Themes and Layouts

Themes

What is theme?

- Responsible for UI representation of a page
- Think of a theme as skin for your application
- May include styles (CSS / SCSS) and JS
- May override page parts (layout updates)
- May have a parent theme
- Defined at global and website configuration
- Backend theme != Frontend theme
  - oro_theme vs oro_layout

Theme - demonstration

- UI - configuration
- Code
  - config.yml (oro_layout.active_theme)
  - theme.yml
  - ThemeManager and OroLayoutExtension

Theme definition

- Resources/views/layouts/<theme>/theme.yml
- May have parent theme

https://github.com/oroinc/platform/blob/3.1/src/Oro/Bundle/LayoutBundle/Resources/doc/theme_definition.md

https://github.com/oroinc/orocommerce/blob/3.1/src/Oro/Bundle/CustomThemeBundle/Resources/views/layouts/custom/theme.yml

Standard themes

- blank [link]
- default [link]
- custom [link]

Demonstration >>>

How to set a theme

- Application config app/config/config.yml
- System configuration
  - Global level
  - Website level
- Installation fixture
  - $configManager->set()

Demonstration >>>

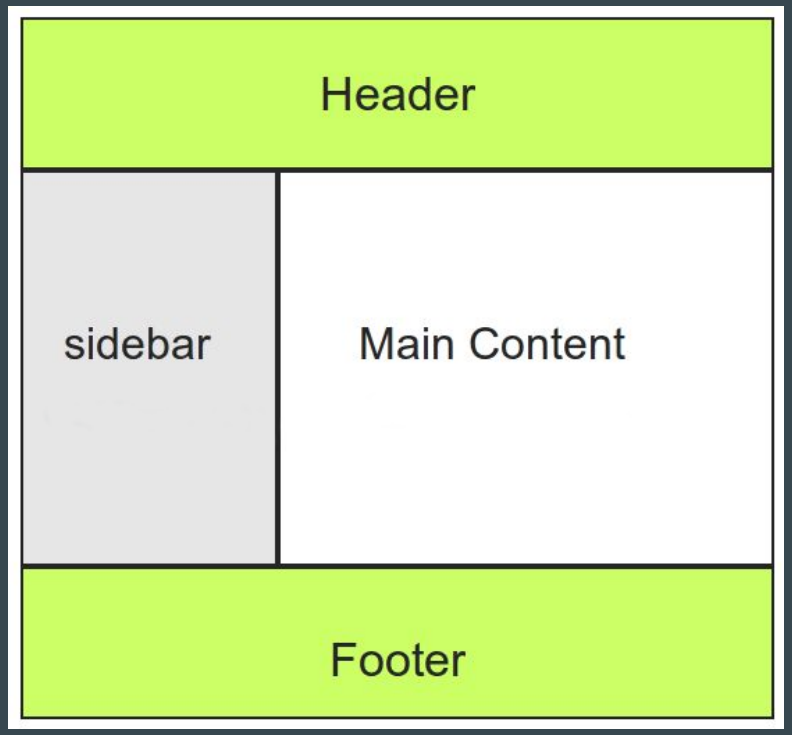
Layouts

Layout and blocks

- View layer of MVC
- Tree structure of blocks [link]
- Blocks have options
- Blocks can be added, moved or removed
- Blocks render pieces of HTML

[Documentation]

Layout and blocks



- root
  - header
  - body
    - sidebar
    - main content
  - footer

Directory structure

Resources/views/layouts/<theme>/

- config [documentation]
- imports [documentation]
  - Define once, use multiple times without copy pasting
- page
  - Layout updates for every page within theme
- <routeName>
  - Layout updates for specific route

[Example]

Layout updates

- Set of actions that should be performed with the layout in order to customize the page
- Update layout tree structure
- Update block options
- Can import other layout updates

[Example]

Layout block types

- Similar to Symfony form types [example]
- Define type of a content
- Define options
- Might be added without a new class
- tag: { name: layout.block_type, alias: xxx }

https://github.com/oroinc/orocommerce/blob/1.6/src/Oro/Bundle/ProductBundle/Resources/config/block_types.yml

Layout actions

- Tree manipulations
- Options manipulations
- Examples
  - ‘@setBlockTheme’
  - ‘@setOption’
  - ‘@remove’
- [All actions]

Layout context

- Contains UI configuration options [example]
  - Shared between different components of the layout
- Does not contain data
- Accessing context [link]
  - from the BlockInterface instance
  - using the Symfony expression component
- @Layout(vars={“name”}) - example
- Context configurators
- bin/console oro:layout:debug --context

[Documentation]

Layout data

- Same layout, different data (ex. product page)
- From layout context's data collection
  - e.g. from controller
- From data providers

[Documentation]

Data providers

- Provide additional data
- Have methods to get data
  - method should begin with get, has or is.
- Tag: { name: layout.data_provider, alias: xxx }
- Example
  - Provider class
  - Services.yml
  - Usage

Layout imports

- Add the same tree several times
- Custom namespace (block prefix)
- Custom options

[Definition]

[Usage]

[Documentation]

Layout conditions

- Defined in the layout update file
- Conditions must be satisfied for layout update to be executed
- Only context variables
- ‘and’, ‘or’ can be used to combine conditions
- Examples
  - Breadcrumbs
  - Customer address count

How to use layouts from controller

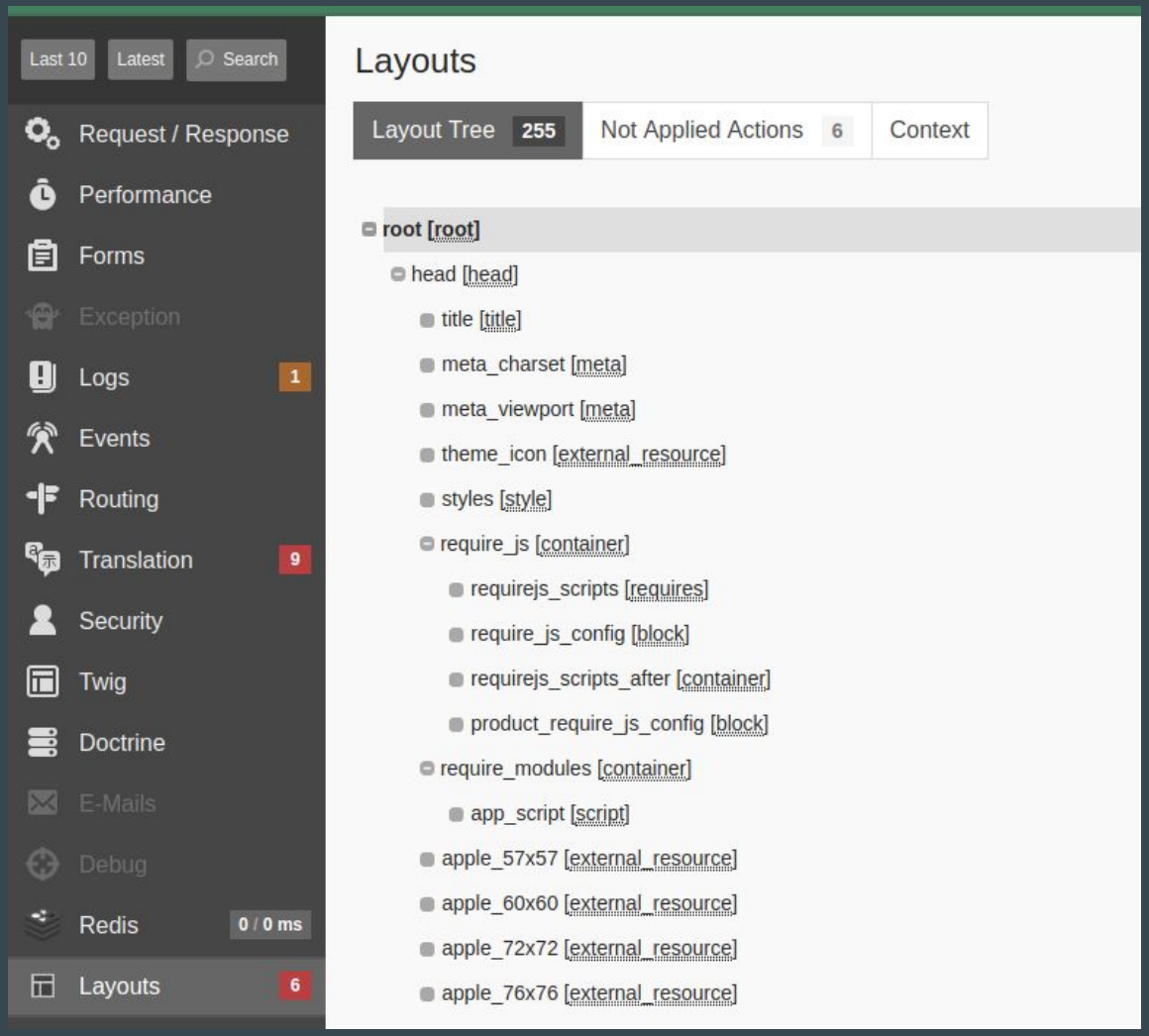
- @Layout annotation
- Data
- Context

Demonstration >>>

Debugging themes and layouts

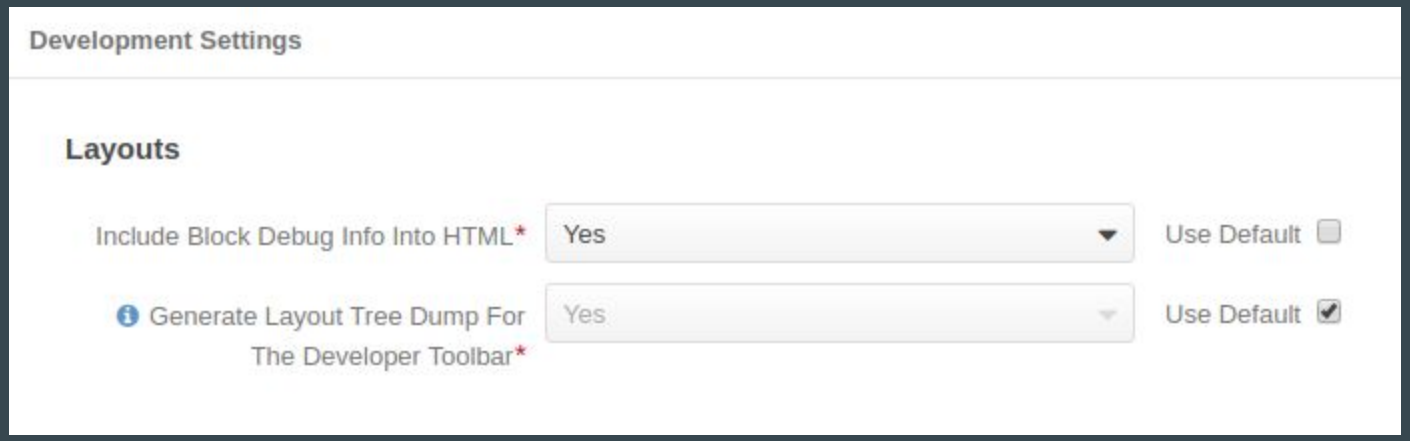
- Dev toolbar
- System configuration
  - Include Block Debug Info Into HTML
- Theme listener [link]
- Layout listener [link]

Debugging themes and layouts



Debugging themes and layouts

General Setup / Development Settings



Best Practices

- Keep theme in the separate bundle and put there all standard styles and elements
- Don’t be afraid to make big blocks with big templates
- Data providers must not change the state of the application - their only purpose is to get a data
- If some business logic is needed inside the template and there is no easy way to call if from controller or model layer then create new data provider to proxy calls of service methods