FirstSpirit Techinar

Neue Eingabekomponenten und DataAccessPlugins

e-Spirit

Michael Bergmann
Senior Consultant

Sebastian Glock
VP Product Marketing

Kontakt: productmarketing@e-spirit.com

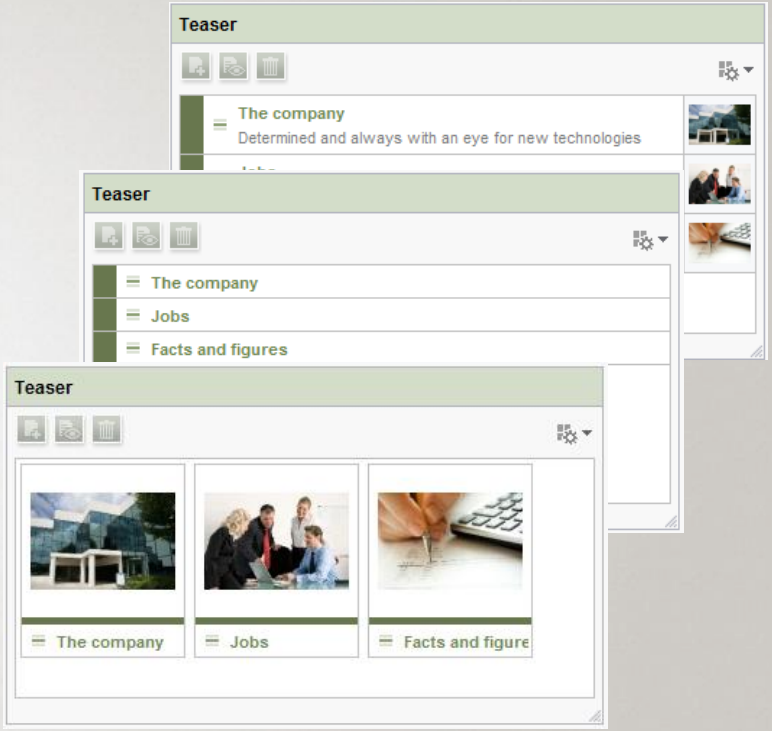
Agenda

- FS_CATALOG
- FS_INDEX
- DataAccessPlugins
- Q & A

FS_CATALOG

General principle: Data "lives" in FS_CATALOG

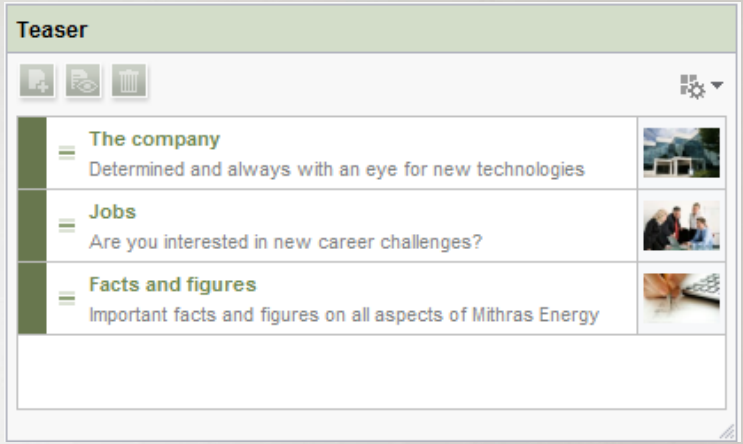
- Like FS_LIST type="inline"
- Uses link- or sectiontemplates for internal data
- Simple configuration
- Uses snippets of inner elements
- Three display variants: Details, Headers, Symbols
  - Editor can switch display mode
  - Default can be set in GOM
- Add and remove buttons can be disabled via rule
  - <PROPERTY name="NEW" source="st_teaser"/>
  - <PROPERTY name="REMOVE" source="st_teaser"/>



DEMO

Compatibility

- CATALOG can read FS_LIST data (type inline)
- Data type: Catalog – Inner type: Catalog.Card
- In most cases: No change to output channel code
  - `$CMS_VALUE(st_catalog)$`
  - ```
    $CMS_FOR(e, st_catalog)$
      $CMS_VALUE(e)$
    $CMS_END_FOR$
    ```
  - Unless you directly access inner elements via API
- Planned: Mechanism to only allow **one** level of language dependency when nesting
  - Allows re-use of section templates directly in pages **and** CATALOG
  - Planned for FS 5.2R5 (end of september 2016)
  - Check release notes when available!


    ```

FS_INDEX

FS_INDEX: Overview

- FS_INDEX for lists of Datasets
  - Much simpler than FS_LIST
  - Can read FS_LIST type database data (=> migration!)
- Three display options (like CATALOG)
- Data type: Index (API), IndexAccessor (template)
- Changes to output channel needed
- Output for DatasetDataAccessPlugin via .formData

```
$CMS_FOR(_product, st_products.values)$
  <h2>$CMS_VALUE(_product.formData.cs_name)$</h2>
$CMS_END_FOR$
```

- Not only for FirstSpirit Datasets!

```
<FS_INDEX name="st_products">
  <LANGINFOS>
    <LANGINFO lang="*" label="Products"/>
  </LANGINFOS>
  <SOURCE name="DatasetDataAccessPlugin">
    <TEMPLATE uid="Products.products"/>
  </SOURCE>
</FS_INDEX>
```

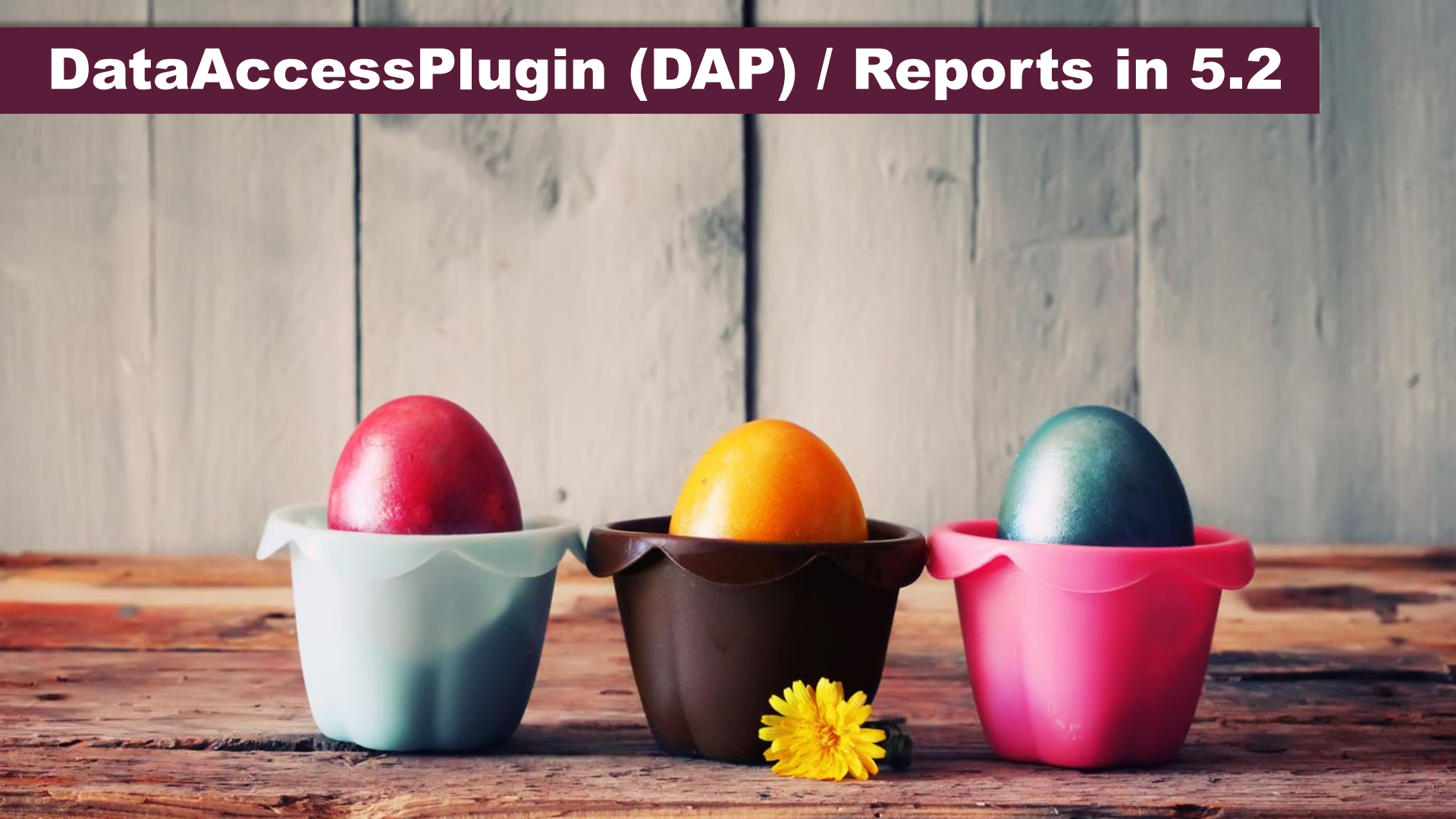
DEMO

DatasetDataAccessPlugin

is just one implementation of a

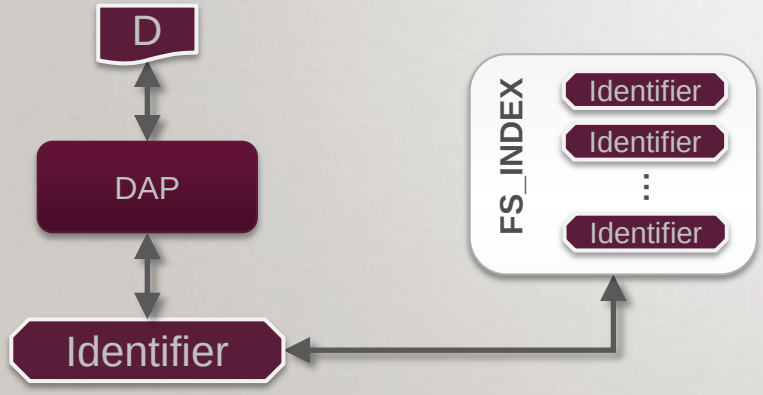
General integration mechanism!

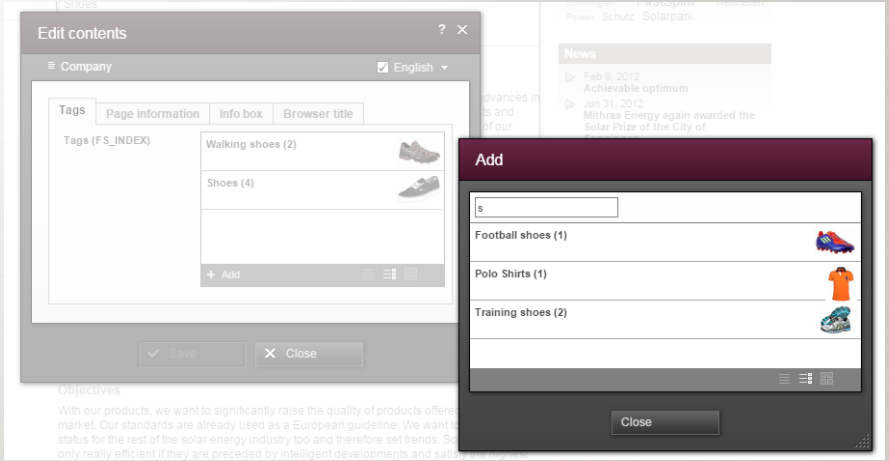
DataAccessPlugin (DAP) / Reports in 5.2



General principle: Integration of "foreign" objects

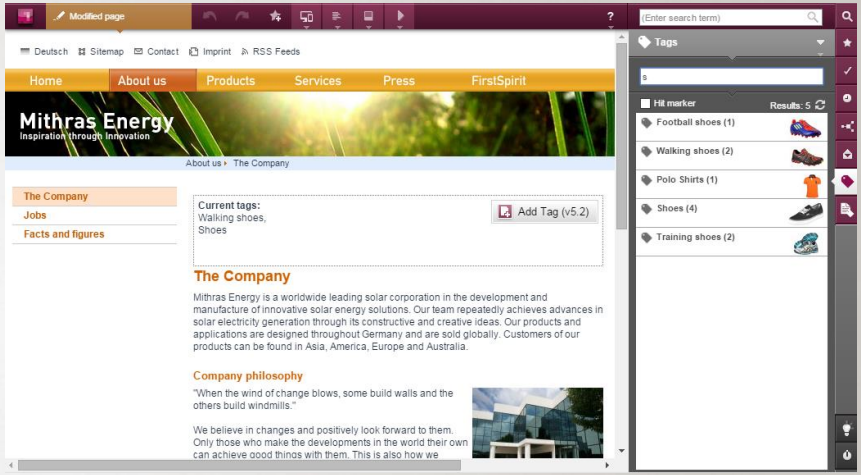
- DAP provide means to reference "foreign" objects of arbitrary type D
  - Objects are "mapped" using an "identifier" (=String)
  - FS_INDEX only saves that **identifier** for each referenced object
- Conversion / translation **object <=> identifier** part of DAP => happens **transparently**
  - Also during **generation** => transparent access using st_index.values()
- Objects displayed using snippets
  - Plugin delivers DataSnippetProvider





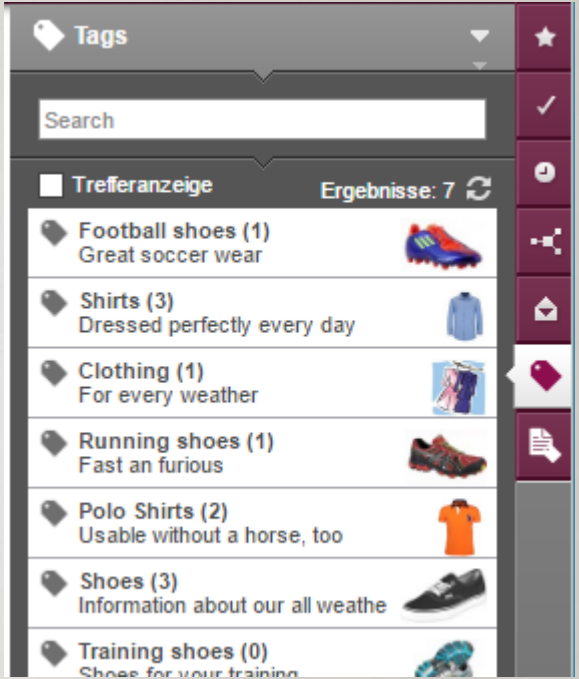
Reports as additional feature / view

- Reports are only a kind of "extension" of the basic functionality (which is the usage in an FS_INDEX)
- Can be triggered using API:
  - JS: top.WE_API.Report.show('DAPclassName')
  - API: ShowReportOperation
  - Optional: With (new) parameters



Example: Tagging

- Tag pages with topics, find pages tagged with those topics
- Data object: TagEntry (that's our "D")
- Here, the tags are managed in a FirstSpirit datasource
  - Just for simplicity
  - To be independent from external system during demo / trainings
  - Instead of using FirstSpirit API to get the tags you could also query an external system (e.g. using REST calls)
- No perfect code, but you should get an idea how it works



DEMO

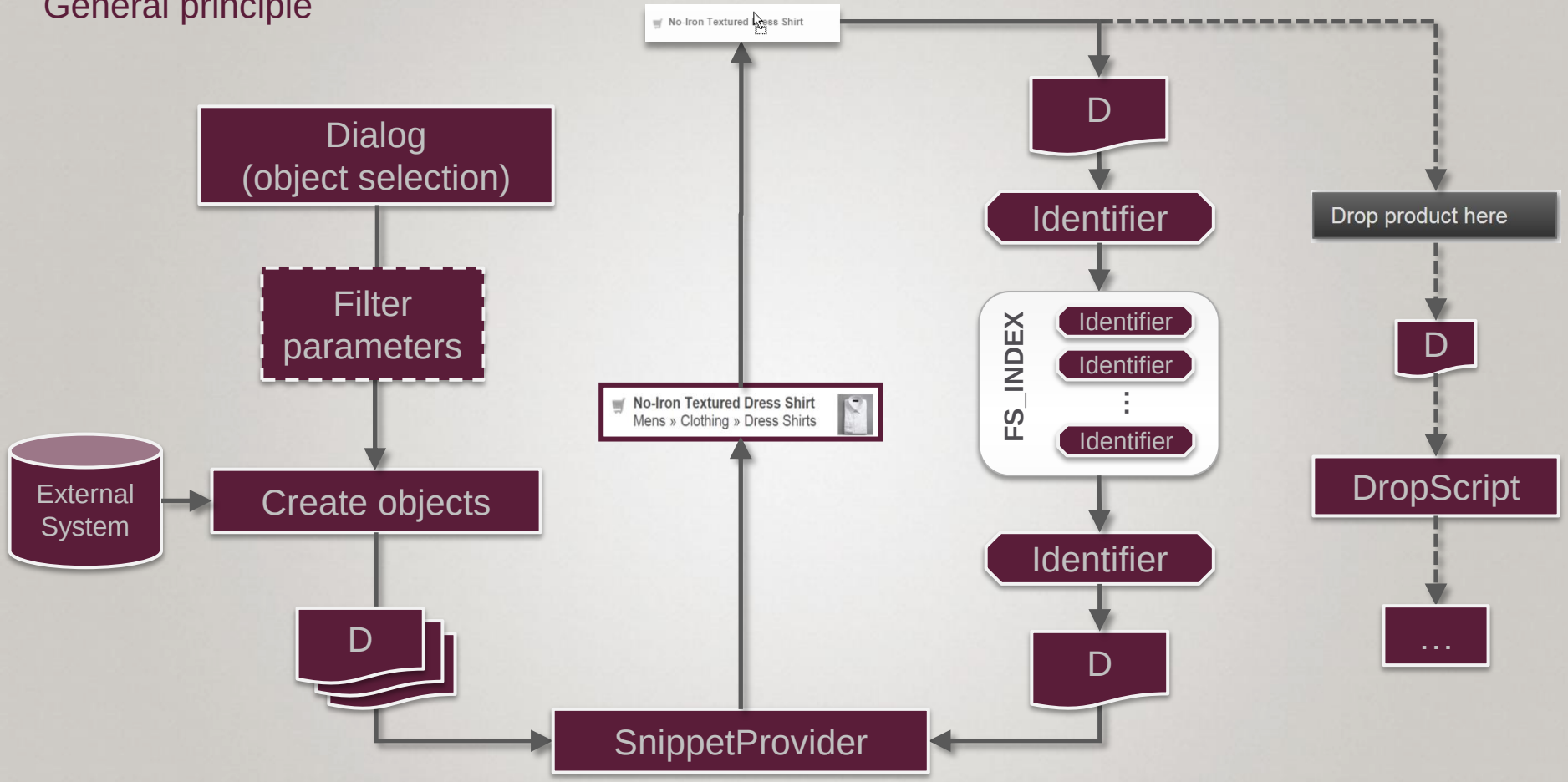
YES

you'll get the sources :-)

YES

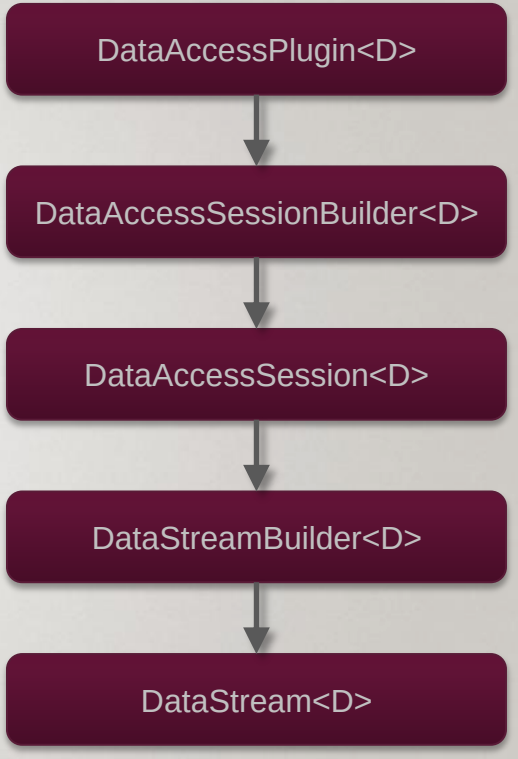
you'll also get the example project ;-)

General principle



Interfaces to implement (Basics)

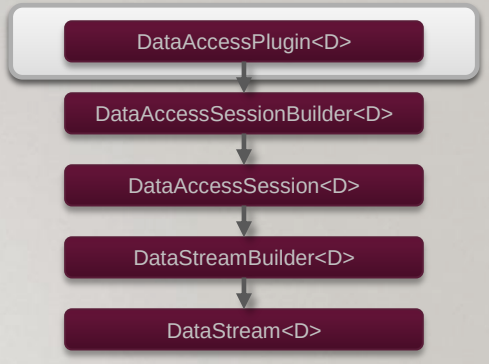
- Package: **de.espirit.firstspirit.client.plugin.dataaccess**
- Plugin: Entry point
- Builder as configuration objects
- Configuration / features via **aspects** for
  - Plugin
  - SessionBuilder
  - Session
  - StreamBuilder
- Basic implementation without additional aspects
  - For use in FS_INDEX only
  - No report, no filter, no Drag&Drop, etc…



DAPs support aspects (like GOMs)

- Objects provide interface implementations which enable specific features
  - But: Without implementing those interfaces themselves (no **implements** statement)
  - Accessed by getAspect(xxxAspectType)
- More flexible than `MyClass implements SomeInterface`
  - Only have to code what you need
  - No implementation of "unused" methods, e.g. using
    - `return null;`
    - `return Collections.EMPTY_LIST;`
  - Decision if an aspect (=feature) is active possible at **runtime**
  - **Adding** new (yet unknown) features possible in FS-API,
    without having to adjust **existing** implementations
  - No risk of method name conflicts when providing multiple features / aspects

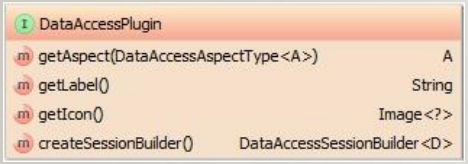
DataAccessPlugin<D>

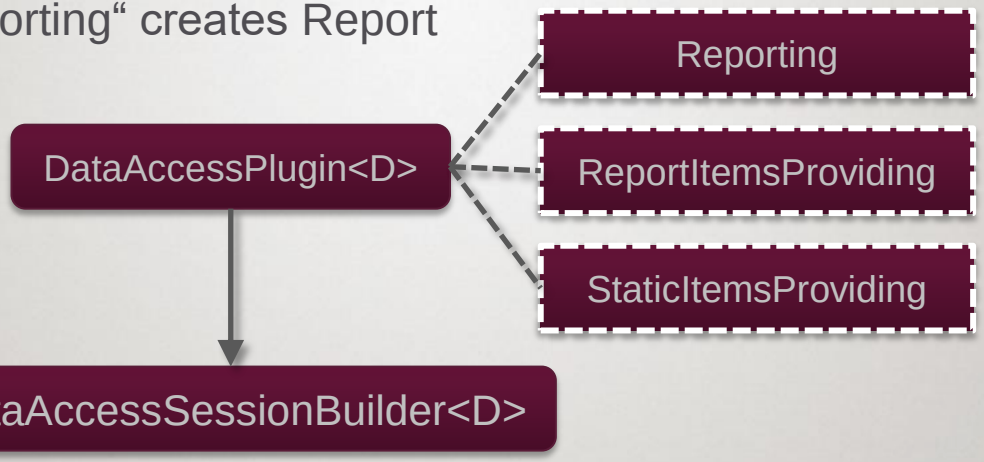


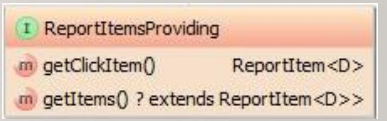
- Entry point, <Public>-component

```
<public>
   <name>TagsDataAccess</name>
   <class>de.espirit.ps.examples.tagging.v52.tags.TagDataAccess</class>
</public>
```

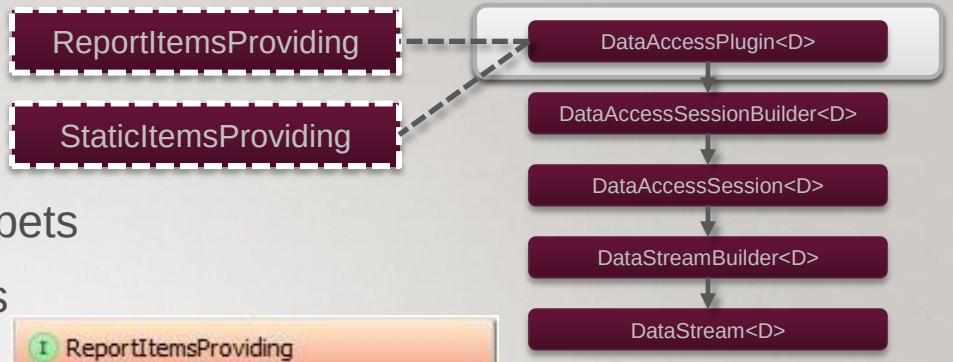
- Provides label, icon and mechanism to create **sessions**
- Usable in FS_INDEX (with CodeCompletion!)
- Adding aspect „Reporting“ creates Report



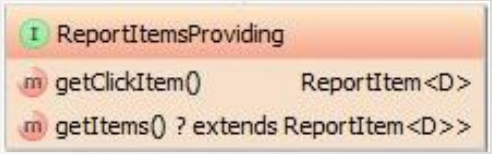




Report Items



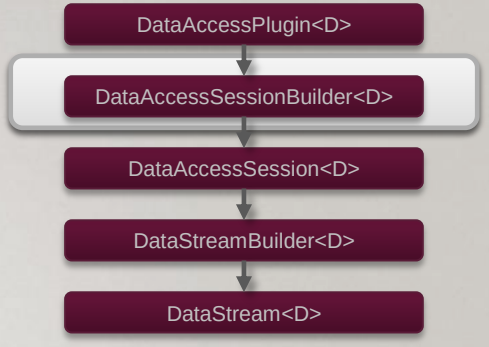
- These are actions and **not** the snippets!
- Allow interaction with the report or ist snippets
- ReportItemsProviding: Actions on snippets
  - Click on snippet itself: getClickItem()
  - Buttons appearing on hover: getItems()

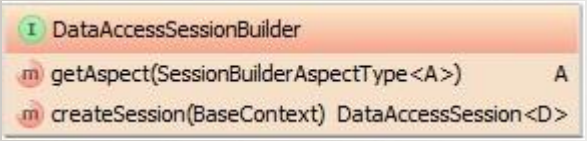


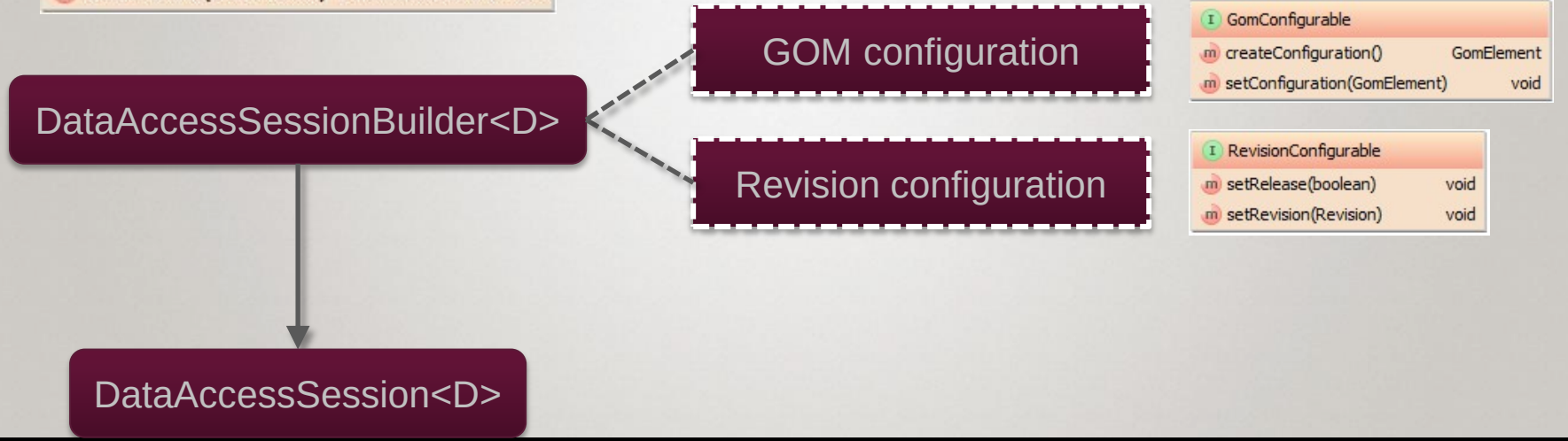
  - CC: ClientScriptProvidingReportItem: Executes JavaScript
  - CC: WebeditExecutableReportItem: execute() method (Java)
  - SA: JavaClientExecutableReportItem: execute() method (Java)
  - Can often be all implemented in one class (multiple implements) for *ExecutableReportItem
- StaticItemsProviding: Offers report wide actions
  - Appear in the dropdown in upper right hand part of report

DataAccessSessionBuilder<D>

- Configuration object for creating a data session
  - Configuration by providing (optional!) aspects
- Simple version without aspects possible

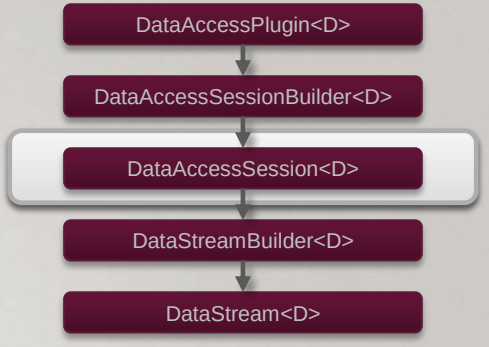


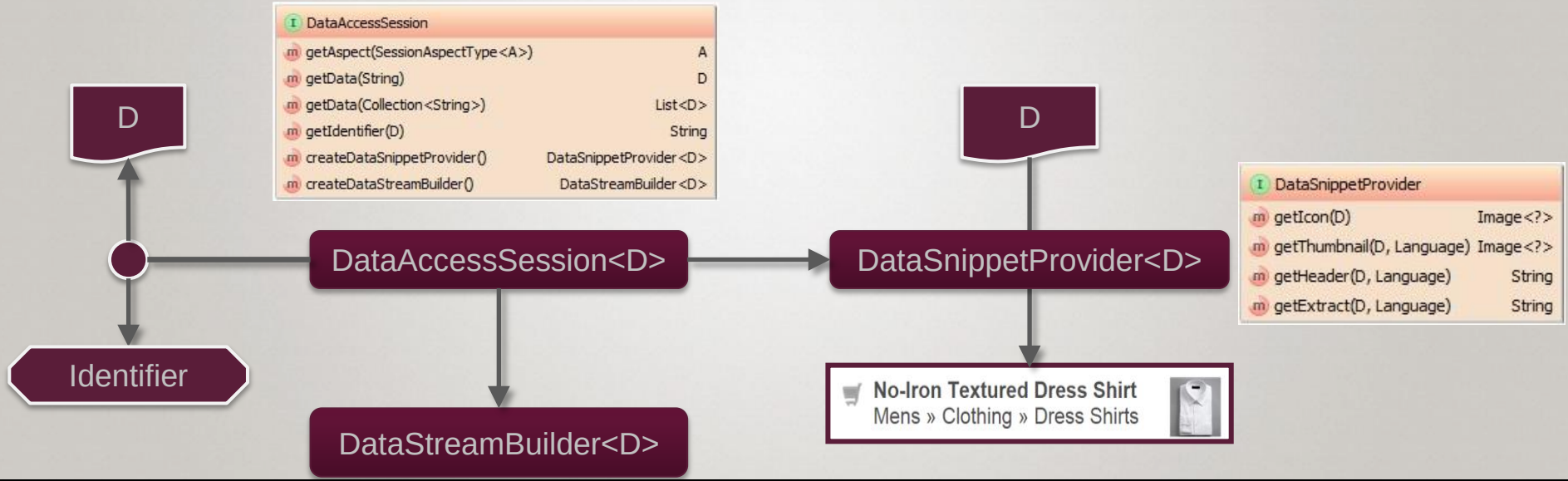




DataAccessSession<D>

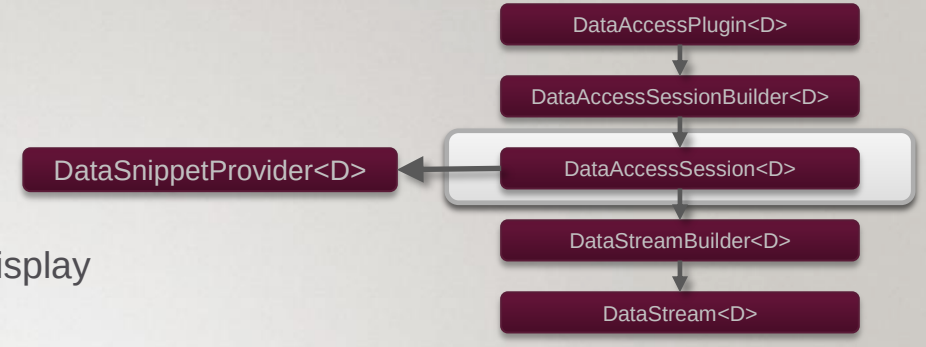
- „Translates“ between identifier(s) and D-object(s)
  - Access objects (D) using identifier
- Provides identifier for an object D
- Provides DataSnippetProvider<D> and DataStreamBuilder<D>

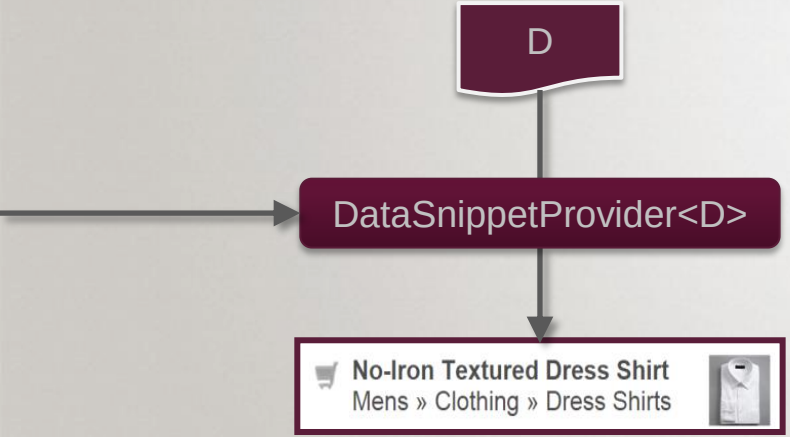




DataSnippetProvider<D>

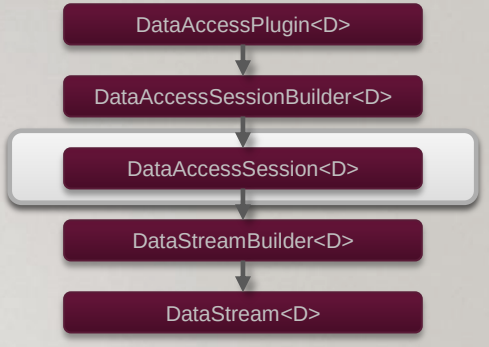
- Extract display information from D-objects
  - Icon, thumbnail, header (=title), extract (=subtitle)
  - Those information is then used by FirstSpirit for display
- Provided by DataAccessSession

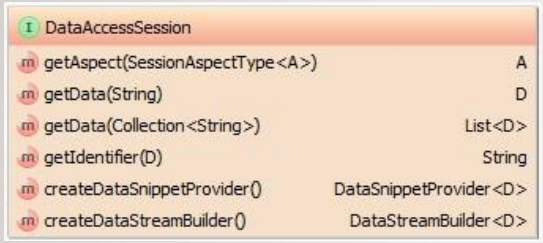


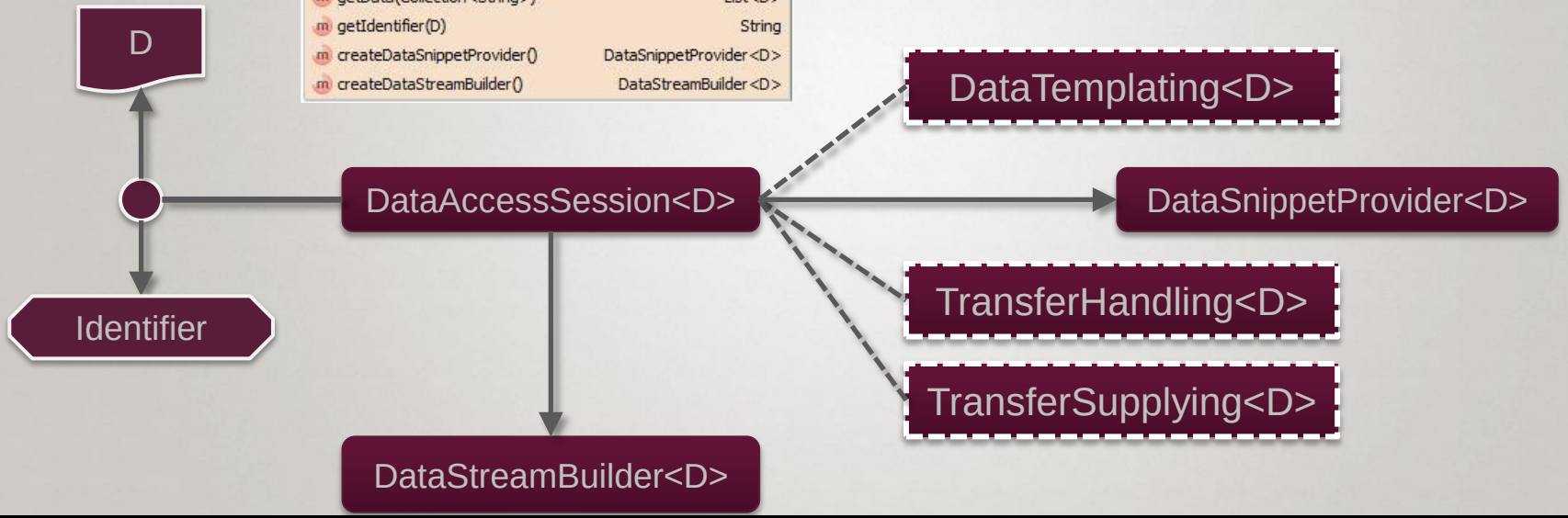


DataAccessSession<D> aspects

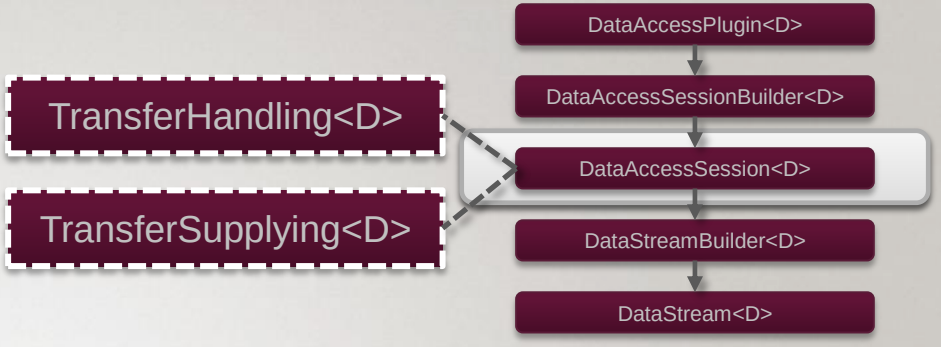
- Knows how to display flyouts (DataTemplating)
- Defines Drag & Drop handling







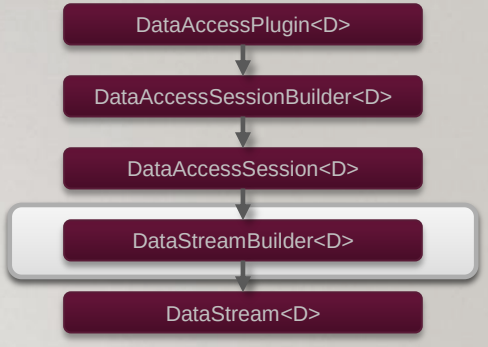
Drag & Drop support

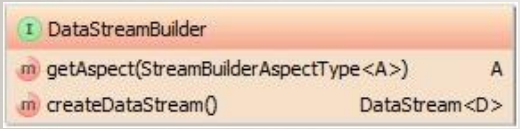


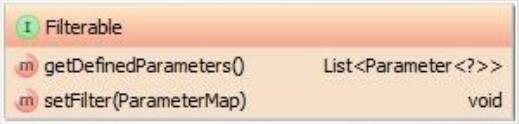
- Split up in "Drag" and "Drop" part
- Drag: TransferSupplying aspect
  - Translates "D" to supported types
  - Multiple types can be supplied at the same time
  - On the "drop side" the needed type is requested via TransferAgent
  - Example: IDProvider **and** Dataset **and** String etc…
  - Can be used to allow dragging report snippets to standard FS input components
- Drop: TransferHandling
  - Allows dropping objects into FS_INDEX
  - Translates dropped object to "D"
  - Handling for multiple types possible
  - Tagging example: You can also drop "normal" datasets to the tagging FS_INDEX

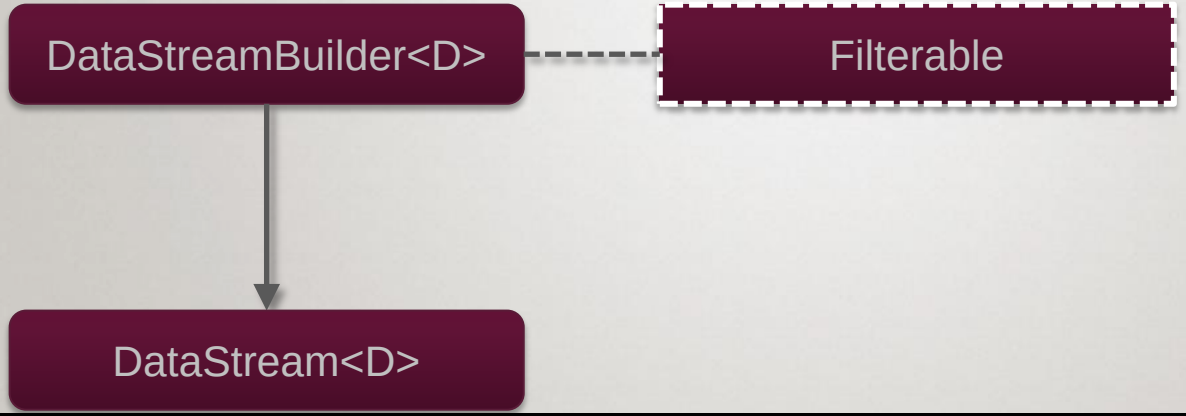
DataStreamBuilder<D>

- Configuration object for the "data stream" of "available" objects
  - Provides (for example) means for filtering when aspect "Filterable" is registered
- Provides "stream" of objects (D) to choose from

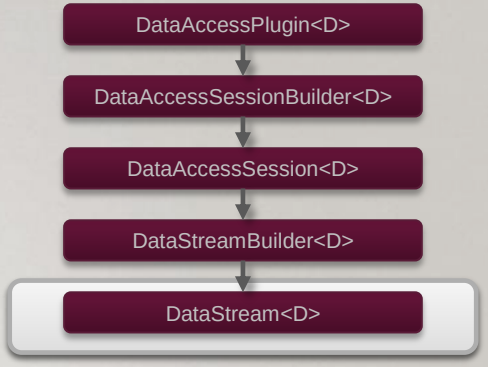




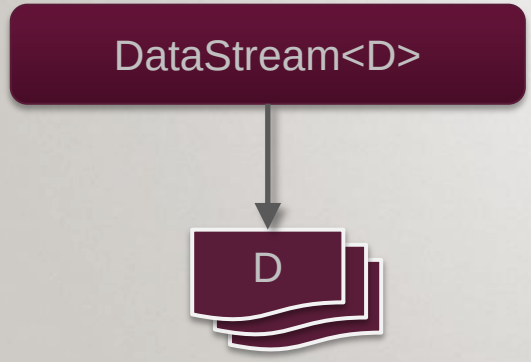


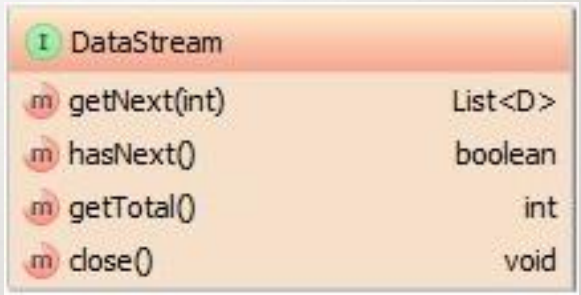


DataStream<D>



- Provides „available“ D-objects for selection
- Used in FS_INDEX and Reports
- SnippetProvider (provided by Session) is used to "display" those objects
- Is configured (e.g. filtered) by the DataStreamBuilder
- Should be implemented "lazy" – don't load everything from start!
  - Use "paged queries" on each **`getNext(int)`** if possible





Best Practices / Hints

- DAP original idea: Only for application integration
  - Turned out that they offer a great UI for internal FirstSpirit objects too!
- Good idea to always use your own wrapper class / interface for "D"
  - You can put conveniance methods here like getIcon(), getTitle(Language) etc.
- Make use of paged and bulk queries against external systems (performance!)
  - Do **not** load everything at the beginning and just "provide" the elements stepwise
- Depending on remote system performance an reliability: Implement caching
  - Example: Two step caching via ClientService => ServerService
- Use a ProjectApp as "marker" if DAP-Report requires special "environment"
  - Disable report if ProjectApp is not installed in project
- DAP needs an AppCenterSlot (only!) when providing a Report
  - Using only the FS_INDEX functionality is "free"

Aspects (1)

Aspectable Interface (Aspect-Type)	**Aspects**	**Usage / Result**
DataAccessPlugin (DataAccessAspectType)	Reporting *	Creates a Report from a DAP
	ReportItemsProviding *	Actions on report result snippets (Buttons)
	StaticItemsProviding *	Global Report actions (Buttons)
DataAccessSessionBuilder (SessionBuilderAspectType)	GomConfigurable	Further configuration via GOM e.g. the table template in DatasetDataAccessPlugin
	RevisionConfigurable	Awareness of revision information and current / release status

* = used in tagging example

Aspects (2)

Aspectable Interface (Aspect-Type)	**Aspects**	**Usage / Result**
DataAccessSession (SessionAspectType)	DataTemplating *	Display of a details flyout (HTML, CC only)
	Identifying	Alternative identification Object => T
	ModelReferencing	Creates model references (e.g. => Templates)
	ValueReferencing *	Creates value references (e.g. => Datasets, Pages)
	ValueIndexing	Provides indexing information
	RequestMatching	Matches a data object against a request
	TransferSupplying *	Provides drop object(s)
	TransferHandling *	Enables handling of drop objects
DataStreamBuilder (StreamBuilderAspectType)	Filterable *	Defines filter parameters - takes set filter values
	DataAssociating *	Aspect providing means to associate data with certain identifiers Allows "Hitmarkers" using "dataAssociation(…)" in templates

* = used in tagging example

Reports – Comparison 5.0 .. 5.2

	5.0	5.1	5.2
Display of objects [from remote systems] as Snippets	+	+	+
Drag & Drop provides Strings	+	+	+
Available in WebEdit / ContentCreator	+	+	+
Drag & Drop provides arbitrary / self definable objects (TransferHandler)	-	+	+
Available in JavaClient / SiteArchitect	-	+	+
Actions (buttons) on result snippets	-	+	+
Global Report actions (e.g. create new object)	-	-	+
Identification mechanism (Mapping Identifier ⇔ Object) usable outside Reports	-	-	+
Persistent references to "foreign" objects without project specific implementation (drop scripts)	-	-	+
Editorial object infos without proxy objects or saving additional info in FS	-	-	+


That's cool!
Wanna have that!

Questions?

Thanks!

United States | Germany | United Kingdom | Austria | Switzerland