Best Practices: Data Access Strategies

Richard Bair

SwingLabs Lead
Sun Microsystems
http://swinglabs.org

TS-4635

Main Takeaway

Swing: Powerful Rich Client Platform

Learn different data access strategies and application design patterns for building occasionally connected rich client applications

Agenda

Smart Clients Defined

Application Design Overview

Demo Application

Summary

Agenda

Smart Clients Defined

Application Design Overview

Demo Application

Summary

Smart Clients

- Access to local machine resources
  - Massive hard drive capacity
  - Graphics card
  - Memory
- Rich user experience
- Connect to network resources
- Occasionally connected

Occasionally Connected

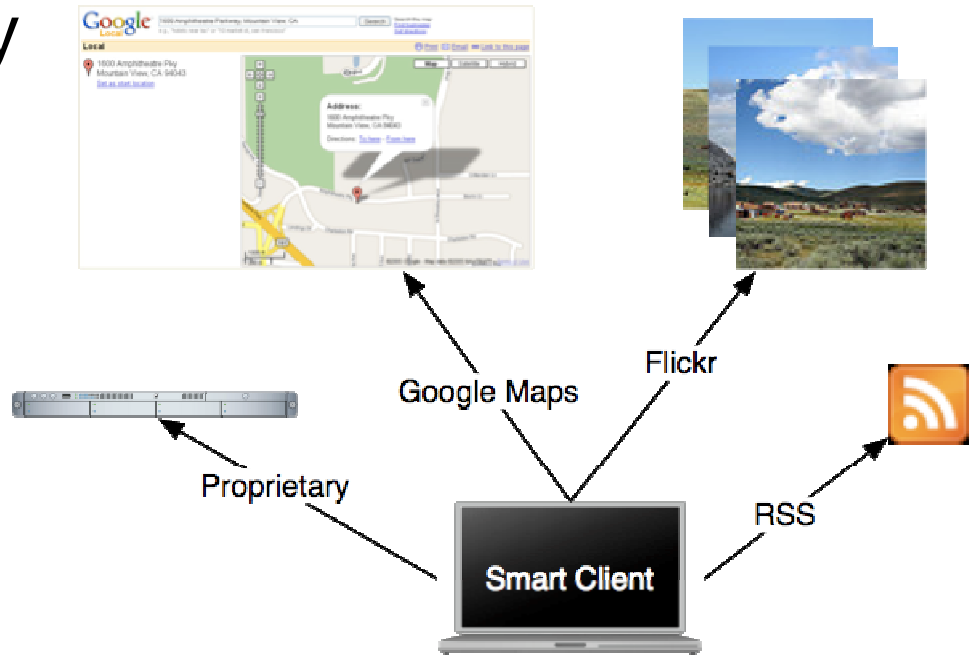
- Users demand reliable access to their data in an unreliable world
- Smart clients must be prepared to cache
  - If disconnected, users can continue working
  - Cache must be synchronized with server
  - Comprehensive caching technology is still a work in progress

ResponseCache

- ResponseCache provides a pluggable mechanism for caching the results of URL connections
  - @since 1.5
  - Works well for storing read-only data

Web Services + Smart Clients

- Smart Clients use Web Services extensively
  - Google Maps
  - Flickr
  - RSS
  - Proprietary
- Mashups work well in Smart Clients
- Best of both worlds



Why Web Services?

- Security
  - Doesn’t require any new open ports
  - Security is built in at the web server level
- Scalability
  - Load balancers, clusters, etc.
- Accessible
  - Easily accessible from any type of client on any type of platform

Web Service Types

- REST
- XML-RPC
- SOAP
- RSS
- Screen Scraping

Note: Learn With Flickr

- Flickr is excellent for learning web services
  - Supports REST, XML-RPC, SOAP, RSS
  - Good documentation
  - Tons of freely available data
  - Read access doesn't require registration
  - Resulting applications are interesting and fun
- Today's Demos use Flickr

REST

- Acronym for Representational State Transfer
- Extremely simple web service
  - It's the way the web works!
- A single URL leads to a unique resource
  - Example: http://www.flickr.com/photos/romainguy/112798971/

REST—Yes, that's it!

- REST is not a framework, its an architectural style
- Can be combined with other webservices such as XML-RPC
- Often the data returned is
  - XML
  - PDF
  - Image (PNG, JPG, SVG, etc.)

XML-RPC

- Somewhat more complicated that REST for binary data (such as images)
- Provides a structured XML document for making remote procedure calls
  - Supports only basic primitives
    - Usually enough to get the job done
  - Very simple
  - Supported in most programming languages

SOAP

- “Simple Object Access Protocol”
- Technically much more that RPC, but usually SOAP is used for RPC
  - A (much) more advanced form of XML-RPC
  - Sometimes too heavyweight for the job
  - Industry Support
    - Microsoft
    - IBM
    - Sun

Flickr REST Request

To request the flickr.test.echo service, invoke like this:

```
http://www.flickr.com/services/rest/?method=flickr.test.echo&name=value
```

Flickr XML-RPC Request

To request the flickr.test.echo service, send a request like this

```
<methodCall>
   <methodName>flickr.test.echo</methodName>
   <params>
      <param>
         <value>
             <struct>
                <member>
                   <name>name</name>
                   <value><string>value</string></value>
                </member>
                <member>
                   <name>name2</name>
                   <value><string>value2</string></value>
                </member>
             </struct>
          </value>
       </param>
   </params>
</methodCall>
```

Flickr SOAP Request

To request the flickr.test.echo service, send an envelope like this

```
<s:Envelope
    xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema"
>
    <s:Body>
        <x:FlickrRequest xmlns:x="urn:flickr">
            <method>flickr.test.echo</method>
            <name>value</name>
        </x:FlickrRequest>
   </s:Body>
</s:Envelope>
```

Flickr REST Response

A method call returns this

```
<?xml version="1.0" encoding="utf-8" ?>
<rsp stat="ok">
    [xml-payload-here]
</rsp>
```

Flickr XML-RPC Response

A simple call to the echo service returns this

```
<?xml version="1.0" encoding="utf-8" ?>
<methodResponse>
    <params>
        <param>
            <value>
                <string>
                    [escaped-xml-payload]
                </string>
            </value>
        </param>
    </params>
</methodResponse>
```

Flickr SOAP Response

A Simple Call to the Echo Service Returns This

```
<?xml version="1.0" encoding="utf-8" ?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-
envelope">
   <s:Body>
      <FlickrResponse xmlns="http://flickr.com/ns/api#">
         [xml-payload]
      </FlickrResponse>
   </s:Body>
</s:Envelope>
```

Flickr REST Code

```
public static List<Photo> search(String tag) {
    URL url = new URL(
        "http://www.flickr.com/services/rest/?” +
        “method=flickr.photos.search&api_key=” +
        myFlickrKey + “&tags=java");
    InputStream is = url.openStream();
    return FlickrParser.parsePhotos(is);
}
```

RSS

- Polling technology that consumes an XML document
- RSS readers subscribe to RSS “feeds” Whenever the feed is updated, it is consumed by the reader
- Very useful, simple, works well in combination with REST or some other web service
- http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html

Screen Scraping

- Poor webservice
- Essentially imitating a browser
  - Parse the HTML from a URL and “scrape” out the relevant info
  - Use HTTP PUT/GET/POST requests to request pages or post forms
- Breaks whenever the page you are scraping changes incompatibly

Web Service Types Summary

- REST: easiest and most straightforward solution
- SOAP: somewhat heavyweight for typical desktop requirements
- XML-RPC: nice balance between easy and powerful

Database Roundup

- ResultSet
- RowSet
- DataSet
- EJB™ 3 Architecture/Hibernate

Database Roundup

- ResultSet
- RowSet
- DataSet
- EJB™ 3 Architecture/Hibernate

ResultSet

- Traditional database access technology
- Easy to write
- Harder to read
- Hard to interact with from a GUI
  - Has a cursor
  - May be unidirectional
  - Throws checked exceptions on most methods

ResultSet Code

```
public static List<Photo> search(String tag) {
    Connection c = createConnection();
    ResultSet rs = c.createStatement().
        executeQuery("select url from PHOTO");
    List<Photo> results =
            new ArrayList<Photo>();
    while (rs.next()) {
        results.add(new Photo(rs.getString(1)));
    }
    rs.close();
    c.close();
    return results;
}
```

Database Roundup

- ResultSet
- RowSet
- DataSet
- EJB 3 Architecture/Hibernate

RowSet

- Extends ResultSet
- Provides for disconnected data (CachedRowSet)
- Provides API for serializing results to XML (WebRowSet)
- Normally has a bidirectional cursor
- Still hard to use in the GUI
  - Moving the cursor around makes it perform poorly in JTable
  - Still throws a lot of checked exceptions

RowSet Code

```
public static List<Photo> search(String tag) {
    Connection c = createConnection();
    CachedRowSet rs = new CachedRowSetImpl();
    rs.setCommand("select url from PHOTO");
    rs.execute(c);
    List<Photo> results = new ArrayList<Photo>();
    while (rs.next()) {
        results.add(new Photo(rs.getString(1)));
    }
    rs.close();
    c.close();
    return results;
}
```

Database Roundup

- ResultSet
- RowSet
- DataSet
- EJB™ 3 Architecture/Hibernate

DataSet

- New in Mustang (Java™ SE 6 platform, pending approval from the Mustang EG)
- Part of JDBC™ 4.0 software
- Extends java.util.List
- Provides for both connected and disconnected states
- Part of the Ease Of Development enhancements in JDBC 4.0 software

DataSet Code

```
public static List<Photo> search(String tag) {
    Connection c = createConnection();
    Queries q =
          c.createQueryObject(Queries.class);
    c.close();
    return q.getAllPhotos();
}

interface Queries extends BaseQuery {
    @Select (“select * from PHOTO”)
    DataSet<Photo> getAllPhotos();
}
```

Database Roundup

- ResultSet
- RowSet
- DataSet
- EJB™ 3 Architecture/Hibernate

EJB 3

- Latest Enterprise JavaBeans™ (EJB) Spec, part of Java EE 5 platform
- “This is not your father’s J2EE™ platform!”

  —Bill Shannon
- Can be used in Java SE platform
  - Requires bundling some jars
  - Requires an implementation
    - Hibernate is the popular choice
  - More powerful, more complex than JDBC 4.0 software DataSet

Hibernate

- Enormously popular ORM library
- Major participant in JSR-220 (EJB 3.0 architecture)
- http://www.hibernate.org/152.html
- Suffers from dependency madness
  - 36 dependency jars shipped with Hibernate 3!
- EJB™ 3.0 technology docs don't reflect current reality

EJB 3 Architecture Code

```
public static List<Photo> search(String tag) {
    EntityManagerFactory f = Persistence.
       createEntityManagerFactory("flickr");
    EntityManager m = f.createEntityManager();
    Query q = m.
       createNativeQuery("select url from PHOTO");
    return q.getResultList();
}
```

EJB 3 Architecture Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence">
  <persistence-unit name="flickr"
                    transaction-type="RESOURCE_LOCAL">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>delme.hibernate.Photo</class>
    <properties>
      <property name="hibernate.connection.driver_class"
                value="org.apache.derby.jdbc.EmbeddedDriver"/>
      <property name="hibernate.connection.url"
                value="jdbc:derby:flickr;create=true"/>
      <property name="hibernate.dialect"
                value="org.hibernate.dialect.HSQLDialect"/>
    </properties>
 </persistence-unit>
</persistence>
```

Database Summary

- Desktop applications should use DataSet
  - RowSet and ResultsSet are just transport types
- Hibernate and EJB beans are too heavyweight for typical desktop needs

Agenda

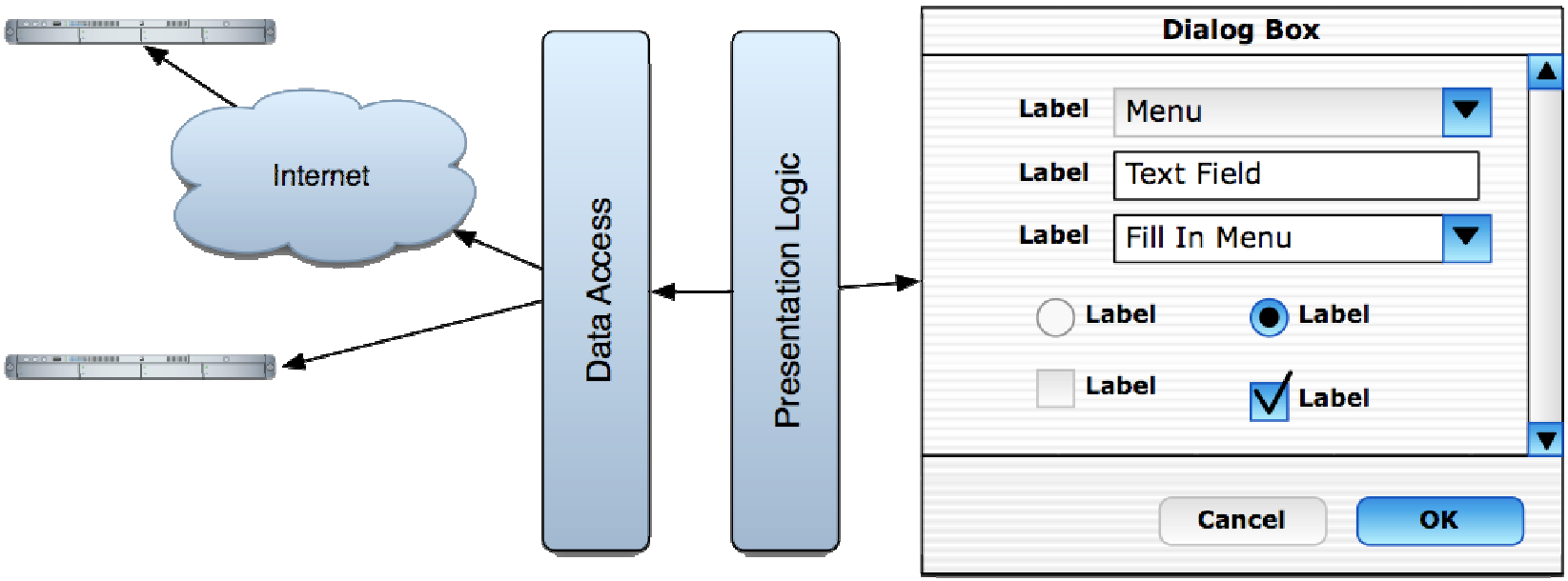
Smart Clients Defined

Application Design Overview

Demo Application

Summary

Separate Data Access Logic From UI!

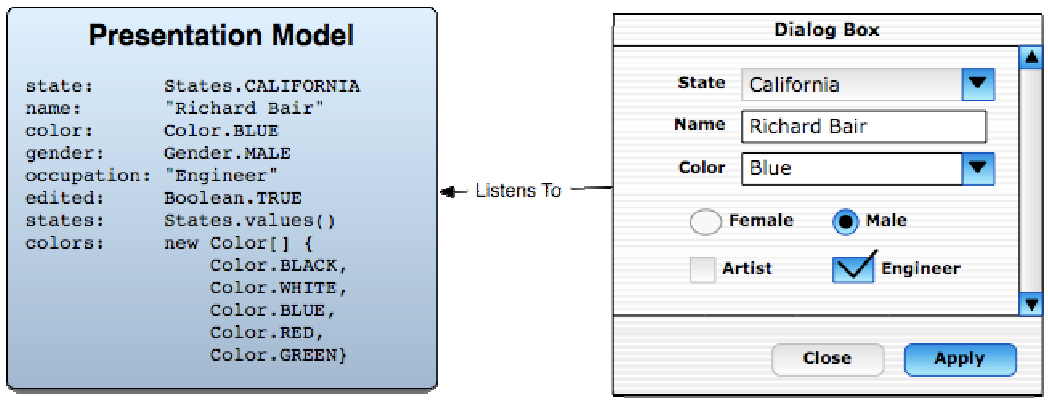


Separate Concerns

- Single most important concept
- UI should evolve according to users needs
- Data access should evolve according to technical needs
- Presentation Logic layer acts as adaptive layer to “glue” the UI to the data layer
  - Popular pattern: Presentation Model
- Individual circumstances may require adaptation

Observability

- Key part of JavaBeans™ architecture component model
- Allows construction of loosely coupled components
- Extremely useful in the data access layer and in the GUI layer



Typical Uses

- Data binding
  - Allows framework to observe changes and synchronize between beans
- Tools
  - For both visual and non visual components
- Observe status and update GUI as appropriate
  - isLoaded()

Design Problem #1

How Do I Send a Diff to the Server?

"How do I efficiently synchronize the server with small changes to data that is cached locally?"

Solution: Observability

- Write a DataObserver
  - Listens to property change events on your data objects
  - Records what data has been changed
  - Records the original value
- When preparing the diff
  - Consult the DataObserver for those fields that have changed

Design Problem #2

What Do I Do About Conflicts?

"Data on the server has changed; how do I detect conflicts between local data and changed data on the server?"

Solution: Observability

- Write a DataObserver
  - Listens to property change events on your data objects
  - Records what data has been changed
  - Records the original value
- Consult the DataObserver for conflicts
- Notify the UI of conflicts so the user can resolve them

Easy Observability

```
import org.jdesktop.swingx.JavaBean;
public class MyDataObject extends JavaBean {
    private boolean loaded;

    public boolean isLoaded() {
        return loaded;
    }
    protected void setLoaded(boolean b) {
        boolean old = isLoaded();
        this.loaded = b;
        firePropertyChange("loaded", old,
                isLoaded());
    }
}
```

Code Notes

- SwingLabs provides the JavaBeans technology class
  - Methods for adding/removing listeners
  - Methods for firing events
  - Trivial to implement observability
- “isLoaded()” is called in the setLoaded method
  - Notification works correctly even if “isLoaded” is overridden
- Methods don’t have to be public

Designing for the Network

- The network is unreliable
- Users are impatient

Rule #1: The Network Is Unreliable

- Network requests may not succeed
- Network response time may be unpredictable
  - Typical requests range from 150–5000 milliseconds

Rule #2: Users Are Impatient

- If an application is unresponsive users may
  - Distrust the application
  - Think the application has crashed
  - Quit the application and lose data
  - Look for other competing solutions
  - Take you off their Christmas list
- Users must be kept informed of progress

Solution: Threading

- Spawn a thread for all network related requests
  - SwingWorker makes it easier to manage background threads
    - http://swingworker.dev.java.net
  - Where possible, keep track of progress
    - HTTP responses may have a header indicating the number of bytes in the “page”

Threading

- The data access layer or presentation model layer is responsible for creating and managing these threads
- The UI layer must be notified of the status of these tasks
  - Progress indicators may need to be shown
  - Components may need to be disabled until the task is completed
- Threads must be cancelable, where possible

Threading

- Swing GUI should be updated only on the Event Dispatch Thread (EDT)
- This happens by default in Swing
  - Swing listeners always run on the EDT
- To update the GUI from a background thread
  - SwingUtilities.invokeLater (asynchronous)
  - SwingUtilities.invokeAndWait (synchronous)
- In Java 1.5 platform, java.util.concurrent package contains many very useful threading utilities
  - Thread pools
  - Cancelable tasks
  - Concurrent collections

Async Data Access

- Presentation Model for a GUI is notified of some action (button press)
- PM notifies data access layer to load some data
- Data Access layer creates background thread, returns a SwingWorker
- PM observes SwingWorker, updates UI state to reflect the loading status
- PM updates UI state when task is complete

Async Redux

- Presentation Model for a GUI is notified of some action (button press)
- PM observes “loaded” property of data object
- While not loaded, PM observes “status” property of data object, updates UI accordingly
- When “loaded”, UI is updated with data

Agenda

Smart Clients Defined

Application Design Overview

Demo Application

Summary

Swing + Mashups = Smashups

- Easy and powerful
  - The hard work is done by the web service providers on the server
  - Many popular web services already have data access logic written for Java technology
  - New Swing components for popular web services are being implemented in SwingLabs
- Best of both worlds

Slickr

- Sample “Smashup” application
  - The beginnings of a general Flickr client
  - Flickr photo viewer and trip reporter
- http://aerith.dev.java.net

Demo

Aerith

Demo Notes

- Flickr images are wrapped in a PhotoWrapper
  - PhotoWrapper notifies when the real photo image is loaded
- Google Map tiles are wrapped in a Tile
  - Notifies when the tile image is loaded
  - Maintains a SoftReference for the image
- ResponseCache implementation for caching images (both tiles and photos)

Agenda

Smart Clients Defined

Application Design Overview

Demo Application

Summary

Summary

- Smart Clients use WebServices
- Smashups are easy and powerful additions to your applications
- Cache data for offline use
- Perform network tasks asynchronously
  - Inform the user!
- Inform the user of their connection state
- Many database access options available

For More Information

- Technical Sessions
  - TS-1074: Desktop Patterns and Data Binding
  - TS-4855: Swing Threading 101
- BOFs
  - BOF-0614: SwingLabs
- URLs
  - http://www.martinfowler.com/eaaDev/PresentationModel.html
  - http://swinglabs.org
  - Google for: Microsoft Smart Client design guide
  - http://www.flickr.com/services/api

Q&A

Richard Bair

Best Practices: Data Access Strategies

Richard Bair

SwingLabs Lead
Sun Microsystems
http://swinglabs.org

TS-4635