Advanced layout and block management

Ivo Lukač *@ Drupal Heart Camp, Zagreb 2017*

netgen

About me

- **Ivo Lukač**, co-founder of Netgen, Croatia
- 16 years of **experience building complex content-centric web solutions,** mostly using eZ Publish
- from 2010 working mostly with clients and partners in Germany, Switzerland, Norway, Austria, Kuwait, Thailand etc.
- Roles: developer, site builder, architect, consultant, project manager, evangelist, speaker, event organiser, business developer, … and more :)

About Netgen

- Web agency, Zagreb, Croatia, ~20 employees
- focused on eZ Publish CMS since 2004 and Symfony PHP framework since 2013
- Organising summer camps since 2012 :)
- Clients with complex web projects in Norway, Switzerland, Germany, ….

HOW IT ALL STARTED

Our building process was too slow, with too much waste between phases


The biggest gap is between design and development phases, as they use completely different tools

Jason Pamental, UX expert

“Web projects’ fatal flaw is when **designers define specs before talking to developers**”

We wanted to improve the process of building complex web solutions and make it more efficient!

Improving efficiency for each phase is good but with no significant overall gains

Other ways

- SaaS solutions like Webflow, Wix, etc
  - not on enterprise level
  - hard to extend
- prototyping/wireframing tools exporting to HTML/CSS
  - what about backend?
  - responsiveness still hard
- CMS specific tools
  - usually focused on one page, one persona

We wanted to find the intersection of most phases and improve there

So what is common ground for backend, frontend, UX design and content?

We figured its about layouts and blocks :)

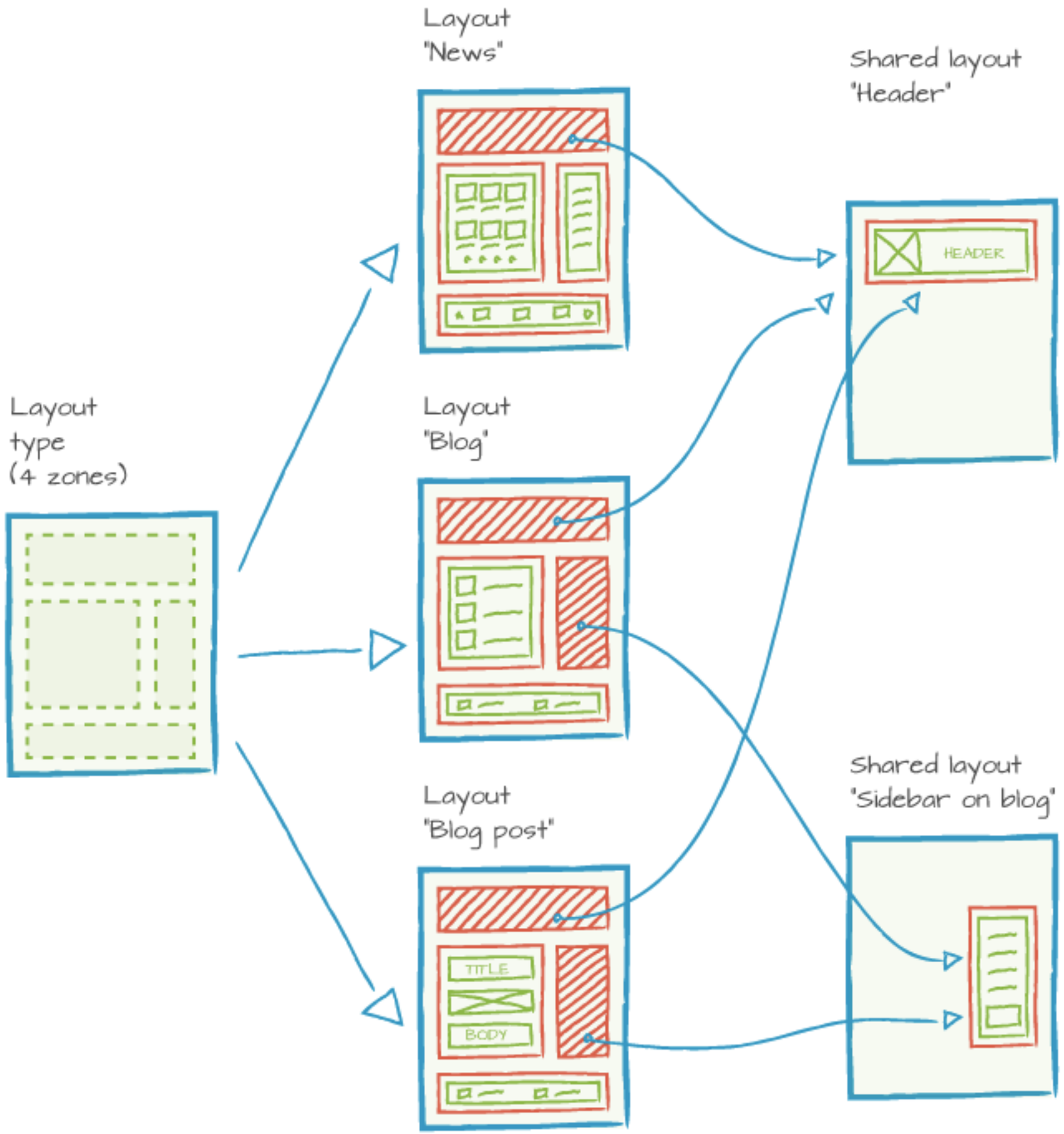
Layouts and blocks

- backend developers implement features that need to be placed somewhere on the interface layout, reuse when necessary
- frontend developers apply design on HTML markup which describes the layout
- editors place content in prepared blocks
- UX designer define the interface with set of layouts and blocks

MAIN CONCEPTS

Defining layouts

- **Layout** - responsible to render the layout of a page with all **Blocks** added to it. It is instantiated from a **Layout Type** from which it gets the template and list of **Zones** in which **Blocks** can be placed
- **Zone** - a placeholder for **Blocks** which are added to a **Layout**. There could be more Zones in a Layout. More Blocks can be added in one Zone, they are rendered one after other.
- **Shared Layout** - a special layout that can’t be applied to any page directly, its purpose is to hold Blocks that can be reused in other Layouts
- **Zone linking** - a concept of showing a **Zone** from a Shared Layout as a Zone in regular Layout.


Layout
"News"
Shared layout
"Header"
HEADER
Layout
type
(4 zones)
Layout
"Blog"
Shared layout
"Sidebar on blog"
Layout
"Blog post"
TITLE
BODY

Defining blocks

- **Block** - responsible for handling specific features based on its definition. It renders the content using a defined **Block View**. Certain blocks have **Block Items** which are filled from Collections
- **Collection** - used to fill the list of Block Items. Collection can be manual or dynamic. In the case of manual collection the items are picked from a backend system by the editor. In the case of dynamic collection a **Query** can be chosen and given parameters to fetch the data from the backend.
- **Container** (Column, 2 columns, etc) - is a special kind of Block which purpose is to hold other Blocks and render them.

Resolving vs mapping

- **Layout Resolver** - a module that is deciding what **Layout** will be used for which request. It traverses **Layout mappings** and pick the first one which fits the conditions. It works on top of Symfony routing
- **Layout mappings** - a configuration that defines which **Layout** is applied to which **Targets** under what **Conditions**
- **Targets** - abstraction which defines one or more URLs in a generic way or backend specific way
- **Conditions** - additional conditions which need to match for **Layout** to be chosen. Could be anything known in request context

NETGEN LAYOUTS!

TA-DAA!

Benefits

- makes web design & development process more agile
- reduces the time needed to design and develop a site
- produces less waste
- makes site maintenance easier
- gives quicker time to market
- provides head to headless
- gives a single collaboration point for UX designers, developers and editors

Currently

- adding more features (like caching, user policies, translations etc.)
- finished several projects using Layouts, starting few more
- preparing a second release for selected partners and their pilot projects (Germany, Italy, Spain)

Future

- **open source** the core
- make it **sustainable** with commercial add ons
- integrate on deeper level with all important Symfony based solutions, **Drupal as well**
- make integrating with everything else easy via REST/SOAP, …

Thank you

ivo@netgen.hr

ilukac.com/twitter

ilukac.com/linkedin