cs_dynamicpages and behavior driven ~~Classic UI~~ Blicca layouts

Why we reinvented the wheel: cs_dynamicpages

CodeSyntax

Interneten adituak

Part 1: Why did we create this?

- **Modern Design Needs**: Customers provided Figma designs based on modular blocks.
- **Volto Experience:** We wanted the block-building feel in Plone Classic.
- **The "Mosaic" Challenge**: Other tools (Mosaic, contentsections) gave too much freedom.
- **Design Consistency**: Every row type has a fixed, professional style.
- **Editor Focus**: Editors only manage data, not layout or styling (kind of).

Why Another Layout Tool?

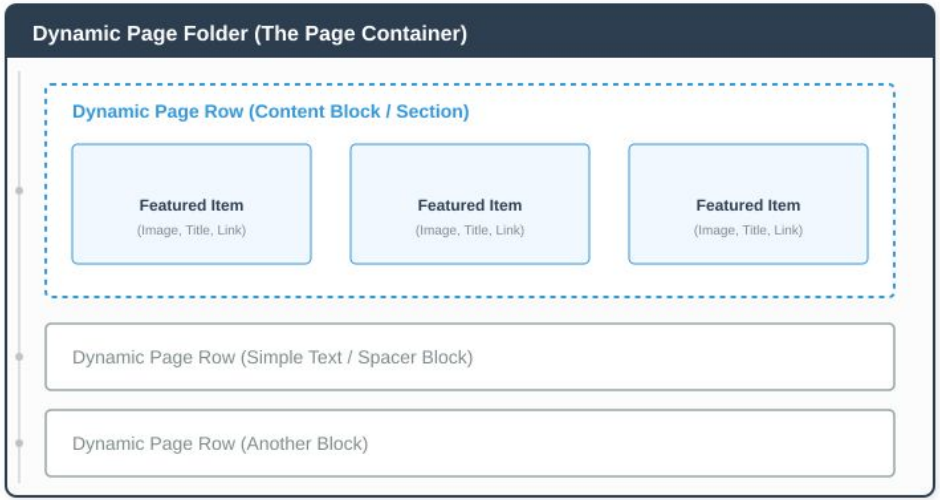
Framework	**Architecture**	**Main Drawback**
plone.app.mosaic	Layout templates, tiles & custom tiles	Complex layout engine, content is not *native Plone content*
collective.contentsections	Content-tree based	Proliferation of distinct *portal_types*
cs_dynamicpages	1 Container + 1 Row Types	Behavior driven, native views

How to activate it?

- **Simple setup**: Just change the display view.
- **Dynamic View**: Select it from the "Display" menu.
- **Works on**:
  - Folders
  - Plone Site root
  - Language Root Folders (LRF)
- **Ready to go**: Once activated, you can start adding blocks.

The Three Main Elements

1. **Dynamic Page Folder**: Where the rows live.
2. **Dynamic Page Row**: The building blocks (sections).
3. **Dynamic Page Row Featured**: Items inside a row



Total Control (Behaviors)

- **Common Visual Settings**: Options shared by all row types.
- **Key Settings**:
  - Row Width (Full / Contained / Narrow).
  - Vertical Spacing (Margins and paddings above/below).
  - Custom CSS classes for specific styling.

Default Row Types at a Glance

- **Content**: Title/Description, Text & Image, Accordion(FAQs)
- **Hero:** Slider (Carousels), Featured View, Featured Overlay
- **Grids & Listings**: Features Grid, Query Columns (Collections)
- **Structure & UI**: Horizontal Rules, Spacers

Row Type Configuration Control Panel

- **Field Visibility**: Restrict which behavior fields are displayed per row view to prevent field proliferation.
- **Row Activation**: Enable or disable specific row types across the site.
- **Export/Import**: Move row configurations between dev, staging, and prod via JSON.

Content Insights & Usage Tracking

- **Global Overview**: Access via a Control Panel.
- **Usage Statistics**: See exactly where each row type is being used across the entire site.
- **Better Management**: Help editors understand how their content is structured globally.

Editor Tools: Speed up your workflow

- **Clear Cache Button**: Purge Varnish cache for the current page.
- **Edit Mode Toggle**: Enter the visual page builder.
- **Easy Reordering**:
  - Move rows **Up and Down** with buttons.
  - **Drag & Drop** support.
  - **Flexible Insertion**: Add new rows anywhere on the page.

Content Lifecycle & Workflow

- **Standard Plone Workflows**: Every Row and Featured item has its own state.
- **Independent Publishing**: Publish or hide individual blocks.
- **Security First**: Uses standard Plone permissions.

SEO and Search Indexing

- **Invisible Blocks**: Rows and items are removed from Plone sitemap and listings.
- **Full-Text Search**: All text is indexed in the main page (Folder) using Plone's native text indexing.
- **Related Images**: Images are linked from your library, keeping URLs clean and SEO-friendly.
- **Performance**: Support for `fetchpriority` & lazy-loading to optimize LCP.

Templates: Reusable layouts

- **Save & Reuse**: Save any configured `DynamicPageFolder` layout as a named template.
- **Instant Deployment**: Build new landing pages or Language Root Folders (LRFs) in seconds selecting which template to use.
- **Dedicated Template Control Panel**:
  - Independent management interface strictly for page templates.
  - Audit all saved templates across the site.
  - Delete retired or campaign-specific templates to keep options clean for editors.
  - Templates save reference to a content-object, so each time a template is applied the current status of that page is copied (no more *Lorem ipsum*).

Theme & Bootstrap: Integrated with Plone 6

- **Bootstrap Based**: Built with the same framework as Plone 6 Classic.
- **Default Styles and Classes**: Ready to use out of the box.
- **Fully Customizable**: Easy to override and adapt to your specific project theme.

Create a new row (3 steps)

1. **Schema**: Register a standard Plone Behavior for custom fields.
2. **View**: Create a standard ZCML view prefixed with `cs_dynamicpages-`.
3. **Control Panel**: Toggle view visibility and restrict behavior fields.

Summary

- **Simple for Editors**: No technical skills needed.
- **Flexible for Designers**: Easy to style with Bootstrap.
- **Solid for Developers**: Based on standard Plone technologies.

Future developments

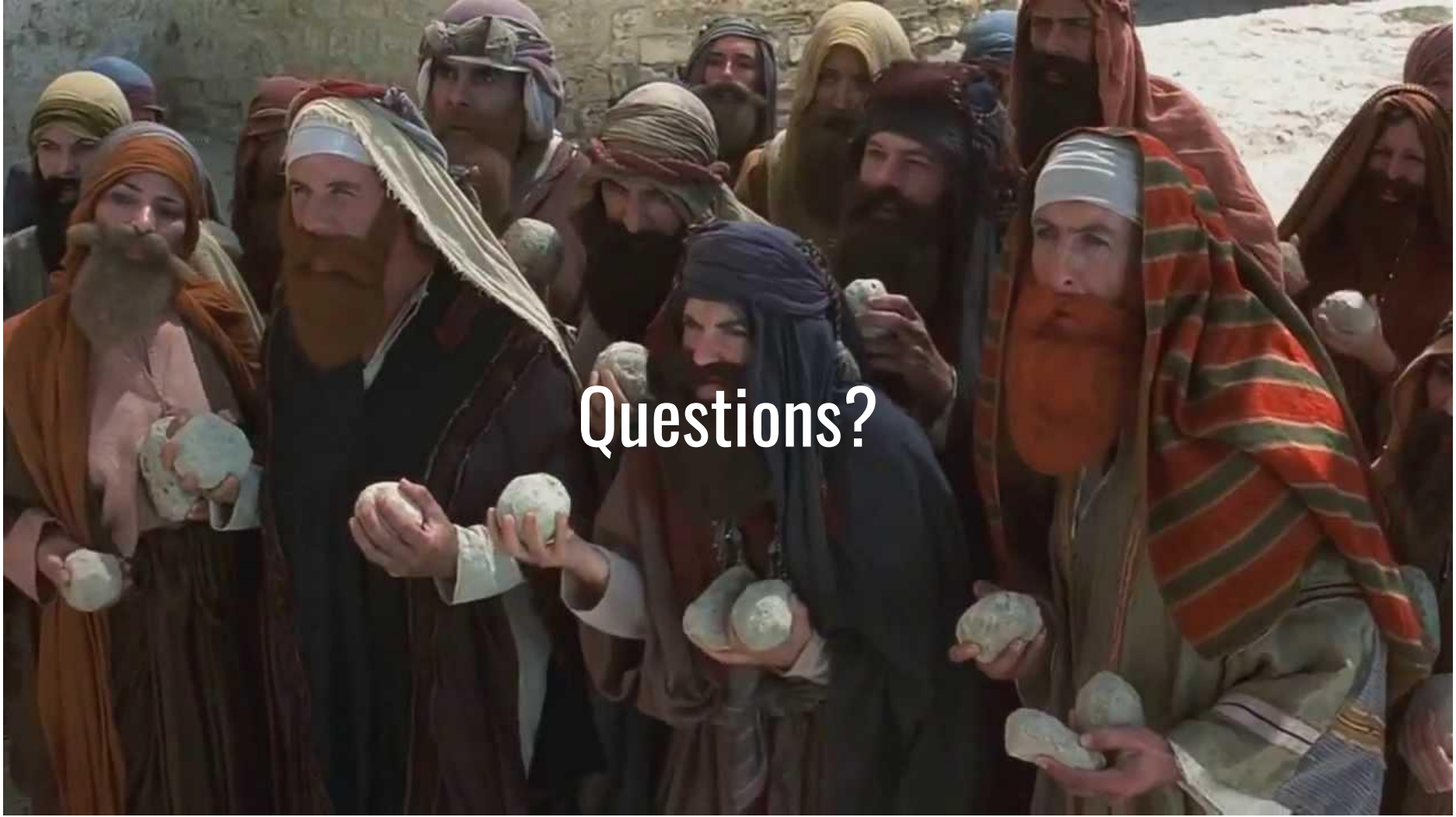
- **Single content-type**: DynamicPageRow “everything is a block”
  - Rename Row to Block? 😁
  - UX Challenges: editing interface?
  - How to migrate existing sites?
- **Simplify the workflow**
  - Simple Publication Workflow is too much (private - published only)?
- Specific **permissions for template management?**
- Specific **permissions for UI related fields?**
- MCP

Open Source & QA

- **Repository**: https://github.com/codesyntax/cs_dynamicpages
- **Contributions**: Issues & PRs welcome!

Real-World Examples: see it in action

- **PloneConf 2026 Home**: Built entirely with default row types.
- **Custom Projects**: Examples of professional designs from CodeSyntax customers.
- **Versatility**: From simple landing pages to complex corporate sites.


Questions?