6.17 Mẫu thiết kế TEMPLATE

6.17.1 Mở đầu

Mẫu thiết kế Template là mẫu hành vi và như tên nó đề xuất, nó cung cấp bản nháp template hoặc cấu trúc của một thuật toán mà được sử dụng bởi user. Người sử dụng cung cấp cài đặt của riêng họ mà không thay đổi cấu trúc của thuật toán.

Sẽ dễ hiểu hơn mẫu này thông qua một ví dụ. Chúng ta sẽ hiểu kịch bản trong mục này và sẽ cài đặt lời giải sử dụng mẫu Template trong mục sau nữa.

Đã khi nào bạn kết nối với cơ sở dữ liệu quan hệ sử dụng ứng dụng Java chưa. Thử nhắc lại một số bước quan trọng mà được yêu cầu để kết nối và chèn dữ liệu vào cơ sở dữ liệu. Trước hết, bạn cần một driver phù hợp với cơ sở dữ liệu mà chúng ta muốn kết nối với nó. Sau đó, chúng ta truyền một số chứng từ ủy nhiệm cho cơ sở dữ liệu, và chúng ta chuẩn bị lệnh, set dữ liệu cho lệnh chèn và insert sử dụng lệnh chèn. Cuối cùng ta đóng mọi kết nối và tùy chọn xóa mọi đối tượng kết nối.

Bạn cần phải viết mọi bước này bất kể cơ sở dữ liệu quan hệ của nhà cung cấp nào. Xem xét bài toán ở đó chèn dữ liệu nào đó vào các cơ sở dữ liệu khác nhau. Bạn cần đẩy dữ liệu từ một file CSV (file ở đó các giá trị được phân cách bởi dấu phảy) và phải chèn vào cơ sở dữ liệu MySQL. Một số dữ liệu đến từ file text và nó cần được chèn vào cơ sở dữ liệu Oracle. Sự khác biệt chỉ ở driver và dữ liệu, còn lại là các bước giống nhau, như JDBC cung cấp tập chung các giao diện để trao đổi với cơ sở dữ liệu quan hệ chuyên biệt của bất kỳ nhà cung cấp nào.

Chúng ta có thể tạo một bản nháp template mà sẽ thực hiện một số bước cho client và chúng ta sẽ để lại một số bước cho phép client cài đặt chúng theo cách của riêng họ. Tùy chọn, client có thể ghi đè hành vi mặc định của một số bước đã được định nghĩa trước.

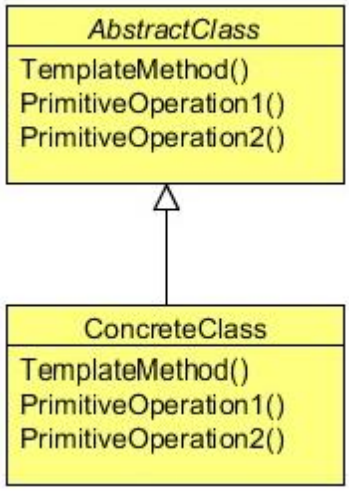
Bây giờ trước khi cài đặt code, chúng ta sẽ tìm hiểu thêm về mẫu thiết kế *Template.*

6.17.2 Mẫu Template là gì

Mẫu *Template* định nghĩa bộ khung của một thuật toán trong thao tác mà trì hoãn một số bước cho các lớp con. Phương thức *Template* cho phép các lớp con định nghĩa lại một số bước của một thuật toán mà không thay đổi cấu trúc của thuật toán.

Mẫu *Template Method* có thể được sử dụng trong một số tình huống khi mà có một thuật toán, một số bước của nó cần được cài đặt theo nhiều cách khác nhau. Trong kịch bản này, mẫu *Template Method* đề xuất giữ bộ khung của thuật toán trong một phương thức riêng biệt được tham chiếu như *Template Method* bên trong một lớp mà có thể được tham chiếu như *Template Class*, gác lại cài đặt chuyên biệt một số phần khác nhau (các bước mà có thể được cài đặt theo các cách khác nhau) của thuật toán cho các lớp con của lớp đó.

Lớp *Template* không cần thiết phải gác lại cài đặt cho các lớp con một cách toàn bộ. Thay vào đó, như một phần cung cấp bộ khung của thuật toán, lớp *Template* có thể cũng cung cấp một số phần cài đặt mà có thể được coi là bất biến qua các cài đặt khác nhau. Nó có thể ngay cả cung cấp cài đặt mặc định cho một số phần thay đổi, nếu nó phù hợp. Chi tiết chuyên biệt sẽ được cài đặt bên trong các lớp con. Kiểu cài đặt này loại bỏ cần thiết lặp lại code mà có nghĩa là khối lượng tối thiểu code được viết.



AbstractClass

- Định nghĩa các thao tác nguyên thủy trừu tượng mà các lớp con cụ thể sẽ cài đặt các bước của thuật toán

ConcreteClass.

- Cài đặt *Template Method* xác định bộ khung của thuật toán. *Template method* gọi các thao tác nguyên thủy là các thao tác được định nghĩa trong *AbstractClass* hoặc các thao tác của các đối tượng *ConcreteClass* khác
- Cài đặt các thao tác nguyên thủy để chạy.

6.17.3 Cài đặt mẫu thiết kế Template

Dưới đây chúng ta có thể thấy lớp *Template* kết nối được sử dụng để cung cấp *template* cho *client* kết nối và trao đổi thông tin với các cơ sở dữ liệu khác nhau.

```
package com.javacodegeeks.patterns.templatepattern;

public abstract class ConnectionTemplate {

        public final void run() {
                setDBDriver();
                setCredentials();
                connect();
                prepareStatement();
                setData();
                insert();
                close();
                destroy();
        }

        public abstract void setDBDriver();

        public abstract void setCredentials();

        public void connect() {
                System.out.println("Setting connection...");
        }

        public void prepareStatement() {
                System.out.println("Preparing insert statement...");
        }

        public abstract void setData();

        public void insert() {
                System.out.println("Inserting data...");
        }

        public void close() {
                System.out.println("Closing connections...");
        }

        public void destroy() {
                System.out.println("Destroying connection objects...");
        }
}
```

Lớp trừu tượng cung cấp các bước để kết nối, trao đổi và sau này là đóng kết nối. Tất cả các bước này là được yêu cầu theo thứ tự đó để hoàn thành công việc. Lớp này cung cấp cài đặt mặc định cho một số bước chung và gác lại các bước chuyên biệt như trừu tượng mà buộc *client* phải cung cấp cài đặt cho chúng.

Phương thức *setDBDriver* cần được cài đặt bởi người sử dụng để cung cấp *Driver* chuyên biệt cho cơ sở dữ liệu. Các tài liệu chứng thực có thể là khác nhau đối với các cơ sở dữ liệu khác nhau, do đó *setCredentials* cũng được để lại như trừu tượng cho người sử dụng cài đặt chúng.

Tương tự, kết nối với cơ sở dữ liệu sử dụng *JDBC API* và chuẩn bị một lệnh là thường gặp. Nhưng, dữ liệu có thể chuyên biệt, vì người sử dụng sẽ tạo ra nó, và phần các bước còn lại như chạy lệnh *insert*, đóng kết nối, hủy đối tượng là tương tự với mọi cơ sở dữ liệu, vì vậy cài đặt chúng sẽ giữ là chung bên trong *Template*.

Phương thức chính của lớp trên là phương thức *run*. Phương thức *run* được sử dụng để chạy các bước này theo thứ tự. Phương thức này được đặt là không đổi *final*, vì các bước cần phải được giữ an toàn và không được thay đổi bởi người sử dụng.

Hai lớp dưới đây mở rộng lớp *template* và cung cấp cài đặt chuyên biệt cho một số phương thức.

```
package com.javacodegeeks.patterns.templatepattern;

public class MySqLCSVCon extends ConnectionTemplate {

        @Override
        public void setDBDriver() {
                System.out.println("Setting MySQL DB drivers...");
        }

        @Override
        public void setCredentials() {
                System.out.println("Setting credentials for MySQL DB...");
        }

        @Override
        public void setData() {
                System.out.println("Setting up data from csv file....");
        }
}
```

Lớp trên được sử dụng để kết nối cơ sở dữ liệu MySQL và cung cấp dữ liệu bằng cách đọc file CSV.

```
package com.javacodegeeks.patterns.templatepattern;

public class OracleTxtCon extends ConnectionTemplate {

        @Override
        public void setDBDriver() {
                System.out.println("Setting Oracle DB drivers...");
        }
```

```
        @Override
        public void setCredentials() {
                System.out.println("Setting credentials for Oracle DB...");
        }

        @Override
        public void setData() {
                System.out.println("Setting up data from txt file....");
        }
}
```

Lớp trên được sử dụng để kết nối cơ sở dữ liệu Oracle và cung cấp dữ liệu bằng cách đọc file text.

Bây giờ chúng ta kiểm tra code trên.

```
package com.javacodegeeks.patterns.templatepattern;

public class TestTemplatePattern {

        public static void main(String[] args) {
                System.out.println("For MYSQL....");
                ConnectionTemplate template = new MySqLCSVCon();
                template.run();
                System.out.println("For Oracle...");
                template = new OracleTxtCon();
                template.run();
        }
}
```

Code trên cho kết quả đầu ra như sau:

```
For MYSQL....
Setting MySQL DB drivers...
Setting credentials for MySQL DB...
Setting connection...
Preparing insert statement...
Setting up data from csv file....
Inserting data...
Closing connections...
Destroying connection objects...

For Oracle...
Setting Oracle DB drivers...
Setting credentials for Oracle DB...
Setting connection...
Preparing insert statement...
Setting up data from txt file....
Inserting data...
Closing connections...
Destroying connection objects...
```

Đầu ra bên trên chỉ rõ mẫu *Template* làm việc để kết nối và trao đổi với cơ sở dữ liệu khác nhau sử dụng cách tương tự. Mẫu giữ code chung dưới một lớp và hỗ trợ tái sử dụng code. Nó đặt bộ khung và kiểm soát nó cho người sử dụng và cho phép người sử dụng mở rộng *template* theo thứ tự để cung cấp cài đặt chuyên biệt của chúng cho một số bước.

Bây giờ, nếu chúng ta phát triển ví dụ trên bằng cách bổ sung cơ chế ghi lưu lại. Nhưng một số người sử dụng code này không muốn tiện ích này, để cài đặt nó chúng ta có thể sử dụng một mẹo nhỏ. Mẹo là một phương thức bên trong lớp *Template* với hành vi mặc định, hành vi này có thể được sử dụng để thay đổi một số bước tùy chọn. Người sử dụng cần cài đặt phương thức này mà có thể tạo mẹo bên trong lớp *Template* để thay đổi các bước tùy chọn của thuật toán.

6.17.4 Tạo mẹo bên trong Template

Thử bổ sung vào code trên với mẹo nhỏ:

```
package com.javacodegeeks.patterns.templatepattern;

import java.util.Date;

public abstract class ConnectionTemplate {

        private boolean isLoggingEnable = true;

        public ConnectionTemplate() {
                isLoggingEnable = disableLogging();
        }

        public final void run() {
                setDBDriver();
                logging("Drivers set [" + new Date() + "]");
                setCredentials();
                logging("Credentails set [" + new Date() + "]");
                connect();
                logging("Conencted");
                prepareStatement();
                logging("Statement prepared [" + new Date() + "]");
                setData();
                logging("Data set [" + new Date() + "]");
                insert();
                logging("Inserted [" + new Date() + "]");
                close();
                logging("Conenctions closed [" + new Date() + "]");
                destroy();
                logging("Object destoryed [" + new Date() + "]");
        }

        public abstract void setDBDriver();

        public abstract void setCredentials();

        public void connect() {
                System.out.println("Setting connection...");
        }

        public void prepareStatement() {
                System.out.println("Preparing insert statement...");
        }

        public abstract void setData();

        public void insert() {
                System.out.println("Inserting data...");
        }

        public void close() {
                System.out.println("Closing connections...");
        }

        public void destroy() {
                System.out.println("Destroying connection objects...");
        }

        public boolean disableLogging() {
```

```
                return true;
        }

        private void logging(String msg) {
                if (isLoggingEnable) {
                        System.out.println("Logging....: " + msg);
                }
        }
}
```

Chúng ta đưa ra hai phương thức mới bên trong lớp *Template.* Phương thức *disableLogging* là mẹo mà trả về giá trị Bool. Mặc định, giá trị Bool *isLoggingEnable*, mà làm cho có khả năng ghi lưu lại là true. Người sử dụng ghi đè phương thức này nếu ghi lưu lại cần phải là không thể có cho code của anh ta. Phương thức khác là phương thức *Private* được sử dụng để ghi lưu thông điệp.

Lớp dưới đây cài đặt phương thức mẹo và trả về false, một số cái được tắt khỏi cơ chế ghi lưu cho công việc chuyên biệt đó.

```
package com.javacodegeeks.patterns.templatepattern;

public class MySqLCSVCon extends ConnectionTemplate {

        @Override
        public void setDBDriver() {
                System.out.println("Setting MySQL DB drivers...");
        }

        @Override
        public void setCredentials() {
                System.out.println("Setting credentials for MySQL DB...");
        }

        @Override
        public void setData() {
                System.out.println("Setting up data from csv file....");
        }

        @Override
        public boolean disableLogging() {
                return false;
        }
}
```

Hãy kiểm tra code trên

```
package com.javacodegeeks.patterns.templatepattern;

public class TestTemplatePattern {
        public static void main(String[] args) {
                System.out.println("For MYSQL....");
                ConnectionTemplate template = new MySqLCSVCon();
                template.run();
                System.out.println("For Oracle...");
                template = new OracleTxtCon();
                template.run();
        }
}
```

Lớp trên trả về kết quả sau.

```
For MYSQL....
Setting MySQL DB drivers...
Setting credentials for MySQL DB...
```

```
Setting connection...
Preparing insert statement...
Setting up data from csv file....
Inserting data...
Closing connections...
Destroying connection objects...

For Oracle...
Setting Oracle DB drivers...
Logging....: Drivers set [Sat Nov 08 23:53:47 IST 2014]
Setting credentials for Oracle DB...
Logging....: Credentails set [Sat Nov 08 23:53:47 IST 2014]
Setting connection...
Logging....: Conencted
Preparing insert statement...
Logging....: Statement prepared [Sat Nov 08 23:53:47 IST 2014]
Setting up data from txt file....
Logging....: Data set [Sat Nov 08 23:53:47 IST 2014]
Inserting data...
Logging....: Inserted [Sat Nov 08 23:53:47 IST 2014]
Closing connections...
Logging....: Conenctions closed [Sat Nov 08 23:53:47 IST 2014]
Destroying connection objects...
Logging....: Object destoryed [Sat Nov 08 23:53:47 IST 2014]
```

Bạn có thể thấy rõ trong đâu ra, rằng logging được tắt cho cài đặt MySQL, trong khi đó là bật cho cài đặt Oracle.

6.17.5 Khi nào sử dụng mẫu Template

Mẫu *Template Method* có thể được sử dụng trong các trường hợp sau:

- Cài đặt các phần bất biến của thuật toán một lần và gác lại cho các lớp con hành vi mà nó đa dạng.
- Khi hành vi chung giữa các lớp con cần được phải được tách ra và địa phương hóa trong lớp chung để tránh bị lặp code. Trước hết bạn định danh sự khác biệt trong code đã có và sau đó tách sự khác biệt ra thành thao tác riêng. Cuối cùng bạn thay code trì hoãn với các phương thức *Template* mà gọi một trong các thao tác mới đó.
- Để kiểm soát sự mở rộng của các lớp con. Bạn có thể định nghĩa phương thức *template* mà gọi thao tác "mẹo" tại các điểm chuyên biệt, bằng cách ấy cho phép mở rộng chỉ tại các điểm đó.