

# Fields and Entities: the good, the bad and the ugly

BADCamp 2012

Francesco Placella // plach

Wolfgang Ziegler // fago

# Wolfgang Ziegler // fago

- From Vienna, Austria
- Studied at the TU Vienna
- CTO of drunomics
- Drupal since 2005

[wolfgangziegler.net](http://wolfgangziegler.net)

[twitter.com/the\\_real\\_fago](https://twitter.com/the_real_fago)

[gplus.to/fago](https://plus.google.com/u/0/fago)

# Francesco Placella // plach

- From Venice, Italy
- Studied at Ca' Foscari University
- Owner at PSEGN0
- Drupal since 2006

[twitter.com/plach\\_\\_](https://twitter.com/plach__)

# The Rules ...

- Rule 1:

Speak *loud* and speak *slow*

# The Rules ...

- Rule 1:

Speak *loud* and speak *slow*

- Rule 2:

No urban dictionary

# Outline

- D7: Field API overview
- D7: Entity API overview
- D7: Best practices
- What will Drupal 8 bring us?

# Anatomy of Field data

```
$entity->{$field_name}  
[ $langcode ][ $delta ][ $column ]
```

# Field API

- Main Field API components
  - Info (fields + instances)
  - CRUD
  - Attach
  - Data structure accessors



# Field language

- Field translatability

```
$node->body['en'][0]['value']
```

```
$node->body['it'][0]['value']
```

```
$node->body['de'][0]['value']
```

```
$node->field_tags[LANGUAGE_NONE][0]  
['tid']
```

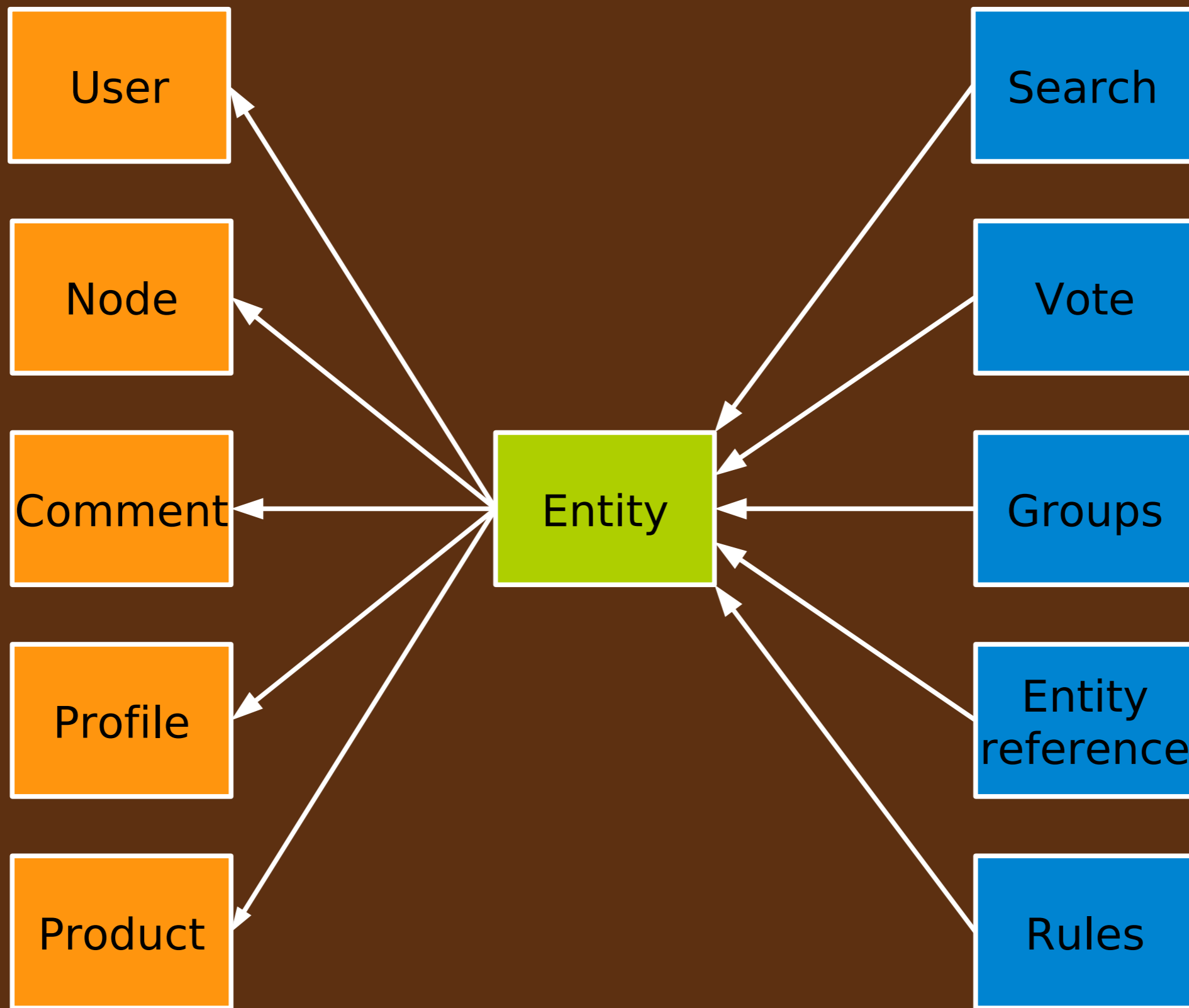
- Field attach behavior
  - all languages for storage
  - single language for forms and view

# Anatomy of Field data

- Good consistency
- Bad DX
- Attached to any object (entity?)

# Entity API – What for?

Unified way to access data.

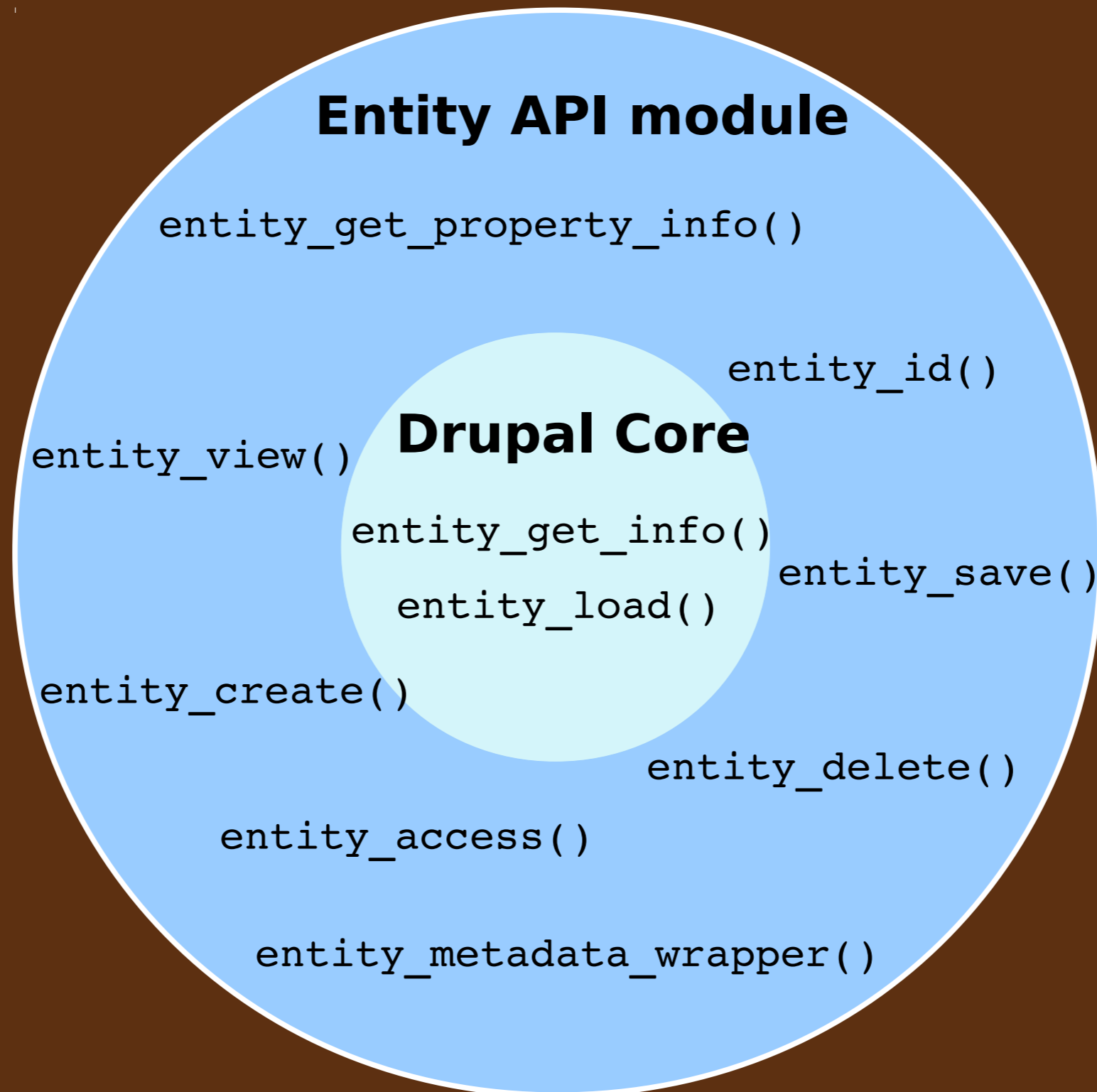


# Entity API module

Assists you with

- interacting with entities
- and providing new entity types

# Entity API Functions



# Best practices

# Accessing field values

```
$node->body['und'][0]['value']
```

Seriously?

- Use `LANGUAGE_NONE`
- Don't use it!
- Try and write language-agnostic code if possible
- Unless you need language-aware code



# Core patterns

- Preprocess functions
  - `field_attach_preprocess()`
  - use field variables directly
- Rendering/display context

```
$items = field_get_items('node', $node,  
'body');
```

```
$body_value = $items[0]['value'];
```

# Core patterns

Any other context:

- Act on a single language
- Detect field translatability
- Use `LANGUAGE_NONE` or `entity_language()`
- Act on every language

Basically you are screwed :(

# Core Field DX resources

- Introduce additional helper functions
  - <http://drupal.org/node/1260640>
  - <http://drupal.org/project/fieldx>

# Different scenarios

- Contrib code needs to support multilingual scenarios
- Every field might become translatable sooner or later
- Test it with Entity Translation
- Custom code can happily ignore field language or better
- Use Entity API!

**Enter the Entity API module**

# Metadata Wrapper

```
→ $wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
→ $mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
→ $wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```



# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
→ $text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
→ $wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
→ $terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
→ $wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
→ $options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
→ $label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Metadata Wrapper

```
$wrapper = entity_metadata_wrapper('node', $nid);  
$mail = $wrapper->author->mail->value();  
$wrapper->author->mail->set('fago@example.com');  
$text = $wrapper->field_text->value();  
$wrapper->language('de')->field_text->value();  
$terms = $wrapper->field_tags->value();  
$wrapper->field_tags[] = $term;  
$options = $wrapper->field_tags->optionsList();  
$label = $wrapper->field_tags[0]->label();  
$access = $wrapper->field_tags->access('edit');
```

# Providing an entity type

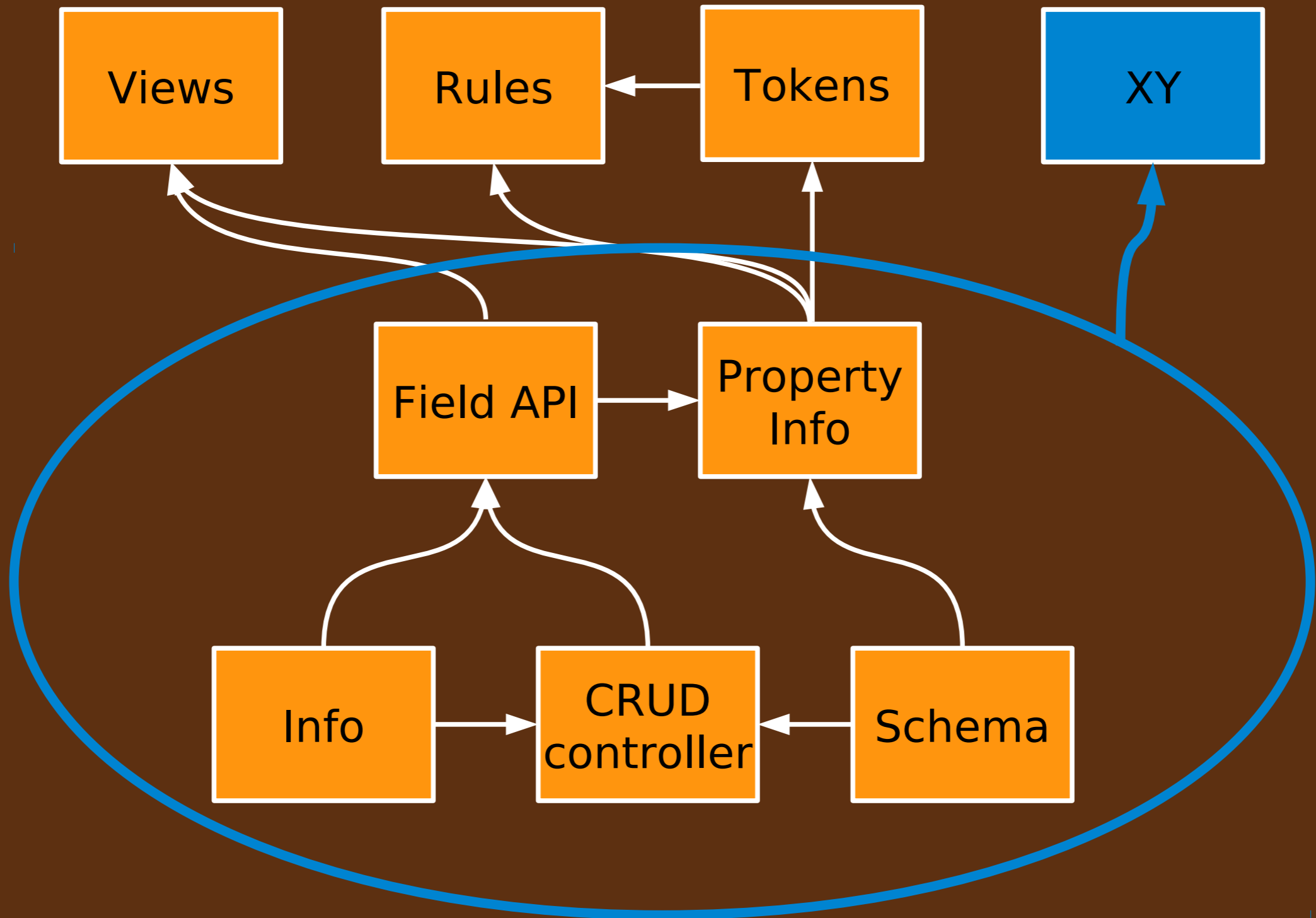
- Implement `hook_entity_info()`
- Specify your 'controller class'
- Implement `hook_schema()`



# Entity API module

```
$profile = entity_create('profile2', array(
    'type' => 'main',
    'user' => $account,
));
$profile->save();
$profile->delete();
entity_delete_multiple('profile2', array(1, 2, 3));
```

# Integrating your entity type



# Drupal 8

## (defeating the ugliness)

# Classed objects...!

```
$comment = entity_create('comment', array(
  'nid' => $node->nid,
));
$comment->save();
echo $comment->id();
$comment->delete();
entity_delete_multiple('comment', array(1, 2, 3));
```

# Comments in Drupal 8

```
class CommentStorageController extends
  DatabaseStorageController {
  ...
}
class Comment extends Entity {
  ...
}
class Entity implements EntityInterface {
  ...
}
```

# Entity Field API

- Similar to the entity metadata wrappers, but improved and built-in!
- Unified API for properties and fields
  - Properties → (Entity) Fields
  - Field API fields → (Configurable) Fields
- Property definitions / metadata
- Improved DX!

# Examples

```
$node->body[0]->value;
```

```
$node->body->value
```

```
$comment->nid->value = 1;
```

```
$node = $comment->nid->entity;
```

```
$translation = $entity->getTranslation('de');
```

```
$old_value = $translation->body->value;
```

```
$translation->body->value = "New value";
```

# More controllers... !!!

Controllers + default implementations:

- Storage (Default: DB) - required
- Form
- Render
- Access (soon)
- Translation
- List



# Configuration entities

- Similar idea as D7 exportable entities
- Another storage controller → CMI
- Leverages existing API for CRUD
  - Caching
  - Forms
  - Translation?
  - List

# Storage + Query

- EFQ v2 → Entity Query
  - Storage independent
  - Improved condition interface
    - Unified interface (similar to DBTNG)
    - AND/OR condition groups
  - Multilingual support
- Unified storage → Entity Storage: WIP

# REST + Entity serialization

- UUIDs (done)
- Entity serialization (WIP)
  - Based upon the new Entity Field API
  - JSON-LD by default
- RESTful web service in core (WIP)
- More infos:  
<http://groups.drupal.org/node/237443>

# Entity Translation

- D7: two translation approaches
  - Core: node translation
  - Contrib: entity translation
- Why not content translation?
  - Only nodes
  - Problem with references
  - Sharing data across translations

# Entity Translation UI

- One module to rule them all
  - Nodes
  - Comments (seriously?)
  - Taxnomy terms
  - User account data
- Per-field translatability
- Translations overview

**Questions?**  
**(remember the rules!)**

**Thank you!**