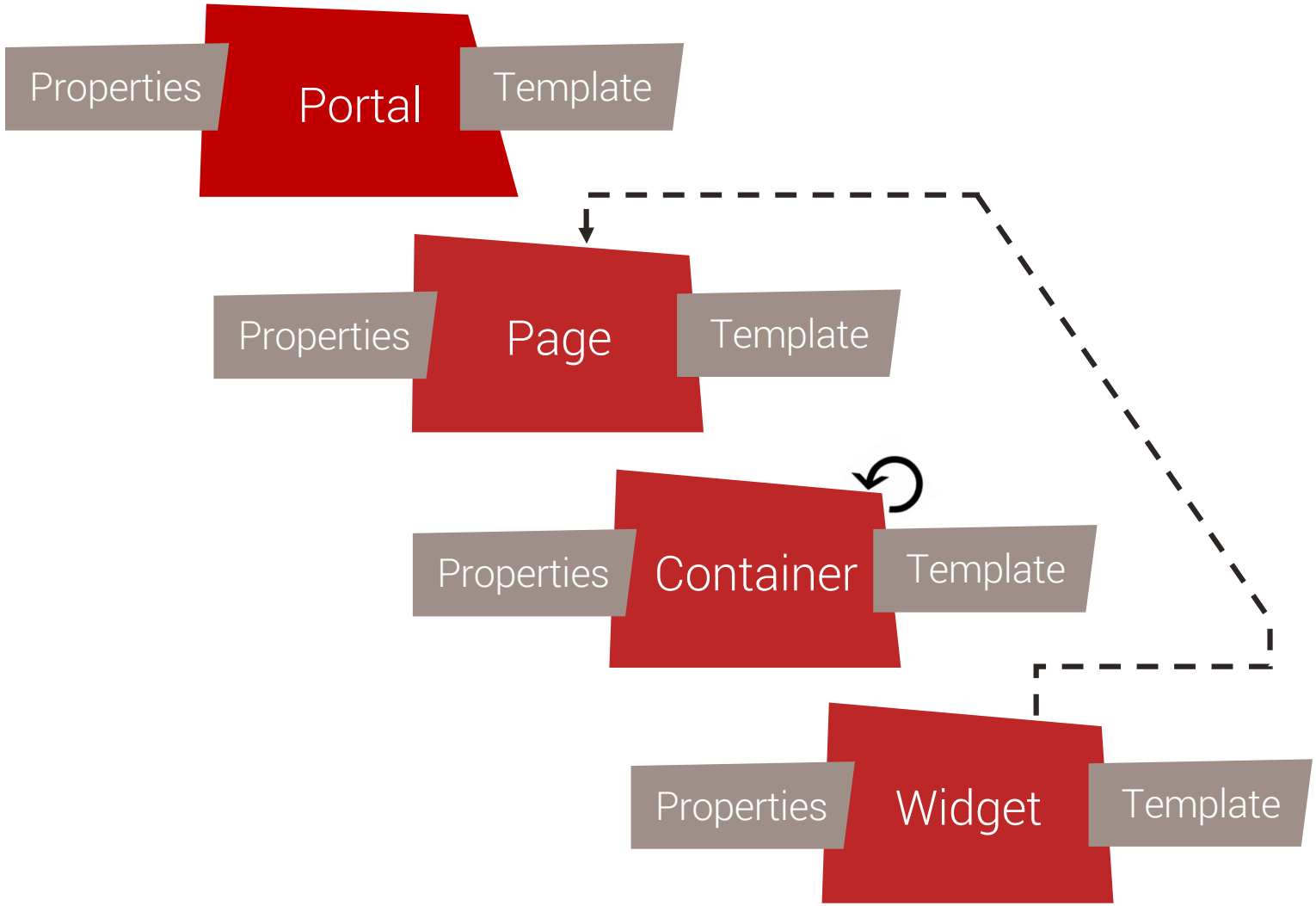


- Widgets are the core of a portal's functionality
- Autonomous mini-applications
- Cannot include containers or widgets

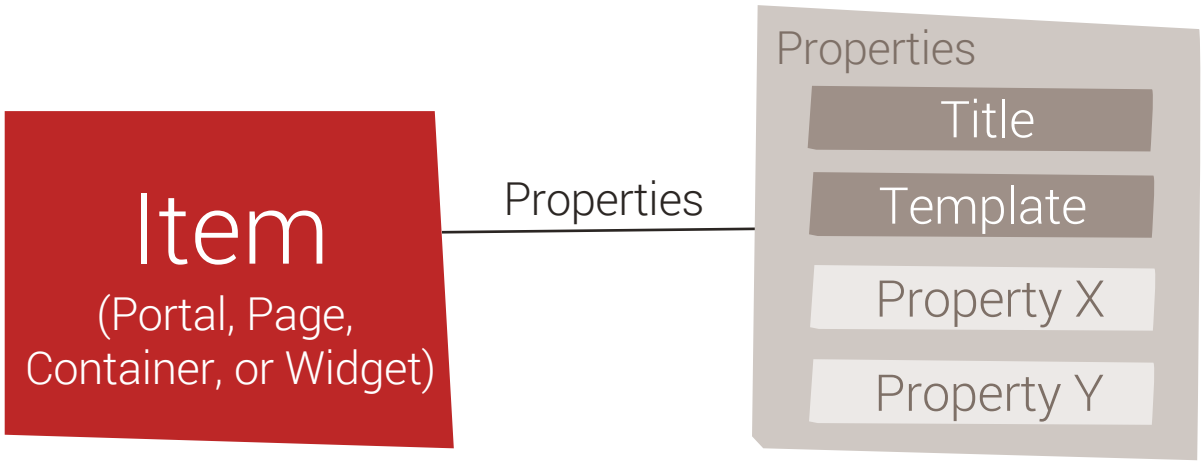


An Item


Item
Item
Parent
Extends
Name
Tags
Main meta
Properties
Title
Template
Property X
Property Y
Properties
My Portal
Context
Item
(Portal, Page,
Container, or Widget)
Children
Item
Item
Item
Rights
Group1
SP1
Group2
SP2
Group3
SP3
Template


Properties
Portal
Template
Properties
Page
Template
Properties
Container
Template
Properties
Widget
Template

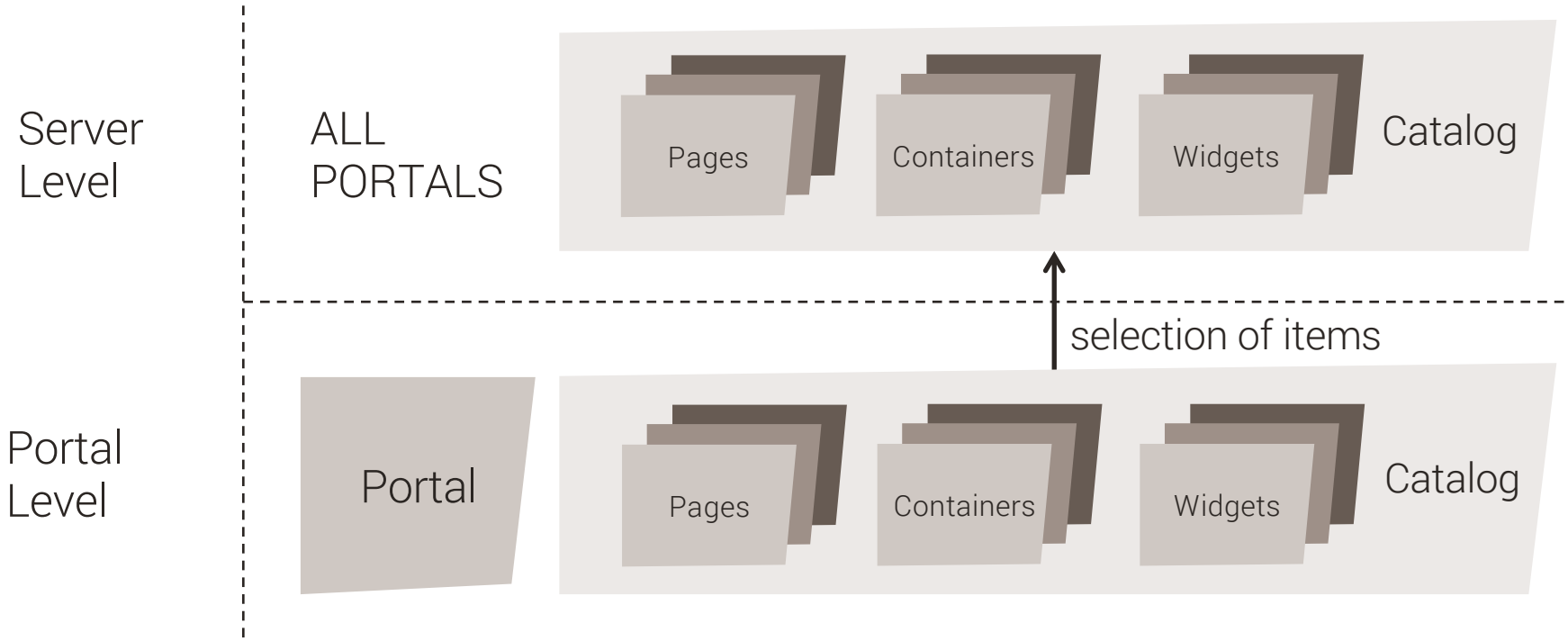
- An item can have **properties** (meta-information)
  - Name/value pairs
  - Typed (e.g. double, integer, string, ..)
- Every item type gets a set of **default properties**
  - e.g. default landing page
- An item can have optional **custom properties**
- A user (depending on permissions) can **set his own value** to an item property (**customization** at the user scope)



- Deletions in Backbase CXP always cascade: children are deleted as well
  - Deleting a page deletes its containers and widgets.
  - Deletions in CXP cannot be undone

Let’s look at how the CXP Manager handles the concepts we have just discussed

- Server Level – Enterprise Catalog (global space BBHOST)
- Portal Level – Portal Catalog



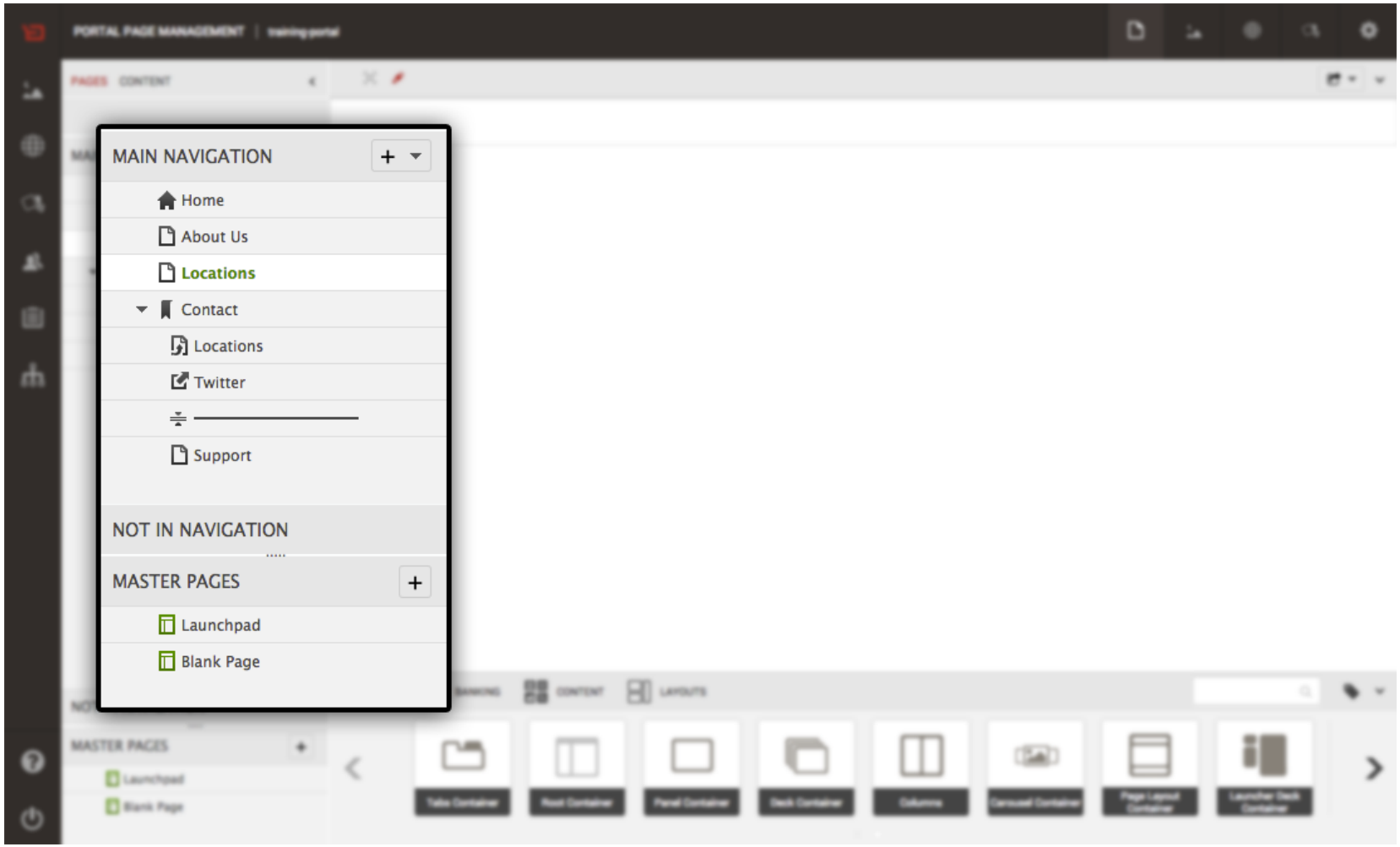
- Rules of inheriting *properties*:
  - Items in a portal catalog inherit properties from the abstract item in the enterprise catalog, unless someone explicitly sets a property.
  - Items instantiated in a portal inherit properties from the abstract item in the portal catalog, unless someone explicitly sets a property.
- Rules of inheriting *rights*:
  - Abstract items in the enterprise catalog inherit rights from Portal Foundation.
  - Abstract items in a portal catalog inherit rights from the portal.
  - Instantiated items inherit rights from their parent (either a portal, a page, or a container).

Let’s look at how the CXP Manager handles the concepts we have just discussed

- The link item stores pointers to other items or external URLs
- Navigation widgets use links to build navigation trees

Type	Description
Page	Refers to a page
External link	Refers to a URL outside CXP
Divider	Displays a divider in the navigation tree
Header	Displays a menu header in the navigation tree. Does not refer to a page itself, but can have children that do
Alias	Refers to a page that already has an associated link

- CXP Manager creates a link for every new page and creates menus as shown in the image below:



- Root links are menu headers that divide the navigation tree into smaller sub-trees
- Root links:
  - **Main navigation** — contains links that should show up in navigation widgets.
  - **Not in navigation** — contains links to floating pages or pages that do not show up in navigation widgets. Floating pages are, for example, landing pages for a marketing campaign.
  - **Master Pages** — contains links that refer to the master pages available in the portal.
- Root links do not point to pages, but to other links

- Link URL types:
  - Hierarchical: automatically generated, human-readable addresses pointing to links
  - Friendly: user-defined and easy-to-remember URL pointing to links
  - External: point to resources outside of a portal
- Dynamic URLs
  - Can change the state of containers and widgets on a page
  - Contain dynamic URL parts that containers and widgets can use to change their behavior
  - The dynamic URL parts appear at the end of the normal URL after the identifier **//**
  - `http://example.com/myportal/mypage//some/values/here`

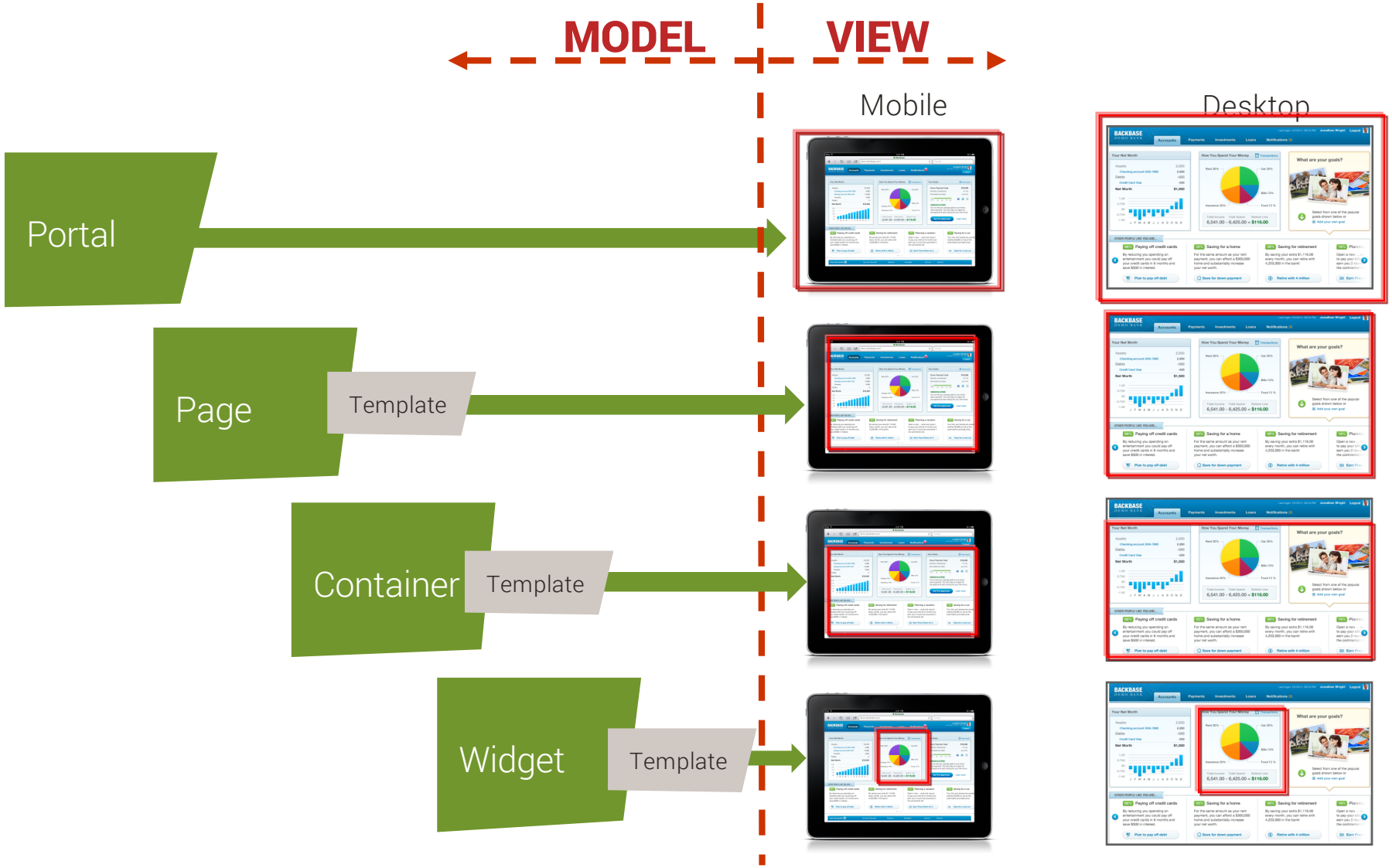
- Deleting a link also deletes the referenced page
- If other links refer to the same page, the following applies:
  - If another link refers to the same page, the page is not deleted

Let's look at how the CXP Manager handles the concepts we have just discussed

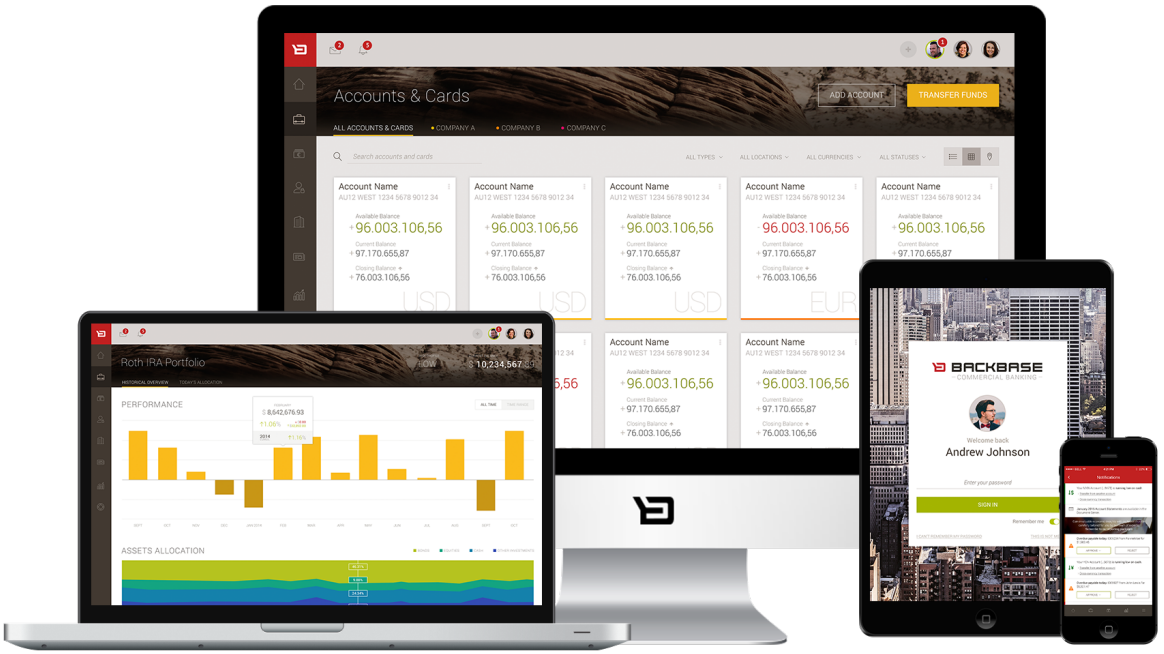
1. A component model
2. **Flexible rendering and support for multi-device delivery**
3. Personalization
4. The ability to integrate with a wide range of technologies
5. Content services
6. Security administration
7. Targeting
8. Publishing

Rendering the Model

Portal Essentials


MODEL
VIEW
Mobile
Desktop
Portal
Page
Template
Container
Template
Widget
Template

- *Responsive design* is preferred over separate templates
- For special situations, you can define specific templates to render a Portal Item
- Each main device group can have its own template
  - Desktop (default)
  - Tablet
  - Mobile

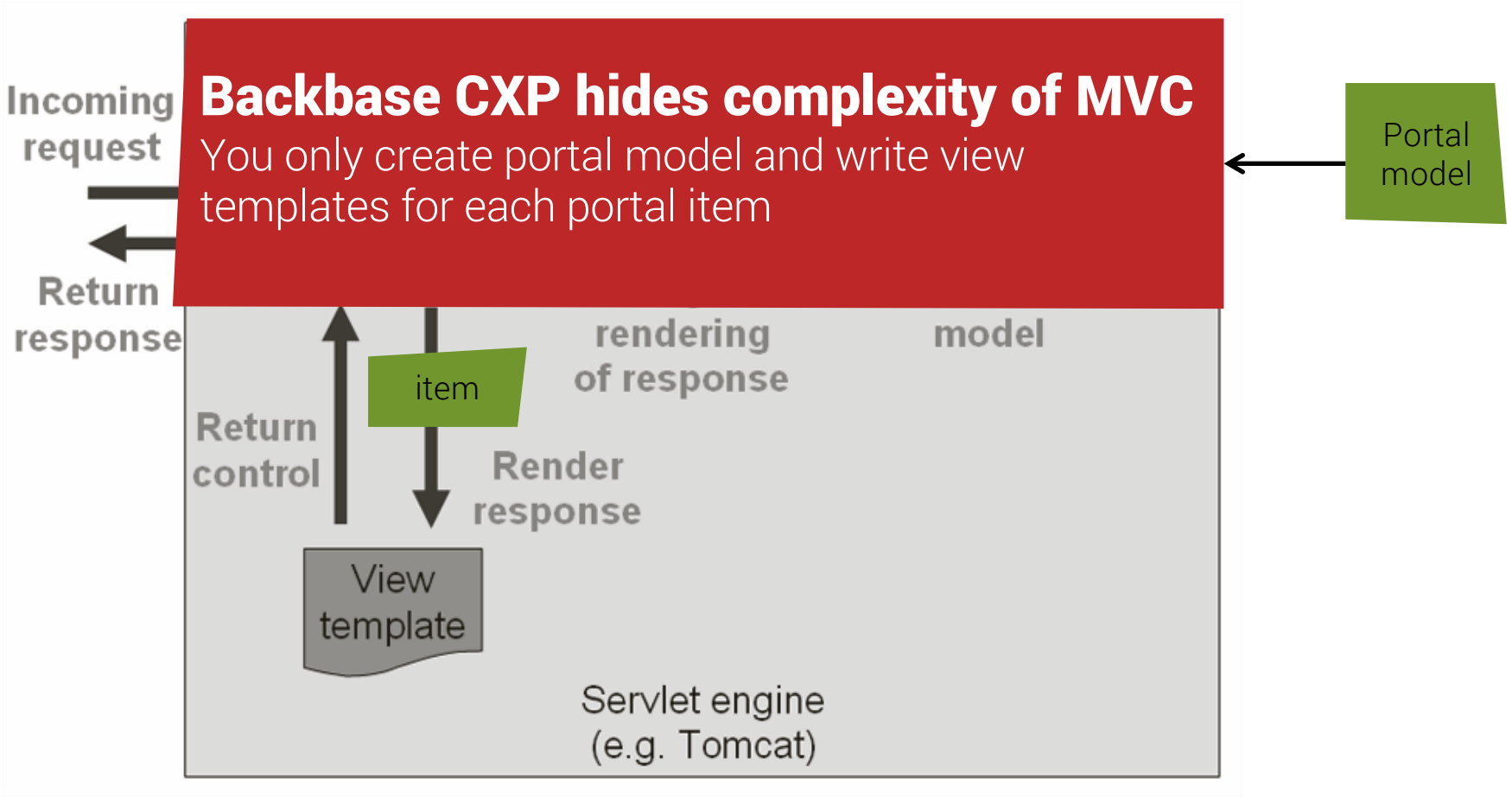


- Model/view/controller (MVC) is an Architectural Pattern
  - Isolates "domain logic" (the application logic for the user) from the user interface (input and presentation)
  - permitting independent development, testing and maintenance of each (separation of concerns).

- Use of the Model/View/Controller (MVC) pattern results in applications that
  - separate the different aspects of the application (input logic, business logic, and UI logic)
  - while providing a loose coupling between these elements

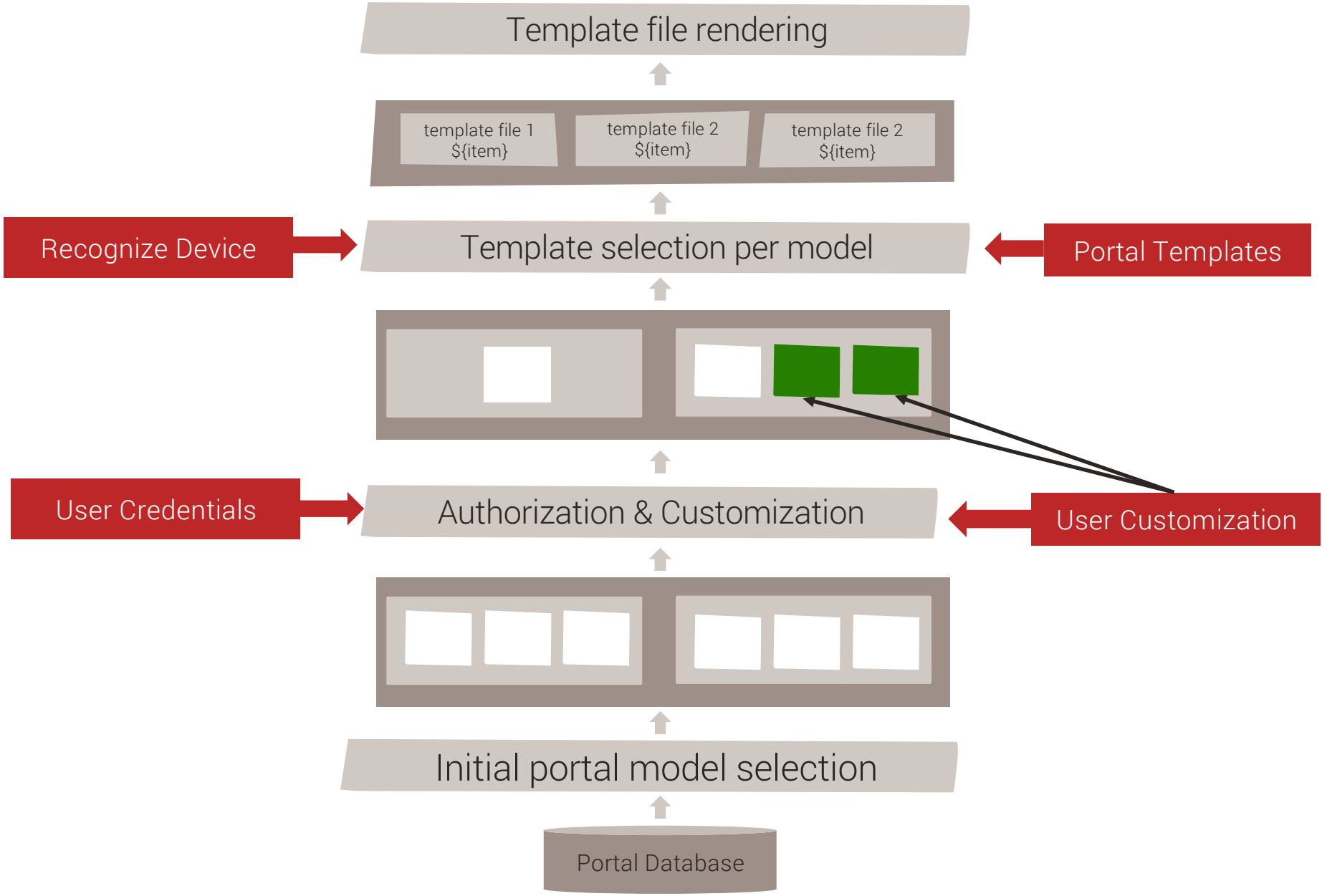
- CXP uses the Spring MVC framework under the hood and implements the model-view-controller pattern
  - Items belonging to the Portal Model are stored in the data store
  - Rendering of the model happens either client-side or server-side
  - Controllers are internal part of Portal Server, you do not work with them directly

SPRING MVC

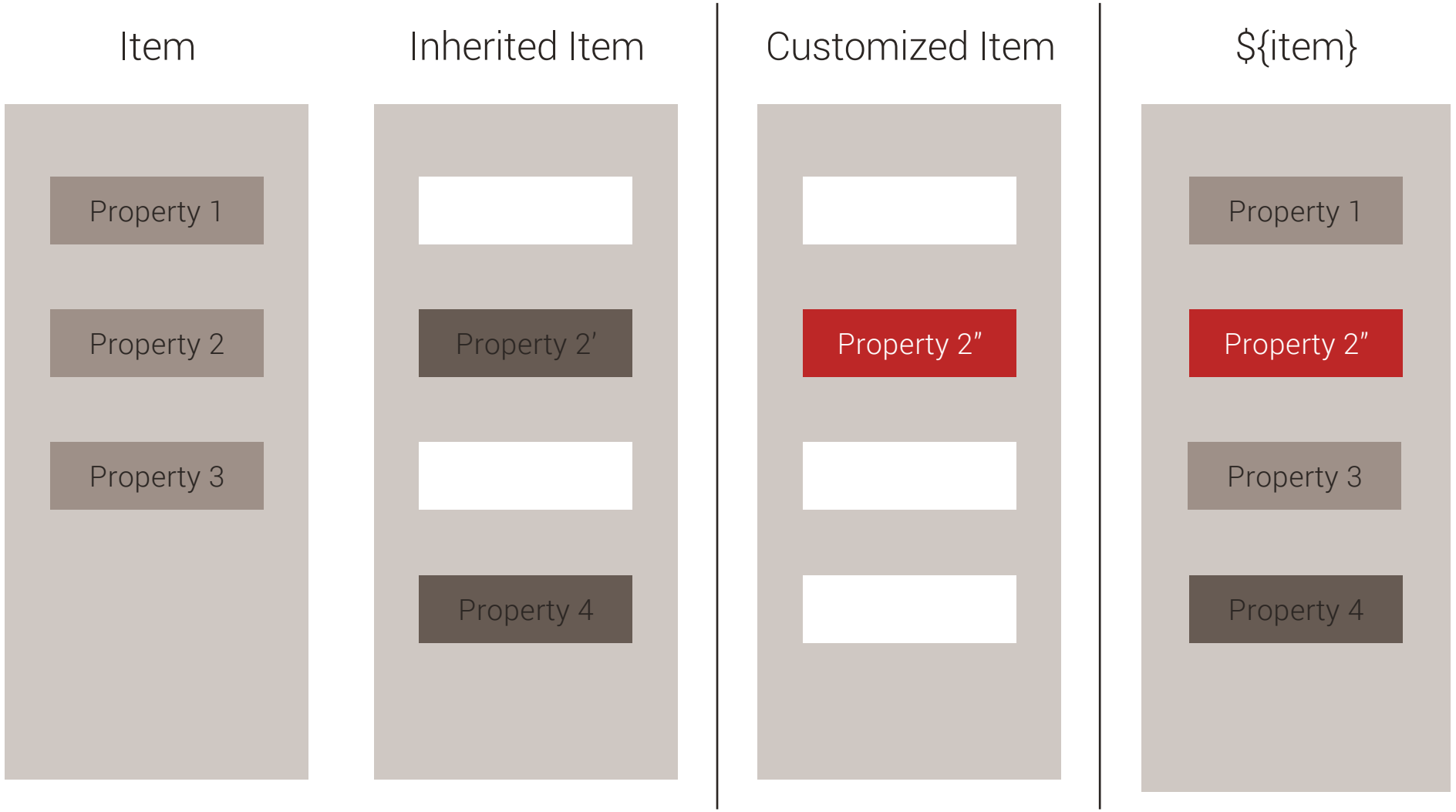

Backbase CXP hides complexity of MVC
You only create portal model and write view templates for each portal item
Incoming request
Return response
Portal model
rendering of response
model
item
Return control
Render response
View template
Servlet engine (e.g. Tomcat)

- Model preparation
  - Security settings
  - Inheritance and customization
- Template selection
  - Based on recognized device
- Template Execution
  - Portal server calls selected templates and injects ${item} variable
  - ${item} contains data about the item and its children

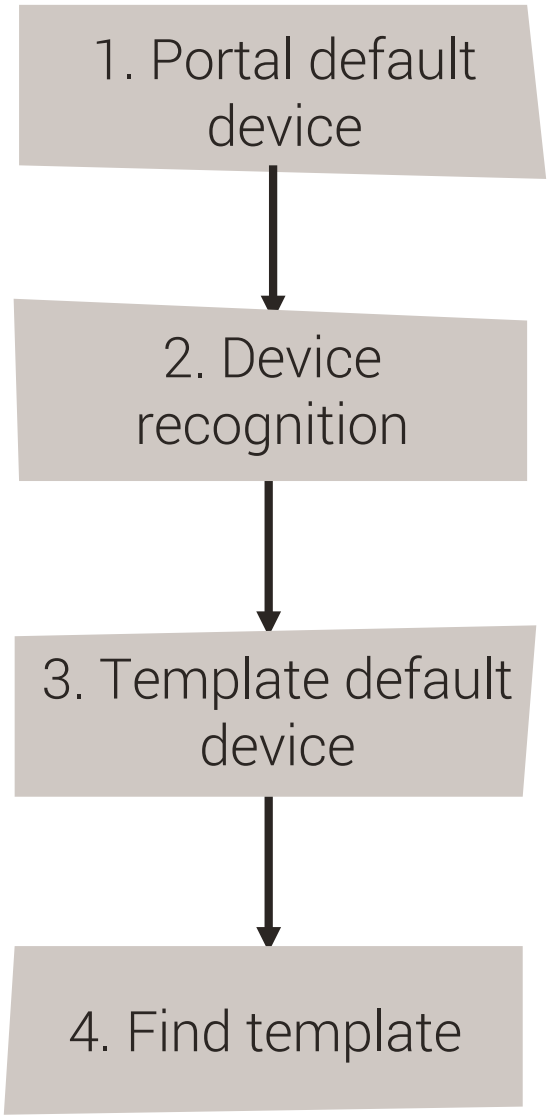
Portal Rendering Flow



Inheritance and Personalization / Properties



Template Selection Process

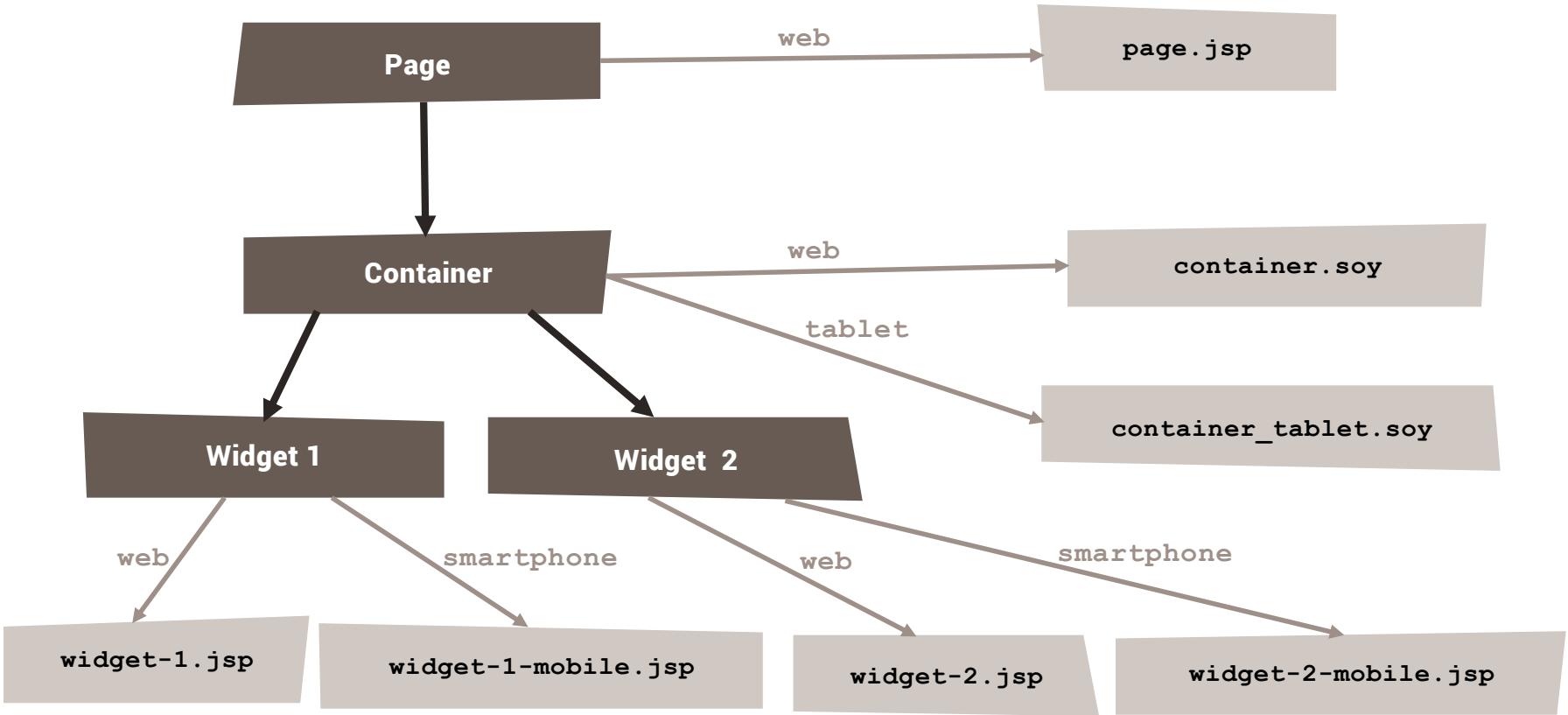


- Portal Foundation first checks if the DefaultDevice portal property is set. If it is, Portal Foundation looks for a template property with a matching name.
- Portal Foundation tries to match the User-Agent header of the request against the device recognition configuration.
- Portal Foundation looks for a property named DefaultDevice in the template object. DefaultDevice currently only accepts a value of Web: Portal Foundation looks for a template property named Web:
- Once a template property matches, Portal Foundation tries to find the template file with the name specified in the matching property.

- Device recognition:
  - If the User-Agent header matches a device, the device configuration specifies a device type and a fallback device type

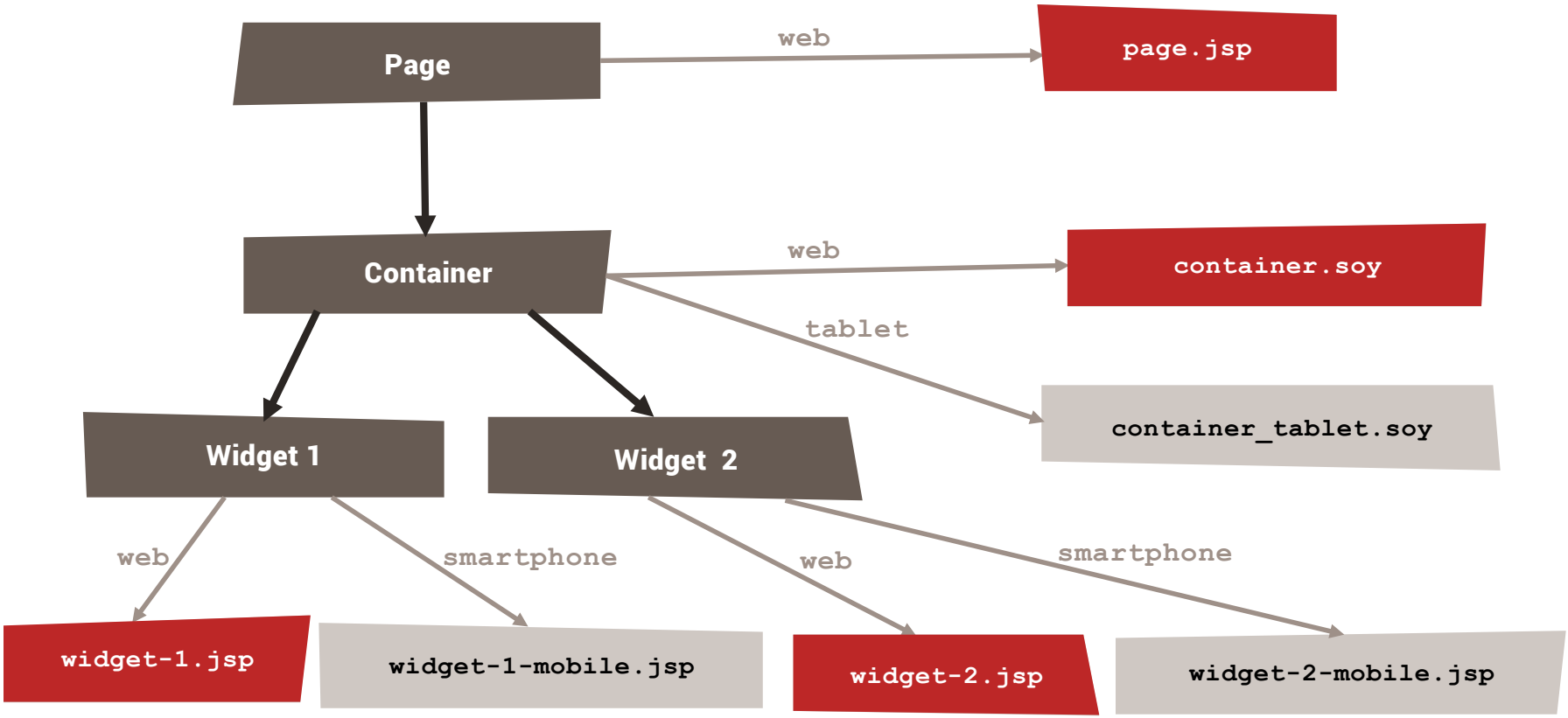
```
<device type="iPhone" fallbackTypes="Smartphone">
```

- Device type:
  - Portal Foundation looks for a template property with name matching the device type
- Fall-back device type:
  - Portal Foundation looks for a template property with name matching the fallback device type

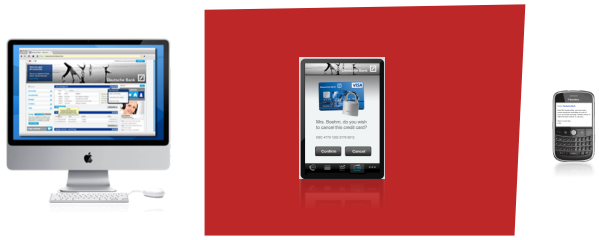

Page
web
page.jsp
Container
web
container.soy
tablet
container_tablet.soy
Widget 1
Widget 2
web
smartphone
web
smartphone
widget-1.jsp
widget-1-mobile.jsp
widget-2.jsp
widget-2-mobile.jsp

Template Selection: Desktop (web)

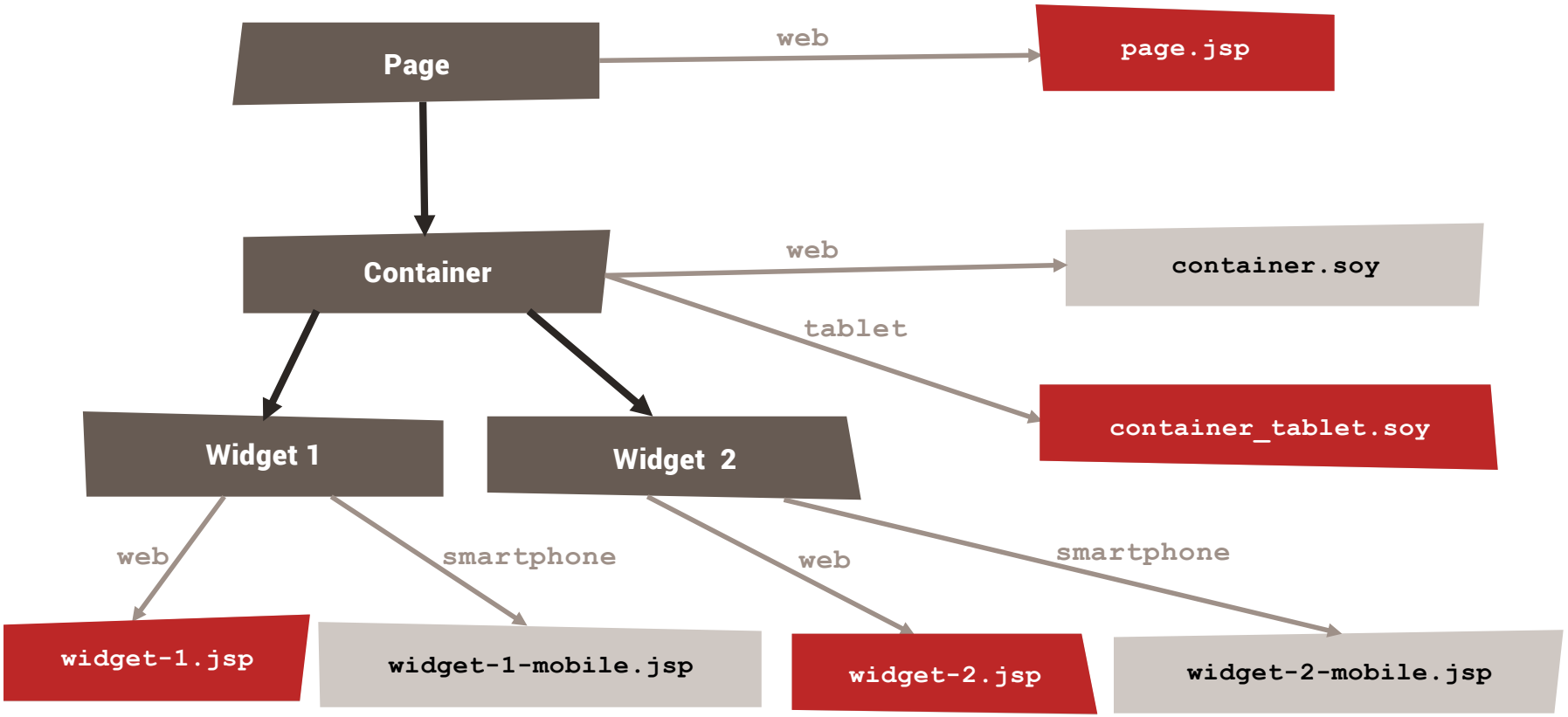
(default = web)



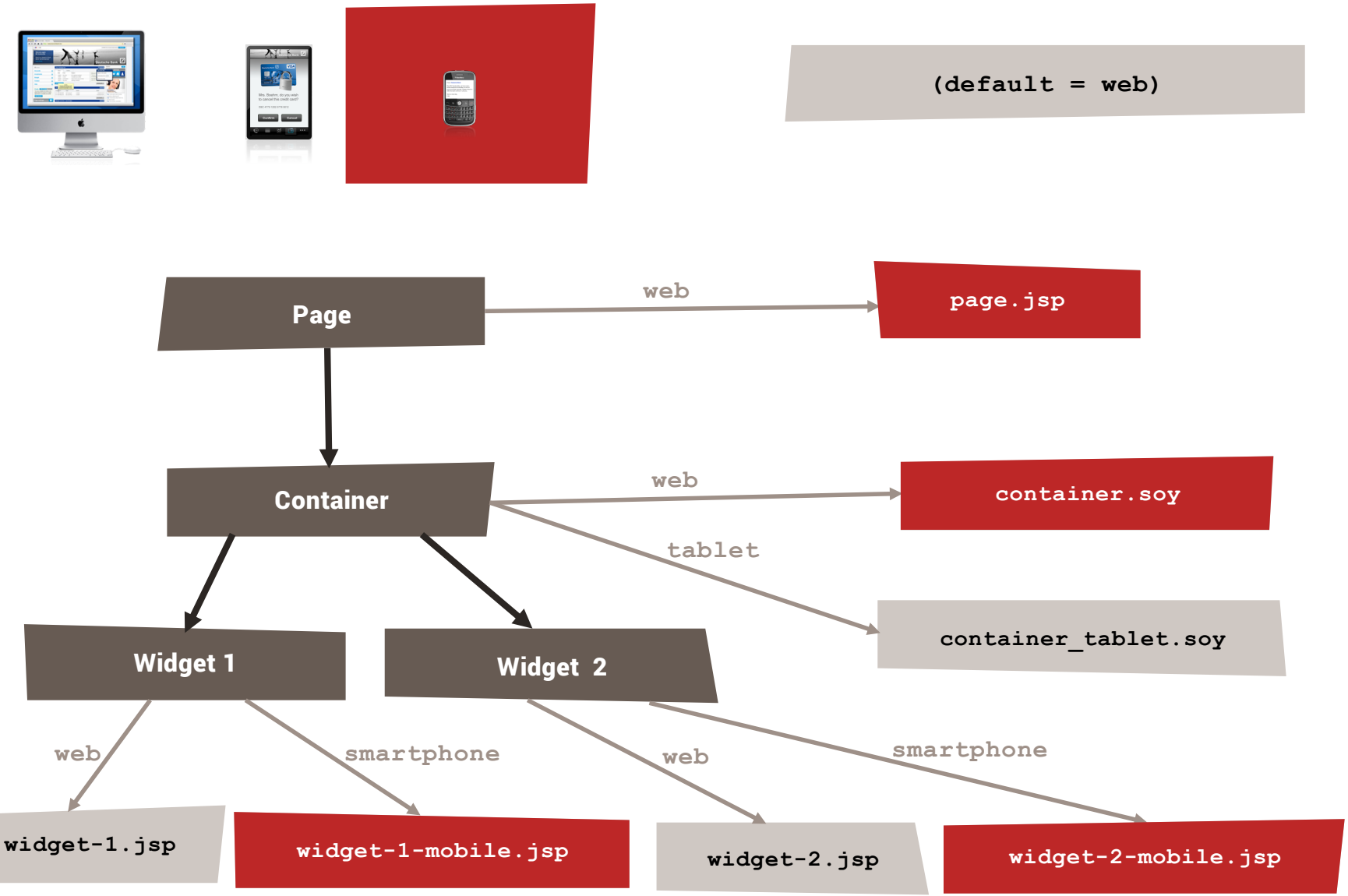
Template Selection: iPad (tablet)



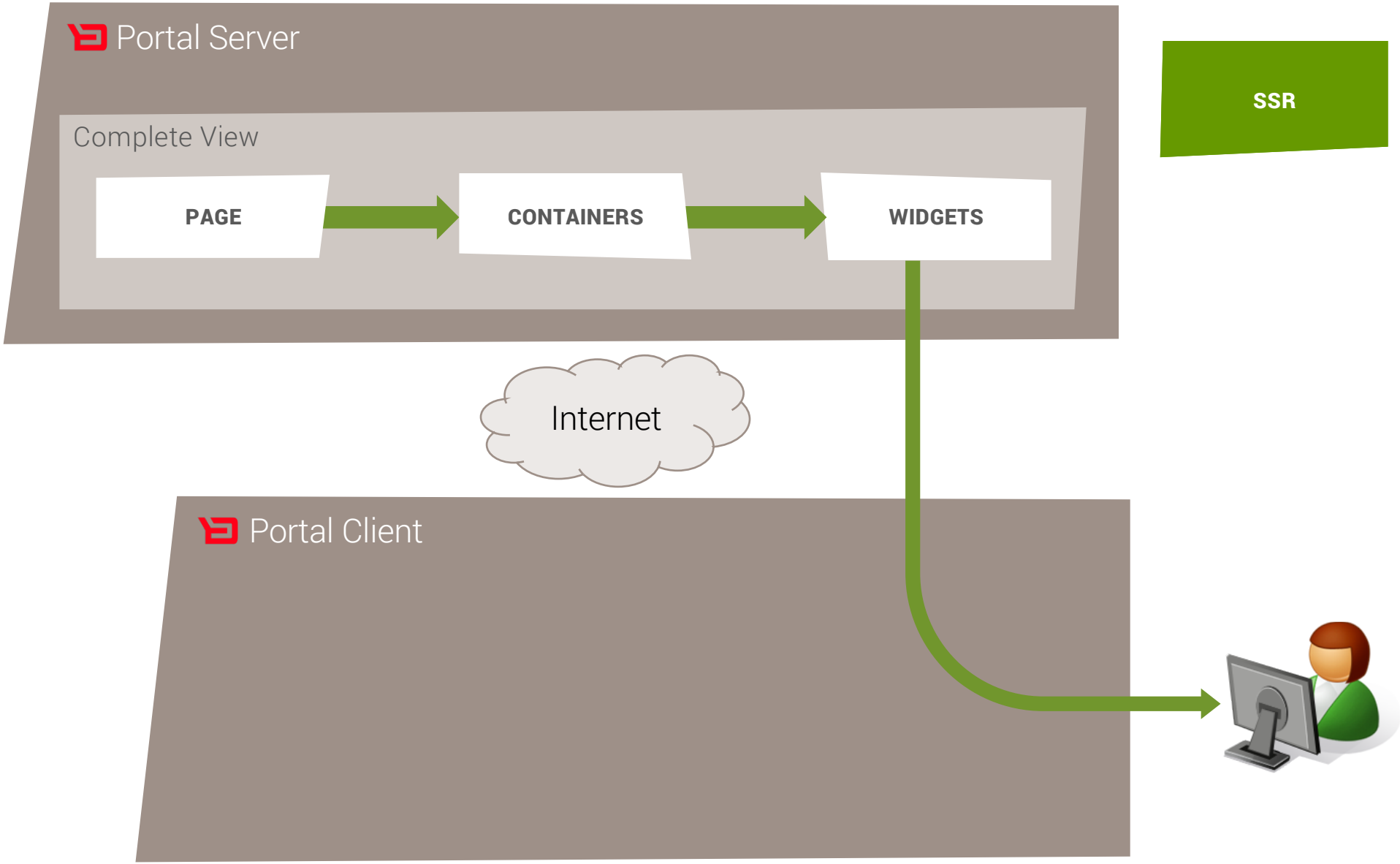
(default = web)

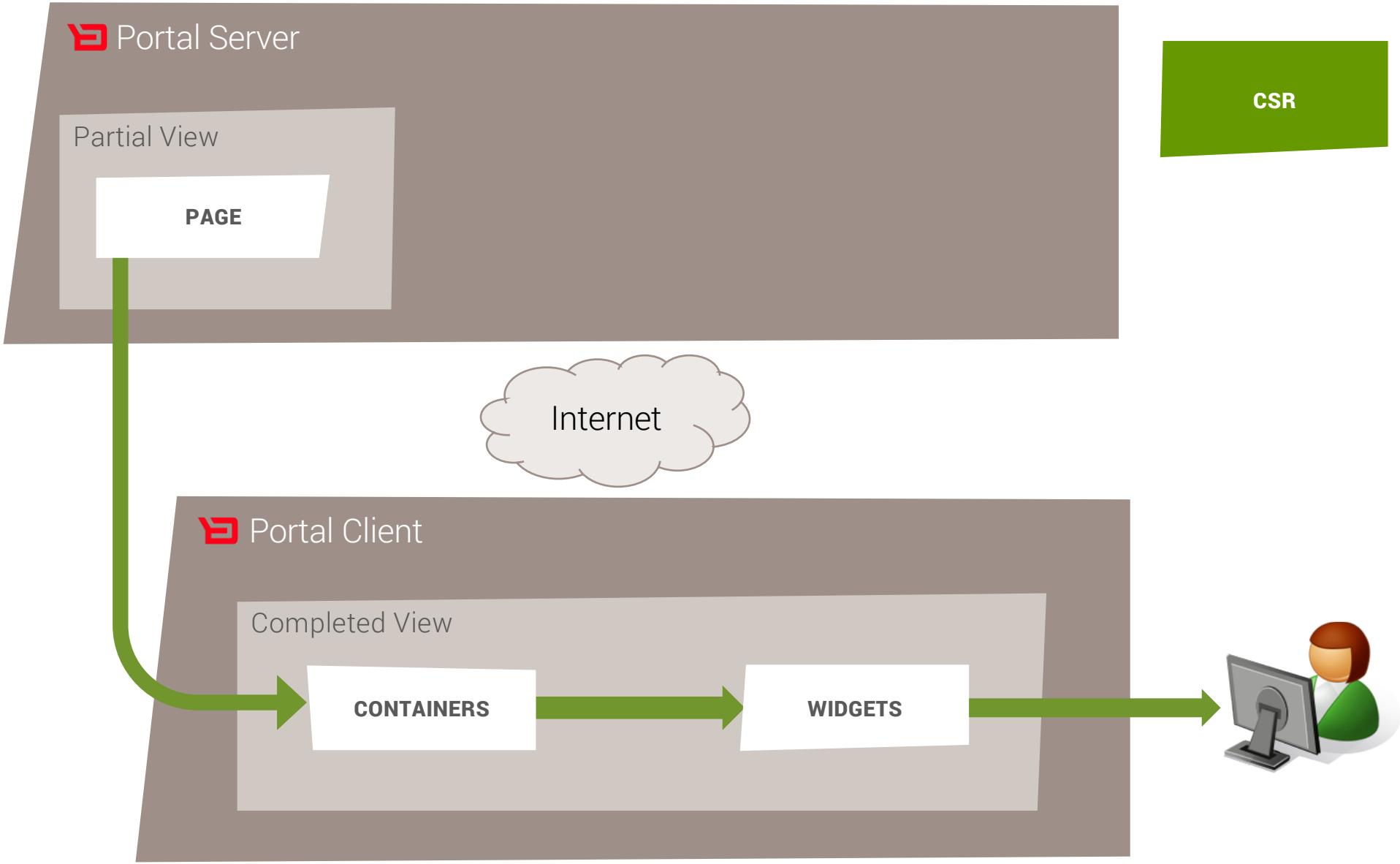


Template Selection: Android (smartphone)



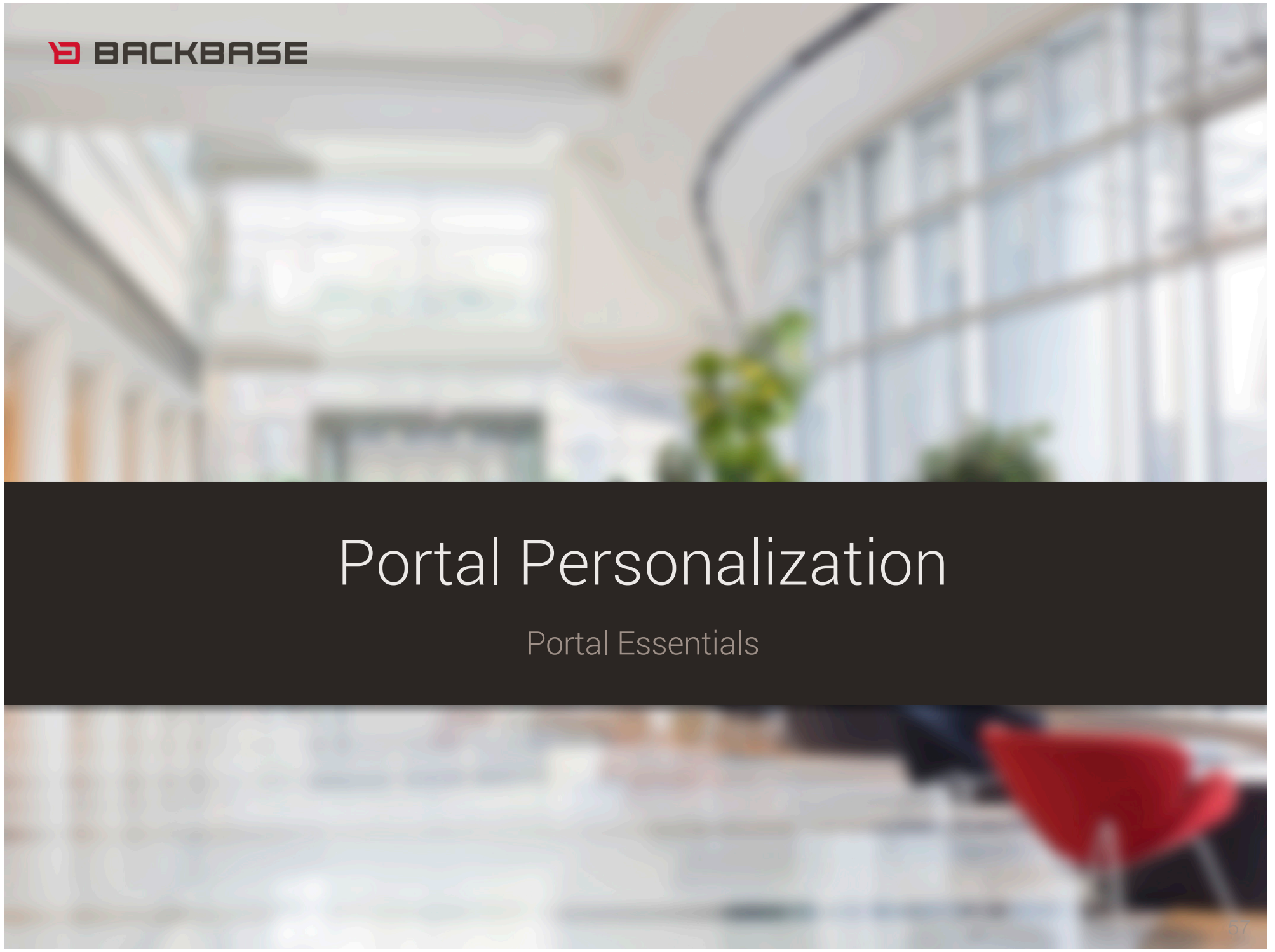
- Each item can have separate templates for server-side rendering and client-side rendering
- **Server-side rendering**
  - Viewable content is completely generated at the server-side, browser can directly render it
- **Client-side rendering**
  - Server does not create viewable content but sends to the client data to be rendered
  - Rendering is done by client Javascript code that directly updates client page DOM


Portal Server
Complete View
PAGE
CONTAINERS
WIDGETS
SSR
Internet
Portal Client


Portal Server
Partial View
PAGE
CSR
Internet
Portal Client
Completed View
CONTAINERS
WIDGETS

- Templates of nodes can be implemented with different templating technologies
- Pages and Containers:
  - Google Closure — extension is soy
  - Java Server Pages — extension is JSP
- Widgets:
  - JSP, Mustache

1. A component model
2. Flexible rendering and support for multi-device delivery
3. **Personalization**
4. The ability to integrate with a wide range of technologies
5. Content services
6. Security administration
7. Targeting
8. Publishing

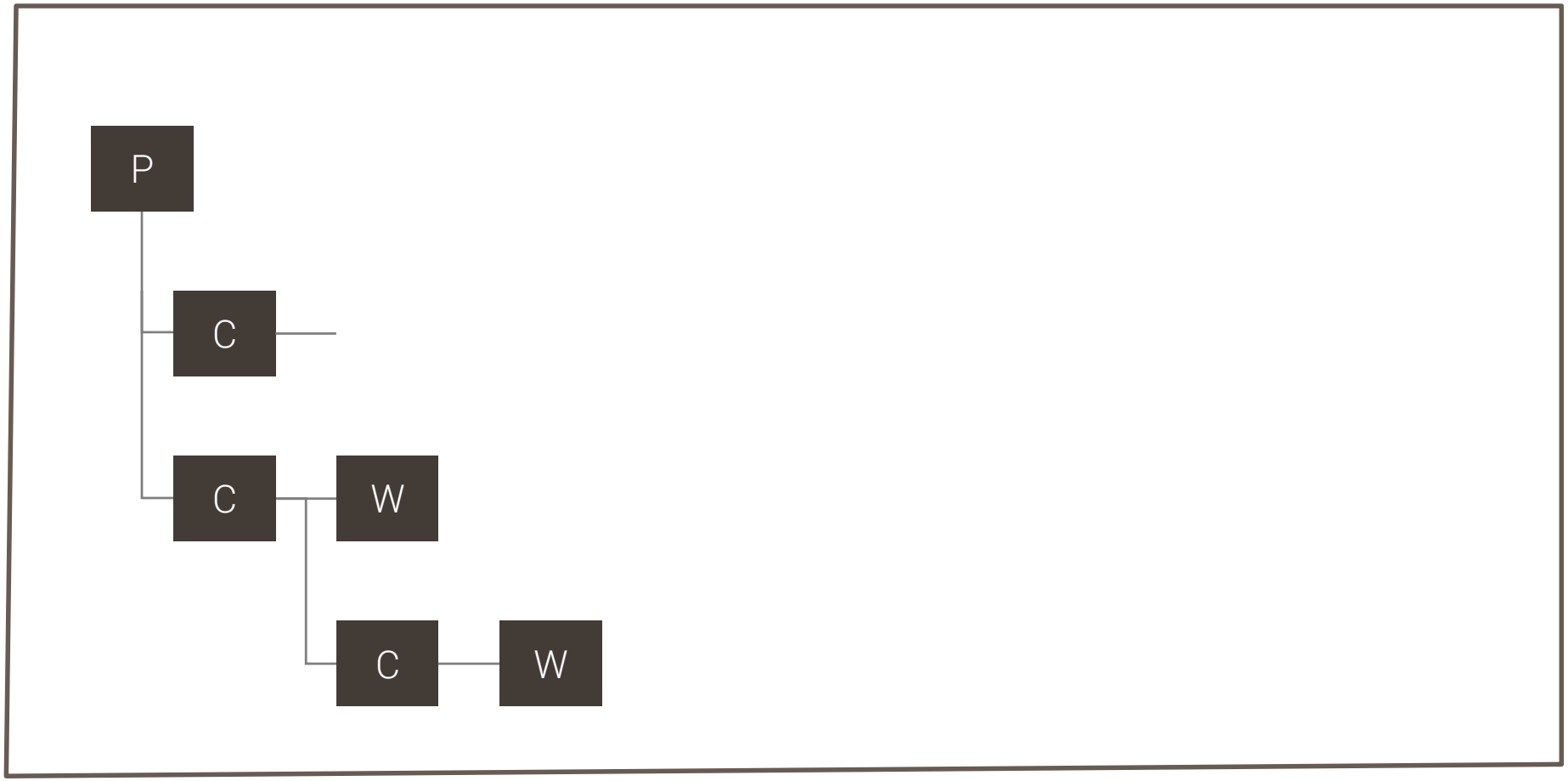


Portal Personalization

Portal Essentials

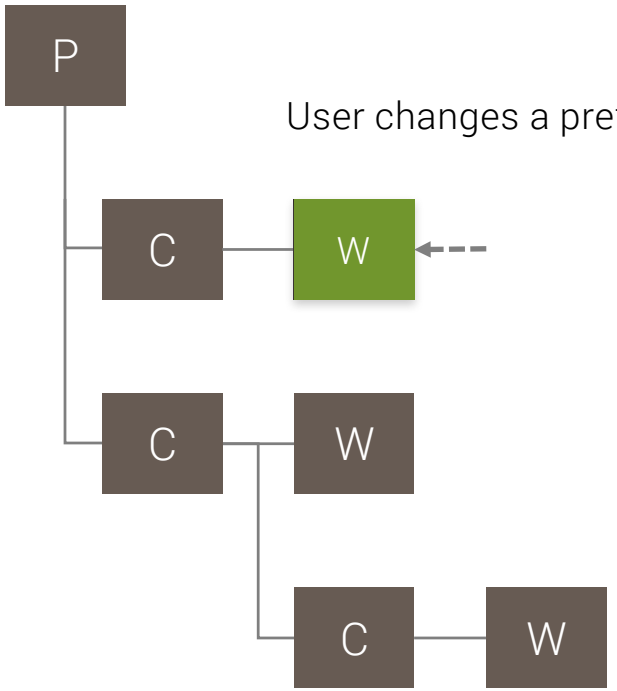
- Personalization - the process of tailoring applications to individual users' characteristics or preferences
- Users can change Widget properties
- User space
  - the space of properties that user can change (not all properties are available in user space)
  - stored separately for each user (each user has its own space)

W W W Catalog



Personalization: Extending Global Model User Model

W W W Base Elements (eg. Widgets/Templates)



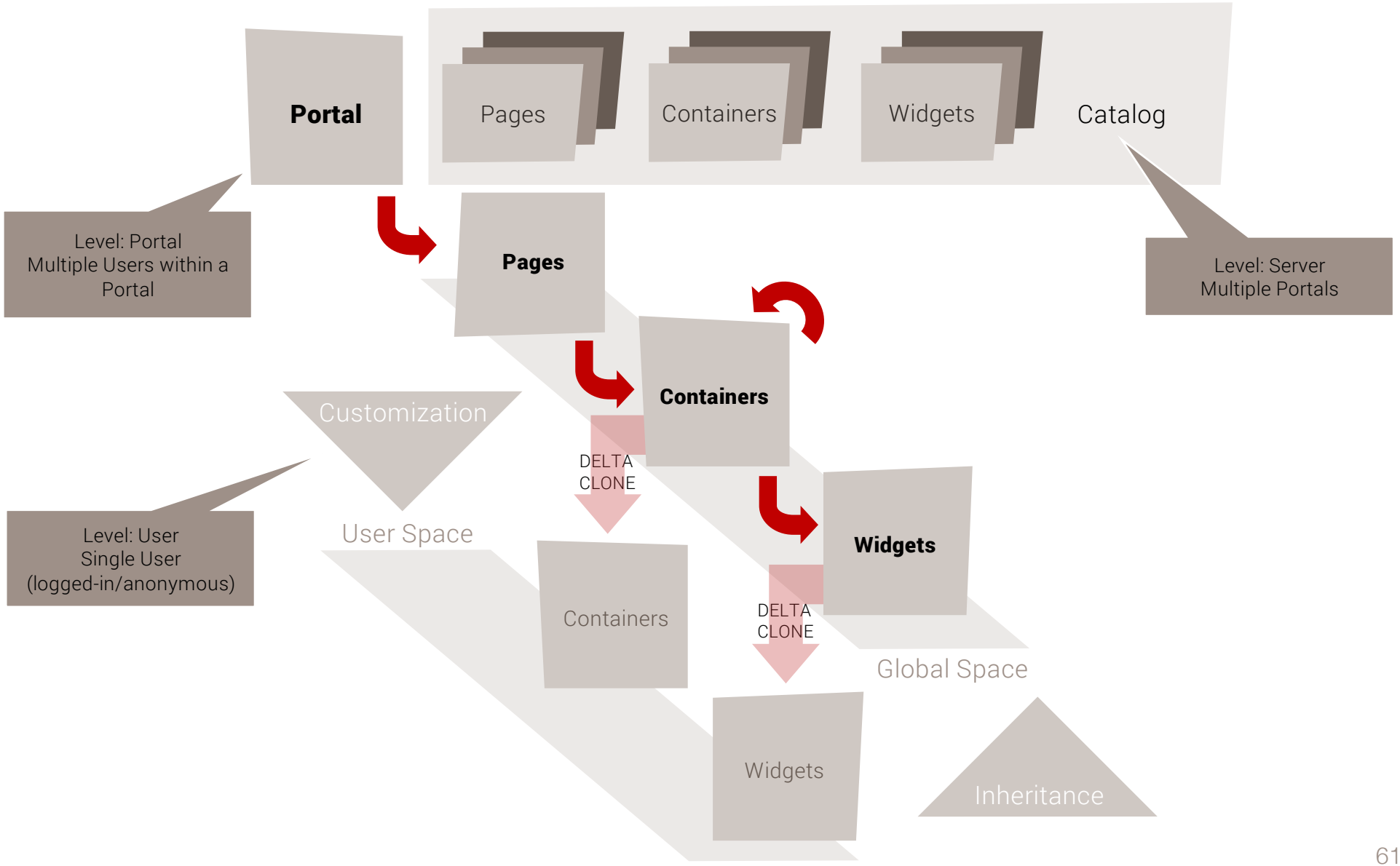
Widget is extended with the changed preference

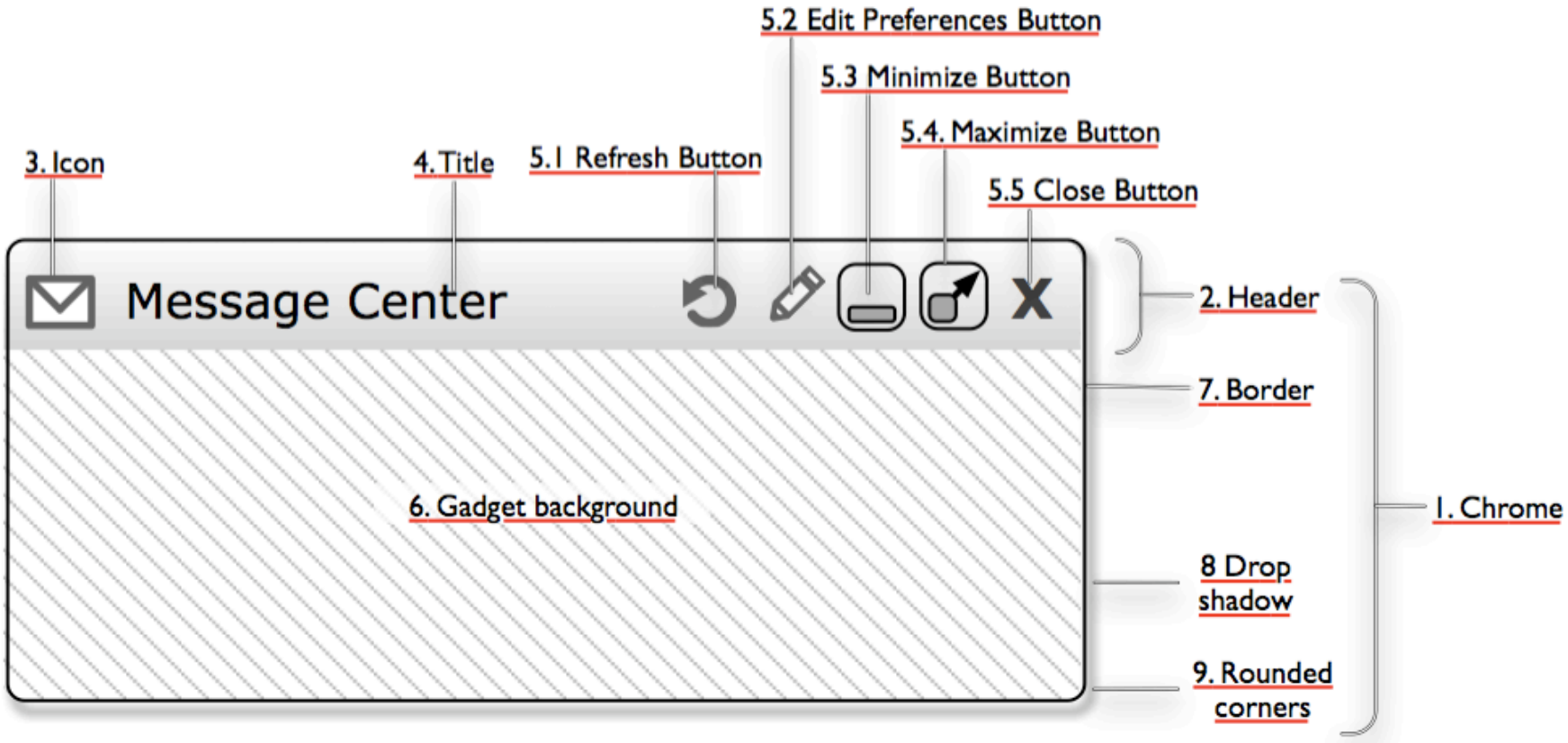
User changes a preference

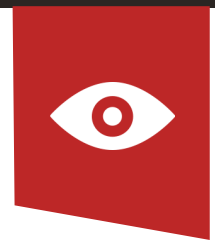
The user model is therefore the delta of the Global model

“Cascading” resolving of properties values

Portal Definition Levels




5.2 Edit Preferences Button
5.3 Minimize Button
5.4. Maximize Button
5.5 Close Button
3. Icon
4. Title
5.1 Refresh Button
Message Center
2. Header
7. Border
6. Gadget background
1. Chrome
8 Drop shadow
9. Rounded corners



Lets look at how the CXP Manager handles the concepts we have just discussed

1. A component model
2. Flexible rendering and support for multi-device delivery
3. Personalization
4. **The ability to integrate with a wide range of technologies**
5. Content services
6. Security administration
7. Targeting
8. Publishing

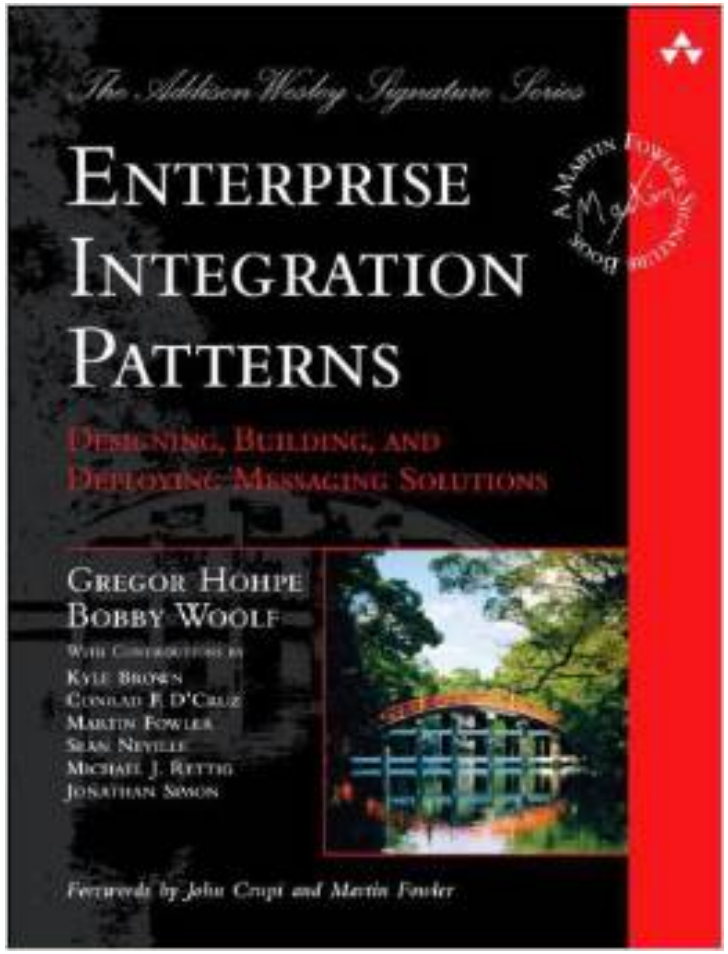
Enterprise Integration with Camel

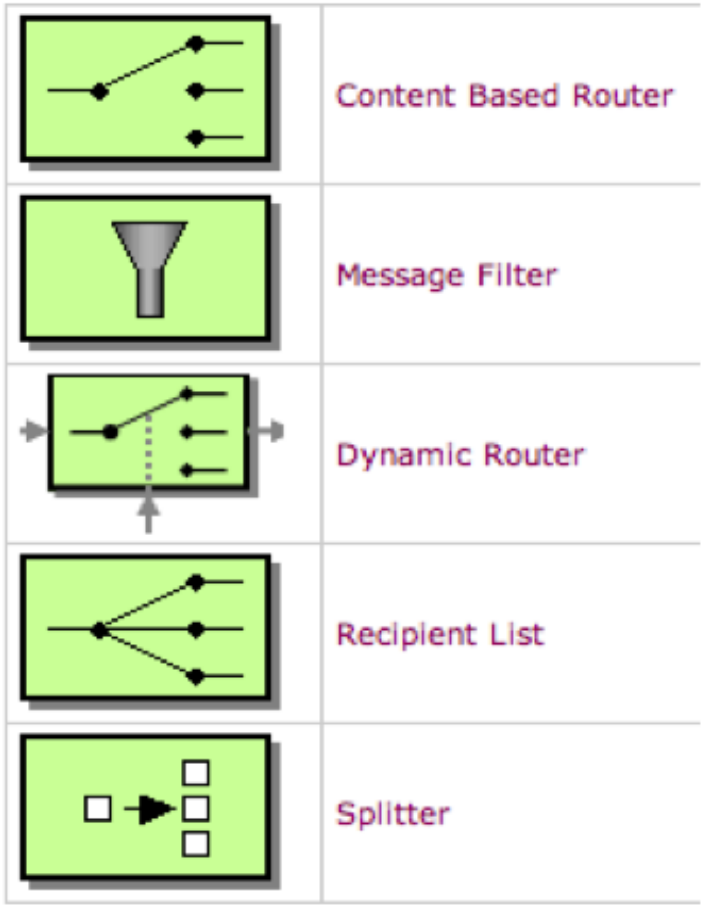
Portal Essentials

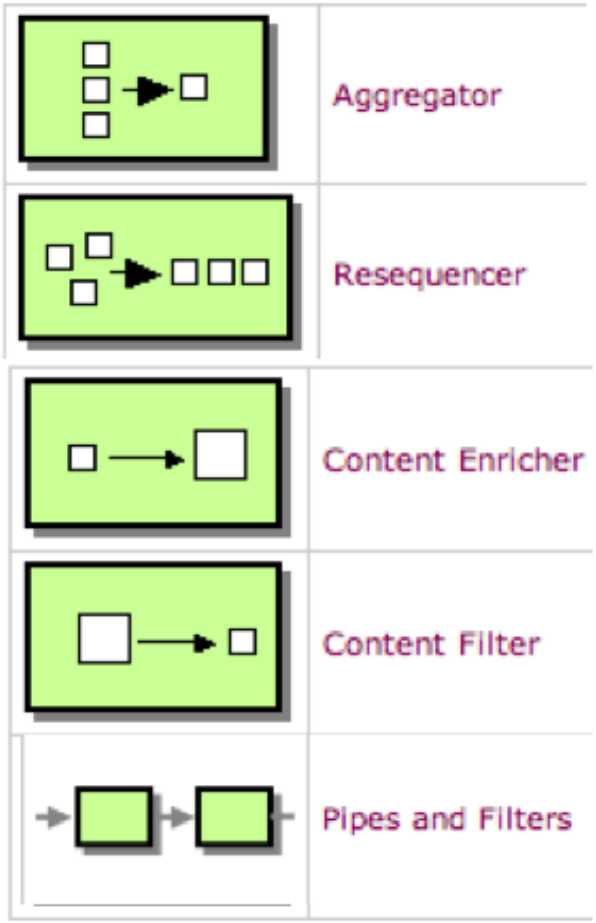
Challenges

- Data integration
  - Web Services
  - REST
  - Messaging
  - Proprietary API
- HTML integration
  - Proxy
  - Screen scraping
- Caching data
- Filter and transform data
  - Convert POJO's to JSON
  - Combine data sources
  - Renditions
    - PDF
    - Excel
    - HTML
- Route messages
  - Send messages to different targets
    - Based on payloads
    - Or other rules

- Patterns
  - Provide guidance
  - Common language
  - Outlines solutions to common challenges
  - 65+ patterns
- Available platforms
  - Spring Integration
  - Mule ESB
  - **Apache Camel**
  - Commercial offerings



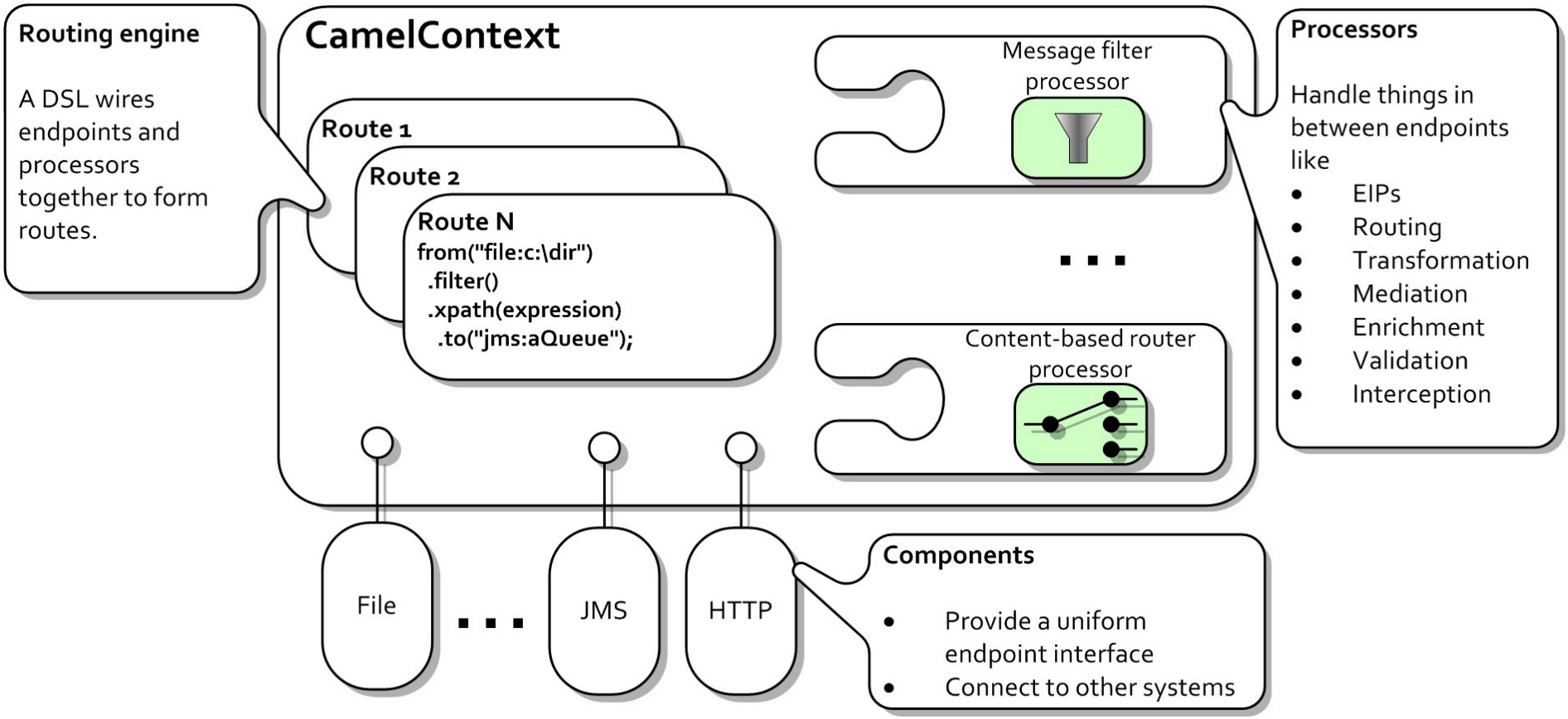


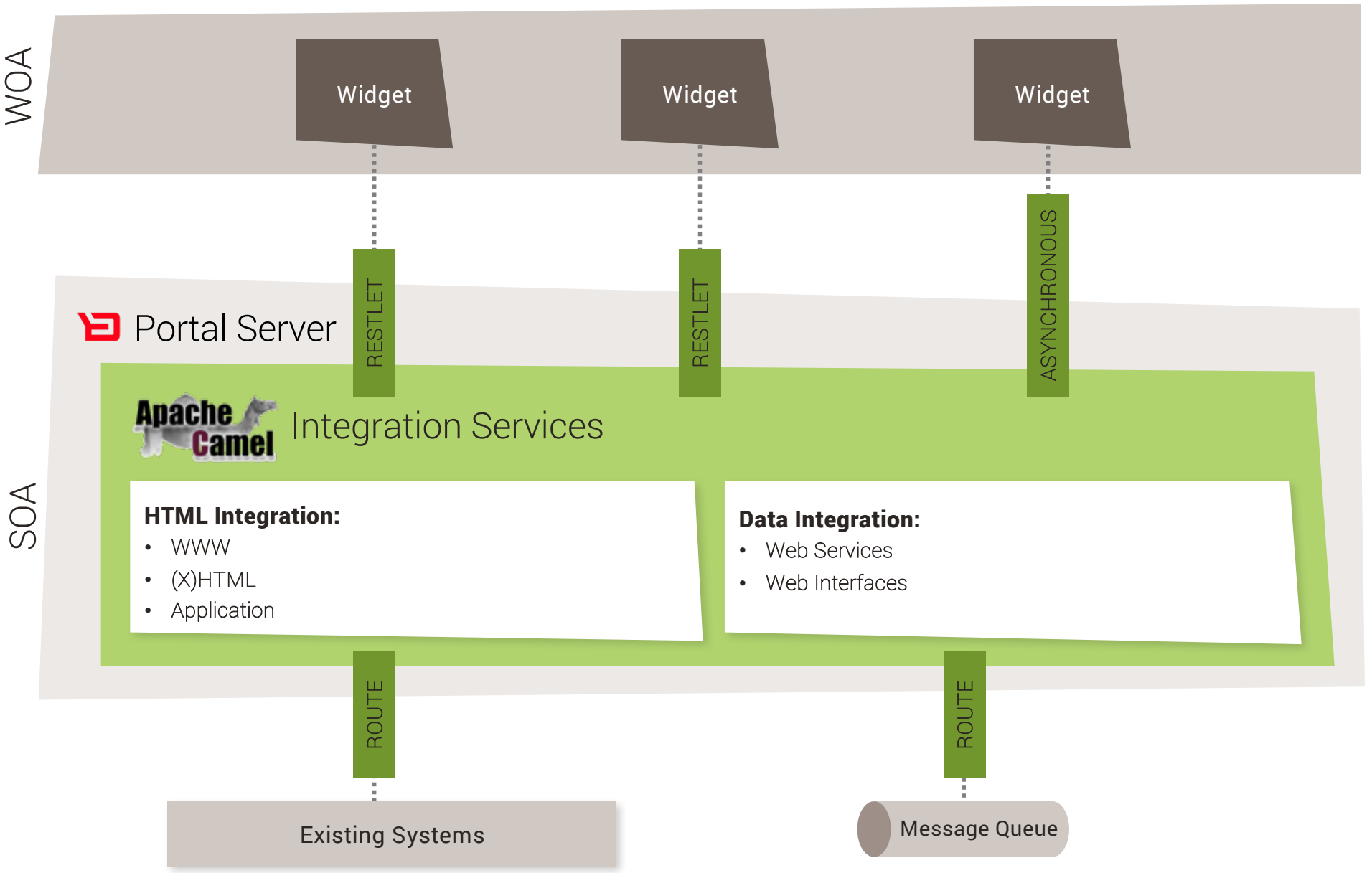


http://camel.apache.org/eip

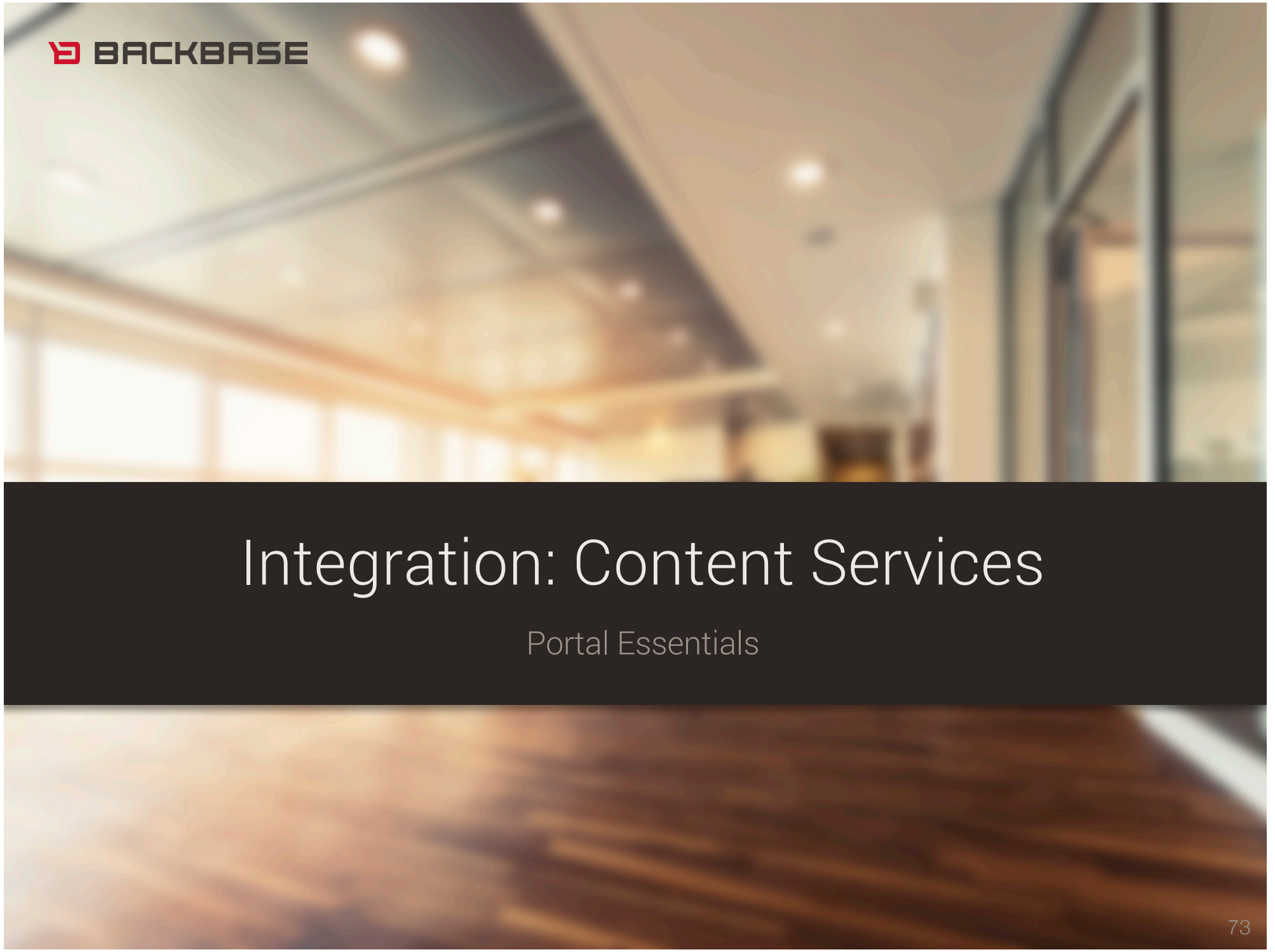
- Light weight integration library
- Kind of an embedded ESB
- Enterprise Integration Patterns
- Domain Specific Language
- Routing and Mediation
- Components
- Built in Transformers
- Active Community
- Easy to extend

Concise
Application
Messaging
Exchange
Language


Routing engine
A DSL wires endpoints and processors together to form routes.
CamelContext
Route 1
Route 2
Route N
from("file:c:\dir")
.filter()
.xpath(expression)
.to("jms:aQueue");
Message filter processor
Content-based router processor
Processors
Handle things in between endpoints like
EIPs
Routing
Transformation
Mediation
Enrichment
Validation
Interception
File
JMS
HTTP
Components
Provide a uniform endpoint interface
Connect to other systems


WOA
Widget
Widget
Widget
RESTLET
RESTLET
ASYNCHRONOUS
Portal Server
Apache Camel
Integration Services
SOA
HTML Integration:
• WWW
• (X)HTML
• Application
Data Integration:
• Web Services
• Web Interfaces
ROUTE
ROUTE
Existing Systems
Message Queue

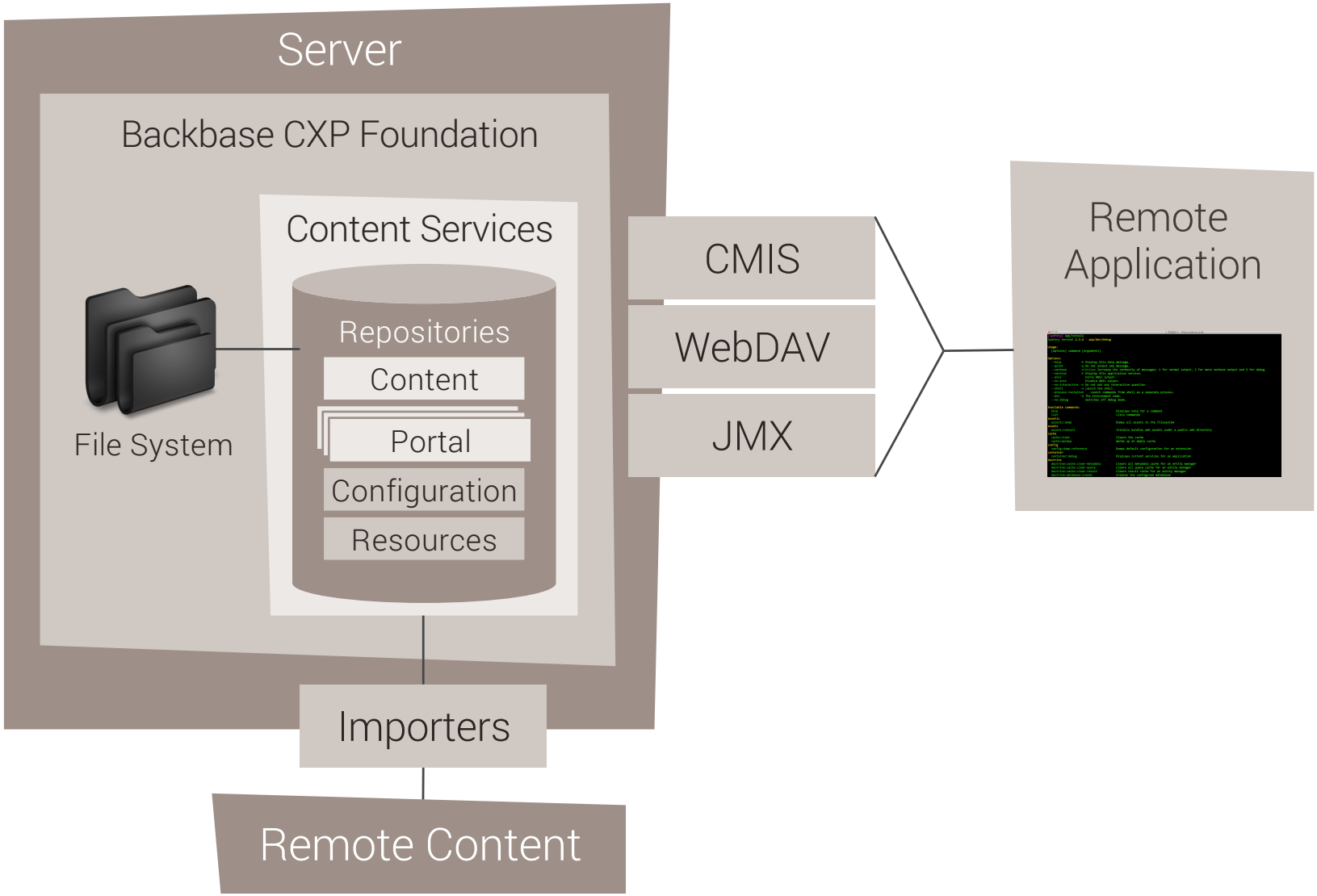
1. A component model
2. Flexible rendering and support for multi-device delivery
3. Personalization
4. The ability to integrate with a wide range of technologies
5. **Content services**
6. Security administration
7. Targeting
8. Publishing



Integration: Content Services

Portal Essentials

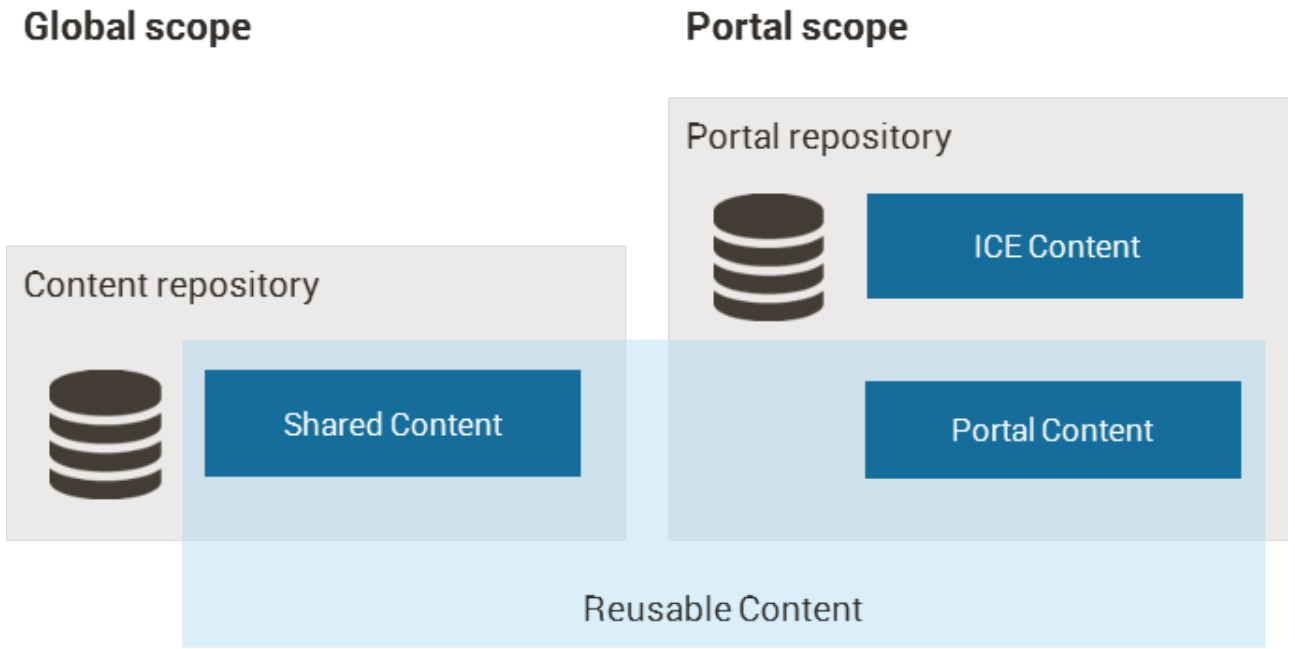
- Content Services is a content delivery platform provided as a CXP module.
- Centralized Content Repository
- Main functions:
  - Stores “In Context Editing” (a.k.a. ICE) content from CXP Manager.
  - Stores reusable content (potentially harvested through importers from remote sources)
  - Stores configuration and resources


Server
Backbase CXP Foundation
Content Services
Repositories
Content
Portal
Configuration
Resources
File System
CMIS
WebDAV
JMX
Remote Application
Importers
Remote Content

- Content Services main features:
  - **Content Aggregation:** imports data from remote sources and stores it in repositories
  - **Integration:** exposes CMIS and WebDAV interfaces to work with content
  - **Administration:** exposes the JMX interface for monitoring
  - **Scalability:** support for clustering means that Content Services instances can be configured for a variety of backend technologies
  - **Failover:** support for clustering means that requests can be routed to different Content Services instances if one instance is unavailable

- There are two kinds of content that may appear in portal pages:
  - **In-Context Editing (ICE) content** — rich text inserted during page management. ICE Content is published implicitly as part of the page in which it resides. ICE Content is stored automatically and is not manageable.
  - **Reusable Content** — content items that can be used across different pages of the same or different portals. Reusable Content is uploaded, managed and published centrally in the Shared Content and Portal Content apps and can be referenced in pages.
    - **Simple** — uploaded media type files which can be directly referenced in pages, such as images, videos, PDFs, etc.
    - **Structured** — instances of custom Structured Content types.

- Regarding scope, reusable content can be:
  - Shared (or global)
  - Portal-specific

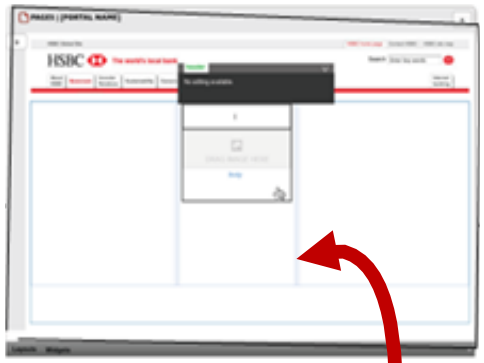


CONTENT AUTHORING

(Existing CMS)

PAGE EDITING

VISITORS



CMS
Authoring GUI

CMIS
interface

CMS Content
Repository

CMIS interface

Portal
Content
Repository

staging / live

CMIS interface

Portal Content
Repository

2

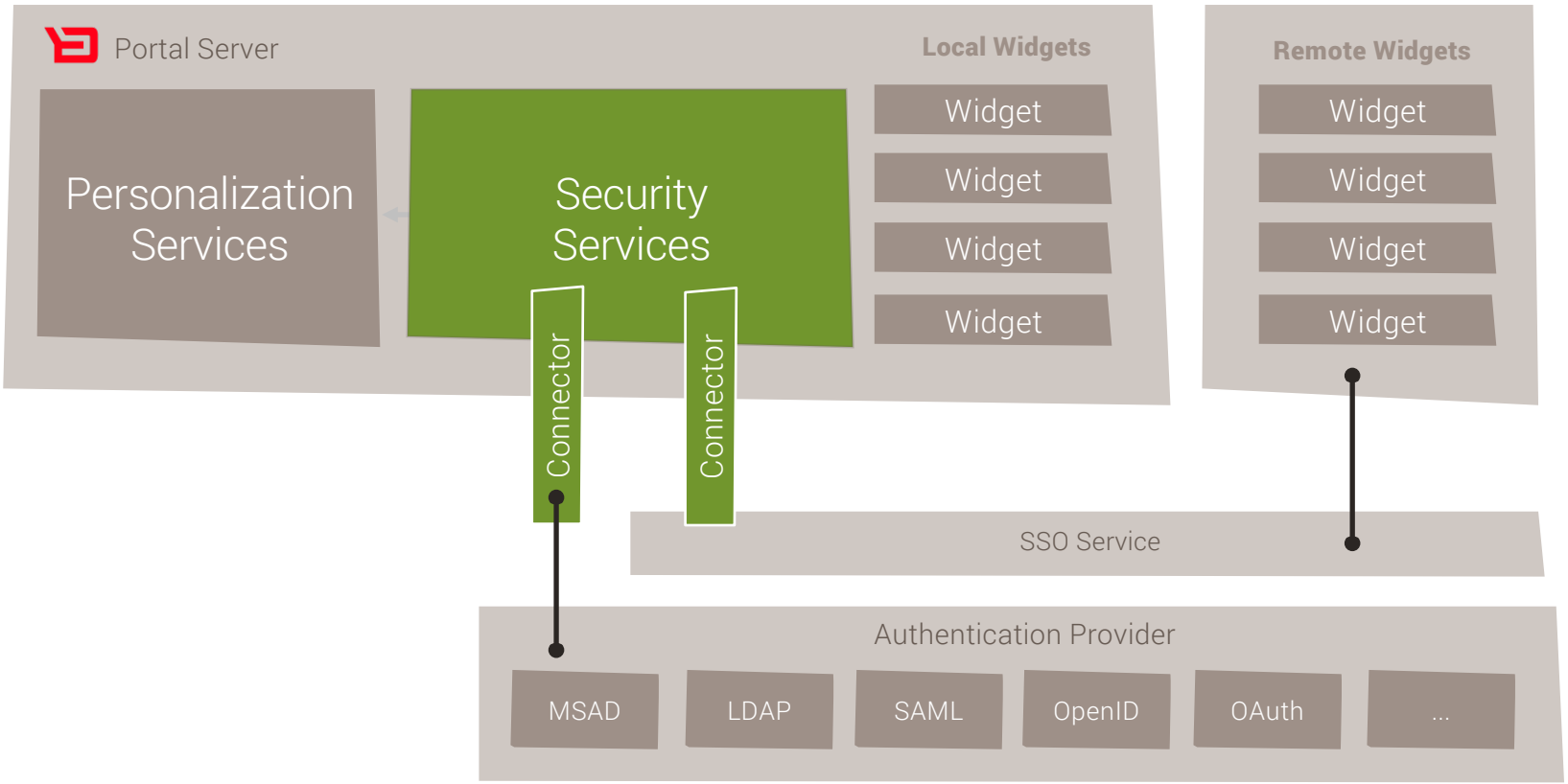
3

1 Import

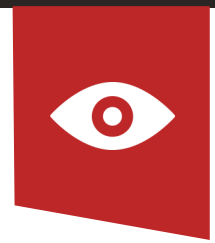
1. A component model
2. Flexible rendering and support for multi-device delivery
3. Personalization
4. The ability to integrate with a wide range of technologies
5. Content services
6. **Security administration**
7. Targeting
8. Publishing

Portal Security

Portal Essentials


Portal Server
Local Widgets
Remote Widgets
Personalization Services
Security Services
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Widget
Connector
Connector
SSO Service
Authentication Provider
MSAD
LDAP
SAML
OpenID
OAuth
...

- Based on **Spring Security**
- **Application Level Security**
  Spring Authentication Providers
  - HTTP BASIC/Digest authentication, HTTP X.509 client certificate exchange, LDAP, Form-based authentication, OpenID authentication, Automatic "remember-me" authentication, Anonymous authentication, Kerberos…,
  - Your own authentication systems
- **Additional Authentication Options for Portal Items**
- **Advanced topic, covered in depth in a separate module**



- Apply different rights to various items in CXP Manager
- Use manager and admin accounts to test the differences

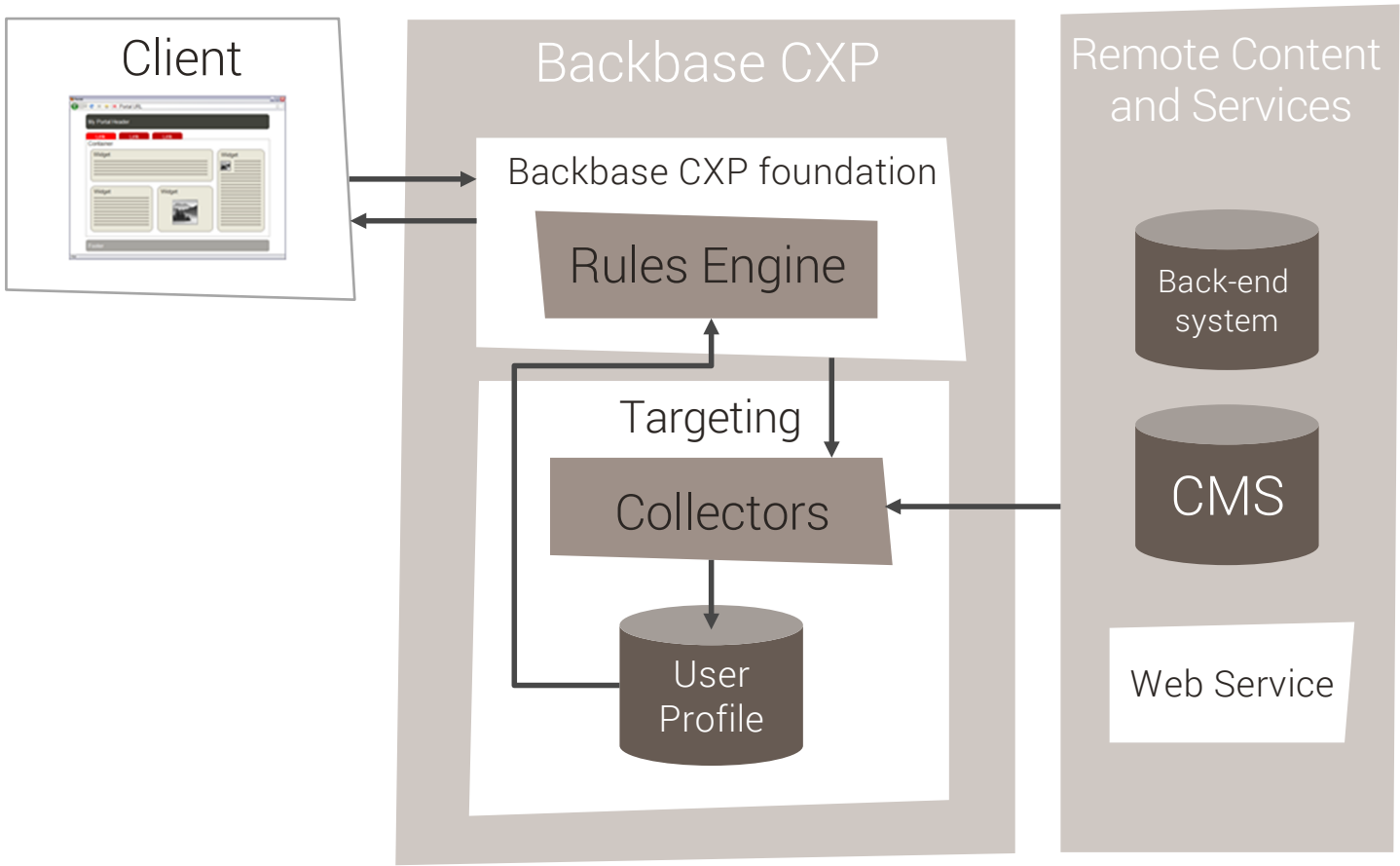
1. A component model
2. Flexible rendering and support for multi-device delivery
3. Personalization
4. The ability to integrate with a wide range of technologies
5. Content services
6. Security administration
7. **Targeting**
8. Publishing

Targeting

Portal Essentials

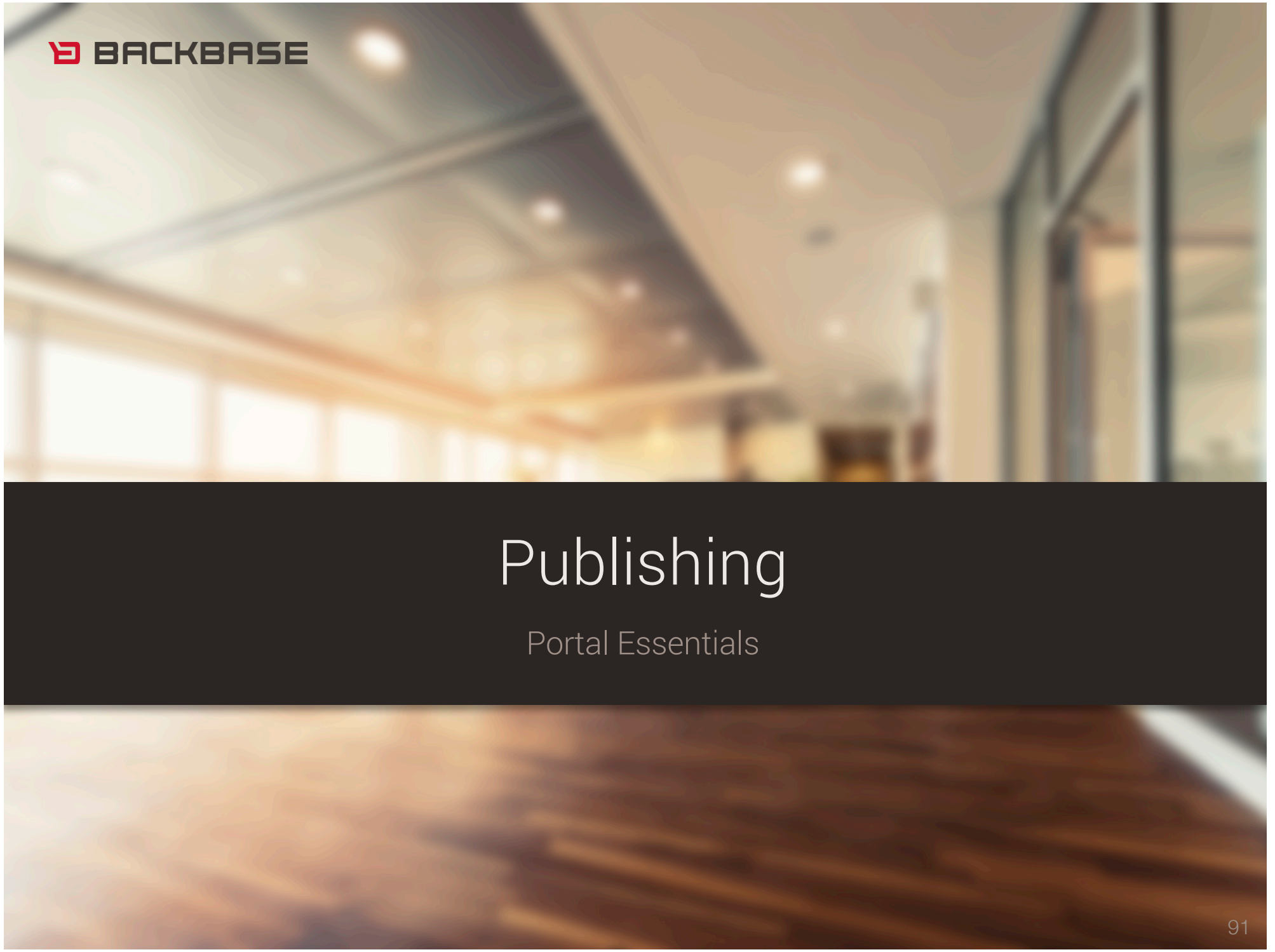
- Displays pages differently depending on certain criteria, for example, the user's browser, geographic location, demographic, etc.
- Targeting uses the **Targeting Container**, a container with multiple alternative displays
- For each alternative, **conditions** can be specified.
- **Collectors** collect data about the user from external sources and merge them in a user profile
- **Rules Engine** evaluates conditions sequentially, and returns the first alternative whose set of conditions evaluates to true.
- The **fallback** – the ”last” alternative being taken if none of the conditions evaluate to true.

Targeting Process Flow Overview



Lets look at how the CXP Manager handles the concepts we have just discussed

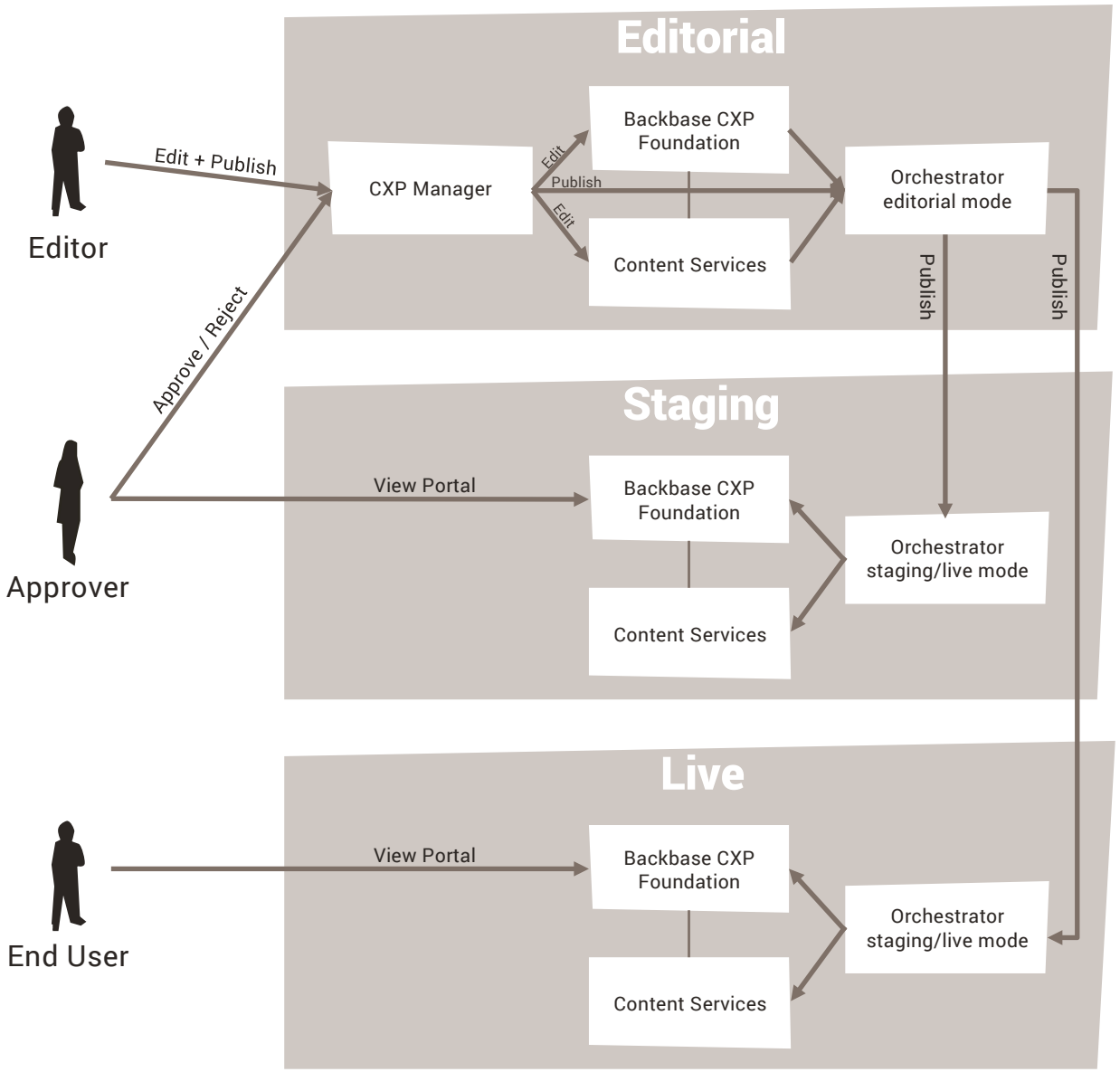
1. A component model
2. Flexible rendering and support for multi-device delivery
3. Personalization
4. The ability to integrate with a wide range of technologies
5. Content services
6. Security administration
7. Targeting
8. **Publishing**

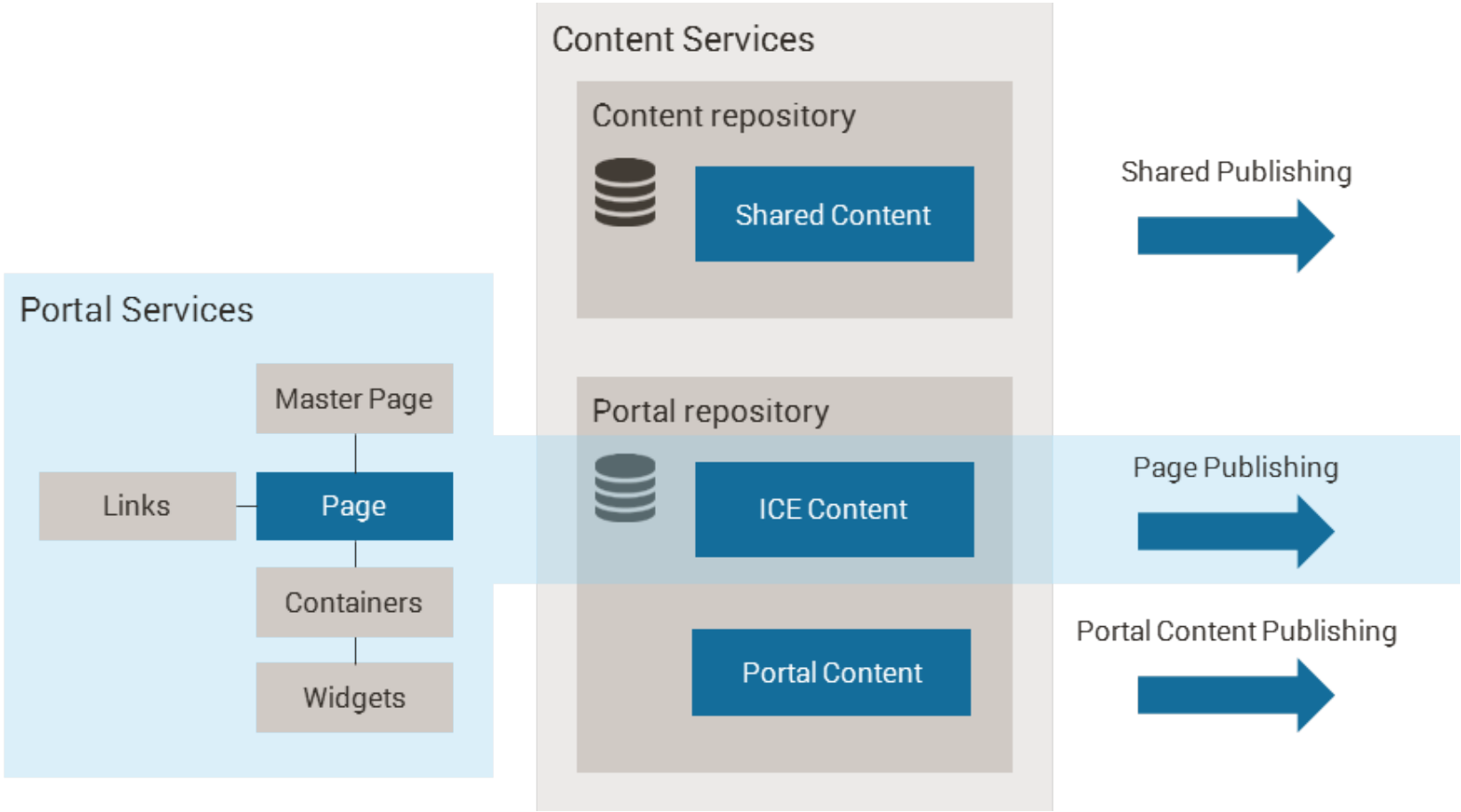


Publishing

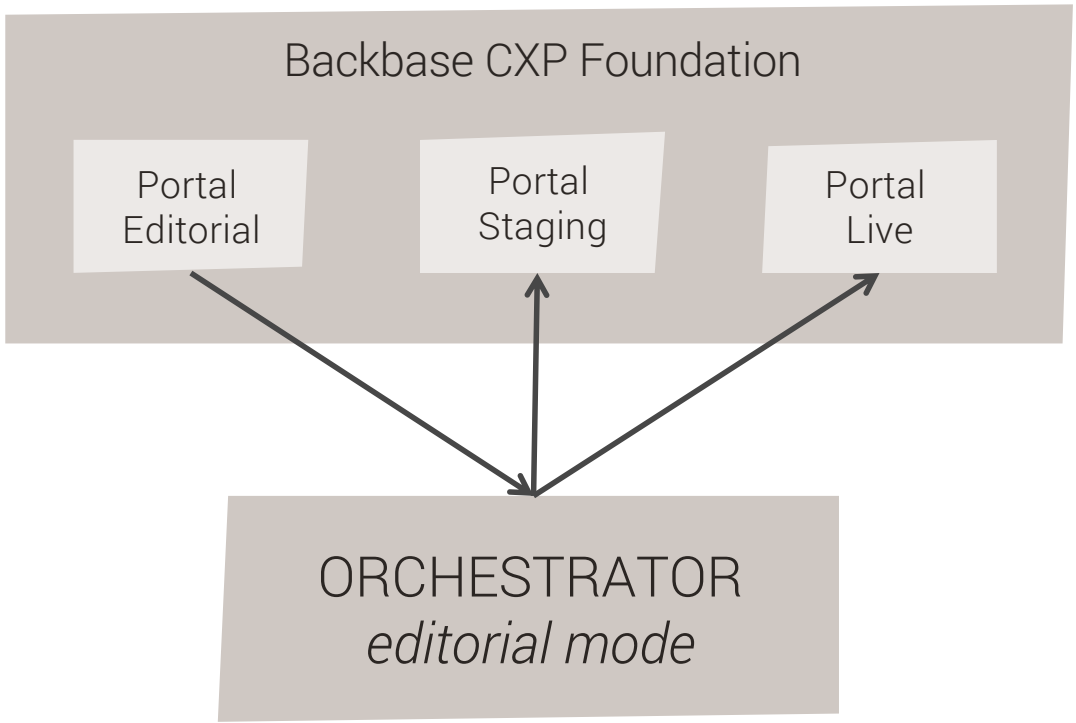
Portal Essentials

- Backbase CXP supports publishing pages and content to different environments in a controlled manner:
  - **Editorial environment**
  - **Staging environment**
  - **Live environment**
- The Orchestrator module is in charge of the publishing process.
  - It must be present on all environments alongside Backbase CXP Foundation.
  - Orchestrator locks data, sends and retrieves packages, manages approvals, and unlocks data.


Editorial
Editor
Edit + Publish
CXP Manager
Edit
Publish
Edit
Backbase CXP Foundation
Content Services
Orchestrator editorial mode
Publish
Publish
Approve / Reject
Staging
Approver
View Portal
Backbase CXP Foundation
Content Services
Orchestrator staging/live mode
Live
End User
View Portal
Backbase CXP Foundation
Content Services
Orchestrator staging/live mode


Content Services
Content repository
Shared Content
Shared Publishing
Portal Services
Master Page
Links
Page
Containers
Widgets
Portal repository
ICE Content
Page Publishing
Portal Content
Portal Content Publishing

- Publishing takes place across different portals instead of environments
- For testing and demo purposes (configurable in the Orchestrator configuration file)



- *Orchestrator* has two roles that you can map to groups in Backbase CXP Foundation:
  - Publisher – Allows users to submit publish requests.
  - Approver – Allows *Approvers* to approve or reject publish requests.
- Note that Backbase CXP enforces the *four-eye principle*: a user who submits a publish request may not approve it

Lets look at how the CXP Manager handles the concepts we have just discussed

- Portal has an item based component model
- Personalization
- Content services
- Rendering per item, per device
- Security administration
- Powerful integration mechanism
- Targeting
- Publishing

BACKBASE

Thank you!

www.backbase.com
sales-eu@backbase.com

New York: +1 646 478 7538
Amsterdam: +31 20 465 8888