Prototype Design Pattern – Real life example

<https://examples.javacodegeeks.com/core-java/java-prototype-design-pattern-example/>

similar scenario where prototype design pattern actually comes in handy is Web applications.

In Java web applications, the developers utilise **Hibernate ORM framework or JPA annotations to enable prototyping of database objects.**

These objects are cached on startup of the web application and hence eliminates the time taken in fetching the data each time from the database. Such prototype objects are normally used for passing repeated data like the list of countries, states, cities, salutations, qualifications and others.

## Implementing Prototype Design Pattern

Let us now get going towards implementing the prototype design pattern manually. We will take a scenario of fetching a user list in a simple application. In such a scenario, the normal process of object creation is that the data will be created or fetched from the database each time the new object is created.

package com.classes;

public class User {

    String name;

    String email;

    public User(String name,String email) {

        this.name= name;

        this.email = email;

    }

    public String toString() {

        return "Name of user:"+this.name+"\nEmail of User:"+this.email;

    }

}

package com.service;

import java.util.ArrayList;

import com.classes.User;

public class UserService {

    ArrayList users;

    public UserService() {

        users = new ArrayList();

        users.add(new User("Albert","albert@gmail.com"));

        users.add(new User("Bob","bob@gmail.com"));

        users.add(new User("Gary","gary@gmail.com"));

        users.add(new User("Geno","geno@gmail.com"));

    }

    public ArrayList getUsers(){

        return users;

    }

}

package com.classes;

import java.util.ArrayList;

import java.util.List;

import com.service.UserService;

public class UserDetails implements Cloneable{

    private ArrayList userList;

    public UserDetails() {

        this.userList=new ArrayList<>();

    }

    public UserDetails(ArrayList users) {

        this.userList = users;

    }

    public void getUsers() {

        UserService s = new UserService();

        this.userList = s.getUsers();

    }

    public ArrayList getUsersList(){

        return userList;

    }

    @Override

    public Object clone() throws CloneNotSupportedException{

            List temp = new ArrayList();

            for(User s : this.getUsersList()){

                temp.add(s);

            }

            return new UserDetails((ArrayList) temp);

    }

}

As it can be seen, the class provides a function clone(). This function will be further used to obtain cloned objects. This function uses the existing list of users and creates a new list out of it. It does not go for a service call as it can be noted. This ensures that the application does not hit the database repeatedly and all the objects have the required data with just one hit.This saves the overhead of database calls and results in extremely good performance thereafter.  
Let us now create an implementation class with the main function.

## 5. Benefits of prototype Design Pattern

The prototype design pattern has been developed and is being used by a number of popular frameworks in Java. In fact, one of the bean scopes in the Spring framework is the prototype scope. This scope signifies the creation of a new object each time it is required. This pattern has a lot of benefits. A few of them are listed below:

* Provides a lower latency in creation of objects
* Provides easier way of cloning objects
* Reduces the load on the database for fetching the data
* Improves overall application performance
* Reduces the effort of cloning an object for the application developer
* Make the cloning code independent of the implementing classes
* Allows you to easily modify the existing class and its prototyping function which would not have been the case if cloning takes place elsewhere

## 6. Conclusion

The prototype design pattern is widely used and preferred in frameworks like Spring, Hibernate and Struts. It is of great use when an application demands repeated creation of similar objects. In the prototype design pattern, every class that requires prototyping, implements the Clonable interface and overrides the relevant function.

