Dictionaries

Kit Eason
www.kiteason.com
@kitlovesfsharp

hardcore dev and IT training

Dictionary Basics

- **Generic mapping from keys to values**
- **Create:**
  - `let capitals = new Dictionary<string,string>()`
- **Add values:**
  - `capitals.Add("United Kingdom","London")`
  - `capitals.Add("France", "Paris")`
- **Access values:**
  - `printfn "The capital of France is %s" capitals.["France"]`

Adding by Assignment

- Assigning using <- to the indexed value…
  - `capitals.["Spain"] <- "Madrid"`
- Adds if the value doesn't pre-exist
- Or updates if the value does pre-exist

Key	Value
United Kingdom	London
France	Paris

Key	Value
United Kingdom	London
France	Paris
Spain	Madrid

Dictionary Methods

- Count – how many key-value pairs?
- Keys – sequence of keys
- Values – sequence of values
- ContainsKey/ContainsValue – does the key or value exist?
- ContainsValue is *O(n)*
- ContainsKey approaches *O(1)*

Immutable Style

- Create and populate in one
- Never in an invalid state
- Use ‘dict’
  - `let dictionary = dict myValues`
  - `let dictionary = myValues |> dict`
- Input must consist of tuples

```
let capitals =
      [
         "United Kingdom", "London"
         "United States of America","Washington D.C."
         "France", "Paris"
      ] |> dict
```

Immutability by Exception!

- IDictionary returned by ‘dict’ has Add, Clear methods etc…
- …but they raise a NotSupportedException
- …at runtime!
- Consider using Map

In-depth Demos

- **Mutable style**
- **Immutable style**

49452	79.4	80	
49446	72.3	74.4	
49448	74.5	77.3	
49452	79.5	80.1	
49449	71.8	73.4	
49450	69.3	71	
49451	71.2	74.3	
49452	79.4	80.2	
49453	77.1	79.7	
49448	73.2	76.9	✓
49452	79.3	80.1	✓

In-depth Demo – Immutable Dictionary

- Store MD5 hashes of files in directory
- Is another file already there? (by MD5 hash)

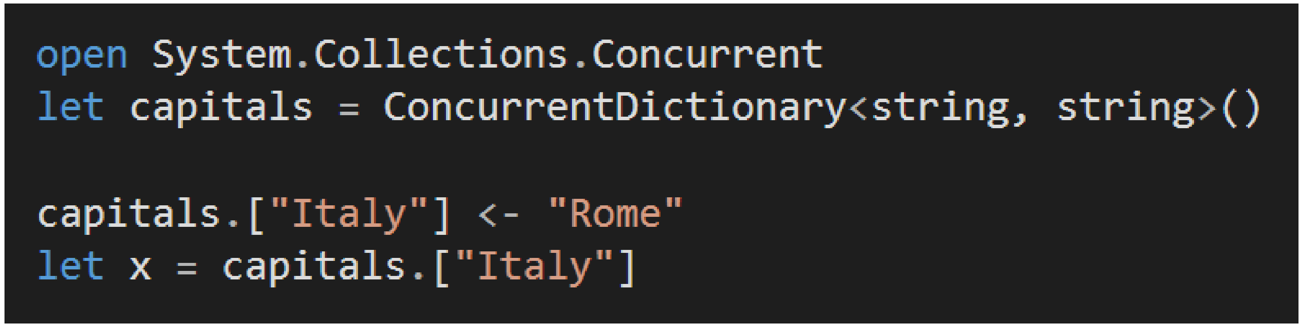
Dictionary and Concurrency

```
System.Collections.Generic.KeyNotFoundException: The
given key was not present in the dictionary
at System.Collections.Generic.Dictionary`2.get_Item(TKey
key)
```

- Put locks round every mutation, or…
- …ConcurrentDictionary to the rescue!

ConcurrentDictionary.AddOrUpdate

- **As Dictionary, access via .[index]**



```
open System.Collections.Concurrent
let capitals = ConcurrentDictionary<string, string>()

capitals.["Italy"] <- "Rome"
let x = capitals.["Italy"]
```

- **AddOrUpdate(key, value, updateDelegate)**
  - If key isn't in dictionary, add it using value
  - If key is in dictionary, call updateDelegate to get a value for update

ConcurrentDictionary Limitation

- **All methods are atomic and thread safe**
- **But are your delegates?**
  - AddOrUpdate()
  - GetOrAdd()

Summary

- **A generic .NET class, maps keys to values**
- **Use in mutable style**
  - new Dictionary<keytype, valuetype>()
  - dictionary.Add(key, value)
  - dictionary.[key] <- value
  - let x = dictionary.[key]
- **Use in immutable style**
  - collection |> dict
  - dictionary.[key]
- **Multi-threaded updates?**
  - Consider ConcurrentDictionary