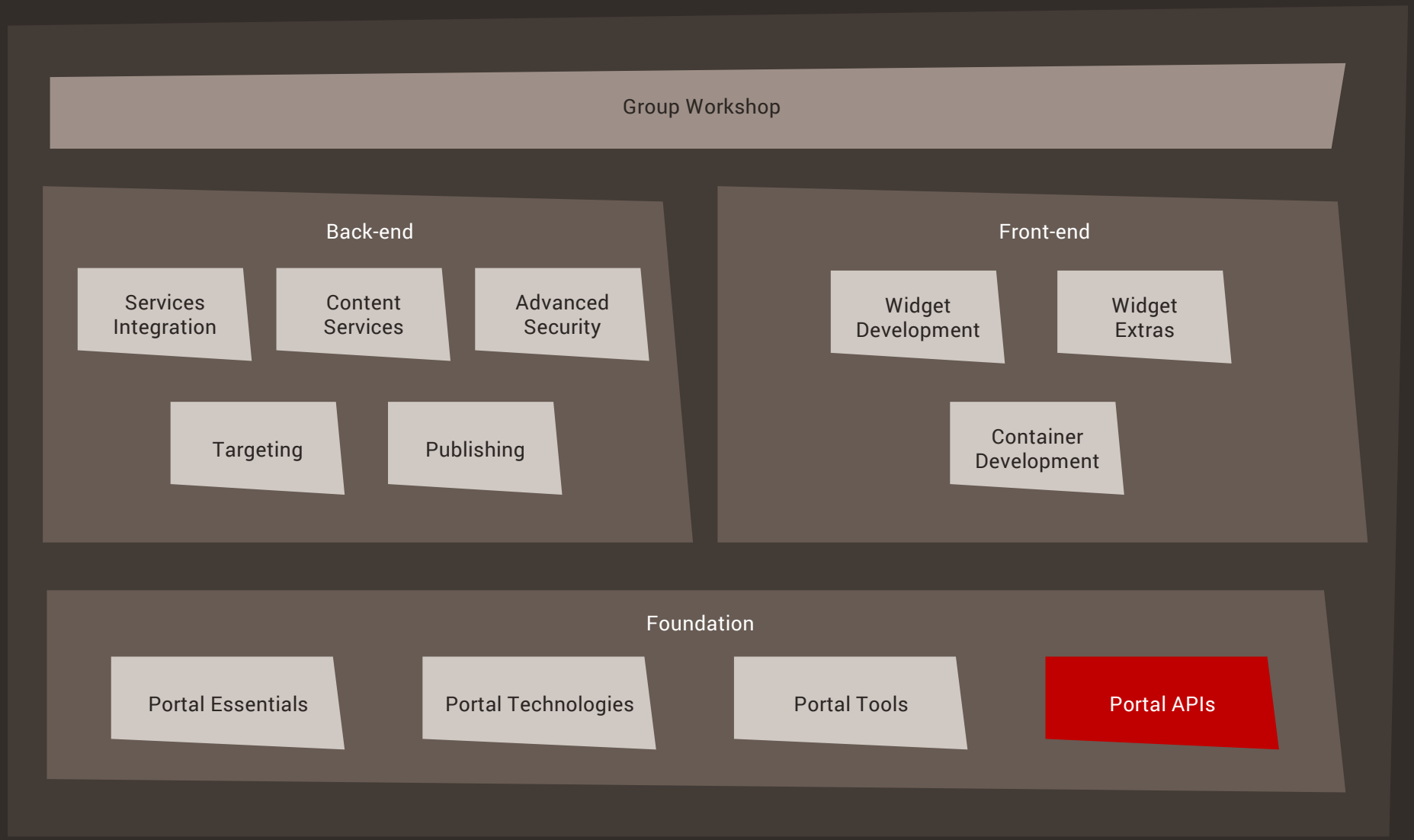
BACKBASE

Portal APIs

Focus Area: Foundation

BACKBASE

Training Overview

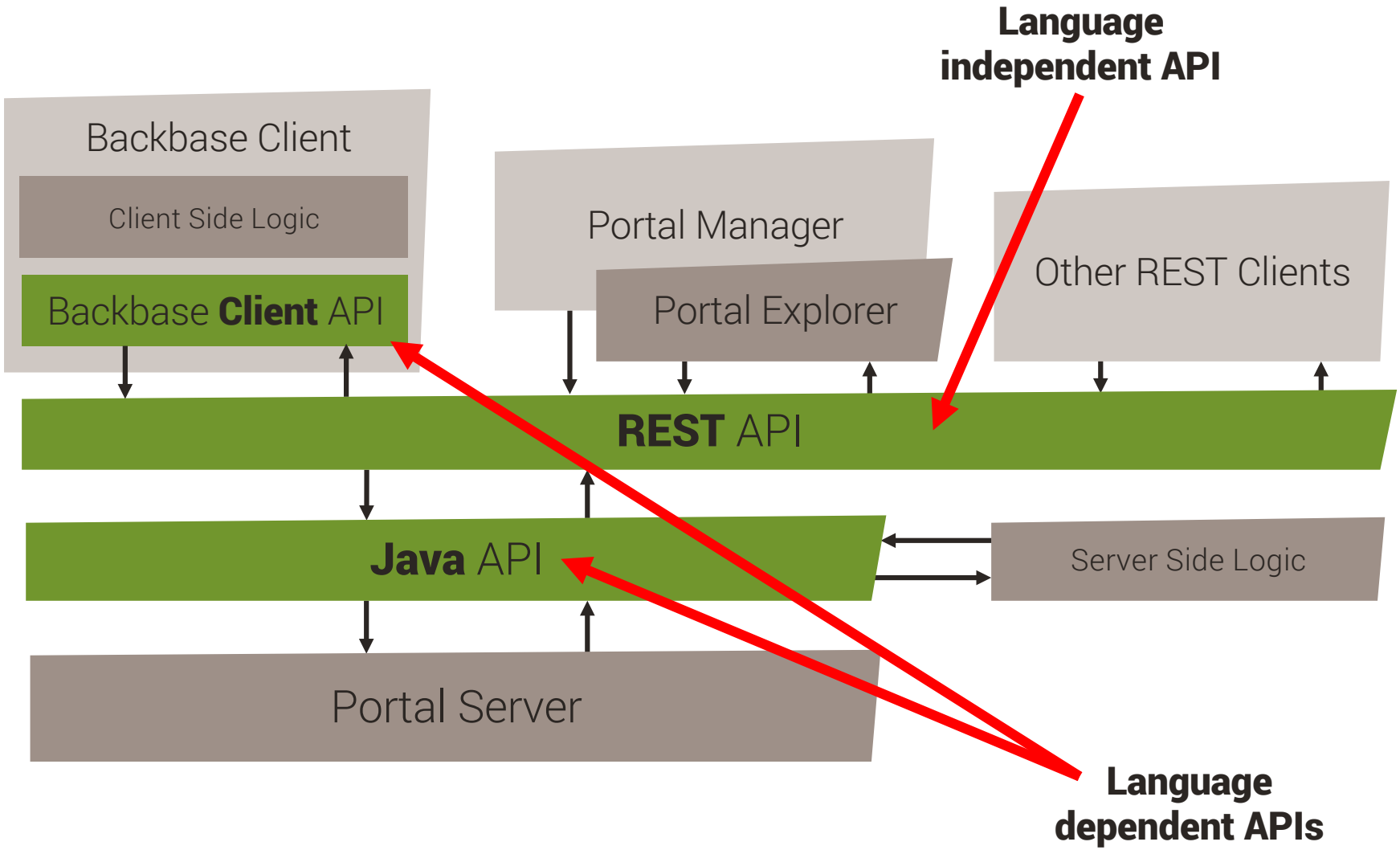


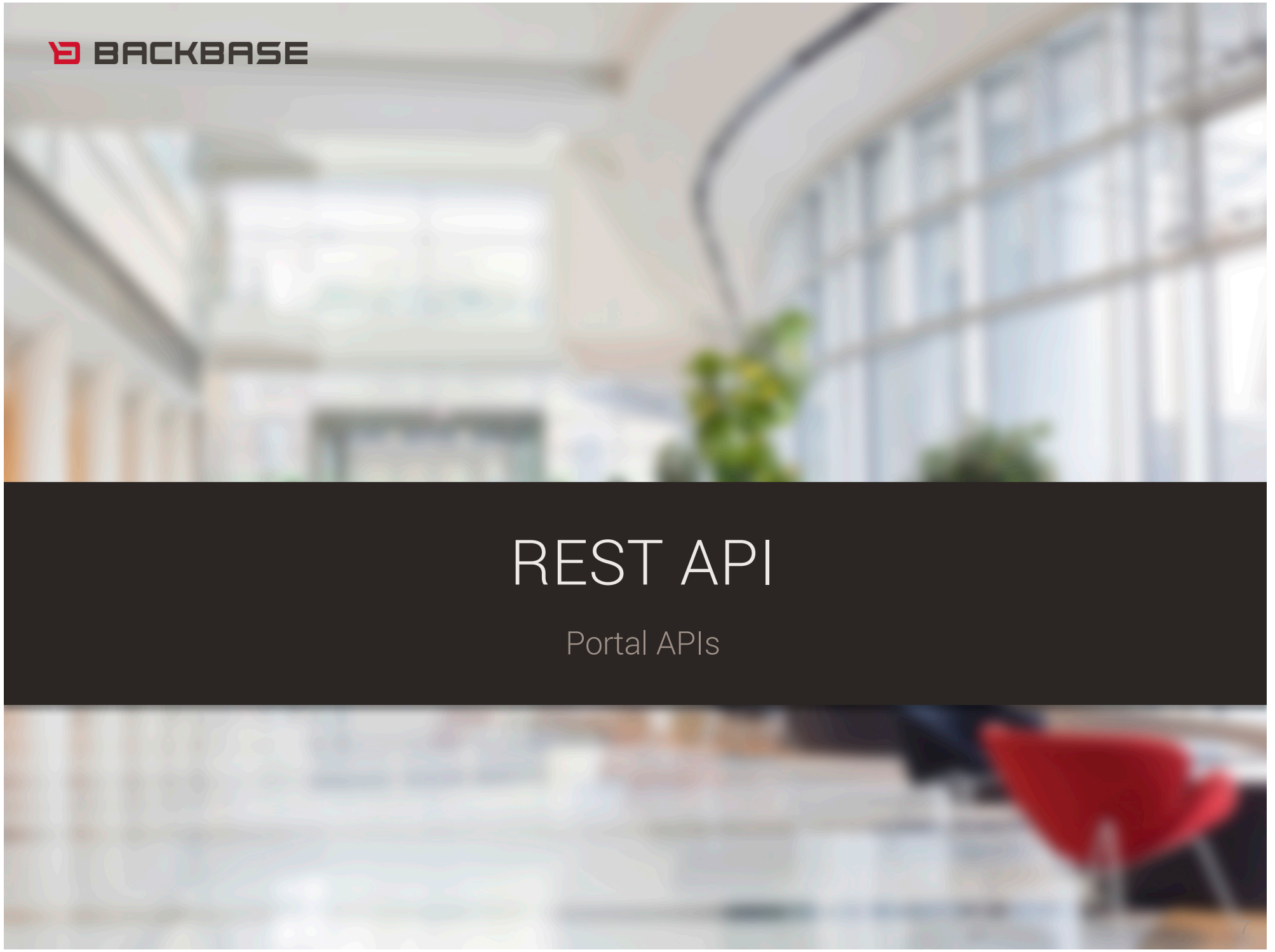
1. Foundational knowledge of APIs
2. Understanding the REST API
3. Awareness of the Java API
4. Awareness of the client API

Introduction

Portal APIs

- API = Application Programming Interface
  - A specification intended to be used as an interface by software components to communicate with each other
  - An API may include specifications for routines, data structures, object classes, and variables
- An API can be:
  - language dependent (e.g. Java, Javascript)
  - language independent
    - Can be called from several programming languages
    - Service-oriented APIs (REST, SOAP)

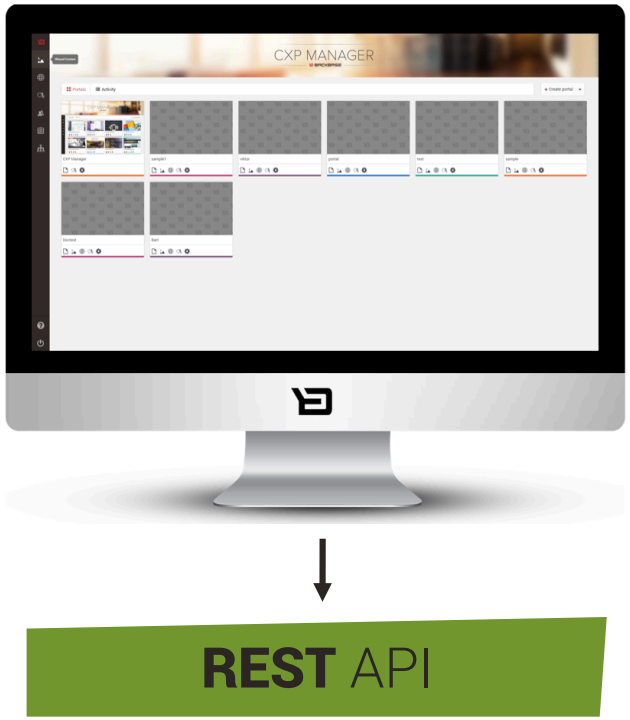

Language independent API
Backbase Client
Client Side Logic
Backbase Client API
Portal Manager
Portal Explorer
Other REST Clients
REST API
Java API
Server Side Logic
Portal Server
Language dependent APIs


BACKBASE
REST API
Portal APIs

- Representational
  - XML, JSON, HTML…
- State
  - More concerned with state of resource than with actions
- Transfer
  - Transferring resource data in some representational form

- Architectural style, resource-centered
  - Compare to service-centered style
- Key principles:
  - Give every item an ID (URL)
  - Link items together
  - Use standard methods
    - HTTP: POST, GET, PUT, DELETE
  - Resources can have multiple representations
  - Stateless communication
    - Server should not have to retain communication state for any of its clients beyond a single request

- CXP Manager uses the REST API
  - Manipulate the portal model
    - portals, pages, containers, widgets
  - Manage users and groups
- Common XSD schema definition
- CRUD operations (create, read, update, delete), sorting, filtering, paging



URI	Method	URL modifiers
`/portals`	`GET`	`f`, `of`, `ps`, `s`
`/portals/{portalName}`	`GET`	`pc`
`/portals/{portalName}/pages`	`GET`	`f`, `of`, `ps`, `s`, `depth`
`/portals/{portalName}/pages/{pageName}`	`GET`	`pc`
`/portals/{portalName}/links`[a]	`GET`	`f`, `of`, `ps`, `s`, `depth`
`/portals/{portalName}/links/{linkName}`	`GET`	`pc`, `depth`
`/portals/{portalName}/containers`	`GET`	`f`, `of`, `ps`, `s`, `depth`
`/portals/{portalName}/containers/{containerName}`	`GET`	`pc`
`/portals/{portalName}/catalogTags`	`GET`	`f`, `of`, `ps`
`/portals/{portalName}/widgets`	`GET`	`f`, `of`, `ps`, `s`, `depth`
`/portals/{portalName}/tags`	`GET`	`f`, `of`, `ps`, `s`
`/portals/{portalName}/tags/{tagName}`	`DELETE`	`type`
`/catalog`	`GET`	`f`, `of`, `ps`, `s`
`/catalogTags`	`GET`	`f`, `of`, `ps`
`/templates`	`GET`	`f`, `of`, `ps`, `s`, `depth`
`/features`	`GET`	`f`, `of`, `ps`, `s`, `depth`
`/groups`	`GET`	`f`, `of`, `ps`, `s`
`/groups/{groupName}/users`	`GET`	`f`, `of`, `ps`, `s`
`/auditEvents`	`GET`	`of`, `ps`, `context`, `user`, `item`, `itemTitle`, `from`, `to`, `itemtype`, `action`, `dates`
`/export/portal`	`GET`	`portalName`

[a] GET calls on the `/links` require the `depth` modifier. Requests with no modifier return a `400 Bad request` code.

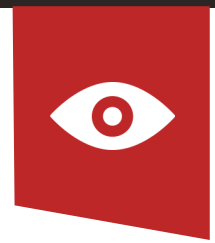
- It is possible to run REST API queries about items specifying modifiers and parameters
- Types of modifiers:
  - Filter queries (f)
  - Sorting (s)
  - Paging (ps, of)
  - Processing children (pc, depth)

- Filter queries (f)
  - The following syntax is supported:
    - ?f=element_name(operator)value
      Filters elements by <name> elements.
    - ?f=property.property_name(operator)value
      Filters elements by the value of a property.
  - Available operators: lt, gt, eq, not, like
  - Example:
    /portals?f=property.DefaultLandingPage(eq)index

- Sorting (s)
  - Sorts results of queries in either ascending or descending order
  - The syntax: s=element_name(asc|dsc)
    - asc – (ascending)
    - dsc – (descending)
  - Example: /portals/[portal_name]/pages?s=name(asc)

- Paging results (ps, of)
  - Paging modifiers allow us to page responses:
    - ps – the number of results in the response. The default page size varies per object.
    - of – the number of results to be skipped in the response. Defaults to 0.
  - Example: /portals/[portal_name]/widgets?of=8&ps=4

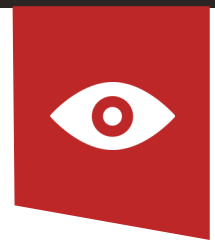
- Processing children (pc, depth)
  - Specify whether the children of an item will be processed or not:
    - pc – applies to GET requests on the URLs that process items capable of containing child items. If set to false, the response does not include the object children. Defaults to true.
    - depth – specifies the number of children levels to be displayed.



Returns the list of portals
/portals

Returns the portal model
/portals/{portal_name}.xml

Returns a list of pages from 5 to 10, sorted ascending by name
/portals/{portal_name}/pages.xml?of=5&ps=5&s=name(asc)



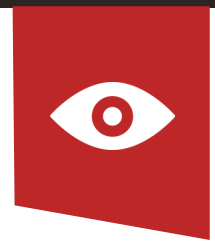
An item can be requested in either HTML or XML format, depending on the extension of the request

REST API response formats

- .html
- .xml

1. Get all the widgets in a portal sorted by name

2. Create a new user



Portal Manager uses REST API to communicate with Portal Server

Java API

Portal APIs

- https://docs.backbase.com/portal/5.6.2/javadocapi/index.html
- Important business services defined in
  - `com.backbase.portal.foundation.business.service`
- Used in custom server logic instead of the REST API
  - Internally used by the REST API

Java API – Important Interface/Classes

- Package com.backbase.portal.foundation.business.service
  - PortalBusinessService
  - *GroupBusinessService*
  - *ItemBusinessService*
- Instances created by portal server as beans:

```
@Autowired private ItemBusinessService<Item> itemBusinessService;
```

or

```
HttpServletRequest req = (HttpServletRequest) pageContext.getRequest();
ApplicationContext cxt = RequestContextUtils.getWebApplicationContext(req);
ItemBusinessService<Item> itemService = cxt.getBean("itemBusinessService",ItemBusinessService.class);
```

- com.backbase.portal.foundation.business.service

Method Summary	
Portal	**createPortal**(Portal pPortal) Create a new portal in the system.
void	**deletePortal**(String pPortalName, boolean softDelete) (soft) delete a portal.
Portal	**getPortal**(String pPortalName, boolean processChildren) Get a portal entity for a specific name.
List<Portal>	**getPortals**(Sorter sorter, Filter filter) Returns a list of all portal structures with properties and without children.
List<Right<? extends org.springframework.security.acls.model.Sid>>	**getRightsForItem**(String portalName) Retrieves a list of Rights for a Portal
Portal	**updatePortal**(String pOriginalPortalName, Portal pUpdatesPortal) update an existing portal
void	**updatePortals**(List<Portal> pPortals) Update a set of portals.
void	**updateRightsForPortal**(String portalName, List<Right<? extends org.springframework.security.acls.model.Sid>> rights) Updates the rights of a Portal

- com.backbase.portal.foundation.business.service

Method Summary	
void	assignUsersToGroup(String groupname, Users users) Assigns a list of users to the group
Group	createGroup(Group pGroup) Create a new group.
void	deleteGroup(String groupname, boolean softDelete) delete a group for the specified groupId
Group	getGroup(String groupname) Get a user for a specific userId
Groups	getGroups(Sorter sorter, Filter filter) Get a list of groups for the specific sorter and filter
Users	getUsersForGroup(String groupname, Sorter sorter, Filter filter) get a list of groups for a single user
void	updateGroup(String groupname, Group pGroup) Update the group defined by groupname.

- com.backbase.portal.foundation.business.service

Method Summary	
void	**assertContextExists**(String pContextItemName) Check if a context (portal) exists
void	**cascadingDeleteItem**(String pContextItemName, String pItemName) Hard delete an item to implicitly include extensions and children
T	**createItem**(String pPortalName, T pItem) Create an item.
void	**deleteItem**(String pPortalName, String pItemName, boolean softDelete) Delete an item.
List<T>	**getCatalogItems**(String pPortalName, Sorter pSorter, Filter pFilter) Get a list of items from the given Server or Portal catalog.
T	**getItem**(String pPortalName, String pItemName, boolean processChildren) Get an item.
List<T>	**getItems**(String pPortalName, Sorter pSorter, Filter pFilter)

- com.backbase.portal.foundation.business.service

	Get a list of items
List<T>	getItemsWithSpecificState(String pPortalName, State state) Gets all the items with the specified state
T	getItemWithSpecificState(String pPortalName, String pItemName, boolean processChildren, State state) same as get item but to select the item with specific state
List<Right<? extends org.springframework.security.acls.model.Sid>>	getRightsForItem(String pPortalName, String pItemName) Get the current rights for the item
T	updateItem(String pPortalName, String pItemName, T pItem) Update an item (in this case the given item name and the name field of the item object, are not the same...
void	updateItems(String pPortalName, List<T> items) Update a list of items of 1 type.
void	updateRightsForItem(String pPortalName, String itemName, List<Right<? extends org.springframework.security.acls.model.Sid>> rights) Update the rights for the item

- CXP will notify listeners of many important events:
  - AddGroupsToUserEvent
  - AssignUsersToGroupEvent
  - BeforeServerSideRenderingEvent
  - BeforeURLCacheResponseEvent
  - ContentEvent
  - CreateContentEvent
  - CreateGroupEvent
  - CreateItemEvent
  - CreateUserEvent
  - DeleteContentEvent
  - DeleteGroupEvent
  - DeleteItemEvent
  - DeleteUserEvent
  - GroupEvent
  - GroupUserEvent
  - ItemCollectionTransformedEvent
  - ItemEvent
  - NonAuditableItemUpdateEvent
  - PublishingApproveEvent
  - PublishingCreateSetEvent
  - PublishingEvent
  - PublishingFailEvent
  - PublishingRejectEvent
  - PublishingUnlockEvent
  - RemoveAllUsersFromGroupEvent
  - RemoveUserFromGroupEvent
  - UpdateContentEvent
  - UpdateGroupEvent
  - UpdateItemEvent
  - UpdateItemListEvent
  - UpdateItemRightsEvent
  - UpdateUserEvent
  - UrlLevelCacheRemovalEvent
  - UserEvent

JavaScript API

Portal APIs

- Javascript API is a Javascript wrapper around the REST API

- b$.portal.loggedInUserId =>"admin”
- b$.portal.portalName => “training-portal”
- b$.portal.config.serverRoot => "/portalserver"

1. Language dependent and language independent APIs
2. REST API
3. Java API
4. Client API

BACKBASE

Thank you!

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

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