Initialization in Python

Previous Next

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

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

You initialize the Ice run time by calling `Ice.initialize`, for example:

Python

```
import sys, Ice

communicator = Ice.initialize(sys.argv)
```

`Ice.initialize` accepts the argument list that is passed to the program by the operating system. The function scans the argument list for any command-line options that are relevant to the Ice run time; any such options are removed from the argument list so, when `Ice.initialize` returns, the only options and arguments remaining are those that concern your application. If anything goes wrong during initialization, `initialize` throws an exception.

Before leaving your program, you must call `Communicator.destroy`. The `destroy` method is responsible for finalizing the Ice run time. In particular, in an Ice 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 program to terminate without calling `destroy` first.

The general shape of our application becomes:

Python

```
import sys, traceback, Ice

status = 0
communicator = None
try:
    # correct but suboptimal, see below
    communicator = Ice.initialize(sys.argv)
    # ...
except:
    traceback.print_exc()
    status = 1

if communicator:
    # correct but suboptimal, see below
    communicator.destroy()

sys.exit(status)
```

This code is a little bit clunky, as we need to make sure the communicator gets destroyed in all paths, including when an exception is thrown.

Fortunately, `Communicator` implements the Python context manager protocol: this allows us to call `initialize` in a `with` statement, which destroys the communicator automatically, without an explicit call to the `destroy` method.

The preferred way to initialize the Ice run time in Python is therefore:

Python

```
import sys, Ice

with Ice.initialize(sys.argv) as communicator:
    # ...

# communicator is destroyed automatically at the end of the 'with' statement
```

Back to Top ^

See Also

- Communicators
- Communicator Initialization
- Application Helper Class

Previous

Next