Multi-Step Forms with CTools

Capital Camp and Gov Days 2014

Who Are We?

Jeremy Gottwig
@jgottwig
Senior Developer
RepEquity

Miro Scarfiotti
@smiro2000
Senior Developer
RepEquity

Downloads

Handout Available at:

- http://goo.gl/5ozQfi

Example Module available at:

- https://github.com/jgottwig/ctools-wizard-example

The Goal

- Create multi-page forms
- In Drupal (not using a 3rd party service)
- Without using Webform
- That maintains data between pages
- And allows us to process each page as it's submitted using data from current and previous pages

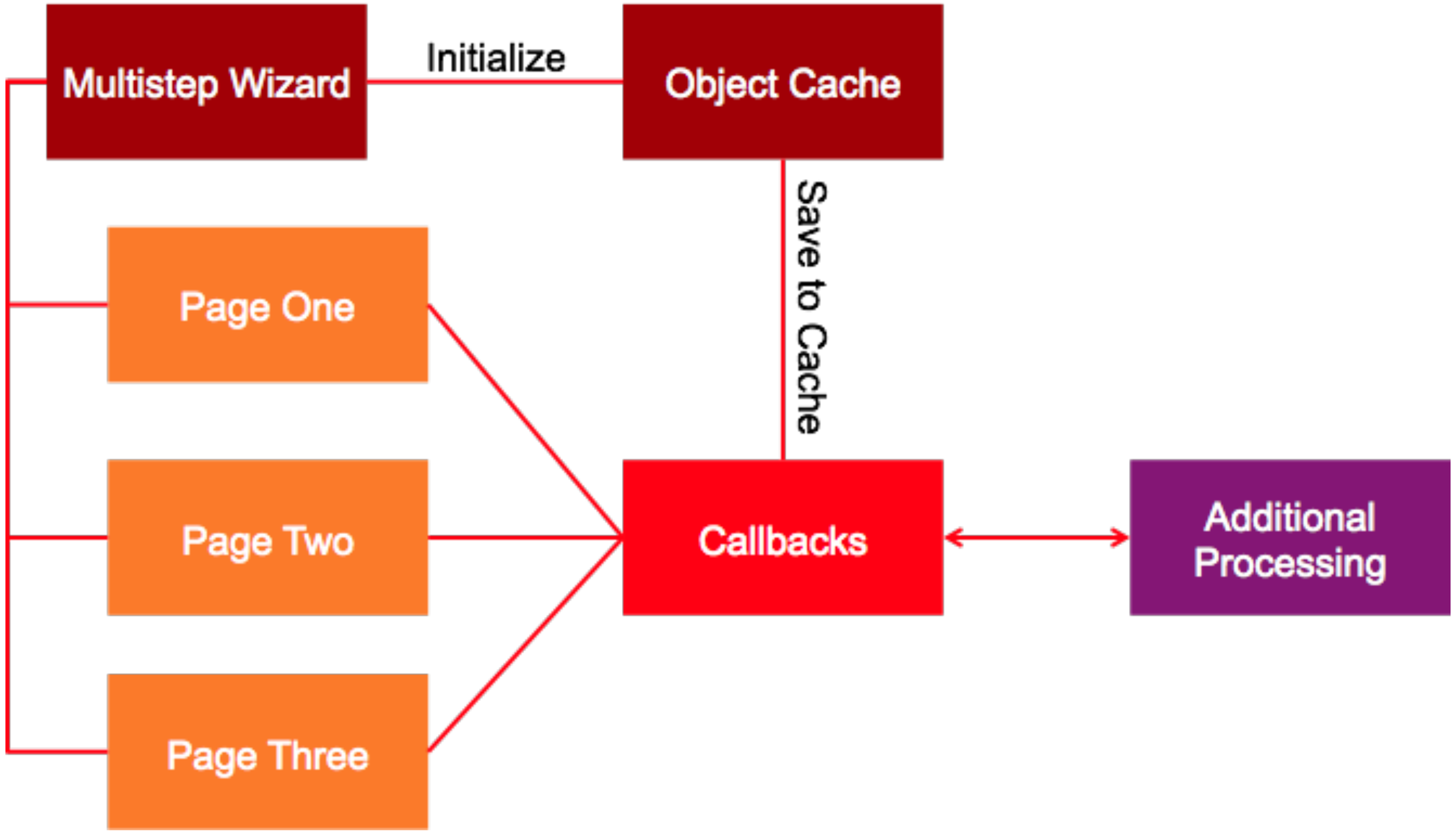
The Elephant in the Room

- Why not just use Webform?
  - Webform is great for data collection forms
  - Creates multi-step forms out of the box
  - Supports many field types and has many contrib modules to extend it
- Webform is not build for advanced user driven data processing
  - Geolocation api calls to process zip codes
  - SOLR queries to compare user input to indexed content and return an appropriate response
  - Generate PDF’s, zip them and email them to user

Our Toolkit

- Form API in all its glory
- Ctools
  - Multistep Wizard (This is just what ctools calls its multi step form processor)
  - Object Cache (This is where ctools stores form data between steps. It's still up to the developer to pass information to the object cache in the form callbacks.)
- Additional project-specific tools (more on how these fit in later)
  - Solr
  - Batch API
  - PDF generation tools
  - Zip file builders

Understanding the Pieces



Understanding the Wizard

- The heart of your form:
  - ctools_wizard_multistep_form($form_info, $step, $form_state);
- Three arguments:
  - $form_info - This defines our workflow, options, and callbacks
    - If we click the ‘Next’ button, what happens?
    - What are our page steps and how are they ordered?
  - $step - Where we are in the flow
  - $form_state - Stores the object cache along with the object ID, and passes it to each individual page’s $form_state
    - This ensures that previous page submission data is always available

The Setup

- We’re working with two CTools includes
- We will also want to define our object ID for cache
  - Note that may be easier to do this as a constant
  - We will need it throughout the module

```
function example_form_builder($step = NULL) {
  ctools_include(‘wizard’);
  ctools_include(‘object-cache’);
  $object_id = 1;
```

ctools_wizard_multistep_form($form_info, $step, $form_state);

- $form_info contains all information the CTools wizard needs to understand and process the form
- Here are some basic options (there are more)

```
$form_info = array(
  ‘id’ => ‘example_form’,
  ‘path’ => ‘example/%step’,
  ‘show cancel’ => FALSE,
  ‘next callback’ => ‘example_builder_next’,
  ‘finish callback’ => ‘example_builder_finish’,
  ‘next text’ => t(‘Next Step’),
);
```

ctools_wizard_multistep_form($form_info, $step, $form_state);

- The keys of this array dual but related purposes
  - Your steps (as seen in $step)
  - Your page as seen by the user ‘path’ => ‘example/%step’,

```
$form_info[‘order’] => array(
    ‘step-1’ => t(‘Step One’),
    ‘step-2’ => t(‘Step Two’),
    ‘step-3’ => t(‘Step Three’),
    // and so on...
  );
```

ctools_wizard_multistep_form($form_info, $step, $form_state);

- Details for all forms using your $form_info[‘order’] keys
    - You don’t need to use includes (but they do make things easier)
    - Each form is standard Form API using all available callbacks

```
$form_info[‘forms’] = array(
    ‘step-1’ => array(
      ‘form id’ => ‘example_step_one_form’,
      ‘include’ => ‘path_to_form_include’,
    ),
     // Rinse and repeat for each additional step
```

ctools_wizard_multistep_form($form_info, $step, $form_state);

- Remember this?
  - function example_form_builder($step = NULL)
  - We need a behavior in case $step does, in fact, equal NULL

```
if (empty($step)) {
  $step = 'step-1';
}
// Not much to see here
```

ctools_wizard_multistep_form($form_info, $step, $form_state);

- CTools Wizard will add this to the $form_state of all form steps
- Includes our storage $object
- And our Object-Cache ID ($object_id)

```
$object = _example_get_obj_cache($object_id);

$form_state = array(
  ‘object_id’ => $object_id,
  ‘object’ => &$object,
);
```

ctools_wizard_multistep_form($form_info, $step, $form_state);

- $object_id instructs CTools how to find the storage object in the cache table
- It prevents collisions in tools with lots of possible storage objects
  - E.g., Views and Panels

```
$object = _example_get_obj_cache($object_id);

$form_state = array(
  'object_id' => $object_id,
  'object' => &$object,
);
```

All Done!

- ctools_wizard_multistep_form($form_info, $step, $form_state);
  - Hopefully $form_info makes sense at this point
  - Ditto for $step
  - But $form_state?
    - We should get deeper into the whole $object piece

```
$object = _example_get_obj_cache($object_id);

$form_state = array(
  ‘object_id’ => $object_id,
  ‘object’ => &$object,
);
```

Introducing CTools Object-Cache

- Injecting $object into $form_state gives us a place to store form content across forms
- The wizard will see to it that all our forms get this
- ctools_object_cache_get stores data in memory
  - You can pass an option to skip memory cache

```
$object = _example_get_obj_cache($object_id);

function _example_get_obj_cache($object_id) {
  ctools_include(‘object-cache’);
  if (!$cache = ctools_object_cache_get(‘example’, $object_id)) {
    $cache = new stdClass();
    $cache->locked = ctools_object_cache_test(‘example’, $object_id);
  }
}
```

Updating the Object Cache

- Remember this from our $form_info?
  - ‘next callback’ => ‘_example_builder_next’,

```
function _example_builder_next(&$form_state) {
  _example_update_obj_cache(OBJECT_ID, $form_state[‘object’]);
}

function _example_update_obj_cache($id, $content) {
  ctools_include(‘object-cache’);
  $cache = ctools_object_cache_set(‘example’, $id, $content);
}
```

Shutting Things Down

- Remember this from our $form_info?
  - ‘finish callback’ => ‘_example_builder_finish’,

```
function _example_builder_finish() {
  _example_clear_obj_cache(OBJECT_ID);
  // Anything else?  Want to redirect to a thank you page?
}

function _example_clear_obj_cache($id) {
  ctools_include(‘object-cache’);
  ctools_object_cache_clear(‘example, $id);
}
```

Caveats and Special Cases from our Implementation

- In our application, we had to fire _example_builder_finish from outside of $form_info
  - A Batch API at the end of our process derailed things and dropped us out of the ctools wizard
    - We called the _example_builder_finish from Batch API's own finish function
  - You might find that other tools can derail things as well
- Maybe something about Ajax and our Select Meds here?
- This might be a no-duh for some, but additional functions in our various form .inc files are not available to other form pages
  - Maybe place these in your .module
  - For example, we had Solr calls that we sometimes needed one one step, sometimes a later step

Special Thanks to...

- Abby Milberg, designer extraordinaire
  - @AbbyMilberg

Thank You

REPEQUITY®

www.RepEquity.com
@RepEquity