# The ResourceBundle class

The ResourceBundle class is an abstract class in the java.util package.

You can implement classes that extend this abstract class.

You can get an instance of a ResourceBundle, by calling one of several static getBundle methods, on the ResourceBundle class.

The latter approach is dependent on resource data, either stored in a series of files, or provided by a service.

# ResourceBundle data in a .properties file

Data that's customized is often textual, in the form of user messages, button labels or menu items, but may contain other elements, such as images or audio components.

To date, the most common method of supplying data for a ResourceBundle, is using the properties file.

This is a simple text file, containing key value pairs.

The key is a string, a name to be used when the data is requested, and the value is also a single text literal.

The properties file name includes a base name, called the bundle name, and some part of the Locale identifier, and ends with the extension .properties.

Here, I'm showing an example of a base TextMessages properties file.

**TextMessages.properties**

```
yes = yes
no = no
hello = hello
world = world
```

# ResourceBundle data

You'll create additional properties files, to support other languages.

You can think of a bundle as a series of files that have the same base bundle name, but are differentiated by Locale specifics.

If you need to support the spanish language for example, you might create a TextMessages_es.properties file, with spanish language text literals for yes, no, hello and so on.

# ResourceBundle data

I show an example of this file here.

Notice that the keys for both these properties files are in English, and are the same for both files.

Whatever language you choose for your keys is up to you, but they do need to be consistent across files, so that the key can be used to look up the value.

Of course, there are many language dialects, so you may need to also provide additional language variations for these, which can be done by including country, script and or a variant.

**TextMessages_es.properties**

```
yes = si
no = no
hello = hola
world = mundo
```

# Java's matching process to locate the best bundle

Java has specific rules for searching and matching one of these properties files, to a Locale.

These rules can be found in the ResourceBundle class documentation.

Here, I include the link to this information.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ResourceBundle.html#default_behavior

# ResourceBundle - alternatives to .properties files

You're not limited to using properties files for your internationalization support.

You can extend the ResourceBundle class, or another related abstract class called the ListResourceBundle class. You'd have the subclasses house your data in code, or source it from another place.

You can use other file formats, such as xml, with a little extra configuration, by extending both the ResourceBundle and ResourceBundle.Control classes.

You can find instructions for this in the ResourceBundle.Control API documentation, the link is displayed here.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ResourceBundle.Control.html

Alternately, you can make a call to a service provider.

You can also mix and match any of these methods.