what-are-keymaps)

Keymap

This hashmap-like storage structure uses generic typed keys to store objects.

What are Keymaps?

AKeymap isSecret toolkit hashmap-like storage structure that uses generic typed keys to store objects. It allows iteration with paging over keys and/or items without guaranteed ordering, although the order of insertion is preserved until you remove objects.

An example use-case for a keymap is if you want to contain a large amount of votes and iterate over them in the future. Since iterating over large amounts of data at once may be prohibitive, a keymap allows you to specify the amount of data that will be returned in each page. We will implement this voting example below to show how keymaps can be utilized in your own projects.

Importing the subpackage

To import this package (and also the packages that we will be using for unit tests), add the following dependencies to yourCargo.toml file

```

Copy cosmwasm-std={ package="secret-cosmwasm-std", version="1.1.10"} cosmwasm-storage={ package="secret-cosmwasm-storage", version="1.1.10"} secret-toolkit-storage="0.9.0"

```

Configuring the testing environment

To import and initialize a keymap, use the following packages in your test environment:

```

Copy

# [cfg(test)]

modtests { usesecret_toolkit_storage::{Item,Keymap}; useserde::{Deserialize,Serialize}; usecosmwasm_std::testing::MockStorage; usecosmwasm_std::StdResult; };

```

Now let's write our first test!

Inserting key-value pairs into a Keymap

Let's start by creating a function thatinserts key-value pairs into a keymap. This code defines a test function calledtest_keymap_perf_vote_insert() that creates a newKeymap that mapsVec keys toString values, and then inserts 1000 key-value pairs into theKeymap . Each key is aVec containing an integer from 0 to 999, and each value is aString containing the text "I vote yes".

```

Copy

# [test]

fntest_keymap_perf_vote_insert()->StdResult<()> { letmutstorage=MockStorage::new();
```

lettotal_items=1000;

letkeymap:Keymap,String>=Keymap::new(b"votes");

foriin0..total_items { letkey:Vec=(iasi32).to_be_bytes().to_vec(); letvalue=String::from("I vote yes"); keymap.insert(&mutstorage,&key,&value)?; }

assert_eq!(keymap.get_len(&storage)?,1000);

Ok(()) }

```

This test is passing! Which means that it asserts that theKeymap contains 1000 key-value pairs, which map a number from 0 - 999 with the string "I vote yes."

Iterating with Keymaps

There are two methods that create an iterator in Keymap. These are.iter and.iter_keys .iter_keys only iterates over the keys whereasiter iterates over (key, item) pairs. Let's useiter to test the iterator functionality of aKeymap that mapsVec keys to a structVote.

We are going to create a newKeymap that mapsVeckeys to a structVote that has two fields,vote andperson . It then inserts two key-value pairs into theKeymap , where the keys are the byte vectorsb"key1".to_vec() andb"key2".to_vec() , and the values areVote structs containing information about the vote and the person who cast it.

We then useiter() to check that the size of the iterator is 2, which means there are two key-value pairs in theKeymap:

```

Copy

[test]

fntest_keymap_votes_iter()->StdResult<()> { letmutstorage=MockStorage::new();

letalice="alice"; letbob="bob";

[derive(Serialize,Deserialize,Eq,PartialEq,Debug,Clone)]

structVote{ vote:String, person:String, }

letkeymap:Keymap,Vote>=Keymap::new(b"votes"); letvote1=Vote{ vote:"I vote yes".to_string(), person:alice.to_string(), }; letvote2=Vote{ vote:"I vote no".to_string(), person:bob.to_string(), };

keymap.insert(&mutstorage,&b"key1".to_vec(),&vote1)?; keymap.insert(&mutstorage,&b"key2".to_vec(),&vote2)?;

letmutx=keymap.iter(&storage)?; let(len, _)=x.size_hint(); assert_eq!(len,2);

assert_eq!(x.next().unwrap()?, (b"key1".to_vec(), vote1));

assert_eq!(x.next().unwrap()?, (b"key2".to_vec(), vote2));

Ok(()) }

```

Our test is passing! This means that the first element returned by the iterator matches the expected key-value pair forkey1 , and that the second element returned by the iterator matches the expected key-value pair forkey2 .

Additional Resources

For further examples demonstrating the usage of keymaps, refer to the Secret Toolkit repo here . Previous Singleton Next Append Store Last updated1 month ago
```