The Search API in Drupal 8

Thomas Seidl (drunken monkey)

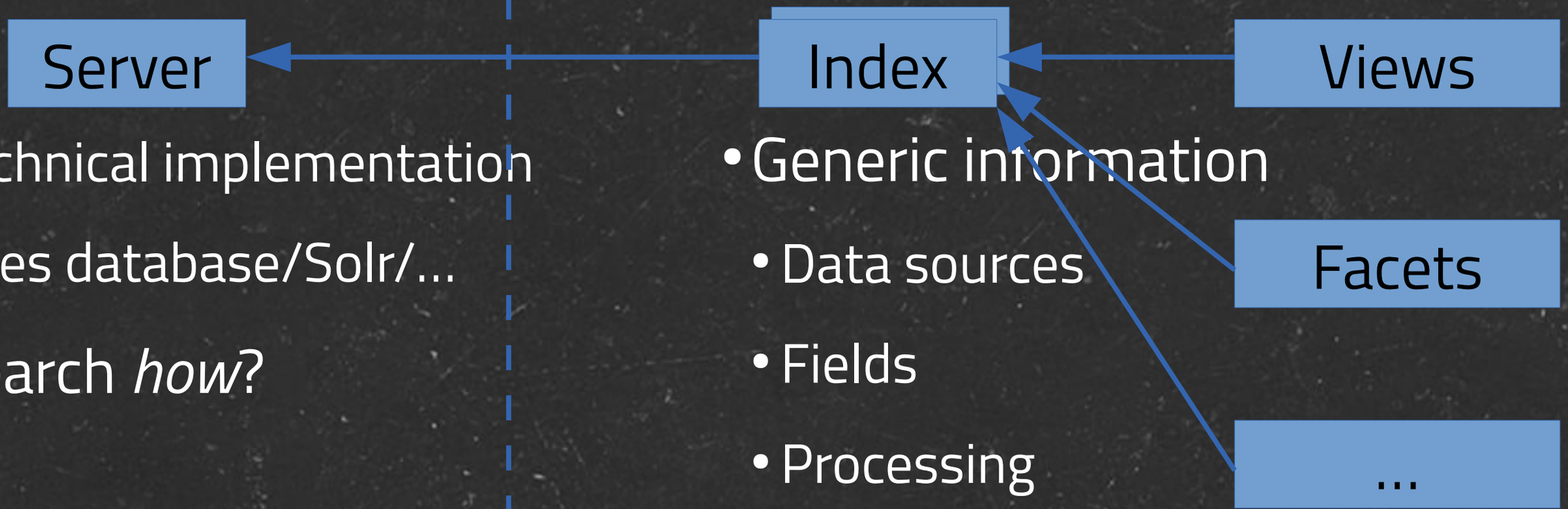
#drupaldevdays

Disclaimer

Everything shown here is still a work in progress.

Details might change until 8.0 release.

Basic architecture



- Technical implementation
- Uses database/Solr/...
- Search *how*?

- Generic information
- Data sources
- Fields
- Processing
- Search *what*?

Server

- Configuration entity
- Uses „backend“ plugin for operations
- Stores plugin and specific settings

Backend

- Plugin for servers
- Implementation of server operations
  - Indexing, deleting, searching
  - Reaction to changed indexes
  - Custom configuration form
- Connects to storage (DB, Solr, …)

Index

- Configuration entity
- Main connection to other modules
- Plugins:
  - „Datasources“ provide data
  - „Tracker“ tracks indexed data
  - „Processors“ alter data/workflow

Datasource

- Index plugin providing a specific kind of data
  - E.g., nodes, comments, external data
- Loading, viewing, metadata for types
- Datasource-specific configuration

Tracker

- Index plugin tracking state of items
- Reacts to new/changed/deleted items
- Which items still need to be indexed?

Processor

- Index plugin changing data/workflow
- Alter indexed items and search queries
- Configurable
- (Fused with D7 „data alterations“)

Indexing

Index	Tracker	Datasources	Processors
5. Extracts fields 7. Sends items to server	1. Determines changed items 10. Marks items as indexed	2. Load items 3. Provide metadata	4. Alter metadata 6. Preprocess fields / items

Server
8. Indexes items 9. Returns indexed items

Searching

Search	Query	Processors	Server
1. Creates query	3. Parses keywords	6. Preprocess query	7. Retrieves results
2. Adds keywords		8. Postprocess search results	
4. Adds filters, sort, etc.			
5. Executes query			
9. Displays results			

Customizations

Backend plugin

- Integrate Search API with new engines

```
/**
 * @SearchApiBackend(
 *   id = "MODULE_my_service",
 *   label = @Translation("My backend"),
 *   description = @Translation("Really cool!")
 * )
 */
class MyService extends BackendPluginBase {
  function indexItems($index, $items) {}
  function deleteItems($index, $ids) {}
  function deleteAllIndexItems($index) {}
  function search($query) {}
}
```

Backend plugin

- Also available:
  - Configuration form
  - React to new/changed/removed indexes
  - CRUD „hooks“
  - `supportsFeature()`, `supportsDatatype()`

Features

- Add backend-specific functionality
- Defined by contrib modules
- Using modules check for support
- Creation: No code, just documentation
- E.g.: facets, MLT, autocomplete, spellcheck

Data type

- Let backends support non-default data types
  - E.g., location coordinates, special text formats
- Backends state support similar to features
- Provided as plugins
- Documentation important

Datasource

- Support for custom item types

```
/**
 * @SearchApiDatasource(
 *   id = "MODULE_my_datasource",
 *   name = @Translation("My datasource"),
 *   description = @Translation("My great type.")
 * )
 */
class MyDatasource extends DatasourcePluginBase {
  function getPropertyDefinitions() {}
  function loadMultiple($ids) {}
  function getItemId($item) {}
  function getItemIds() {}
  // …
}
```

Datasource

- Also available:
  - Configuration form
  - Viewing and view modes
  - Get item's ID, label, URL
- Call `search_api_track_item_*()`!

Tracker

- Change tracking implementation

```
/**
 * @SearchApiTracker(
 *   id = "MODULE_my_tracker",
 *   name = @Translation("My tracker"),
 *   description = @Translation("It tracks.")
 * )
 */
class MyTracker extends TrackerPluginBase {
  function trackItemsInserted($ids) {}
  function getRemainingItems($limit, $datasource) {}
  function getTotalItemsCount() {}
  // …
}
```

Tracker

- Usually default tracker will suffice
- Can also have a configuration form

Processor

- Pre-/postprocess indexed items and results

```
/**
 * @SearchApiProcessor(
 *   id = "MODULE_my_processor",
 *   label = @Translation("My processor"),
 *   description = @Translation("Does stuff."),
 *   stages = {
 *     "preprocess_query" = 0
 *   }
 * )
 */
class MyProcessor extends ProcessorPluginBase {
  function alterPropertyDefinitions(&$info) {}
  function preprocessIndexItems(&$items) {}
  function preprocessSearchQuery($query) {}
  function postprocessSearchResults(&$response, $query) {}
}
```

Processor

- Also available:
  - `supportsIndex()`
  - Configuration form
  - Default implementation `FieldsProcessorPluginBase`
    - `process()`, `processFieldValue()`, `processKey()`, `processFilterValue()`, `testField()`, `testType()`

Parse modes

- Interpretation of keywords
- By default: „direct“, „single term“, „multiple terms“
- Selection when creating search
- Not done yet, but will be pluggable

Query

- D7: Pluggable in odd way
- D8: TBD

Further D8 changes

- Several data sources per index
- More query functionality (like DB layer)
- Major changes for processors
  - Possibly: „query extenders“ / per-query processor config
- Loads of other stuff

General information

Thank you!

What do you think?

Other questions?