DrupalCon

SEATTLE **2019**

APRIL 8-12

DrupalCon
SEATTLE 2019
APRIL 8-12

New Wave Module Development

Kris “EclipseGc” Vanderwater

Kris Vanderwater

Sr. Software Engineer @ Acquia

@EclipseGc

- Drupal Development 13+ years
- CTools Co-Maintainer
- Multi-year focus on Page Layout
- Drupal 8 Contributor
- Co-Author: Drupal 8 Plugin System
- Major contributor to Layout Builder
- Technical Architect ContentHub 2.0

Topics

Topics

- Drupal 7 & Earlier Development

Topics

- Drupal 7 & Earlier Development
- Fossils of the Fore-bearers

Topics

- Drupal 7 & Earlier Development
- Fossils of the Fore-bearers
- Upgrades and Replacements

Topics

- Drupal 7 & Earlier Development
- Fossils of the Fore-bearers
- Upgrades and Replacements
- Practical Application

DrupalCon
SEATTLE 2019
APRIL 8-12

Topics

- Drupal 7 & Earlier Development
- Fossils of the Fore-bearers
- Upgrades and Replacements
- Practical Application
- Question/Answer

Drupal 7 & Earlier Development

“It's OK to figure out murder mysteries, but you shouldn't need to figure out code. You should be able to read it.”

Steve McConnell

Drupal Hooks

Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named foo_bar(), where "foo" is the name of the module (whose filename is thus foo.module) and "bar" is the name of the hook.

Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named foo_bar(), where "foo" is the name of the module (whose filename is thus foo.module) and "bar" is the name of the hook.
- Translation: Drupal hooks are magically named functions that “do things”.

Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named foo_bar(), where "foo" is the name of the module (whose filename is thus foo.module) and "bar" is the name of the hook.
- Translation: Drupal hooks are magically named functions that “do things”.
- Each module/theme can only implement a hook once.

Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named foo_bar(), where "foo" is the name of the module (whose filename is thus foo.module) and "bar" is the name of the hook.
- Translation: Drupal hooks are magically named functions that “do things”.
- Each module/theme can only implement a hook once.
- Hooks are procedural functions (No dependency injection)

Drupal Hooks

- Drupal's module system is based on the concept of "hooks". A hook is a PHP function that is named foo_bar(), where "foo" is the name of the module (whose filename is thus foo.module) and "bar" is the name of the hook.
- Translation: Drupal hooks are magically named functions that “do things”.
- Each module/theme can only implement a hook once.
- Hooks are procedural functions (No dependency injection)
- drupal_static()…

Drupal Hooks

```
function foo_bar($arg1, $arg2) {
  if ($arg1 === 'foo') {
    // do stuff.
  }
}
```

Drupal Hooks

```
function foo_bar($arg1, $arg2) {
  switch($arg1) {
    case 'foo':
      // do stuff
      break;
    case 'bar':
      // do other stuff
      break;
  }
}
```

Drupal Hooks

```
function foo_bar($arg1, $arg2) {
  if ($arg1 && $arg1 === 'foo' && $arg2 === 'bar') {
    // do stuff
  }
  if ($arg2 === 'baz') {
    // do other stuff
  }
  elseif ($arg2 === 'foobaz') {
    // some other stuff
  }
}
```

Drupal Hooks

```php
function foo_bar($arg1, $arg2) {
  $foo = &drupal_static(__FUNCTION__, []);
  if (empty($foo)) {
    // Do something expensive and store it in $foo
  }
  return $foo;
}
```

Drupal Hooks (ALL TOGETHER NOW!)

```php
function foo_bar($arg1, $arg2) {
  $foo = &drupal_static(__FUNCTION__, []);
  if (empty($foo)) {
    // Get a service!
    /** @var \Drupal\foo\MyServiceInterface $service */
    $service = \Drupal::service('get.some.service');
    // Do something expensive and store it in $foo
  }
  return $foo;
}
```

Drupal Hooks

Drupal Hooks

- Hooks started life manipulating strings (HTML output).

Drupal Hooks

- Hooks started life manipulating strings (HTML output).
- Alter hooks are a completely separate thing that pass references.

Drupal Hooks

- Hooks started life manipulating strings (HTML output).
- Alter hooks are a completely separate thing that pass references.
- Alter hooks only support a limited number of references at at time.

Fossils of the Fore-bearers

“Even the best planning is not so omniscient as to get it right the first time.”

Fred Brooks

DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub

Acquia ContentHub

- Content Syndication Service

Acquia ContentHub

- Content Syndication Service
- 8.x-1.x really a straight port from Drupal 7

Acquia ContentHub

- Content Syndication Service
- 8.x-1.x really a straight port from Drupal 7
- Guilty of many of the criticism outlined

Acquia ContentHub

- Content Syndication Service
- 8.x-1.x really a straight port from Drupal 7
- Guilty of many of the criticism outlined
- Looks like a stereotypical Drupal module

Acquia ContentHub

- Content Syndication Service
- 8.x-1.x really a straight port from Drupal 7
- Guilty of many of the criticism outlined
- Looks like a stereotypical Drupal module
- Difficult to maintain

DrupalCon
SEATTLE 2019
APRIL 8-12

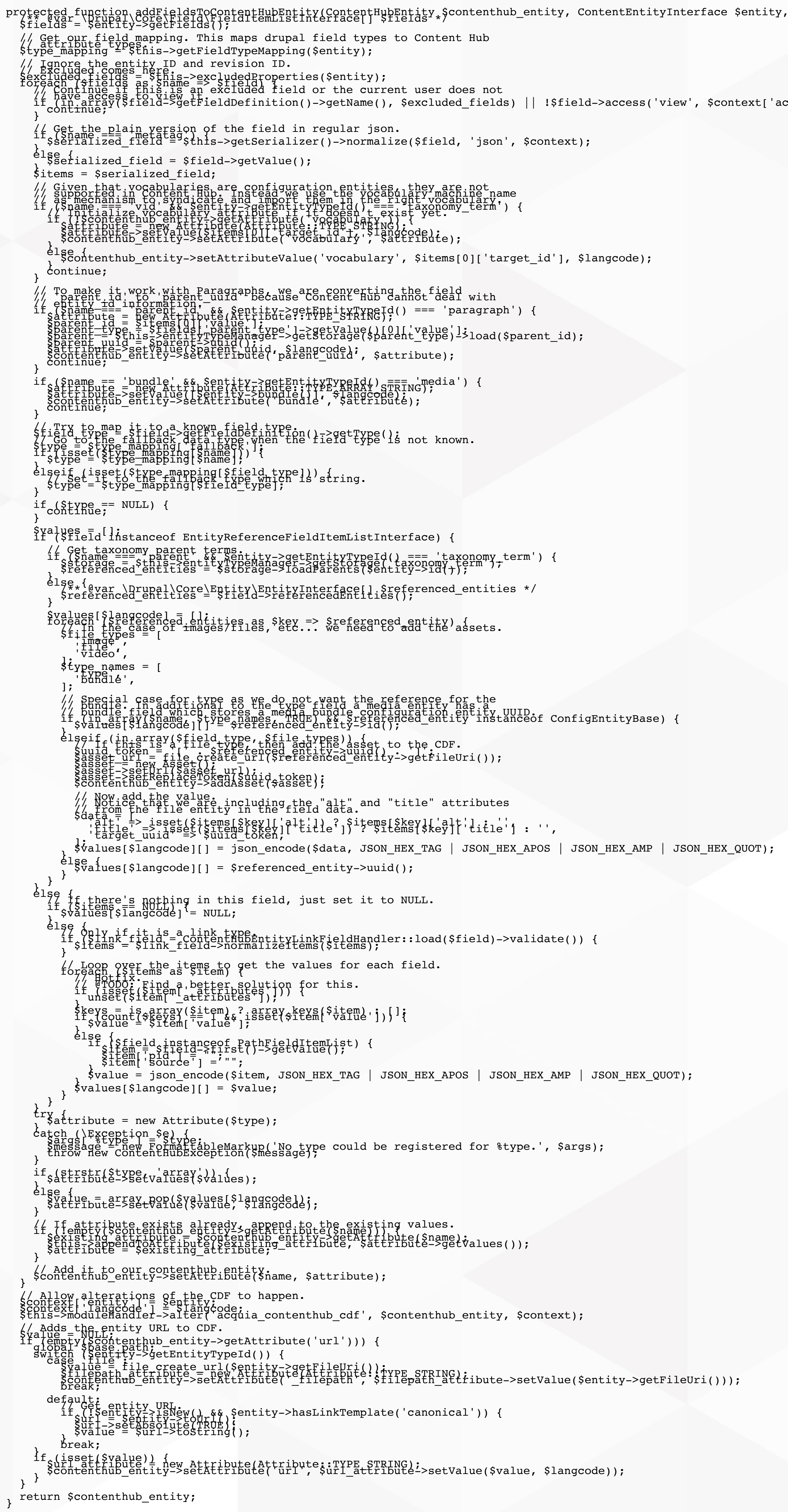
Acquia ContentHub

```php
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
  /** @var \Drupal\Core\Field\FieldItemListInterface[] $fields */
  $fields = $entity->getFields();

  // Get our field mapping. This maps drupal field types to Content Hub
  // attribute types.
  $type_mapping = $this->getFieldTypeMapping($entity);

  // Ignore the entity ID and revision ID.
  // Excluded comes here.
  $excluded_fields = $this->excludedProperties($entity);
  foreach ($fields as $name => $field) {
    // Continue if this is an excluded field or the current user does not
    // have access to view it.
    if (in_array($field->getFieldDefinition()->getName(), $excluded_fields) || !$field->access('view', $context['account'])) {
      continue;
    }

    // Get the plain version of the field in regular json.
    if ($name === 'metatag') {
      $serialized_field = $this->getSerializer()->normalize($field, 'json', $context);
    }
    else {
      $serialized_field = $field->getValue();
    }
    $items = $serialized_field;

    // Given that vocabularies are configuration entities, they are not
    // supported in Content Hub. Instead we use the vocabulary machine name
    // as mechanism to syndicate and import them in the right vocabulary.
    if ($name === 'vid' && $entity->getEntityTypeId() === 'taxonomy_term') {
      // Initialize vocabulary attribute if it doesn't exist yet.
      if (!$contenthub_entity->getAttribute('vocabulary')) {
        $attribute = new Attribute(Attribute::TYPE_STRING);
        $attribute->setValue($items[0]['target_id'], $langcode);
        $contenthub_entity->setAttribute('vocabulary', $attribute);
      }
      else {
        $contenthub_entity->setAttributeValue('vocabulary', $items[0]['target_id'], $langcode);
      }
      continue;
    }

    // To make it work with Paragraphs, we are converting the field
    // 'parent_id' to 'parent_uuid' because Content Hub cannot deal with
    // entity_id information.
    if ($name === 'parent_id' && $entity->getEntityTypeId() === 'paragraph') {
      $attribute = new Attribute(Attribute::TYPE_STRING);
      $parent_id = $items[0]['value'];
      $parent_type = $fields['parent_type']->getValue()[0]['value'];
      $parent = $this->entityTypeManager->getStorage($parent_type)->load($parent_id);
      $parent_uuid = $parent->uuid();
      $attribute->setValue($parent_uuid, $langcode);
      $contenthub_entity->setAttribute('parent_uuid', $attribute);
      continue;
    }

    if ($name == 'bundle' && $entity->getEntityTypeId() === 'media') {
      $attribute = new Attribute(Attribute::TYPE_ARRAY_STRING);
      $attribute->setValue([$entity->bundle()], $langcode);
      $contenthub_entity->setAttribute('bundle', $attribute);
      continue;
    }

    // Try to map it to a known field type.
    $field_type = $field->getFieldDefinition()->getType();
    // Go to the fallback data type when the field type is not known.
    $type = $type_mapping['fallback'];
    if (isset($type_mapping[$name])) {
      $type = $type_mapping[$name];
    }
    elseif (isset($type_mapping[$field_type])) {
      // Set it to the fallback type which is string.
      $type = $type_mapping[$field_type];
    }

    if ($type == NULL) {
      continue;
    }

    $values = [];
    if ($field instanceof EntityReferenceFieldItemListInterface) {

      // Get taxonomy parent terms.
      if ($name === 'parent' && $entity->getEntityTypeId() === 'taxonomy_term') {
        $storage = $this->entityTypeManager->getStorage('taxonomy_term');
        $referenced_entities = $storage->loadParents($entity->id());
      }
      else {
        /** @var \Drupal\Core\Entity\EntityInterface[] $referenced_entities */
        $referenced_entities = $field->referencedEntities();
      }

      $values[$langcode] = [];
      foreach ($referenced_entities as $key => $referenced_entity) {
        // In the case of images/files, etc... we need to add the assets.
        $file_types = [
          'image',
          'file',
          'video',
        ];
        $type_names = [
          'type',
          'bundle',
        ];

        // Special case for type as we do not want the reference for the
        // bundle. In additional to the type field a media entity has a
        // bundle field which stores a media bundle configuration entity UUID.
        if (in_array($name, $type_names, TRUE) && $referenced_entity instanceof ConfigEntityBase) {
          $values[$langcode][] = $referenced_entity->id();
        }
        elseif (in_array($field_type, $file_types)) {
          // If this is a file type, then add the asset to the CDF.
          $uuid_token = '[' . $referenced_entity->uuid() . ']';
          $asset_url = file_create_url($referenced_entity->getFileUri());
          $asset = new Asset();
          $asset->setUrl($asset_url);
          $asset->setReplaceToken($uuid_token);
          $contenthub_entity->addAsset($asset);

          // Now add the value.
          // Notice that we are including the "alt" and "title" attributes
          // from the file entity in the field data.
          $data = [
            'alt' => isset($items[$key]['alt']) ? $items[$key]['alt'] : '',
            'title' => isset($items[$key]['title']) ? $items[$key]['title'] : '',
            'target_uuid' => $uuid_token,
          ];
          $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
        }
        else {
          $values[$langcode][] = $referenced_entity->uuid();
        }
      }
    }
    else {
      // If there's nothing in this field, just set it to NULL.
      if ($items == NULL) {
        $values[$langcode] = NULL;
      }
      else {
        // Only if it is a link type.
        if ($link_field = ContentHubEntityLinkFieldHandler::load($field)->validate()) {
          $items = $link_field->normalizeItems($items);
        }

        // Loop over the items to get the values for each field.
        foreach ($items as $item) {
          // Hotfix.
          // @TODO: Find a better solution for this.
          if (isset($item['_attributes'])) {
            unset($item['_attributes']);
          }
          $keys = is_array($item) ? array_keys($item) : [];
          if (count($keys) == 1 && isset($item['value'])) {
            $value = $item['value'];
          }
          else {
            if ($field instanceof PathFieldItemList) {
              $item = $field->first()->getValue();
              $item['pid'] = "";
              $item['source'] = "";
            }
            $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
          }
          $values[$langcode][] = $value;
        }
      }
    }
    try {
      $attribute = new Attribute($type);
    }
    catch (\Exception $e) {
      $args['%type'] = $type;
      $message = new FormattableMarkup('No type could be registered for %type.', $args);
      throw new ContentHubException($message);
    }

    if (strstr($type, 'array')) {
      $attribute->setValues($values);
    }
    else {
      $value = array_pop($values[$langcode]);
      $attribute->setValue($value, $langcode);
    }

    // If attribute exists already, append to the existing values.
    if (!empty($contenthub_entity->getAttribute($name))) {
      $existing_attribute = $contenthub_entity->getAttribute($name);
      $this->appendToAttribute($existing_attribute, $attribute->getValues());
      $attribute = $existing_attribute;
    }

    // Add it to our contenthub entity.
    $contenthub_entity->setAttribute($name, $attribute);
  }

  // Allow alterations of the CDF to happen.
  $context['entity'] = $entity;
  $context['langcode'] = $langcode;
  $this->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);

  // Adds the entity URL to CDF.
  $value = NULL;
  if (empty($contenthub_entity->getAttribute('url'))) {
    global $base_path;
    switch ($entity->getEntityTypeId()) {
      case 'file':
        $value = file_create_url($entity->getFileUri());
        $filepath_attribute = new Attribute(Attribute::TYPE_STRING);
        $contenthub_entity->setAttribute('_filepath', $filepath_attribute->setValue($entity->getFileUri()));
        break;

      default:
        // Get entity URL.
        if (!$entity->isNew() && $entity->hasLinkTemplate('canonical')) {
          $url = $entity->toUrl();
          $url->setAbsolute(TRUE);
          $value = $url->toString();
        }
        break;
    }
    if (isset($value)) {
      $url_attribute = new Attribute(Attribute::TYPE_STRING);
      $contenthub_entity->setAttribute('url', $url_attribute->setValue($value, $langcode));
    }
  }
  return $contenthub_entity;
}
```

DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub

- 1 method



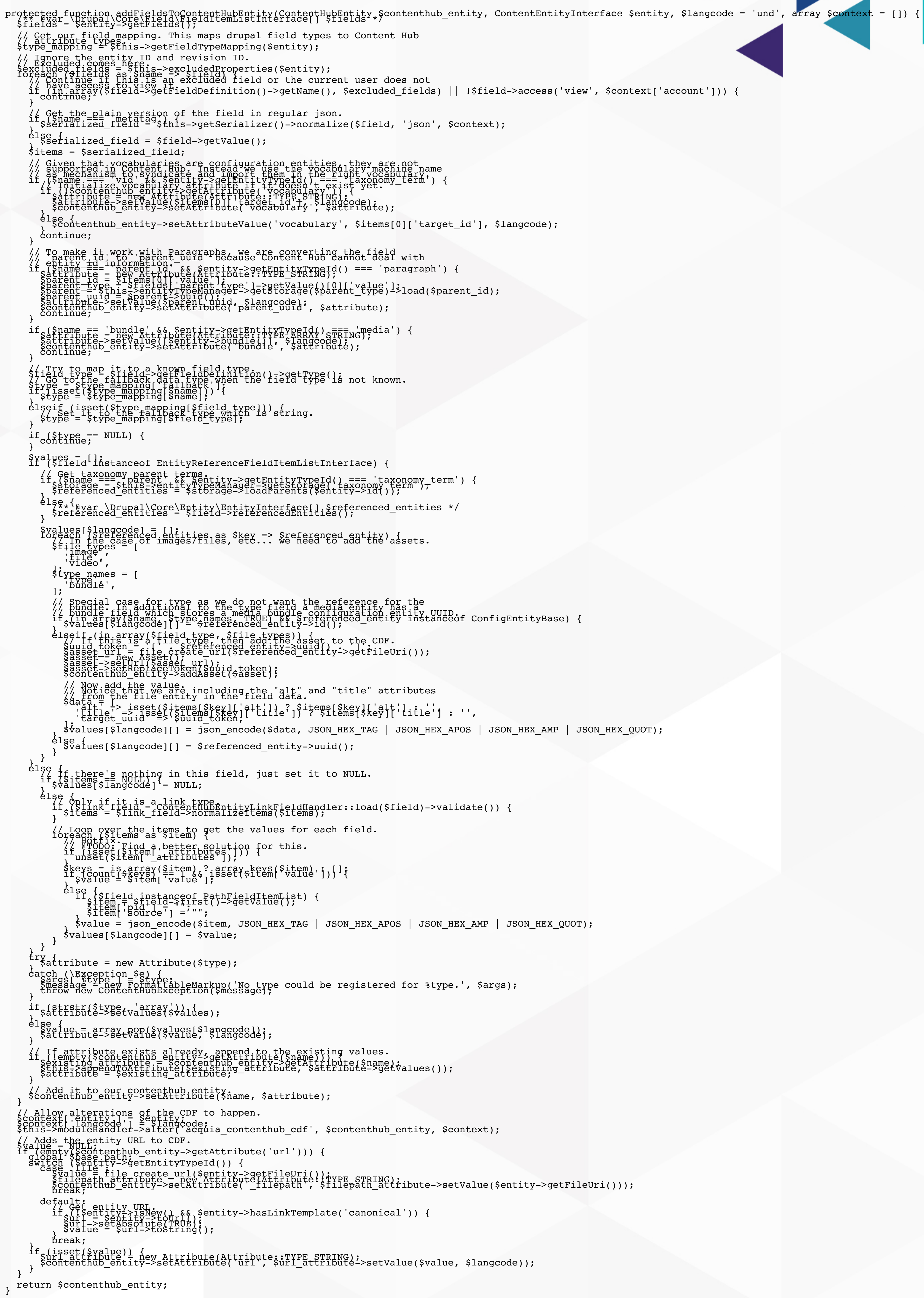
Acquia ContentHub

- 1 method
- 229 lines of code

```php
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
  /** @var \Drupal\Core\Field\FieldItemListInterface[] $fields */
  $fields = $entity->getFields();

  // Get our field mapping. This maps drupal field types to Content Hub
  // attribute types.
  $type_mapping = $this->getFieldTypeMapping($entity);

  // Ignore the entity ID and revision ID.
  // Excluded comes here.
  $excluded_fields = $this->excludedProperties($entity);
  foreach ($fields as $name => $field) {
    // Continue if this is an excluded field or the current user does not
    // have access to view it.
    if (in_array($field->getFieldDefinition()->getName(), $excluded_fields) || !$field->access('view', $context['account'])) {
      continue;
    }

    // Get the plain version of the field in regular json.
    if ($name === 'metatag') {
      $serialized_field = $this->getSerializer()->normalize($field, 'json', $context);
    }
    else {
      $serialized_field = $field->getValue();
    }
    $items = $serialized_field;

    // Given that vocabularies are configuration entities, they are not
    // supported in Content Hub. Instead we use the vocabulary machine name
    // as mechanism to syndicate and import them in the right vocabulary.
    if ($name === 'vid' && $entity->getEntityTypeId() === 'taxonomy_term') {
      // Initialize vocabulary attribute if it doesn't exist yet.
      if (!$contenthub_entity->getAttribute('vocabulary')) {
        $attribute = new Attribute(Attribute::TYPE_STRING);
        $attribute->setValue($items[0]['target_id'], $langcode);
        $contenthub_entity->setAttribute('vocabulary', $attribute);
      }
      else {
        $contenthub_entity->setAttributeValue('vocabulary', $items[0]['target_id'], $langcode);
      }
      continue;
    }

    // To make it work with Paragraphs, we are converting the field
    // 'parent_id' to 'parent_uuid' because Content Hub cannot deal with
    // entity_id information.
    if ($name === 'parent_id' && $entity->getEntityTypeId() === 'paragraph') {
      $attribute = new Attribute(Attribute::TYPE_STRING);
      $parent_id = $items[0]['value'];
      $parent_type = $fields['parent_type']->getValue()[0]['value'];
      $parent = $this->entityTypeManager->getStorage($parent_type)->load($parent_id);
      $parent_uuid = $parent->uuid();
      $attribute->setValue($parent_uuid, $langcode);
      $contenthub_entity->setAttribute('parent_uuid', $attribute);
      continue;
    }

    if ($name == 'bundle' && $entity->getEntityTypeId() === 'media') {
      $attribute = new Attribute(Attribute::TYPE_ARRAY_STRING);
      $attribute->setValue([$entity->bundle()], $langcode);
      $contenthub_entity->setAttribute('bundle', $attribute);
      continue;
    }

    // Try to map it to a known field type.
    $field_type = $field->getFieldDefinition()->getType();
    // Go to the fallback data type when the field type is not known.
    $type = $type_mapping['fallback'];
    if (isset($type_mapping[$name])) {
      $type = $type_mapping[$name];
    }
    elseif (isset($type_mapping[$field_type])) {
      // Set it to the fallback type which is string.
      $type = $type_mapping[$field_type];
    }

    if ($type == NULL) {
      continue;
    }

    $values = [];
    if ($field instanceof EntityReferenceFieldItemListInterface) {

      // Get taxonomy parent terms.
      if ($name === 'parent' && $entity->getEntityTypeId() === 'taxonomy_term') {
        $storage = $this->entityTypeManager->getStorage('taxonomy_term');
        $referenced_entities = $storage->loadParents($entity->id());
      }
      else {
        /** @var \Drupal\Core\Entity\EntityInterface[] $referenced_entities */
        $referenced_entities = $field->referencedEntities();
      }

      $values[$langcode] = [];
      foreach ($referenced_entities as $key => $referenced_entity) {
        // In the case of images/files, etc... we need to add the assets.
        $file_types = [
          'image',
          'file',
          'video',
        ];
        $type_names = [
          'type',
          'bundle',
        ];

        // Special case for type as we do not want the reference for the
        // bundle. In additional to the type field a media entity has a
        // bundle field which stores a media bundle configuration entity UUID.
        if (in_array($name, $type_names, TRUE) && $referenced_entity instanceof ConfigEntityBase) {
          $values[$langcode][] = $referenced_entity->id();
        }
        elseif (in_array($field_type, $file_types)) {
          // If this is a file type, then add the asset to the CDF.
          $uuid_token = '[' . $referenced_entity->uuid() . ']';
          $asset_url = file_create_url($referenced_entity->getFileUri());
          $asset = new Asset();
          $asset->setUrl($asset_url);
          $asset->setReplaceToken($uuid_token);
          $contenthub_entity->addAsset($asset);

          // Now add the value.
          // Notice that we are including the "alt" and "title" attributes
          // from the file entity in the field data.
          $data = [
            'alt' => isset($items[$key]['alt']) ? $items[$key]['alt'] : '',
            'title' => isset($items[$key]['title']) ? $items[$key]['title'] : '',
            'target_uuid' => $uuid_token,
          ];
          $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
        }
        else {
          $values[$langcode][] = $referenced_entity->uuid();
        }
      }
    }
    else {
      // If there's nothing in this field, just set it to NULL.
      if ($items == NULL) {
        $values[$langcode] = NULL;
      }
      else {
        // Only if it is a link type.
        if ($link_field = ContentHubEntityLinkFieldHandler::load($field)->validate()) {
          $items = $link_field->normalizeItems($items);
        }

        // Loop over the items to get the values for each field.
        foreach ($items as $item) {
          // Hotfix.
          // @TODO: Find a better solution for this.
          if (isset($item['_attributes'])) {
            unset($item['_attributes']);
          }
          $keys = is_array($item) ? array_keys($item) : [];
          if (count($keys) == 1 && isset($item['value'])) {
            $value = $item['value'];
          }
          else {
            if ($field instanceof PathFieldItemList) {
              $item = $field->first()->getValue();
              $item['pid'] = "";
              $item['source'] = "";
            }
            $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
          }
          $values[$langcode][] = $value;
        }
      }
    }
    try {
      $attribute = new Attribute($type);
    }
    catch (\Exception $e) {
      $args['%type'] = $type;
      $message = new FormattableMarkup('No type could be registered for %type.', $args);
      throw new ContentHubException($message);
    }

    if (strstr($type, 'array')) {
      $attribute->setValues($values);
    }
    else {
      $value = array_pop($values[$langcode]);
      $attribute->setValue($value, $langcode);
    }

    // If attribute exists already, append to the existing values.
    if (!empty($contenthub_entity->getAttribute($name))) {
      $existing_attribute = $contenthub_entity->getAttribute($name);
      $this->appendToAttribute($existing_attribute, $attribute->getValues());
      $attribute = $existing_attribute;
    }

    // Add it to our contenthub entity.
    $contenthub_entity->setAttribute($name, $attribute);
  }

  // Allow alterations of the CDF to happen.
  $context['entity'] = $entity;
  $context['langcode'] = $langcode;
  $this->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);

  // Adds the entity URL to CDF.
  $value = NULL;
  if (empty($contenthub_entity->getAttribute('url'))) {
    global $base_path;
    switch ($entity->getEntityTypeId()) {
      case 'file':
        $value = file_create_url($entity->getFileUri());
        $filepath_attribute = new Attribute(Attribute::TYPE_STRING);
        $contenthub_entity->setAttribute('_filepath', $filepath_attribute->setValue($entity->getFileUri()));
        break;

      default:
        // Get entity URL.
        if (!$entity->isNew() && $entity->hasLinkTemplate('canonical')) {
          $url = $entity->toUrl();
          $url->setAbsolute(TRUE);
          $value = $url->toString();
        }
        break;
    }
    if (isset($value)) {
      $url_attribute = new Attribute(Attribute::TYPE_STRING);
      $contenthub_entity->setAttribute('url', $url_attribute->setValue($value, $langcode));
    }
  }
  return $contenthub_entity;
}
```

Acquia ContentHub

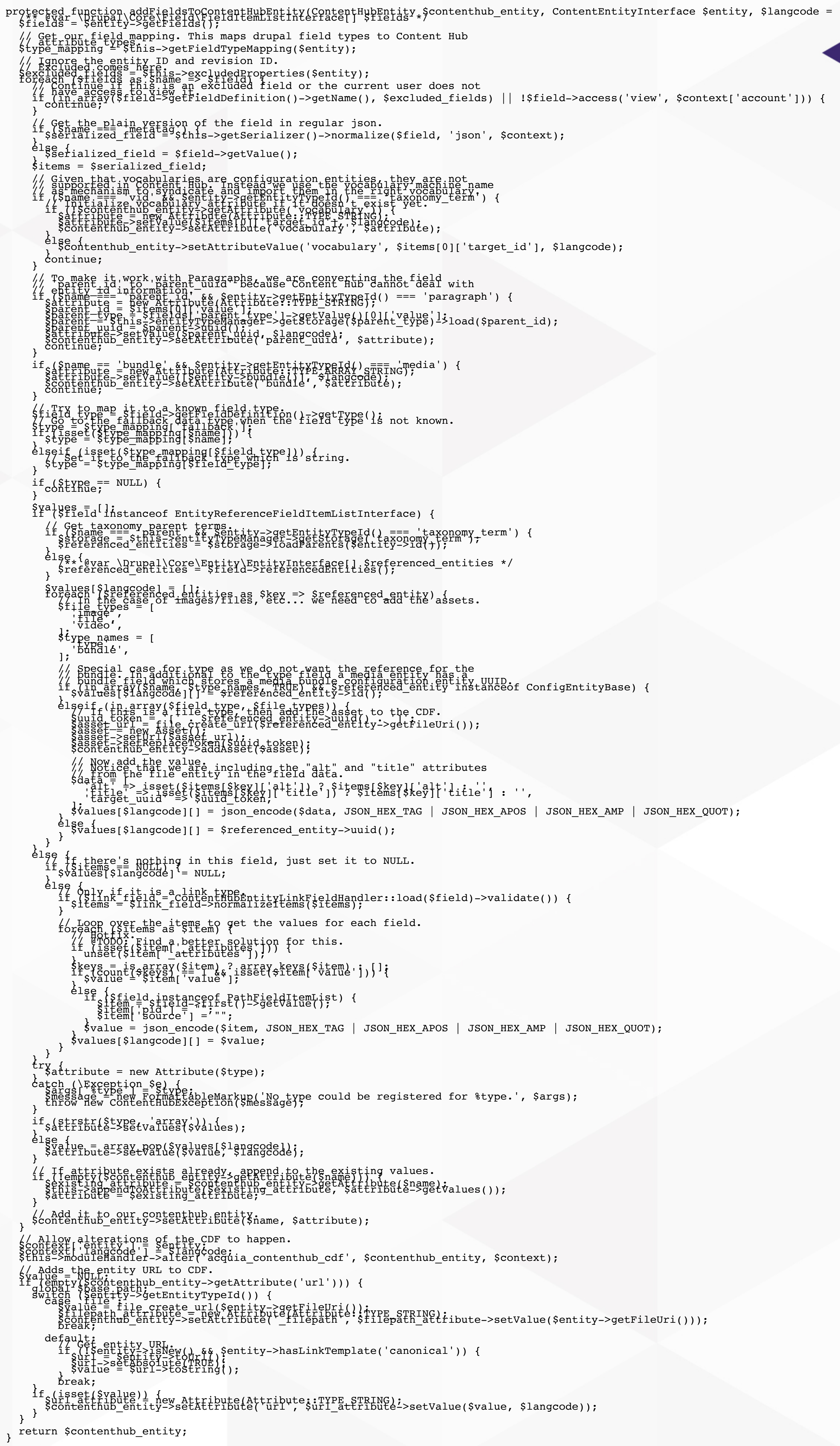
- 1 method
- 229 lines of code
- 4 separate entity type checks



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub

- 1 method
- 229 lines of code
- 4 separate entity type checks
  - Implies a need for custom logic



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub

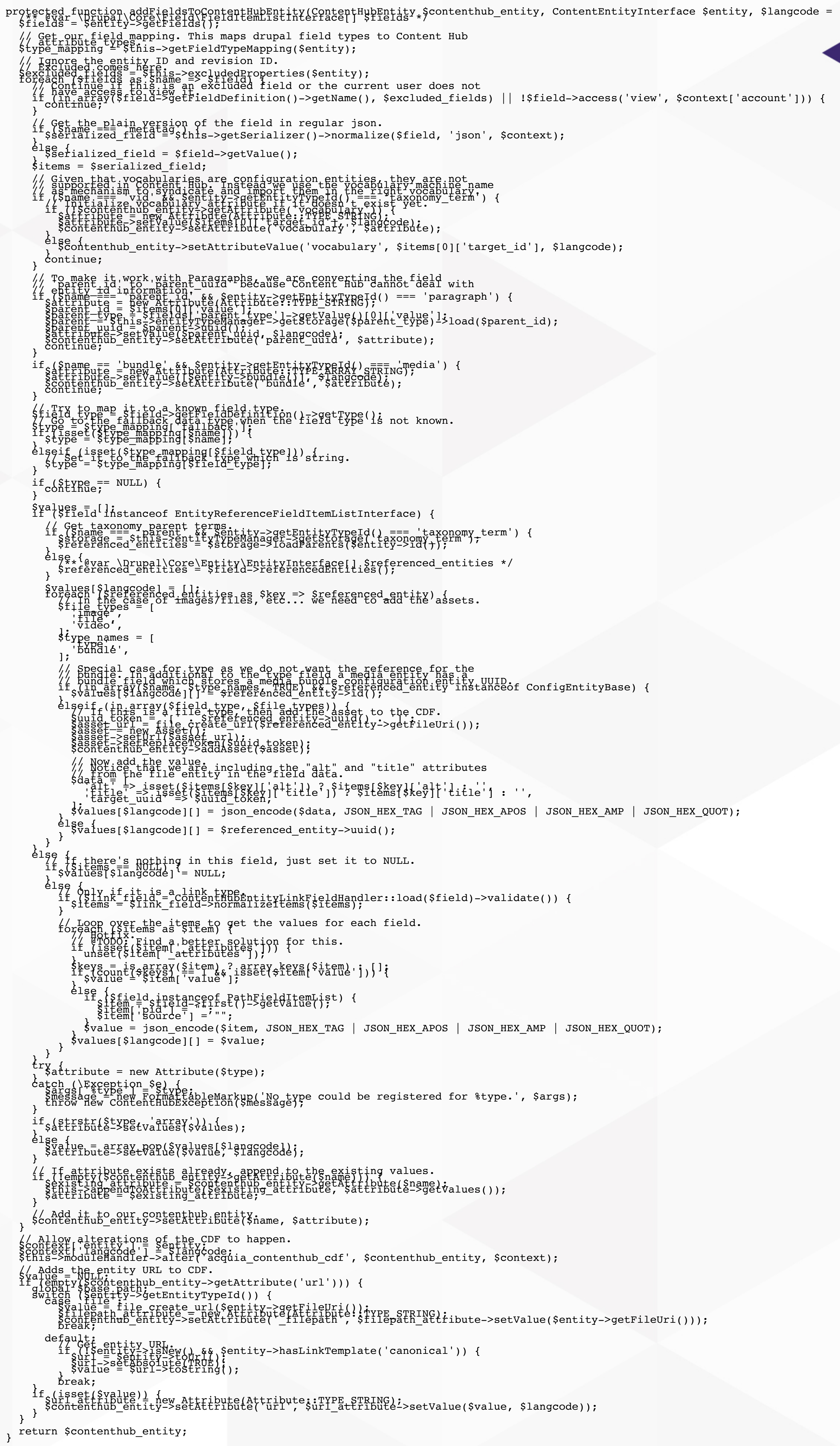
- 1 method
- 229 lines of code
- 4 separate entity type checks
  - Implies a need for custom logic
- 7 different field name checks

```
protected function addFieldsToContentHubEntity(ContentHubEntity $contenthub_entity, ContentEntityInterface $entity, $langcode = 'und', array $context = []) {
  /** @var \Drupal\Core\Field\FieldItemListInterface[] $fields */
  $fields = $entity->getFields();

  // Get our field mapping. This maps drupal field types to Content Hub
  // attribute types.
  $type_mapping = $this->getFieldTypeMapping($entity);

  // Ignore the entity ID and revision ID.
  // Excluded comes here.
  $excluded_fields = $this->excludedProperties($entity);
  foreach ($fields as $name => $field) {
    // Continue if this is an excluded field or the current user does not
    // have access to view it.
    if (in_array($field->getFieldDefinition()->getName(), $excluded_fields) || !$field->access('view', $context['account'])) {
      continue;
    }

    // Get the plain version of the field in regular json.
    if ($name === 'metatag') {
      $serialized_field = $this->getSerializer()->normalize($field, 'json', $context);
    }
    else {
      $serialized_field = $field->getValue();
    }
    $items = $serialized_field;

    // Given that vocabularies are configuration entities, they are not
    // supported in Content Hub. Instead we use the vocabulary machine name
    // as mechanism to syndicate and import them in the right vocabulary.
    if ($name === 'vid' && $entity->getEntityTypeId() === 'taxonomy_term') {
      // Initialize vocabulary attribute if it doesn't exist yet.
      if (!$contenthub_entity->getAttribute('vocabulary')) {
        $attribute = new Attribute(Attribute::TYPE_STRING);
        $attribute->setValue($items[0]['target_id'], $langcode);
        $contenthub_entity->setAttribute('vocabulary', $attribute);
      }
      else {
        $contenthub_entity->setAttributeValue('vocabulary', $items[0]['target_id'], $langcode);
      }
      continue;
    }

    // To make it work with Paragraphs, we are converting the field
    // 'parent_id' to 'parent_uuid' because Content Hub cannot deal with
    // entity_id information.
    if ($name === 'parent_id' && $entity->getEntityTypeId() === 'paragraph') {
      $attribute = new Attribute(Attribute::TYPE_STRING);
      $parent_id = $items[0]['value'];
      $parent_type = $fields['parent_type']->getValue()[0]['value'];
      $parent = $this->entityTypeManager->getStorage($parent_type)->load($parent_id);
      $parent_uuid = $parent->uuid();
      $attribute->setValue($parent_uuid, $langcode);
      $contenthub_entity->setAttribute('parent_uuid', $attribute);
      continue;
    }

    if ($name == 'bundle' && $entity->getEntityTypeId() === 'media') {
      $attribute = new Attribute(Attribute::TYPE_ARRAY_STRING);
      $attribute->setValue([$entity->bundle()], $langcode);
      $contenthub_entity->setAttribute('bundle', $attribute);
      continue;
    }

    // Try to map it to a known field type.
    $field_type = $field->getFieldDefinition()->getType();
    // Go to the fallback data type when the field type is not known.
    $type = $type_mapping['fallback'];
    if (isset($type_mapping[$name])) {
      $type = $type_mapping[$name];
    }
    elseif (isset($type_mapping[$field_type])) {
      // Set it to the fallback type which is string.
      $type = $type_mapping[$field_type];
    }

    if ($type == NULL) {
      continue;
    }

    $values = [];
    if ($field instanceof EntityReferenceFieldItemListInterface) {

      // Get taxonomy parent terms.
      if ($name === 'parent' && $entity->getEntityTypeId() === 'taxonomy_term') {
        $storage = $this->entityTypeManager->getStorage('taxonomy_term');
        $referenced_entities = $storage->loadParents($entity->id());
      }
      else {
        /** @var \Drupal\Core\Entity\EntityInterface[] $referenced_entities */
        $referenced_entities = $field->referencedEntities();
      }

      $values[$langcode] = [];
      foreach ($referenced_entities as $key => $referenced_entity) {
        // In the case of images/files, etc... we need to add the assets.
        $file_types = [
          'image',
          'file',
          'video',
        ];
        $type_names = [
          'type',
          'bundle',
        ];

        // Special case for type as we do not want the reference for the
        // bundle. In additional to the type field a media entity has a
        // bundle field which stores a media bundle configuration entity UUID.
        if (in_array($name, $type_names, TRUE) && $referenced_entity instanceof ConfigEntityBase) {
          $values[$langcode][] = $referenced_entity->id();
        }
        elseif (in_array($field_type, $file_types)) {
          // If this is a file type, then add the asset to the CDF.
          $uuid_token = '[' . $referenced_entity->uuid() . ']';
          $asset_url = file_create_url($referenced_entity->getFileUri());
          $asset = new Asset();
          $asset->setUrl($asset_url);
          $asset->setReplaceToken($uuid_token);
          $contenthub_entity->addAsset($asset);

          // Now add the value.
          // Notice that we are including the "alt" and "title" attributes
          // from the file entity in the field data.
          $data = [
            'alt' => isset($items[$key]['alt']) ? $items[$key]['alt'] : '',
            'title' => isset($items[$key]['title']) ? $items[$key]['title'] : '',
            'target_uuid' => $uuid_token,
          ];
          $values[$langcode][] = json_encode($data, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
        }
        else {
          $values[$langcode][] = $referenced_entity->uuid();
        }
      }
    }
    else {
      // If there's nothing in this field, just set it to NULL.
      if ($items == NULL) {
        $values[$langcode] = NULL;
      }
      else {
        // Only if it is a link type.
        if ($link_field = ContentHubEntityLinkFieldHandler::load($field)->validate()) {
          $items = $link_field->normalizeItems($items);
        }

        // Loop over the items to get the values for each field.
        foreach ($items as $item) {
          // Hotfix.
          // @TODO: Find a better solution for this.
          if (isset($item['_attributes'])) {
            unset($item['_attributes']);
          }
          $keys = is_array($item) ? array_keys($item) : [];
          if (count($keys) == 1 && isset($item['value'])) {
            $value = $item['value'];
          }
          else {
            if ($field instanceof PathFieldItemList) {
              $item = $field->first()->getValue();
              $item['pid'] = "";
              $item['source'] = "";
            }
            $value = json_encode($item, JSON_HEX_TAG | JSON_HEX_APOS | JSON_HEX_AMP | JSON_HEX_QUOT);
          }
          $values[$langcode][] = $value;
        }
      }
    }
    try {
      $attribute = new Attribute($type);
    }
    catch (\Exception $e) {
      $args['%type'] = $type;
      $message = new FormattableMarkup('No type could be registered for %type.', $args);
      throw new ContentHubException($message);
    }

    if (strstr($type, 'array')) {
      $attribute->setValues($values);
    }
    else {
      $value = array_pop($values[$langcode]);
      $attribute->setValue($value, $langcode);
    }

    // If attribute exists already, append to the existing values.
    if (!empty($contenthub_entity->getAttribute($name))) {
      $existing_attribute = $contenthub_entity->getAttribute($name);
      $this->appendToAttribute($existing_attribute, $attribute->getValues());
      $attribute = $existing_attribute;
    }

    // Add it to our contenthub entity.
    $contenthub_entity->setAttribute($name, $attribute);
  }

  // Allow alterations of the CDF to happen.
  $context['entity'] = $entity;
  $context['langcode'] = $langcode;
  $this->moduleHandler->alter('acquia_contenthub_cdf', $contenthub_entity, $context);

  // Adds the entity URL to CDF.
  $value = NULL;
  if (empty($contenthub_entity->getAttribute('url'))) {
    global $base_path;
    switch ($entity->getEntityTypeId()) {
      case 'file':
        $value = file_create_url($entity->getFileUri());
        $filepath_attribute = new Attribute(Attribute::TYPE_STRING);
        $contenthub_entity->setAttribute('_filepath', $filepath_attribute->setValue($entity->getFileUri()));
        break;

      default:
        // Get entity URL.
        if (!$entity->isNew() && $entity->hasLinkTemplate('canonical')) {
          $url = $entity->toUrl();
          $url->setAbsolute(TRUE);
          $value = $url->toString();
        }
        break;
    }
    if (isset($value)) {
      $url_attribute = new Attribute(Attribute::TYPE_STRING);
      $contenthub_entity->setAttribute('url', $url_attribute->setValue($value, $langcode));
    }
  }
  return $contenthub_entity;
}
```

DrupalCon SEATTLE 2019 APRIL 8-12

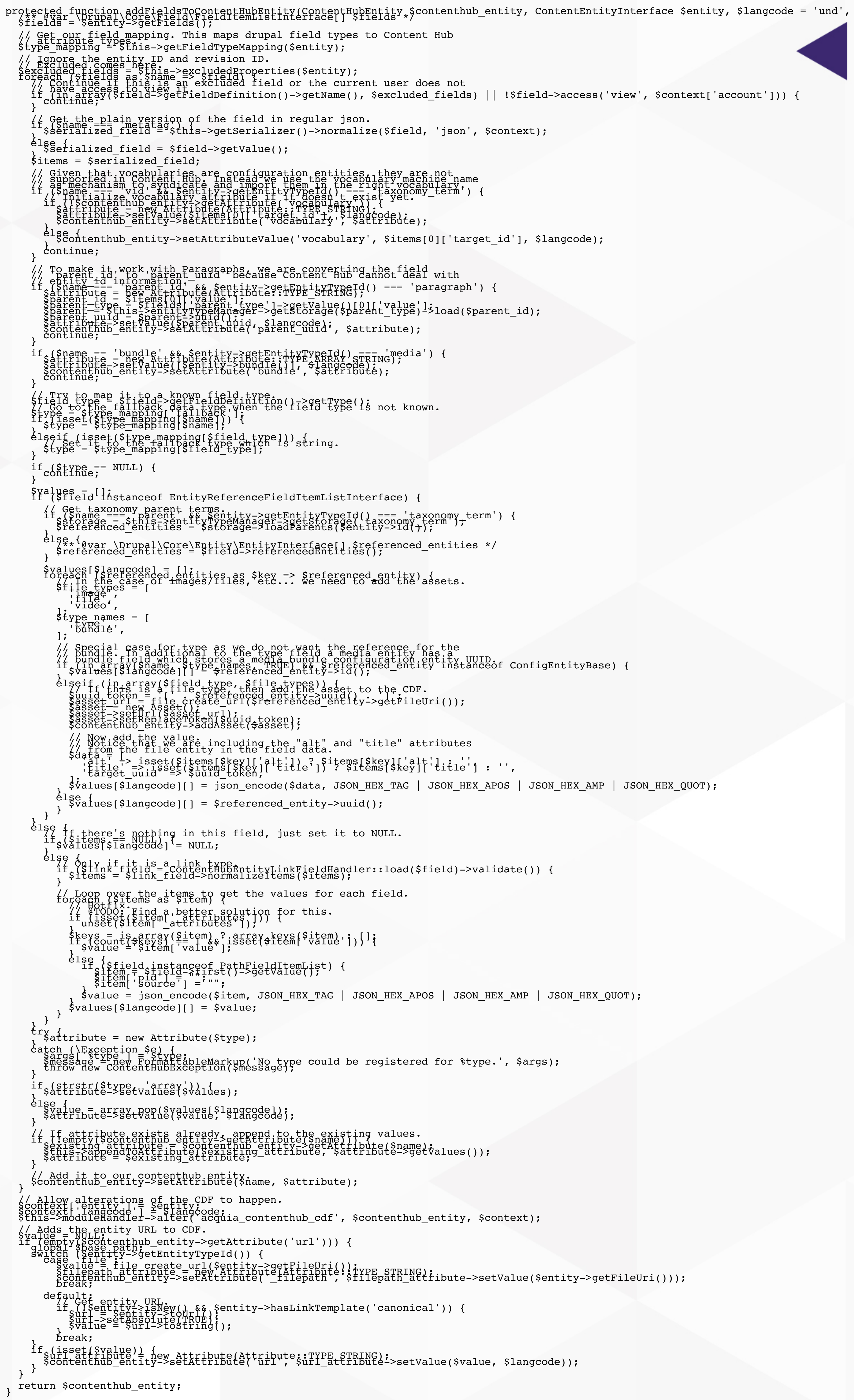
Acquia ContentHub

- 1 method
- 229 lines of code
- 4 separate entity type checks
  - Implies a need for custom logic
- 7 different field name checks
- 2 field instanceof checks



Acquia ContentHub

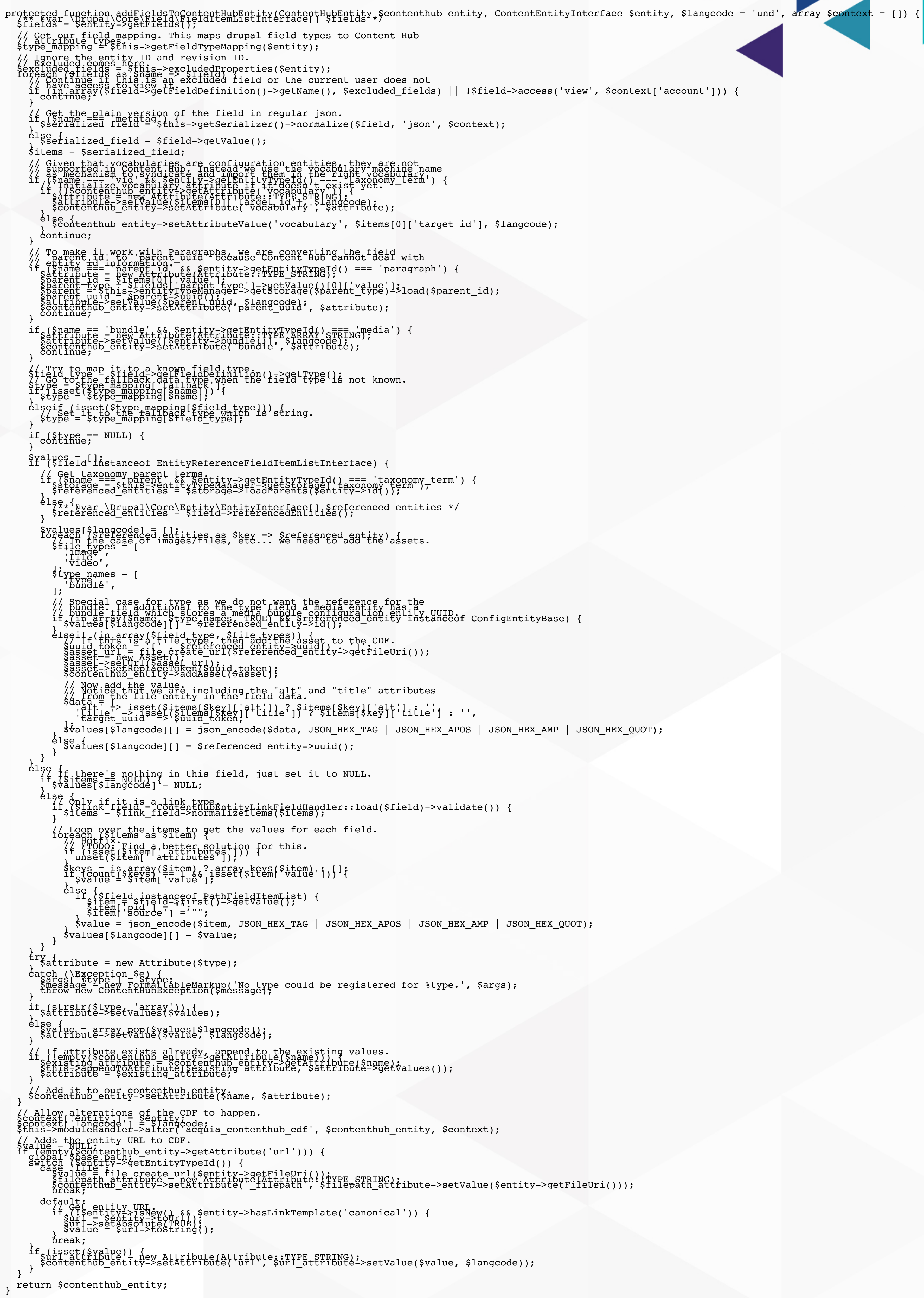
- 1 method
- 229 lines of code
- 4 separate entity type checks
  - Implies a need for custom logic
- 7 different field name checks
- 2 field instanceof checks
- 3 calls to other local methods



DrupalCon
SEATTLE 2019
APRIL 8-12

Acquia ContentHub

- 1 method
- 229 lines of code
- 4 separate entity type checks
  - Implies a need for custom logic
- 7 different field name checks
- 2 field instanceof checks
- 3 calls to other local methods
- 1 alter hook



Acquia ContentHub

- Why bring up this method? It’s not a hook.

Acquia ContentHub

- Why bring up this method? It’s not a hook.
- Hooks normalize this sort of coding

Acquia ContentHub

- Why bring up this method? It’s not a hook.
- Hooks normalize this sort of coding
- Hooks break/fuzz SRP

Acquia ContentHub

- Why bring up this method? It’s not a hook.
- Hooks normalize this sort of coding
- Hooks break/fuzz SRP
- Hooks have to concern themselves with implementation details

Acquia ContentHub

- Why bring up this method? It’s not a hook.
- Hooks normalize this sort of coding
- Hooks break/fuzz SRP
- Hooks have to concern themselves with implementation details
- Hooks simultaneously respond to an event and hold logic for all possible returns

Drupal 7 module_invoke ‘block_view’

```
$array = module_invoke($block->module, 'block_view', $block->delta);
```

```php
function system_block_view($delta = '') {
  $block = array();
  switch ($delta) {
    case 'main':
      $block['subject'] = NULL;
      $block['content'] = drupal_set_page_content();
      return $block;
    case 'powered-by':
      $block['subject'] = NULL;
      $block['content'] = theme('system_powered_by');
      return $block;
    case 'help':
      $block['subject'] = NULL;
      $block['content'] = menu_get_active_help();
      return $block;
    default:
      // All system menu blocks.
      $system_menus = menu_list_system_menus();
      if (isset($system_menus[$delta])) {
        $block['subject'] = t($system_menus[$delta]);
        $block['content'] = menu_tree($delta);
        return $block;
      }
      break;
  }
}
```

```php
function system_block_view($delta = '') {
  $block = array();
  switch ($delta) {
    case 'main':
      $block['subject'] = NULL;
      $block['content'] = drupal_set_page_content();
      return $block;
    case 'powered-by':
      $block['subject'] = NULL;
      $block['content'] = theme('system_powered_by');
      return $block;
    case 'help':
      $block['subject'] = NULL;
      $block['content'] = menu_get_active_help();
      return $block;
    default:
      // All system menu blocks.
      $system_menus = menu_list_system_menus();
      if (isset($system_menus[$delta])) {
        $block['subject'] = t($system_menus[$delta]);
        $block['content'] = menu_tree($delta);
        return $block;
      }
      break;
  }
}
```

```php
function system_block_view($delta = '') {
  $block = array();
  switch ($delta) {
    case 'main':
      $block['subject'] = NULL;
      $block['content'] = drupal_set_page_content();
      return $block;
    case 'powered-by':
      $block['subject'] = NULL;
      $block['content'] = theme('system_powered_by');
      return $block;
    case 'help':
      $block['subject'] = NULL;
      $block['content'] = menu_get_active_help();
      return $block;
    default:
      // All system menu blocks.
      $system_menus = menu_list_system_menus();
      if (isset($system_menus[$delta])) {
        $block['subject'] = t($system_menus[$delta]);
        $block['content'] = menu_tree($delta);
        return $block;
      }
      break;
  }
}
```

```php
function system_block_view($delta = '') {
  $block = array();
  switch ($delta) {
    case 'main':
      $block['subject'] = NULL;
      $block['content'] = drupal_set_page_content();
      return $block;
    case 'powered-by':
      $block['subject'] = NULL;
      $block['content'] = theme('system_powered_by');
      return $block;
    case 'help':
      $block['subject'] = NULL;
      $block['content'] = menu_get_active_help();
      return $block;
    default:
      // All system menu blocks.
      $system_menus = menu_list_system_menus();
      if (isset($system_menus[$delta])) {
        $block['subject'] = t($system_menus[$delta]);
        $block['content'] = menu_tree($delta);
        return $block;
      }
      break;
  }
}
```

```php
function system_block_view($delta = '') {
  $block = array();
  switch ($delta) {
    case 'main':
      $block['subject'] = NULL;
      $block['content'] = drupal_set_page_content();
      return $block;
    case 'powered-by':
      $block['subject'] = NULL;
      $block['content'] = theme('system_powered_by');
      return $block;
    case 'help':
      $block['subject'] = NULL;
      $block['content'] = menu_get_active_help();
      return $block;
    default:
      // All system menu blocks.
      $system_menus = menu_list_system_menus();
      if (isset($system_menus[$delta])) {
        $block['subject'] = t($system_menus[$delta]);
        $block['content'] = menu_tree($delta);
        return $block;
      }
      break;
  }
}
```

Upgrades and Replacements

“

Perfection is achieved not when there is nothing more to add, but rather when there is nothing more to take away.

”

Antoine de Saint-Exupery

Upgrades & Replacements

Upgrades & Replacements

- Plugins

Upgrades & Replacements

- Plugins
- Event Subscribers

Spoiler Alert!!

Spoiler Alert!!

- This is not a plugins talk

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis
  - Plugins usually have a UI interaction.

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis
  - Plugins usually have a UI interaction.
  - Plugins are backed by interfaces.

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis
  - Plugins usually have a UI interaction.
  - Plugins are backed by interfaces.
  - Plugins are collections of methods that used to be related hook invocations.

Spoiler Alert!!

- This is not a plugins talk
- Quick Synopsis
  - Plugins usually have a UI interaction.
  - Plugins are backed by interfaces.
  - Plugins are collections of methods that used to be related hook invocations.
  - i.e. hook_block_*

Event Subscribers

Event Subscribers

- Individual classes which respond to one or more dispatched events.

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
  - Event classes define property mutability

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
  - Event classes define property mutability
  - Can replace both traditional hooks and alter hooks simultaneously

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
  - Event classes define property mutability
  - Can replace both traditional hooks and alter hooks simultaneously
- Prioritize-able

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
  - Event classes define property mutability
  - Can replace both traditional hooks and alter hooks simultaneously
- Prioritize-able
- “Propagation” can be stopped

Event Subscribers

- Individual classes which respond to one or more dispatched events.
- Multiple classes in the same code base (module) can subscribe to the same events.
- Dependency inject-able services
- Depend on Event classes
  - Event classes define property mutability
  - Can replace both traditional hooks and alter hooks simultaneously
- Prioritize-able
- “Propagation” can be stopped
- Easily testable

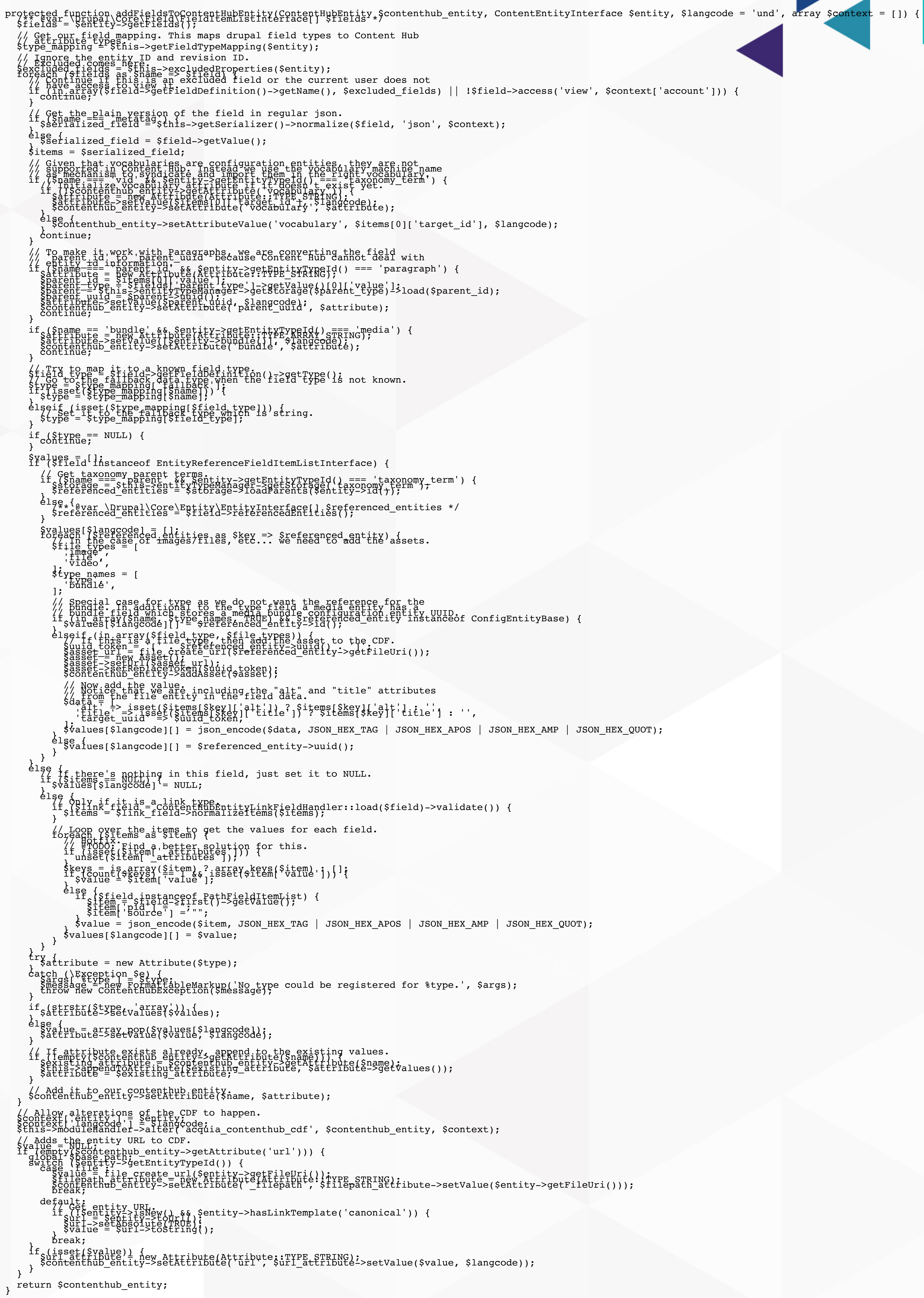
Practical Application

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

Martin Fowler

Acquia ContentHub

- 1 method
- 229 lines of code
- 4 separate entity type checks
- 7 different field name checks
- 2 field instanceof checks
- 3 calls to other local methods
- 1 alter hook



Acquia ContentHub

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub

- 1 method

```php
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub

- 1 method
- 58 lines of code

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub

- 1 method
- 58 lines of code
- 0 separate entity type checks

```php
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub

- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub

- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks
- 0 field instanceof checks

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub

- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks
- 0 field instanceof checks
- 1 call another local method

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub

- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks
- 0 field instanceof checks
- 1 call another local method
- 0 alter hook

```
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub

- 1 method
- 58 lines of code
- 0 separate entity type checks
- 0 different field name checks
- 0 field instanceof checks
- 1 call another local method
- 0 alter hook
- 5 dispatched events

```php
public function unserializeEntities(CDFDocument $cdf, DependencyStack $stack) {
  if (!$cdf->hasEntities()) {
    throw new \Exception("Missing CDF Entities entry. Not a valid CDF.");
  }
  $event = new PruneCdfEntitiesEvent($cdf);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::PRUNE_CDF, $event);
  $cdf = $event->getCdf();
  $this->handleModules($cdf, $stack);
  // Allows entity data to be manipulated before unserialization.
  $event = new EntityDataTamperEvent($cdf, $stack);
  $this->dispatcher->dispatch(AcquiaContentHubEvents::ENTITY_DATA_TAMPER, $event);
  $cdf = $event->getCdf();
  // Organize the entities into a dependency chain.
  // Use a while loop to prevent memory expansion due to recursion.
  while (!$stack->hasDependencies(array_keys($cdf->getEntities()))) {
    // @todo add tracking to break out of the while loop when dependencies cannot
be further processed.
    $count = count($stack->getDependencies());
    foreach ($cdf->getEntities() as $uuid => $entity_data) {
      if (!$stack->hasDependency($uuid) && $this-
>entityIsProcessable($entity_data, $stack)) {
        $event = new LoadLocalEntityEvent($entity_data, $stack);
        $this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
        $event = new ParseCDFEntityEvent($entity_data, $stack, $event-
>getEntity());
        $this->dispatcher->dispatch(AcquiaContentHubEvents::PARSE_CDF, $event);
        $entity = $event->getEntity();
        if ($entity) {
          $entity->save();
          $wrapper = new DependentEntityWrapper($entity);
          // Config uuids can be more fluid since they can match on id.
          if ($wrapper->getUuid() != $uuid) {
            $wrapper->setRemoteUuid($uuid);
          }
          $stack->addDependency($wrapper);
          if ($entity->isNew()) {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_NEW;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          else {
            $event_name = AcquiaContentHubEvents::ENTITY_IMPORT_UPDATE;
            $event = new EntityImportEvent($entity, $entity_data);
          }
          $this->dispatcher->dispatch($event_name, $event);
        }
        else {
          // Remove CDF Entities that were processable but didn't resolve into
          // an entity.
          $cdf->removeCDFEntity($uuid);
        }
      }
    }
    if ($count === count($stack->getDependencies())) {
      // @todo get import failure logging and tracking working.
      $event = new FailedImportEvent($cdf, $stack, $count);
      $this->dispatcher->dispatch(AcquiaContentHubEvents::IMPORT_FAILURE, $event);
      if ($event->hasException()) {
        throw $event->getException();
      }
    }
  }
}
```

Acquia ContentHub 8.x-2.x

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

Acquia ContentHub 8.x-2.x

```php
public static function getSubscribedEvents() {
  $events[AcquiaContentHubEvents::LOAD_LOCAL_ENTITY][] = ['onLoadLocalEntity', 10];
  return $events;
}

public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()
[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id,
$cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
}
```

Acquia ContentHub 8.x-2.x

```php
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  if ($cdf->getType() !== 'drupal8_content_entity') {
    return;
  }
  $attribute = $cdf->getAttribute('entity_type');
  // We only care about user entities.
  if (!$attribute || $attribute->getValue()['und'] !== 'user') {
    return;
  }
  // Don't do anything with anonymous users.
  if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
    return;
  }
  $mail_attribute = $cdf->getAttribute('mail');
  if (!$mail_attribute) {
    return;
  }
  $mail = $mail_attribute->getValue()['und'];
  /** @var \Drupal\user\UserInterface $account */
  $account = user_load_by_mail($mail);
  if ($account) {
    $event->setEntity($account);
    $event->stopPropagation();
    return;
  }
}
```

Pseudo-hook-comparison

```php
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
  $cdf = $event->getCdf();
  if ($cdf->getType() !== 'drupal8_content_entity') {
    return;
  }
  $attribute = $cdf->getAttribute('entity_type');
  // We only care about user entities.
  if (!$attribute || $attribute->getValue()['und'] !== 'user') {
    return;
  }
  // Don't do anything with anonymous users.
  if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
    return;
  }
  $mail_attribute = $cdf->getAttribute('mail');
  if (!$mail_attribute) {
    return;
  }
  $mail = $mail_attribute->getValue()['und'];
  /** @var \Drupal\user\UserInterface $account */
  $account = user_load_by_mail($mail);
  if ($account) {
    $event->setEntity($account);
    $event->stopPropagation();
    return;
  }
}
```

Pseudo-hook-comparison

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
  $cdf = $event->getCdf();
  if ($cdf->getType() !== 'drupal8_content_entity') {
    return;
  }
  $attribute = $cdf->getAttribute('entity_type');
  // We only care about user entities.
  if (!$attribute || $attribute->getValue()['und'] !== 'user') {
    return;
  }
  // Don't do anything with anonymous users.
  if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
    return;
  }
  $mail_attribute = $cdf->getAttribute('mail');
  if (!$mail_attribute) {
    return;
  }
  $mail = $mail_attribute->getValue()['und'];
  /** @var \Drupal\user\UserInterface $account */
  $account = user_load_by_mail($mail);
  if ($account) {
    $event->setEntity($account);
    $event->stopPropagation();
    return;
  }
}
```

Pseudo-hook-comparison

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
  $cdf = $event->getCdf();
  if ($cdf->getType() !== 'drupal8_content_entity') {
    return;
  }
  $attribute = $cdf->getAttribute('entity_type');
  // We only care about user entities.
  if (!$attribute || $attribute->getValue()['und'] !== 'user') {
    return;
  }
  // Don't do anything with anonymous users.
  if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
    return;
  }
  $mail_attribute = $cdf->getAttribute('mail');
  if (!$mail_attribute) {
    return;
  }
  $mail = $mail_attribute->getValue()['und'];
  /** @var \Drupal\user\UserInterface $account */
  $account = user_load_by_mail($mail);
  if ($account) {
    $event->setEntity($account);
    $event->stopPropagation();
    return;
  }
}
```

Event Definition

Event Definition

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

Event Definition

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

Event Definition

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()
[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id,
$cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
}
```

Event Definition

```
$event = new LoadLocalEntityEvent($entity_data, $stack);
$this->dispatcher->dispatch(AcquiaContentHubEvents::LOAD_LOCAL_ENTITY,
$event);
```

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()
[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id,
$cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
}
```

Event Definition

```
LoadLocalEntityEvent
    cdf:Acquia\ContentHubClient\CDF\CDFObject
    stack:Drupal\depcalc\DependencyStack
    entity:Drupal\Core\Entity\EntityInterface
    propagationStopped:bool
    __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
    isPropagationStopped():bool
    getCdf():\Acquia\ContentHubClient\CDF\CDFObject
    getStack():\Drupal\depcalc\DependencyStack
    stopPropagation():void
    getEntity():\Drupal\Core\Entity\EntityInterface
    hasEntity():bool
    setEntity(entity : \Drupal\Core\Entity\EntityInterface):void
```

Event Definition

```
LoadLocalEntityEvent
    cdf:Acquia\ContentHubClient\CDF\CDFObject
    stack:Drupal\depcalc\DependencyStack
    entity:Drupal\Core\Entity\EntityInterface
    propagationStopped:bool
    __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
    isPropagationStopped():bool
    getCdf():\Acquia\ContentHubClient\CDF\CDFObject
    getStack():\Drupal\depcalc\DependencyStack
    stopPropagation():void
    getEntity():\Drupal\Core\Entity\EntityInterface
    hasEntity():bool
    setEntity(entity : \Drupal\Core\Entity\EntityInterface):void
```

DrupalCon
SEATTLE 2019
APRIL 8-12

Event Definition

```
LoadLocalEntityEvent
    cdf:Acquia\ContentHubClient\CDF\CDFObject
    stack:Drupal\depcalc\DependencyStack
    entity:Drupal\Core\Entity\EntityInterface
    propagationStopped:bool
    __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
    isPropagationStopped():bool
    getCdf():\Acquia\ContentHubClient\CDF\CDFObject
    getStack():\Drupal\depcalc\DependencyStack
    stopPropagation():void
    getEntity():\Drupal\Core\Entity\EntityInterface
    hasEntity():bool
    setEntity(entity : \Drupal\Core\Entity\EntityInterface):void
```

Event Definition

```
LoadLocalEntityEvent
    cdf:Acquia\ContentHubClient\CDF\CDFObject
    stack:Drupal\depcalc\DependencyStack
    entity:Drupal\Core\Entity\EntityInterface
    propagationStopped:bool
    __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
    isPropagationStopped():bool
    getCdf():\Acquia\ContentHubClient\CDF\CDFObject
    getStack():\Drupal\depcalc\DependencyStack
    stopPropagation():void
    getEntity():\Drupal\Core\Entity\EntityInterface
    hasEntity():bool
    setEntity(entity : \Drupal\Core\Entity\EntityInterface):void
```

Event Definition

```
LoadLocalEntityEvent
    cdf:Acquia\ContentHubClient\CDF\CDFObject
    stack:Drupal\depcalc\DependencyStack
    entity:Drupal\Core\Entity\EntityInterface
    propagationStopped:bool
    __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
    isPropagationStopped():bool
    getCdf() \Acquia\ContentHubClient\CDF\CDFObject
    getStack() \Drupal\depcalc\DependencyStack
    stopPropagation():void
    getEntity() \Drupal\Core\Entity\EntityInterface
    hasEntity():bool
    setEntity(entity : \Drupal\Core\Entity\EntityInterface) void
```

Event Propagation

```
LoadLocalEntityEvent
    cdf:Acquia\ContentHubClient\CDF\CDFObject
    stack:Drupal\depcalc\DependencyStack
    entity:Drupal\Core\Entity\EntityInterface
    propagationStopped:bool
    __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
    isPropagationStopped():bool
    getCdf():\Acquia\ContentHubClient\CDF\CDFObject
    getStack():\Drupal\depcalc\DependencyStack
    stopPropagation():void
    getEntity():\Drupal\Core\Entity\EntityInterface
    hasEntity():bool
    setEntity(entity : \Drupal\Core\Entity\EntityInterface):void
```

Event Propagation

```
LoadLocalEntityEvent
    cdf:Acquia\ContentHubClient\CDF\CDFObject
    stack:Drupal\depcalc\DependencyStack
    entity:Drupal\Core\Entity\EntityInterface
    propagationStopped:bool
    __construct(cdf : \Acquia\ContentHubClient\CDF\CDFObject, stack : \Drupal\depcalc\DependencyStack)
    isPropagationStopped():bool
    getCdf():\Acquia\ContentHubClient\CDF\CDFObject
    getStack():\Drupal\depcalc\DependencyStack
    stopPropagation() void
    getEntity():\Drupal\Core\Entity\EntityInterface
    hasEntity():bool
    setEntity(entity : \Drupal\Core\Entity\EntityInterface):void
```

Event Propagation

```php
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
  $cdf = $event->getCdf();
  if ($cdf->getType() !== 'drupal8_content_entity') {
    return;
  }
  $attribute = $cdf->getAttribute('entity_type');
  // We only care about user entities.
  if (!$attribute || $attribute->getValue()['und'] !== 'user') {
    return;
  }
  // Don't do anything with anonymous users.
  if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
    return;
  }
  $mail_attribute = $cdf->getAttribute('mail');
  if (!$mail_attribute) {
    return;
  }
  $mail = $mail_attribute->getValue()['und'];
  /** @var \Drupal\user\UserInterface $account */
  $account = user_load_by_mail($mail);
  if ($account) {
    $event->setEntity($account);
    $event->stopPropagation();
    return;
  }
}
```

Event Propagation

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
  $cdf = $event->getCdf();
  if ($cdf->getType() !== 'drupal8_content_entity') {
    return;
  }
  $attribute = $cdf->getAttribute('entity_type');
  // We only care about user entities.
  if (!$attribute || $attribute->getValue()['und'] !== 'user') {
    return;
  }
  // Don't do anything with anonymous users.
  if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
    return;
  }
  $mail_attribute = $cdf->getAttribute('mail');
  if (!$mail_attribute) {
    return;
  }
  $mail = $mail_attribute->getValue()['und'];
  /** @var \Drupal\user\UserInterface $account */
  $account = user_load_by_mail($mail);
  if ($account) {
    $event->setEntity($account);
    $event->stopPropagation();
    return;
  }
}
```

Event Propagation

```
public function onLoadLocalEntity(LoadLocalEntityEvent $event) {
  $cdf = $event->getCdf();
  $entity_type_id = $cdf->getAttribute('entity_type')->getValue()[CDFObject::LANGUAGE_UNDETERMINED];
  if ($entity = $this->getEntityRepository()->loadEntityByUuid($entity_type_id, $cdf->getUuid())) {
    $event->setEntity($entity);
    $event->stopPropagation();
  }
  $cdf = $event->getCdf();
  if ($cdf->getType() !== 'drupal8_content_entity') {
    return;
  }
  $attribute = $cdf->getAttribute('entity_type');
  // We only care about user entities.
  if (!$attribute || $attribute->getValue()['und'] !== 'user') {
    return;
  }
  // Don't do anything with anonymous users.
  if ($anonymous = $event->getCdf()->getAttribute('is_anonymous')) {
    return;
  }
  $mail_attribute = $cdf->getAttribute('mail');
  if (!$mail_attribute) {
    return;
  }
  $mail = $mail_attribute->getValue()['und'];
  /** @var \Drupal\user\UserInterface $account */
  $account = user_load_by_mail($mail);
  if ($account) {
    $event->setEntity($account);
    $event->stopPropagation();
    return;
  }
}
```

Return

“Event Propagation" in D7

“Event Propagation" in D7

- module_implements()

“Event Propagation" in D7

- module_implements()
  - module_invoke()

“Event Propagation" in D7

- module_implements()
  - module_invoke()
- module_invoke_all()

EventSubscriber Priority

EventSubscriber Priority

- EventSubscribers default to a 0 priority

EventSubscriber Priority

- EventSubscribers default to a 0 priority
- Priorities of same weight non-deterministically execute
- Priority is opposite of weight (higher priority happens earlier)

EventSubscriber Priority

```php
public static function getSubscribedEvents() {
  $events[AcquiaContentHubEvents::LOAD_LOCAL_ENTITY][] = ['onLoadLocalEntity', 10];
  return $events;
}
```

EventSubscriber Priority

```php
public static function getSubscribedEvents() {
  $events[AcquiaContentHubEvents::LOAD_LOCAL_ENTITY][] = ['onLoadLocalEntity', 10];
  return $events;
}
```

EventSubscriber Testing

EventSubscriber Testing

- Mock the dependencies

EventSubscriber Testing

- Mock the dependencies
- Create the event

EventSubscriber Testing

- Mock the dependencies
- Create the event
- Pass it to the public method on the event subscriber

EventSubscriber Testing

- Mock the dependencies
- Create the event
- Pass it to the public method on the event subscriber
- Assert as necessary

Key Take Aways

Key Take Aways

- Drupal hooks are a solution for a bygone era

Key Take Aways

- Drupal hooks are a solution for a bygone era
- They worked for us, but today promote control checks that should happen at a different layer

Key Take Aways

- Drupal hooks are a solution for a bygone era
- They worked for us, but today promote control checks that should happen at a different layer
- We should adopt other available solutions in core today

Key Take Aways

- Drupal hooks are a solution for a bygone era
- They worked for us, but today promote control checks that should happen at a different layer
- We should adopt other available solutions in core today
- Profit

Questions & Answers

DrupalCon
SEATTLE 2019
APRIL 8-12

Join us for contribution opportunities

Friday, April 12, 2019

Mentored Contributions

9:00-18:00
Room: 602

First Time Contributor Workshop

9:00-12:00
Room: 606

General Contributions

9:00-18:00
Room: 6A

#DrupalContributions

What did you think?

Locate this session at the DrupalCon Seattle website:

http://seattle2019.drupal.org/schedule

Take the Survey!

https://www.surveymonkey.com/r/DrupalConSeattle