Using Cookies with Servant Locators

Occasionally, it can be useful for a servant locator to pass information between `locate` and `finished`. For example, the implementation of `locate` could choose among a number of alternative database backends, depending on load or availability and, to properly finalize state, the implementation of `finished` might need to know which database was used by `locate`. To support such scenarios, you can create a cookie in your `locate` implementation; the Ice run time passes the value of the cookie to `finished` after the operation invocation has completed. The cookie must derive from `Ice::LocalObject` and can contain whatever state and member functions are useful to your implementation:

C++

```
class MyCookie : public virtual Ice::LocalObject {
public:
    // Whatever is useful here...
};

typedef IceUtil::Handle<MyCookie> MyCookiePtr;

class MyServantLocator : public virtual Ice::ServantLocator {
public:

    virtual Ice::ObjectPtr locate(const Ice::Current& c, Ice::LocalObjectPtr& cookie)
    {
        // Code as before...

        // Allocate and initialize a cookie.
        //
        cookie = new MyCookie(...);

        return new PhoneEntryI;
    }

    virtual void finished(const Ice::Current& c, const Ice::ObjectPtr& servant,
                          const Ice::LocalObjectPtr& cookie)
    {
        // Down-cast cookie to actual type.
        //
        MyCookiePtr mc = MyCookiePtr::dynamicCast(cookie);

        // Use information in cookie to clean up...
        //
        // ...
    }

    virtual void deactivate(const std::string& category);
};
```

See Also

- Servant Locators

Copyright © 2017, ZeroC, Inc.