Enrollment Flow Plugins (PE)

Background

Enrollment Flow Plugins implement functionality for Enrollment Flows.

Entry Point Model

Each Entry Point Model for Enrollment Flow Plugins will be made available when a new Enrollment Flow Step is defined. The corresponding controller should extend `StandardEnrollerController`, and implement an edit-only `fields.inc`.

Plugins can be instantiated multiple times in the same Enrollment Flow (as multiple Steps).

The Entry Point Model's *Controller* is expected to implement two functions: `dispatch()` and `display()` (described below). The model should set permissions for both of these functions to `true`, as detailed permission calculation will be handled by `StandardEnrollerController`.

In addition, the Entry Point Model may optionally implement a `finalize()` call, as described below.

```
public function initialize(array $config): void {
  ...

  $this->setAllowLookupPrimaryLink(['dispatch', 'display']);

  $this->setPermissions([
    'entity' => [
      'delete'   => false, // delete the parent object instead
      'dispatch' => true,  // StandardEnrollerController will handle this
      'display'  => true,  // StandardEnrollerController will handle this
      'edit' =>     ['platformAdmin', 'coAdmin'],
      'view' =>     ['platformAdmin', 'coAdmin']
    ],
    'table' => [
      'add' =>      false, // This is added by the parent model
      'index' =>    ['platformAdmin', 'coAdmin']
    ],
    ...
  ]);
}
```

`dispatch()`

When an Enrollment Flow Step executes, `StandardEnrollerController` will perform authorization and validation checks before passing through to the Plugin's `dispatch` function. The function will be passed the instantiated plugin ID. A utility function is available to obtain the current Petition.

Plugins should process all actions through `dispatch` to avoid complications with calculating permissions. While ordinarily the request type should be sufficient to distinguish actions (eg: `GET` to render a form, `POST` to process it), plugins can also insert flags into the request URL or form data to track state (eg: `/dispatch/2?petition_id=18&action=verify`).

Views required by `dispatch()` should be generated by creating a view template `dispatch.inc`. Doing so will leverage standard infrastructure to create a form and insert Petition and (if applicable) Token information so that the Plugin does not need to worry about carrying this metadata across the form submission. The view variable `$vv_petition` will have information about the current Petition.

Plugins should *not* update operational records as part of `dispatch()`. Instead, state should be saved in Plugin-specific tables. See `finalize()` below for more information.

⚠ An Enrollment Flow Step can be rerun if the Petition is not yet considered complete. Plugins should present already submitted data, or otherwise behave in a manor that permits the Actor to change their previous decisions.

```
// In the Plugin Entry Point Model's Controller

public function dispatch(string $id) {
  $petition = $this->getPetition();

  if($this->request->is(['post', 'put']) {
    try {
      // Back from the form, do something with the data
      $data = $this->request->getData();

      // On success, indicate the step is completed and generate a redirect to the next step

      $link = $this->getPrimaryLink(true);

      return $this->finishStep(
        enrollmentFlowStepId: $link->value,
        petitionId:           $petition->id,
        // This comment will be stored in the PetitionStepResult artifact
        comment:              __d('widget_enroller', 'result.widget.saved')
      );
    }
    catch(\Exception $e) {
       $this->Flash->error($e->getMessage());
    }
  }

  // Let the form render (for GETs and failed POSTs)
  $this->render('/Standard/dispatch');
}
```

display()

When a Petition is rendered, each configured Step will be given an opportunity to render Step-specific information.

XXX

finalize()

Plugins are expected *not* to touch operational data during Petition execution. For example, in a new Enrollee signup Flow, the new Person record will *not* be created until the Enrollment Flow `finalize` step runs. During `dispatch()`, Plugins should only write to their own specific tables.

When the Petition is finalized, each Plugin will be given the opportunity to update the operational record based on the state it maintained during the Petition. `finalize()` will be called for each Plugin in the same order as the original Enrollment Flow Step execution. If the Plugin is instantiated in multiple Steps, it will be called once for each Step.

Any work performed by `finalize()` in the Plugin should be quickly, as all calls to all Plugins defined in the Enrollment Flow must be completed during a single web browser request, and the longer this request takes the more likely the user is to prematurely terminate the page or for the browser to time out. For example, calls to external systems via REST APIs should *not* be performed in `finalize()`.

Plugins cannot terminate the finalization process. If something goes wrong, an error should be logged, an appropriate Petition History Record (and Person History Record, if appropriate) should be created, and the Plugin should fail gracefully.

If the Plugin does not implement `finalize()`, then the Plugin is expected not to perform any finalization actions.

```
// In the Entry Point Model

use Cake\ORM\TableRegistry;

public function finalize(int $id, int $petitionId): bool {
  // Pull our configuration
  $widgetEnroller = $this->get($id);

  // And find the subject Person from the Petition
  $Petitions = TableRegistry::getTableLocator()->get('Petitions');

  $petition = $Petitions->get($petitionId, ['contain' => 'EnrolleePeople']);

  return true;
}
```

See Also

- Writing Registry PE Plugins
- Registry PE Enrollment Flows
- Registry PE TID: Enrollment Flows and Petitions (revision 3)