API Reference

Welcome to the API documentation for the **BelNytheraSeiche.TrieDictionary** library. This section provides detailed information on all public classes and interfaces.

Below are links to the main components. For a complete list of all namespaces and types, please use the navigation sidebar.

High-Level Dictionaries

These are the primary, high-level classes for creating and using dictionaries.

- **KeyRecordDictionary**

  The abstract base class for all dictionary implementations, defining the core API.

- **DoubleArrayDictionary**

  Recommended for most cases. A **mutable** dictionary based on a Double-Array trie. Ideal for scenarios where keys need to be added or removed dynamically.

- **BitwiseVectorDictionary**

  Recommended for read-only scenarios. A **read-only**, high-performance dictionary based on a compact, array-based trie. A good balance of speed and memory.

- **LoudsDictionary**

  A **read-only** dictionary based on a pure LOUDS trie, optimized for extreme memory efficiency.

- **DirectedAcyclicGraphDictionary**

  A **read-only** dictionary based on a DAWG (Directed Acyclic Word Graph), which compresses a standard trie by merging common nodes.

Low-Level Storage

These classes form the building blocks of the high-level dictionaries.

- **PrimitiveRecordStore**

  A low-level, append-only store for raw byte records, used for persistent data.

- **BasicRecordStore**

  The abstract base class for various in-memory key-value stores.

Self-Balancing Binary Tree Stores

These classes are concrete implementations of `BasicRecordStore` and provide different strategies for in-memory key-value storage with sorted enumeration.

- ### AVLTreeRecordStore

  An implementation based on the classic AVL tree.

- ### AATreeRecordStore

  An implementation based on the AA tree, a simplified variation of a Red-Black tree.

- ### TreapRecordStore

  An implementation using a Treap, which uses randomized priorities to maintain balance.

- ### ScapegoatTreeRecordStore

  An implementation based on the Scapegoat tree, which rebuilds subtrees only when they become too unbalanced.