

Applying Object-Oriented Principles to Site Building



Mentored Sprints



- Sunday, August 5th from 9am-1pm
- Music and Dance Studio
Room 120 of the King Center.

Caleb Thorne



 @calebthorne

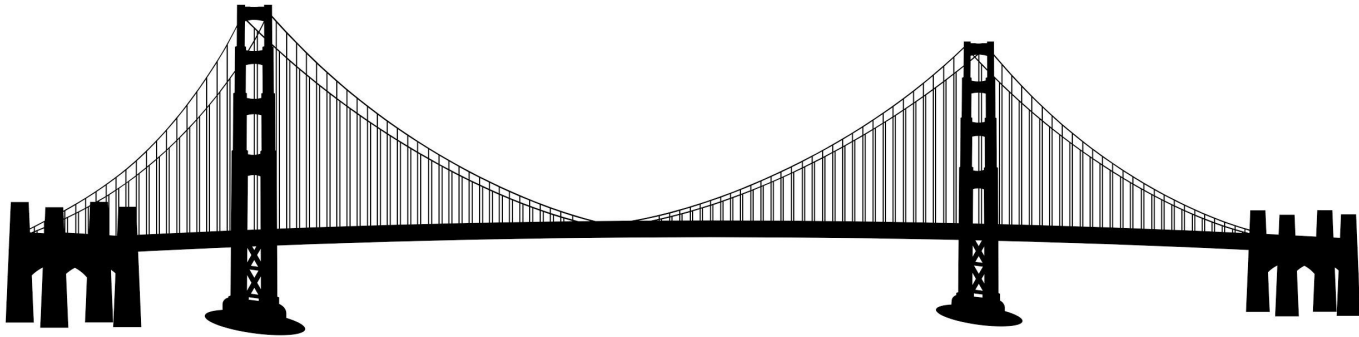
 www.drupal.org/u/draenen

 www.calebthorne.com



Why?

1. Intermediate/Advanced Site builders
2. Building bridges between devs and non-devs



What is Object-Oriented Programming?

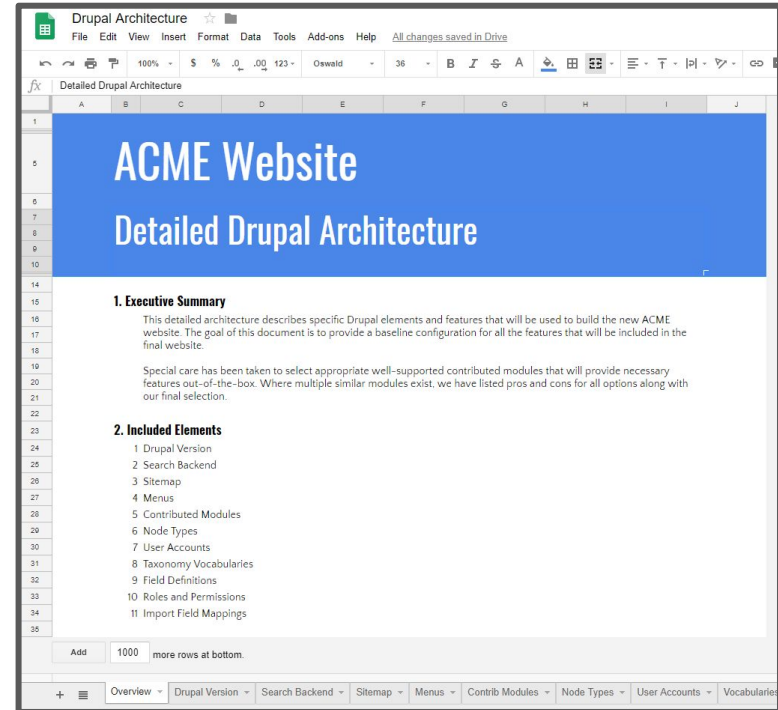


*A programming paradigm that emphasises the use of data structures (objects) containing both properties (fields) and operations (functions) to **model real-world solutions.***

What is Drupal Architecture

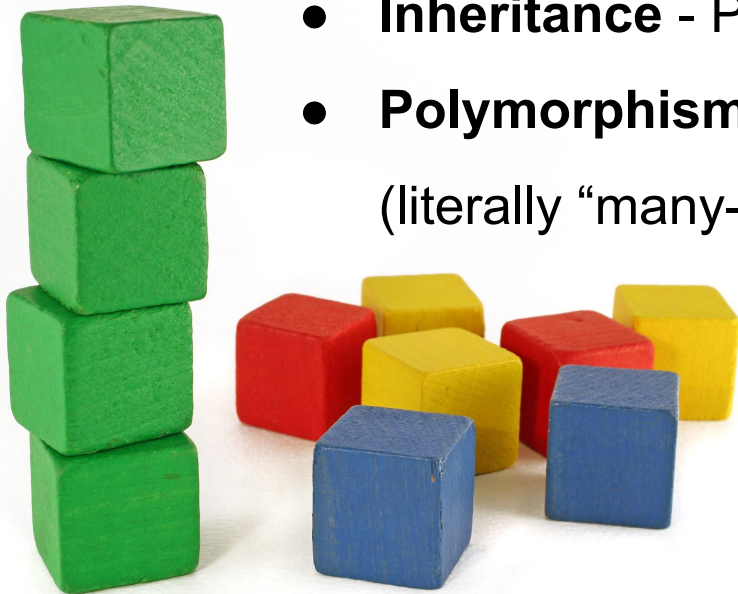


- How you structure content
- How you configure the display
- How you organize functionality



Object-Oriented Principles

- **Abstraction** - Models of real life
- **Encapsulation** - Data protection
- **Inheritance** - Parent-child relationships between objects
- **Polymorphism** - Interchangeable parts
(literally “many-forms”)



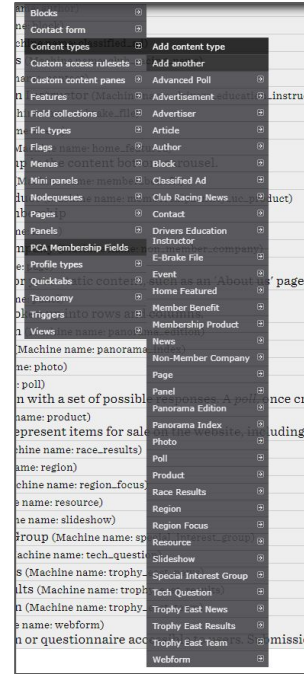
Abstraction

Abstraction are models of real life

- Entities and bundles (Users, Nodes, Groups, etc)
- Define a common interface
- Abstractions should be as generic but accurate as possible

Abstraction Application

1. Plan. Resist the urge to jump right in
2. Don't create too many content types
 - Use Taxonomy to categorize
 - Reduces overhead



Encapsulation

Encapsulation assists with data protection

- Limit access to those who need it
- Enforce data validation rules

Encapsulation Application



3. Make use of form modes (similar to view modes)
https://www.drupal.org/project/form_mode_manager
4. Use field permissions
https://www.drupal.org/project/field_permissions
5. Use field validation
https://www.drupal.org/project/field_validation

Form Modes

Form modes ☆

Home » Administration » Structure » Display modes

[+ Add form mode](#)

Content

NAME	OPERATIONS
Administrator	Edit ▼
Contributor	Edit ▼

[Add new Content form mode](#)

User

NAME	OPERATIONS
Register	Edit ▼

[Add new User form mode](#)

Manage form display ☆

[Edit](#) [Manage fields](#) [Manage form display](#) [Manage display](#)

[Default](#) [Administrator](#) [Contributor](#)

Home » Administration » Structure » Content types » Article

FIELD	WEIGHT	PARENT	REGION
Content			
Title	<input type="text" value="0"/>	- None - ▼	Content ▼
Body	<input type="text" value="1"/>	- None - ▼	Content ▼
Tags	<input type="text" value="2"/>	- None - ▼	Content ▼

Field Validation

Edit *Not Joomla* rule


[Home](#) » [Administration](#) » [Structure](#) » [Field validation rule set](#) » [Edit field validation rule set](#)

Field Validation Rule title *

Field name *

Column of field *

Blacklisted words *



Specify illegal words, seperated by commas. Make sure to escape reserved regex characters with an escape (\) character.

Error message *

[Update rule](#) [Cancel](#)

Inheritance

Parent-child relationships

- Children inherit all properties and operations from the parent
- Defines an “IS-A” relationship (A Node “IS-AN” Entity)

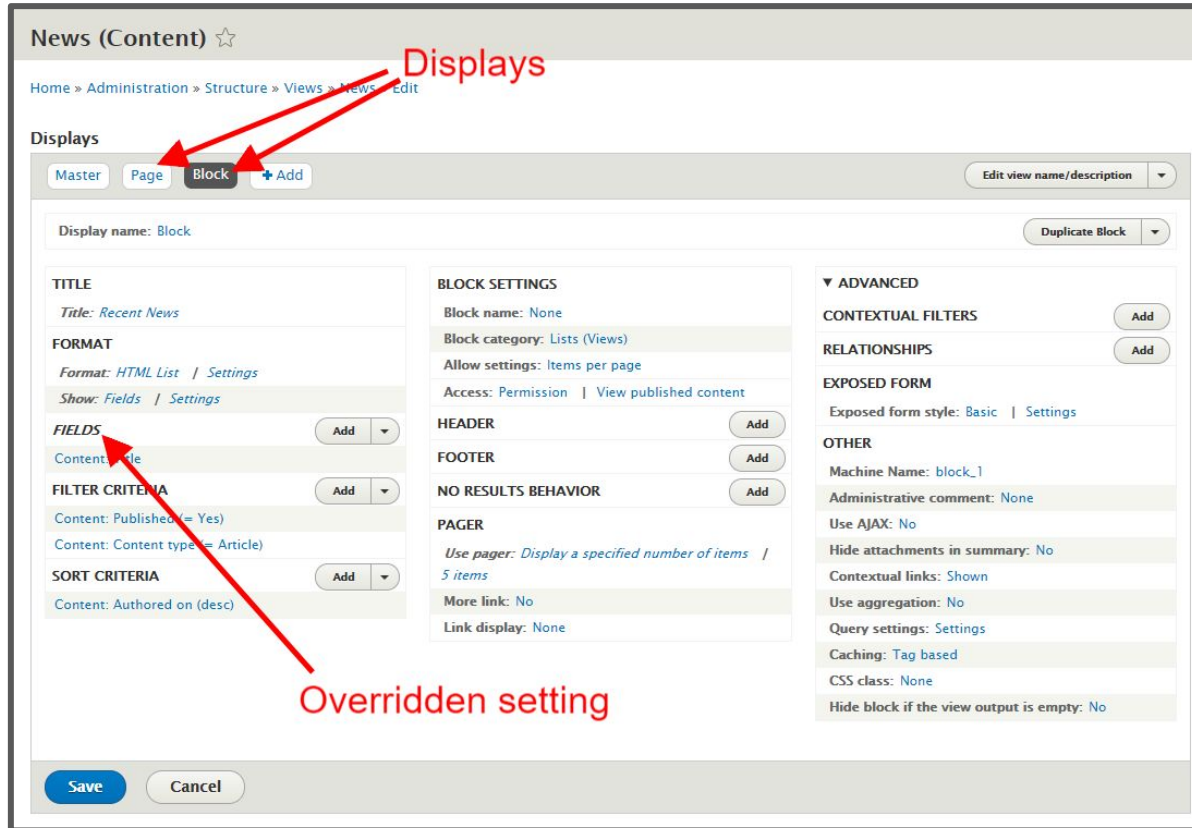
Inheritance Application



- 6. Views: Use multiple displays
 - Displays inherit all features from the Master display
 - Individual settings may be overridden
 - Example: News lists

Wish List: Entity bundle inheritance

Views: Displays



News (Content) ☆

Home » Administration » Structure » Views » News » Edit

Displays

Master Page **Block** + Add Edit view name/description

Display name: Block Duplicate Block

TITLE <i>Title: Recent News</i>	BLOCK SETTINGS Block name: None Block category: Lists (Views) Allow settings: Items per page Access: Permission View published content	ADVANCED
FORMAT <i>Format: HTML List Settings</i> <i>Show: Fields Settings</i>	HEADER Add	CONTEXTUAL FILTERS Add
FIELDS Add	FOOTER Add	RELATIONSHIPS Add
FILTER CRITERIA Add	NO RESULTS BEHAVIOR Add	EXPOSED FORM Exposed form style: Basic Settings
SORT CRITERIA Add	PAGER <i>Use pager: Display a specified number of items 5 items</i> More link: No Link display: None	OTHER Machine Name: block_1 Administrative comment: None Use AJAX: No Hide attachments in summary: No Contextual links: Shown Use aggregation: No Query settings: Settings Caching: Tag based CSS class: None Hide block if the view output is empty: No

Save Cancel

Displays

Overridden setting

Polymorphism



Entities of different types can be interchanged

- Anything that accepts an Entity will accept a Node, User, Group, etc
- Entity reference fields

Polymorphism Application



7. Wish List: generic Entity Reference field for multiple entity types
 - Example: A membership could reference both individual members (User) or company members (Group)

S.O.L.I.D

- **Single Responsibility** - Do one thing and do it well.
- **Open/Closed** - Open for extension, closed for modification
- **Liskov Substitution** - Substitutable abstractions (polymorphism)
- **Interface Segregation** - Thin-focused interfaces
- **Dependency Inversion** - Abstractions should not depend on details.





SINGLE RESPONSIBILITY PRINCIPLE

Just Because You Can, Doesn't Mean You Should

Single Responsibility Principle



Do one thing and do it well

- Don't cause unnecessary complexity
- Reuse existing components instead of creating new ones

Single Responsibility Application



8. Content type structure: Reuse fields and paragraphs, don't duplicate content types.
9. Don't have too few content types.

Single Responsibility Application



8. Content type structure: Reuse fields and paragraphs, don't duplicate content types.
9. Don't have too few content types. But you said...

Single Responsibility Application



8. Content type structure: Reuse fields and paragraphs, don't duplicate content types.
9. Don't have too few content types. But you said...
 - Too many == more overhead
 - Too few == more complexity

Open/Closed Principle



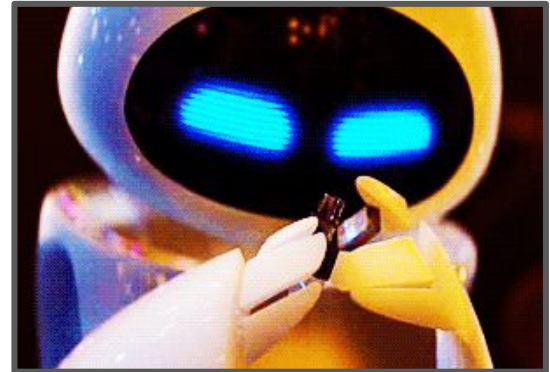
Open for extension, closed for modification

- Support changing requirements (implementation details)
- Define a common interface but leave the details for later

Open/Closed Application

10. Use paragraphs for shared functionality
 - Define components (eg. Media Gallery)
 - Allow those components to be used in flexible ways

11. EVA: Entity Views Attachment
<https://www.drupal.org/project/eva>



Paragraphs + EVA

Gallery (Paragraph) ☆

Home » Administration » Structure » Views » Gallery » Edit

✓ The view *Gallery* has been saved.

Displays

Master EVA + Add Edit view name/description

Display name: EVA Duplicate EVA

TITLE

Title: None

FORMAT

Format: Unformatted list | Settings

Show: Fields | Settings

FIELDS

Paragraph: Gallery Title Add

Paragraph: Slides Add

FILTER CRITERIA

Paragraph: Published (= True) Add

Paragraph: Paragraph type (= Media Gallery) Add

SORT CRITERIA

Add

ENTITY CONTENT SETTINGS

Entity type: Paragraph

Bundles: Media Gallery

Arguments: id

Show title: No

Access: None

HEADER Add

FOOTER Add

NO RESULTS BEHAVIOR Add

PAGER

Use pager: Mini | Mini pager, 10 items

More link: No

Link display: None

ADVANCED

CONTEXTUAL FILTERS

Paragraph: ID Add

RELATIONSHIPS

Add

EXPOSED FORM

Exposed form style: Basic | Settings

OTHER

Machine Name: entity_view_1

Administrative comment: None

Use AJAX: No

Contextual links: Shown

Use aggregation: No

Query settings: Settings

Caching: Tag based

CSS class: None

Save Cancel

Attach to Gallery Paragraph

Paragraphs + EVA

Manage display ☆

Edit Manage fields Manage form display **Manage display**

Default **Preview** Token

Home » Administration » Structure » Paragraphs types » Media gallery [Show row weights](#)

FIELD	LABEL	FORMAT
Content		
⊕ EVA: Gallery - EVA		
Disabled		
⊕ Slides	Above ▼	Rendered entity ▼
⊕ Gallery Title	Above ▼	Plain text ▼

Liskov Substitution Principle



Substitutable abstractions

```
return new RedirectResponse(\Drupal::url('polymorphism'));
```

Interface Segregation Principle



Thin-focused interfaces

- Don't force users to depend on features they won't use
- Simplicity (KISS)
 - Don't show options that aren't needed

Interface Segregation Application



12. Use required fields sparingly
13. Include permissions when planning
 - Hide fields and options that a user doesn't need
14. Recap: Make use of form modes and field permissions

Dependency Inversion Principle

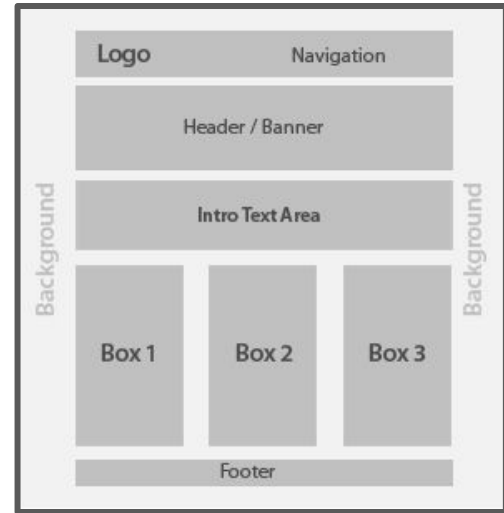


*Abstractions should not depend on details.
Details should depend on Abstractions.*

- Provide flexibility for shifting requirements
- Dependency Injection (Services)

Dependency Inversion Application

15. Define consistent layouts (abstractions) but allow specific content to change (details)
- Component based design
 - Don't mix layout modules (Core Layouts, Panels, Display Suite, etc)



Resources

- 📖 Best Practice Site Architecture in Drupal 8 (Pantheon)
<https://www.youtube.com/watch?v=4GLkujT3xFU>
- 📖 Drupal Theming and Site Building: 6 Best Practices
<https://evolvingweb.ca/blog/drupal-theming-and-site-building-6-best-practices>

Slides available at: <https://goo.gl/n2STuK>