Clean Code Development - Quality Seal

Improve your skills and share your expirience!

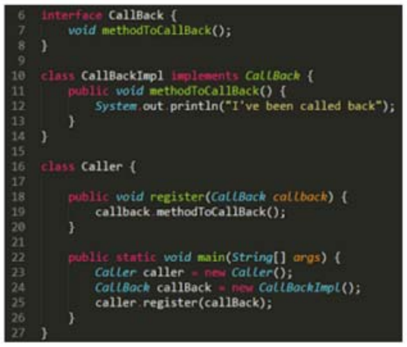
SATURDAY, OCTOBER 13, 2012

Understanding Callbacks with Java - Inversion of control

CALLBACK PATTERN IN JAVA ENVIRONMENT

Hi there! today i wanna share something with you, that it is very common and widely used in javascript for example. I'm speaking of callbacks. Do you know how and when this "pattern" is used? Do you really understand it in a java context (environment)? Well i was asking me also some of those questions and that's the reason i started to learn more about it. The ideia behind it is the **inversion of control** (abbreviated IoC). This paradigma describes the way frameworks work. It is also known as the **"Hollywood principle - Don't call me, we will call you"**

SIMPLIFIED CALLBACK PATTERN IN JAVA JUST TO UNDERSTAND IT. CONCRETE EXAMPLE FOLLOWS BELLOW.



```
interface CallBack {
    void methodToCallBack();
}

class CallBackImpl implements CallBack {
    public void methodToCallBack() {
        System.out.println("I've been called back");
    }
}

class Caller {

    public void register(CallBack callback) {
        callback.methodToCallBack();
    }

    public static void main(String[] args) {
        Caller caller = new Caller();
        CallBack callBack = new CallBackImpl();
        caller.register(callBack);
    }
}
```

```
interface CallBack {
    void methodToCallBack();
}

class CallBackImpl implements CallBack {
    public void methodToCallBack() {
        System.out.println("I've been called back");
    }
}

class Caller {

    public void register(CallBack callback) {
        callback.methodToCallBack();
    }

    public static void main(String[] args) {
        Caller caller = new Caller();
        CallBack callBack = new CallBackImpl();
        caller.register(callBack);
    }
}
```

Ok you may be asking you, when this usefull or may be asking you what's the difference between calling directly callback.methodToCallBack() right?

ANSWER: well, this example just shows to you how to construct such a callBack function when working in a java environment. Certainly it doesn't make it any sense to use it that way. Let's get a little deeper into a concrete useful example now.

The idea behind it is the **"INVERSION OF CONTROL"**. Let's take a timer as a realistic example. Let's supose that you know, that a specific timer supports callback functions every hour. Exactly it means, that every hour, the timer will call your registed call method function.

BLOG'S POPULARITY

360956

OTHER INTERESTING POSTS

ABOUT ME

Ricardo Ferreira

Application Engineer, Visionary and Ideologist for a better World.

View my complete profile

WHO ARE JOINING US?

Join this site
with Google Friend Connect

Members (26) More »

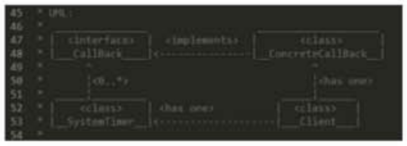
Already a member? Sign in

TAGS / LABELS

howto (36) Java (33) android (33) clean code (26) design pattern (24) Almanac (9) best practise (9) Eclipse (8) app (5) Database (4) design (4) junit (4) pattern (4) reusable (4) singleton (4) DAO (3) Data Transfer Object (3) SQLite (3) agility (3)

CONCRETE EXAMPLE:

Let's say we wanna update the time of a website every hour. Here is the UML of the following example:



```
 * UML:
 *
 *  ________________               _______________________
 * | <interface>    |  <implements> |       <class>         |
 * |__CallBack______|<--------------|__ConcreteCallBack____|
 *         ^                                  ^
 *         |<0..*>                            |<has one>
 *         |                                  |
 *  ________________                  ____________
 * |   <class>      |  <has one>     |  <class>   |
 * |__SystemTimer___|<---------------|___Client___|
 *
```

CALLBACK INTERFACE

Let's define first the callback interface:

```
import java.util.ArrayList;
import java.util.List;

// For example: Let's assume that this interface is offered from your OS to b
interface TimeUpdaterCallBack {
    void updateTime(long time);
}

// this is your implementation.
// for example: You want to update your website time every hour
class WebSiteTimeUpdaterCallBack implements TimeUpdaterCallBack {

    @Override
    public void updateTime(long time) {
        // print the updated time anywhere in your website's example
        System.out.println(time);
    }
}
```

THE SYSTEMTIMER THAT SUPPORTS CALLBACK FUNCTIONS IN OUR EXAMPLE:

```
// This is the SystemTimer implemented by your Operating System (OS)
// You don't know how this timer was implemented. This example just
// show to you how it could looks like. How you could implement a
// callback by yourself if you want to.
class SystemTimer {

    List<TimeUpdaterCallBack> callbacks = new ArrayList<TimeUpdaterCallBack>(

    public void registerCallBackForUpdatesEveryHour(TimeUpdaterCallBack timer
        callbacks.add(timerCallBack);
    }

    // ... This SystemTimer may have more logic here we don't know ...

    // At some point of the implementaion of this SystemTimer (you don't know
    // this method will be called and every registered timerCallBack
    // will be called. Every registered timerCallBack may have a totally
    // different implementation of the method updateTime() and my be
    // used in different ways by different clients.
    public void oneHourHasBeenExprired() {
```

refactoring (3) test (3) Activity (2) AdMob (2) Advertisement (2) AndroidAnnotations (2) Color (2) DTO (2) Factory (2) View (2) Windows (2) adapter pattern (2) agile (2) annotations (2) applied (2) builder pattern (2) command pattern (2) decorator (2) generic (2) mobile (2) null object (2) null object pattern (2) observer pattern (2) phonegap (2) pitfall (2) quality (2) reusable comments (2) singleton pattern (2) state (2) state pattern (2) synchronized (2) visitor (2) AD_UNIT_ID (1) ARGB (1) AndroidStudio (1) AsyncTask (1) Bean (1) Bridge (1) Callback (1) Canvas (1) Coloring (1) Colour (1) DMO (1) DO (1) Data Access Object (1) Dependency Inversion (1) Domain Object (1) Draw (1) Drawing (1) Fragment (1) Gson (1) Hollywood Principle (1) IDE (1) Install (1) IntelliJ (1) Interpreter (1) Inversion of control (1) IoC (1) Iterator (1) JBoss (1) JBoss7 (1) JBossAS7 (1) Javascript (1) Json (1) MVC (1) Mapper test (1) Mediator (1) Mixing (1) ORM (1) ORMLite (1) Paint (1) ProgressBar (1) RCP (1) RGB (1) Registry (1) Repository (1) Snippet (1) SoC (1) Splash (1) Tools (1) Vista (1) abstract factory (1) actionBar (1) adapter (1) appearance (1) appl (1) attachment (1) autosave (1) baseadapter (1) bitstate (1) chain of responsibility (1) checklist (1) class design (1) clean code developer theory coder (1) code convention (1) code style (1) composite (1) concept (1) contest (1) contraindication (1) customize (1) dash (1) db (1) dialog (1) documentation (1) dotted (1) email (1) enum (1) facade (1) factory pattern (1) flyweight (1) good (1) if (1) image (1) imageloader (1) initialization (1) instanceOf (1) layout (1) lazy (1) line (1) manifest (1) memento (1) monetize (1) multilanguage (1) multilingual (1) observer (1) parse (1) polymorphism (1) project (1) prototype (1) proxy (1) rename (1) resources (1) roboelectric (1) scrollview (1) seal (1) shortcuts (1) simple factory (1) statements (1) strategy (1) strings (1) stroke (1) switch (1) syntax (1) syntaxionary (1) system (1) tdd (1) template method (1) tutorial (1) util (1) valuable (1)

```
        for (TimeUpdaterCallBack timerCallBack : callbacks) {
            timerCallBack.updateTime(System.currentTimeMillis());
        }
    }
}
```

AND FINALLY OUR WEBSITETIMEUPDATER WHICH IS OUR CLIENT IN THIS FICTIVE AND SIMPLE EXAMPLE:

```
// This is our client. It will be used in our WebSite example. It shall update
// the website's time every hour.
class WebSiteTimeUpdater {

    public static void main(String[] args) {
        SystemTimer SystemTimer = new SystemTimer();
        TimeUpdaterCallBack webSiteCallBackUpdater = new WebSiteTimeUpdaterCallBack();
        SystemTimer.registerCallBackForUpdatesEveryHour(webSiteCallBackUpdater);
    }
}
```

```
// This is our client. It will be used in our WebSite example. It shall updat
// the website's time every hour.
class WebSiteTimeUpdater {

    public static void main(String[] args) {
        SystemTimer SystemTimer = new SystemTimer();
        TimeUpdaterCallBack webSiteCallBackUpdater = new WebSiteTimeUpdaterCa
        SystemTimer.registerCallBackForUpdatesEveryHour(webSiteCallBackUpdate
    }
}
```

Advertising:

Optimized bets for playing EuroMillion's lottery on your Mobile Phone!

ANDROID APP ON Google play

Reactions: helpful (1) interesting (0) cool (0)

Posted by Ricardo Ferreira at 3:13 PM

+2 Recommend this on Google

Labels: Callback, Hollywood Principle, Inversion of control, IoC, Java, pattern

14 comments:

Andranik Azizbekyan April 24, 2013 at 5:02 AM

What is the theme you are using in eclipse?
Would you please send the name or, preferably, the .epf file? I like the blue tone :)

Reply

Replies

Ricardo Ferreira April 27, 2013 at 9:44 AM

Hi! i like it too and i'm trying to create something similar to that in eclipse. I'm using sublime text. as soon as i have something like that, i'll be sending to you ok.

regards,
Ricardo

Anonymous July 22, 2013 at 5:48 PM

It is Sublime 2 or 3 its free to use or you can buy a license.

Reply

Anonymous April 25, 2013 at 12:03 AM

That doesn't seems like eclipse - that's probably QT.

Reply

Replies

Anonymous May 16, 2013 at 10:44 PM

Did anybody say vim?

Anonymous May 19, 2013 at 5:57 AM

It seems like IntelliJ IDEA http://www.jetbrains.com/idea/

Reply

Anonymous May 6, 2013 at 2:12 PM

I'm not understanding how your example this is anymore concrete than the original example you pasted that appears everywhere else on the internet.

Reply

Faissal Elamraoui September 27, 2013 at 4:08 AM

Great post :). We could also call the callback like this:

```
caller.register(new Callback() {
@Override
public void methodToCallback() {
// Show a message
}
});
```

Thanks,

Reply

Anonymous January 4, 2014 at 12:48 AM

Great post, thank you!
Then, you could say, for a newbie, that a callback is a kind of listener for OS calls?

Reply

Anonymous May 26, 2014 at 8:43 PM

Thank you it helped me, but I did't get this, the callback is excecute automatically?if I'm right, when does it happen?

Reply

alejandro relañez May 27, 2014 at 6:29 AM

Perfect post, just what I needed, I did't get that, when is execute the callback??

Reply

Replies

Anonymous October 12, 2014 at 2:47 AM

er mane putki mara voda

Anonymous October 29, 2014 at 10:12 AM

er mane putki mara voda ... LOL. translate it man.
can we say that, Faced Design Pattern is an example of callback

Reply

Anonymous January 26, 2015 at 5:48 AM

Hi Richard ,
That's a really nice explanation. Could you please use a bright background for you future posts so that it won't hurt our poor eyes :)

Reply

Enter your comment...

Comment as: Select profile...

Publish Preview

Links to this post

Create a Link

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)