Fields in core

Drupalcon Szeged 2008

Barry Jaspan

Motivation

Motivation

- "Drupal's strategic advantage is an architecture that allows contributed modules to easily add value to content."

Motivation

- "Drupal's strategic advantage is an architecture that allows contributed modules to easily add value to content."
- The future of the web is interoperability, high-fidelity data, and web services
- i.e. The Semantic Web

Motivation

- "Drupal's strategic advantage is an architecture that allows contributed modules to easily add value to content."
- The future of the web is interoperability, high-fidelity data, and web services
  - i.e. The Semantic Web
- Drupal's content model is based on Nodes
  - Monolithic, local, and (currently) disorganized

Motivation

Motivation

- Field API addresses these issues:
  - Preserves the power and flexibility of hook_nodeapi
  - Fields are well-defined data elements, easily exported
  - Fields can be attached to any Content object, not just Nodes (phase 2)

Motivation

- Field API addresses these issues:
  - Preserves the power and flexibility of hook_nodeapi
  - Fields are well-defined data elements, easily exported
  - Fields can be attached to any Content object, not just Nodes (phase 2)
- Field API must be in core for this to work

Design goals

Design goals

- Unify hook_nodapi & CCK field data
  - Node properties: title, body
  - Core & contrib modules: taxonomy, fivestar
  - Custom CCK fields

Design goals

- Unify hook_nodapi & CCK field data
  - Node properties: title, body
  - Core & contrib modules: taxonomy, fivestar
  - Custom CCK fields
- API, not forms and UI
- Simpler and cleaner D6 CCK API

Design goals

- Unify hook_nodapi & CCK field data
  - Node properties: title, body
  - Core & contrib modules: taxonomy, fivestar
  - Custom CCK fields
- API, not forms and UI
  - Simpler and cleaner D6 CCK API
- (At least) as functional and performant as D6 CCK

Field API: Base concepts

Field API: Base concepts

- Field Type: data type implemented by a module
  - e.g. text, nodereference, address

Field API: Base concepts

- Field Type: data type implemented by a module
  - e.g. text, nodereference, address
- Field: a specific configuration of a type
  - Settings that define the unique characteristics
  - Base properties: type, name, cardinality, sharable
  - Per-type properties: Text, formatted

Field API: Base concepts

- Field Type: data type implemented by a module
  - e.g. text, nodereference, address
- Field: a specific configuration of a type
  - Settings that define the unique characteristics
  - Base properties: type, name, cardinality, sharable
  - Per-type properties: Text, formatted
- Field Instance: Field + Content Type
  - “Subtitle field on Article nodes”
  - Base properties: display format, weight, widget
  - Per-type properties: GMap, width/height, scale...

Field API: Data types

```
// Base Field data structure.
class Field {
  public $field_name;
  public $type;
  public $required = 0;
  // ... more properties here ...
}

// Additional field properties needed for text fields.
class TextField extends Field {
  // Plain text or formatted?
  public $text_processing;
  // The length property of the 'value' column.
  public $max_length;
}
```

Field API: Data types

```
// Base Field data structure
cl

}

//                                    ls.
cl

  public $max_length;

}
```

Don't Panic!

No OO code here (god forbid)!

Field API: Data types

```
// Base Field data structure.
class Field {
  public $field_name;
  public $type;
  public $required = 0;
  // ... more properties here ...
}

// Additional field properties needed for text fields.
class TextField extends Field {
  // Plain text or formatted?
  public $text_processing;
  // The length property of the 'value' column.
  public $max_length;
}
```

Field API: Data types

```
// Base Field data structure.
class Field {
  public $field_name;
  public $type;
  public $required = 0;
  // ... more properties here ...
}

// Additional field properties needed for text fields.
class TextField extends Field {
  // Plain text or formatted?
  public $text_processing;
  // The length property of the 'value' column.
  public $max_length;
}
```

Field API: Data types

```
// Base Field Instance data structure.
class FieldInstance  {
  // Public properties
  public $field_name;
  public $type_name;
  public $widget;
  // ... more properties here ...
}
// Defines a text FieldInstance.
class TextFieldInstance extends FieldInstance {
  protected $widget_class = 'TextWidgetSettings';
}
// Additional widget settings for text fields/areas
class TextWidgetSettings extends WidgetSettings {
  public $rows;
}
```

Field API: Create fields

- API Paradigm: Instantiate, Customize, Create

```
// Instantiate a TextField (in memory only)
// with default values
$subtitle = new TextField('subtitle');

// Allow users to select input format
$subtitle->text_processing = TRUE;

// Set the maximum length
$subtitle->max_length = 100;

// Create the field
field_create_field($subtitle);
```

Field API: Create instances

- Same paradigm.

```
// Instantiate a new TextFieldInstance, specifying
// field name, content type, and (because text.module
// requires it) input widget.
$instance = new TextFieldInstance('subtitle', 'article',
                                  'text_textarea');

// Set the input rows for the textarea widget.
$instance->widget->rows = 3;

// Create the field instance
field_create_instance($instance);
```

Field API: Update instances

Field API: Update instances

- Instance properties are lightweight and easily changed

```
// Retrieve the instance
$instance = field_get_instance('subtitle', 'article');

// Change something
$instance->widget->rows = 5;

// Save it
field_update_instance($instance);
```

Field API: Update instances

- Instance properties are lightweight and easily changed

```
// Retrieve the instance
$instance = field_get_instance('subtitle', 'article');

// Change something
$instance->widget->rows = 5;

// Save it
field_update_instance($instance);
```

- Fields, however, are more complicated...

Field storage: CCK today

Field storage: CCK today

- CCK fields are stored in one of three ways:

Field storage: CCK today

- CCK fields are stored in one of three ways:

	Single	Multiple
Not shared	per-type	per-field, delta
Shared	per-field	per-field, delta

  - All per-type fields for a type are stored in table `content_type_<type>`, one row/node
  - Per-field fields are stored in table `content_field_<field>`, one or more (if multiple) rows/node
- CCK converts between them willy-nilly

Field storage: CCK today

- CCK fields are stored in one of three ways:

	Single	Multiple
Not shared	per-type	per-field, delta
Shared	per-field	per-field, delta

- All per-type fields for a type are stored in table `content_type_<type>`, one row/node

Field storage: CCK today

- CCK fields are stored in one of three ways:

	Single	Multiple
Not shared	per-type	per-field, delta
Shared	per-field	per-field, delta

- All per-type fields for a type are stored in table `content_type_<type>`, one row/node
- Per-field fields are stored in table `content_field_<field>`, one or more (if multiple) rows/node

Field storage: CCK today

- CCK fields are stored in one of three ways:

	Single	Multiple
Not shared	per-type	per-field, delta
Shared	per-field	per-field, delta

  - All per-type fields for a type are stored in table `content_type_<type>`, one row/node
  - Per-field fields are stored in table `content_field_<field>`, one or more (if multiple) rows/node
- CCK converts between them willy-nilly

Field storage: Core

Field storage: Core

- Conflicting requirements:

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable
  - Field conversion is slow and currently fragile

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable
  - Field conversion is slow and currently fragile
  - Per-type storage is more efficient (?)

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable
  - Field conversion is slow and currently fragile
  - Per-type storage is more efficient (?)
  - Views query optimization

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable
  - Field conversion is slow and currently fragile
  - Per-type storage is more efficient (?)
    - Views query optimization
- Per-field storage does not solve all problems

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable
  - Field conversion is slow and currently fragile
  - Per-type storage is more efficient (?)
    - Views query optimization
- Per-field storage does not solve all problems
  - Converting text to nodereference

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable
  - Field conversion is slow and currently fragile
  - Per-type storage is more efficient (?)
    - Views query optimization
  - Per-field storage does not solve all problems
    - Converting text to nodereference
- Solution:

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable
  - Field conversion is slow and currently fragile
  - Per-type storage is more efficient (?)
    - Views query optimization
  - Per-field storage does not solve all problems
    - Converting text to nodereference
- Solution:
  - Fields retain per-type and per-field storage

Field storage: Core

- Conflicting requirements:
  - Humans cannot predict the future; fields must be changeable
  - Field conversion is slow and currently fragile
  - Per-type storage is more efficient (?)
    - Views query optimization
  - Per-field storage does not solve all problems
    - Converting text to nodereference
- Solution:
  - Fields retain per-type and per-field storage
  - Field conversions

Field API: Update field

Field API: Update field

- Updating a field is a “field conversion”

Field API: Update field

- Updating a field is a “field conversion”
- field_update_field() in Field Admin (contrib)

Field API: Update field

- Updating a field is a “field conversion”
- field_update_field() in Field Admin (contrib)
  - Updates per-type/-field storage based on cardinality and sharable (code from CCK)

Field API: Update field

- Updating a field is a “field conversion”
- field_update_field() in Field Admin (contrib)
  - Updates per-type/-field storage based on cardinality and sharable (code from CCK)
  - Invokes hook_field_convert($old, $new)

Field API: Update field

- Updating a field is a "field conversion"
- field_update_field() in Field Admin (contrib)
  - Updates per-type/-field storage based on cardinality and sharable (code from CCK)
  - Invokes hook_field_convert($old, $new)
  - A module returns TRUE if it handled the conversion

Field API: Update field

- Updating a field is a "field conversion"
- field_update_field() in Field Admin (contrib)
  - Updates per-type/-field storage based on cardinality and sharable (code from CCK)
  - Invokes hook_field_convert($old, $new)
  - A module returns TRUE if it handled the conversion
  - Usage:

```
// Load the existing field
$old = field_read_field('old_field');
// Instantiate a new field (could clone instead)
$new = new <Type>Field('new_field');
// Convert old to new (in this case, renames too)
field_update_field($old, $new);
```

Field API: Update examples

Field API: Update examples

1. Convert field from single to multiple
   - field_update_field() uses existing code

Field API: Update examples

1. Convert field from single to multiple
   - field_update_field() uses existing code
2. Convert text field from plain to formatted
   - field_update_field() calls hook_field_convert()
   - text module: add new column, return TRUE

Field API: Update examples

1. Convert field from single to multiple
   - field_update_field() uses existing code
2. Convert text field from plain to formatted
   - field_update_field() calls hook_field_convert()
   - text module: add new column, return TRUE
3. Convert text field to node reference
   - **nodereference module:** create node type, SELECT DISTINCT, new node per value, create new field and instance, UPDATE nid of new instance based on join, delete old instance and field, rename new field to old name, return TRUE.

Field API: Update examples

1. Convert field from single to multiple
   - field_update_field() uses existing code
2. Convert text field from plain to formatted
   - field_update_field() calls hook_field_convert()
   - text module: add new column, return TRUE
3. Convert text field to node reference
   - nodereference module: create node type, SELECT DISTINCT, new node per value, create new field and instance, UPDATE nid of new instance based on join, delete old instance and field, rename new field to old name, return TRUE.

- CCK can't (really) do 2 or 3 at all

Field API: Update examples

1. Convert field from single to multiple
   - field_update_field() uses existing code
2. Convert text field from plain to formatted
   - field_update_field() calls hook_field_convert()
   - text module: add new column, return TRUE
3. Convert text field to node reference
   - nodereference module: create node type, SELECT DISTINCT, new node per value, create new field and instance, UPDATE nid of new instance based on join, delete old instance and field, rename new field to old name, return TRUE.

- CCK can't (really) do 2 or 3 at all
- All require Batch API during page request

Next steps & how to help

Next steps & how to help

- Update code to CCK HEAD
  - SVN: https://jaspan.devguard.com/svn/fields
  - Apparently some patches need to be ported...

Next steps & how to help

- Update code to CCK HEAD
  - SVN: https://jaspan.devguard.com/svn/fields
  - Apparently some patches need to be ported...
- Port field modules to new API
  - text.module is done as example

Next steps & how to help

- Update code to CCK HEAD
  - SVN: https://jaspan.devguard.com/svn/fields
  - Apparently some patches need to be ported...
- Port field modules to new API
  - text.module is done as example
- Port CCK UI to Field API
  - Staying in contrib

Next steps & how to help

- Update code to CCK HEAD
  - SVN: https://jaspan.devguard.com/svn/fields
  - Apparently some patches need to be ported...
- Port field modules to new API
  - text.module is done as example
- Port CCK UI to Field API
  - Staying in contrib
- Write field conversions
  - Be the DabbleDB-killing hero!

Next steps & how to help

- Update code to CCK HEAD
  - SVN: https://jaspan.devguard.com/svn/fields
  - Apparently some patches need to be ported...
- Port field modules to new API
  - text.module is done as example
- Port CCK UI to Field API
  - Staying in contrib
- Write field conversions
  - Be the DabbleDB-killing hero!
- Dries *really* wants all this in D7

Questions?

Barry Jaspan
barry.jaspan@acquia.com