Data Management

In this lesson, we will learn how to manage test data that are specific to the environment that we are running our test suite against.

Need for `DataManager`

Oftentimes, we will be using some static test data that is specific to an environment. The environment could be staging, production, development, etc. For maintaining the environment, we can have a separate class for managing that. This allows us to manage test data in an environment-agnostic way.

Creating `DataManager`

We can store the test data in any file format. For demonstration purposes, `.properties` file is considered.

```
import java.io.IOException;
import java.io.InputStream;
import java.net.URL;
import java.util.Collections;
import java.util.List;
import java.util.Properties;

public class DataManager {

    private static final Properties PROPERTIES = new Properties();
    private static final String ENV = ConfigurationManager.getInstance().g
etProperty("env");

    private DataManager() throws IOException {
        PROPERTIES.load(getInputStream("env-test-data.properties"));
    }

    private static DataManager manager;

    public static DataManager getInstance() {

        if (manager == null) {
            synchronized (ConfigurationManager.class) {
                if (manager == null) {
                    try {
                        manager = new DataManager();
                    } catch (IOException e) {
                    }
                }
            }
        }
        return manager;
    }

    public String getString(String name) {
        String key = ENV + "." + name;
        return System.getProperty(key, PROPERTIES.getProperty(key));
    }

    private InputStream getInputStream(String file) {

        try {
            List<URL> urls = Collections.list(Thread.currentThread().getCo
ntextClassLoader().getResources(file));
            return urls == null || urls.isEmpty() ? null : urls.get(0).ope
nStream();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

In the above code snippet, we are:

- creating a singleton instance of `DataManager` class in the `getInstance` method.
- reading properties file from `src/test/resources/env-test-data.properties` in the constructor of the class.
- fetching the property key's value from the `System.properties`. If not present, then fetch from the properties from `src/test/resources/env-test-data.properties`. This gives us the flexibility to override the properties from the command line as well.

Creating a test data file

`env-test-data.properties` should to be created under `src/test/resources`, as it is a test resource, so that it can be read from classpath when running the test suite.

```
stage.user_email = test_user@example.com

prod.user_email = test_user@example.com
```

Here, we prefix the test data key with the environment and set their respective values.

Usage

Assuming `env` (or an environment that is passed as configuration from command line as `-Denv=stage`) is set to some value and the respective values exist in `src/test/resources/env-test-data.properties` in the format expected and discussed previously, we can access the value of the `env` test data using the following code.

```
String userEmail = DataManager.getInstance().getString("user_email");
```

In the next lesson, we will learn how to put all the concepts learned till now and design a complete UI framework.