ResourcesUserGuideIntro

1. Getting Started

1.1 Using Resources

The first thing is to get hold of / create is a ResourcesFactory for the **Resources** implementation you want to use...

```
        ResourcesFactory factory = new PropertyResourcesFactory();
        factory.setReturnNull(true);
```

N.B. See below for an explantion of the **return null** configuration option.

Then you can get a Resources instance (the factory may either create a new instance or provide a cached instance, depending on its implementation)...

```
        Resources resources = factory.getResources("Bar", "file:c:/myapp/foo/Bar");
```

Once you have the resources instance, you can then use it to retrieve *localized resources* using one of the five *content retrieval* methods provided...

```
        Object resource = resources.getObject("foo.key", locale, null);
        String resource = resources.getString("foo.key", locale, null);
        byte[] resource = resources.getBytes("foo.key", locale, null);
        Reader resource = resources.getReader("foo.key", locale, null);
        InputStream resource = resources.getInputStream("foo.key", locale, null);
```

1.1.1 Return Null Option

The **return null** configuration option indicates what the above five *content retrieval* methods should do when a resource key is not found - they *either* return a **`null`** value or throw a ResourcesKeyException.

Use the factory's write method for that property to configure your Resources instances for this option:

```
        factory.setReturnNull(true);
```

1.1.2 Resources Implementations

OK the next question is what **Resources implementation do I use?** You have two choices...

- Standard Implementations - Commons Resources comes with a set of standard implementations for **properties** files, **XML** documents, **database** resources and versions for use in a **Web Application**
- Creating an Implementation - create your own implementation.

1.2 Messages

The majority of what Commons Resources is about centers around the Resources type. However there are also two types defined for handling messages:

- Message - a message representation.
- MessageList - a set of Message objects.

There is also a convenience *wrapper* class, that provides message string lookups from a Resources instance, and parameter replacement:

- Messages - convenience class for handling messages.

See the Messages section in this User Guide for more details.