Initialization in Java Compat

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 Java object that represents an instance of the Ice run time. Most Ice-based applications create and use a single `Communicato` 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.Util.initialize`, for example:

Java Compat

```
public static void main(String[] args)
{
    Ice.Communicator communicator = Ice.Util.initialize(args);
    ...
}
```

`Util.initialize` accepts the argument vector that is passed to `main` by the operating system. The method scans the argument vector for any command-line options that are relevant to the Ice run time. If anything goes wrong during initialization, `Util.initialize` throws an exception.

> The semantics of Java arrays prevents this simple `Util.initialize` from modifying the argument vector. You can use another overload of `Util.initialize` to receive an argument vector with all Ice-related arguments removed.

Before leaving your `main` method, 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 `main` method to terminate without calling `destroy` first.

The general shape of our `main` method becomes:

Java Compat

```
public class App
{
    public static void main(String[] args)
    {
        int status = 0;
        Ice.Communicator communicator = null;
        try
        {
            // correct but suboptimal, see below
            communicator = Ice.Util.initialize(args);
            // ...
        }
        catch(Exception e)
        {
            e.printStackTrace();
            status = 1;
        }

        if(communicator != null)
        {
            // correct but suboptimal, see below
            communicator.destroy();
        }
        System.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, the `Communicator` interface implements `java.lang.AutoCloseable`: this allows us to call `initialize` in a try-with-resources statement, which closes (destroys) the communicator automatically, without an explicit call to the `destroy` method.

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

Java Compat

```
public class App
{
    public static void main(String[] args)
    {
        int status = 0;
        try(Ice.Communicator communicator = Ice.Util.initialize(args))
        {
            // ...
        } // communicator is destroyed automatically here
        System.exit(status);
    }
}
```

Back to Top ^

See Also

- Communicators
- Communicator Initialization
- Application Helper Class

Previous Next