

# Operational Frameworks for e-Business (Part 2)

---

**Gary Murphy**

**Hilbert Computing, Inc.**

**glm@hilbertinc.com**

**913-780-5051**

# Using Hilbert Frameworks

---

- This code is copyrighted by Hilbert Computing, Inc.
- Open-source, royalty-free license
- Not GNU “copy-left”. You don’t have to give back any code derived from this code.
- Don’t directly modify anything in the Hilbert packages.

# Scope of this Session

---

- How to Use
  - What is available
  - Programming examples
  - Goals behind the code
- Internals
  - This info is not required to use the framework
  - Focus on this at Colorado Software Summit

# Servlet Framework

---

- Operational support below the application programming level:
  - Logging using the logging framework
  - Response time metrics
  - Service level agreement (SLA) capabilities
- These can be extended with no change in your application programs

# Servlets - Design

---

- Use a classic application programming model. Don't use servlet chaining, etc.
- Use modern design principles (e.g. separation of presentation and application logic, loosely coupled objects, etc.)
- Create an opportunity for infrastructure code

# Servlets - Implementation

---

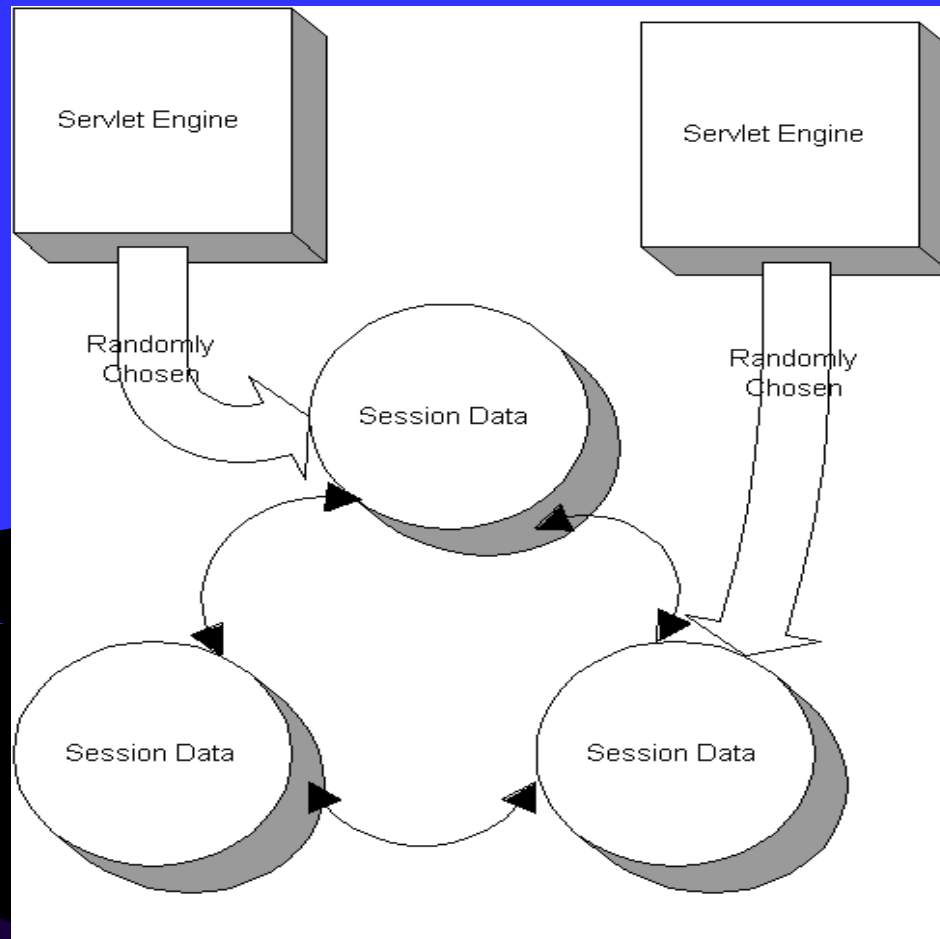
- HInstance handles “lifecycle” complexity and helps with thread-safety. Objects have different lifecycles:
  - Servlet
  - Session
  - Request/response

# Servlets - Implementation

---

- Failover and Load balancing
  - Network equipment (e.g. Cisco Local Directors) round-robin workload among multiple web servers
  - Session data needs to be shared among multiple servlet engines running in a JVM.
  - Shared session data is implemented as distributed objects with replication and redundancy via a properties entry

# Servlets - Implementation



# Servlets - Implementation

---

- HInstance primarily handles GET/POST variables
- Facilitates event-driven programming
  - Similar to Windows, OS/2 or Motif programming
  - The `handleRequest()` is a switch statement that processes requests

# Servlet - Source Code

---

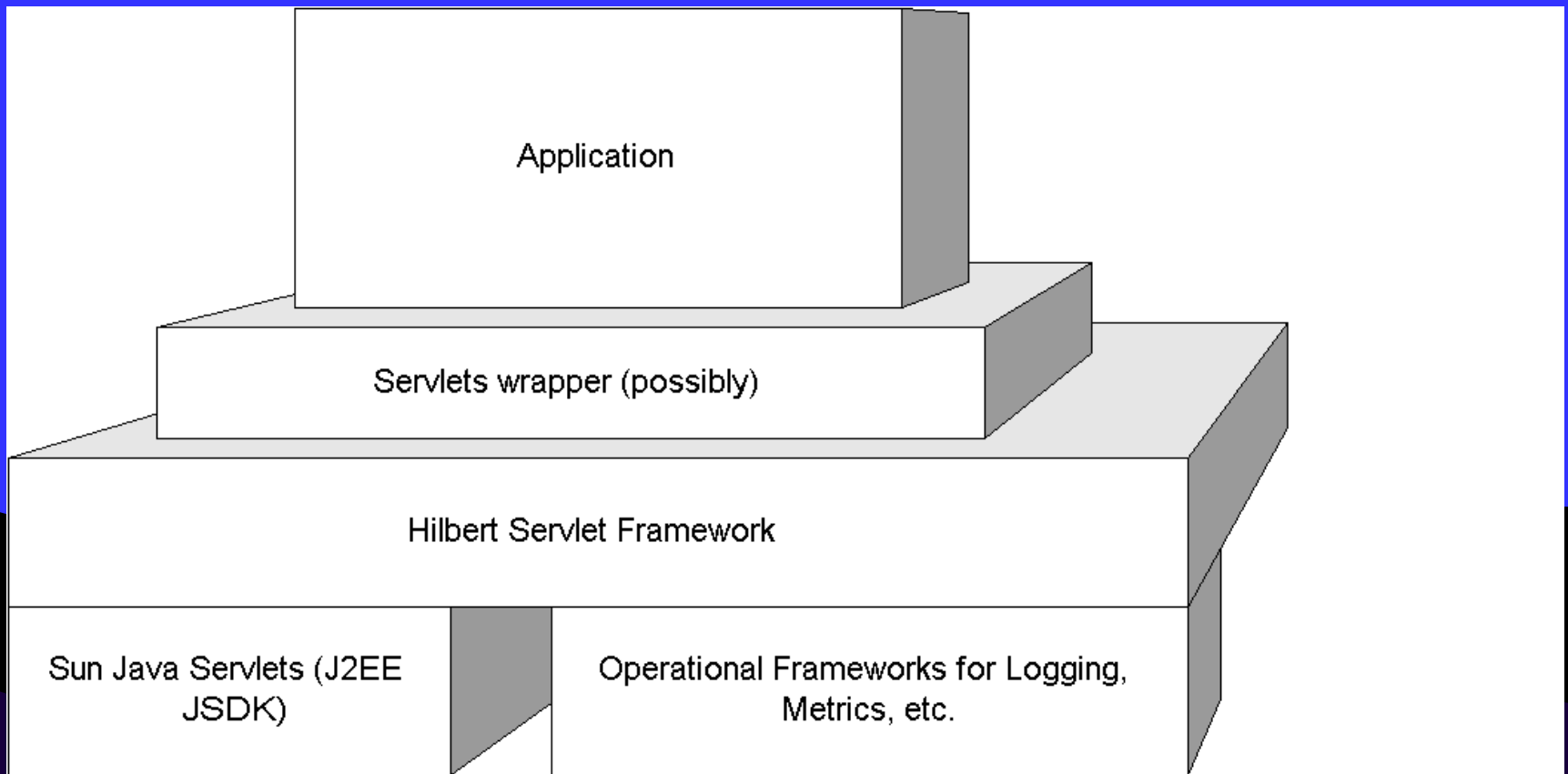
- Let's look at the source code for a sample servlet.
  - `handleRequest(...)` is the main dispatching method. Transitions the event-driven HTTP model to your object model
  - `getRequest()` maps the POST/GET variable to an integer for the switch statement

# Servlets - Implementation

---

- HttpServlet abstract class
  - Must subclass to provide a name for the logging framework
  - `handleRequest(HInstance)` will be overridden to respond to HTTP requests
  - `handleSessionTracking(...)` automatically establishes local/remote session data.
  - Let's look at the code...

# Servlet Layers



# Servlets - Implementation

---

- The intent, although not enforced in code, is that all presentation be done through static HTML or JSPs. Don't code HTML in the servlet. *HTML is the next legacy.*
- The JSP Rendering Framework is an appropriate place for presentation beans.

# JSP Rendering Framework

---

- Separate the presentation of information from the business logic
- Minimize the interaction between graphic arts and developers
- HTML is legacy. Business logic should be able to be used in XML and WAP/WXML applications with no change.

# JSP - Design Points

---

- Everything is rendered by the same class
- Rendering objects such as strings is trivial, but we also want to render JDBC result sets with the same ease.
- This must be easy to prevent programmers from coding HTML directly in their servlets

# JSP - Examples

---

- Let's look at some example JSPs...

# JSP - Implementation

---

- HOutputRender - an interface for any JSP “widget”
  - render(“command”);
  - display();
- The JSP coder (e.g. graphic artist) always codes the same ‘type=’ in the `<useBean>` and codes `<%=bean.display()%>` for the widget to be displayed.

# JSP - Implementation

---

- Canned Classes
  - HHtmlDateRenderer - Date as combo boxes
  - HHtmlStringRenderer - Handles meta characters (<, > and &)
  - HResultSetListRenderer - Display a JDBC result set as an HTML list
  - HResultSetTableRenderer - Display a JDBC result set as table with nice columns, etc.

# JSP - Implementation

---

- Near-Ready Classes
  - HDefaultListRenderer - Takes a customized handler. (Subclass HAbstractListHandler) The handler starts and ends the list, and provides the next element in the list to render.
  - Some handlers are already provided for template-based, list items, and choice boxes

# JSP - Implementation

---

- Use of the canned classes is easy. Creating customized presentation has complexity similar to Swing.
  - Frameworks will provide “cookbooks” to show how to reuse certain parts and override other parts.
  - Grow a list of “widgets” over time so programmers aren’t exposed to this level of complexity

# JSP - Forms Management

---

- Tedious Programming
  - Filling in default values
  - User input validation.
    - Re-sending the form with invalid fields flagged
    - Saving the valid fields keyed in
  - Marshalling data between the form and the POST/GET values

# JSP - Forms Management

---

- HHtmlFormRenderer class handles these issues and...
- Can be configured via a XML document so that the form can be changed with minimal or no impact to the application

# JSP - Forms Management

---

- Drawbacks
  - Layout is not as flexible as HTML forms
  - Can't use HTML IDEs such as Visual Studio for forms layout

# Database Framework

---

- This wrappers JDBC calls to implement a transaction of multiple SQL statements
- Provides database connection pooling
- Provides an SQL “factory” for plugging values into an SQL statement more easily.

# Database - Implementation

---

- HDatabaseConnection - This handles the JDBC driver load with better error messages.
- HConnectionPool - Implements synchronous or asynchronous execution of SQL.

# Database - Implementation

---

- HSqlDescriptor - Describes a single SQL statement.
  - Provides a contained object for transactions
  - Provides sophisticated caching and thread-safety for prepared and callable statements
- HSqlTransaction
  - Manages the running of descriptors and automatic commit or rollback on error

# Database - Implementation

---

- Allows callback of code within a transaction using a callback descriptor
- Allows a unit of work to be suspended in the middle of a transaction for additional user interaction. This is useful in “FOR UPDATE” processing.

# Database - Prepared Stmts

---

- Prepared and Callable (stored procs) Statements have a conflict with the multithreaded servlet model
  - Statements are cached in the thread pool
  - Access to the statement is thread-safe
  - Automatic statement creation after timeout
  - Programmer must “.release()” the statement when done with the statement/result set

# Summary

---

- Frameworks provides several key benefits:
  - Quicker time-to-market through reuse
  - Ability for those new to OO to reuse code
  - Ability for those skilled in OO to significantly extend the frameworks
- Hilbert Frameworks specifically:
  - Focus on operational aspects to improve availability