ResourcesUserGuideCreating

4. Creating a Resources Implementation

You can implement the Resources and ResourcesFactory interfaces directly, however, by far the easiest way is to use one of the **base** implementations provided.

The two **base** Resources classes are:

- ResourcesBase - implements Resources
  - **init()** - (optional) for initialization processing.
  - **getKeys()** - (required) return the set (Iterator) of keys in this Resources
  - **getObject(key, Locale, TimeZone)** - (required) retrieve the content for a key/Locale/Timezone
- CollectionResourcesBase - extends **ResourcesBase** and provides a mechanism for caching a set of *Map*s containing key/value pairs for a Locale.
  - **init()** - (optional) for initialization processing.
  - **getLocaleMap(baseUrl, Locale)** (required) returns a Map of key/value pairs for a Locale.

4.1 ResourcesBase

```
public class MyResources extends ResourcesBase {

    private String config;

    public MyResources(String name, config) {
        super(name);
        this.config = config;
    }

    public void init() {
        // do initialization here, if required
    }

    public Iterator getKeys() {
        // must implement this method
    }

    public Object getObject(String key, Locale locale, TimeZone timeZone) {
        // must implement this method
    }
}
```

4.2 CollectionsResourcesBase

```
public class MyResources extends CollectionsResourcesBase {

    public MyResources(String name, config) {
        super(name, config);
    }

    public void init() {
        // do initialization here, if required
    }

    public Object getLocaleMap(String baseUrl, Locale locale) {
        // must implement this method
    }
}
```

4.3 ResourcesFactory

To create a **factory** for your **Resources** implementation, extend the ResourcesFactoryBase ...

```
    public class MyResourcesFactory extends ResourcesFactoryBase {

        public MyResources() {
            super();
        }

        protected Resources createResources(String name, String config) {
            Resources resources = new MyResources(name);
            Resources res = new PropertyResources(name, config);
            resources.setReturnNull(isReturnNull());
            resources.init();
            return resources;
        }
    }
```

N.B for implementations to be used in a Web Application there is a WebappResourcesFactoryBase factory which includes a **ServletContext** property with appropriate read/write methods.
