3118
J2EE Patterns

Ken Sipe
Code Mentor, Inc.
Overall Presentation Goal

Review of Core J2EE Patterns.
Best Practices of J2EE Patterns.
Speaker’s Qualifications

• Chief Technology Officer at Code Mentor, Inc.
• Sun Certified Java 2 Architect.
• Instructor for Visibroker for Java, OOAD, Rational Rose, and Java Development.
• Frequently speaks on the subject of distributed computing programming, including CORBA and EJB architecture.
Introduction

• In the Beginning…
  • There was code
  • How do we get beyond copy and paste
Thought for the Session

A good scientist is a person with original ideas.

A good engineer is a person who makes a design that works with as few original ideas as possible.

—Freeman Dyson
Reuse Solutions

- OO
  - Models
- Component Oriented Development
  - Binaries
Why / Where do they come from

OOAD

- Notation – UML
- Process (Raging)
- Software Development
  - Technical Requirements
    - Patterns
- Business Requirements
  - Need further standardization
Patterns Definition

- Patterns communicate a solution to a recurring problem in context.
  - John Crupi
Four Pattern Elements

- Name
- Problem (with Context)
- Solution
- Consequences (Pro / Cons)
Pattern Value

- Design “Reuse”
- Best Practices
- Easy code mappings
- Higher level vocabulary
Early Patterns Solutions

- Gang of Four
  - Singleton
  - MVC
  - Command
- Context is technical and broad

*Tend to be technical in nature and span language choices.*

*Limited understanding*
J2EE Core Patterns

- Sun developed best practices
  - John Crupi, Dan Malks, Deepak Alur
J2EE Patterns

Patterns in the “context” of J2EE
Presentation Tier

- Intercepting Filter
- Front Controller
- **Context Object**
- **Application Controller**
- View Helper
- Composite View
- Service To Worker
- Dispatcher View
Business Tier

- Business Delegate
- Service Locator
- Session Façade
- Application Service
- Business Object
- Composite Entity
- Transfer Object
- Transfer Object Assembler
- Value List Handler
Integration Tier

- Data Access Object (DAO)
- Service Activator
- Domain Store
- Web Service Broker
Non-Core J2EE Patterns 😊

- Fast Lane Reader
- MVC
- Mapper Pattern
- Pluggin Pattern
- Service Stub
- Token Pattern *
Patterns for this Session

- Front Controller
- MVC
- Business Delegate
- Mapper Pattern
- Service Stub
- Session Façade
- Fast Lane Reader
- Token Pattern
- Value List Handler

Focus on development and JBuilder
What Is a MVC?

• MVC stands for model / view / controller.

• A software pattern where logic is separated from the model and view in order to provide for better reuse possibilities.

• A software pattern recognized in the early days of small talk.

• Documented in the GoF book.
Web Application MVC Pattern

- **Model**
  - Information is provided in objects or beans
- **View**
  - The JSP provide the view
- **Controller**
  - Servlet provides control logic and becomes the controller
MVC Collaboration Diagram

Controller

1: Post

Browser

Servlet

2: Retrieve Data

3: Establish bean state, then place in session or request object

4: redirect to appropriate view

Data Resource

5: Access beans

Beans

The View

The Model

JSP
Front Controller / MVC Solutions

- J2EE Context
- Struts
  - Concept of a Front Controller
    - ActionServlet
    - RequestProcessor
Demo

Jbuilder / Struts
Business Delegate

• Presentation Tiers
• Reduction of coupling of the client tiers to the business tiers
• Centralizes code for lookups, access details
Business Delegate

• Great place for:
  • Data transformation
    • Tier to tier coupling
    • Exception translation
  • Manage unit tests
  • Service Stubs
Mapper Pattern

• An object that sets up a communication between two independent objects - Fowler

• Provides a configurable access to service stubs
Pluggin Pattern

- Links classes during configuration rather than compilation - Fowler
Service Stub

• *Removes dependence upon problematic services during testing*
  – Fowler
Mapper / Pluggin Solution

Client

Mapper Interface

EJB Delegate Mapper

Stub Mapper

EJB Tier
Session Façade

• Removes the number of client / server method invocations
• Uniform client access strategy
• Remove direct Entity Bean access
• Provides course grain access to Business
Data Transfer Object (DTO)

- Data Structure for serialization across tiers.
Demo

Jbuilder / Facades / Delegates / DTO
Fast Lane Reader

- Fast access to read-only data
- By pass Entity layer.
Token Pattern

• Synchronize the Client Presentation Tier
  • Effort to stop multiple requests
Value List Handler

• Provides a scalable solution to providing lists of data to the client.
Value List Diagram

interface ValueListIterator

+getSize(): int
+getCurrentElement(): Object
+getPreviousElements(count: int): List
+getNextElements(count: int): List
+resetIndex(): void

Client

ValueListHandler

<<List>>

ValueList

ValueObject

DataAccessObject

iterates

provides data

accesses
Concluding Thoughts

• Patterns presented are in heavy use today and add value

• JBuilder continues to aid in providing wizards to accelerate pattern focused development.
Resources

• http://java.sun.com/blueprints/corej2eepatterns/
• http://www.martinfowler.com/eaaCatalog/
Questions?
Thank You

Please fill out the speaker evaluation

You can contact me further at ...
kensipe@codementor.net
2003 Borland Conference