# Concept Multi-step forms

Your Rating: Results: 92 rates

**Implemented in 4.4 Form 1.2**

**Official Documentation Available**

This topic is now covered in Creating a mulitstep form.

## Introduction

This page describes the changes to the form module to add support for multi step forms.

The goal is to add multi step support to the existing form module. The good user experience of the module should remain, that is, forms should be buildable by editors using paragraphs. Since thihs makes is easy for editors to mix form elements with editorial content (text, images, instructions and so on). Each step is configured on a page of its own.

## Update Team Meeting 2010-10-11

We're in favor of doing a subpage oriented navigation, see alternative 1 below. The reasoning is:

- Backwards compatibility with existing configuration
- Less changes
- Intuitive to use

## Requirements

- Support the back button
- Span over multiple pages, each page is validated before the user can proceed to the next step/page
- ~~Must support multiple forms on the same page (at least multiple first steps on the same page).~~
- Must support filling in the same form simultaneously in multiple tabs.

- Refreshing the page while it is showing error messages should return it with the same error messages (error messages need to be in the server state).
- Refreshing the page should render it with the previous values filled in.

- Since a user can move from a step that has errors (kept on server side too) to another step and do submit there validation must not fail due to errors on other steps. So, things in the server side state needs to be associated with the step they come from.

## Server-side state

When the form is rendered for the first time (step 1) it generates a unique token, the formStateToken. The token is passed from step to step and is used to identify the state on the server that has been collected for this form. The token is not stored in the session since we want support for multiple tabs.

The actual state is stored in the session keyed using the token.

**Definition**: formStateToken a unique id passed from step to step that identifies the server side state

**Definition**: formState the server side state kept in session keyed using the formStateToken

## Navigation

Supporting the back button means we do POST-REDIRECT-GET to avoid resubmission and browser warnings about POST data.

- When validation succeeds the user is redirected to the next step.
- If the user navigates to a page that has step 2 of a form then the form needs to send him back to the first step. Or at least instruct him to go there instead of showing the form.
    - Except when the author is creating the form..

### Alternative 1 - Subpage based navigation (this is the one we decided for)

The next step is found by looking at subpages or siblings depending on if your on the first page or not.

The first step is always the parent node.

The processors are configured on the paragraph on the first step.

### Alternative 2 - Connected pages based navigation

The pages are connected with a uuidLink called nextStep, when validation succeeds the user is redirected to the next step.

The first step is configured using a uuidLink called firstStep.

The same paragraph is used on all steps, processors are added as paragraphs to the last step. In a special region/paragraph-system. These paragraphs will have no output except in editing mode where they will display instructions and info.

## Form processing

Form processing is what happens when the last step is submitted. The collected data kept in formState is passed to a number of processors.

## Algorithm

This is all done in FormEngine and its subclasses

### When form is to be rendered (GET request)

- If there is a formStateToken and there is formState in session for it
    - If this step hasnt been rendered before
        - place default values for this step in formState
    - render page

- If there isnt a formStateToken as a request parameter and this is not the first step (not done in author instances)
    - render page with a message that the form starts on another page, include a link to the first step

- If there isnt a formStateToken as a request parameter and this is the first step
    - create a fresh token and place a new formState in session
    - place default values for this step in formState
    - render page

- If there is a formStateToken but there is no formState in the session for this token
    - render the page with a message about the session having been closed and display a link to the first step

### When form is submitted (POST request using pre-execution)

- If there isnt a formStateToken as a request parameter
    - redirect to the first step with an error message (should never happen)

- If there is no formState in session for this token
    - redirect to the first step with a message about the session having been closed and display a link to the first step

- Loop sub paragraphs, collect submitted values and do validation
- Store the collected values and their error messages in formState (as a group keyed by page uuid)
- If there were validation errors
    - redirect to this step
- If there is another step
    - redirect to the next step
- Execute processors
- If a processor failed
    - place an error message about this (message supplied by processor)in formState
    - redirect to this step
- Remove formState from session
- If succesPage configured
    - redirect to successPage
- redirect to this page with a success message

## Security concerns

Input accepted via the form must be encoded before displayed to avoid cross site scripting attacks

## Problems and weak points

Some things to think about...

### Navigation

- When the user goes back to the first step and presses refresh he loses his server side state since the formStateToken is lost and a new one is generated.
- The user can skip steps by manipulating the url and there's no way to detect it since form configuration is split across pages.
- The user can manipulate the url to go from one multi step form into another, effectively combining two multi step forms.

## File upload

The form module supports files that are sent along as attachments in form processors. The addition of steps means that files uploaded in a previous step needs to be stored somewhere while the user fills in the current step. The file should also be deleted when the form completes. If the user never completes the form the file needs to be automatically cleaned up.

If magnolia is running in a clustered environment where sessions are shared among instances extra care must be taken so that all instances have access to the files. Possibly on network storage, the location would need to be configurable.

**As a first step we can add multi step support but support files only on the last step**

### Security concerns

- Files uploaded must not be publicly accessible.
- Files must have a upper limit to avoid a malicious user filling up the disk.
- Files need to be removed as early as possible to avoid the disk filling up. Ideally the form can be considered abandoned before the users session ends.
- When a user resubmits a step the file previously uploaded should be deleted.