generische Listen / Collection

Zwar braucht die Dictionary<T, K> weniger Zeit beim hinzufügen/ löschen von Werten, braucht die SortedList doch weniger Ramkapazitäten und ist im großen und ganzen schneller. Deutlich langsamer ist die Hashtable und SortedDictionary.

<T> = Typenparameter, erwartet wird ein Datentyp als Parameter. Beispiel string, int, object,…

<V, K> = siehe <T>, jedoch wegen besserer Lesbarkeit stellt dieser Typenparameter den Wert des **V**alue bzw. **K**ey da.

List<T>
[crayon-67678eedd64ae809292604/]
SortedList<V, K>
[crayon-67678eedd64b4350454909/]
Beinhaltet einen Key und Value. Über den Key lässt sich das Value herausfinden. Beispiel:
[crayon-67678eedd64b5673399840/]
liefert den Wert Boolean Wert True

Dictionary<V, K>
[crayon-67678eedd64b7170945830/]
ArrayList (Object)
[crayon-67678eedd64b8558185997/]
Hashtable (Object K, Object V)
[crayon-67678eedd64b9617802305/]
ObservableCollection
[crayon-67678eedd64ba595913083/]
Im Gegensatz zur List<T> nutzt die ObservableCollection die INotifyCollectionChanged Schnittstelle. Diese gibt eine Meldung, sobald sich in der Collection etwas geändert hat.

Sehr Sinnvoll, wenn man diese Collection an ein Steuerelement per WPF binden möchte, aktualisiert sich das Element so automatisch. Ein Nachteil ist jedoch, dass man die Liste nicht aus der Liste suchen/sortieren kann. Hier kann man nachlesen, wie man dies doch mit einbauen kann -> Link

List<Tuble<T,A,B>

Bisher hatten wir immer nur die Möglichkeit über Key / Value ein Pärchen vom Eintrag zu bilden. Manchmal kommt man aber in die Situation wo man nicht das Key/Value Prinzip haben möchte, oder mehr als 2 Argumente übergeben möchte. Da kommt das seit .NET 4.0 eingeführte Tuble ins Spiel. Die einzelnen Einträge werden dann als Item1, Item2 usw. innerhalb des Eintrags geführt.
[crayon-67678eedd64bc986484601/]

Quelle: http://blog.bodurov.com/Performance-SortedList-SortedDictionary-Dictionary-Hashtable/