Generic Dictionaries

Deborah Kurata
CONSULTANT | SPEAKER | AUTHOR

@deborahkurata | blogs.msmvps.com/deborahk/

Kinds of Collections

Lists

"Red"
"Espresso"
"White"
"Navy"

Dictionaries

"CA"	"California"
"WA"	"Washington"
"NY"	"New York"

"CA"	"California"
"WA"	"Washington"
"NY"	"New York"

Generic Dictionary

A strongly typed collection of keys and values

Key:

- Must be unique
- Must not be changed
- Cannot be null

List vs. Dictionary

List	Dictionary
Contains elements	Contains elements defined as key and value pairs
Accessed by a positional index	Accessed by key
Allows duplicate elements	Allows duplicate values but unique keys
Marginally faster iteration	Marginally faster look ups

Generic Dictionary

Dictionary<TKey, TValue>

Dictionary<int, int>

Dictionary<int, string>

Dictionary<string, string>

Dictionary<int, Product>

Dictionary<string, Product>

Overview

- Declaring and Populating a Generic Dictionary
- Using Collection Initializers
- Initializing a Dictionary of Objects
- Retrieving an Element from a Generic Dictionary
- Iterating Through a Generic Dictionary
- Types of C# Dictionaries
- FAQ

Declaring a Generic Dictionary

- **Dictionary of what?**
  - **Value**
  - **Key**

"California"
"Washington"
"New York"

Declaring a Generic Dictionary

```
Dictionary<string, string> states;
```

- Dictionary of what?
  - Value
  - Key
- Dictionary<TKey, TValue>
  - TKey is the type of the key
  - TValue is the type of the value

"CA"	"California"
"WA"	"Washington"
"NY"	"New York"

Initializing a Generic Dictionary

```
Dictionary<string, string> states;
states = new Dictionary<string, string>();
```

- Reference type
- new keyword

Declaring and Initializing a Dictionary

```
Dictionary<string, string> states;
states = new Dictionary<string, string>();
```

```
Dictionary<string, string> states =
                            new Dictionary<string, string>();
```

```
var states = new Dictionary<string, string>();
```

Populating a Dictionary

```
states.Add("CA", "California");
```

Populating a Dictionary

```
states.Add("CA", "California");
states.Add("WA", "Washington");
states.Add("NY", "New York");
```

"CA"	"California"
"WA"	"Washington"
"NY"	"New York"

Dictionary Best Practices

Do:

Use a generic dictionary to manage a collection by key

Avoid:

Using a dictionary if there is no clear or unique key

Using a dictionary if you don't plan to look elements up by key

Declaring and Populating a Dictionary

```
var states = new Dictionary<string, string>();
```

```
states.Add("CA", "California");
states.Add("WA", "Washington");
states.Add("NY", "New York");
```

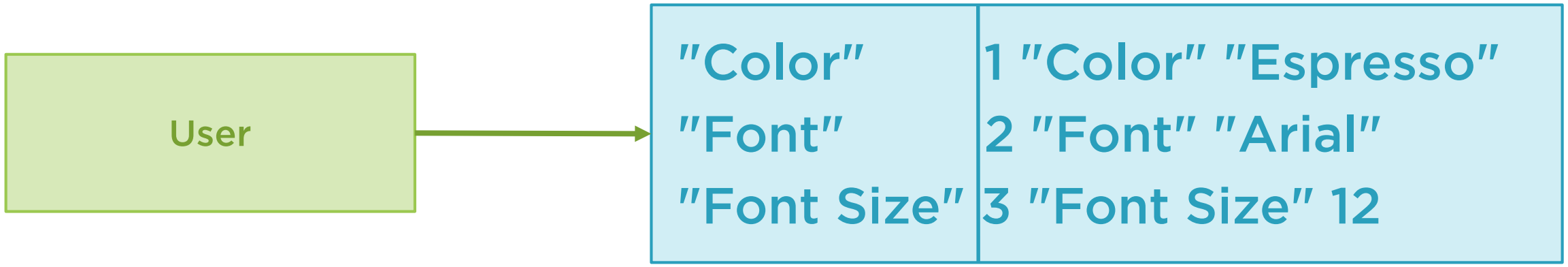
Collection Initializers

```
var states = new Dictionary<string, string>();
```

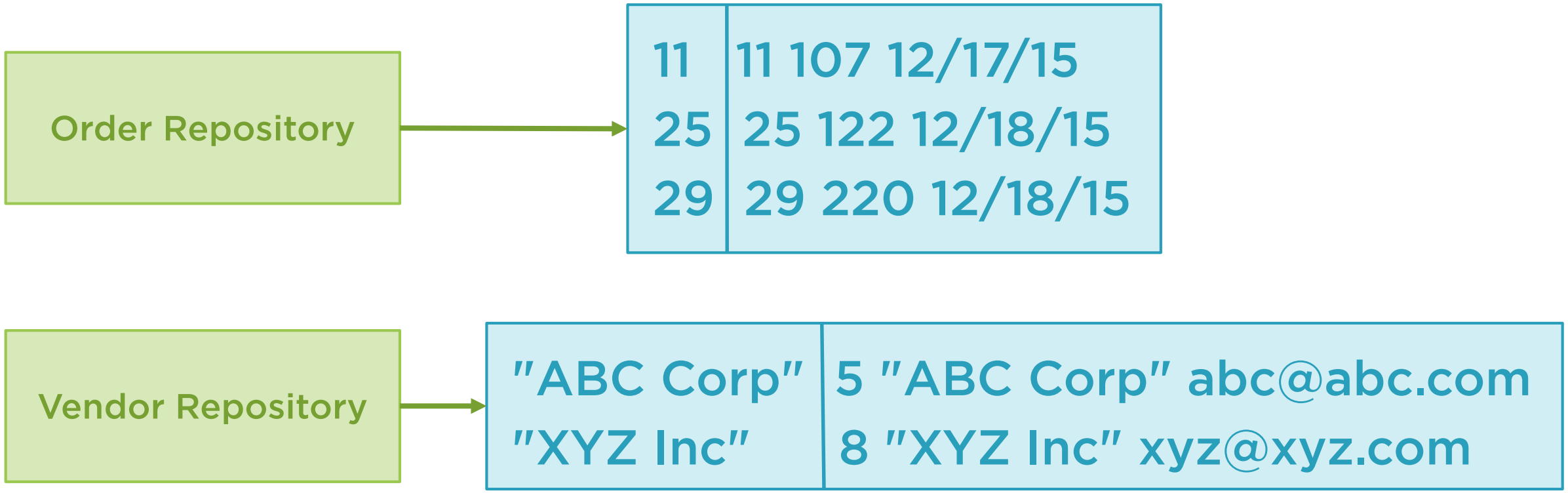
```
states.Add("CA", "California");

states.Add("WA", "Washington");

states.Add("NY", "New York");
```

```
var states = new Dictionary<string, string>()
{
    {"CA", "California" },
    {"WA", "Washington"},
    {"NY", "New York" },
};
```

Dictionary of Objects



Dictionary of Objects



Declaring, Initializing, and Populating a Dictionary

```
var vendors = new Dictionary<string,Vendor>();
```

```
var vendor = new Vendor() {VendorId=5,CompanyName="ABC Corp",Email="abc@abc.com"};
vendors.Add(vendor.CompanyName, vendor);

vendor = new Vendor() {VendorId = 8,CompanyName = "XYZ Inc",Email = "xyz@xyz.com"};
vendors.Add(vendor.CompanyName, vendor);
```

Collection Initializers

```
var vendors = new Dictionary<string, Vendor>()
{
    { "ABC Corp", new Vendor()
       { VendorId = 5, CompanyName = "ABC Corp", Email = "abc@abc.com" } },
    { "XYZ Inc", new Vendor()
       { VendorId = 8, CompanyName = "XYZ Inc", Email = "xyz@xyz.com" } }
};
```

Retrieving an Element from a Dictionary

"ABC Corp"	5 "ABC Corp" abc@abc.com
"XYZ Inc"	8 "XYZ Inc" xyz@xyz.com

```
vendors["XYZ Inc"];
```

Retrieving Dictionary Element Best Practices

Do:

Retrieve elements by key

Avoid:

Retrieving elements by key if you are not sure the key is valid

Use ContainsKey or TryGetValue

Retrieving elements by key when you need all elements

Iterate through instead

Iterating Through a Generic Dictionary

"CA"	"California"
"WA"	"Washington"
"NY"	"New York"

"ABC Corp"	5 "ABC Corp" abc@abc.com
"XYZ Inc"	8 "XYZ Inc" xyz@xyz.com

- Elements
- Keys
- Values

Iterating Through a Generic Dictionary

"CA"	"California"
"WA"	"Washington"
"NY"	"New York"

"ABC Corp"	5 "ABC Corp" abc@abc.com
"XYZ Inc"	8 "XYZ Inc" xyz@xyz.com

`foreach`

Iterating a Dictionary Best Practices

Do:

Use `foreach` to iterate a dictionary

Avoid:

Avoid iterating through the elements

Iterate through the keys or values instead

Common C# Dictionaries by Namespace

System.Collections (.NET 1)

System.Collections.Generic

- Dictionary<TKey,TValue>
- SortedList<TKey,TValue>
- SortedDictionary<TKey,TValue>

Selecting an Appropriate Dictionary

Dictionary<T,V>

- Use most often
- Not sorted

SortedList<T,V>

- Sorted by key
- Faster when populating from sorted data

SortedDictionary <T,V>

- Sorted by key
- Faster when populating from unsorted data

Selecting an Appropriate Dictionary (cont)

System.Collections.ObjectModel	System.Collections.Specialized	System.Collections.Concurrent
• Appropriate for a reusable library • ReadOnlyDictionary • KeyedCollection	• Specialty collections • OrderedDictionary	• Thread-safe dictionary classes

Frequently Asked Questions

- When is it appropriate to use a generic dictionary?
  - Any time the application needs to manage a collection of things by key.
- What are the primary differences between a generic list and a generic dictionary?
  - A generic list contains elements accessible by index.
  - A generic dictionary contains elements with keys, accessible by key.

Frequently Asked Questions (cont)

- **What are the limitations of a dictionary key?**
  - Must be unique within the collection.
  - Must not be changed.
  - Cannot be null.
- **What is the difference between `foreach` and `for` when iterating through a dictionary?**
  - `for` is not useful.
  - `foreach` iterates all elements in a dictionary.
    - Iterate the elements, keys, or values.

Summary

- Declaring and Populating a Generic Dictionary
- Using Collection Initializers
- Initializing a Dictionary of Objects
- Retrieving an Element from a Generic Dictionary
- Iterating Through a Generic Dictionary
- Types of C# Dictionaries