Initialization in Objective-C

Previous Next

Every Ice-based application needs to initialize the Ice run time, and this initialization returns an `ICECommunicator` object.

An `ICECommunicator` is a local Objective-C object that represents an instance of the Ice run time. Most Ice-based applications create and use a single `ICECommunicator` object, although it is possible and occasionally desirable to have multiple `ICECommunicator` objects in the same application.

You initialize the Ice run time by calling `createCommunicator` on class `ICEUtil`. `createCommunicator` returns an instance of type `id<ICECommunicator>`:

Objective-C

```
id<ICECommunicator> communicator = [ICEUtil createCommunicator:&argc argv:argv];
```

`createCommunicator` accepts a *pointer* to `argc` as well as `argv`. The class method scans the argument vector for any command-line options that are relevant to the Ice run time; any such options are removed from the argument vector so, when `createCommunicator` returns, the only options and arguments remaining are those that concern your application. If anything goes wrong during initialization, `createCommunicator` throws an exception.

Before leaving your `main` function, you must call `Communicator::destroy`. The `destroy` method is responsible for finalizing the Ice run time. In particular in a server, `destroy` waits for any operation implementations that are still executing to complete. In addition, `destroy` ensures that any outstanding threads are joined with and reclaims a number of operating system resources, such as file descriptors and memory. Never allow your `main` function to terminate without calling `destroy` first.

The general shape of our `main` function is therefore:

Objective-C

```
#import <objc/Ice.h>

int
main(int argc, char* argv[])
{
    int status = EXIT_SUCCESS;
    @autoreleasepool
    {
        id<ICECommunicator> communicator = nil;
        @try
        {
            communicator = [ICEUtil createCommunicator:&argc argv:argv];
            ...
        }
        @catch(NSException* ex)
        {
            NSLog(@"%@", ex);
            status = EXIT_FAILURE;
        }

        [communicator destroy];
    }
    return status;
}
```

Back to Top ^

See Also

- Communicators
- Communicator Initialization

Previous Next