

**adaptTo()**

APACHE SLING & FRIENDS TECH MEETUP  
2 - 4 SEPTEMBER 2019

Everything is a Component - Site Development with Composium Pages  
Ralf Wunsch, IST GmbH Dresden

# who am i



- Developer, Founder of the IST GmbH
- Sling enthusiast - Sling / AEM since 2009
- 2015 Comosum Console - ‘Comosum’
  - ‘warm up’ project, part of the Sling starter
- Comosum ‘Pages’ / ‘Platform’ / ...
  - first ideas 2016 - Release 1.0 2019 just now

Ralf Wunsch



# Composium

- „Nodes“ - Browser, Package / User Manager - Core API
- „Platform“ - Release / Access Management
- „Pages“ - Content Management
- „Assets“ - Asset Management (deferred - work in progress)
- see: <https://www.composium.com/home.html>  
<https://www.composium.com/home/cloud.html>  
<https://cloud.composium.com/>

# Composium Modules

Pages

Components

Models

Widgets

Stage(UI)

Assets

Manager

Renderer

Components

Platform

Staging

Releases

Security

Models

Caching

Nodes

cpn:Taglib

Clientlibs

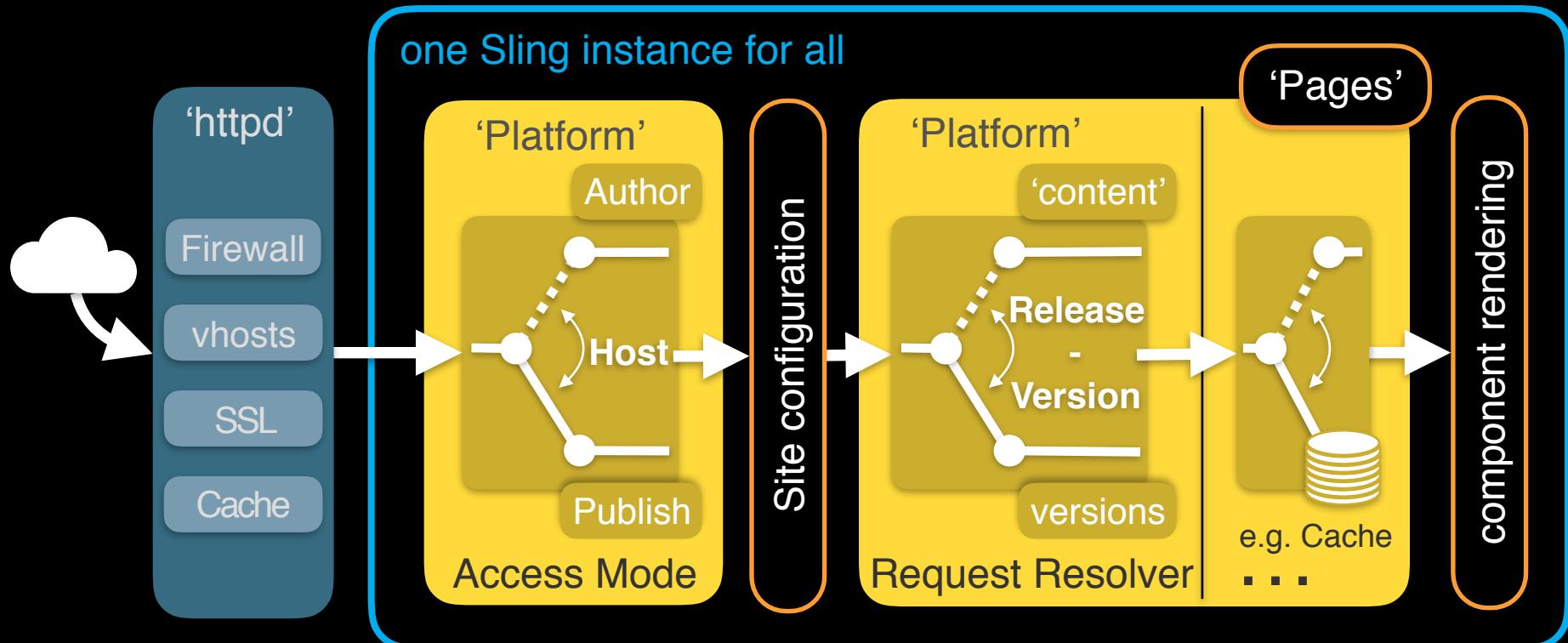
Console

Browser

Packages

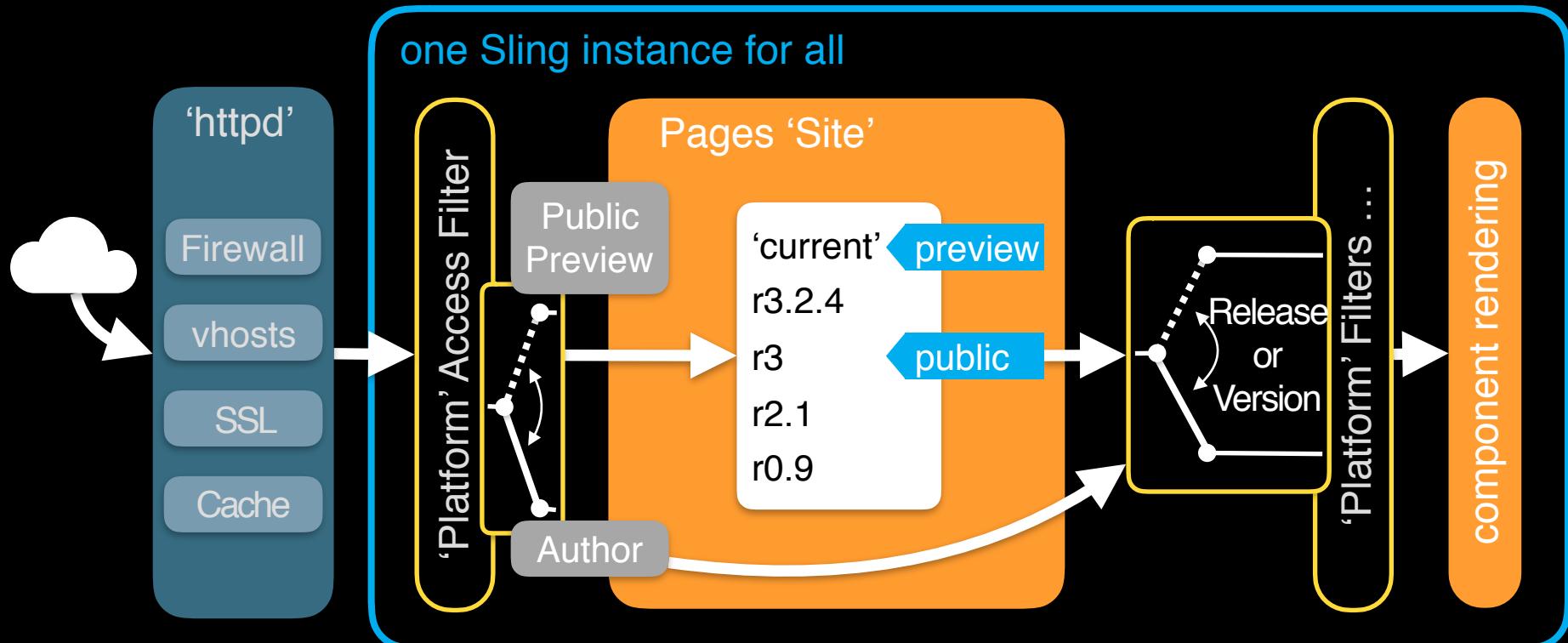
Users

# 'Platform'





# 'Pages' Releases





# Pages - Entities



- Site - the whole website - released
  - Page - a single page - various languages
    - Container - arranges a set of elements
      - Element - more or less complex piece
    - Asset - file based resource - part of the site
  - Templates - Site / Page - design policies



# Pages - Features



- a CMS on top of Composium Nodes / Platform
- content release management and delivery ('preview' / 'public')
- language support on the property and/or the page level
- a modularised client libraries (CSS, JS)
- content structure templates and page content design rules
- extendable CMS user interface powered by Apache Sling
- standard component and widget set (extendable)
- content components development framework



# today: ‘Pages’ components



# Pages UI Components

The screenshot shows the Composum CMS interface for managing page components.

**Left Sidebar:** Shows the file structure under `jcr:root`, with `teasers` selected. Other categories include `content`, `shared`, `ist`, `components`, `composium`, `no-template`, `restricted`, `simple`, `software`, `test-cs`, and `home`.

**Top Bar:** Includes icons for creating, duplicating, deleting, and saving components, along with language selection (en), revision controls, and a search bar.

**Central Content Area:** Displays a `Teaser1` component with the title "Video Teaser" and subtitle "Video Teaser Subtitle". Below it is a description: "a teaser with a video in the background". A horizontal dashed line separates this from the `main` section. The `main` section contains a row of circular icons and a link to `/main/row_326112113/column-0`. A note explains that teasers are combinations of assets or symbols and text, and can be used in various contexts like carousels.

**Right Sidebar:** Lists available teaser components: `Default Teaser`, `Image Teaser`, `Symbol Teaser`, `Video Teaser`, and `Video Teaser` (selected). The selected component is `Video Teaser` (teaser1) located at `/stage/teaser1`.

**Bottom Status Bar:** Shows the current page (`test-cs`), the number of items (0), and the number of revisions (6).

# Everything is a (Sling) component



- Pages, Containers, Elements
- Component Tiles / Toolbars
- Pages Tools
  - Navigation (left)
  - Context (right)
- Edit Dialogs / Widgets

Demo



adaptTo()

# an image component example



# the component declaration

```
<jcr:root xmlns:...  
  jcr:primaryType="cpp:Component"  
  jcr:title="Image (Simple)"  
  jcr:description="a single image element"  
  category="[Image, Prototype, Simple]"  
  sling:resourceSuperType="comosum/.../components/element"/>
```

```
<jcr:root xmlns:...  
  jcr:primaryType="cpp:Component" ...  
  componentType="cpp:Element" />
```



# the rendering template

```
<cpp:element var="model"
    type="com.comosum.pages.common.model.Image"
    test="@{model.valid || model.editMode}">
    <cpp:dropZone property="imageRef" filter="asset:image">
        <cpn:div test="${model.valid}" class="${modelCSS}_frame">
            <cpn:image class="${modelCSS}_picture"
                src="${model.src}" alt="${model.alt}"
                title="${model.title}"/>
        </cpn:div>
    </cpp:dropZone>
</cpp:element>
```



# Yes, a JSP taglib!

- keep component code as simple as possible
- avoid complex layering (edit hooks are embedded in the page)
- a ‘language’ on top of the scripting engine
- JSP as proven ‘old’ standard
- extendable - probably if HTML supports a ‘taglib’ feature
- Thymeleaf? - ideas are welcome
- Questions? - after the talk!

Demo



# the various component types



# content components / page components

- rendering template
- dialogs: edit [, create, delete]
- component tile and thumbnail
- component actions (toolbar)
- component help (content)
- use Pages Components as template

Demo



# dialog widgets

- Sling Component
  - primaryType: cpp:Widget
  - rendering template (for the POST servlet)
  - widget model class (tag attribute handling)
  - CSS / JS scripts if necessary
- ▶ use Pages Commons standard widgets as example

Demo



# a simple widget (,confidentiality')

```
<jcr:root ...  
    jcr:primaryType="cpp:Widget"  
    sling:resourceSuperType="composum/pages/commons/widget/select">  
<attributes  
    jcr:primaryType="nt:unstructured"  
    options="secret,confidential,internal,public"  
    default="internal" required="true"/>  
</jcr:root>  
  
<cpp:widget label="Confidentiality" property="confidentiality"  
    type="confidentiality" required="false"/>
```

Demo



# how it works together

- containers and elements
  - tree structure navigation, DnD between tools
- sites and pages
  - current release - activation / deactivation
- releases and publishing
  - markers for 'preview' / 'public' release

Demo



adaptTo()

try it...



# getting started

- <https://cloud.composum.com> (for testing)
  - request your own tenant
  - create your first site, content and component
- for local installation see:  
<https://www.composum.com/home/pages/setup.html>
- this talk - see:  
<https://www.composum.com/home/news/adaptTo.html>



# sources on GitHub

- Pages (with Components module)
  - ▶ <https://github.com/ist-dresden/comosum-pages>
- more Examples (e.g. the simple image)
  - ▶ <https://github.com/ist-dresden/comosum-prototype>



adaptTo()

# Appendix



# Taglib tags - Page tags

```
<html ${currentPage.htmlLangAttribute} ${currentPage.htmlDirAttribute}
      class="${currentPage.htmlClasses}" data-context-path="${slingRequest.contextPath}">
<cpp:head>
  <cpn:clientlib type="css" category="${currentPage.viewClientlibCategory}" />
  <cpn:clientlib type="css" test="${not empty currentPage.themeClientlibCategory}"
                  category="${currentPage.themeClientlibCategory}" />
  <cpn:clientlib type="css" test="${currentPage.editMode}"
                  category="${currentPage.editClientlibCategory}" />
</cpp:head>
<cpp:body>
  <sling:call script="body.jsp"/>
  <cpn:clientlib type="js" category="${currentPage.viewClientlibCategory}" />
  <cpn:clientlib type="js" test="${currentPage.editMode}"
                  category="${currentPage.editClientlibCategory}" />
</cpp:body>
</html>
```

Example



# the edit dialog

```
<cpp:editDialog var="model" type="com.composium.pages.common.model.Image"  
                title="Image Properties">  
    <cpp:widget label="Image" property="imageRef" type="imagefield"  
                hint="the path to the image asset in the repository"/>  
    <cpp:widget label="Title" property="title" type="textfield" i18n="true"  
                hint="the title text normally shown on mouse over"/>  
    <cpp:widget label="Alt Text" name="alt" type="textfield"  
                value="${model.altText}" i18n="true"  
                hint="the text for a text based view"/>  
</cpp:editDialog>
```

Example



# toolbar and tile example

```
<cpp:element var="model" type="com.composium.pages.common.model.GenericModel"
    cssBase="composium-pages-component-tile
```

Example



# content templates

- resolvers search path
- page and site templates
- predefined structure (copied)
- design rules (referenced)
- each template can reference another template
- use Pages Components templates as example

more...