

Designing with Plone 6: The Page Builder Experiment



Abstract

“This case study covers the development of booost CMS, a visual page builder built on top of Plone 6 Classic. It allows to create fully customized, high-quality websites directly through the web - without writing code, except for optional SCSS/CSS. booost.cms evolved from earlier work on Plone 5 and has since been used in more than 100 projects, ranging from small business sites to mid-sized platforms.”

“.. The talk shares my experience creating Boost CMS and explores how it leverages Plone Classic, Mosaic layouts, and TinyMCE to provide a flexible site-building toolkit. Key features include visual on-the-fly styling, custom content and layout tiles, and dynamic header and footer management ..”

“.. Beyond the feature set, I discuss the advantages and trade-offs of building a page builder on top of Plone: how it empowers non-developers, the risks involved in working against the framework, and how it fits into the broader theming ecosystem. I also briefly cover hosting and deployment (mxdev, Docker, Traefik, RelStorage) and a bit of AI.” - Thank you Chat-GPT for correcting my awful english language!

Disclaimer

- This talk does not always follow official Plone CMS documentation
- Purpose of this real life use case is delivering some ideas on theming & building small web projects on Plone
- Plone Classic talk - no volto inside
- May contain traces of AI

About me

- Integrator/dev & web designer
- Self employed since 2004
- Owner @ cloud19 web solutions & services, Melk - Lower Austria
- Plone since 2002, Hosting since 2005
- Plone sponsor since 2017
- Chess player, craft beer brewer, 2 kids



Origin story



- Idea of booost.cms goes back to a customer project @ NOEKU “Wireframe Plone”
- Setup with Plone 5 (till ~2023): every project a single buildout , customer addons with Plone-CLI / boiler plates, Docker, ZEO - lot of tasks to bring it up
- Q: Smaller customer projects are often very similar - change the process - or change the system?

Motivation behind

- I love to design for web and we all love Plone - why can't i build every web site with the technology i like?
- Conclusio: There is a need for a system to build smaller projects fast & visually - preferably python based
- Some Plone based solutions in past: Plone Mini Sites, theme roller..

Goals

- Over all: Build fully custom websites w/o code, w/o installation hassle
- Faster, visual workflow - Prototyping
- Easy migrations from Plone 4.0+
- Easy upgrades & longterm support
- 'Hybrid' solution (Tiles/blocks & dx CTs)
- Rapid, scalable deployments

Tech. requirements

- Plone CMS - bullet proof back bone
- Flexibility - Block based grid system
- Modern frontend toolkit, maybe a production ready theme - Bootstrap 5
- Server side rendering - no volto yet
- Hosting: Less installation hassle - Docker, k8s/Rancher, Traefik, relstorage

Start of the “experiment”

- Could Plone act as a real page builder?
- Plan: Implement another layer on top of Plone CMS that allows build & design websites quickly
- Formula: Plone + plone.app.mosaic + theme + theming tool = booost.cms

Receipe

1. CMS Evaluation
2. Theme Evaluation
3. Grid system Evaluation
4. Theme, UI kit integration into tiles
5. Dev of Content types & templates
6. Customisation tools
7. Hosting setup



Theme evaluation

- Bootstrap 5.3 / Sass (Scss) mandatory
- Multipurpose theme is a must - should cover many business cases
- Many components for max flexibility
- Node/npm & clean project structure
- Extensible & long term maintainable
- Documented

Template Intro Page
Mobile App Showcase
Product Landing
SaaS v.1
SaaS v.2
SaaS v.3
SaaS v.4 **New**
Shop Homepage v.1
Shop Homepage v.2 **New**
Marketing Agency

Landing Pages

Creative Agency
Conference (Event)
Web Studio
Corporate
Insurance Company **New**
Business Consulting
Coworking Space
Yoga Studio
Influencer
Blog Homepage

Content

- Typography
- Icon font
- Code
- Images & figures
- Tables

Components

- Accordion
- Alerts
- Badges
- Blog components
- Breadcrumb
- Buttons
- Button group
- Cards: Basic
- Cards: Hover effects
- Carousel (Slider)
- Charts
- Collapse
- Countdown
- Date / time picker
- Dropdowns
- Event calendar

- Forms
- Gallery (Lightbox)
- Hotspots
- Image comparison slider
- Input group
- List group
- Maps
- Modal
- Navbar
- Offcanvas

- Progress
- Shop components
- Social buttons
- Spinners
- Steps
- Tabs
- Team
- Testimonials / reviews
- Toasts
- Tooltips
- Video popup

Auth pages

- Overview
- Settings
- Billing
- Orders
- Earnings
- Chat (Messages)
- Favorites (Wishlist)

Portfolio ➤
Shop ➤
Blog ➤
About ➤
Services ➤
Pricing
Contacts ➤
Specialty Pages ➤

List View v.1
List View v.2
Grid View v.1
Grid View v.2
Slider View
Single Project v.1
Single Project v.2

Theme pages & components

Pros / Cons of integrating a commercial bs theme

- Pro: Responsive & tested components - ready to use
- Pro: Designers are guided & can re-use existing layouts
- Pro: no 'pixel pushing' webdesign - more consistent
- Pro: Work is done by someone else (\$)
- Con: Limited structures
- Con: Still a lot of work - will it pay off?

Demo

- Chess Club Demo (5 min.)

Plone Classic Theming Overview

- Lot of theming approaches in past
- Bootstrap 5 in Plone FTW!
- Traditional theming great for bigger sites
- No Diazo (too scientific for me)
- SCSS/LESS is a must

Dev principles

- No buildout, mxdev is the way to go
- No monkey code, quality features
- Monolithic as possible - not too granular
- Lightweight upgrade strategy
- No Diazo
- No fight against the framework (..)

Architecture

- Plone 6 Classic & plone.app.mosaic as base, some addons (easyform, pdfjs)
- Frontend replacement - Ready to use multipurpose Bootstrap 5.3 theme
- Custom tiles all over (Layout, content)
- Dynamic header, footer management
- Lot of configuration options (400+ records in registry)

Layout Tiles

- Mostly for footer & header rendering
- Render the html fragments of the bootstrap theme (Logo, Nav, etc.)
- Special tiles for GDPR, customizer, analytics, scroll arrow etc.
- Tile portlet: reuse of layout tiles

Content Tiles

- Listing tiles (Accordion, Slider, Map), render CTs - extended catalog records
- Cards (10 variants) cover most needs
- Lot of configuration (Animation, SVG, Parallax, CSS classes)
- Tile Portlet: can render layout tiles

Content Types

- 2 Mosaic Page types: Page & Plone Site (DX Site Root) -> layout_view
- Other types have fixed mosaic layouts
- Classic Plone dx CTs (Migration!)

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  lang="en"
  xml:lang="en"
  data-layout="./@@page-site-layout">
<body>
<main class="page-wrapper boost-layout layout-document" data-panel="content">
  <section class="container pt-5 pb-sm-3 mb-2 mb-md-3 mb-lg-4 mb-xxl-5">
    <div data-tile="./@@boost.tiles.titlebar"/>
  </section>
  <section data-tile="./@@boost.tiles.leadimage"/>
  <section data-tile="./@@boost.tiles.slideshow"/>
  <section class="container pb-5 mb-md-2 mb-lg-3 mb-xxl-4">
    <div class="row pt-xxl-2">
      <aside class="col-lg-3 offset-xl-1 pt-4 pt-lg-0 order-lg-1" style="margin-top: -13rem;">
        <div class="position-sticky top-0 mt-2 mt-md-3 mt-lg-0" style="padding-top: 13rem;">
          <div data-tile="./@@boost.tiles.tableofcontents"></div>
        </div>
      </aside>
      <div class="col-lg-9 col-xl-8 pe-lg-4 pe-xl-0">
        <div class="mosaic-tile mosaic-IDublinCore-description-tile">
          <div class="mosaic-tile-content">
            <div data-tile="./@@plone.app.standardtiles.field?field=IDublinCore-description"></div>
          </div>
        </div>

        <div class="movable removable mosaic-tile mosaic-IRichTextBehavior-text-tile">
          <div class="mosaic-tile-content">
            <div data-tile="./@@plone.app.standardtiles.field?field=IRichTextBehavior-text"></div>
          </div>
        </div>

        <div data-tile="./@@boost.tiles.tags"></div>
        <div data-tile="./@@boost.tiles.socialsharing"></div>
      </div>
      <div data-tile="./@@boost.tiles.lightbox"></div>
    </div>
  </section>
  <section data-tile="./@@boost.tiles.discussion"/>
  <section data-tile="./@@boost.tiles.relatedpersons"></section>
  <section data-tile="./@@boost.tiles.relateditems"></section>

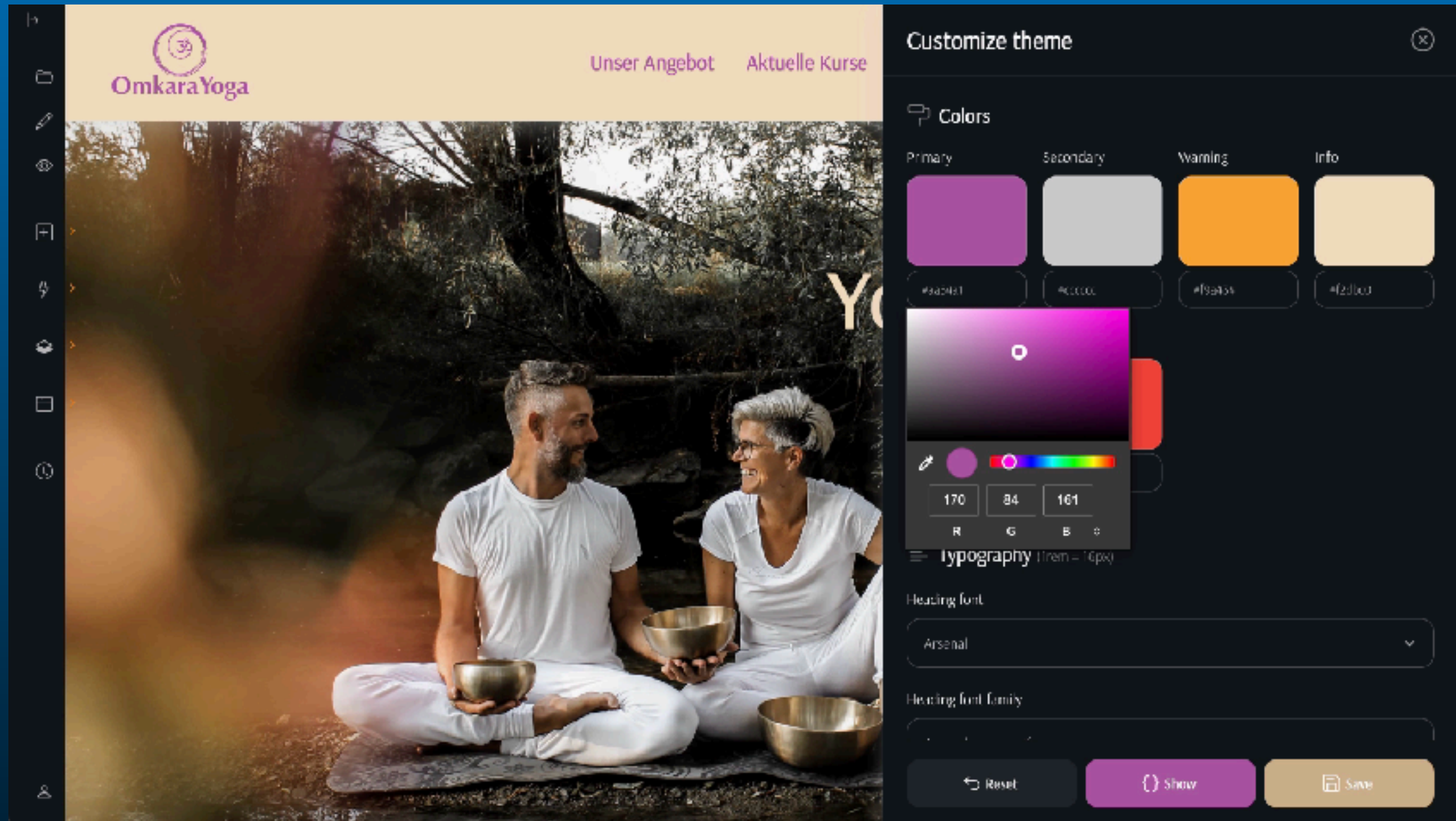
</main>
</body>
</html>

```

layouts/content/document.html

Tile architecture

- Base/mixin classes for Animation, UI, Image UI, Background features, Parallax, Grid, Settings (global, local)
- 'Item factory' (decorated catalog records)
- Layout tiles (re-useable), content tiles (banner, card, form proxy, image)
- Listing tiles: Listing (+50 views), mostly rendering content types



Theming Demo

Theming approach

- Customisation with SCSS root vars most efficient, CSS inserts should be possible
- Optional: Theming by configuration (replacing css classes)
- Frontend setup: Node, Npm, Sass, BS
- Dynamic Header, footer
- Lot of configuration

Styling process

- SCSS root vars are kept in local storage until save - on the fly styling
- vars are stored in registry on save and local storage get cleared, css is saved as @@custom.css (custom_css of plone.app.theming)
- CSS code editor - gets joined with @@custom.css

THEME SETTINGS

[Site Setup](#)

Themes

Advanced settings

ADVANCED SETTINGS

Theme details

Theme base

Custom Styles

Custom CSS

Define your own custom CSS in the field below. This is a good place for quick customizations of things like colors and the toolbar. Definitions here will override previously defined CSS of Plone. Please use this only for small customizations, as it is hard to keep track of changes here. For bigger changes you most likely want to customize a full theme and make your changes there.

[Theming of Classic UI documentation](#)

```
:root{--ar-primary:#000;--ar-primary-rgb:0,0,0;--ar-secondary:#edf1f5;--ar-secondary-rgb:237,241,245;--ar-warning-rgb:254,223,89;--ar-info-rgb:63,127,202;--ar-success-rgb:63,202,144;--ar-danger-rgb:237,80,80;--ar-body-font-family: Open Sans,sans-serif;--ar-heading-font-family: Gagalin,sans-serif;--ar-root-font-size:1.15rem;--ar-body-font-size:1.15rem;--ar-border-width:2px;--ar-border-radius:0.25rem;--ar-border-radius-sm:calc(var(--ar-border-radius)*0.75);--ar-border-radius-lg:calc(var(--ar-border-radius)*1.25);--ar-border-radius-xl:calc(var(--ar-border-radius)*1.5);--ar-border-radius-2xl:calc(var(--ar-border-radius)*2.25);--ar-link-color:#000;--ar-link-hover-color:#000000;}.btn-primary{--ar-btn-bg:#000;--ar-btn-border-color:#000;--ar-btn-hover-bg:#000000;--ar-btn-hover-border-color:#000000;--ar-btn-active-bg:#000000;--ar-btn-active-border-color:#000000;--ar-btn-disabled-bg:#000;--ar-btn-disabled-border-color:#000;}.btn-secondary{--ar-btn-bg:#edf1f5;--ar-btn-border-color:#edf1f5;--ar-btn-disabled-bg:#edf1f5;--ar-btn-disabled-border-color:#edf1f5;}.btn-warning{--ar-btn-bg:#fedf59;--ar-btn-border-color:#fedf59;--ar-btn-disabled-bg:#fedf59;--ar-btn-disabled-border-color:#fedf59;}.btn-info{--ar-btn-bg:#3f7fca;--ar-btn-border-color:#3f7fca;--ar-btn-disabled-bg:#3f7fca;--ar-btn-disabled-border-color:#3f7fca;}.btn-success{--ar-btn-bg:#3fca90;--ar-btn-border-color:#3fca90;--ar-btn-disabled-bg:#3fca90;--ar-btn-disabled-border-color:#3fca90;}.btn-danger{--ar-btn-bg:#ed5050;--ar-btn-border-color:#ed5050;--ar-btn-disabled-bg:#ed5050;--ar-btn-disabled-border-color:#ed5050;}.btn-outline-primary{--ar-btn-color:#000;--ar-btn-border-color:#000;--ar-btn-hover-bg:#000;--ar-btn-hover-border-color:#000;--ar-btn-active-bg:#000;--ar-btn-active-border-color:#000;--ar-btn-disabled-color:#000;--ar-btn-disabled-border-color:#000;}.btn-outline-secondary{--ar-btn-color:#edf1f5;--ar-btn-border-color:#edf1f5;--ar-btn-hover-bg:#edf1f5;--ar-btn-hover-border-color:#edf1f5;--ar-btn-active-bg:#edf1f5;--ar-btn-active-border-color:#edf1f5;--ar-btn-disabled-color:#edf1f5;--ar-btn-disabled-border-color:#edf1f5;}.btn-outline-warning{--ar-btn-color:#fedf59;--ar-btn-border-color:#fedf59;--ar-btn-hover-bg:#fedf59;--ar-btn-hover-border-color:#fedf59;--ar-btn-active-bg:#fedf59;--ar-btn-active-border-color:#fedf59;--ar-btn-disabled-color:#fedf59;--ar-btn-disabled-border-color:#fedf59;}.btn-outline-info{--ar-btn-color:#3f7fca;--ar-btn-border-color:#3f7fca;--ar-btn-hover-bg:#3f7fca;--ar-btn-hover-border-color:#3f7fca;--ar-btn-active-bg:#3f7fca;--ar-btn-active-border-color:#3f7fca;--ar-btn-disabled-color:#3f7fca;--ar-btn-disabled-border-color:#3f7fca;}.btn-outline-success{--ar-btn-color:#3fca90;--ar-btn-border-color:#3fca90;--ar-btn-hover-bg:#3fca90;--ar-btn-hover-border-color:#3fca90;--ar-btn-active-bg:#3fca90;--ar-btn-active-border-color:#3fca90;--ar-btn-disabled-color:#3fca90;--ar-btn-disabled-border-color:#3fca90;}.btn-outline-danger{--ar-btn-color:#ed5050;--ar-btn-border-color:#ed5050;--ar-btn-hover-bg:#ed5050;--ar-btn-hover-border-color:#ed5050;--ar-btn-active-bg:#ed5050;--ar-btn-active-border-color:#ed5050;--ar-btn-disabled-color:#ed5050;--ar-btn-disabled-border-color:#ed5050;}
```

Custom styles /custom.css

Custom CSS

CSS

User defined CSS

```
.navbar-toggler-icon, .navbar-toggler-icon::after,  
.navbar-toggler-icon::before {  
    background-color: var(--ar-primary) !important;  
}
```

CSS code editor widget

Difficulties

- Balance between flexibility & fixed structures
- Fighting the framework

Features

- Full integration of UI Kit (Components)
- Animations TTW (aos.js)
- Utility classes, Typography, UI icon font
- TinyMCE: Templates, ChatGPT plugin
- Color modes, swiper.js, leaflet.js ..

AI Usage (ChatGPT)

- Plugin for TinyMCE for AI text (collective.txt42.chatgpt - Github)
- JS Development: Script generation for CSS class update for visual editor
- AI Web design generator: Color palette and root variables upon Logo upload & target group (wip)

BearbeitenTabelleFormatAnsichtEinfügen

↶↷

Absatz

▼

B

I

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

≡≡≡≡

...

▶GPT Text

▶

▶

Beer yoga overview

Beer yoga is a trendy new fitness craze that combines the relaxation of yoga with the enjoyment of drinking beer. Participants can expect to enjoy a fun and light-hearted yoga session while sipping on their favorite brew. This unique fusion of exercise and indulgence is a great way to unwind and have a good time with friends. Just remember to drink responsibly and stay hydrated during your practice!

P

71 WÖRTER BETRIEBEN VON TINY

text/html

▼

AI usage - ChatGPT Plugin



Thought for 3m 31s >

Got it! | extracted the palette from the BRUCKNER logo and generated a matching tokenized CSS (same structure as before, with leading zeros and "Arsenal" as the font).

CSS

Copy

```
/* === BRUCKNER - Brand Design Tokens (derived from logo) === */
:root {
  /* Core palette */
  --ar-primary: #512937;          /* deep burgundy from wordmark */
  --ar-primary-rgb: 81, 41, 55;
  --ar-secondary: #9a9192;        /* cool medium gray from tagline/line */
  --ar-secondary-rgb: 154, 145, 146;

  /* Semantic states (neutral, harmonious with brand) */
  --ar-warning: #edcb50;
  --ar-warning-rgb: 237, 203, 80;
  --ar-info: #3f7fca;
  --ar-info-rgb: 63, 127, 202;
  --ar-success: #3fca90;
  --ar-success-rgb: 63, 202, 144;
  --ar-danger: #ed5050;
  --ar-danger-rgb: 237, 80, 80;

  /* Typography */
  --ar-body-font-family: "Arsenal", system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans";
  --ar-heading-font-family: "Arsenal", system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial, "Noto Sans";
}
```

AI usage - web design generator

Here's the **corrected UMD module** (includes your dark mode + `_ui` select token purge + nested targets + smart matching). Drop-in replacement:

js

Copy

```
/*!
 * form-edit-target-sync.js
 * - Highlights targets on focus
 * - FIELDSETS with data-fieldset containing "_ui": every field updates target classes.
 *   • SELECTS: purge only classes listed in <option> values (exact tokens), then add the new one
 *   • OTHER CONTROLS: purge by prefix groups derived from new value (e.g., text-, border-)
 *   • If select value == "--NOVALUE--": purge option tokens and add nothing
 * - OUTSIDE fieldsets: fields with "_css" in id or name -> class update; others -> text update
 * - SPECIAL: *_dark_mode fields toggle data-bs-theme="dark" on their mapped targets
 * - Single-field target lookup prefers name; fallbacks: base_icon -> base_ui -> base -> rawKey
 * - Supports nested & multiple targets, and repeated changes without leaking old classes
 */

(function (root, factory) {
  if (typeof define === 'function' && define.amd) {
    define([], factory); // AMD
  } else if (typeof exports === 'object') {
    module.exports = factory(); // CommonJS
  } else {
    root.FormEditTargetSync = factory(); // Browser global
  }
})(typeof self !== 'undefined' ? self : this, function () {

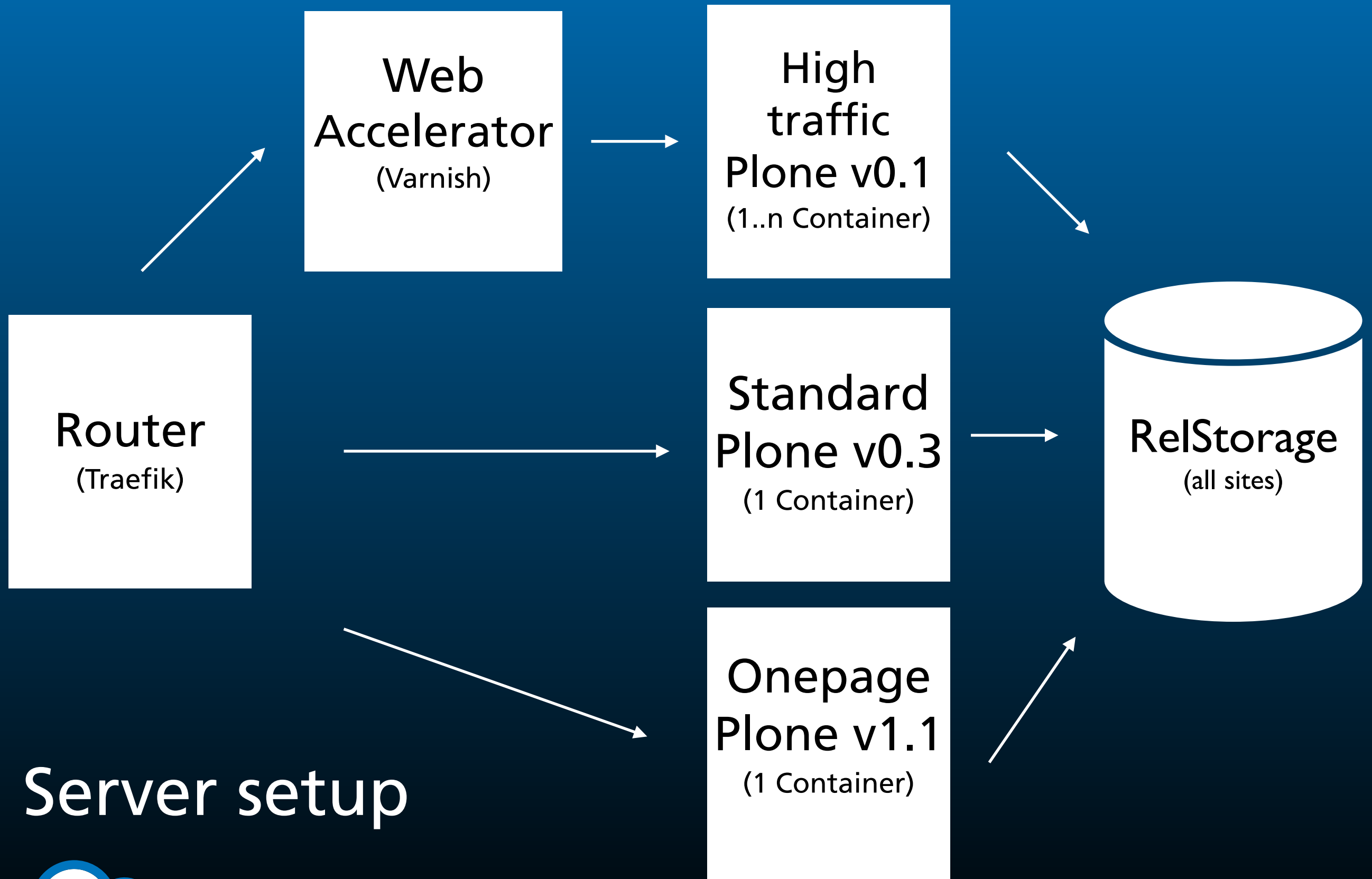
  function FormEditTargetSync(options = {}) {
    const CONTROL_SELECTOR = options.controlSelector || [
      'input.text-widget',
      'input.asciiline-field',
      'textarea.text-widget',
      'select.select-widget'
    ].join(',');

    const lastAppliedClasses = new WeakMap(); // control -> "classes string"
    const lastTargets = new WeakMap(); // control -> Set<Element>
  }
});
```

AI usage - Javascript generator

Deployment/Hosting

- mxdev (for dev and deployment)
- Docker
- Traefik
- Rancher -> k8s in future
- RelStorage instead of ZEO



Server setup

Deployment Process

1. Image build, new version:

```
docker buildx build. -t $(NS)/$(IMAGE_NAME):$(IMAGE_TAG) ..
```

2. Export Plone site as <project>.zexp

3. Copy .zexp to server & import to ZOPE

```
docker cp ~/<project>.zexp <container id>:/app/import
```

4. Deploy

```
rancher-compose -f docker-compose.yml -r rancher-compose.yml -p  
${IMAGE_NAME} up ..
```

Lessons learned

- Advantages: Visual design approach, faster prototyping, consistent design. RelStorage makes Hosting and long term support easier
- Disadvantages / Risks: Fighting the framework is a risk, maintaining a large site builder on top of an enterprise level CMS.

Thank you for you
attention!