View Source: Design Patterns in the Wild

Patterns in Apache Tomcat

© Ross Orr

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

- Command
- Strategy
- Intercepting Filter
- Visitor
Rules as Commands

![Diagram of Digester and Rule classes with associations]

- **Digester**
  - #rules: Rules
  - +addRule(pattern: String, rule: Rule)
  - +startElement(namespaceURI: String, localName: String, qName: String, attributes: List)
  - +endElement(namespaceURI: String, localName: String, qName: String)

- **Rule**
  - +begin(namespace: String, name: String, attributes: Attributes)
  - +body(namespace: String, name: String, text: String)
  - +end(namespace: String, name: String)
  - +finish()

- **Rules**
  - +match(namespaceURI: String, pattern: String): List
  - +addRule(pattern: String, rule: Rule)

- **RulesBase**
  - +match(namespaceURI: String, pattern: String): List
  - +add(pattern: String, rule: Rule)
Using Command

- Decouple invoker of an operation from command object performing operation
- Invoker doesn't need to know what command does
- Command may carry internal state
Strategy

```
Strategy
+doSomething()

StrategyFoo
+doSomething()

StrategyBar
+doSomething()
```
ProtocolHandler as Strategy

Diagram:

- **ConnectorCreateRule**
  - `+begin(attributes: Attributes)`

- **Connector**
  - `#protocolHandler: ProtocolHandler`
  - `+Connector(protocol: String)`

- **ProtocolHandler**

- **Http11Protocol**

- **Http11NioProtocol**

- **AjpProtocol**
Using Strategy

- Client operates on interface or abstract class
- Implementation varies independent of client
- Strategies are “pluggable”
Intercepting Filter

**UML Class Diagram**

- **FilterManager**
  - `+filters: List`
  - `+doFilter()`

- **Filter**
  - `+doFilter()`

**Sequence Diagram**

- **FilterManager**
- **FilterFoo**
- **FilterBar**

Messages:
- `doFilter`

### Author Information

Ashwin J Mathew  
Spring 2008  
School of Information, UC Berkeley
ApplicationFilterChain as Intercepting Filter

FilterChain
+doFilter(request: ServletRequest, response: ServletResponse)

ApplicationFilterChain
+doFilter(request: ServletRequest, response: ServletResponse)
-internalDoFilter(request: ServletRequest, response: ServletResponse)

Filter
+doFilter(request: ServletRequest, response: ServletResponse, chain: FilterChain)

ApplicationFilterConfig
+getFilter(): Filter

ApplicationFilterChainFactory
+createFilterChain(request: servletRequest, wrapper: Wrapper, servlet: Servlet)
Using Intercepting Filter

- Filter chain “intercepts” all requests
- Filters are loosely coupled
- Filter Manager oversees invocations of Filters
- Filters may choose to abort processing for a request
- Spaghetti Code Warning: don't let your Filters reference one another!
Visitor

- Element
  - +accept(visitor:Visitor)

- ElementFoo
  - +accept(visitor:Visitor)

- ElementBar
  - +accept(visitor:Visitor)

- Visitor
  - +visit(element:ElementFoo)
  - +visit(element:ElementBar)

- VisitorOne
- VisitorTwo
  - +visit(element:ElementFoo)
  - +visit(element:ElementBar)
Double Dispatch

Client

Visitor

Element

new

accept(visitor)

visit(element)
Visiting Nodes

```
ELFunctionMapper
    +acceptVisitor(visitor:Visitor)
    #doVisit(node:Node)
    #visitBody(node:Node)
    +visit(node:CustomTag)
    +visit(node:IncludeAction)
    +visit(node:ELEExpression)

Visitor
    #doVisit(node:Node)
    #visitBody(node:Node)
    +visit(node:CustomTag)
    +visit(node:IncludeAction)
    +visit(node:ELEExpression)

Node
    +acceptVisitor(visitor:Visitor)

CustomTag
    +acceptVisitor(visitor:Visitor)

ELEExpression
    +acceptVisitor(visitor:Visitor)

IncludeAction
    +acceptVisitor(visitor:Visitor)
```
Using Visitor

- Decouple operations to perform on a collection from the collection itself
- Define new operations without affecting the structure of the collection
- Polymorphic double dispatch based on type of object being visited
- The right code is executed based on types of both objects involved in double dispatch