## **Search and Filtering Utility Documentation**

### **Overview**

This utility provides a flexible and powerful way to search and filter any list of objects based on dynamic criteria. It allows users to filter data based on any property of the objects in the list, whether it's a string, integer, or datetime. The utility supports both exact matching and substring searching for strings.

### **Key Features**

* **Generic Search and Filter**: Can be used on any list of objects, regardless of the object type.
* **Supports Multiple Data Types**: Works with strings, integers, and DateTime fields.
* **Flexible Filtering**: Supports filtering by exact match, substring match, and date comparisons.
* **Sorting Capability**: Results can be sorted by any property in either ascending or descending order.

### **How to Use the Utility**

#### **Step 1: Add the Utility to Your Project**

1. **Install the Utility**: Add the class library that contains the SearchFilterUtility class to your project.
   * If you have a pre-built library, reference the library in your project.
   * If the code is part of your project, make sure the SearchFilterUtility class is accessible.

**Import the Utility**: Import the namespace where the class is located.  
  
**Step 2: Create Your List of Objects**

You can use the utility with any collection of objects. Below is an example with a list of Person objects.

var people = new List<Person>

{

new Person { Name = "John", Age = 30, BirthDate = new DateTime(1991, 5, 10) },

new Person { Name = "Jane", Age = 25, BirthDate = new DateTime(1996, 7, 15) },

new Person { Name = "Joe", Age = 35, BirthDate = new DateTime(1986, 3, 20) }

};

#### **Step 3: Instantiate and Use the SearchFilterUtility**

Create an instance of the SearchFilterUtility class and call the SearchAndFilter method with your list, search query, and optional parameters like sort order.

var utility = new SearchFilterUtility();

var filteredResults = utility.SearchAndFilter(people, "Joe", sortBy: "Age", sortOrder: "desc");

#### **Step 4: Handle the Results**

After filtering and sorting, the SearchAndFilter method will return a filtered list of objects.

foreach (var person in filteredResults)

{

Console.WriteLine($"{person.Name}, {person.Age}, {person.BirthDate.ToShortDateString()}");

}

### **Method Signature**

public List<T> SearchAndFilter<T>(

List<T> list,

string searchQuery = "",

string sortBy = "",

string sortOrder = "asc"

)

### **Parameters**

* list (List<T>): The list of objects to search and filter.
* searchQuery (string): The search query used to filter the list. The search is performed on all string, integer, and DateTime properties. If no value is provided, no filtering is applied.
* sortBy (string): The property name used for sorting the result list. If no value is provided, the list is not sorted.
* sortOrder (string): Defines the sort order. Accepted values are "asc" for ascending and "desc" for descending. Default is "asc".

### **Return Value**

* Returns a filtered and optionally sorted list of objects that match the search criteria.

### **Example Usage**

#### **Example 1: Simple Search**

var utility = new SearchFilterUtility();

var filteredResults = utility.SearchAndFilter(people, "Jane");

This will filter the people list to only include the person whose name contains the word "Jane".

#### **Example 2: Search with Sorting**

var utility = new SearchFilterUtility();

var filteredResults = utility.SearchAndFilter(people, "Joe", sortBy: "Age", sortOrder: "desc");

This will filter the people list to only include the person whose name contains "Joe", and then sort the result by Age in descending order.

#### **Example 3: Search with Date Filtering**

var utility = new SearchFilterUtility();

var filteredResults = utility.SearchAndFilter(people, "1991");

This will filter the list to include people whose BirthDate contains the year "1991".

### **Additional Notes**

* **DateTime Filtering**: When searching with dates, the utility will attempt to match the year, month, or day depending on the input.
* **Case Sensitivity**: The search is case-insensitive by default.
* **Invalid Sort Property**: If you provide a sortBy value that does not exist in the object properties, the list will not be sorted.