Anatomía de una petición de formulario Drupal

Ricardo Sanz Ante

drupalcampSpain 2013 CACERES

Formularios

- Por formulario entiendemos una abstracción que usa Drupal para gestionar los formularios HTML

- Esta abstracción es capaz de definir estructuras que luego son renderizadas a HTML

Formularios

- Gestiona los envíos, persistencia, construcción y renderizado
- Ofrece una API para crear y modificar el comportamiento
- Gestión peticiones AJAX

Formularios

- form contiene la estructura del formulario y comportamiento
- form_state contiene el estado del formulario:
  - Inputs
  - Flags e info del form (estado)
  - Afecta a la cosntrucción del formulario en todas sus fases
- form_build_id y form_id

Petición de formulario

- Petición normal a Drupal que llama a drupal_get_form
- Petición AJAX
- Formularios ejecutados internamente (drupal_form_submit)

Primera petición HTTP

- El usuario accede a una URL
- Drupal ejecuta callback
- En algún momento se llama drupal_get_form
- Se inicia el procesado del form
- Se entregará un formulario 'limpio' recién construido

Segunda petición HTTP

- El usuario envía el formulario
- Drupal ejecuta callback
- En algún momento se llama drupal_get_form
- Se inicia el procesado del form
- Se ejecutan las funciones validate y en su caso las submit

Segunda petición: Respuesta

- Posibles respuestas
  - Propia de los submits
  - Redirección
  - Versión reconstruida del form
  - Formulario con errores de validación marcados

Petición HTTP vía AJAX

- Posibles respuestas
  - Fragmento del formulario (o completo)
  - AJAX Commands

Fases

- Preparación del form
- Construcción del form
- Validación
- Submit
- Reconstrucción
- Renderizado

Preparación 1/5

- Obtiene el structured array representando el form
- Los elementos están 'limpios', sin procesar
- Son las instrucciones para que Drupal construya el formulario
- Solo se ejecuta si el form NO está en la caché

Preparación 2/5

- Se llama a la función constructora
- Se usa hook_form para localizarla:
  - Mapea uno más form id a su función constructora
- O si no se usa su form id

Preparación 3/5

- Se establece el troken de seguridad
- Se establece el form_build_id
- Se ejecutan los alters sobre el form:
  - form
  - form_BASE_FORM_ID
  - form_FORM_ID

metadrop

drupalcampSpain 2013 CACERES

Preparación 4/5

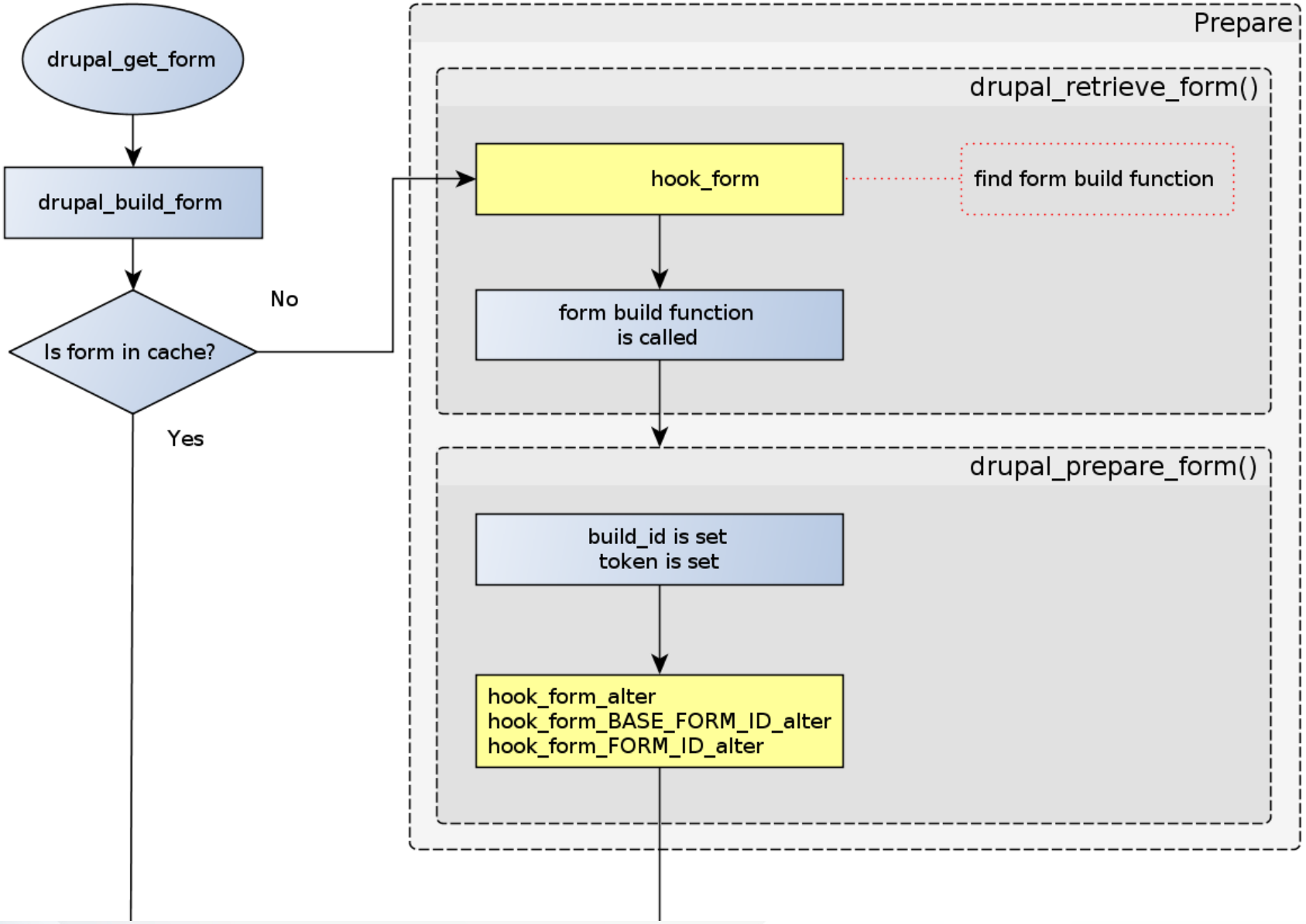
- La salida de esta fase es un structured form array

- Esta es la versión del form que se guarda en caché (si se debe guardar), y lógicamente la versión que se recuperará de caché

Preparación 5/5

- En esta fase se debe modificar la estrucutura general del form: cambios estructurales del form (paso de multistep, número de elementos colección, etc)

- Cuando el form se obtiene de caché no se ejectua, excepto si después de reconstruye el form



Construcción 1/5

- Obtiene un array de formulario renderizable
- Los elementos del árbol del form se procesan recursivamente
- Se fija el #value a los elementos input
- Se ejecutan las funciones #process (top-down)
- Se ejecutan las funciones #after_build (down-top)

Construcción 2/5

- Las funciones #process preparan el elemento
- Pueden crear más elementos hijos (p. Ej: checkboxes, date)
- Activar propiedades del form (#cache)
- Comienza en el elemento raíz y termina en el hijo más profundo
- Normalmente solo uso avanzado

Construcción 3/5

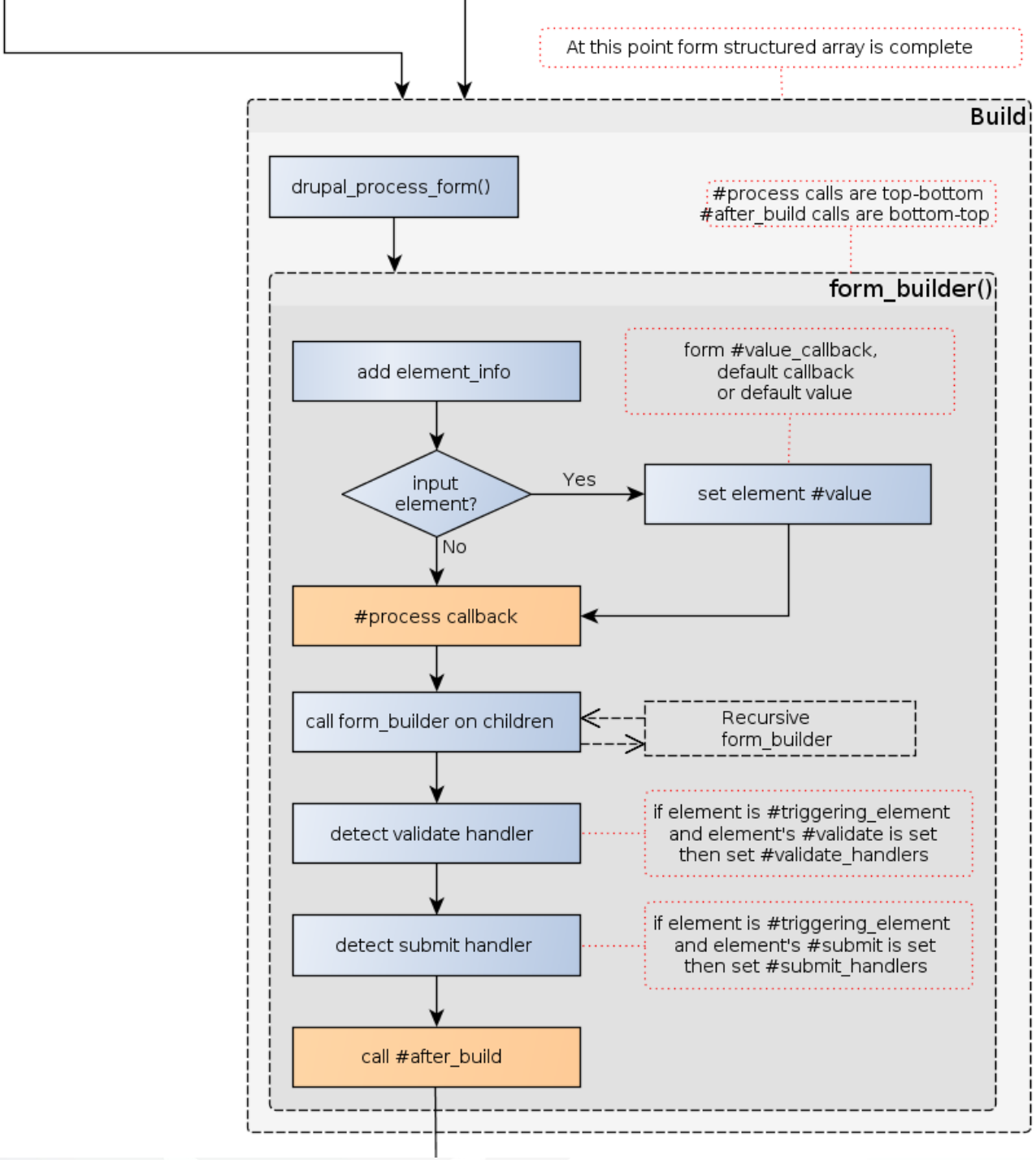
- Las funciones #after_build permiten modificar el elemento tras procesarse el input
- Normalmente solo uso avanzado
- Al ejecutarse siempre pueden usarse para añadir CSS y JS condicionalmente

Construcción 4/5

- Esta fase también detecta los handlers validate y submit
- Si un botón específica handlers validate o submit se marcan estos para ser usados
- form_state['validate_handlers']
- form_state['submit_handlers']
- Si no hay, se usan los del elemento form raíz

Construcción 5/5

- Fase avanzada, al desarrollar solo se usará esta fase en formularios avanzados



Validate 1/3

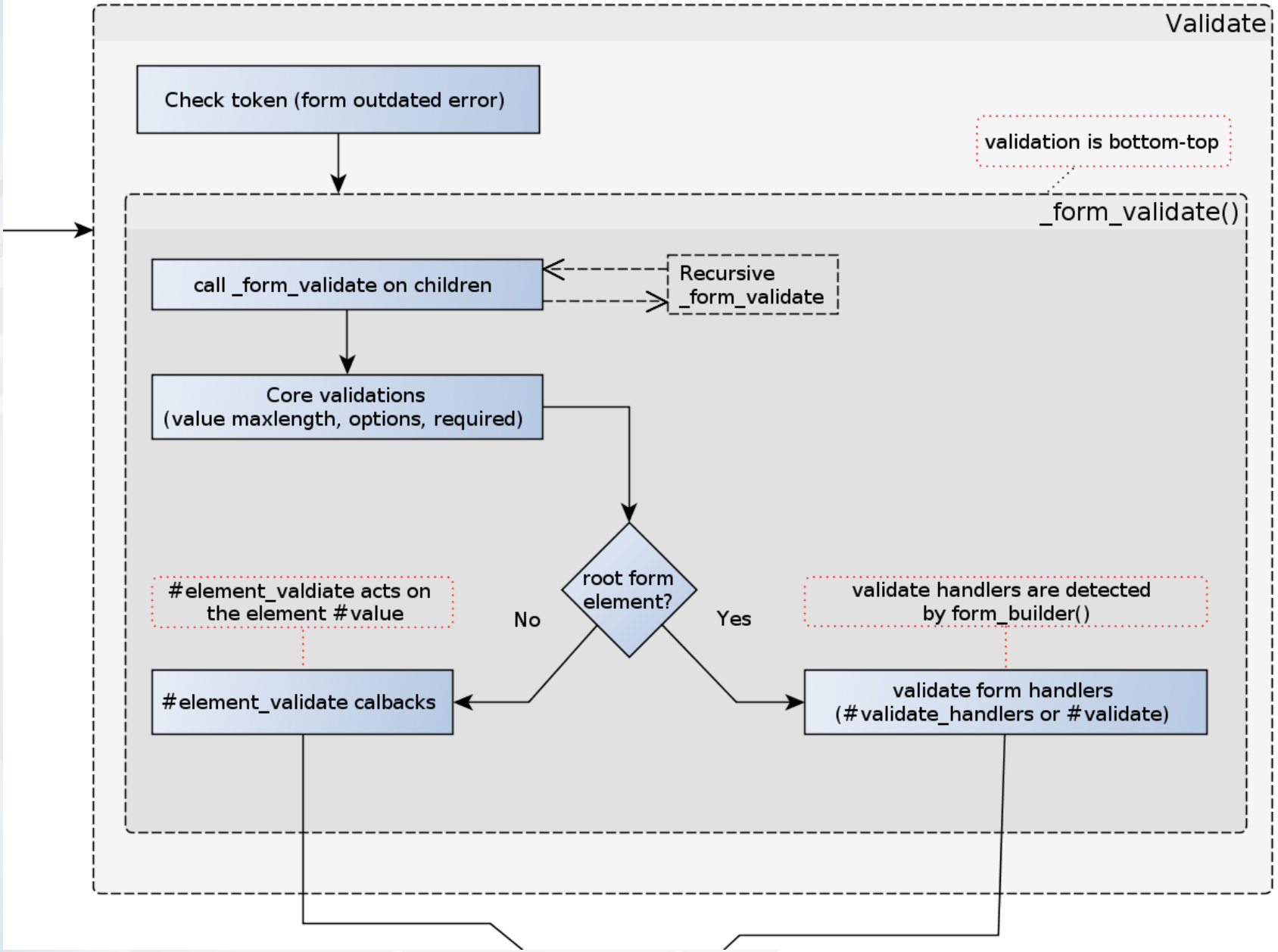
- Validación del formulario: seguridad y lógica
- Solo se ejecuta si hay entrada y coincide el form_id del form con el form_id enviado por el usuario
- Comprueba el token de seguridad
- Se recorre el form recursivamente de abajo a arriba

Validate 2/3

- Core validations (required, options, value max length)
- Se ejecuta #element_validate (si hay)
- Al llegar al elemento form raíz se ejecuta el validate correspondiente
- Los errores se marcan con form_set_error()

Validate 3/3

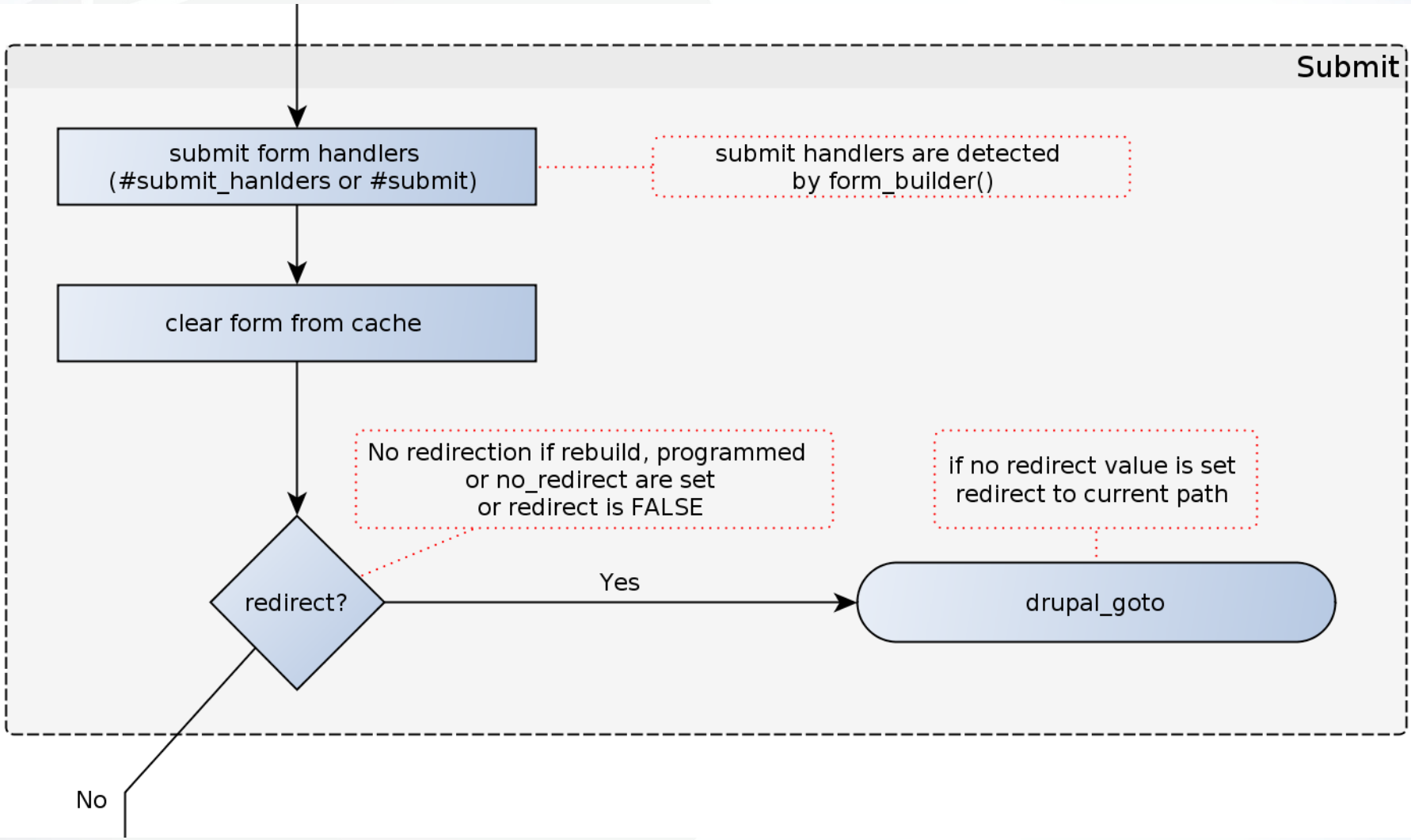
- Si el formulario valida, se pasa a ejecutar sus funciones submit
- Si no valida, el formulario se devuelve con los errores marcados
- Un error de validación impide reconstruir el formulario
- limit_validation_errors: indica qué elementos deben validarse (validaciones parciales para multistep y similares)



Submit

- Ejecuta los submit handlers (del botón pulsado, o del form)
- Se borra el form de la caché si estuviese
- Si se indica, se redirige al usuario (ejecución correcta, proceso de formulario terminado)
- Si no, se reconstruye (proceso parcial, seguimos en el mismo form)

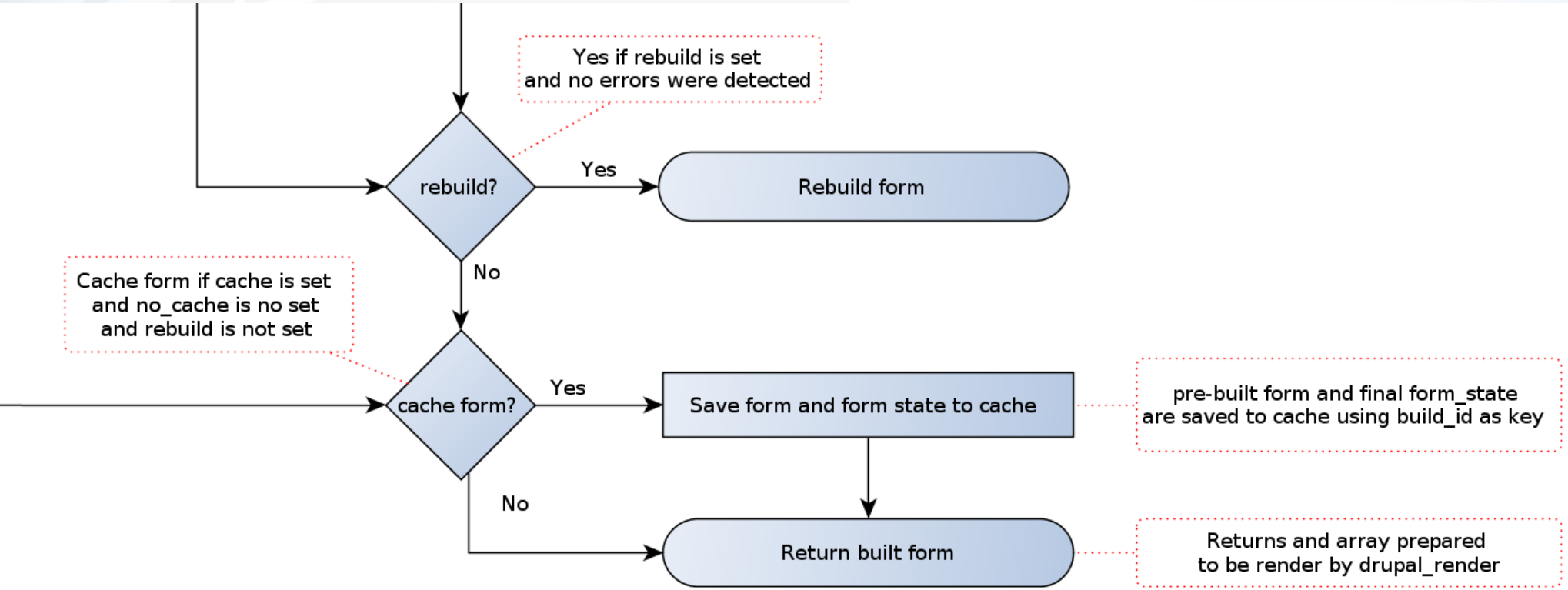
Diagrama submit



Fase final

- Se decide si se reconstruye el form
- Se devuelve el form construido
- Se decide si se cachea, guardando el form_state actual y el form pre-built

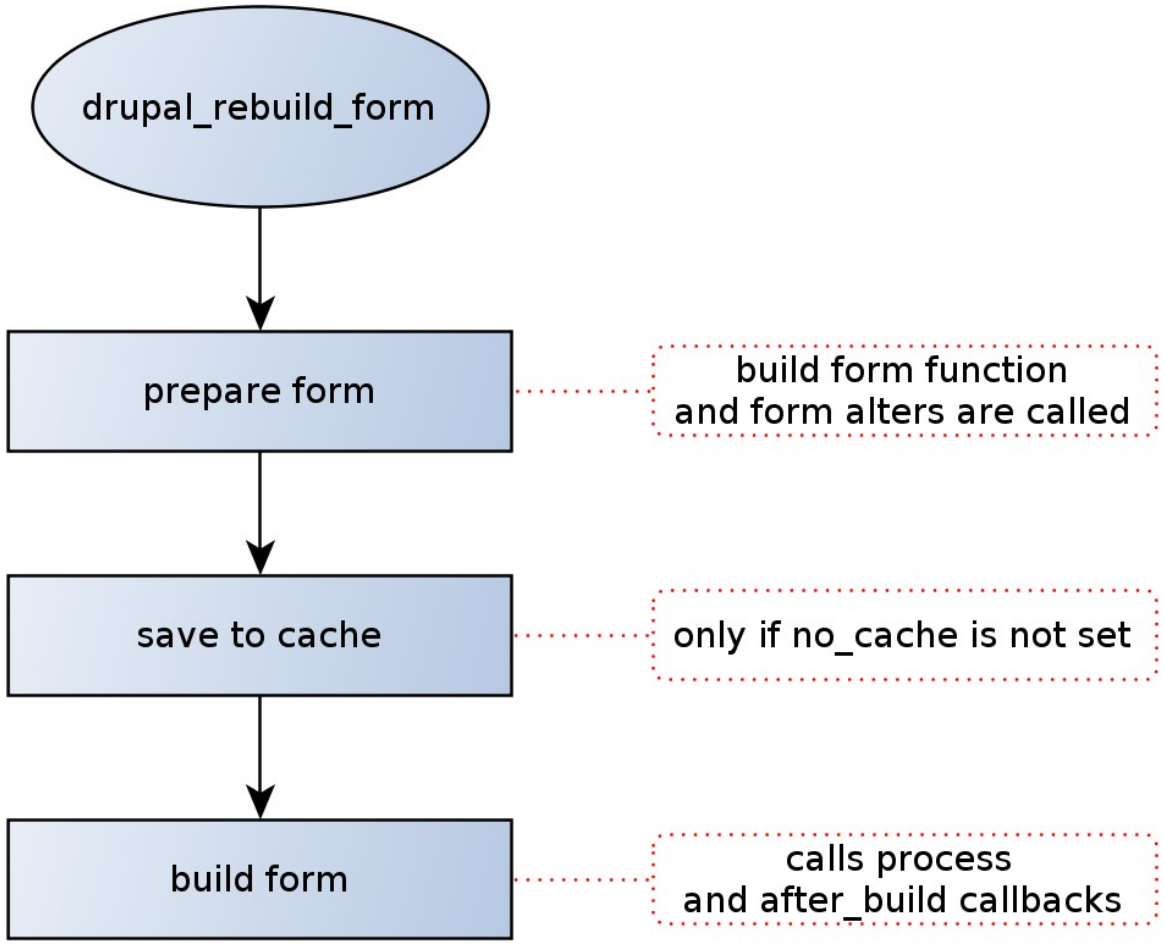
Fase final



Rebuild

- En alguna fase se marca rebuild (fs)
- Se vuelve (en la misma petición HTTP) a la fase de preparación y construcción, pero con el form_state actual
- Se devuelve una versión modificada (reconstruida) del form
- No se reconstruye si falla el validate
- Es lícito haber modificado values del form state

Rebuild



Render

- La capa de render recibe el form

- Aún es posible hacer modificaciones mediante:
  - Función #pre_render
  - Función #post_render

Caché

- Se podría llamar persistencia entre peticiones HTTP
- Permite guardar el form y sobre todo el form_state usando form_build_id
- Las peticiones AJAX SIEMPRE la activan

Petición AJAX 1/3

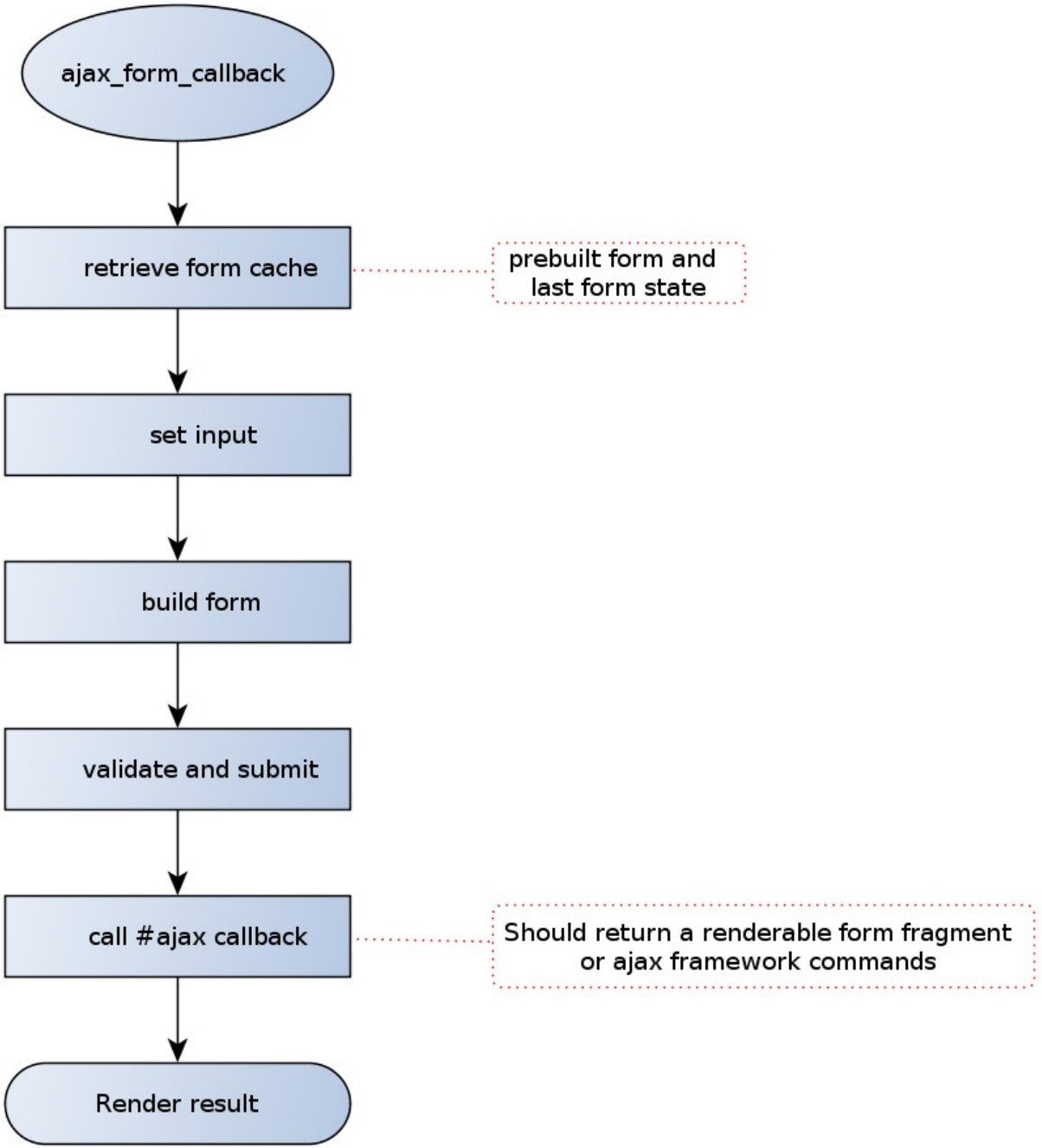
- Usa parte del camino de la petición normal
- Carga el pre-built form y el form state de caché
- Se construye el form
- Hasta aquí se ha rehecho el form para tenerlo de base para ejecutar los handlers

Petición AJAX 2/3

- Se ejecuta el validate y el submit, si procede
- Si se debe reconstruir, se suele fijar aquí el rebuild
- Se ejecuta el calback #ajax del elemento
- Se devuelve el resultado

Petición AJAX 3/3

- Uso típico: submit modifica form_state y activa rebuild
- Se reconstuye el form
- La función callback se encarga de calcular lo que se devolverá
- Podrá ser un fragmento de array renderizable del form
- O AJAX Commands



Diseñar comportamiento form

- Es MUY importante tener claro el esquema para realizar formularios complejos
- Es MUY importante saber dónde y qué tocar
- CTools ofrece facilidades para formularios multi-step
- El código está muy documentado y podréis ver los detalles

Puntos de enganche

- Form alters
- form_builder (#value_callback, #process, #after_build)
- _drupal_validate (#element_validate)
- drupal_render

Drupal 8

- Cada formulario puede ser una clase
- La clase encapsula los métodos de contrucción, validación y submit
- El workflow general es muy similar a Drupal 7