

[UIKit](#) / [View controllers](#) / Displaying searchable content by using a search controller

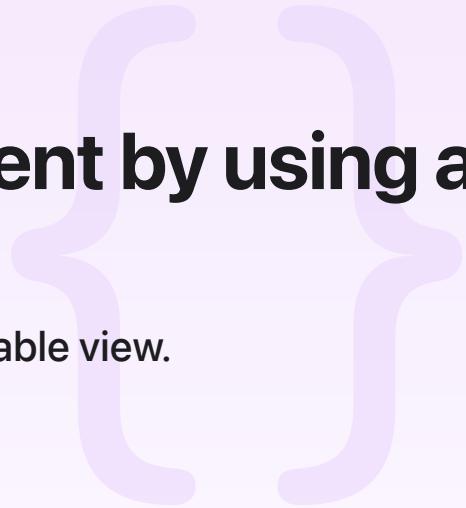
Sample Code

Displaying searchable content by using a search controller

Create a user interface with searchable content in a table view.

[Download](#)

iOS 11.0+ | iPadOS 11.0+ | Xcode 13.0+



Overview

This sample demonstrates how to create a table view controller and search controller to manage the display of searchable content. It creates another custom table view controller to display the search results. This table view controller also acts as the presenter or provides context for the search results so they're presented within their own context.

This sample includes the optional—but recommended—[UIStateRestoring](#) protocol. You adopt this protocol from the view controller class to save the search bar's active state, first responder status, and search bar text and restore them when the app is relaunched.

Create a search controller

Use `MainTableViewController`, a subclass of [UITableViewController](#), to create a search controller. The search controller searches and filters a set of `Product` objects and displays them in a table called `ResultsTableController`. This table controller is displayed as the user enters a search string and is dismissed when the search is complete.

```
override func viewDidLoad() {
    super.viewDidLoad()

    let nib = UINib(nibName: "TableViewCell", bundle: nil)
```

```

tableView.register(nib, forCellReuseIdentifier: tableViewCellIdentifier)

resultsTableController =
    self.storyboard?.instantiateViewController(withIdentifier: "ResultsTableCont
// This view controller is interested in table view row selections.
resultsTableController.tableView.delegate = self

searchController = UISearchController(searchResultsController: resultsTableCont
searchController.delegate = self
searchController.searchResultsUpdater = self
searchController.searchBar.autocapitalizationType = .none
searchController.searchBar.delegate = self // Monitor when the search button is

searchController.searchBar.scopeButtonTitles = [Product.productTypeName(forType:
    Product.productTypeName(forType:
    Product.productTypeName(forType:
    Product.productTypeName(forType:

// Place the search bar in the navigation bar.
navigationItem.searchController = searchController

// Make the search bar always visible.
navigationItem.hidesSearchBarWhenScrolling = false

/** Search presents a view controller by applying normal view controller present
    This means that the presentation moves up the view controller hierarchy until
    view controller or one that defines a presentation context.
*/
/** Specify that this view controller determines how the search controller is pre
    The search controller should be presented modally and match the physical size
*/
definesPresentationContext = true

setupDataSource()
}

```

Update the search results

This sample uses the [UISearchResultsUpdating](#) protocol, along with [NSComparisonPredicate](#), to filter search results from the group of available products. NSComparison Predicate is a foundation class that specifies how data should be fetched or filtered using

search criteria. The search criteria are based on what the user types in the search bar, which can be a combination of product title, year introduced, and price.

To prepare for a search, the search bar content is trimmed of all leading and trailing space characters. Then the search string is passed to the `findMatches` function, which returns the `NSComparisonPredicate` used in the search. The product list results are applied to the search results table as a filtered list.

```
func updateSearchResults(for searchController: UISearchController) {
    // Update the filtered array based on the search text.
    let searchResults = products

    // Strip out all the leading and trailing spaces.
    let whitespaceCharacterSet = CharacterSet.whitespaces
    let strippedString =
        searchController.searchBar.text!.trimmingCharacters(in: whitespaceCharacterSet)
    let searchItems = strippedString.components(separatedBy: " ") as [String]

    // Build all the "AND" expressions for each value in searchString.
    let andMatchPredicates: [NSPredicate] = searchItems.map { searchString in
        findMatches(searchString: searchString)
    }

    // Match up the fields of the Product object.
    let finalCompoundPredicate =
        NSCompoundPredicate(andPredicateWithSubpredicates: andMatchPredicates)

    let filteredResults = searchResults.filter { finalCompoundPredicate.evaluate(with: $0) }

    // Apply the filtered results to the search results table.
    if let resultsController = searchController.searchResultsController as? ResultsController {
        resultsController.filteredProducts = filteredResults
        resultsController.tableView.reloadData()

        resultsController.resultsLabel.text = resultsController.filteredProducts.isEmpty ?
            NSLocalizedString("NoItemsFoundTitle", comment: "") :
            String(format: NSLocalizedString("Items found: %ld", comment: ""),
                   resultsController.filteredProducts.count)
    }
}
```

See Also

Search interface

`class UISearchContainerViewController`

A view controller that manages the presentation of search results in your interface.

`class UISearchController`

A view controller that manages the display of search results based on interactions with a search bar.

`class UISearchBar`

A specialized view for receiving search-related information from the user.

`protocol UISearchResultsUpdating`

A set of methods that let you update search results based on information the user enters into the search bar.

{ } Using suggested searches with a search controller

Create a search interface with a table view of suggested searches.