View Source: Design Patterns in the Wild

Patterns in Log4J

© Michael Brenton

Ashwin J Mathew Spring 2008
School of Information, UC Berkeley
Patterns for Today

- Factory
- Template Method
- Observer
- Mediator
- Facade
Factory

Factory

+makeWidget(): Widget

Widget
new() vs. Factory
Using Factory

- **Functional**
  - Construct objects from different classes depending on parameters
  - Return object instances from a cache

- **Organizational**
  - Break out complex constructor logic to a separate factory class
Template Method

![Diagram of Template Method]

- **Widget**
  - `+call()`
  - `#callTemplate()`

- **JumpyWidget**
  - `#callTemplate()`

- **DeafWidget**
  - `#callTemplate()`

Ashwin J Mathew  Spring 2008
School of Information, UC Berkeley
Appender Template Method

Appender

AppenderSkeleton
+doAppend(event:LoggingEvent)
#append(event:LoggingEvent)

JDBCAppender
+append(event:LoggingEvent)

WriterAppender
+append(event:LoggingEvent)

SMTPAppender
+append(event:LoggingEvent)

ConsoleAppender

FileAppender
Using Template Method

- Wrap common logic around specialized logic
- Concrete subclasses provide specialization for template method
- Conceptually similar to Aspect Oriented Programming (AOP)
Observer

Observed

+observers: Collection
+callObservers()
+ addObserver(observer: Observer)

Observer

+onEvent()
Appenders as Observers

Category

- addAppender(appender: Appender)
- callAppenders(event: LoggingEvent)
- forcedLog(fqcn: String, level: Priority, message: Object, t: Throwable)

AppenderAttachableImpl

- addAppender(appender: Appender)
- appendLoopOnAppenders(event: LoggingEvent)
- getAllAppenders(): Enumeration
- getAppender(name: String): Appender
- isAttached(appender: Appender): boolean
- removeAllAppenders()
- removeAppender(appender: Appender)
- removeAppender(name: String)

Appender

+ doAppend(event: LoggingEvent)
Using Observer

- When arbitrary numbers of objects need to be invoked on events in calling object
- Decouple observers from calling object
Mediator

JumpyWidget
+call()

Mediator
+call()

DeafWidget
+call()
Mediating Loggers and Appenders

PropertyConfigurator

```java
#registry: HasTable
+doConfigure(properties:Properties,hierarchy:LoggerRepository)
configureRootCategory(properties:Properties,
hierarchy:LoggerRepository)
parseCategory(properties:Properties,logger:Logger,
optionKey: String,loggerName: String,
valueName: String)
pauseAppender(props:Properties,appendeName: String)
registryGet(name: String): Appender
registryPut(appender: Appender)
```
Using Mediator

- Keep related classes from referring to one another directly
- Extract logic relating different classes to Mediator
- Warning: potential God class!
Logger as Facade
Using Facade

- Simple interface to complex functionality
- Encapsulates a subsystem of classes
- Single class, or small group of classes