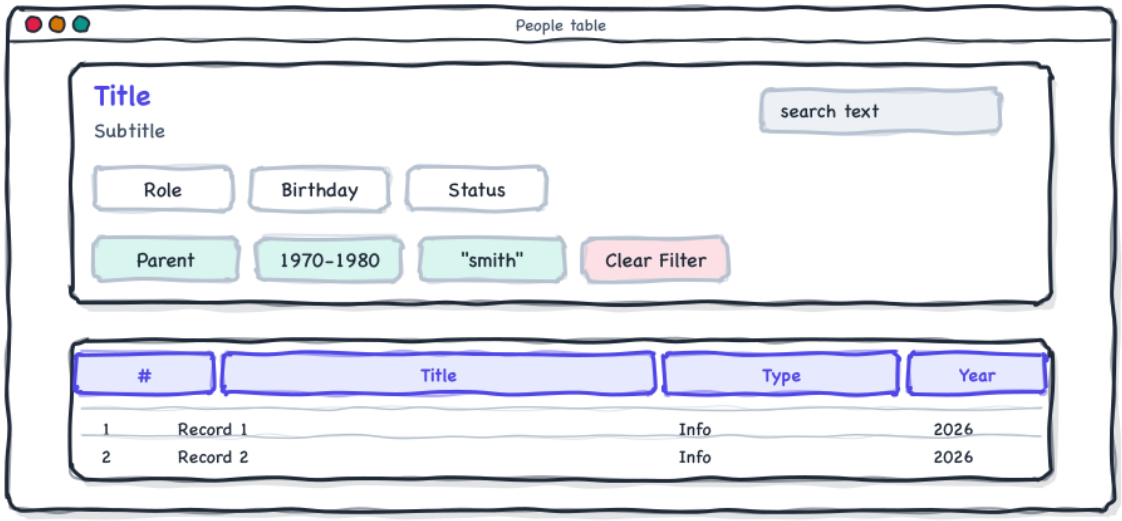
FilterView

Developer Manual

GemsFX • com.dlsc.gemsfx • JavaFX Custom Controls



FilterView combines group menus, chips and a search field into a reusable predicate pipeline.

FilterView is a generic filtering control. It groups selectable Predicate objects into menu buttons, turns selections into chips, optionally combines them with a text-filter provider, and exposes a read-only filteredItems list for tables, lists or custom views.

Table of Contents

1. Introduction

FilterView is a `Control` for building reusable filter bars. It owns source items, selected filters and a read-only filteredItems list.

1.1 Key features

- Filter groups shown as menu buttons with All and None actions.
- Filters in the same group are ORed; different groups are ANDed.
- Optional text filter from a SearchTextField.
- Optional additional predicate for application state.
- Selected filters and filter text become removable chips.
- Header with title, subtitle, graphics and extras.

1.2 Maven dependency

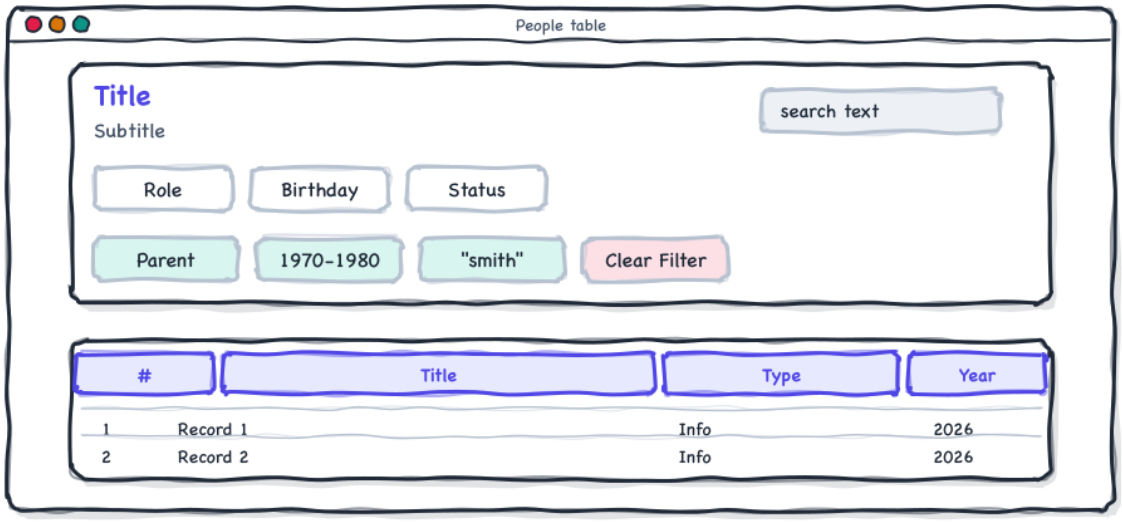
```
<dependency>
    <groupId>com.dlsc.gemsfx</groupId>
    <artifactId>gemsfx</artifactId>
    <version>4.4.1</version>
</dependency>
```

The class is in `com.dlsc.gemsfx`.

2. Getting started

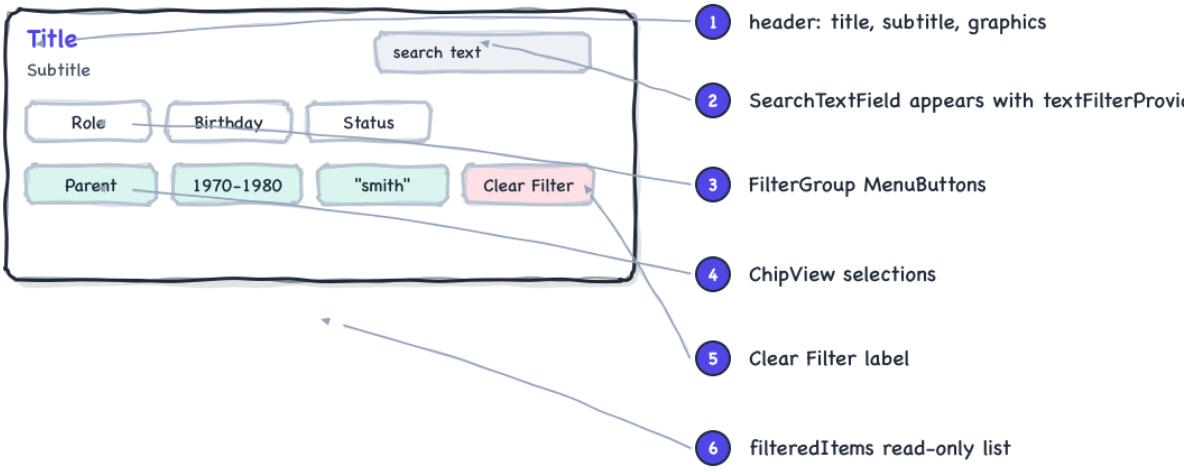
```
FilterView<Person> filterView = new FilterView<>();
filterView.setTitle("People");
filterView.setTextFilterProvider(text -> person ->
    person.getFirstName().toLowerCase().contains(text.toLowerCase()) ||
    person.getLastName().toLowerCase().contains(text.toLowerCase()));

FilterGroup<Person> role = new FilterGroup<>("Role");
role.getFilters().add(new Filter<>("Parents") {
    @Override public boolean test(Person p) { return p.getRole().equals("Parent"); }
});
filterView.getFilterGroups().add(role);
table.setItems(new SortedList<>(filterView.getFilteredItems()));
```

A filter view feeding a table.



FilterView above a table of filtered items.

3. Anatomy



The parts of FilterView.

Part	Style class	Description
.filter-view	Root control style class.	
.filter-container	Main container in the skin.	
.header-box	Title, subtitle, search field and extras.	
.title-subtitle-box, .title-box, .title, .title-postfix, .subtitle	Header text containers and labels.	
.filter-groups	Container for group menu buttons.	

Part	Style class	Description
.filters	Container for selected filter chips.	
.clear-filter-label	Action label that clears text and selected filters.	

4. Control API

4.1 Filtering properties

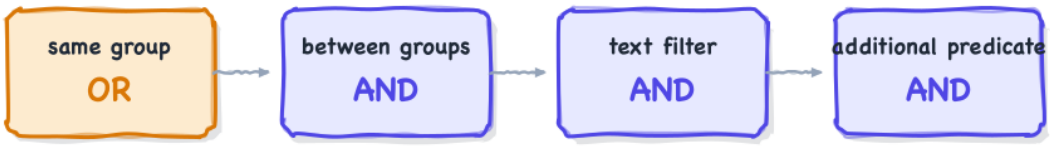
Property	Type	Default	Description
items	ListProperty<T>	empty list	Source items to filter.
filteredItems	ReadOnlyListProperty<T>	FilteredList over items	Result list after all predicates are applied.
filterGroups	ListProperty<FilterGroup<T>>	empty list	Groups shown as menu buttons.
filters	ReadOnlyListProperty<Filter<T>>	flattened groups	All filters from all groups.
filterText	StringProperty	empty string	Text in the SearchTextField.
textFilterProvider	ObjectProperty<Function<String, Predicate<T>>>	null	Creates a predicate from filterText. Applied only when text is not blank.
additionalFilterPredicate	ObjectProperty<Predicate<T>>	item -> true	Extra predicate ANDed with all other filters.
scrollThreshold	IntegerProperty	100	Selected filter count above which chips are wrapped in a ScrollPane.

4.2 Header properties

Property	Type	Default	Description
showHeader	BooleanProperty	true	Shows title, subtitle, search and extras header. Styleable as -fx-show-header.
title	StringProperty	Untitled	Header title.
subtitle	StringProperty	empty string	Header subtitle.
titlePostfix	StringProperty	empty string	Small text after the title.
titleGraphic / titlePostfixGraphic / subtitleGraphic	ObjectProperty<Node>	null	Optional graphics in the header.
extras	ObjectProperty<Node>	null	Optional node at the end of the header.

5. Predicate model

Every selected filter is a Predicate. Filters inside one FilterGroup are alternatives and therefore ORed together. The result of each group is ANDed with the other groups, the optional text predicate, and the additionalFilterPredicate.

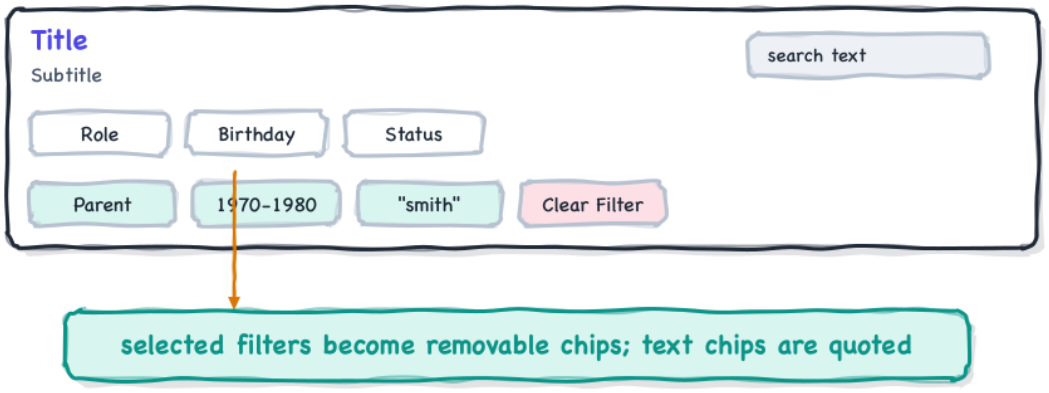


How FilterView combines predicates.

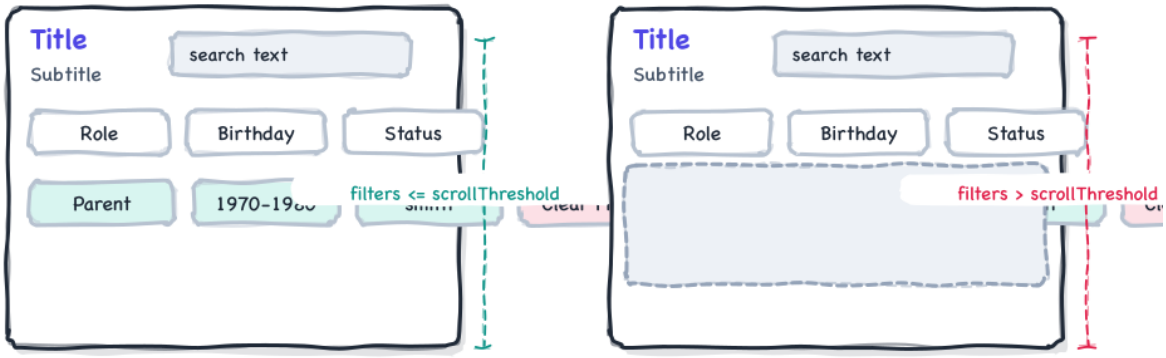
> **Note.** If a group has no selected filters, that group does not restrict the result set.

6. Chips and selected filters

The skin displays selected filters as chips. The filter text also becomes a chip using the bundle pattern `format.filter-text-quoted`.



Selected filters and search text become chips.



Large chip sets use a ScrollPane after scrollThreshold is exceeded.

7. FilterGroup and Filter classes

Property	Type	Default	Description
FilterGroup.name	StringProperty	Untitled	Name shown on the group menu button.
FilterGroup.id	StringProperty	derived from name	Identifier derived from the name by replacing separators.
FilterGroup.filters	ObservableList<Filter<T>>	empty list	Filters displayed in the group menu.
Filter.name	StringProperty	Untitled	Name shown in the menu and chip.
Filter.id	StringProperty	derived from name	Identifier for the filter.
Filter.selected	BooleanProperty	constructor value	Whether the predicate participates in filtering.
Filter.group	ObjectProperty<FilterGroup<T>>	set by group	Back reference to the owning group.

8. Styling

Style class	Description
.filter-view	Root control style class.
.filter-container	Main container in the skin.
.header-box	Title, subtitle, search field and extras.
.title-subtitle-box, .title-box, .title, .title-postfix, .subtitle	Header text containers and labels.
.filter-groups	Container for group menu buttons.
.filters	Container for selected filter chips.
.clear-filter-label	Action label that clears text and selected filters.

CSS property	Type	Default
-fx-show-header	boolean	true

```
.filter-view {
    -fx-show-header: true;
}
.filter-view .clear-filter-label {
    -fx-underline: true;
}
```

Example CSS for FilterView.

9. Localization

Key	English default
title.untitled	Untitled
group.name.untitled	Untitled
filter.name.untitled	Untitled
menu.select-all	All
menu.select-none	None
action.clear-filter	Clear Filter
format.filter-text-quoted	"{0}"
accessible.role-description	filter

10. Accessibility

FilterView sets `AccessibleRole.NODE` and uses AccessibilityUtil with the localized role description *filter*.

11. Recipes

11.1 Bind to a TableView

```
SortedList<Person> sorted = new SortedList<>(filterView.getFilteredItems());
sorted.comparatorProperty().bind(table.comparatorProperty());
table.setItems(sorted);
```

11.2 Add an always-on predicate

```
filterView.setAdditionalFilterPredicate(person -> person.isActive());
```

11.3 Hide the header

```
filterView.setShowHeader(false);
```

11.4 Clear search and selections

```
filterView.getFilters().forEach(f -> f.setSelected(false));
filterView.setFilterText("");
```

11.5 Checklist

1. Group alternatives that should OR together.
2. Put independent dimensions into separate groups.
3. Keep textFilterProvider null unless a search field is useful.
4. Use FilterView when you need data filtering; use SimpleFilterView for lightweight input chips.

12. See also

- Demo app: `FilterViewApp`. Run with `mvn javafx:run -f gemsfx-demo/pom.xml -Dmain.class=com.dlsc.gemsfx.demo.FilterViewApp`.
- Related controls: SimpleFilterView, ChipsViewContainer and SearchTextField.
- API docs: https://dlsc-software-consulting-gmbh.github.io/GemsFX/api/