Deeson.

DrupalCon Dublin.

#futureofwork

Deeson.

Entity validation: The kick-ass road to data integrity.

kristiaanvandeneynde - @Magentix

Coding & Development - Intermediate

#futureofwork

Kristiaan Van den Eynde.

- Drupal developer
- Work at Deeson, UK
- Live near Antwerp, BE
- Happily married
- HSP: Highly Sensitive Person

Deeson.

What is entity validation?

```
EntityMetadataWrapperException: Unknown data property next_slide in
EntityStructureWrapper->getPropertyInfo() (line 339 of /var/www/dublin/sites/all/modules/
entity/includes/entity.wrapper.inc).
```

Deeson.

Entity validation.

- Validates content entities on multiple levels

Entity validation.

- Validates content entities on multiple levels
- Returns a list of violations when validation fails

Deeson.

Entity validation.

- Validates content entities on multiple levels
- Returns a list of violations when validation fails
- Happens automatically in entity form validation

Entity validation.

- Validates content entities on multiple levels
- Returns a list of violations when validation fails
- Happens automatically in entity form validation
- Is not a part of the form system (unlike D7)

Deeson.

Entity validation.

- Validates content entities on multiple levels
- Returns a list of violations when validation fails
- Happens automatically in entity form validation
- Is not a part of the form system (unlike D7)
- Guarantees the data integrity of your entities

Deeson.

When to use entity validation?

On entity manipulation.

- REST
- Custom entity generation code
- Migrations
- ...

How to invoke entity validation?

Simple!

```
$entity->validate();
```

Deeson.

Simple!

```
$entity->get(‘foo’)->validate();
```

Deeson.

Simple!

```
$entity->get(‘foo’)->get(0)->validate();
```

Deeson.

Well... actually

```
$violations = $entity->validate();
```

Deeson.

Working with violations.

Checking for violations.

```
$violations->count();
extends \IteratorAggregate
```

More funky methods.

```
$violations->getEntityViolations();
$violations->getByField(‘foo’);

\Drupal\Core\Entity\
EntityConstraintViolationListInterface
```

Reading a single violation.

```
$violation->getMessage();
```

More groovy methods.

```
$violation->getPropertyPath();
$violation->getInvalidValue();

Symfony\Component\Validator\
ConstraintViolationInterface
```

Where do violations come from?

Four layers of validation.

- The entity as a whole

Four layers of validation.

- The entity as a whole
- A field on an entity

Four layers of validation.

- The entity as a whole
- A field on an entity
- A field entry on an entity

Four layers of validation.

- The entity as a whole
- A field on an entity
- A field entry on an entity
- A property of a field entry

Deeson.

Obligatory car analogy.

- `$car`
- `$car->field_wheels`
- `$car->field_wheels[0]`
- `$car->field_wheels[0][‘tyre_size’]`

Deeson.

In Drupal-speak.

- ContentEntityInterface (FieldableEntityInterface)
- FieldItemListInterface
- FieldItemInterface
- TypedDataInterface

Deeson.

The 4 levels in detail.

The entity as a whole.

- Validation across multiple fields

The entity as a whole.

- Validation across multiple fields

  Fuel tank + electric engine

Deeson.

The entity as a whole.

- Validation across multiple fields

  Fuel tank + electric engine

- Field-independent validation

The entity as a whole.

- Validation across multiple fields

  Fuel tank + electric engine

- Field-independent validation

  Third car when flat broke

Deeson.

A field on an entity.

- Validation across multiple field entries

A field on an entity.

- Validation across multiple fields

  2 monster truck wheels + 2 minibike wheels

Deeson.

A field entry on an entity.

- Validation across multiple field properties

A field entry on an entity.

- Validation across multiple field properties

  16 inch rim + 15 inch tyre

Deeson.

A property of a field entry.

- Validation of single properties

A property of a field entry.

- Validation of single properties

  A tyre made out of wood

Defining your own validation.

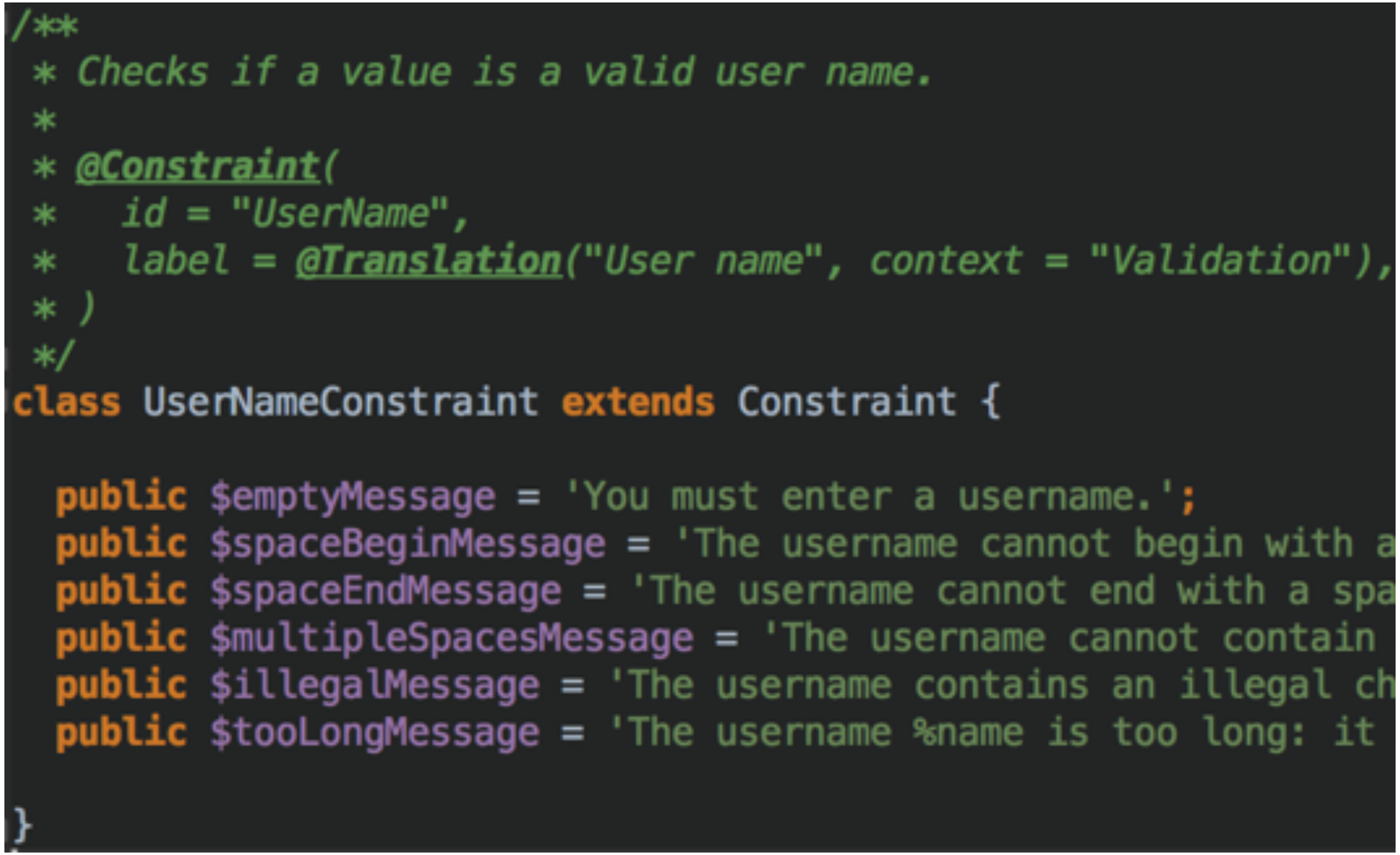
Two classes required.

- A Constraint plugin
- The actual validator

The constraint plugin.

Constraint plugin definition.

- A unique ID
- A human readable label
- A type (string) or list of types (array)
- In mymodule/src/Plugin/Validation/Constraint

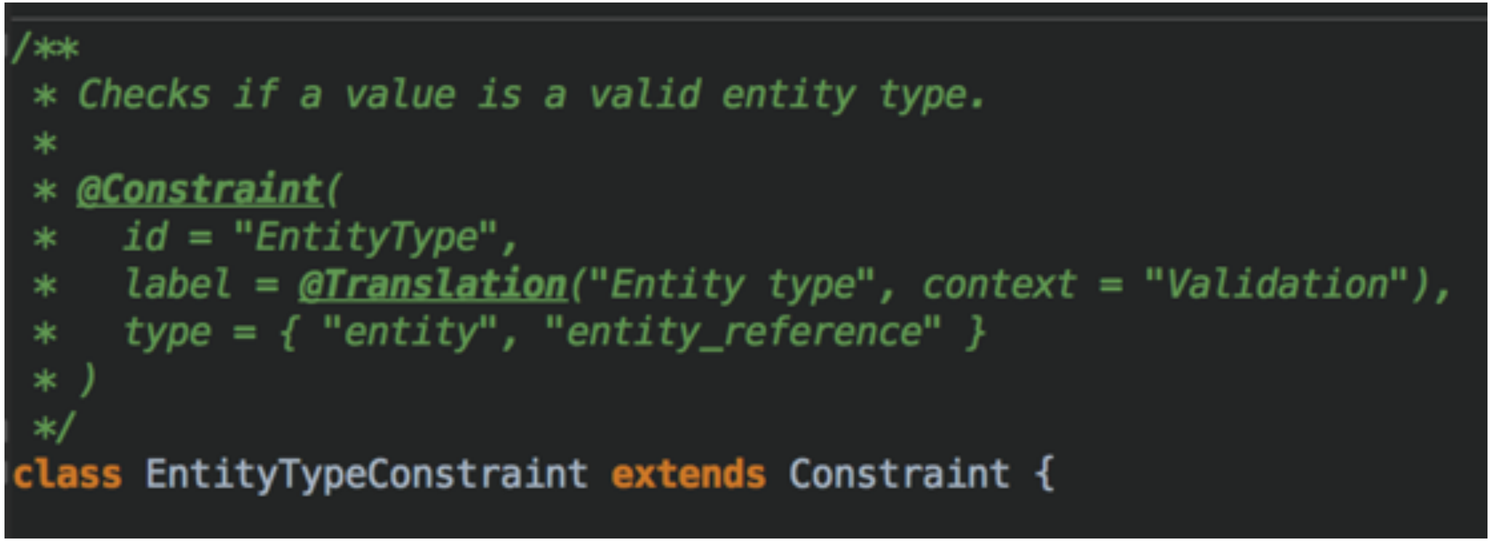


```
/**
 * Checks if a value is a valid user name.
 *
 * @Constraint(
 *   id = "UserName",
 *   label = @Translation("User name", context = "Validation"),
 * )
 */
class UserNameConstraint extends Constraint {

  public $emptyMessage = 'You must enter a username.';
  public $spaceBeginMessage = 'The username cannot begin with a
  public $spaceEndMessage = 'The username cannot end with a spa
  public $multipleSpacesMessage = 'The username cannot contain
  public $illegalMessage = 'The username contains an illegal ch
  public $tooLongMessage = 'The username %name is too long: it

}
```

What about the type?

- Nothing: Empty array (the default)
- **Everything: false**
- Entities: entity, entity:node
- FieldItem: field_item, field_item:entity_reference
- FieldItemList: not supported?
- Other TypedData: string, integer, …

Setting the type key.



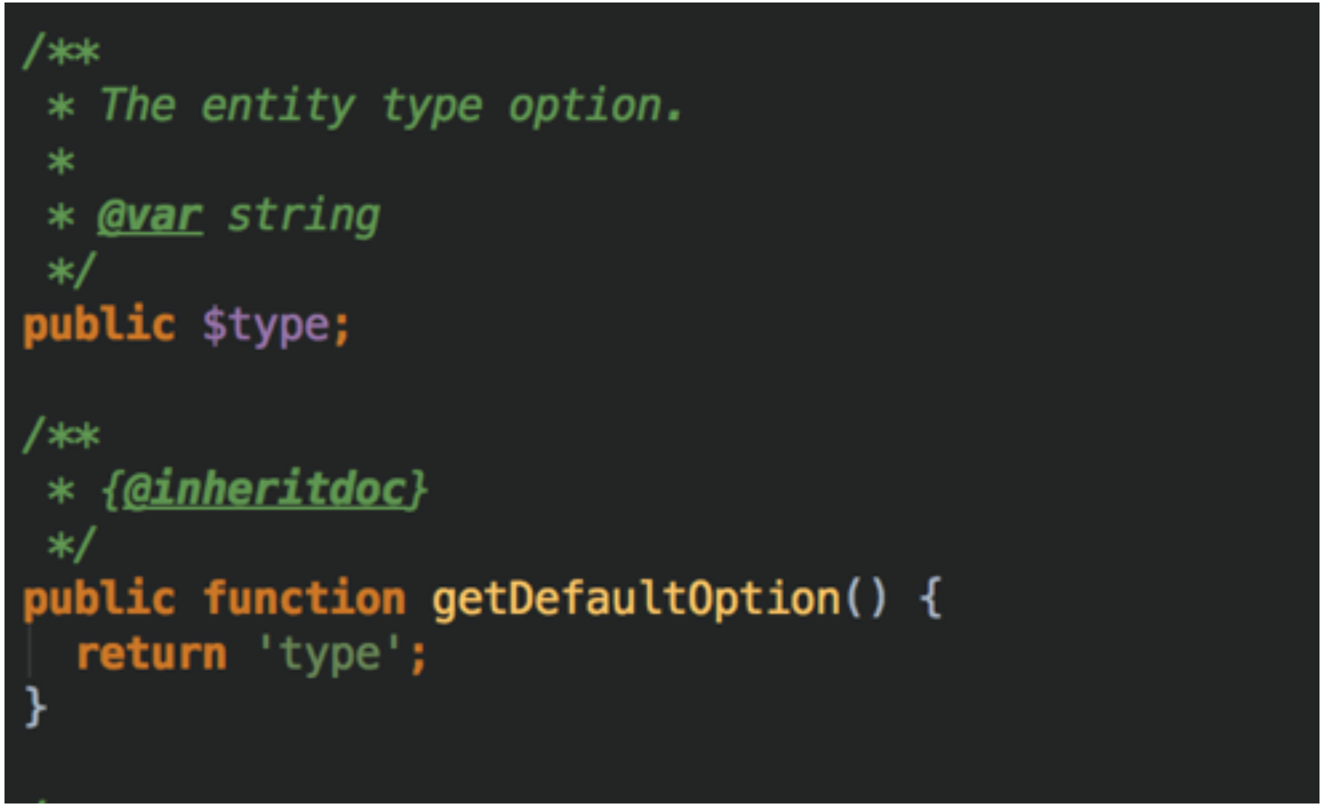
```
/**
 * Checks if a value is a valid entity type.
 *
 * @Constraint(
 *   id = "EntityType",
 *   label = @Translation("Entity type", context = "Validation"),
 *   type = { "entity", "entity_reference" }
 * )
 */
class EntityTypeConstraint extends Constraint {
```

Deeson.

Constraints can take options.

- Completely optional
- Simply define them as object properties
- Can accept any number of options as an array
- How to pass the options is covered later

Accepting options



```
/**
 * The entity type option.
 *
 * @var string
 */
public $type;

/**
 * {@inheritdoc}
 */
public function getDefaultOption() {
  return 'type';
}
```

Deeson.

The validator class.

Magic class name.

- Default: Name of constraint class + Validator
- Can change the default in

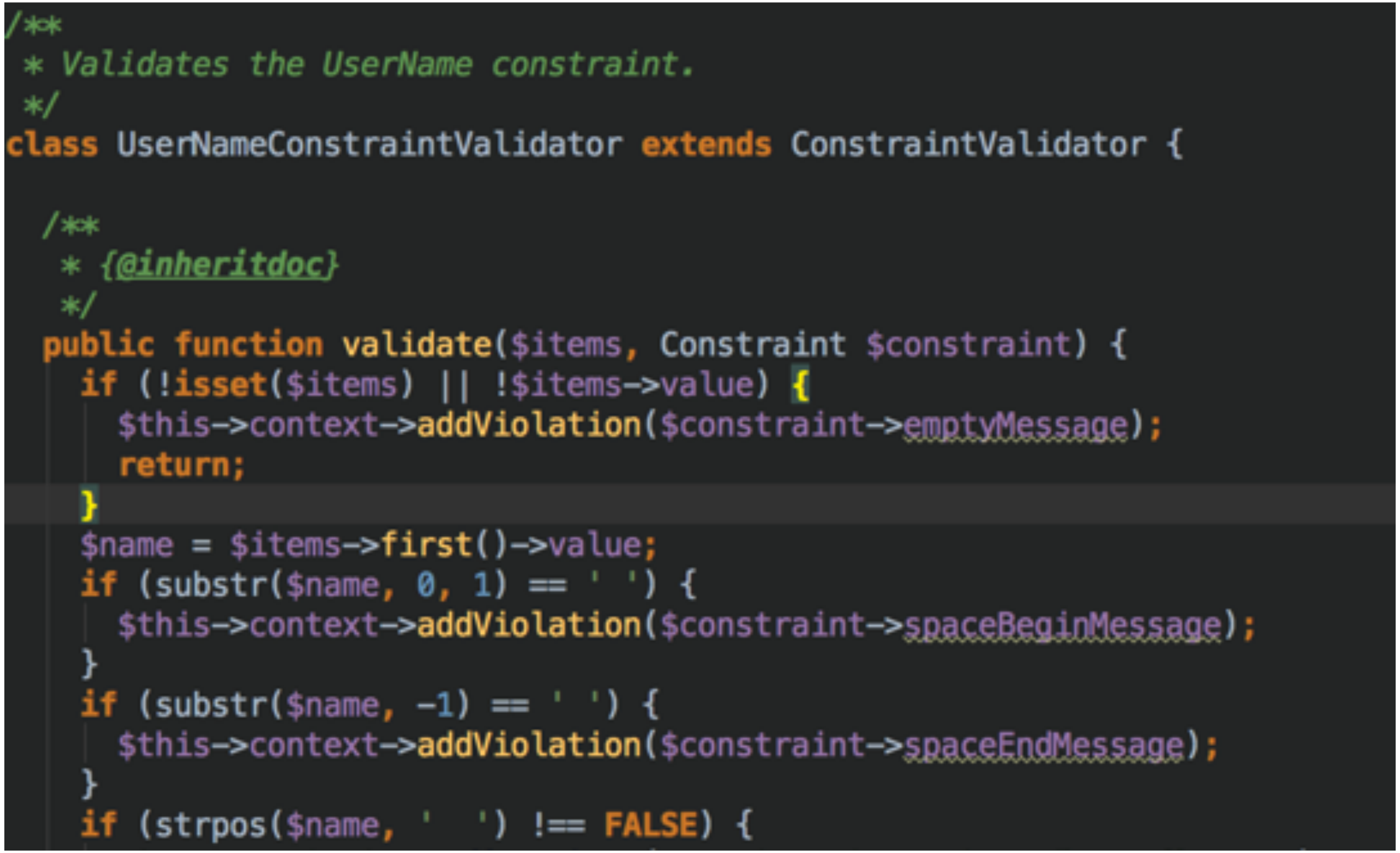
  `Constraint::validatedBy()`

Deeson.

The value it receives.

- For entities: The entity
- For fields: The FieldItemList
- For field entries: The FieldItem
- For properties: The raw value

Deeson.



```php
/**
 * Validates the UserName constraint.
 */
class UserNameConstraintValidator extends ConstraintValidator {

  /**
   * {@inheritdoc}
   */
  public function validate($items, Constraint $constraint) {
    if (!isset($items) || !$items->value) {
      $this->context->addViolation($constraint->emptyMessage);
      return;
    }
    $name = $items->first()->value;
    if (substr($name, 0, 1) == ' ') {
      $this->context->addViolation($constraint->spaceBeginMessage);
    }
    if (substr($name, -1) == ' ') {
      $this->context->addViolation($constraint->spaceEndMessage);
    }
    if (strpos($name, '  ') !== FALSE) {
```

The property path.

Mentioning for completeness.

- Tells you where the error occurred
- Defaults to the item being validated
- Can be more specific in your validator
- Example: ValidReferenceConstraintValidator

Examples.

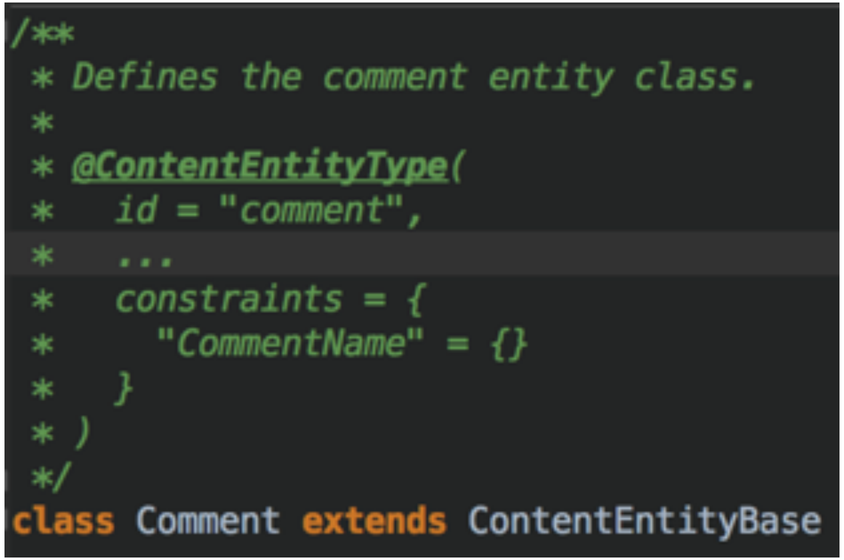
When validating an entity:	`field_foo.0.value` `field_bar.2`
When validating a field:	`3.property_bar`
When validating a field entry:	`value`

Deeson.

How to apply validators.

The entity level.

- As part of the annotation

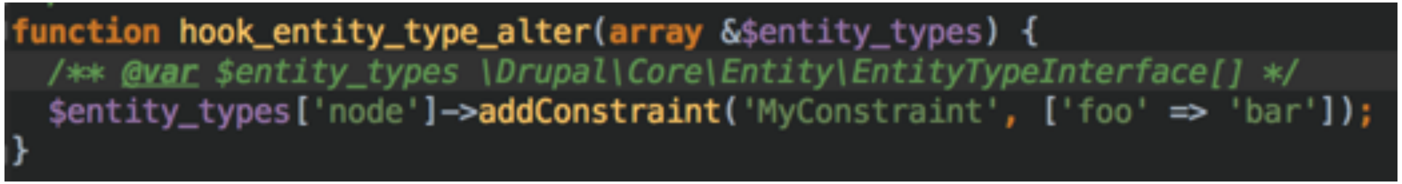


```
/**
 * Defines the comment entity class.
 *
 * @ContentEntityType(
 *   id = "comment",
 *   ...
 *   constraints = {
 *     "CommentName" = {}
 *   }
 * )
 */
class Comment extends ContentEntityBase
```

Deeson.

The entity level.

- As part of the annotation
- `hook_entity_type_alter()`

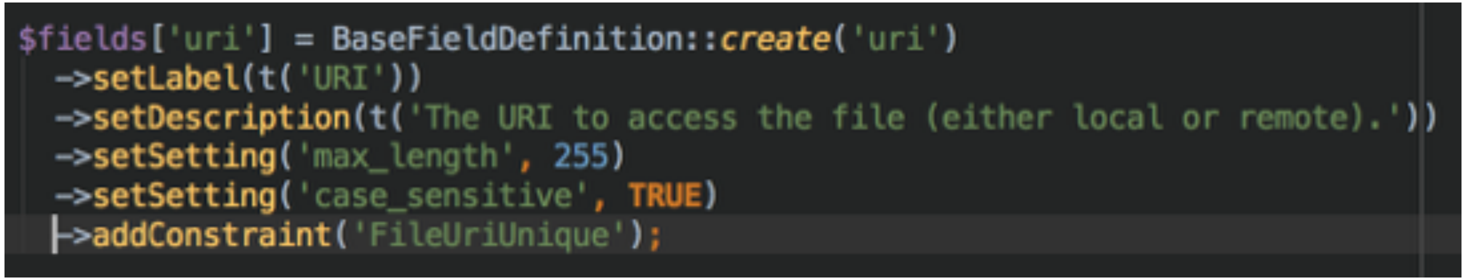


```php
function hook_entity_type_alter(array &$entity_types) {
  /** @var $entity_types \Drupal\Core\Entity\EntityTypeInterface[] */
  $entity_types['node']->addConstraint('MyConstraint', ['foo' => 'bar']);
}
```

Deeson.

The field level.

- BaseFieldDefinition::addConstraint()
- hook_entity_base_field_info_alter()
- hook_entity_bundle_field_info_alter()

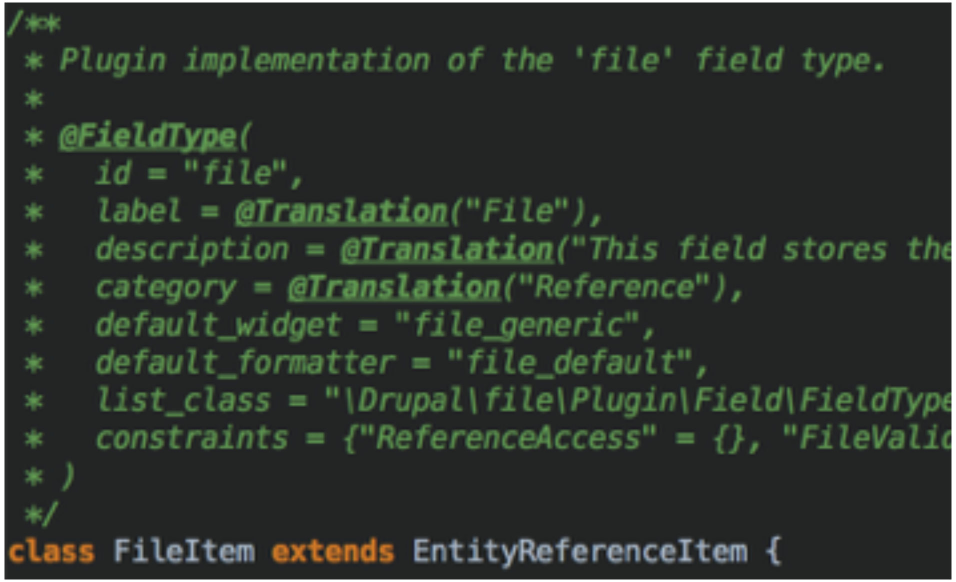


```
$fields['uri'] = BaseFieldDefinition::create('uri')
  ->setLabel(t('URI'))
  ->setDescription(t('The URI to access the file (either local or remote).'))
  ->setSetting('max_length', 255)
  ->setSetting('case_sensitive', TRUE)
  ->addConstraint('FileUriUnique');
```

Deeson.

The field entry level.

- As part of the annotation
- `hook_field_info_alter()`

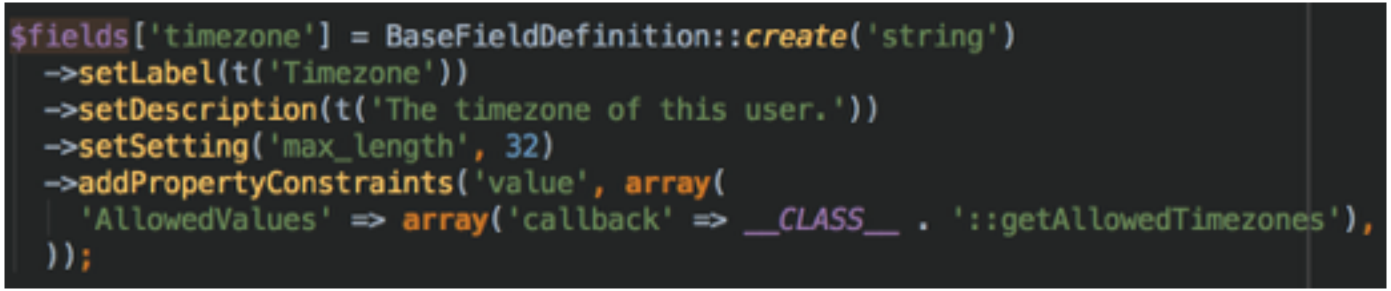


```
/**
 * Plugin implementation of the 'file' field type.
 *
 * @FieldType(
 *   id = "file",
 *   label = @Translation("File"),
 *   description = @Translation("This field stores the
 *   category = @Translation("Reference"),
 *   default_widget = "file_generic",
 *   default_formatter = "file_default",
 *   list_class = "\Drupal\file\Plugin\Field\FieldType
 *   constraints = {"ReferenceAccess" = {}, "FileValic
 * )
 */
class FileItem extends EntityReferenceItem {
```

Deeson.

The field property level.

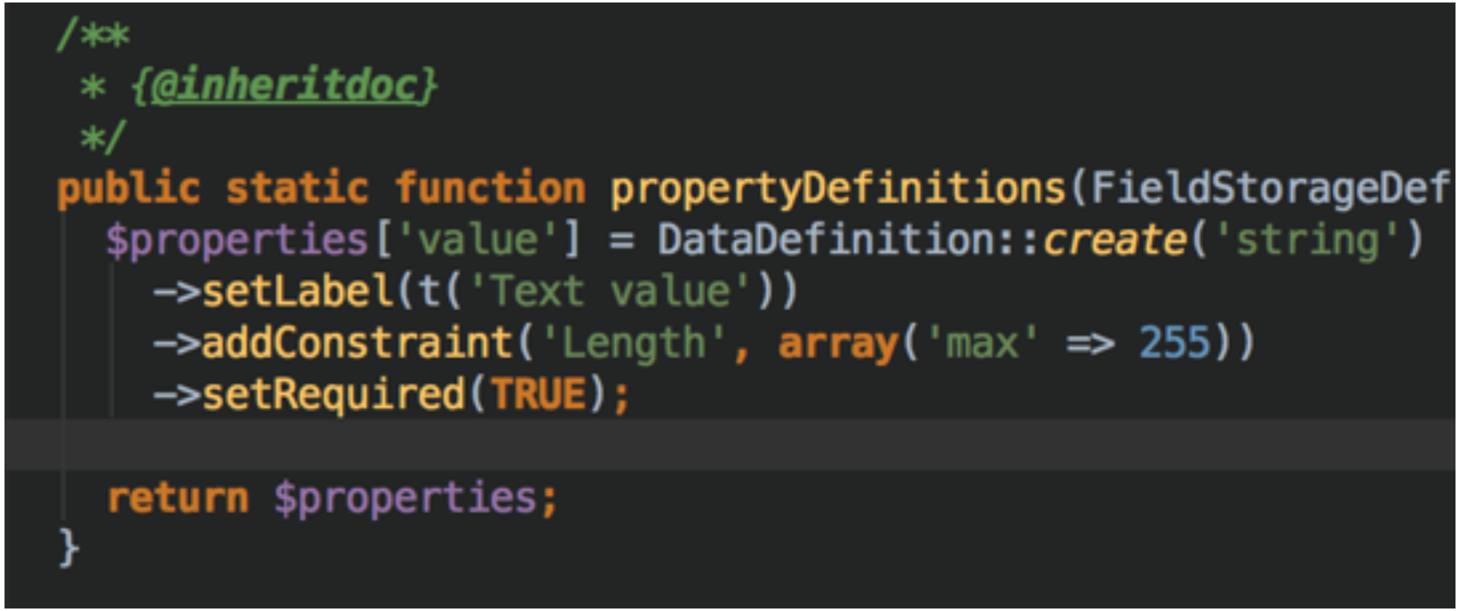
- BaseFieldDefinition::addPropertyConstraints()
- hook_entity_base_field_info_alter()
- hook_entity_bundle_field_info_alter()



```
$fields['timezone'] = BaseFieldDefinition::create('string')
  ->setLabel(t('Timezone'))
  ->setDescription(t('The timezone of this user.'))
  ->setSetting('max_length', 32)
  ->addPropertyConstraints('value', array(
    'AllowedValues' => array('callback' => __CLASS__ . '::getAllowedTimezones'),
  ));
```

Deeson.

The field property level.



```
/**
 * {@inheritdoc}
 */
public static function propertyDefinitions(FieldStorageDef
  $properties['value'] = DataDefinition::create('string')
    ->setLabel(t('Text value'))
    ->addConstraint('Length', array('max' => 255))
    ->setRequired(TRUE);

  return $properties;
}
```

Deeson.

That’s it!

Join us for contributions sprints...

First time sprinter workshop

9:00-12:00
Room Wicklow 2A

Mentored Core Sprint

9:00-18:00
Wicklow Hall 2B

General Sprints

9:00-18:00
Wicklow Hall 2A

Questions?

What did you think?

Evaluate this session:

events.drupal.org/dublin2016/schedule

Deeson.

Thank you!

Deeson.

Deeson.

We're hiring!

Find our more visit deeson.co.uk/careers

or come speak to me after the presentation