DZone RESEARCH
— Presents —
DZone's Definitive Guide to Cloud Providers
I Got 99 Problems, But the Cloud Ain't One...
Get DZone's Definitive Guide!
Download Now
Learn More
DZone RESEARCH

STAFF James Sugrue
Bio Website @sugrue

Design Patterns Uncovered: The Memento Pattern

04.21.2010 21204 views

Like 2 Tweet 1 Share +1 1

Today's pattern is the Memento pattern which is used in undo frameworks to bring an object back to a previous state.

CONNECT WITH DZONE

Publish an Article Share a Tip

Like 7.6k Follow 22.9K follow ers

RECOMMENDED RESOURCES

- Collaborative Filtering at Scale
- Cassandra in the Netflix Architecture

to a previous state.

Memento in the Real World

In the real world, memento's are used a reminder or reference of how something should look. For example, if you decide to take a phone apart in order to replace an internal part, you may have an identical phone available that you use as a reference, ensuring you can take get the phone back to it's original state.

DesignPatterns Refcard
For a great overview of the most popular design patterns, DZone's Design Patterns Refcard is the best place to start.

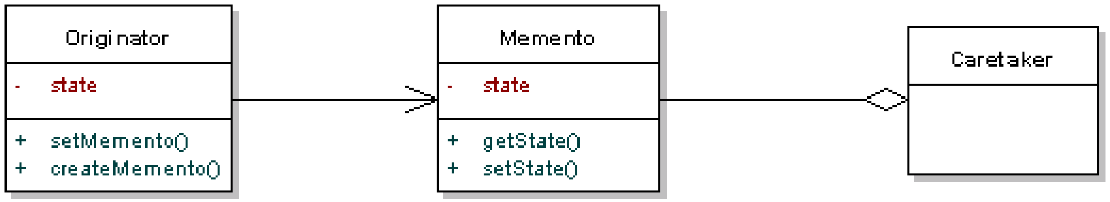
The Memento Pattern

The Memento pattern is known as a**behavioural** pattern, asit's used to manage algorithms, relationships and responsibilities between objects.. Thedefinition of Memento as provided in the original Gang of Four book on DesignPatterns states:

> *Captures and externalizes an object's internal state so that it can be restored later, all without violating encapsulation.*

The following diagram shows how the memento pattern is modelled.

Connecting Applications Everywhere with Red Hat JBoss A-MQ
ActuateOne for OEMs
Configuring and Running FUSE ESB



Let's take a look at each of the participants in this pattern. The **Originator** is the object that knows how to save itself: the class that you want to make stateful. The **Caretaker** is that object that deals with when, and why, the Originator needs to save or restore itself. The **Memento** holds the information about the Originator's state, and cannot be modified by the Caretaker.

The flow of events is that the Caretaker asks the Originator for the Memento object and performs any actions that it needs to. To rollback the state before these actions, it returns the memento object to the originator.

When Would I Use This Pattern?

The Memento pattern is useful when you need to provide an undo mechanism in your applications, when the internal state of an object may need to be restored at a later stage. Using serialization along with this pattern, it's easy to preserve the object state and bring it back later on.

SoHow Does It Work In Java?

Let's use a simple example in Java toillustrate this pattern.As it's a pattern used for undo frameworks, we'll model a editor.

First, the memento needs to be able to save editor contents, which will just be plain text:

Featured Refcardz:
1. HTML5 Mobile
2. Spring Batch
3. Vaadin
4. Drools
5. C++

Top Refcardz:
1. PostGIS
2. PostgreSQL
3. Machine Learning
4. C++
5. Java

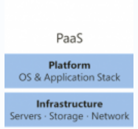
150+ Refcardz Available · Get them all

Spotlight Features

Load Balancing Apache Tomcat with Nginx

Building the Perfect Cassandra Test Environment

A Serious Flaw of PaaS



How I MapReduced a Neo4j Store w/ Hadoop

POPULAR AT DZONE

Exciting New Features in Red Hat JBoss Portal Platform 6.1 Beta 1

Android Custom ListView Items "Row" 2

Regex, Parsing and some benchmarks in Java 3

JavaOne 2013 BOF2605 JavaFX, Widgets and Apps, Oh My! Launching 4

```
//Memento
public class EditorMemento
{

  private final String editorState;

  public EditorMemento(String state)
  {
     editorState = state;
  }

  public String getSavedState()
  {
   return editorState;
  }

}
```

Widgets, and Apps, Oh My! Launching

Custom bean validator with Spring framework and API/implementation 5

Exploring Apache Camel Core - DataSet Component 6

Exploring Apache Camel Core - File Component 7

See more popular at DZone

Subscribe to the RSS feed

Now our Originator class, the editor, can use the memento:

```
//Originator

public class Editor
{

  //state
  public String editorContents;

  public void setState(String contents)
  {
     this.editorContents = contents;
  }

  public EditorMemento save()
  {
    return new EditorMemento(editorContents);
  }

  public void restoreToState(EditorMemento memento)
  {
   editorContents = memento.getSavedState();
  }

}
```

Anytime we want to save the state, we call the save() method, and an undo would call the restoreToState method.

Our caretaker can then keep track of all the memento's in a stack for the undo framework to work.

WatchOut for the Downsides

Some problems with this pattern is that the saving or restoring of state can be a time consuming process. Used incorrectly, it can expose the internal structure of your object, thus allowing any other object to change the state of your object.

OtherArticles in This Series

The Observer Pattern

TheAdapter Pattern

TheFacade Pattern

The Factory Method Pattern

TheAbstract Factory Pattern

The Singleton Pattern

The Strategy Pattern

TheVisitor Pattern

The Decorator Pattern

The Proxy Pattern

TheCommand Pattern

TheChain of Responsibility Pattern

Next Up

As we start to near the end of the series, we'll look at the Mediator pattern next.

Tags: Java Patterns

Comments

Chris Sherlock replied on Sun, 2011/10/23 - 3:59am

Good article, only I tend to agree with the Wikipedia article that a nested class with a private constructor might be better for the Memento pattern.

Login or register to post comments

Murtaza Khurshe... replied on Thu, 2012/12/20 - 4:00am

```
public void setState(String contents)

{
this.editorContents = contents;

}
```

where are you calling this method ?

Login or register to post comments

Comment viewing options

Flat list - expanded | Date - oldest first | 30 comments per page | Save settings

Select your preferred way to display the comments and click "Save settings" to activate your changes.

AROUND THE DZONE NETWORK

ARCHITECTS

Big Data Beyond MapReduce: Google's Big Data Paper...

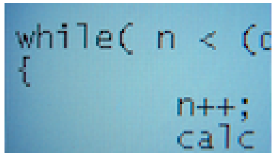
JAVALOBBY

The Principles of Java Application Performance Tun...

ARCHITECTS

5 Things a Java Developer Should Consider This Yea...

JAVALOBBY



There Are Only 2 Roles of Code

JAVALOBBY

Singleton Design Pattern – An Introspection w/ B...

SERVER

Best Best Practices Ever

YOU MIGHT ALSO LIKE

- DZone Weekly Link Roundup (September 3)
- Dev of the Week: Scott Westfall
- Stop Writing Code You Can’t Yet Test
- JPA vs. Hibernate?
- When Will the Golden Age of “The Developer” End?
- This is how Facebook develops and deploys software. Should you care?
- From Hibernate to MongoDB, Part 1
- Learning MongoDB: The CRUD
- The Joys of Redis
- From Hibernate to MongoDB - Part 2
- Working with jQuery Mobile Panels
- Do I need to understand low-level programming and memory management?
- An Introduction to Node's New Streams
- Making Your Own Heroku (On a 7 Year Old OS)
- Load Balancing Apache Tomcat with Nginx

POPULAR ON JAVALOBBY

- Spring Batch - Hello World
- Is Hibernate the best choice?
- How to Create Visual Applications in Java?
- 9 Programming Languages To Watch In 2011
- Lucene's FuzzyQuery is 100 times faster in 4.0
- Introduction to Oracle's ADF Faces Rich Client Framework
- Interview: John De Goes Introduces a Newly Free Source Code Editor
- Time Slider: OpenSolaris 2008.11 Killer Feature

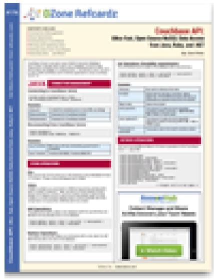
LATEST ARTICLES

- Percentiles Made Easy
- Camel 2.12 Liftoff!
- Rapid Mobile App Development Using the Appery.io Mobile Pack and Salesforce.com [Webinar]
- Crazy Translations of Simple SQL Expressions to Various SQL Dialects
- The RavenDB High Performance Book is Out
- It's Not About the Process
- Building a Spaceship with TDD: The Prequel
- A Mindful Team

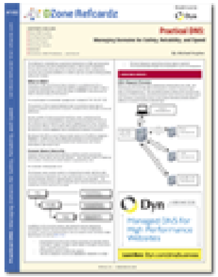
SPOTLIGHT RESOURCES

Camel Essential Components

DZone's 170th Refcard is an essential reference to Camel, an open-source, lightweight, integration library. This Refcard is authored by...



Essential Couchbase APIs: Open Source NoSQL Data Access from Java, Ruby, and .NET



Practical DNS: Managing Domains for Safety, Reliability, and Speed

DZone

Search

"Starting from scratch" is seductive but disease ridden
-Pithy Advice for Programmers

Online Visitors: 433