BACKBASE

Portal Essentials

Focus Area: Foundation

BACKBASE

Group Workshop

Back-end

Services Integration

Content Services

Advanced Security

Targeting

Publishing

Front-end

Overview

Widget Development

Template Development

Portal Client

ICE

Foundation

Portal Essentials
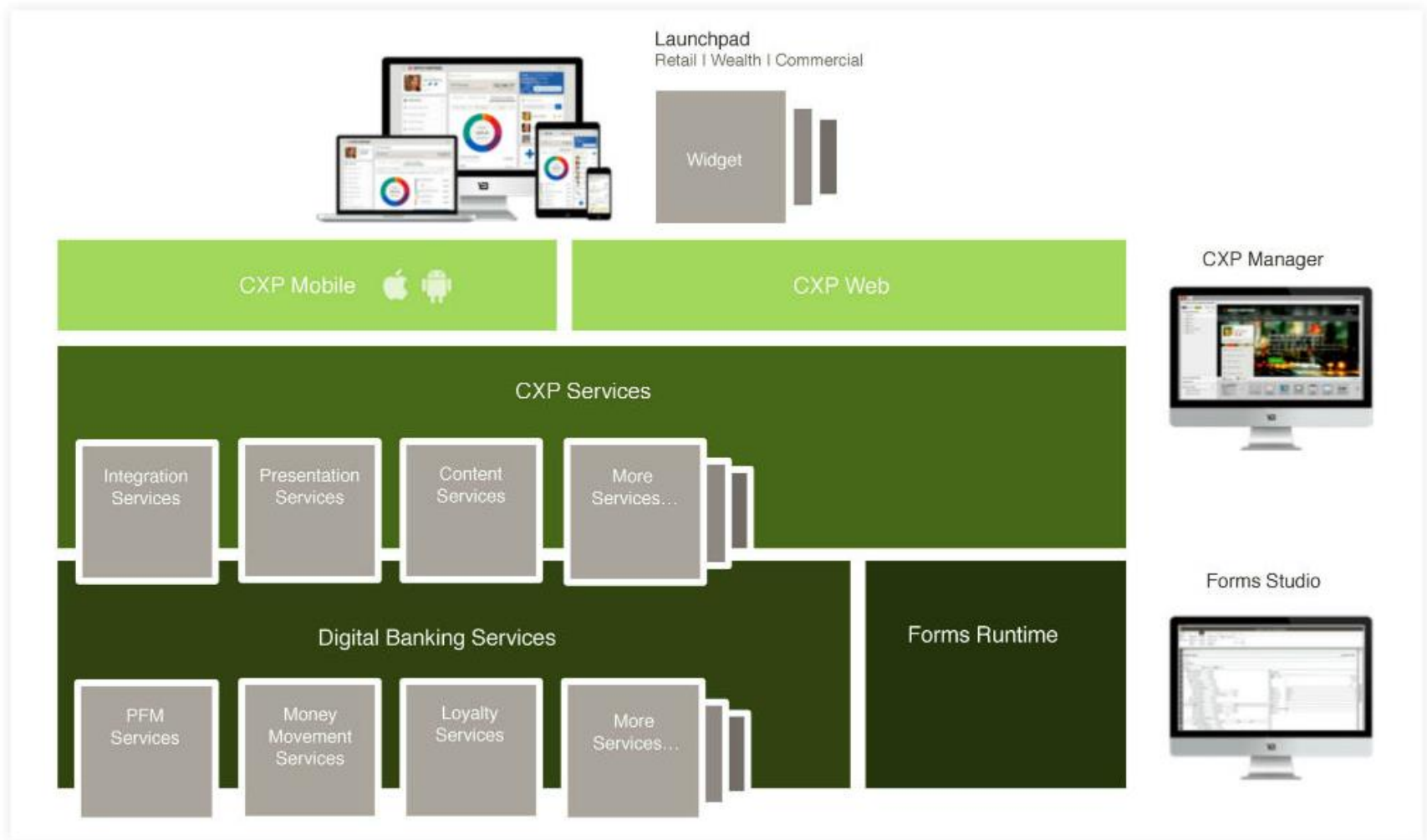
Portal Technologies

Portal Tools

Portal APIs

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

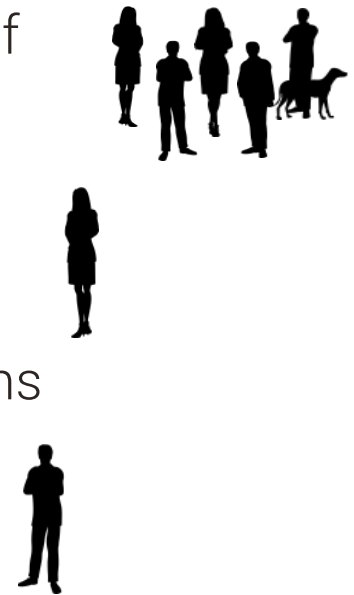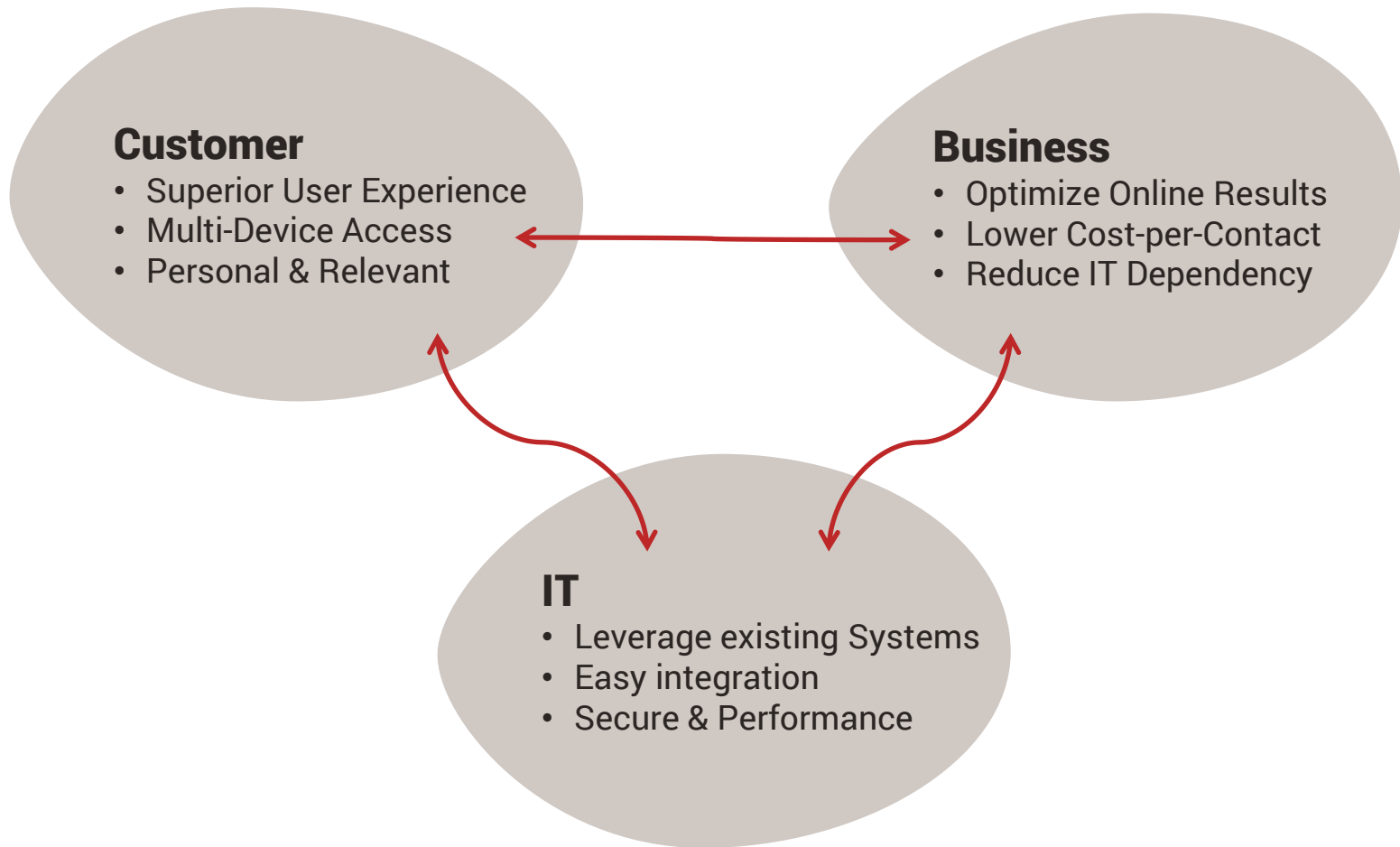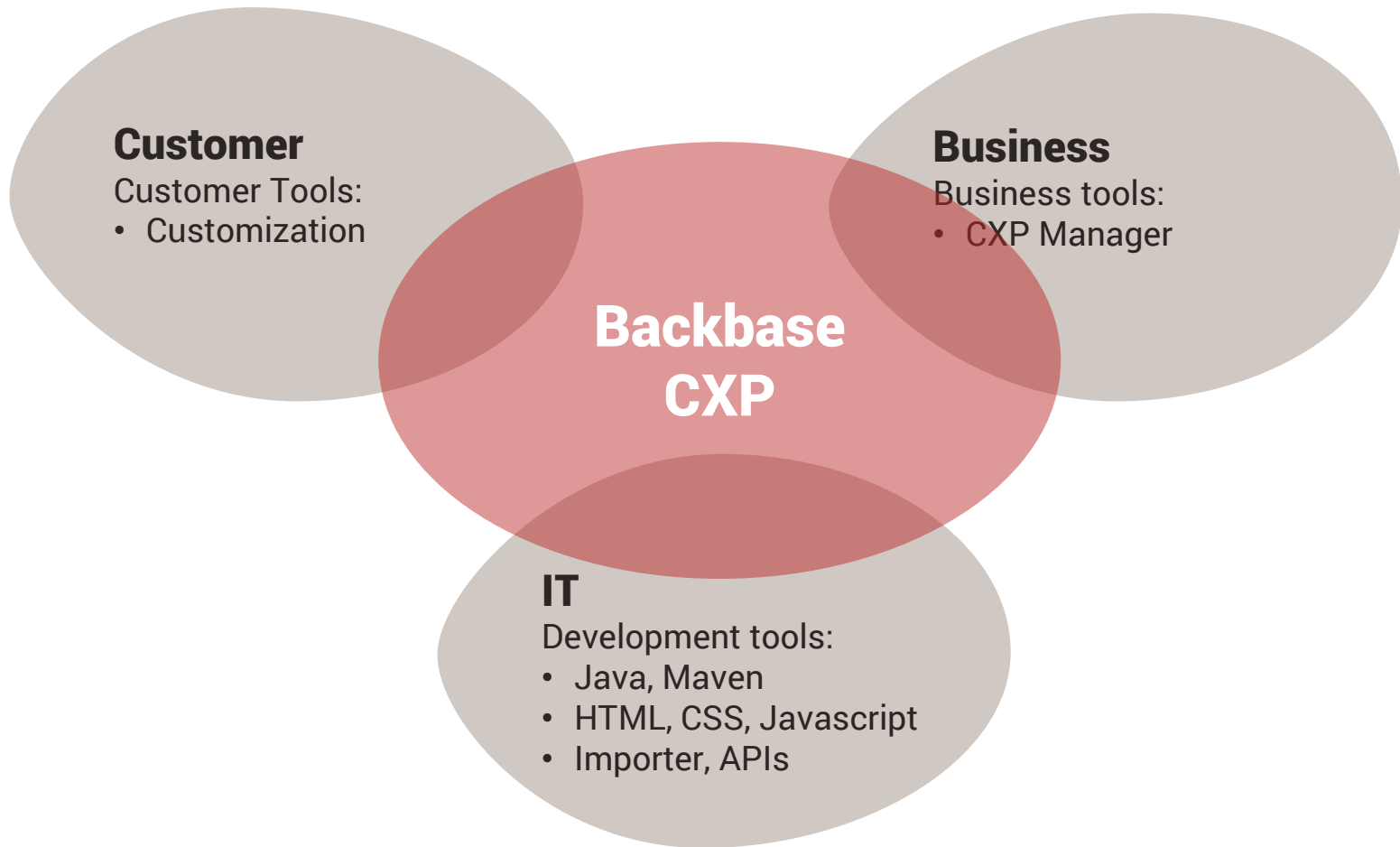**BACKBASE**

Our Products

Portal Essentials

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

Customer
- Superior User Experience
- Multi-Device Access
- Personal & Relevant

Business
- Optimize Online Results
- Lower Cost-per-Contact
- Reduce IT Dependency

IT
- Leverage existing Systems
- Easy integration
- Secure & Performance

Customer
Customer Tools:
• Customization

Business
Business tools:
• CXP Manager

**Backbase
CXP**

IT
Development tools:
• Java, Maven
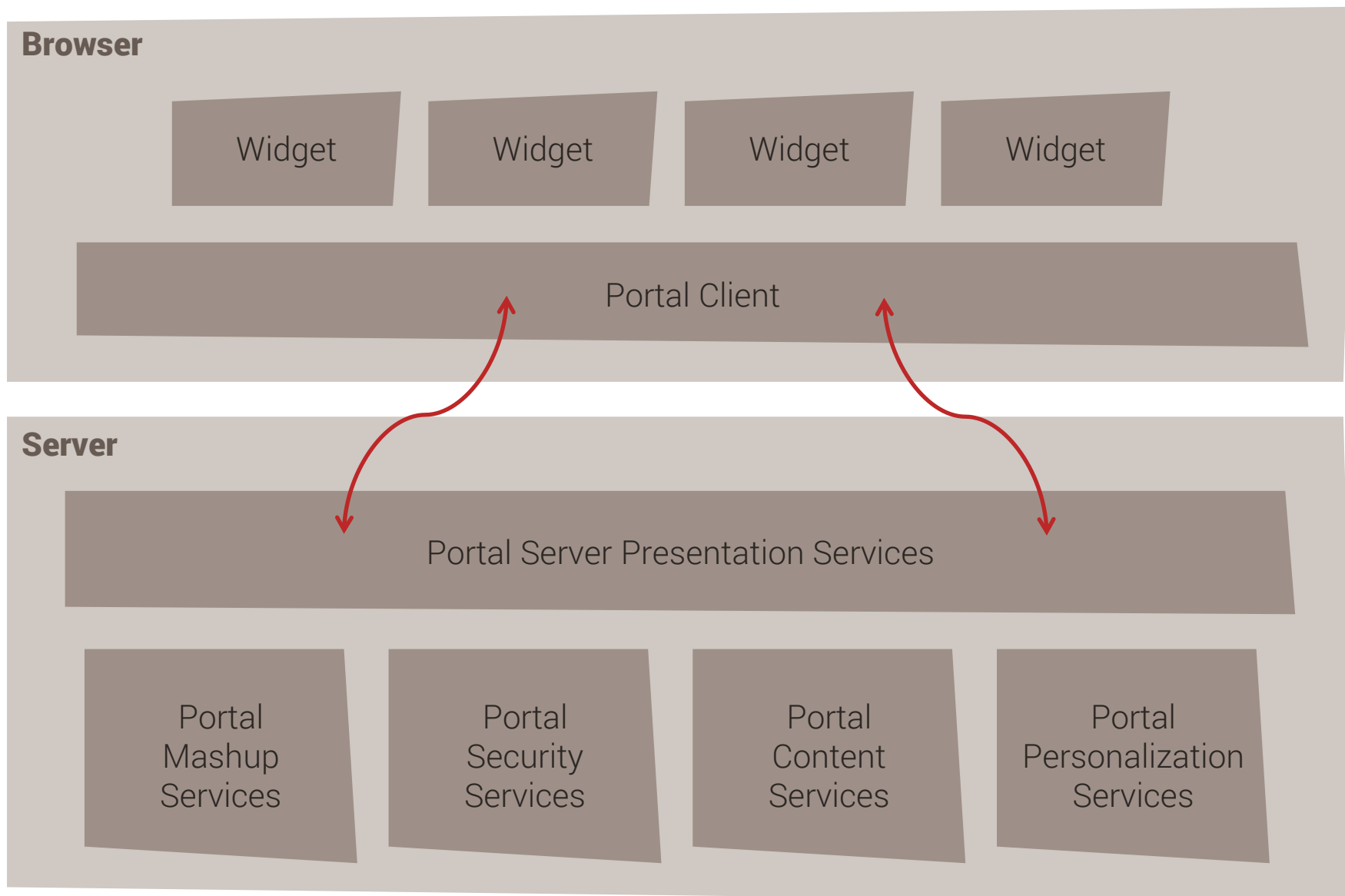• HTML, CSS, Javascript
• Importer, APIs

- 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 providing 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

BACKBASE

Browser

| Widget | Widget | Widget | Widget |

Portal Client

Server

Portal Server Presentation Services

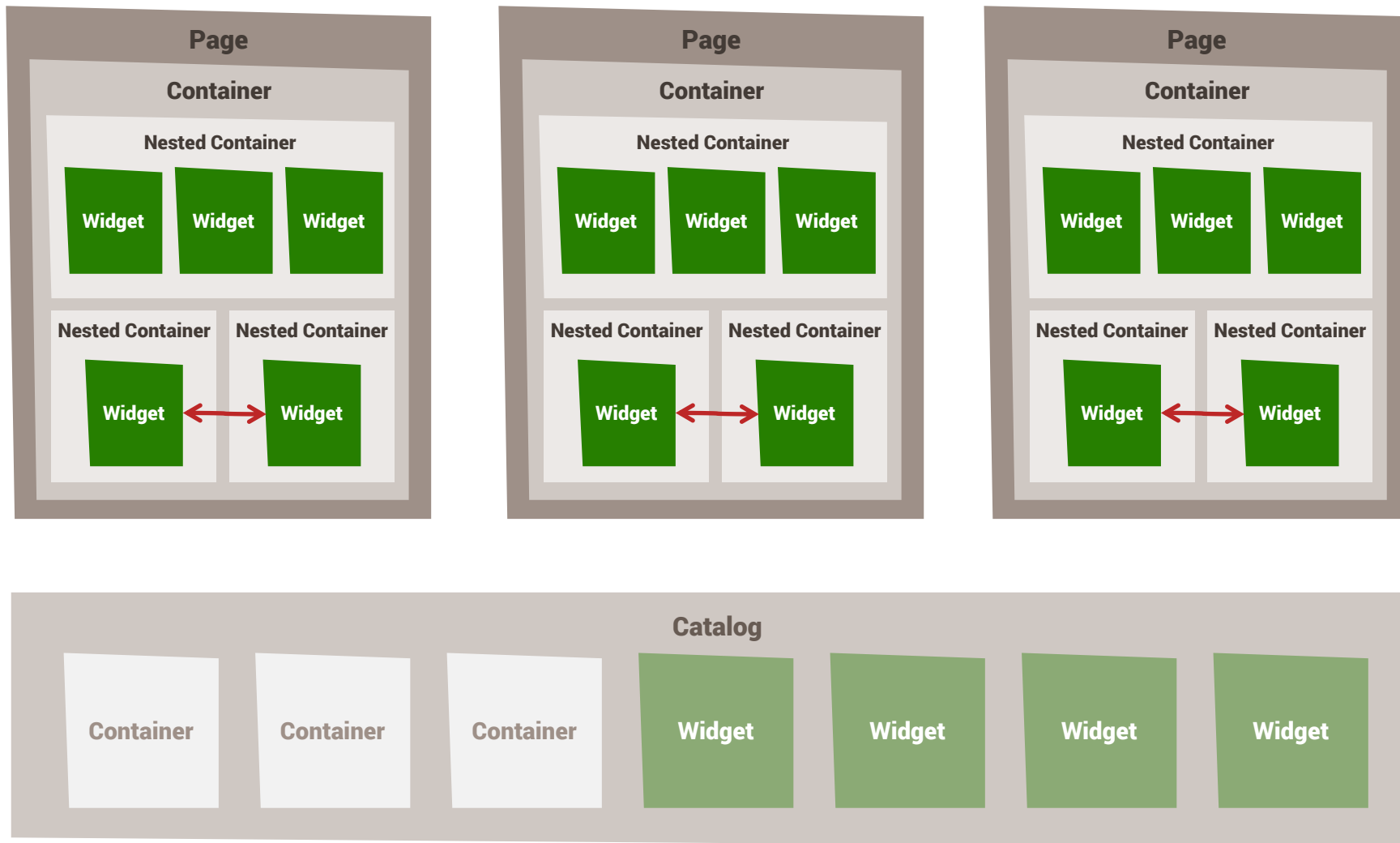| Portal Mashup Services | Portal Security Services | Portal Content Services | Portal Personalization Services |

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 **Portal Importer** (model defined in XML files)

    - **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.
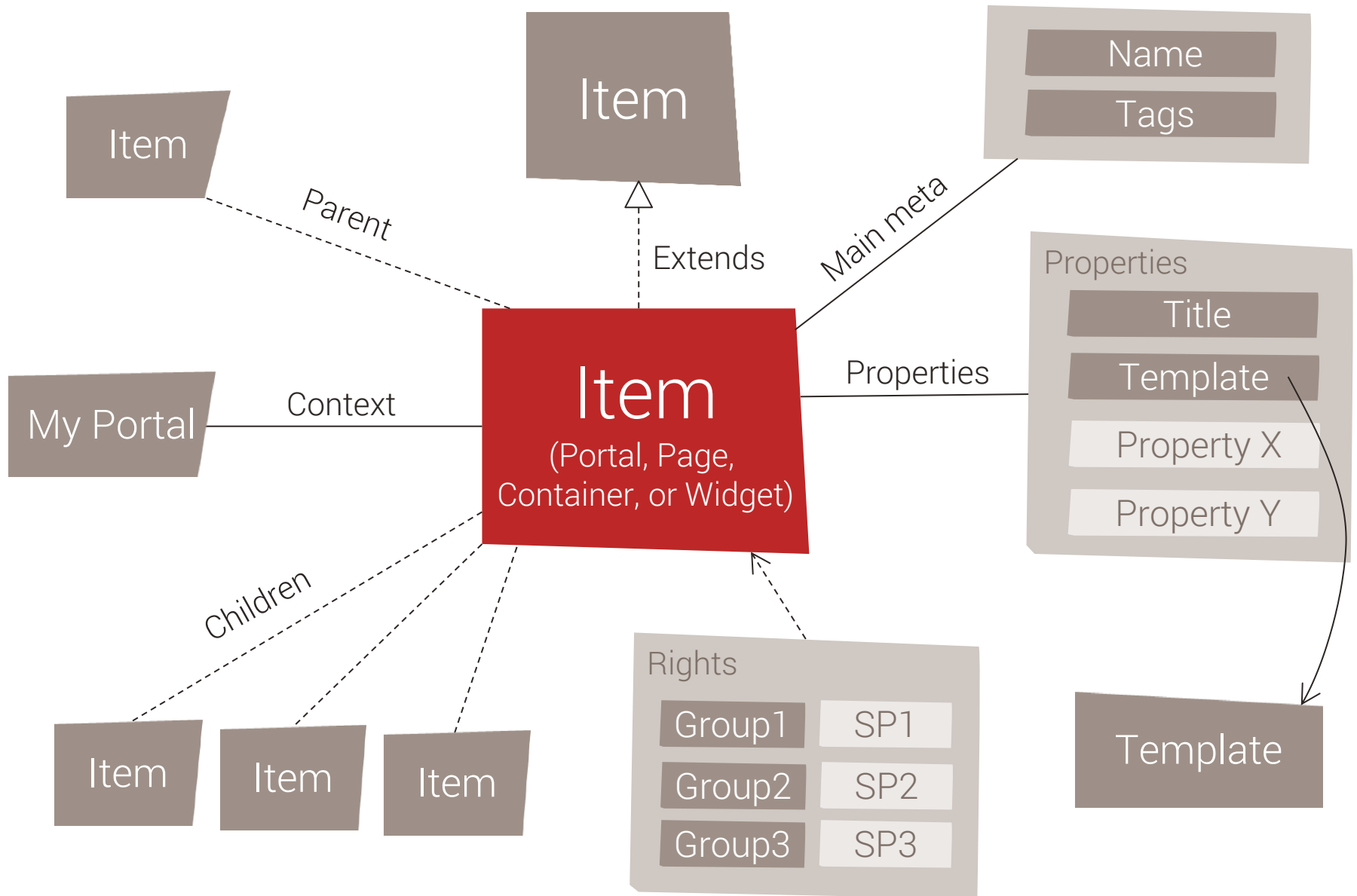
BACKBASE

Page

Container

Nested Container

| Widget | Widget | Widget |

Nested Container

Widget

Nested Container

Widget

Page

Container

Nested Container

| Widget | Widget | Widget |

Nested Container

Widget

Nested Container

Widget

Page

Container

Nested Container

| Widget | Widget | Widget |

Nested Container

Widget

Nested Container

Widget

Catalog

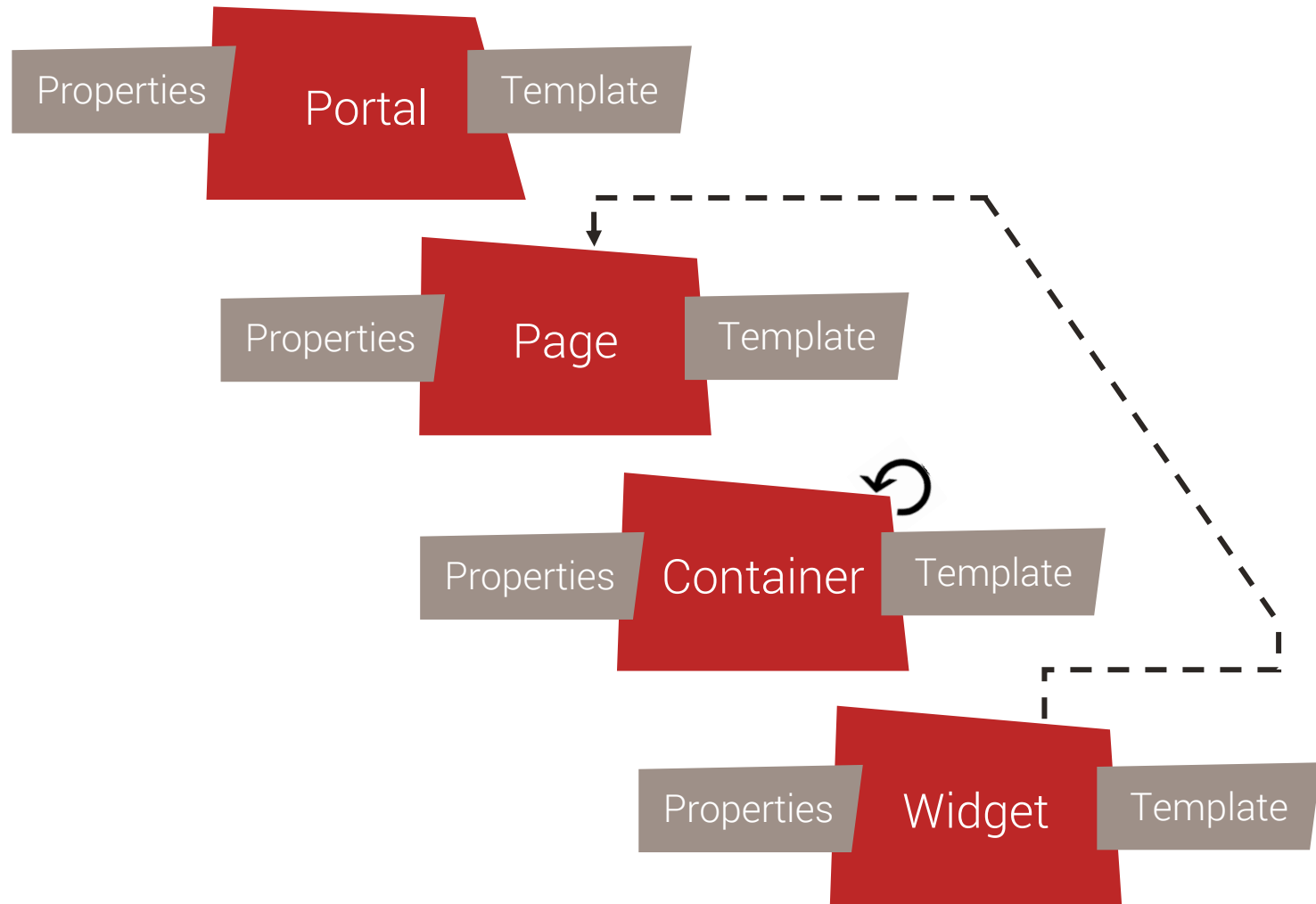| Container | Container | Container | Widget | Widget | Widget | Widget |

- A portal is composed of one or more pages

- Pages may directly include containers and widgets

- Pages can be of two types:

  - Master pages: provide a blueprint for common pages

  - Pages: extend master pages and inherit their template, properties, and children

- A Master page is a blueprint for pages.

- New pages are created by default based on a Master page.

- **Manageable areas**:

  - Manageable areas will map areas on a master pages available for editing and its content can only be defined on regular pages.

  - Areas outside manageable areas are (by default) locked for editing on regular pages.

- Containers logically group other container and widgets together

- Containers have a layout which is the visual representation of a grouping

- The core of a portal's functionality

- Autonomous mini applications

- Cannot include containers or widgets

- Backbase Widgets can utilize Portal's powerful integration mechanisms for including other applications

BACKBASE

Properties Portal Template

Properties Page Template

Properties Container Template

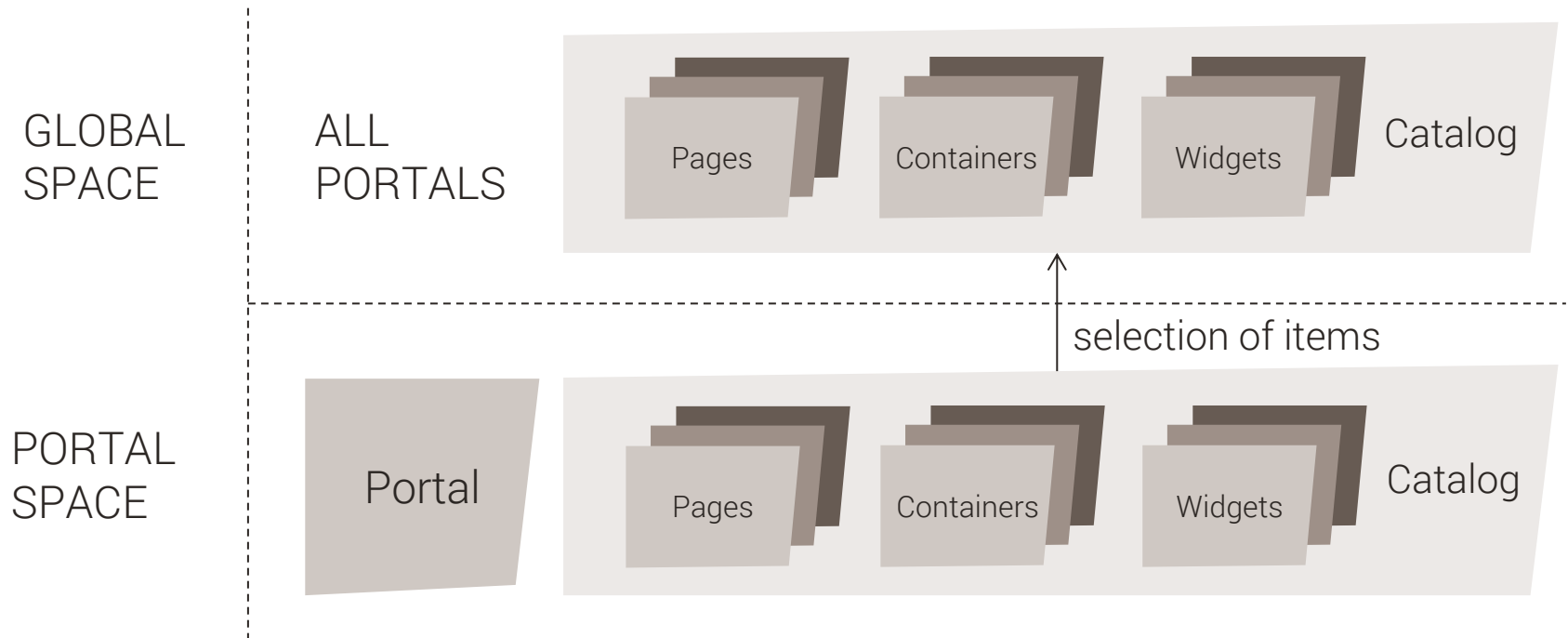Properties Widget Template

23

- An item can have **properties** (meta-information)

  - Name/value pairs

  - Typed (e.g. double, integer, string, ..)

- Every item type get a set of **default properties**

  - e.g. default landing page and default template (web/mobile)

- An item can have optional **custom properties**

- A user (depending on permissions) can **set his own value** to an item properties (**customization** – user scope)

- Deletions in Backbase CXP always cascade: children are deleted as well

- Deleting a page deletes its containers and widgets.

- Only users with `ADMIN` role can delete objects.

- Deletions in Backbase CXP cannot be undone
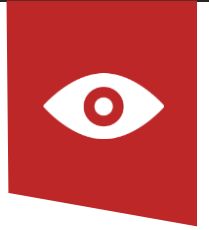
Lets look at how the CXP Manager handles the

concepts we have just discussed

BACKBASE

- Server Level – Enterprise Catalog (global space BBHOST)

- Portal Level – Portal Catalog

GLOBAL SPACE | ALL PORTALS

Pages | Containers | Widgets | Catalog

selection of items

PORTAL SPACE | Portal

Pages | Containers | Widgets | Catalog

- Rules of inheriting *properties*:
  - Objects in a portal catalog inherit properties from the abstract object in the server catalog, unless a user explicitly sets a property.
  - Objects instantiated in a portal inherit properties from the abstract object in the portal catalog, unless a user explicitly sets a property.
- Rules of inheriting *rights*:
  - Abstract objects in the server catalog inherit rights from Portal Foundation.
  - Abstract objects in a portal catalog inherit rights from the portal.
  - Instantiated objects inherit rights from their parent (either a portal, a page, or a container).

Lets look at how the CXP Manager handles the

concepts we have just discussed

■ BACKBASE

- The link item stores pointers to other items or external URLs

- Navigation widgets use links to build navigation trees

BACKBASE PORTAL MANAGER TYPE	TECHNICAL TYPE	DESCRIPTION
Page	page	Refers to a page.
External Link	externalLink	Refers to a URL outside Backbase Portal.
Divider	divider	Displays a divider in the navigation tree. Does not refer to anything.
Header	menuHeader	Displays a menu header in the navigation tree. Does not refer to a page itself, but can have children that do.
Alias	alias	Refers to a page that already has an associated link.
-	state	This link type is reserved for special use cases. The tab layout from the Nexus bundle uses this link type.

- CXP Manager creates a link for every new page and creates menus as shown in the image below:
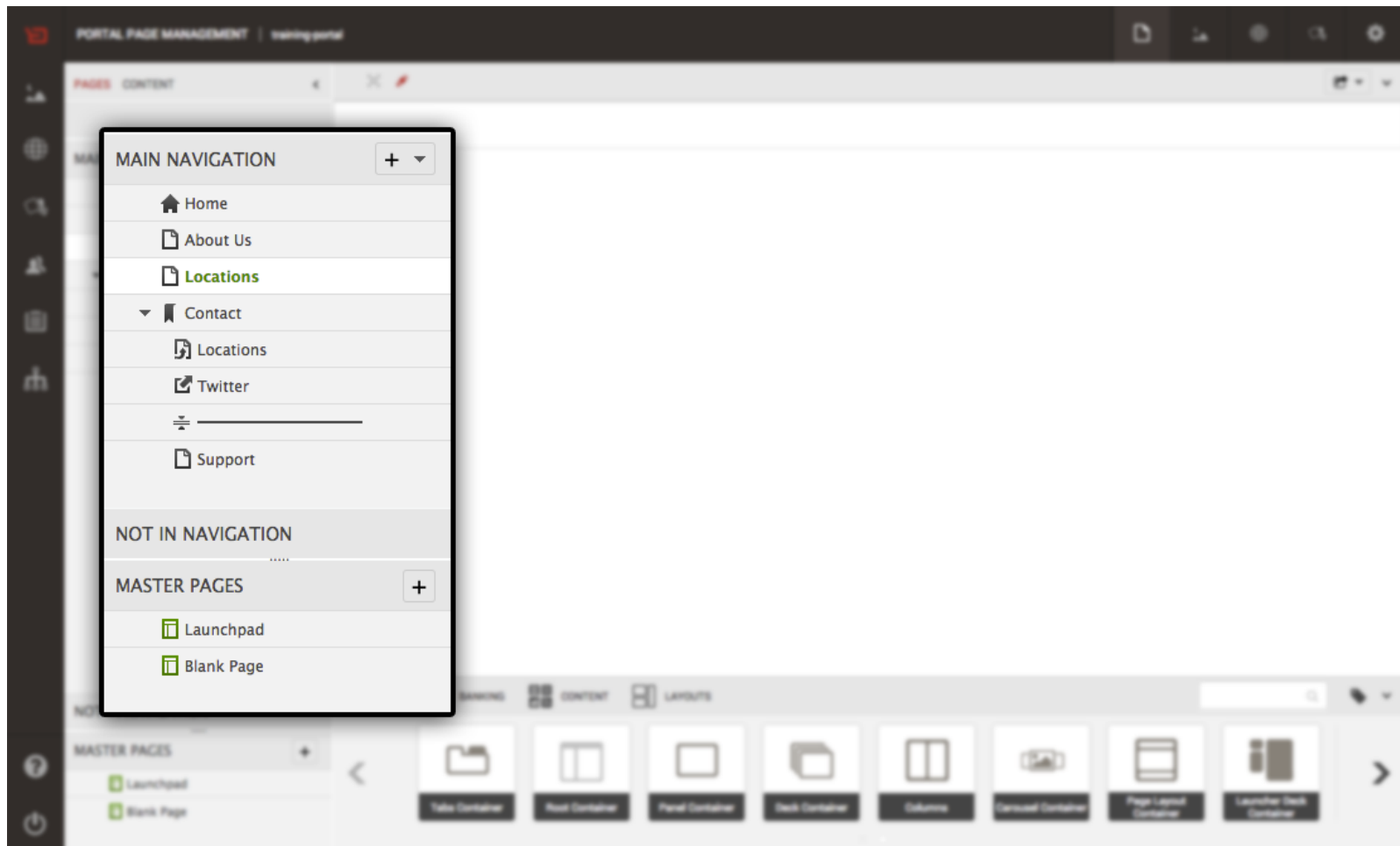
- Root links are menu headers that divide the navigation tree into smaller sub-trees

- 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 URLs: automatically generated, human-readable addresses pointing to links
  - External URLs: point to resources outside of a portal
  - Friendly URLs: easy-to-remember addresses pointing to links; defined by a user

- Dynamic URLs (new in 5.5)
  - 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

 - If a state link refers to the same page, both the page and the state link are deleted

BACKBASE

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
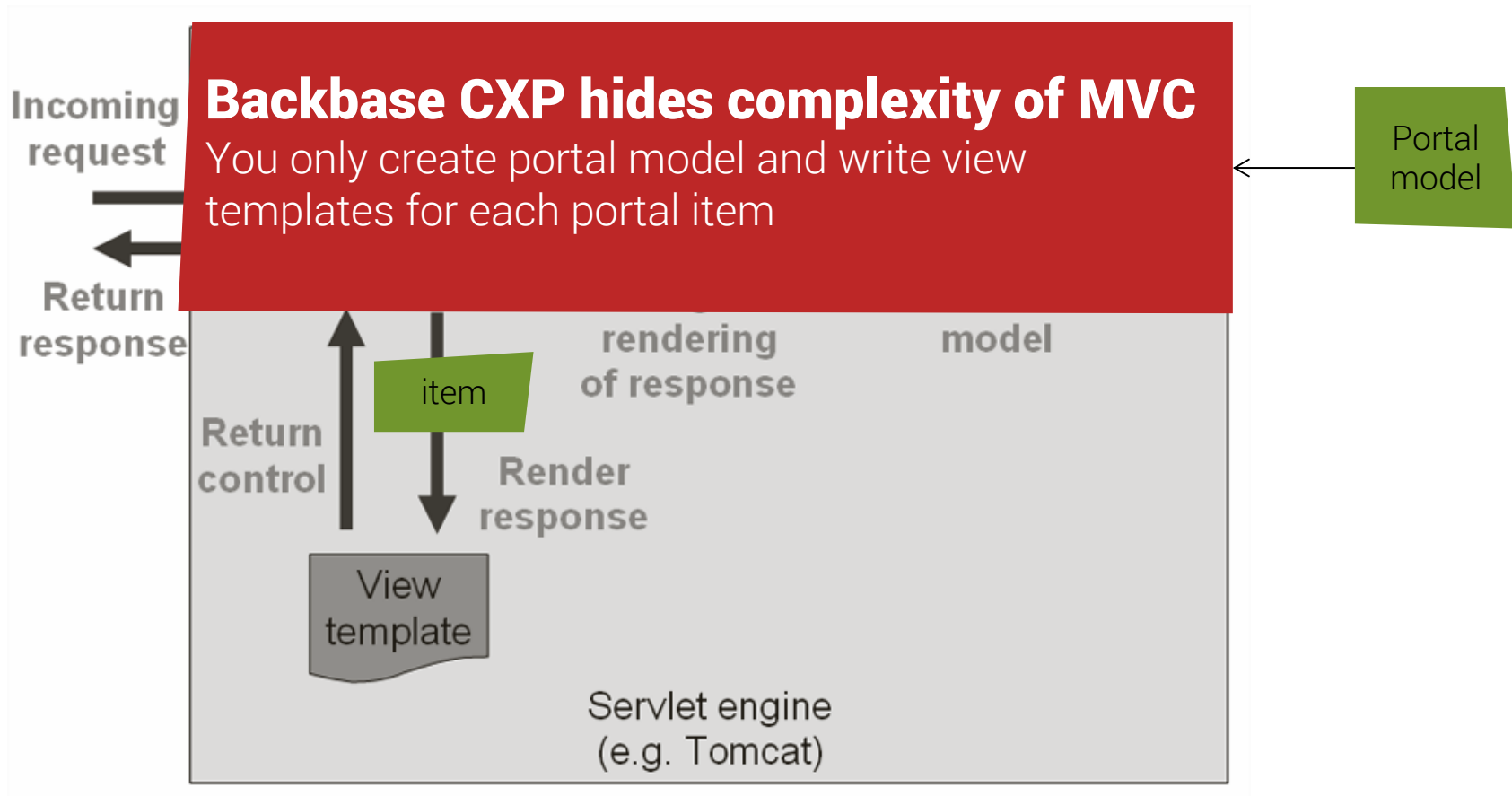8. Publishing

Rendering the Model

Portal Essentials
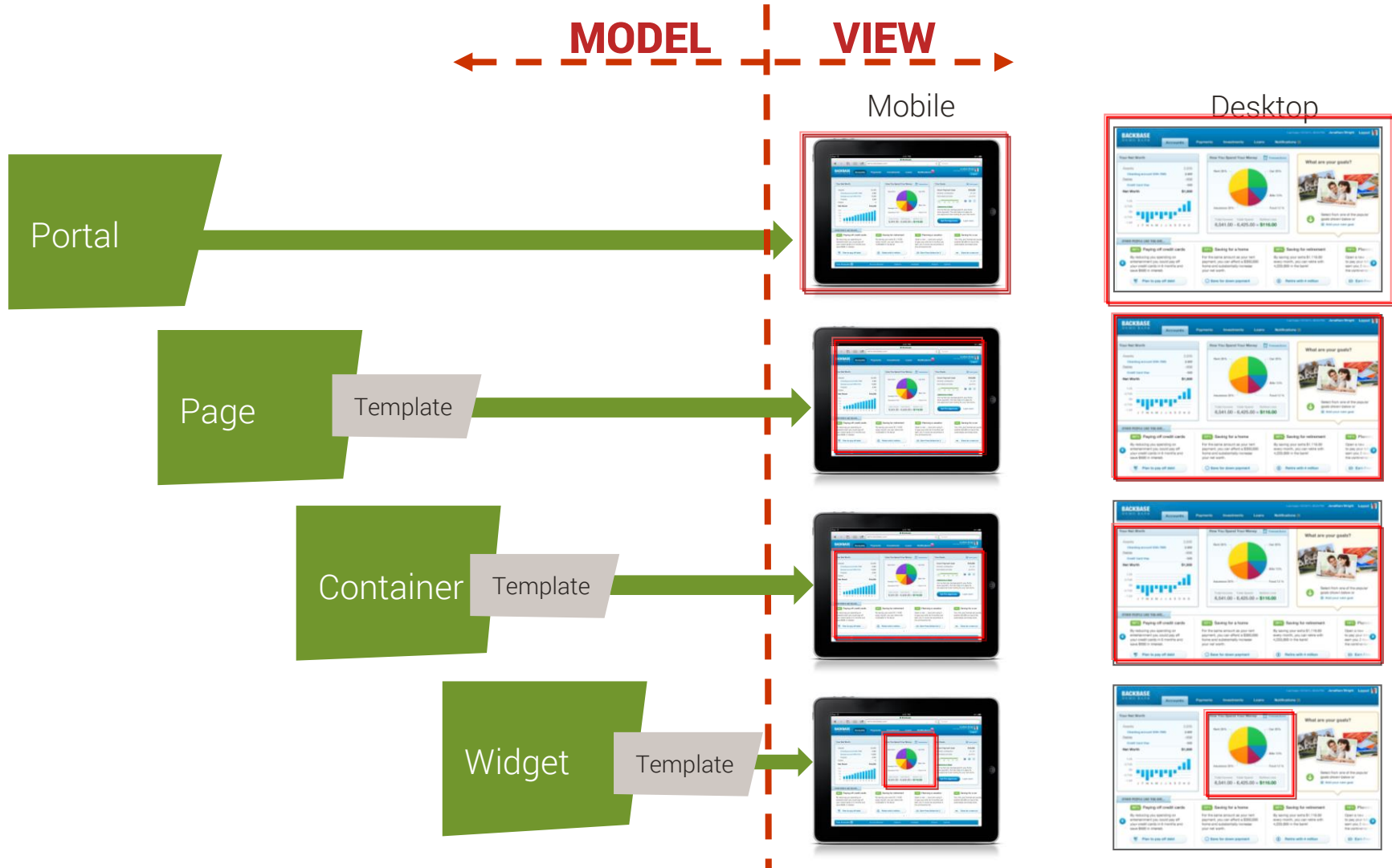
- Define physical files used to render a Portal Item
- Each main device group can have its own template
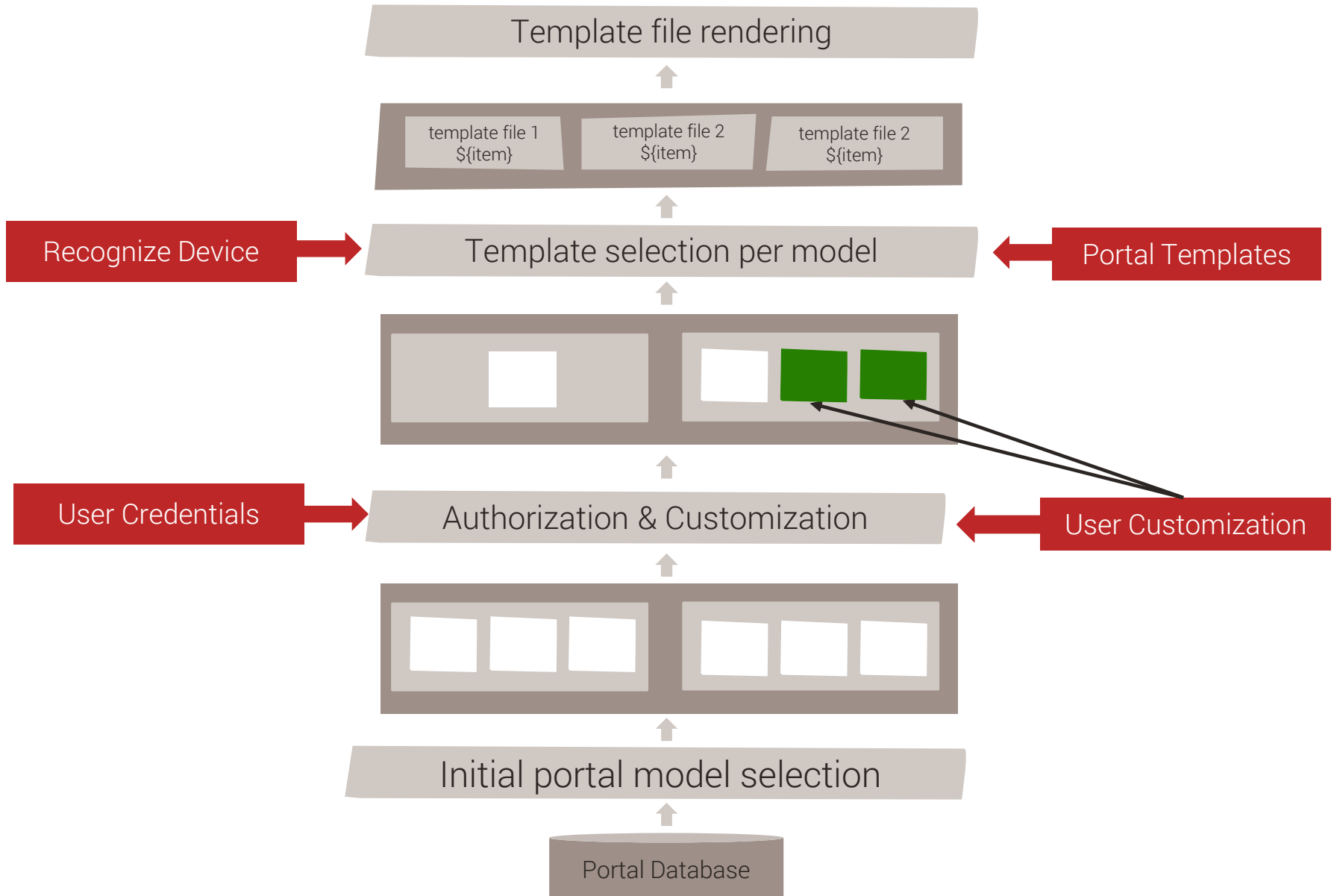 - Desktop (default)
  - Tablet
  - Mobile

- 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)
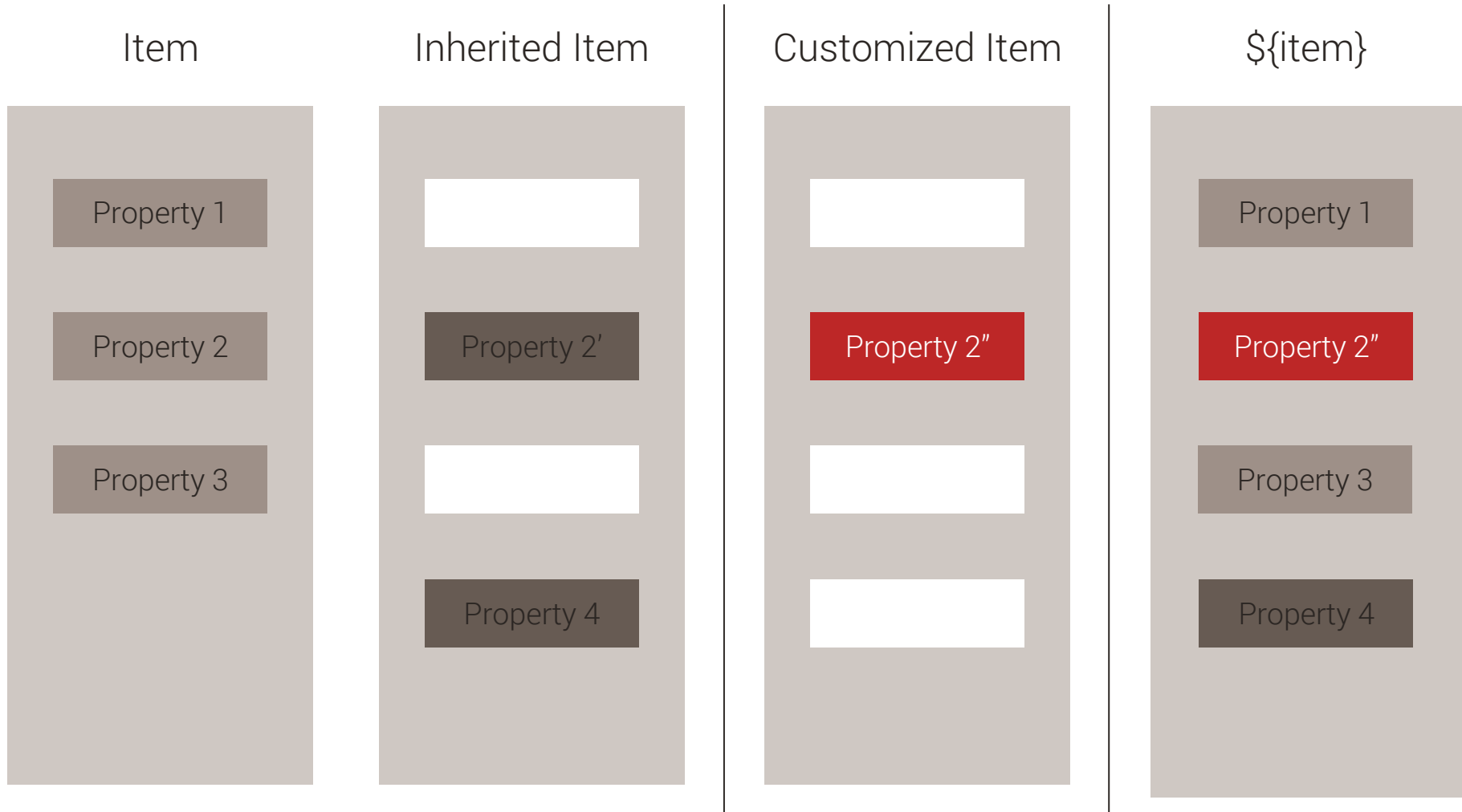 - while providing a loose coupling between these elements

- Backbase CXP uses the Spring MVC framework under the hood and implements the model-view-controller pattern
  - Objects belonging to the Backbase CXP Object 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

Incoming request

Return response

Backbase CXP hides complexity of MVC
You only create portal model and write view templates for each portal item

Portal model

item

rendering of response

model

Return control
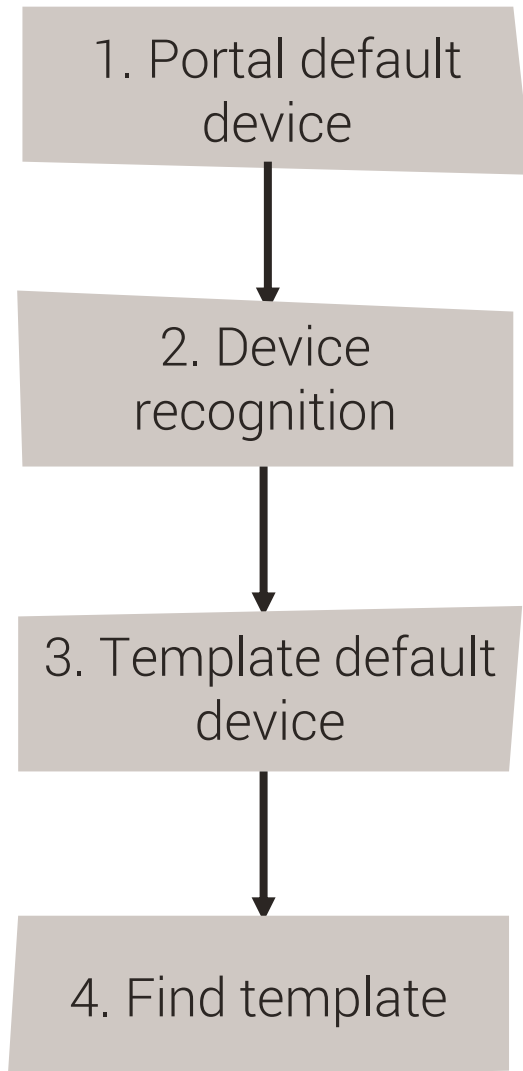
Render response

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
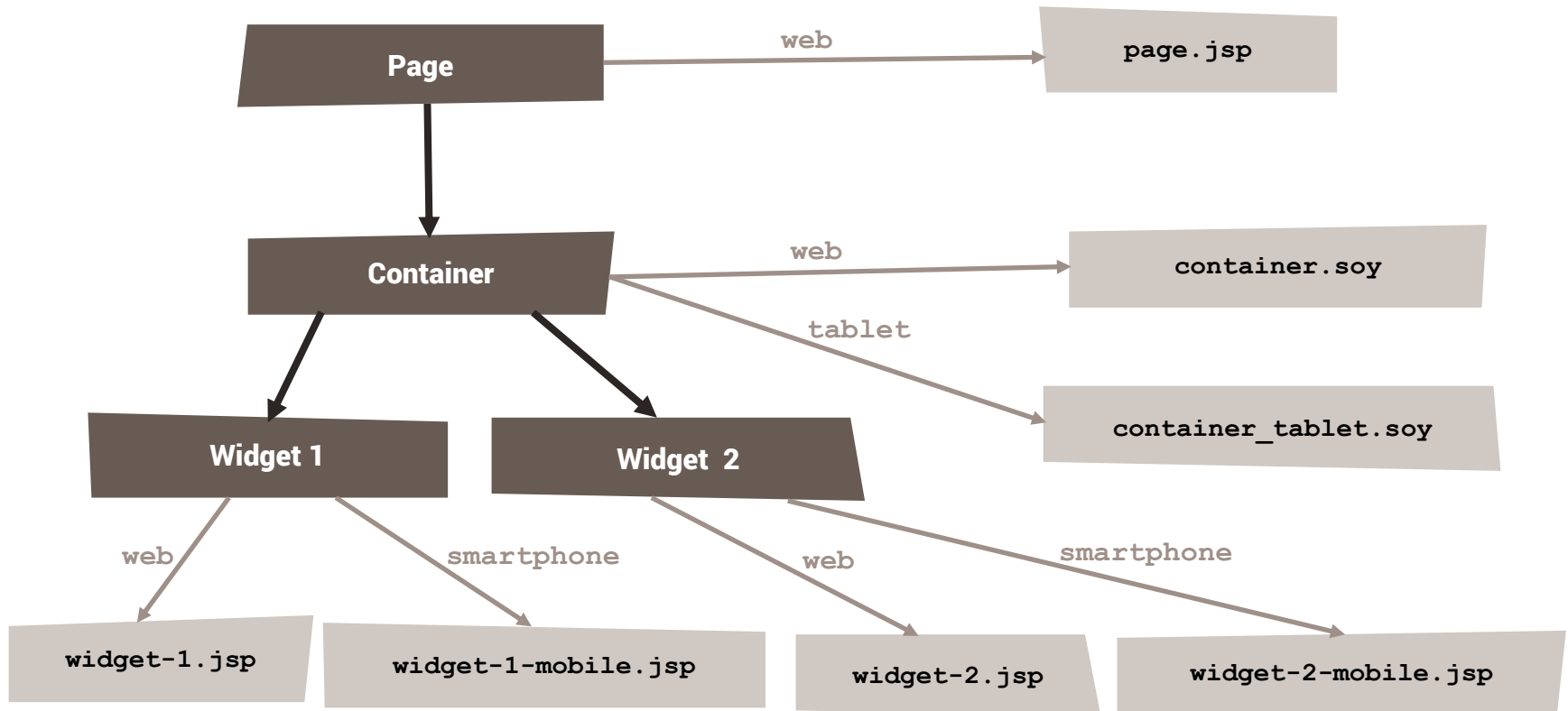 - ${item} contains data about the item and its children

BACKBASE

Template file rendering

| template file 1 ${item} | template file 2 ${item} | template file 2 ${item} |

Recognize Device → Template selection per model ← Portal Templates

User Credentials → Authorization & Customization ← User Customization

Initial portal model selection

Portal Database

44

BACKBASE

Item	Inherited Item	Customized Item	${item}
Property 1			Property 1
Property 2	Property 2'	Property 2"	Property 2"
Property 3			Property 3
	Property 4		Property 4

```
┌─────────────────────┐
│  1. Portal default  │
│       device        │
└─────────────────────┘
          │
          ▼
┌─────────────────────┐
│     2. Device       │
│    recognition      │
└─────────────────────┘
          │
          ▼
┌─────────────────────┐
│ 3. Template default │
│       device        │
└─────────────────────┘
          │
          ▼
┌─────────────────────┐
│  4. Find template   │
└─────────────────────┘
```

- 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.
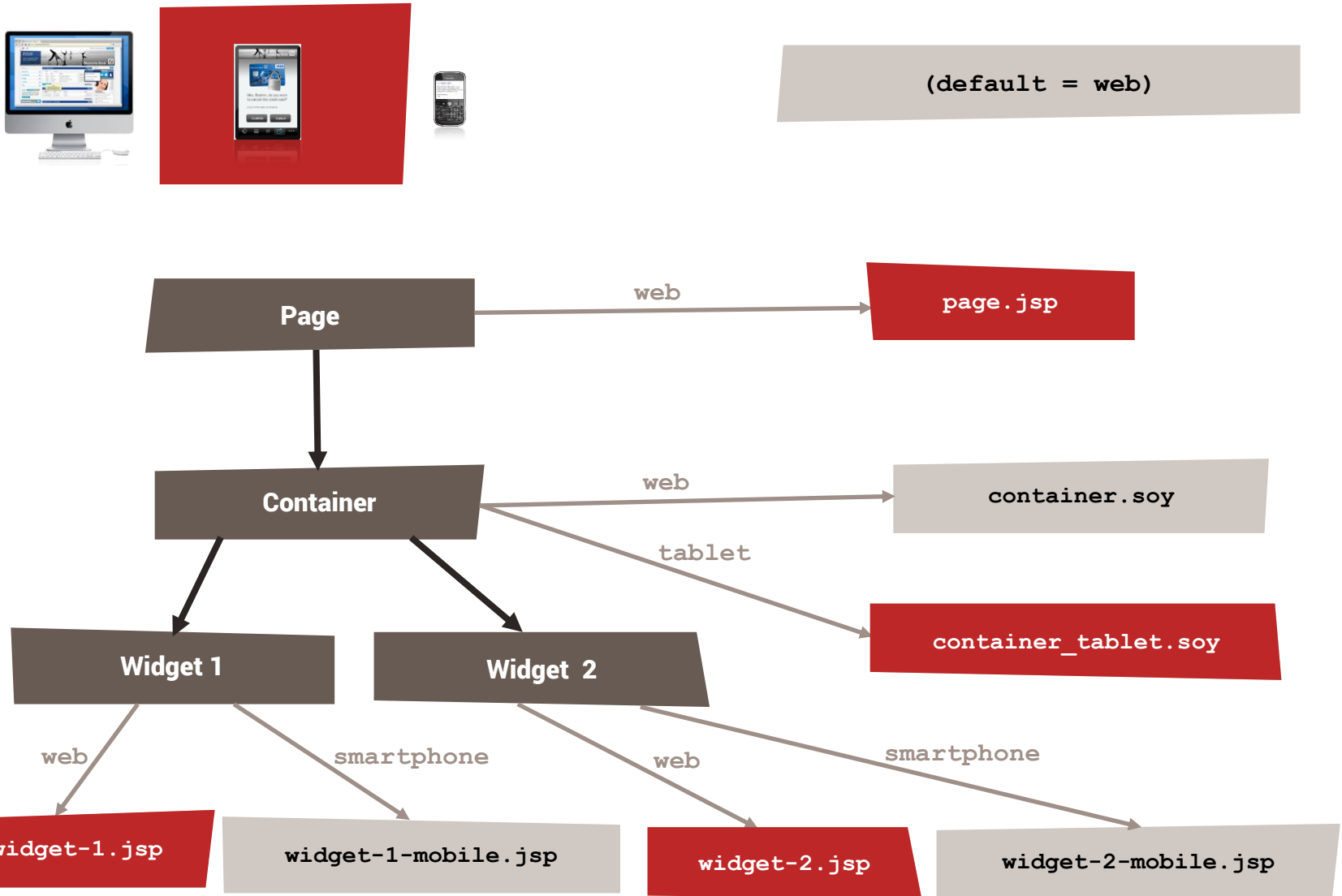
- Device recognition:
  - If the User-Agent header matches a device, the device configuration specifies a device type and a fall-back device type
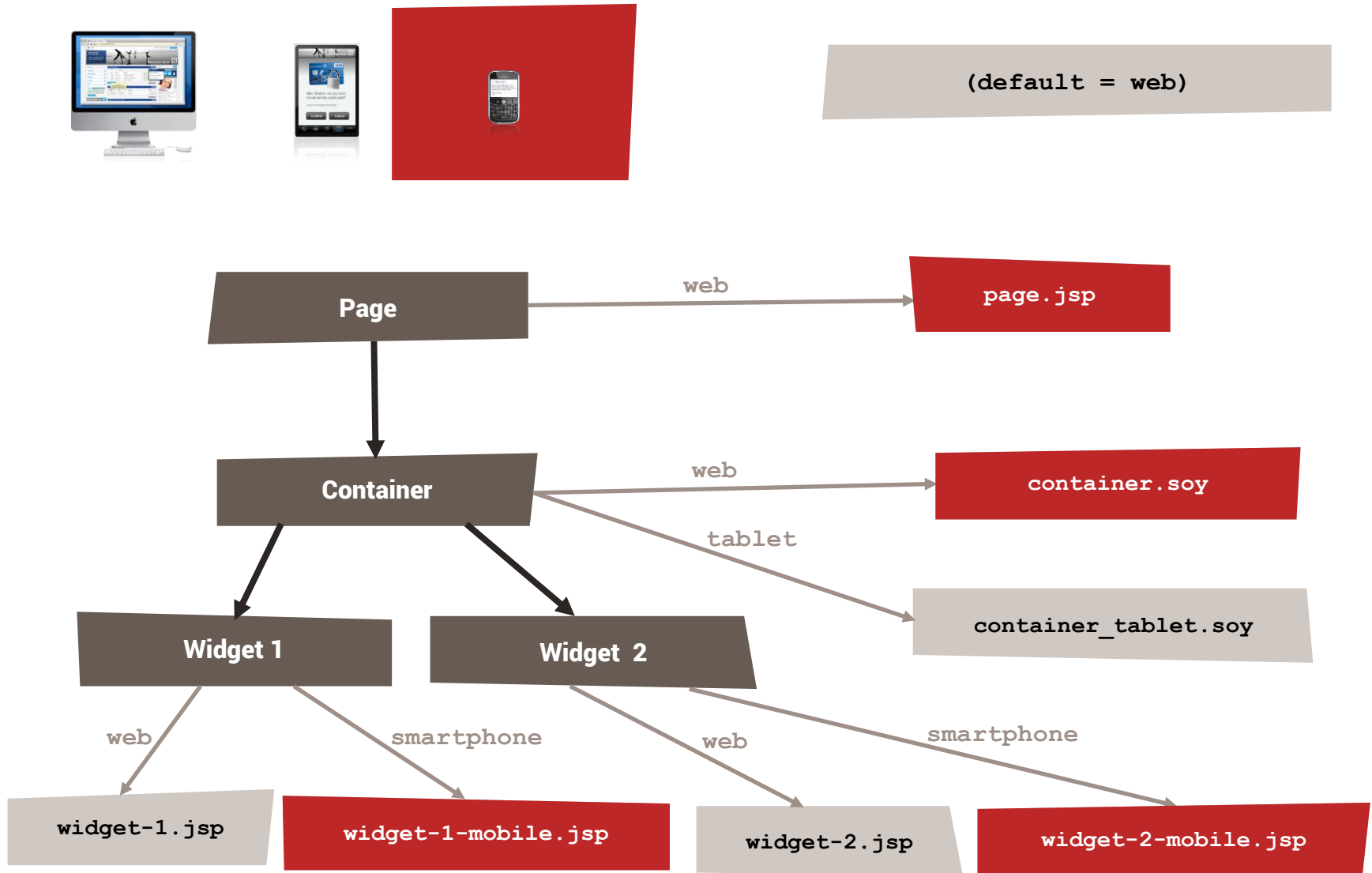
```
<device type="iPhone" fallbackTypes="Smartphone">
```

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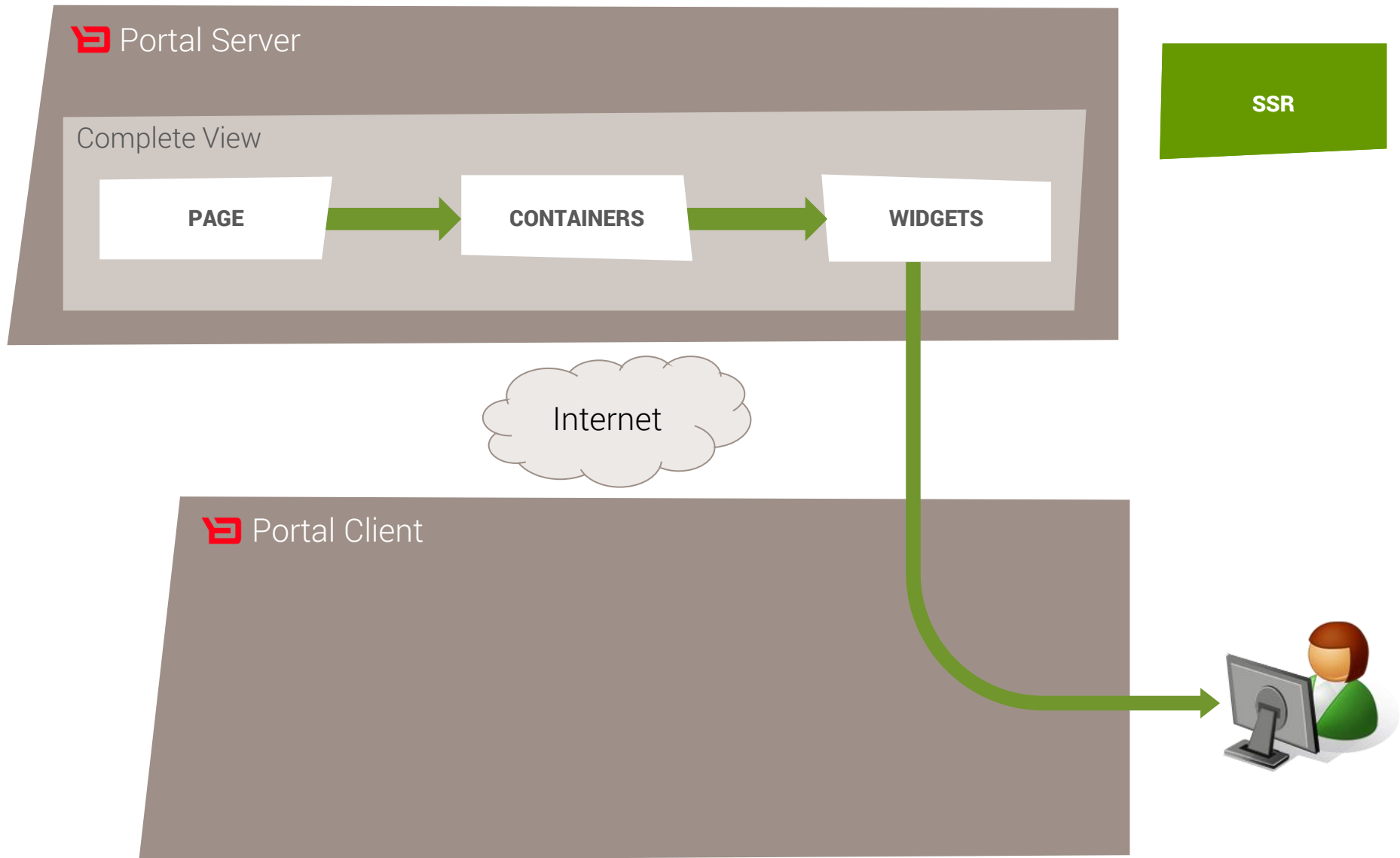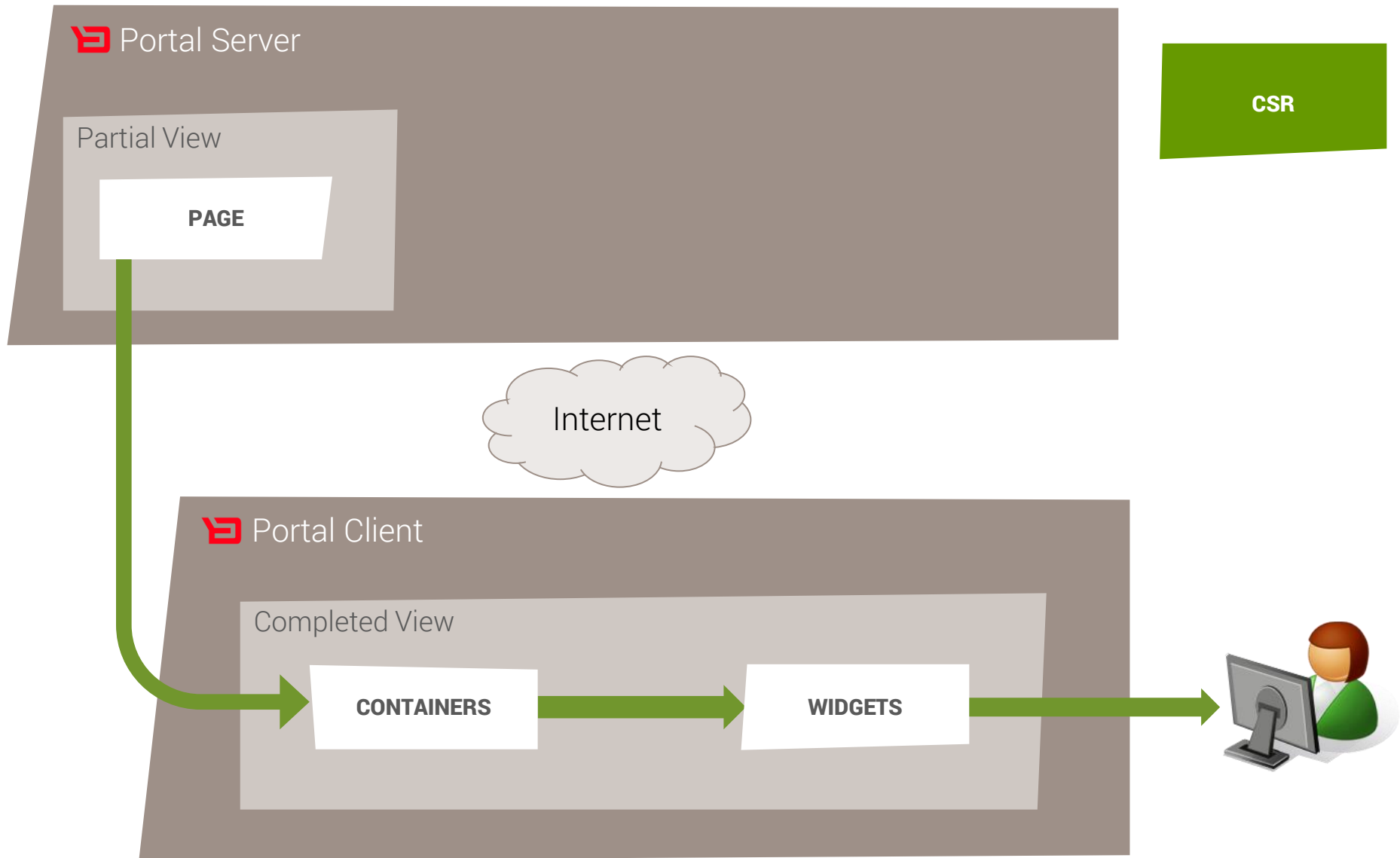 - Portal Foundation looks for a template property with name matching the fall-back device type

BACKBASE

(default = web)

Page — web → `page.jsp`

Container — web → `container.soy`

Container — tablet → `container_tablet.soy`

Widget 1 — web → `widget-1.jsp`

Widget 1 — smartphone → `widget-1-mobile.jsp`

Widget 2 — web → `widget-2.jsp`

Widget 2 — smartphone → `widget-2-mobile.jsp`

BACKBASE

(default = web)

Page → web → `page.jsp`

Container → web → `container.soy`

Container → tablet → `container_tablet.soy`

Widget 1
- web → `widget-1.jsp`
- smartphone → `widget-1-mobile.jsp`

Widget 2
- web → `widget-2.jsp`
- smartphone → `widget-2-mobile.jsp`

(default = web)

Page → web → `page.jsp`

Container → web → `container.soy`

Container → tablet → `container_tablet.soy`

Widget 1 → web → `widget-1.jsp`

Widget 1 → smartphone → `widget-1-mobile.jsp`

Widget 2 → web → `widget-2.jsp`

Widget 2 → smartphone → `widget-2-mobile.jsp`

BACKBASE

- 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

BACKBASE

Portal Server

SSR

Complete View

| PAGE | → | CONTAINERS | → | WIDGETS |

Internet

Portal Client

BACKBASE

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
  - JavaServer Pages — extension is jsp
- Widgets:
  - JSP, HTML, 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)

BACKBASE

| W | W | W | Catalog |

P
C
C — W
C — W

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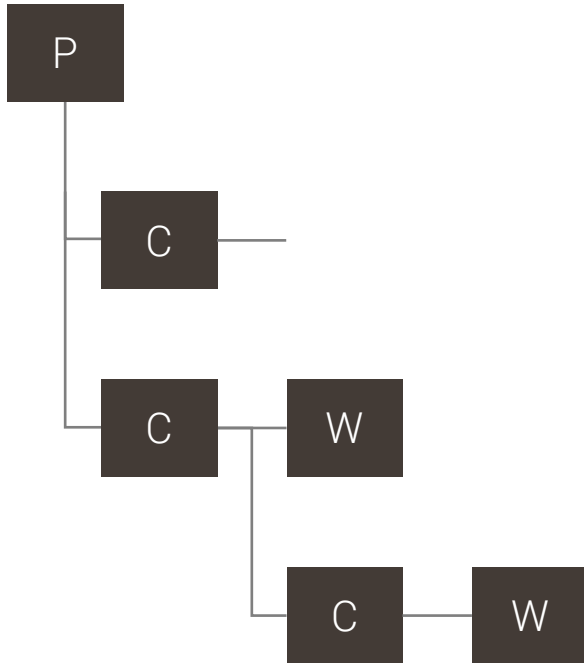
| W | W | W | Base Elements (eg. Widgets/Templates) |

P

C — W ◄------

C — W

C — W

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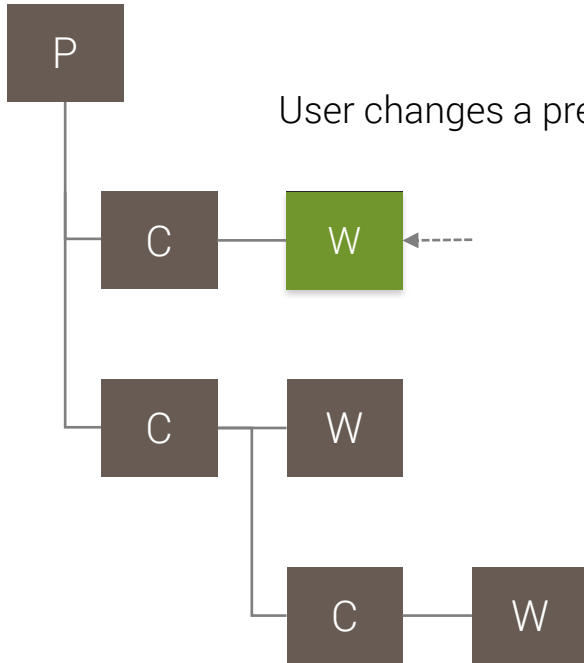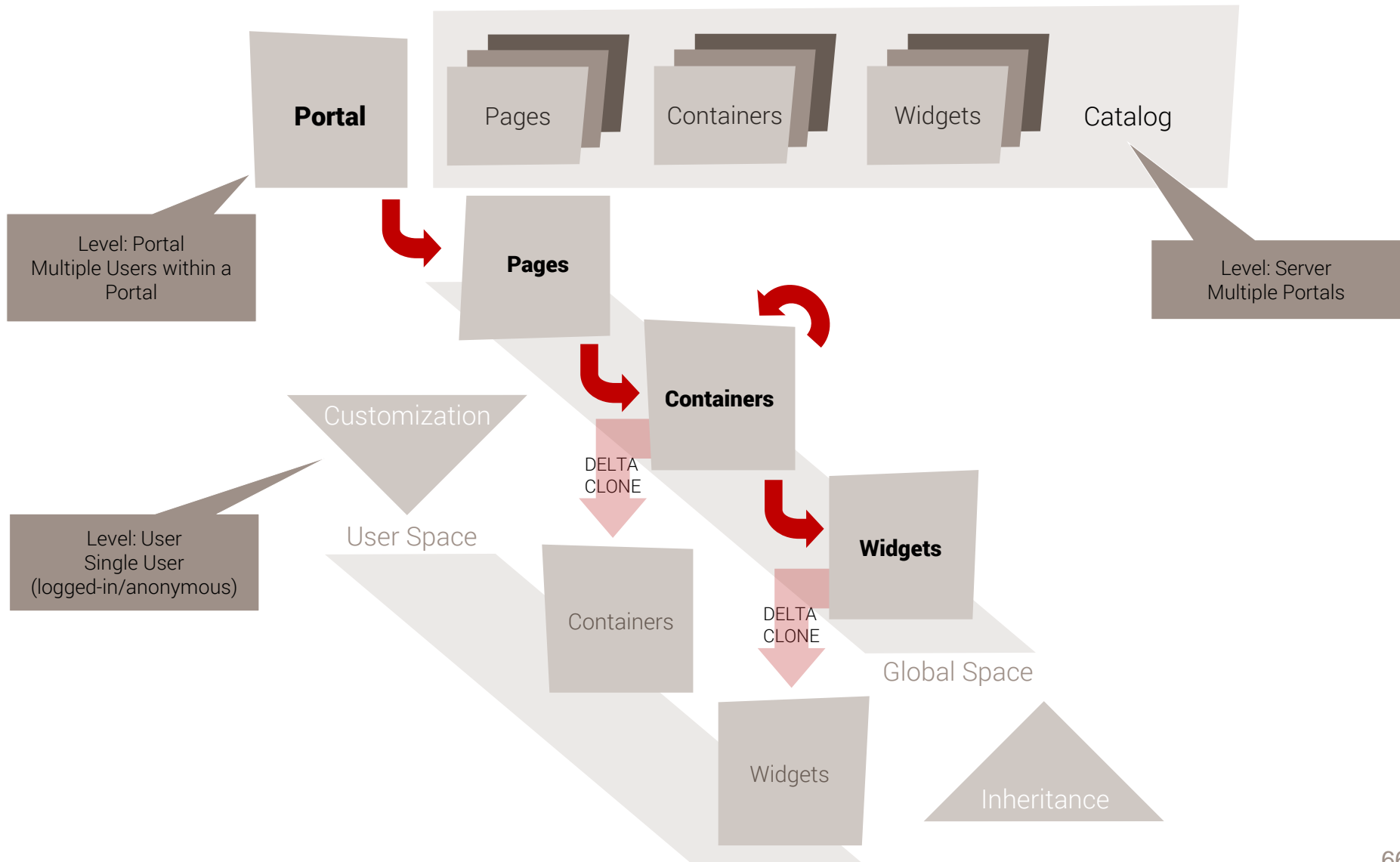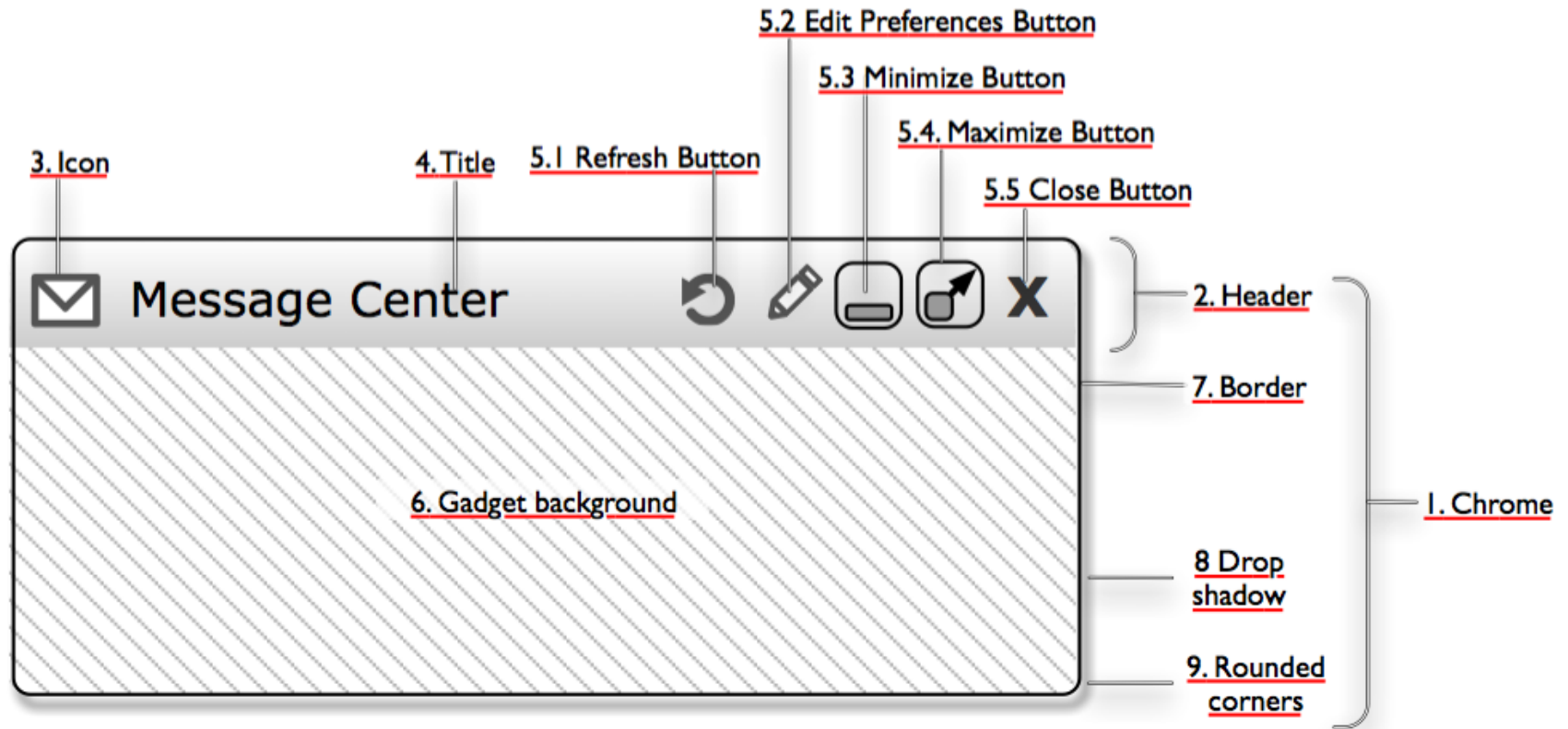
Lets look at how the CXP Manager handles the
concepts we have just discussed

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
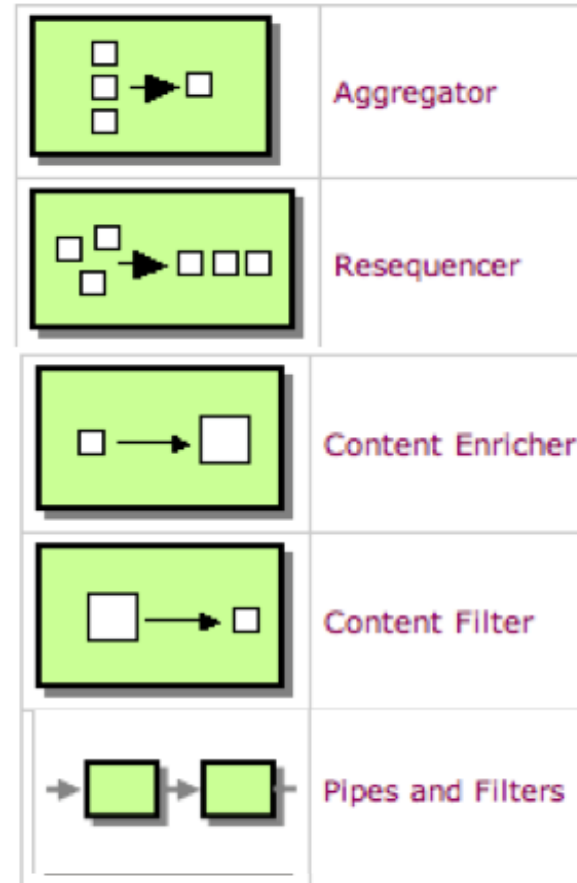- – Combine data sources
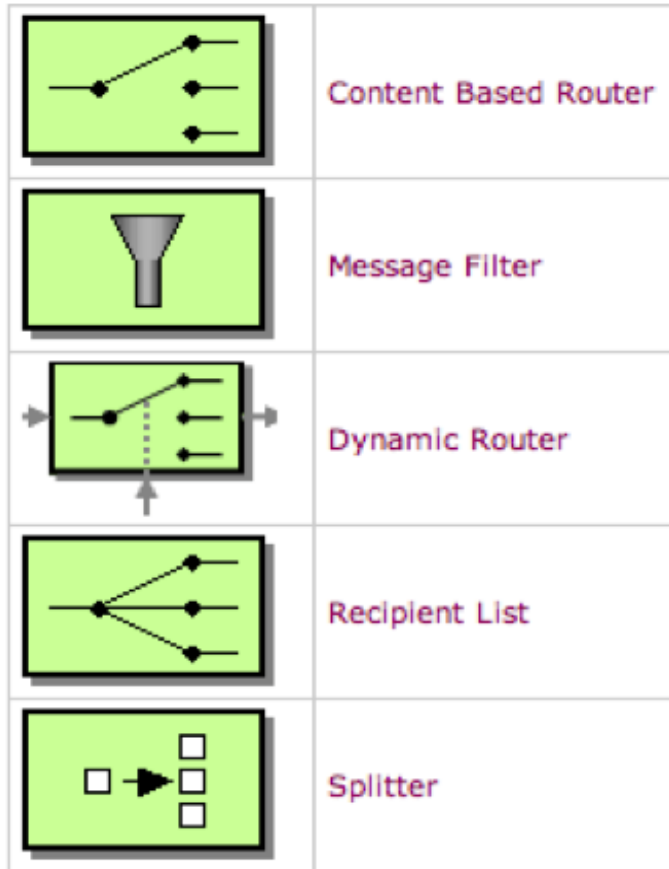- – Renditions
  - • PDF
  - • Excel
  - • HTML

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**
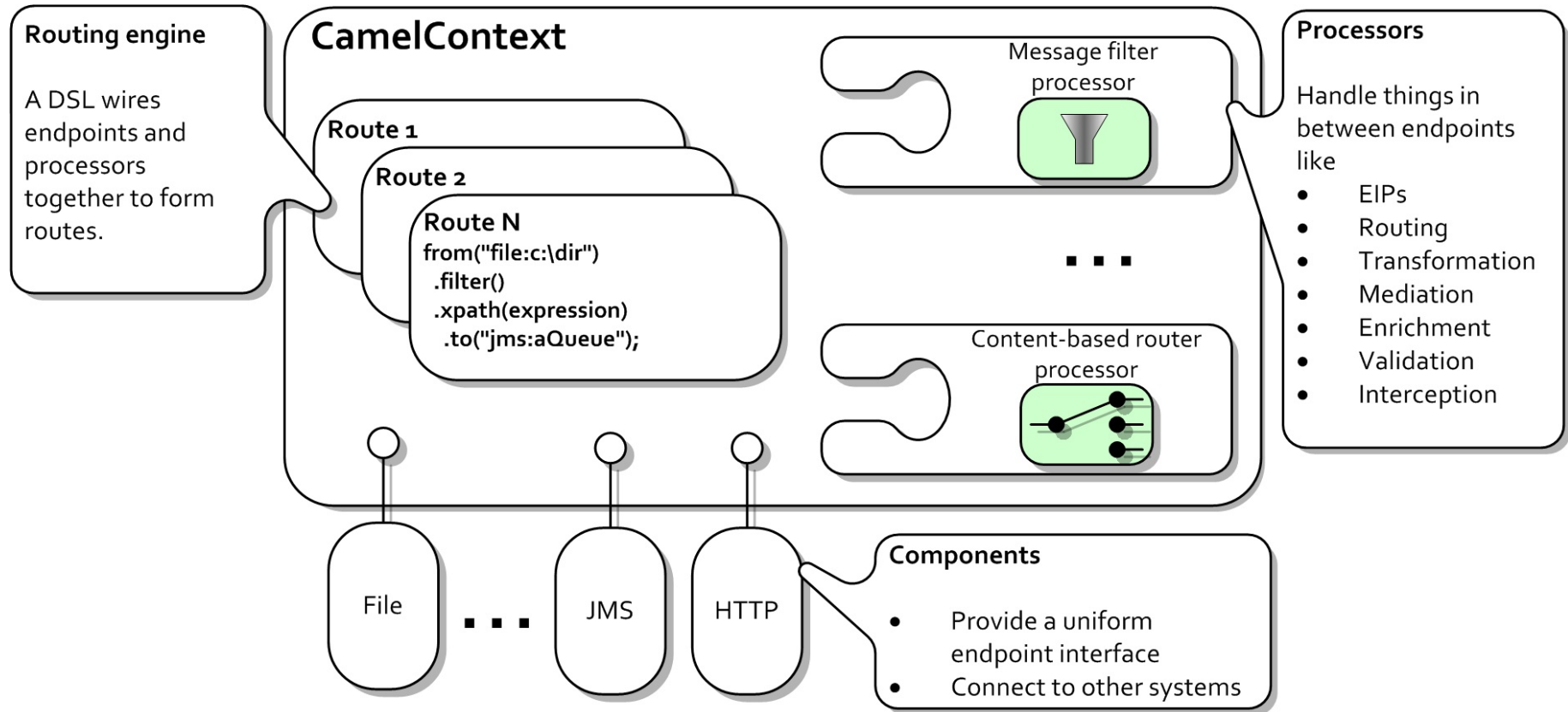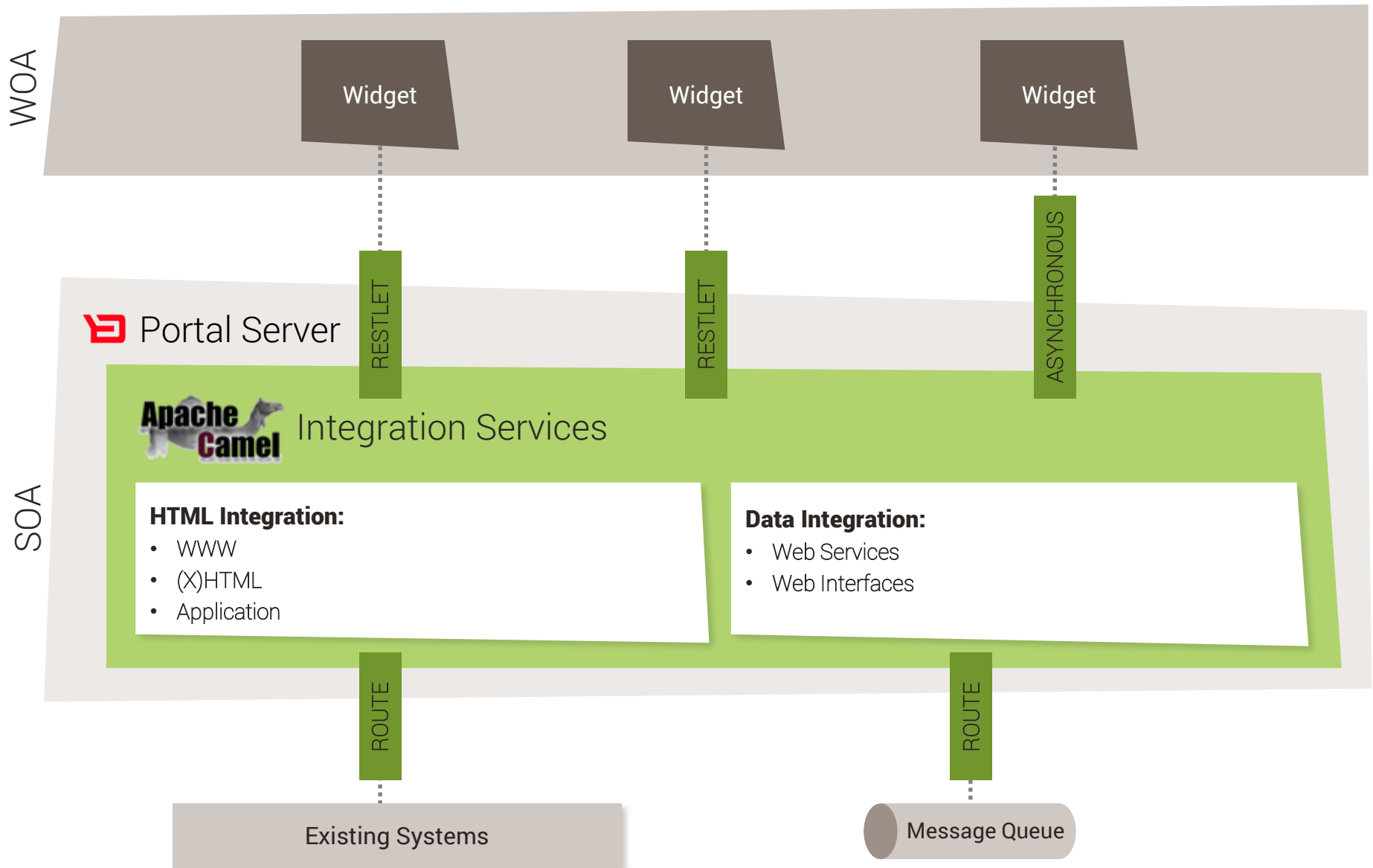 - Commercial offerings
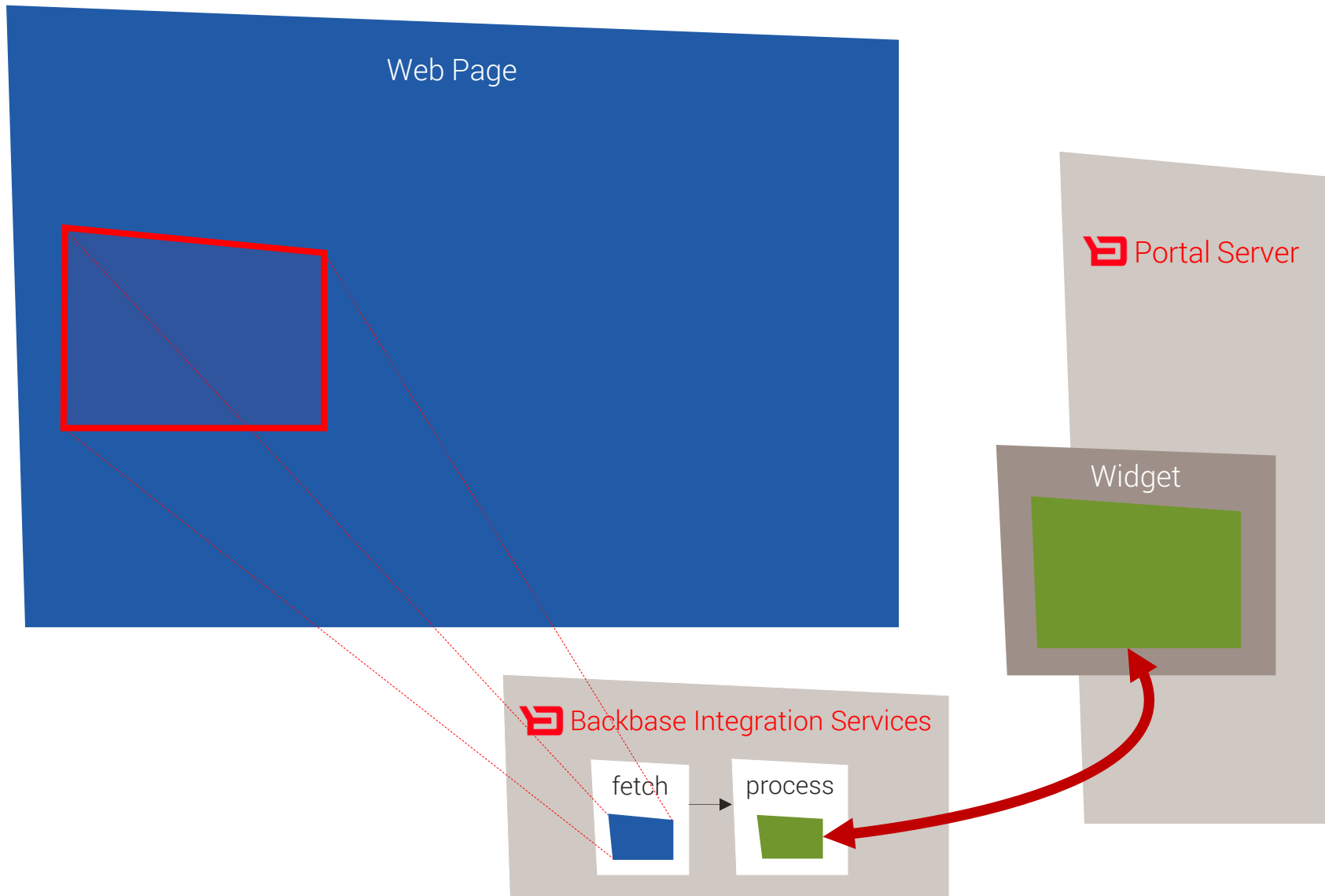
http://camel.apache.org/eip
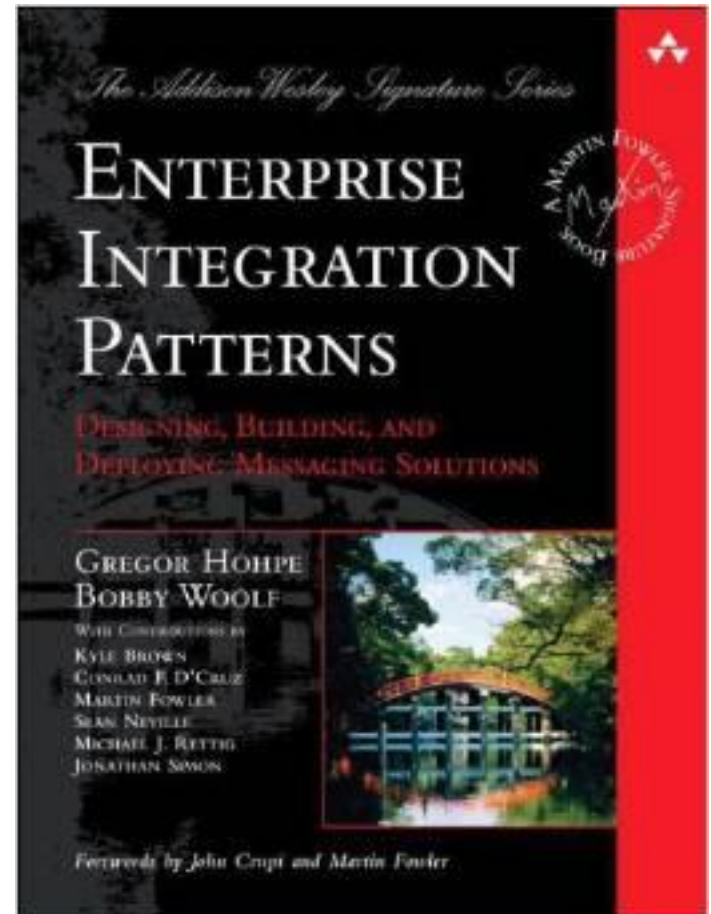
- Light weight integration library
- Kind of an embedded ESB
- Enterprise Integration Patterns
- Domain Specific Language
- Routing and Mediation
- Components
- Built in Transformers
- Active Community
- Easy to extend

Concise
Application
Messaging
Exchange
Language

BACKBASE

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

75

BACKBASE

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

76

BACKBASE

Web Page

Portal Server

Widget

Backbase Integration Services

fetch

process

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 Backbase CXP optional 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

BACKBASE

Server

Backbase CXP Foundation

Content Services

File System

Repositories
- Content
- Portal
- Configuration
- Resources

CMIS

WebDAV

JMX

Remote Application

Importers
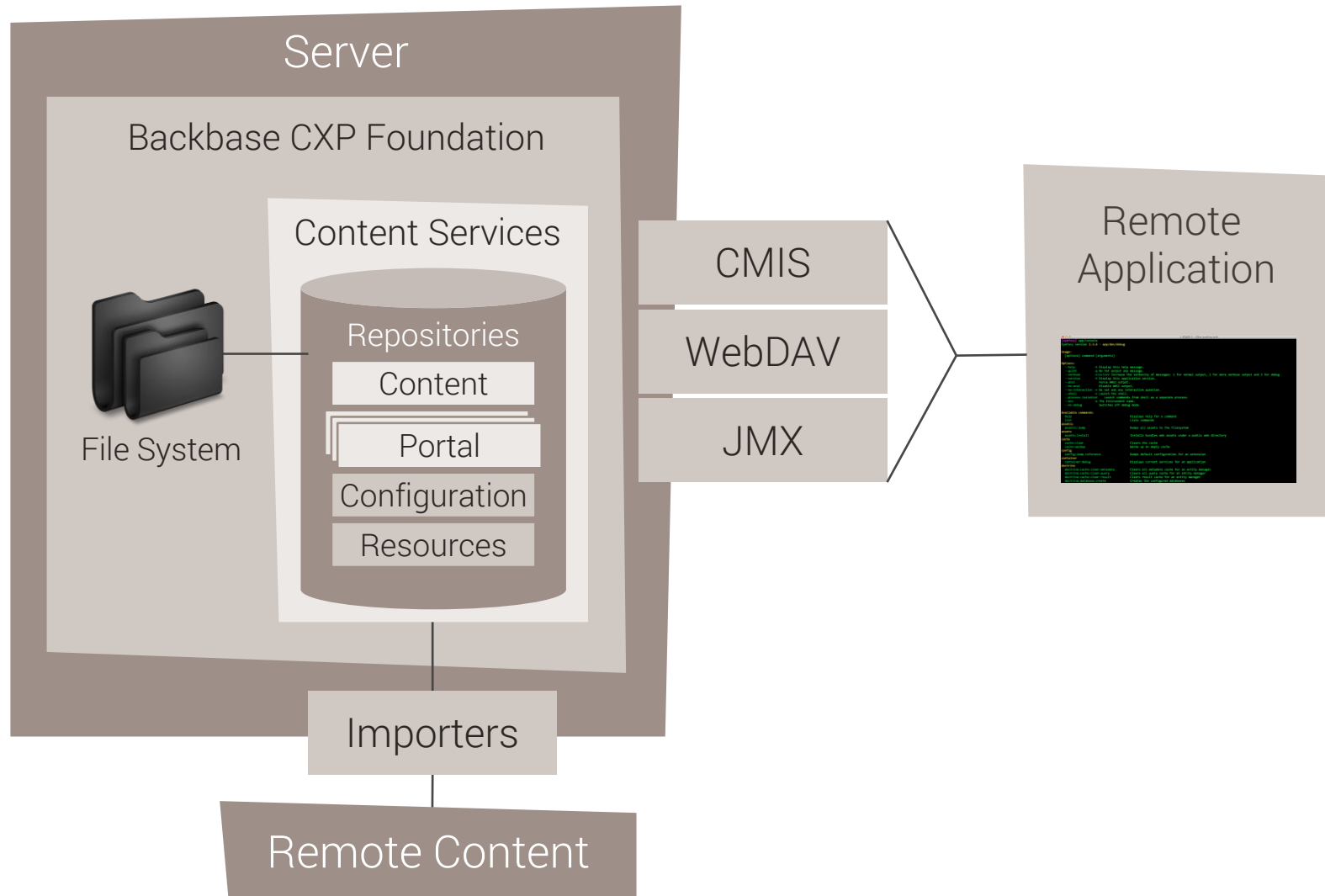
Remote Content

- 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.

BACKBASE

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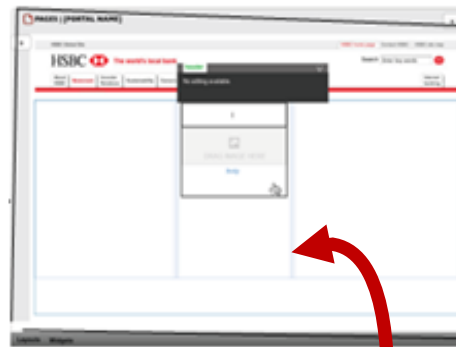 - Shared (or global)
  - Portal-specific

BACKBASE

CONTENT AUTHORING
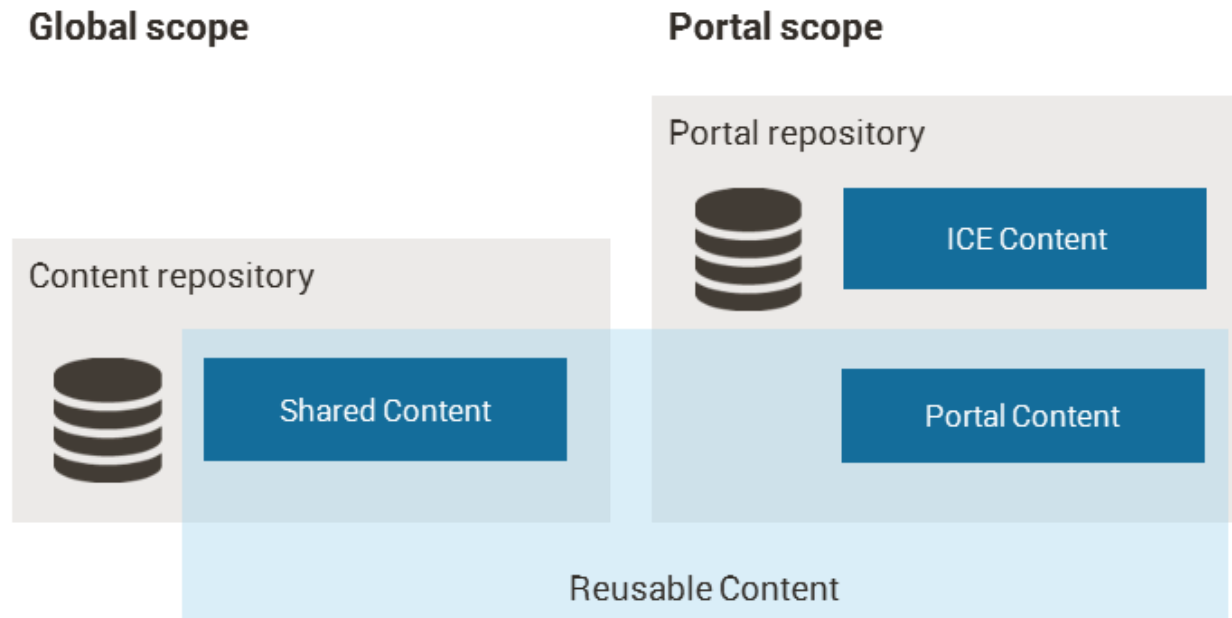
(Existing CMS)

PAGE EDITING

VISITORS

CMS Authoring GUI

CMIS interface

CMS Content Repository

CMIS interface

2

Portal Content Repository

1 Import

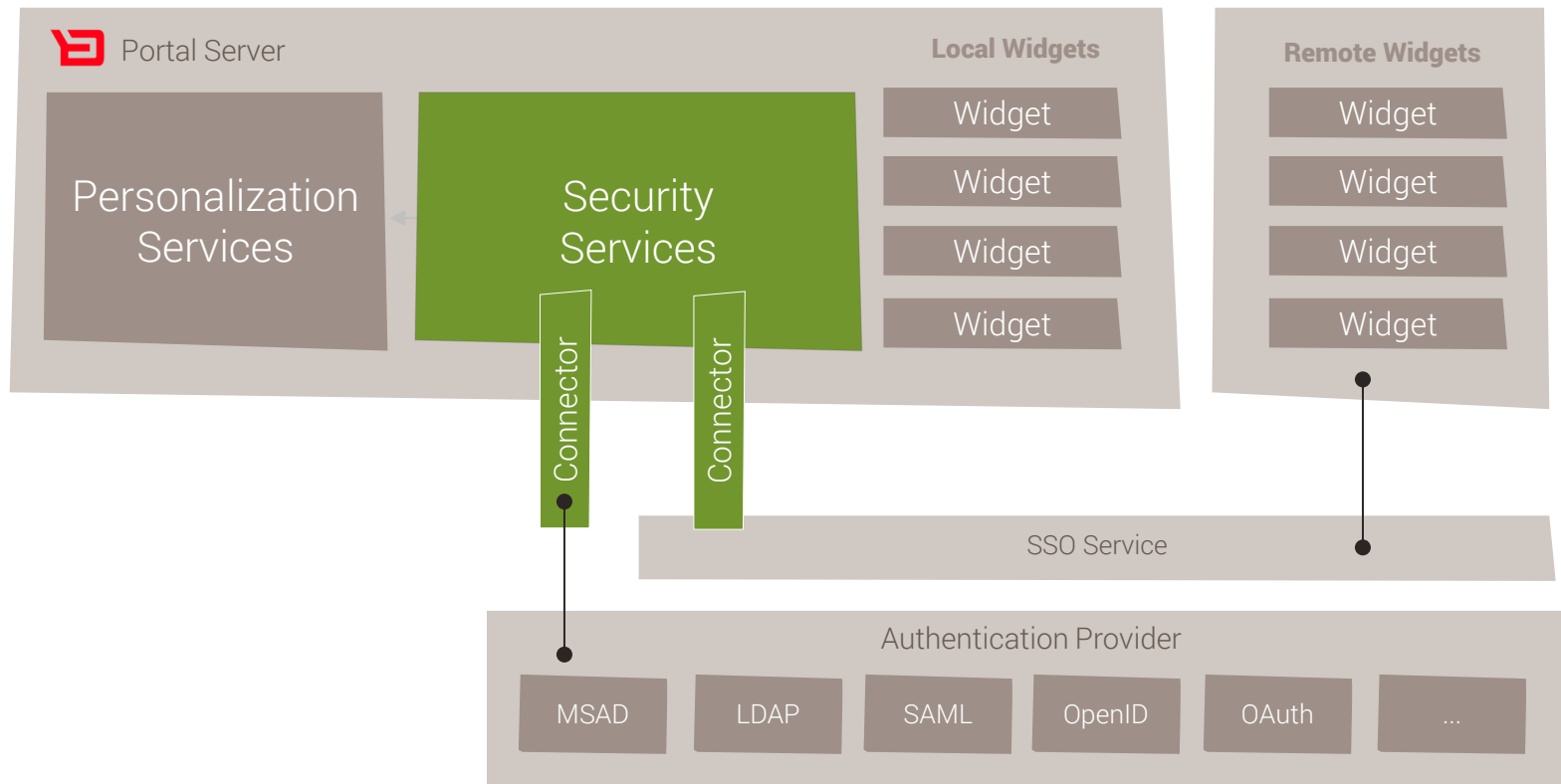staging / live

CMIS interface

Portal Content Repository
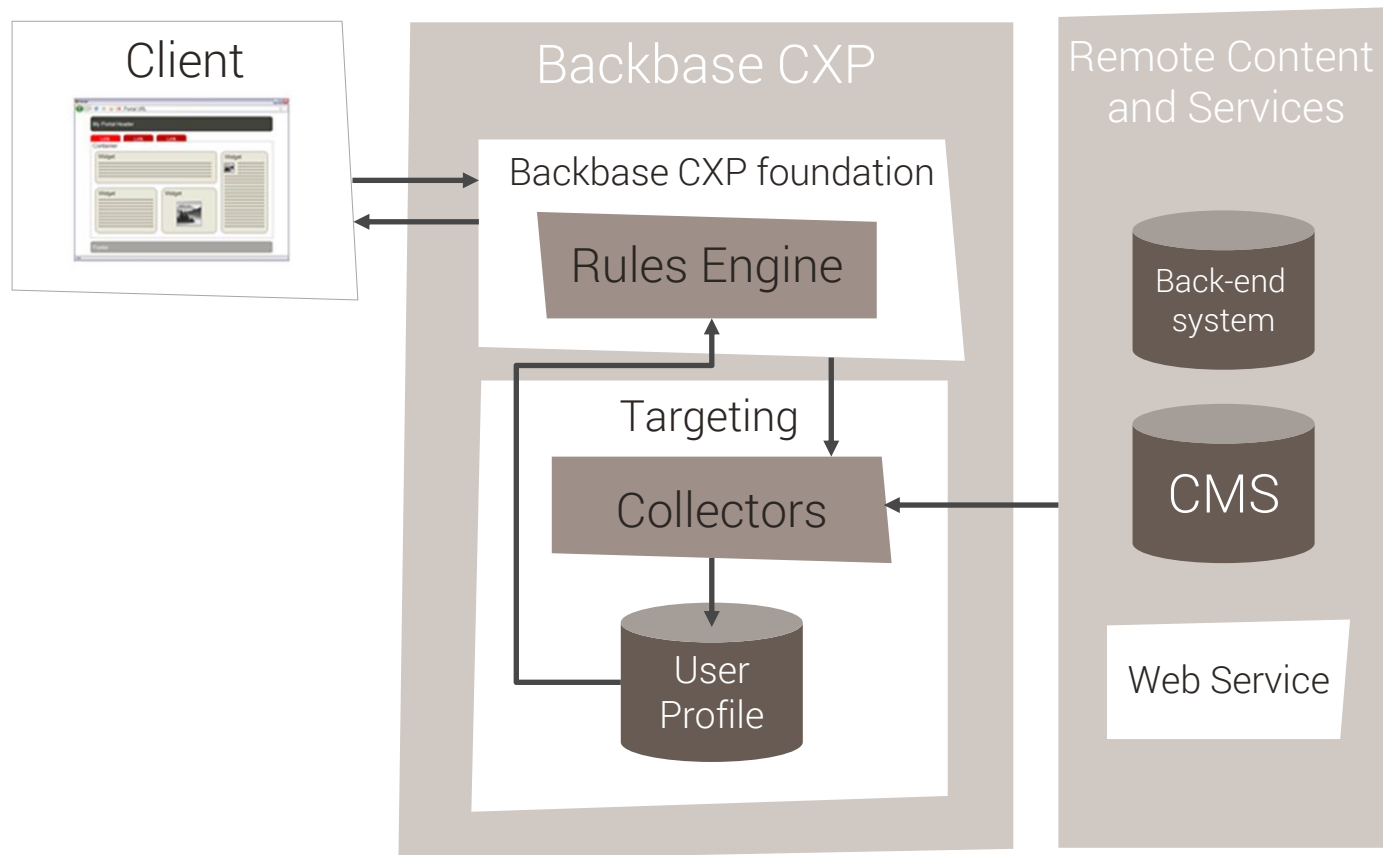
3

96

BACKBASE

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

- 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
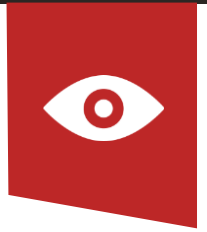
BACKBASE

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.

Client

Backbase CXP

Remote Content and Services

Backbase CXP foundation

Rules Engine

Targeting

Collectors

User Profile

Back-end system

CMS

Web Service

- Start the portal server and create a new page in the training portal

- Select the targeting container from the layouts tab and drag it onto the new page

- Add an alternative

- Add a segment or a condition (for example, use the Chrome browser segment defined by the session collector).

- Add an image viewer widget instance to each alternative but drag different images onto them.

- Test the page on different browsers

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.
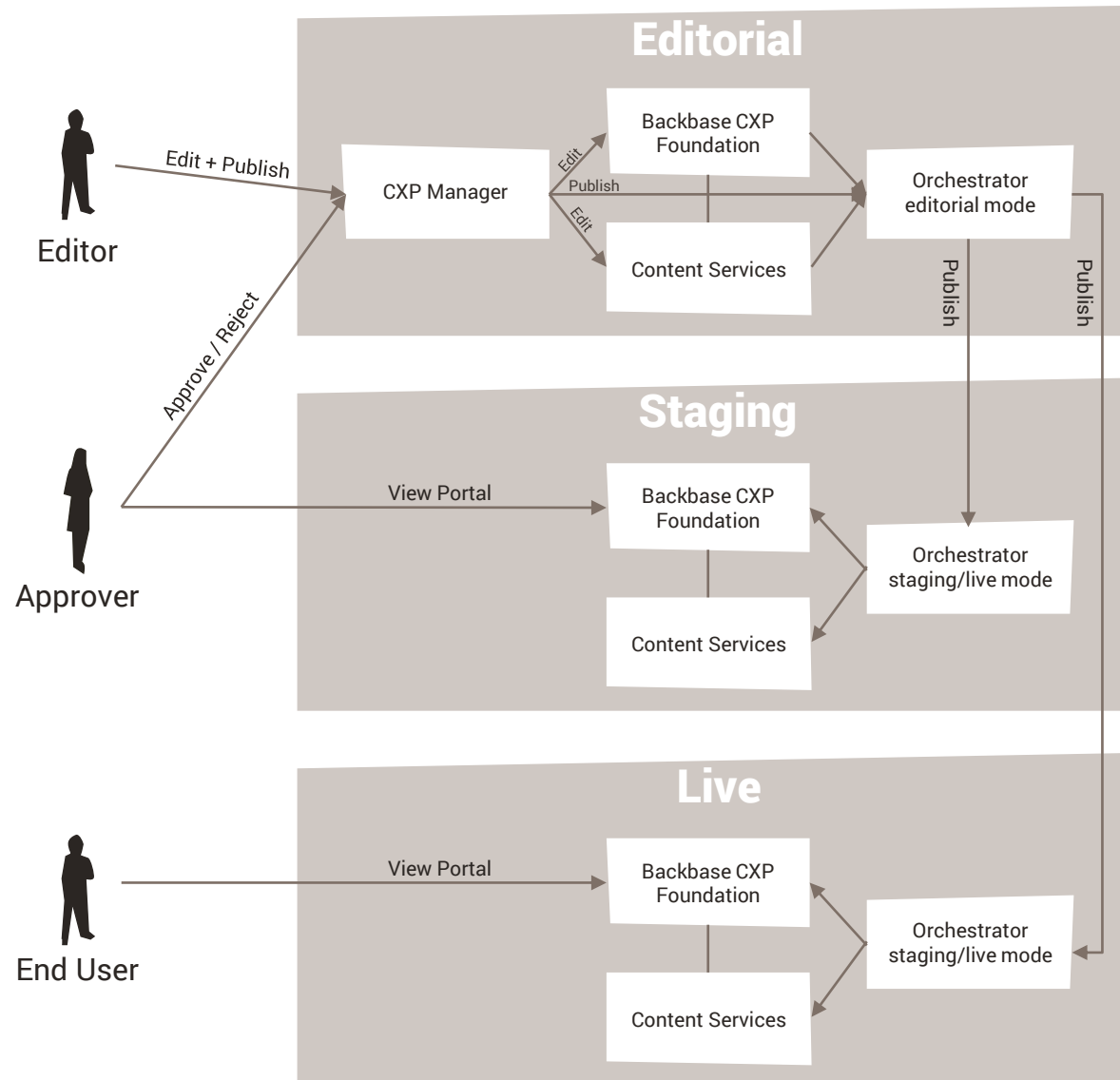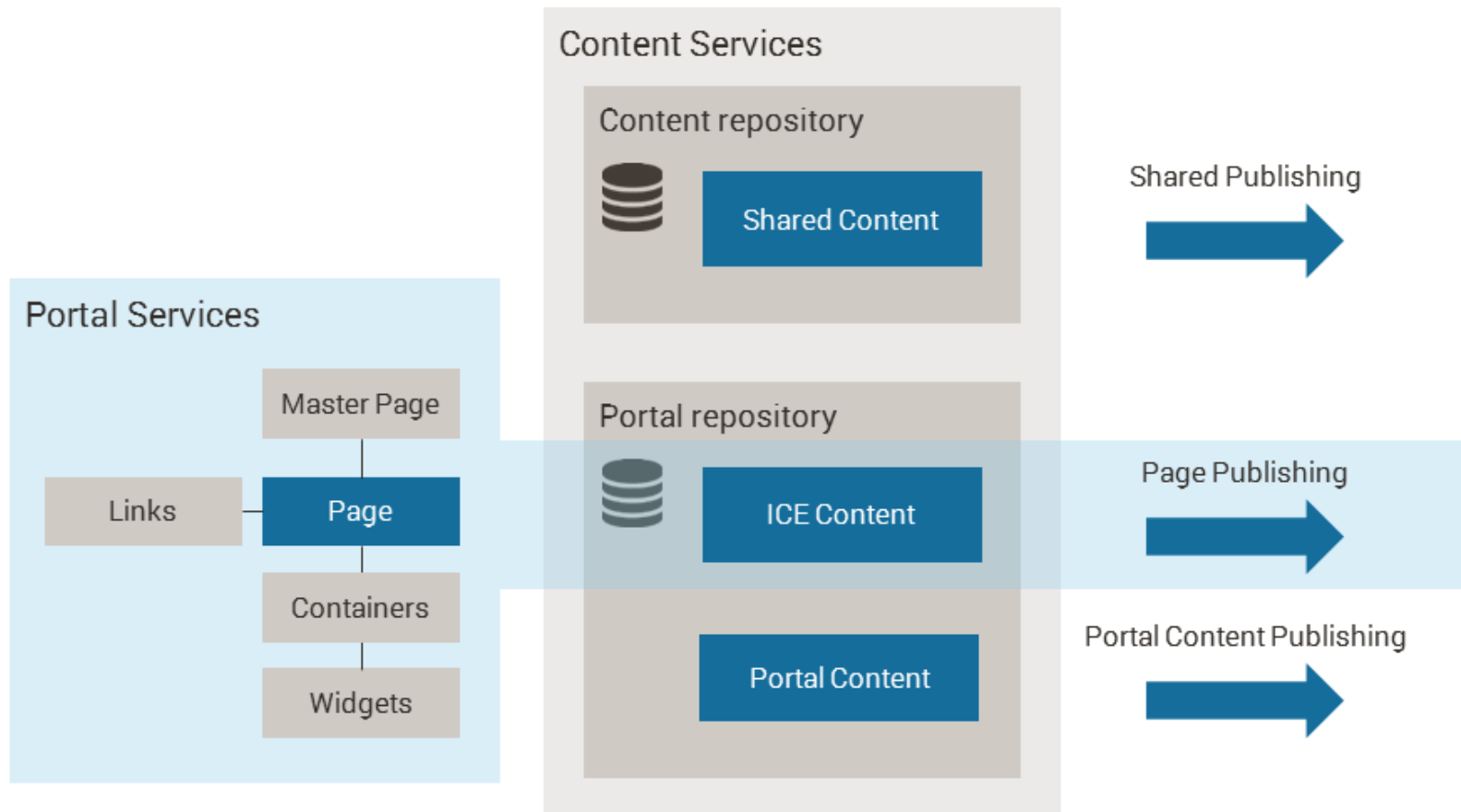
- Publishing takes place across different portals instead of environments
- For testing and demo purposes (configurable in the Orchestrator configuration file)
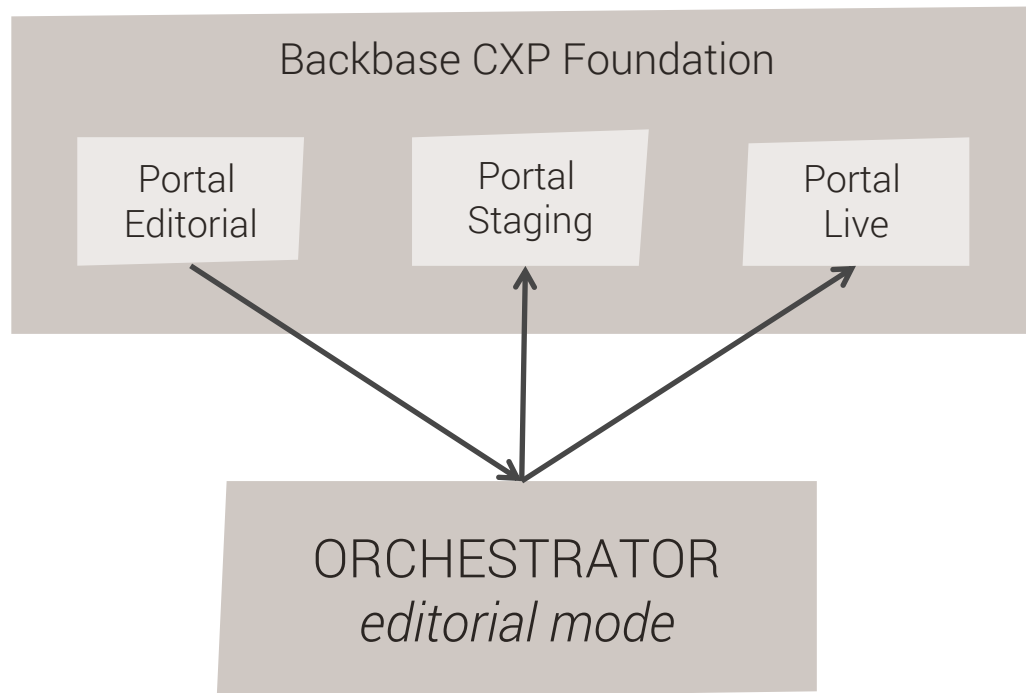
- *Orchestrator* has two roles that you can map to groups in Backbase CXP Foundation:
  - Publisher – Allows users to submit publish requests.
  - Approver – Allows *Approvers* to approve or reject publish requests.
- Note that Backbase CXP enforces the *four-eye principle*: a user who submits a publish request may not approve it

- Login to CXP Manager.
- Configure publisher and approver roles in CXP Manager.
- Create portals required for self-publishing.
- Create a new page in the editorial portal.
- Add the page to a publish package and publish it using the Direct publishing chain.

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

ΣΙ BACKBASE

Thank you!