Collections Reference

Namespaces: `SolidCore.Collections`, `SolidCore.Query`

The `SolidCore.Collections` assembly provides high-performance managed collections and a zero-allocation query (LINQ-like) engine.

- Managed Collections — `FastList<T>`, `IndexMap<TKey, TValue>`
- Query Engine (LINQ-like) — `QueryExtensions`, `Where`, `Select`
- Enumerators — `IRefEnumerator<T>`

Managed Collections

`FastList<T>`

Growable array with ref-return indexing and O(1) unordered removal. Not thread-safe.

Properties

- `int Count` — Number of elements.
- `int Capacity` — Current backing-array length.
- `bool IsEmpty` — True if empty.

Methods

- `FastList(int capacity)` — Constructor.
- `ref T this[int index]` — Reference to the element at `index`. Bounds-checked.
- `void Add(T item)` / `void Push(T item)` — Appends `item`.
- `T Pop()` — Removes and returns the last element. Throws if empty.
- `bool TryPop(out T item)` — Pops the last element; returns false if empty.
- `void RemoveLast()` — Discards the last element.
- `void Clear()` — Clears all elements to `default(T)`.
- `void RemoveAt(int index)` — Removes the element at `index` by swapping in the last. O(1), does not preserve order.
- `bool Contains(T item)` — True if `item` is present. O(n).
- `bool Remove(T item)` — Removes the first occurrence of `item` via unordered removal. O(n).
- `ReadOnlySpan<T> AsSpan()` — Read-only view over the live elements.
- `void EnsureCapacity(int capacity)` — Grows the backing array if needed.
- `Enumerator GetEnumerator()` — Ref-returning struct enumerator; zero-alloc.

IndexMap<TKey, TValue>

Thin keyed store over `System.Collections.Generic.Dictionary<TKey, TValue>` exposing a stable indexing API.

Properties

- `TValue this[TKey key]` — Gets or sets the value. Getter throws if absent.
- `IEnumerable<KeyValuePair<TKey, TValue>> Items` — All key/value pairs.
- `int Count` — Number of entries.

Methods

- `IndexMap()` / `IndexMap(int capacity)` — Constructors.
- `bool TryGetValue(TKey key, out TValue value)` — Gets the value; returns false if absent.
- `void Set(TKey key, TValue value)` — Inserts or overwrites the value.
- `bool Remove(TKey key)` — Removes the entry; returns true if it existed.
- `void Clear()` — Removes all entries.
- `bool ContainsKey(TKey key)` — True if `key` is present.

Query Engine (LINQ-like)

The `SolidCore.Query` namespace provides zero-allocation, ref-returning versions of common LINQ operators.

`QueryExtensions` **(static)**

- `static WhereEnumerator<T> Where<T>(this FastList<T> list, Func<T, bool> predicate)` — Filters a `FastList`. Zero-alloc, ref-returning.
- `static SelectEnumerator<T, TResult, ...> Select<T, TResult>(this FastList<T> list, Func<T, TResult> selector)` — Projects elements. Zero-alloc.
- `static SelectEnumerator<T, TResult, ...> Select<T, TResult>(this WhereEnumerator<T> source, Func<T, TResult> selector)` — Enables `Where().Select()` chaining.

Enumerators

`IRefEnumerator<T>`

Minimal ref-returning enumerator contract; enables zero-copy, in-place iteration and composable query pipelines.

- `bool MoveNext()` — Advances to the next element.
- `ref T Current` — Reference to the current element.

Specialized Enumerators

- `WhereEnumerator<T>` – Filtering ref-enumerator; lazy, zero-alloc.
- `SelectEnumerator<TSource, TResult, TEnumerator>` – Projecting enumerator; lazy, zero-alloc. Result returned by value.