

## Step by Step Guide

Exercise

Errors >

The new Keyword >

Contract to Contract  
Interactions >

Events >

Address and Payable >

## Development with Foundry

Deploying a smart contract  
using Foundry

Foundry: Setting up Foundry  
with Base

Multiple Inheritance Guide

Abstract Contracts

Abstract Contracts Guide

Exercise

## Imports

# Imports

 Copy page

Learn to import code into your contract.

In this lesson, we'll learn how to import code written by others into your contracts.

We'll also explore the [OpenZeppelin](#) library of smart contracts.

## Objectives

By the end of this lesson you should be able to:

Import and use code from another file

Utilize OpenZeppelin contracts within Remix

[Step by Step Guide](#)[Exercise](#)

## Development with Foundry

Deploying a smart contract using Foundry

Foundry: Setting up Foundry  
with Base

Multiple Inheritance Guide

Abstract Contracts

Abstract Contracts Guide

[Exercise](#)

# OpenZeppelin

OpenZeppelin has a robust [library](#) of well-documented smart contracts. These include a number of standard-compliant token implementations and a suite of utilities. All the contracts are audited and are therefore safer to use than random code you might find on the internet (you should still do your own audits before releasing to production).

## Docs

The [docs](#) start with installation instructions, which we'll return to when we switch over to local development. You do **not** need to install anything to use these contracts in Remix.

Find the documentation for the `EnumerableSet` under *Utils*. This library will allow you to create [sets](#) of `bytes32`, `address`, and `uint256`. Since they're enumerated, you can iterate through them. Neat!

## Implementing the OpenZeppelin EnumerableSet

Create a new file to work in and add the `pragma` and license identifier.

In Remix, you can import libraries directly from GitHub!

```
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/
```

## Development with Foundry

Deploying a smart contract  
using Foundry

Foundry: Setting up Foundry  
with Base

Multiple Inheritance Guide

Abstract Contracts

Abstract Contracts Guide

Exercise

You should see `EnumerableSet.sol` pop into your workspace files, nested deeply in a bunch of folders.

## Trying It Out

Add a contract called `SetExploration`. Review the extensive comments within the contract itself.

To use the `EnumerableSet`, you need to use the `using` keyword. This directive attaches all of the library methods to the type. Doing so allows you to call the method on the variable with dot notation, and the variable itself will be supplied as the first argument.

Follow the pattern in the example in the comments, but name the variable `visitors`:

```
using EnumerableSet for EnumerableSet.AddressSet;  
  
EnumerableSet.AddressSet private visitors;
```



Add a function called `registerVisitor` that makes use of the library's `add` function to add the sender of the message to the `visitors` set.



There's also an `_add` function, which is private.

[Step by Step Guide](#)[Exercise](#)

## Reveal code

```
function registerVisitor() public {  
    visitors.add(msg.sender);  
}
```



Add another function to return the `numberOfVisitors`. Thanks to `using`, this can

 Search... Ctrl K[GitHub](#)[Support](#)[Base Build](#)[Get Started](#)[Base Chain](#)[Base Account](#)[Base App](#)[Mini Apps](#)[OnchainKit](#)[Cookbook](#)[Showcase](#)[Learn](#)

Deploying a smart contract  
using Foundry

Foundry: Setting up Foundry  
with Base

[Multiple Inheritance Guide](#)[Abstract Contracts](#)[Abstract Contracts Guide](#)[Exercise](#)

```
function numberOfVisitors() public view returns (uint) {  
    return visitors.length();  
}
```

## Conclusion

In this lesson, you imported a library from [OpenZeppelin](#) and implemented some of its functions. You also learned how to use the `using` keyword.

Was this page helpful?

 [Yes](#) [No](#) [Suggest edits](#) [Raise issue](#) [Imports Overview](#) [Exercise](#)

## Development with Foundry

Deploying a smart contract  
using Foundry

Foundry: Setting up Foundry

with Base

**basedocs**

e.org

Blog

Privacy Policy

Terms of Service

Cookie Policy