To Read Properties File Using Java Owner API (Static Configuration Manager)

What?

Java Owner is a library that simplifies the process of reading and managing configuration properties in Java applications. Owner Configuration supports maintaining the multiple properties files to handle any type of data that requires frequent changes.

Why?

Maintainability: It makes updates and changes easy without altering your code.

Centralization: All your configuration is in one place, simplifying management.

Version Control: It helps track changes effectively, making it easy to roll back if something goes wrong.

Security: You can control who has access to configuration data, which is crucial for sensitive information.

How?

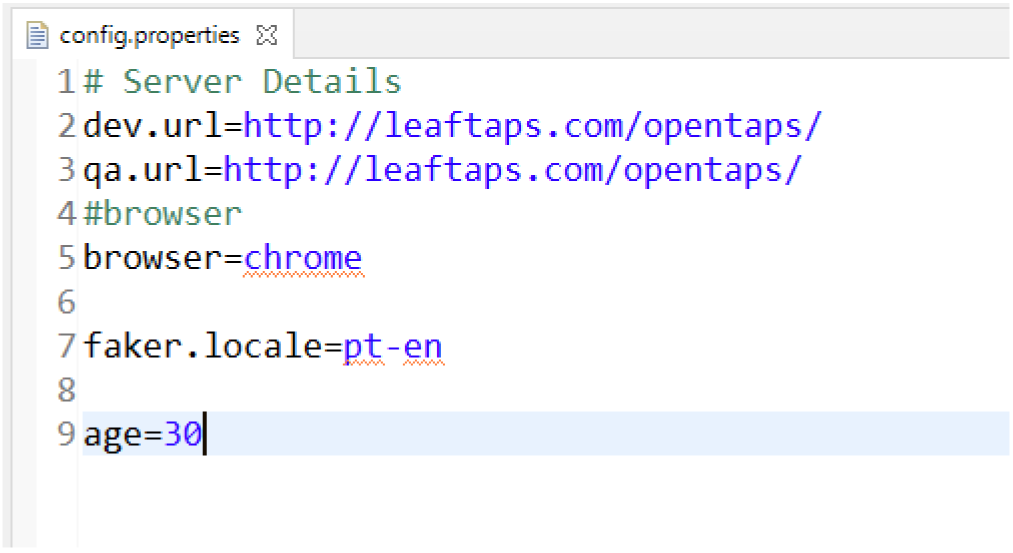
Step-by-step process:

Step 1: Pre–Requisite to integrate the Owner API with Selenium

Add the Owner library in pom.xml

```
<dependency>
        <groupId>org.aeonbits.owner</groupId>
        <artifactId>owner</artifactId>
        <version>1.0.8</version>
    </dependency>
```

Step 2: Create Properties files based on the environment or type of data. The file name should be .properties extend.



config.properties

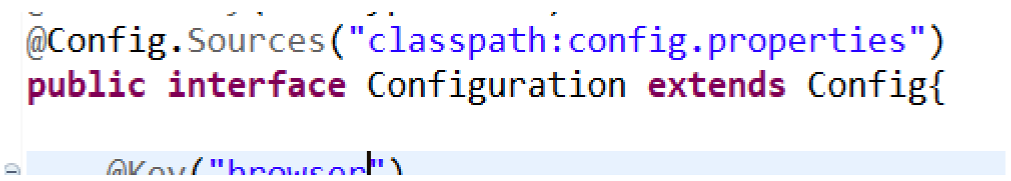
```
# Server Details
dev.url=http://leaftaps.com/opentaps/
qa.url=http://leaftaps.com/opentaps/
#browser
browser=chrome

faker.locale=pt-en

age=30
```

Step 3: Create an interface as Configuration which extends Owner interface Config to set the keys to read the data from the properties files

Step 4: Use @Config and specify the source from which to load the properties file. It has to be specified in a URI string format.



```
@Config.Sources("classpath:config.properties")
public interface Configuration extends Config{

    @Key("browser")
```

Step 5: Create abstract methods to read the value from the properties file using the @Key of config interface

```
@Key("browser")
String browser();

@Key("dev.url")
String devUrl();

@Key("age")
int getAge();
```

Step 6: Create a concrete class as ConfigurationManager and declare a static method to create an instance for the interface.

i) Create a static method

ii) Inside method call the ConfigCache.getOrCreate and pass constructor as your interface class and get the local variable

Eg: Configuration orCreate = ConfigCache.getOrCreate(Configuration.class);

iii) Return the local variable method name.

iv) Go to the interface and mention your classpath before the interface using @Config.Sources

```
public static Configuration configProperties() {
		return ConfigCache.getOrCreate(Configuration.class);

	}
```