Vending Machine

An example project for writing Arbitrum Stylus programs in Rust using the stylus-sdk . It includes a Rust implementation of a vending machine Ethereum smart contract.

- distribute Cupcakes to any given address
- count Cupcakes balance of any given address

Here is the interface for Vending Machine.

interface

IVendingMachine

{ // Function to distribute a cupcake to a user function

giveCupcakeTo (address userAddress)

external

returns

(bool) ;

// Getter function for the cupcake balance of a user function

getCupcakeBalanceFor (address userAddress)

external

view

returns

(uint) ; } Example implementation of the Vending Machine contract written in Rust.

src/lib.rs

note This code has yet to be audited. Please use at your own risk. //! //! Stylus Cupcake Example //! //! The program is ABI-equivalent with Solidity, which means you can call it from both Solidity and Rust. //! To do this, run cargo stylus export-abi. //! //! Note: this code is a template-only and has not been audited. //!

// Allow cargo stylus export-abi to generate a main function if the "export-abi" feature is enabled.

![cfg_attr(not(feature =

"export-abi"), no_main)] extern

crate

alloc ;

use

alloy_primitives :: { Address ,

Uint } ; // Import items from the SDK. The prelude contains common traits and macros. use

stylus_sdk :: alloy_primitives :: U256 ; use

stylus_sdk :: prelude :: * ; use

stylus_sdk :: { block , console } ;

// Define persistent storage using the Solidity ABI. //VendingMachine will be the entrypoint for the contract. sol_storage!

{

[entrypoint]

pub

struct

VendingMachine

{ // Mapping from user addresses to their cupcake balances. mapping (address => uint256) cupcake_balances ; // Mapping from user addresses to the last time they received a cupcake. mapping (address => uint256) cupcake_distribution_times ; }
}

// Declare that VendingMachine is a contract with the following external methods.

[public]

impl

VendingMachine

{ // Give a cupcake to the specified user if they are eligible (i.e., if at least 5 seconds have passed since their last cupcake). pub

fn

give_cupcake_to (& mut

self , user_address :

Address)

->

bool

{ // Get the last distribution time for the user. let last_distribution =

self . cupcake_distribution_times . get (user_address) ; // Calculate the earliest next time the user can receive a cupcake. let five_seconds_from_last_distribution = last_distribution +

U256 :: from (5) ;

// Get the current block timestamp. let current_time =

block :: timestamp ( ) ; // Check if the user can receive a cupcake. let user_can_receive_cupcake = five_seconds_from_last_distribution <=

Uint :: < 256 ,

4

:: from (current_time) ;

if user_can_receive_cupcake { // Increment the user's cupcake balance. let

mut balance_accessor =

self . cupcake_balances . setter (user_address) ; let balance = balance_accessor . get ( )

+

U256 :: from (1) ; balance_accessor . set (balance) ;

// Update the distribution time to the current time. let

mut time_accessor =

self . cupcake_distribution_times . setter (user_address) ; let new_distribution_time =

block :: timestamp ( ) ; time_accessor . set (Uint :: < 256 ,

4

:: from (new_distribution_time)) ; return

true ; }

else

{ // User must wait before receiving another cupcake. console! ("HTTP 429: Too Many Cupcakes (you must wait at least 5 seconds between cupcakes)") ; return

false ; } }

// Get the cupcake balance for the specified user. pub

fn

get_cupcake_balance_for (& self , user_address :

Address)

->

Uint < 256 ,

4

{ // Return the user's cupcake balance from storage. return

self . cupcake_balances . get (user_address) ; } }

Cargo.toml

[package] name

=

"stylus_cupcake_example" version

=

"0.1.7" edition

=

"2021" license

=

"MIT OR Apache-2.0" keywords

=

["arbitrum" ,

"ethereum" ,

"stylus" ,

"alloy"]

[dependencies] alloy-primitives

=

"=0.7.6" alloy-sol-types

=

"=0.7.6" mini-alloc

=

"0.4.2" stylus-sdk

=

"0.6.0" hex

=

"0.4.3"

[dev-dependencies] tokio

=

{

version

=

"1.12.0" ,

features

=

["full"]

} ethers

=

"2.0" eyre

=

"0.6.8"

[features] export-abi

=

["stylus-sdk/export-abi"]

[lib] crate-type

=

["lib" ,

"cdylib"]

[profile.release] codegen-units

=

1 strip

=

true lto

=

true panic

=

"abort" opt-level

=

"s" Edit this page Previous Multi Call Next Gas & Ink Pricing