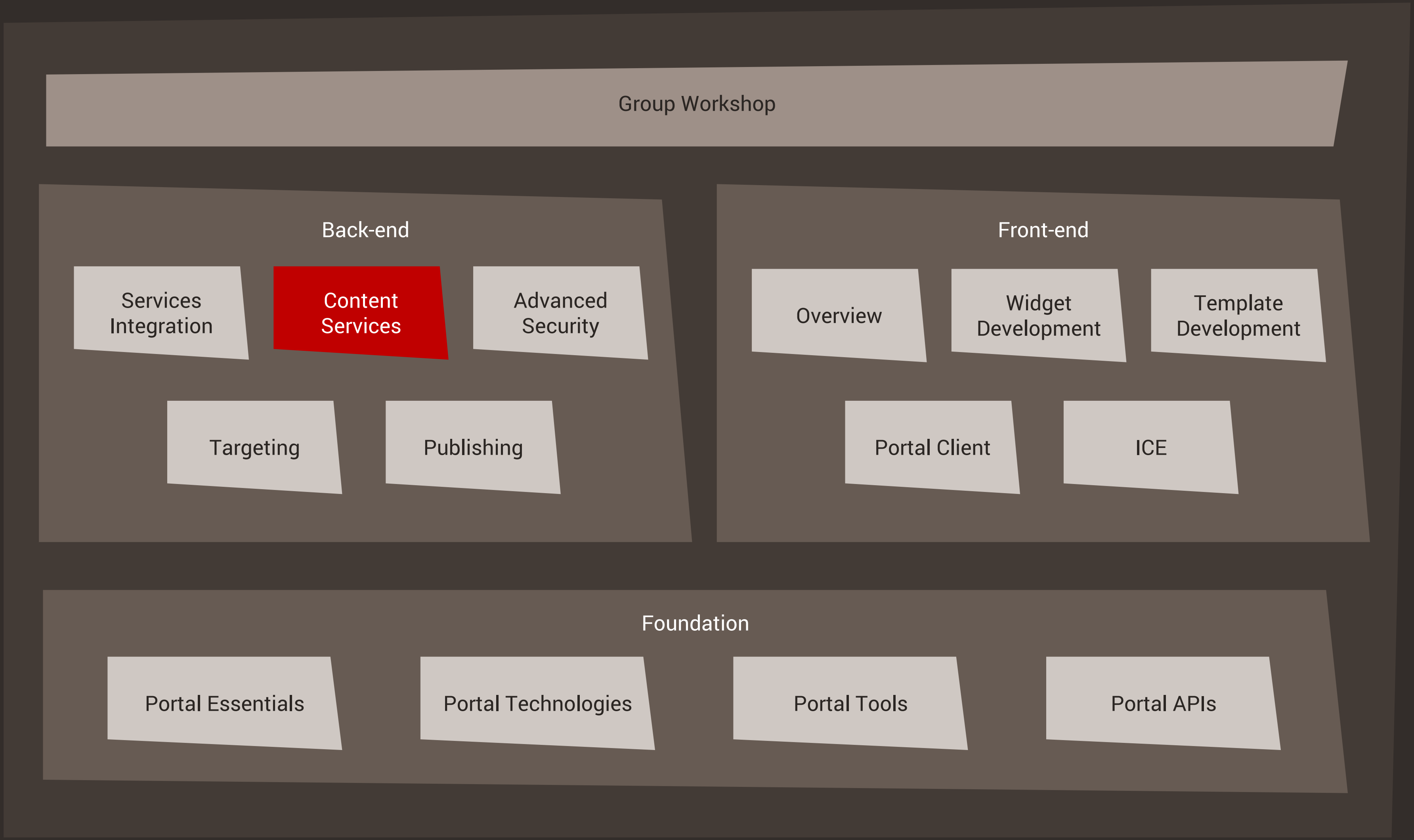
Content Services

Focus Area: Back-end

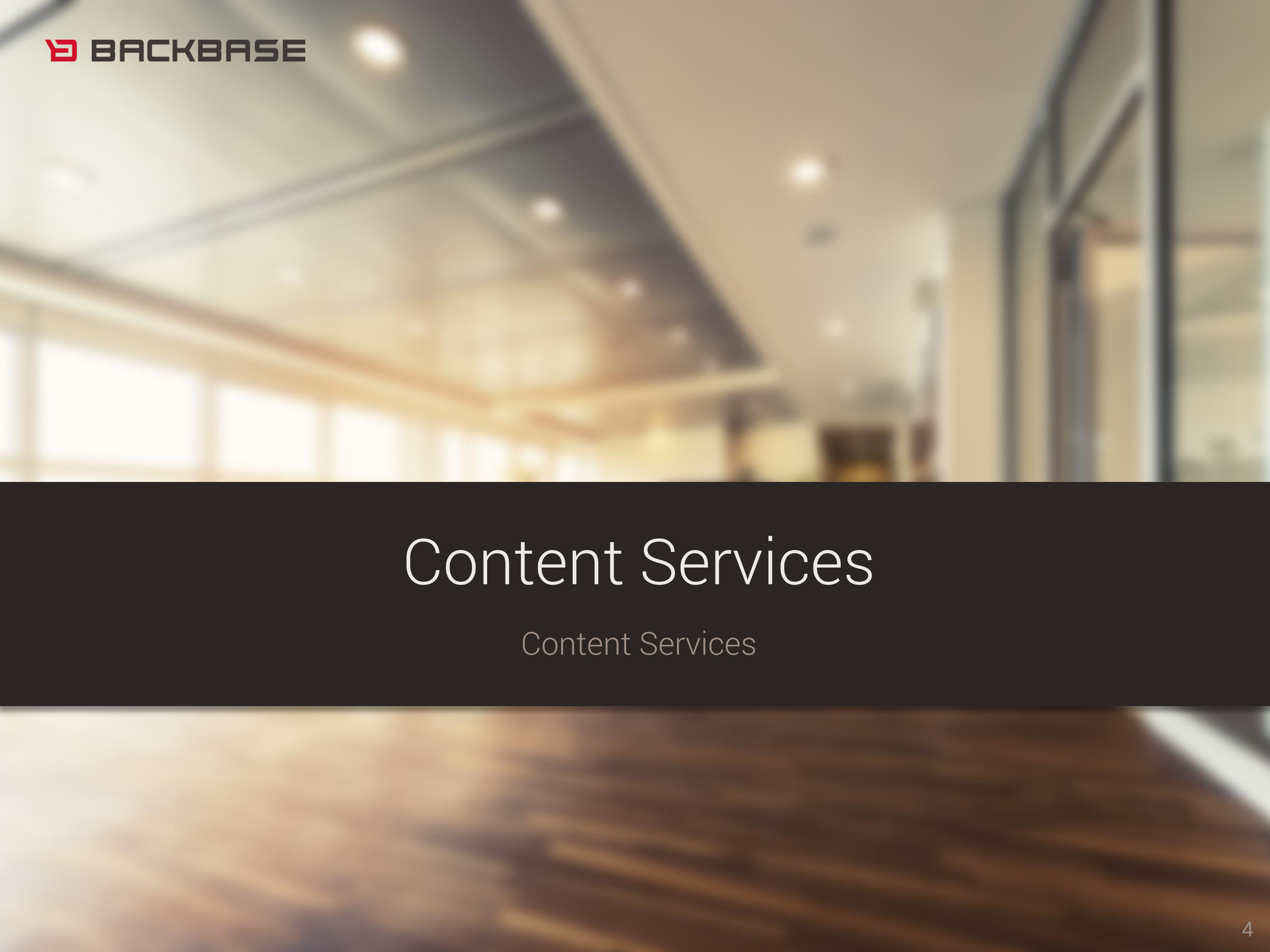
Training Overview



Objectives

1. Content Services Repositories
2. Learn how to work with the different interfaces
3. Learn how to customize content services
4. Content Service Importers and Validators
5. Integrate content services with portal foundation

BACKBASE



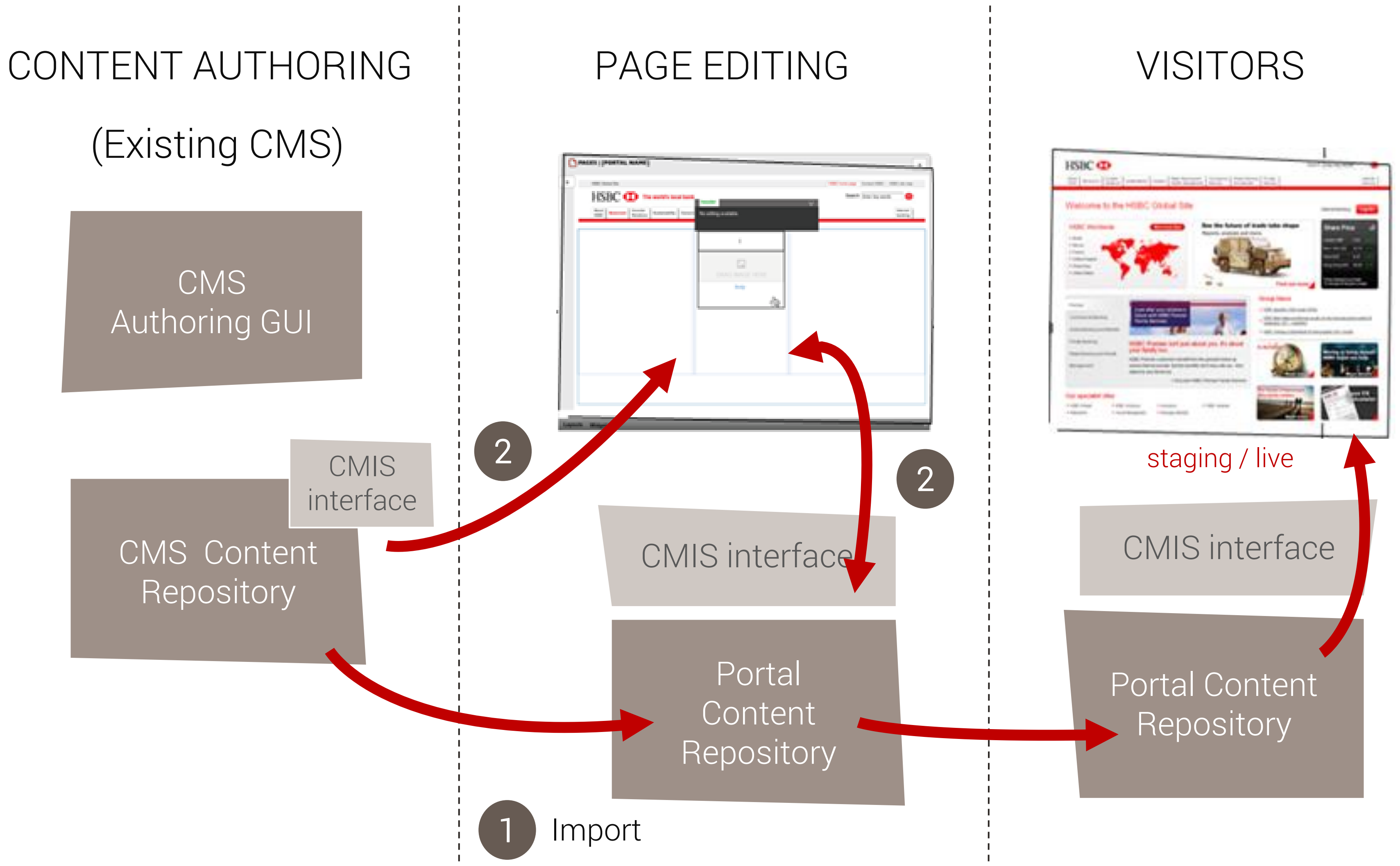
Content Services

Content Services

Content Services

- Content Services is a content delivery platform provided as a Backbase CXP optional module
- Content Services provides a 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 data and resources.

CMS + Portal Content Services + Page Editing

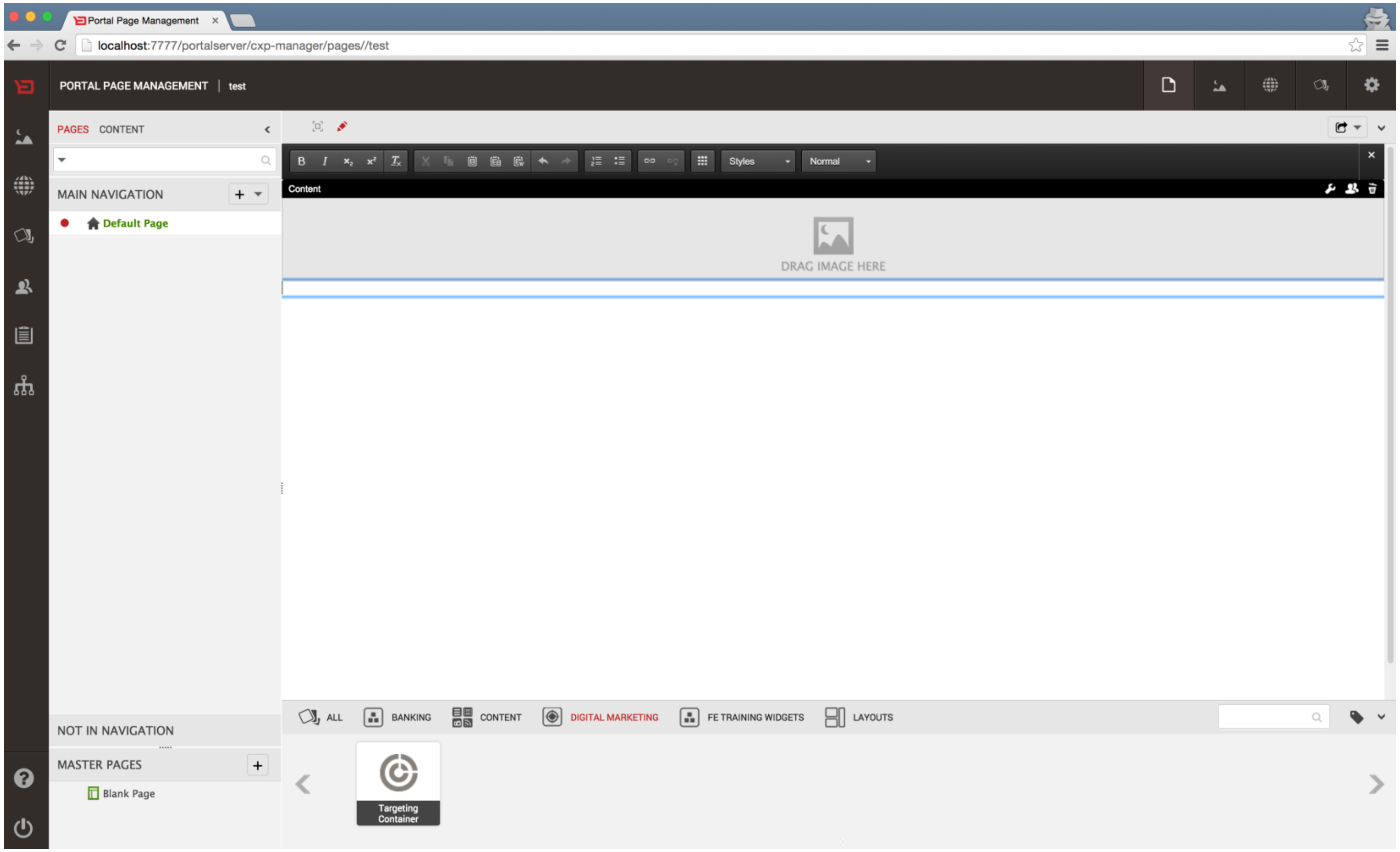


Kinds of Content Used in Portal Pages

- 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.

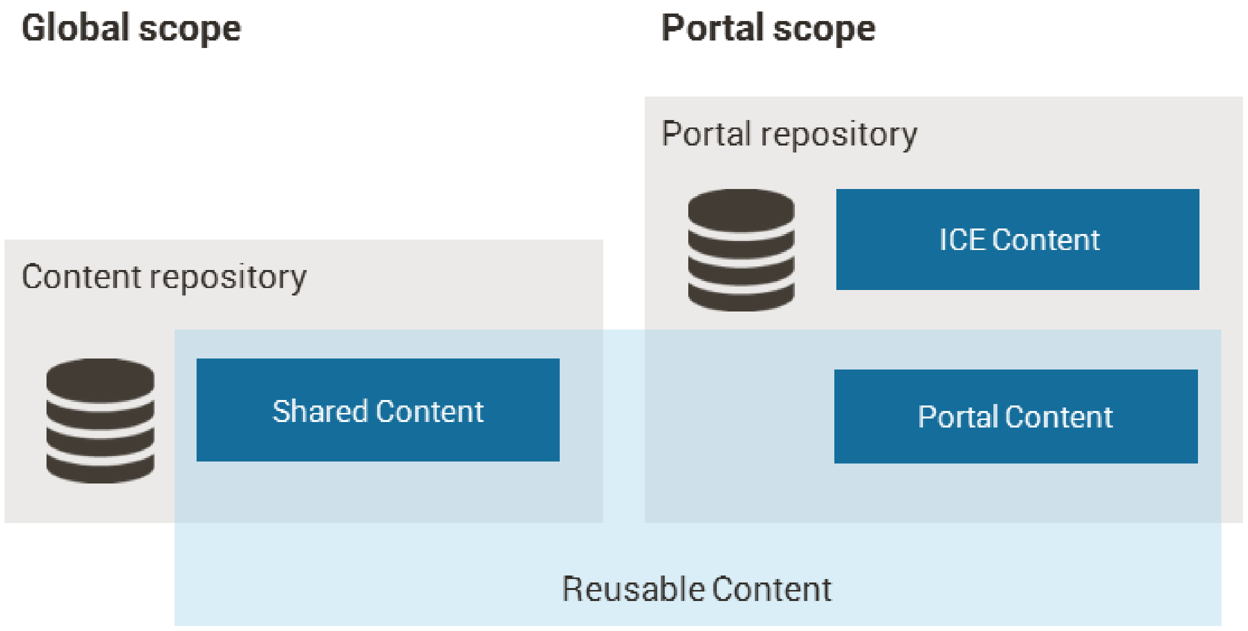
In Content Editing (I.C.E.)

- Stores "In Context Editing" (a.k.a. ICE) content from portal manager.

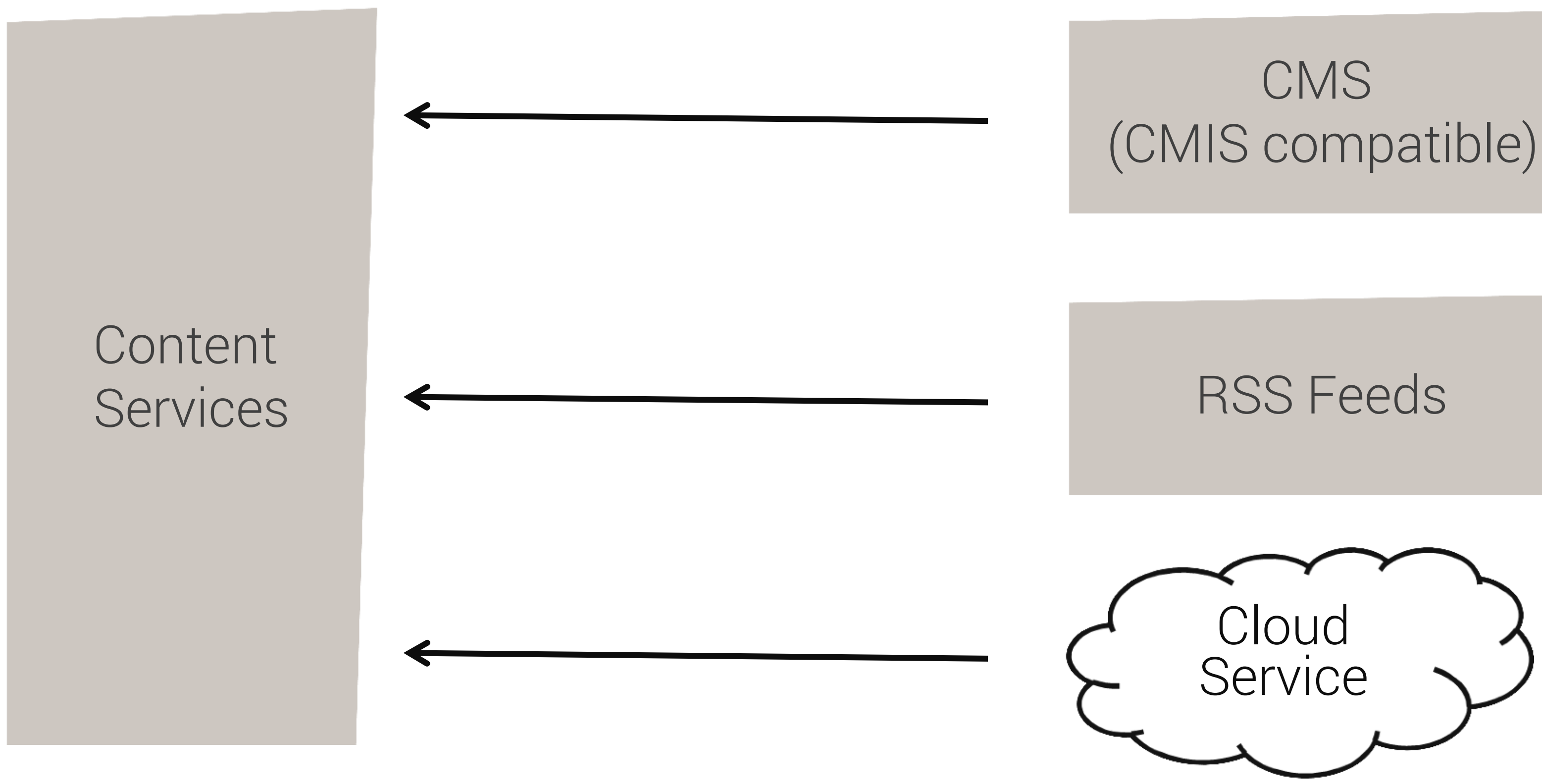


Scope of Reusable Content

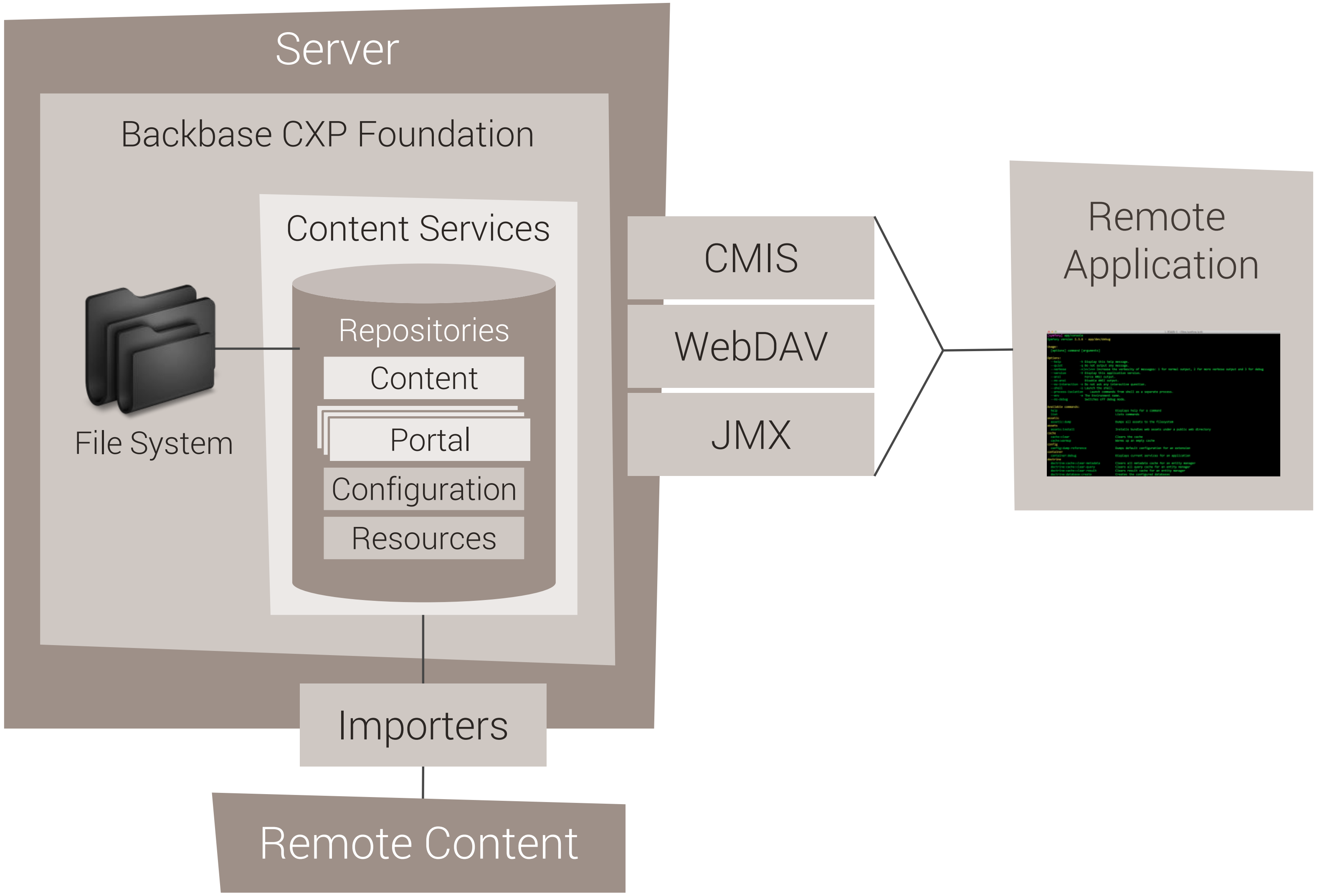
- Regarding scope, reusable content can be:
  - Shared (or global)
  - Portal-specific



Content Harvest



Content Services - Architecture



```
[symfony] app/console
Symfony version 2.3.6 - app/dev/debug

Usage:
  [options] command [arguments]

Options:
  --help           -h Display this help message.
  --quiet          -q Do not output any message.
  --verbose        -v|vv|vvv Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
  --version        -V Display this application version.
  --ansi              Force ANSI output.
  --no-ansi           Disable ANSI output.
  --no-interaction -n Do not ask any interactive question.
  --shell          -s Launch the shell.
  --process-isolation    Launch commands from shell as a separate process.
  --env            -e The Environment name.
  --no-debug          Switches off debug mode.

Available commands:
  help                             Displays help for a command
  list                             Lists commands
assetic
  assetic:dump                     Dumps all assets to the filesystem
assets
  assets:install                   Installs bundles web assets under a public web directory
cache
  cache:clear                      Clears the cache
  cache:warmup                     Warms up an empty cache
config
  config:dump-reference            Dumps default configuration for an extension
container
  container:debug                  Displays current services for an application
doctrine
  doctrine:cache:clear-metadata    Clears all metadata cache for an entity manager
  doctrine:cache:clear-query       Clears all query cache for an entity manager
  doctrine:cache:clear-result      Clears result cache for an entity manager
  doctrine:database:create         Creates the configured databases
```

Content Services - Repositories

- Content Services consists of the following repositories:
  - **Content** - a repository for storing reusable (web) content shared between all portal instances
  - Each portal instance has its own **Portal** repository that stores ICE and reusable content specific to that portal
  - **Configuration** - a repository for storing importer configurations
  - **Resources** - a repository for storing Backbase CXP resources

Content and Portal Repositories

- Store reusable web content
  - Can be imported from remote sources (3th party CMIS, Feeds)
- Can store user generated content (ICE)

Configuration Repository

- Stores the relevant Content Services configuration files
  - RSS and Atom Feed import configuration
  - CMIS import configuration
- Create a properties file and import the configuration
- You can add, edit or remove configurations at runtime

Resources Repository

- Stores resources needed by Backbase CXP:
  - Javascript
  - CSS
  - Images
  - XSLT files
  - Version history
  - …

Exercise: Start Content Services

- Start portal server (do an import if necessary)
- Start content services
  - Change into the content services directory
  - Use jetty:run-war
- Try to add content via portal manager's content widgets

Interfaces

Content Services

Content Services - Interfaces

- Interfaces to interact with the Content Services repositories:
  - CMIS - enables interaction with any of the three repositories (Content, Configuration and Resources) at a very low level, down to metadata.
  - WebDAV - batch processing capabilities make the WebDAV interface suited to run batch jobs on the Content repository.
  - JMX - manages and monitors the delivery service.

CMIS

Content Services / Interfaces

CMIS

- CMIS stands for Content Management Interoperability Services
- CMIS is an OASIS standard designed for the ECM (Enterprise Content Management) industry.
- Examples of CMSs that implement the CMIS standard include: Backbase CMS, Microsoft SharePoint, IBM FileNet, EMC Documentum, and Alfresco.
- CMIS specifies a query language based on SQL-92.
  - It is read only and has limited support for joins.
  - Depending on the repository, it can do both full-text and metadata based queries.
- Backbase Content Services Repository implements a subset of the CMIS specification.

CMIS – Key Concepts

- Web Services and ReSTful AtomPub
- Repository
- Basic Objects Types
  - Document (cmis:document)
  - Folder (cmis:folder)
  - Relationship (cmis:relationship)
  - Policy (cmis:policy)

CMIS - Document

- cmis:document
- Represents pieces of content in the repository, like images, text or CSS
- Lookup using ID or path
- Document object may have content streams.

CMIS - Folder

- cmis:folder
- Represents folders in the repository, capable of containing documents or other folders
- Lookup using ID or path
- Can only have one parent
- One-to-many relationship with cms:document

CMIS – Custom Object Types

- bb:link - extends cmis:document, used to store hyper-links
  - adds the `bb:url` and `bb:title` properties
- bb:richtext - extends cmis:document,
  meant to store content for the Rich Text Editor (from ICE)
  - can store any textually encoded content
- bb:image - extends `cmis:document`, used to store images
  - adds `bb:altText, bb:height, bb:lastPublishedDate, bb:tag, bb:title and bb:width` properties

CMIS Services

- Common Service Elements
- Repository Services
- Navigation Services
- Object Services
- Multi-filing Services
- Discovery Services
- Versioning Services
- Relationship Services
- Policy Services
- ACL Services

BACKBASE

CMIS Query

- Content Services implements a subset of the query language specified in the CMIS Standard.

```
SELECT * FROM cmis:document
```

* mean all columns
<select list> is coming in next release.

CMIS Predictates

SELECT * **FROM** cmis:document **WHERE** …

- Comparison (>, <, <>, >=, <=, =)

```
SELECT * FROM cmis:document
     WHERE cmis:contentStreamLength >= 1024
```

- IN (list of values)

```
SELECT * FROM cmis:document
    WHERE cmis:objectTypeId IN ('bb:image','bb:richtext','bb:link')
```

- LIKE (% and _ are special characters)

```
SELECT * FROM cmis:document WHERE cmis:name LIKE 'j%’

  SELECT * FROM cmis:document WHERE cmis:name LIKE 'j___.xml'
```

Order by (ASC, DESC)

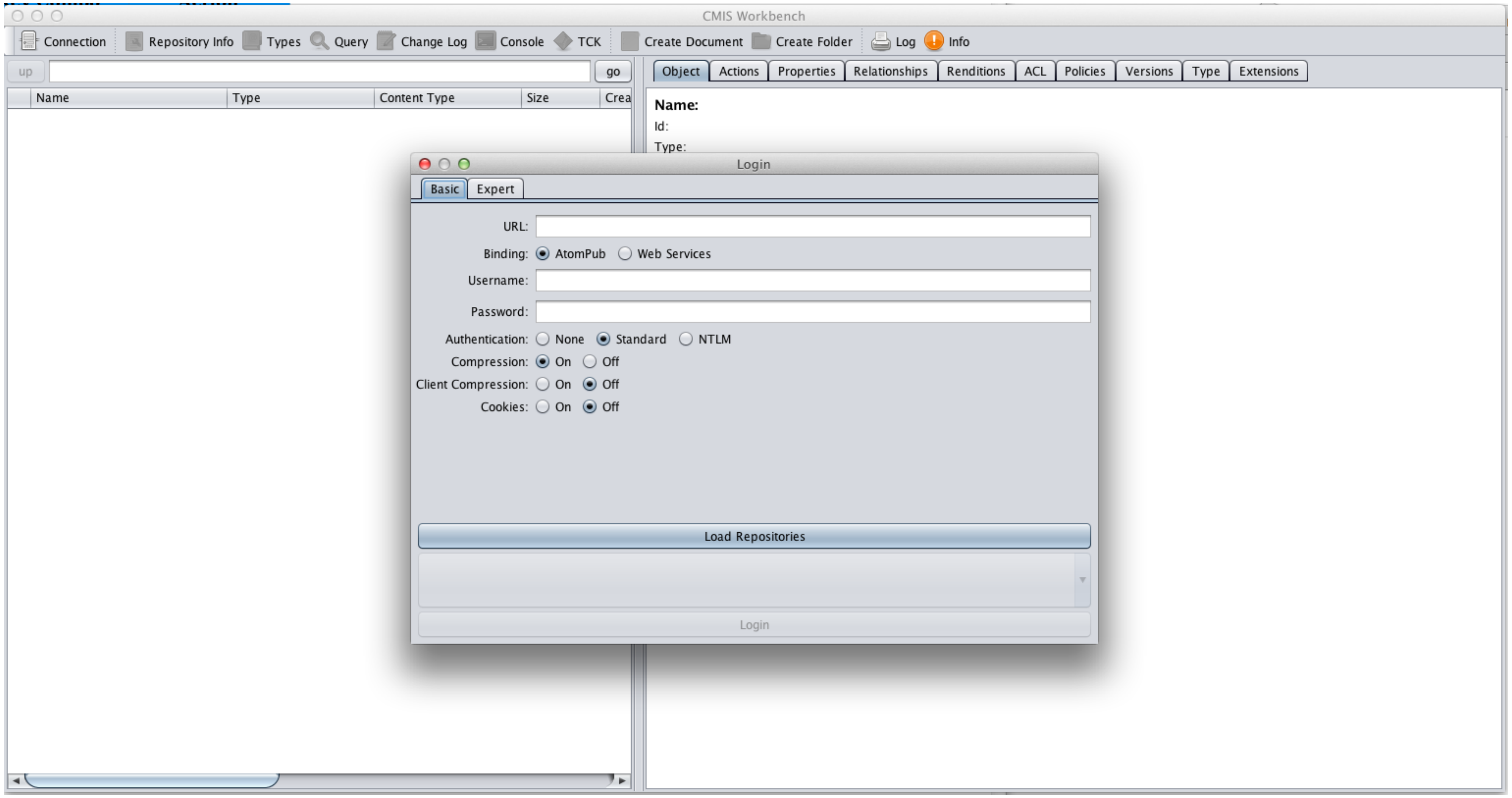
```
    SELECT * FROM cmis:document
        WHERE cmis:contentStreamMimeType = 'text/xml'
        ORDER BY cmis:objectId

  SELECT * FROM cmis:document
        WHERE cmis:contentStreamMimeType = 'text/xml'
        ORDER BY cmis:objectId DESC
```

CMIS Clients

- Apache Chemistry Workbench
  - http://chemistry.apache.org/java/download.html
- Apache Chemistry Java API
- Backbase JavaScript API

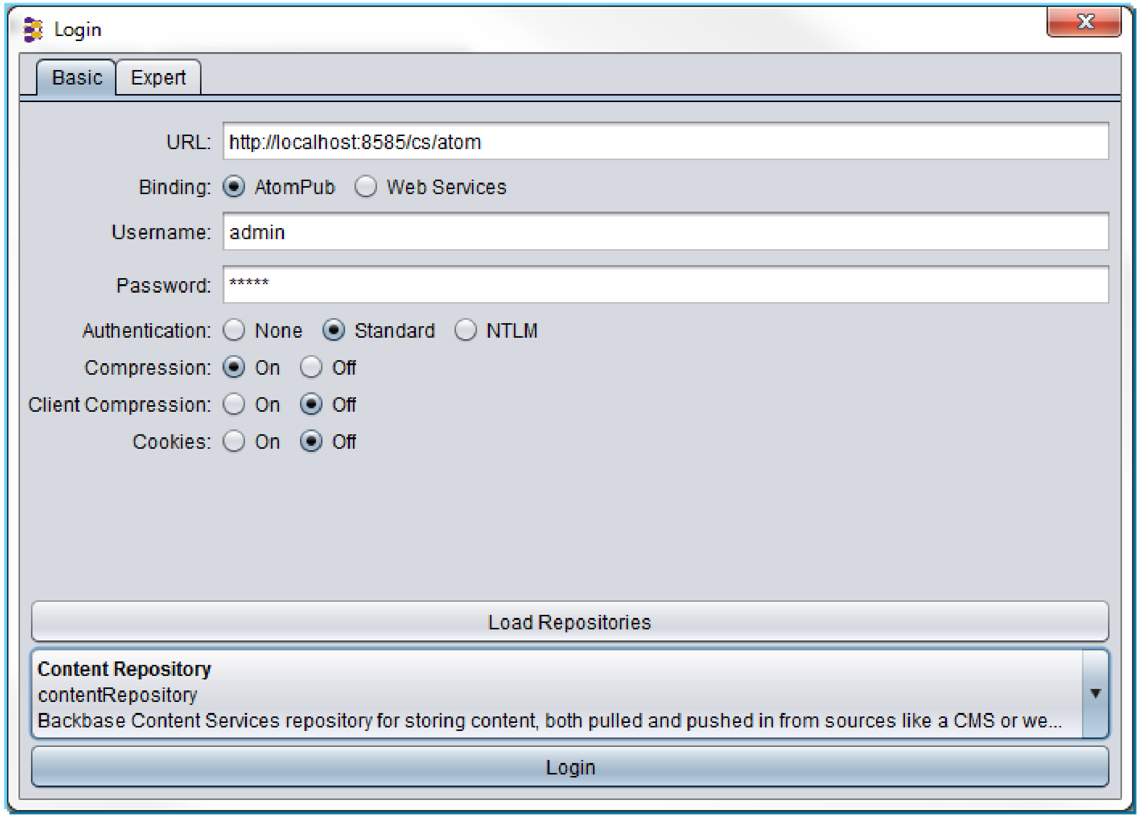
Chemistry Workbench



URL:
Binding: [✓ AtomPub] Web Services
Username:
Password:
Authentication: None [✓ Standard] NTLM
Compression: [✓ On] Off
Client Compression: On [✓ Off]
Cookies: On [✓ Off]

Exercise: CMIS

- Starting the CMIS Workbench
- Login
- Add folders and documents
- Fire some queries


Login

URL: http://localhost:8585/cs/atom
Binding: [✓ AtomPub] Web Services
Username: admin
Password: *****
Authentication: None [✓ Standard] NTLM
Compression: [✓ On] Off
Client Compression: On [✓ Off]
Cookies: On [✓ Off]

Content Repository
contentRepository
Backbase Content Services repository for storing content, both pulled and pushed in from sources like a CMS or we...

CMIS Java API

- Possible to connect to a repository programmatically.
- Using Apache Chemistry CMIS Java API.
- The same functionality that exist in CMIS clients is possible using the Java API.
- Comprehensive example available as a How To guide on my.backbase.com under the heading *“Working with Content Services from Java Using the CMIS API”*

CMIS Java API

- CMIS session – A connection to a CMIS repository with a specific user

```
Map<String, String> connProps = new HashMap<>();
connProps.put(SessionParameter.ATOMPUB_URL,
“http://localhost:7777/portalserver/content/atom”);
connProps.put(SessionParameter.USER, “admin”);
...
SessionFactory factory = SessionFactoryImpl.newInstance();
Session session = factory.createSession(cmisConnectionProperties);
...
Document doc = (Document) session.getObjectByPath(“/img/pic.jpg”);
...
QueryStatement qs = session.createQueryStatement("SELECT * FROM
cmis:document”);
```

- Sessions are purely a client side concept: thread-safe, closing not required. Results cached by default.

Exercise: CMIS Java API

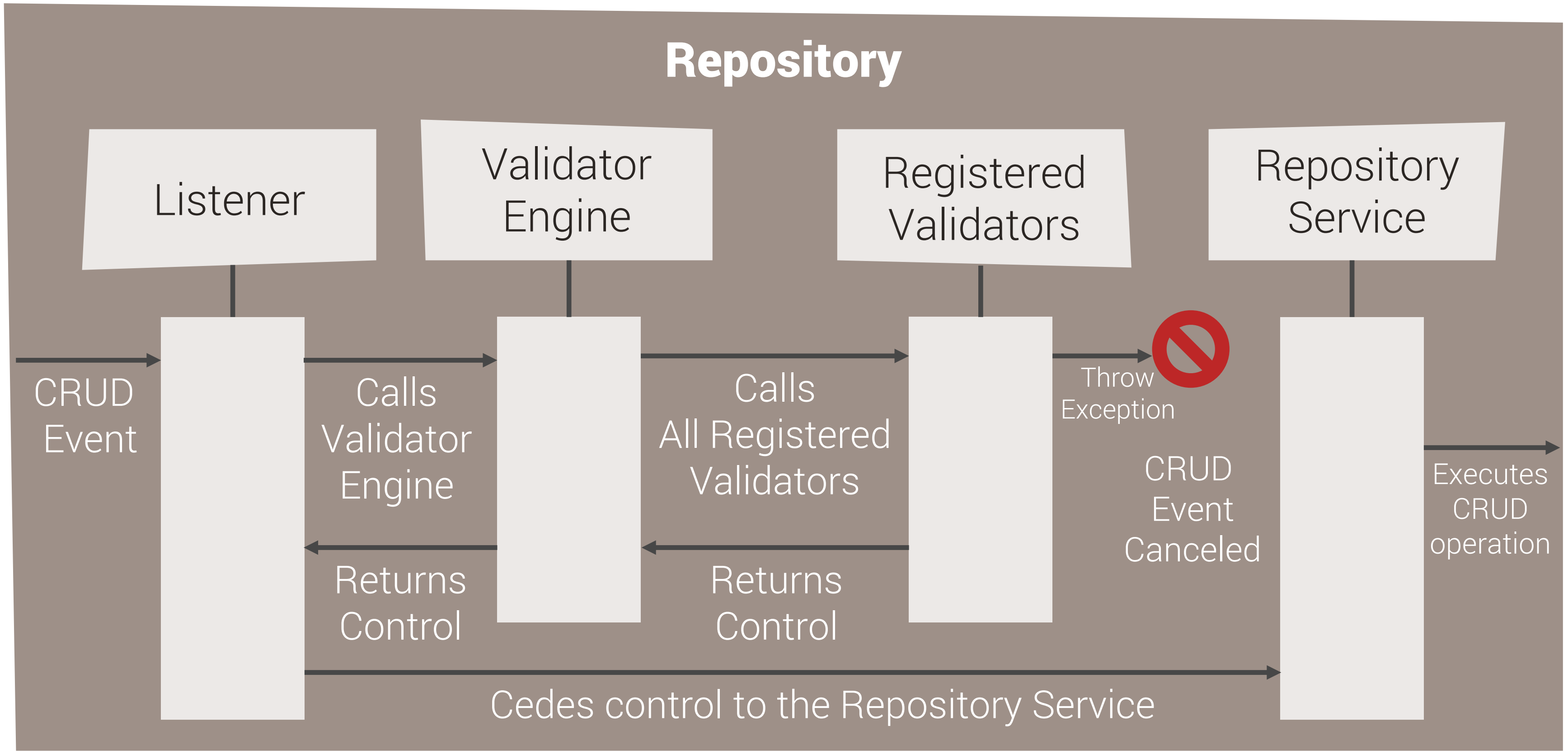
- Create a Camel route listening for file changes in a folder
  - Path configurable in **backbase.properties**
  - Assume flat file structure
  - Handle updates to existing files
- Implement the CMIS endpoint as a Spring bean and invoke it using Camel's bean binding
- Create an endpoint (Spring bean) that returns the MIME type of a file
- Use this route to enrich the content of each message (Content Enricher EIP)
- (OPTIONAL) Handle full folder hierarchy

Validators

Content Services

Validators

- Validate CRUD operations before they execute
- A canceled event will cause a HTTP 500 error.



Validators

- RepositorySchemaValidator:
  - Checks that object complies to its type schema before a create/update operation.
  - Ensures that all required properties are set.
- Custom validators

Custom Validators

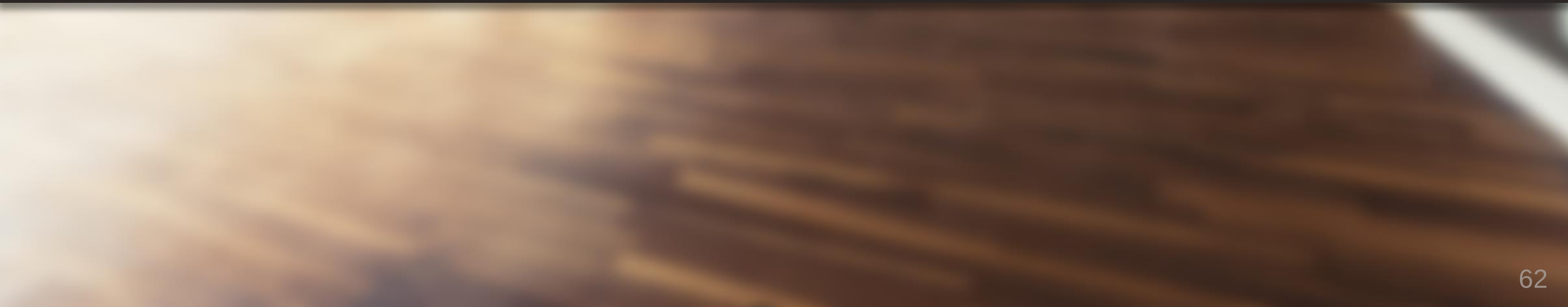
- Implement the `RepositoryValidator` interface
  - `validateCreate()`
  - `validateDelete()`
  - `validateMove()`
  - `validateUpdateContentStream()`
  - `validateUpdateProperties()`
- See also the online javadoc

Exercise: Validators

- Create a validator that will allow creating or updating images (documents of the bb:image type) only if they have one of the following extensions ".jpg", ".jpeg", ".png"

Deployment

Content Services



Deployment

- Test environment jetty:run
- Database setup
- Application Containers

Caching

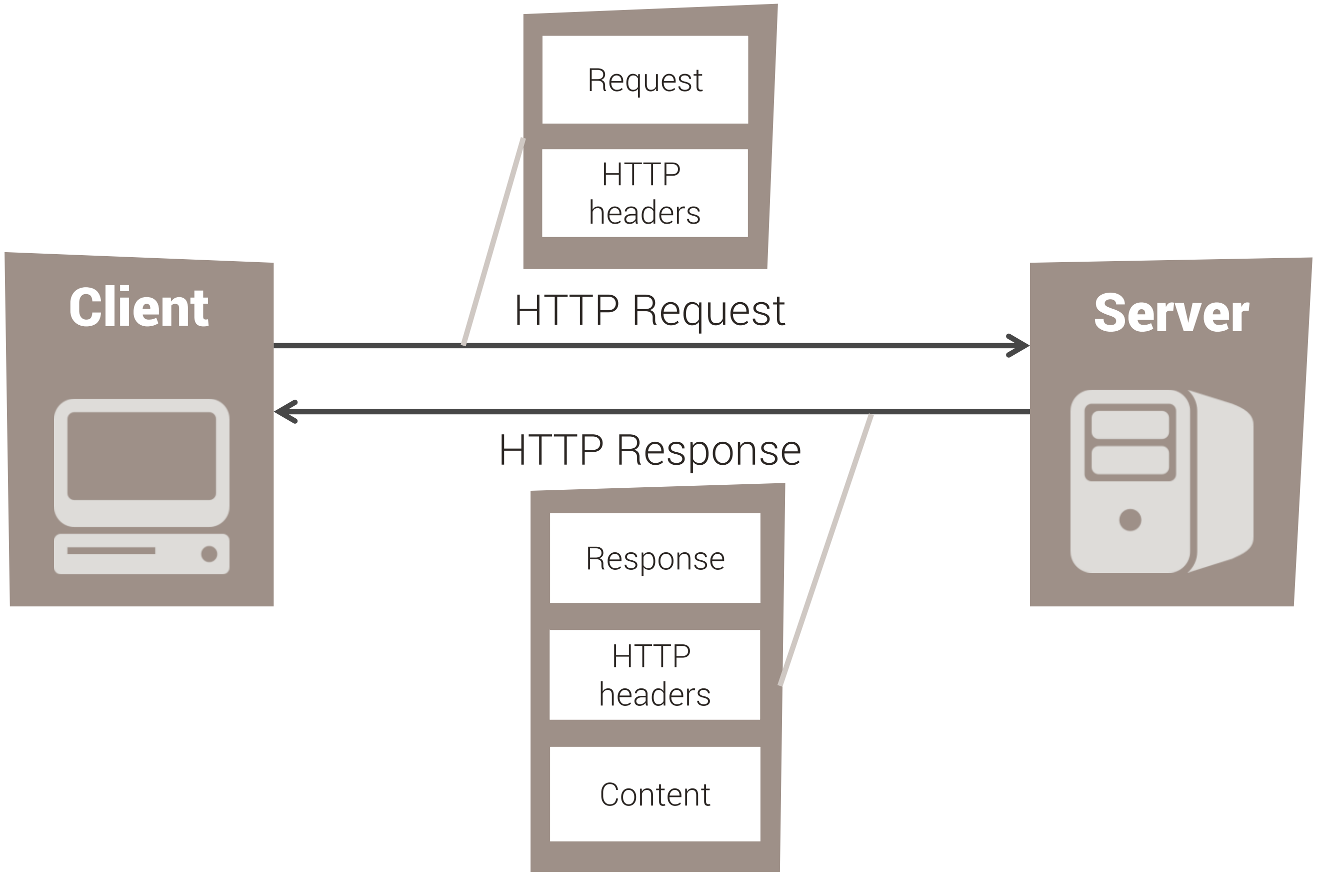
Content Services

Caching

- Browser Cache
  - Reducing latency
  - Reducing network traffic
  - Cons: cache content can be not up to date (stale)
- HTTP `cache-control` headers
- public, private, and max-age

Caching

- backbase.properties
- contentservices.cache-control.default=public; max-age=600



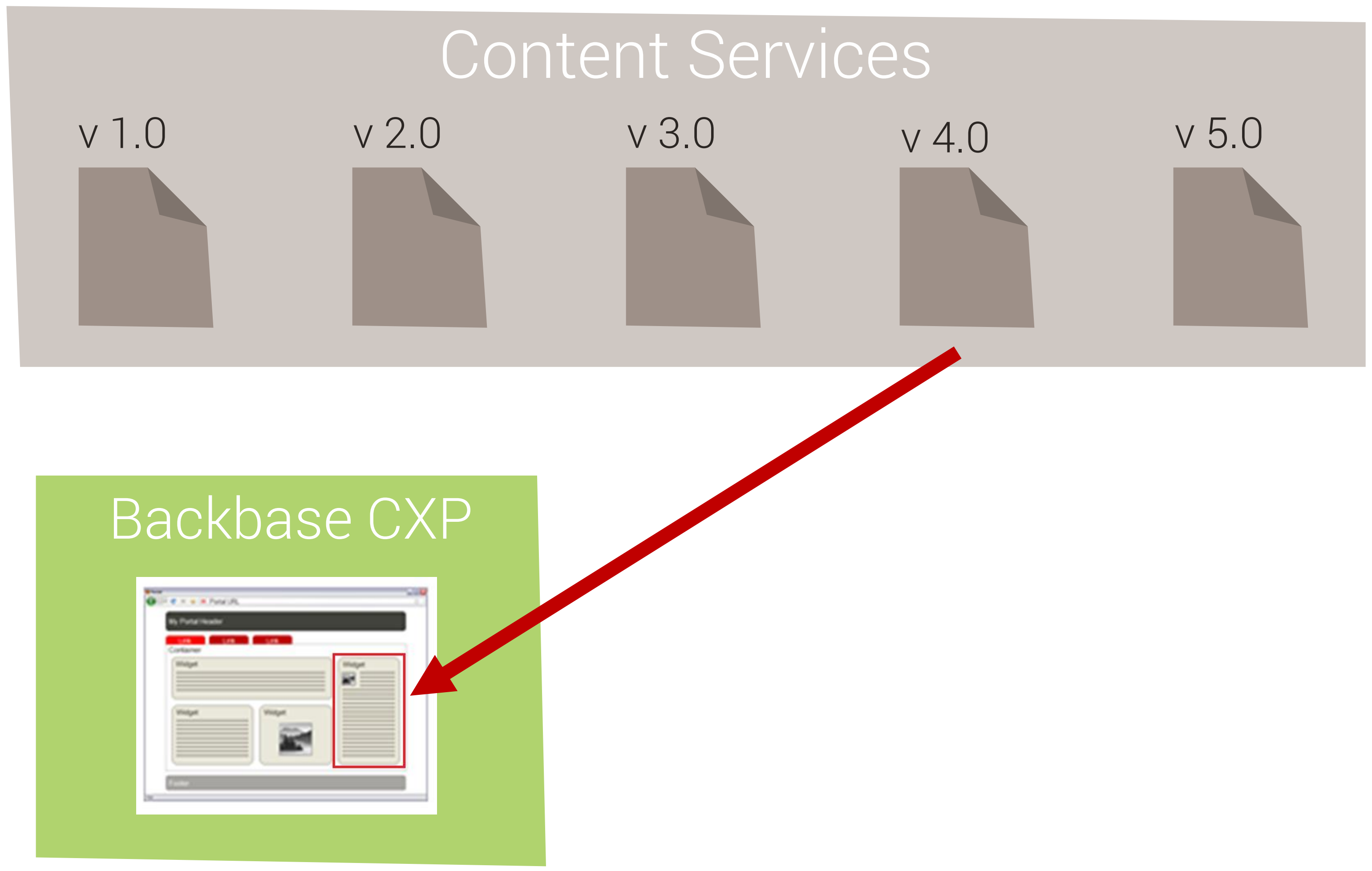
Versioning

Content Services

Versioning Services

- Content Services implements part of the versioning services defined in the CMIS Standard.
- The versioning services
  - store different versions of the same document,
  - reference each individual version according to the value of its properties,
  - make possible to track which user created a particular version.
- The locking mechanism prevents concurrent changes by different users.
- Only Backbase object types support versioning, while pre-defined types (like cmis:document) do not.
- The Versioning feature in Backbase CXP Foundation uses CS to version pages, content items, and links.
- Page versions are stored in the resources repository of CS.

Versioning



Version Series

- A version series is a collection of objects created from an original document.
- Each version series has a unique, system-assigned, and immutable ID, and its members are assigned version numbers.
- Version numbers follow the syntax **major.minor**, and users can decide whether the version of a document is major or minor:
  - **major version** numbering starts with 1.0 and increases by 1 with each subsequent major version
  - **minor version** numbering starts with 0.1 and increases by 0.1 with each subsequent minor version
- Backbase CXP uses only major version numbers.

BACKBASE

Thank you!

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

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