

Application Model for Binding Interactive Java Clients to a Remote Server

java.sun.com/javaone/sf

John T. Kucera
Founder & CTO
Asperon Corporation
<http://www.asperon.com>

ASPERON



How to build a rich client in Java.

Web users are demanding more interactivity

Learn, by example, some techniques for scalably connecting a Java client to a remote J2EE application over the web, and make the client respond like the application is right on your desktop.

Agenda

The resurgence of client-side Java
Challenges of building interactive clients
Binding UI controls to remote data
Example Application Model
Optimizing client-server communications
Summary

Agenda

The resurgence of client-side Java

Challenges of building interactive clients

Binding UI controls to remote data

Example Application Model

Optimizing client-server communications

Summary

Why Use a Java Client?

HTML UIs are too limited for “power users”

- Low productivity
 - Page after page to get anything done
 - Mixes navigation with actions
 - No field-level validation for complex data
 - Inability to handle large data sets
- High runtime bandwidth
 - Resends entire page not just changed data
- JavaScript not a “real” language
 - Good only for simple interactivity (e.g. Rollovers)
 - Poor separation of UI from data
 - Hard to maintain

Many Old Problems Have Been Solved!

What has changed since 1998?

- Broadband connections common (DSL/Cable)
 - 56 Kb/sec => 1.5 Mb/sec
 - 1 MB now downloads in just 6 seconds!
- Improved startup times
 - Older browsers did not pre-load JVM
 - Improvements to Swing initialization
 - JIT compiler improvements
- Much faster CPUs
 - 233 Mhz => 2.0+ Ghz
- Improved compatibility
 - Most PC vendors bundle current Java versions
 - Internet Explorer compatible component sets

Agenda

The resurgence of client-side Java

Challenges of building interactive clients

Binding UI controls to remote data

Example Application Model

Optimizing client-server communications

Summary

Interactive Client Development

It's harder up front, but worth the effort

- Can't just emit HTML and rely on browser to do all the work!
- Programmatic component layout
- Event handling
- Field-level (not page level) validation
 - Must override request-response model to send validation and other data
- Manual data binding and caching
- Multithreaded, asynchronous behavior
 - Must wait for response from server
 - Cannot block rendering thread

Agenda

The resurgence of client-side Java

Challenges of building interactive clients

Binding UI controls to remote data

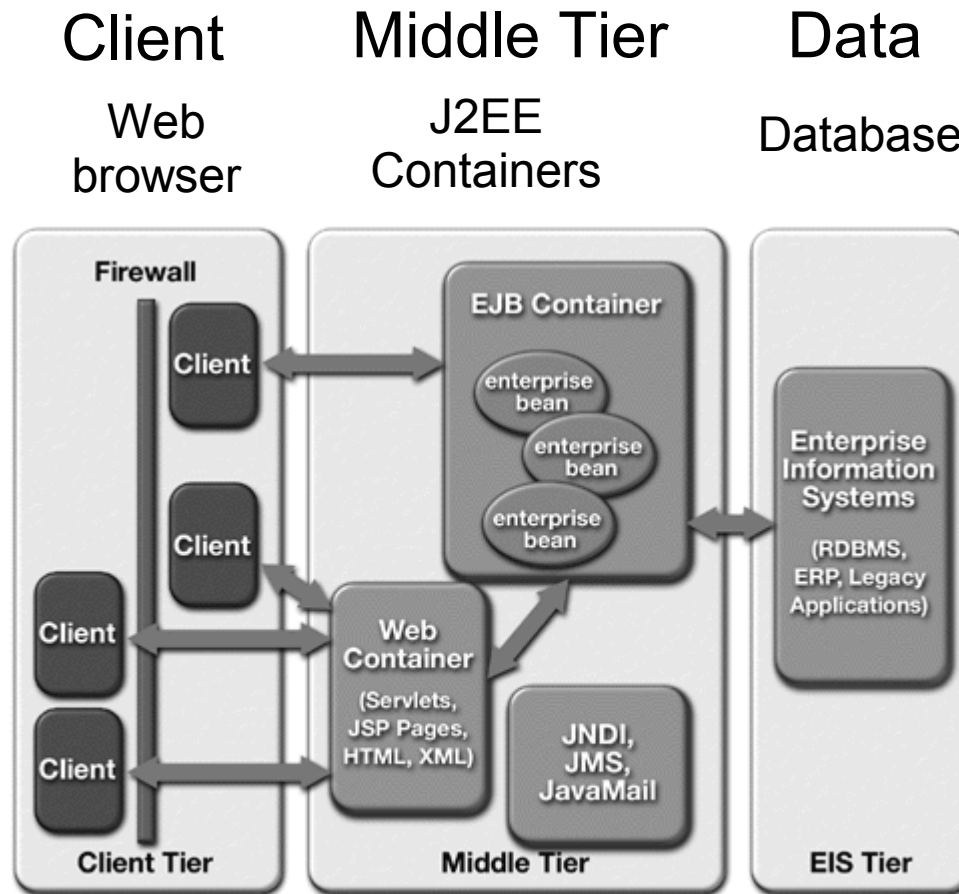
Example Application Model

Optimizing client-server communications

Summary

Application Architecture

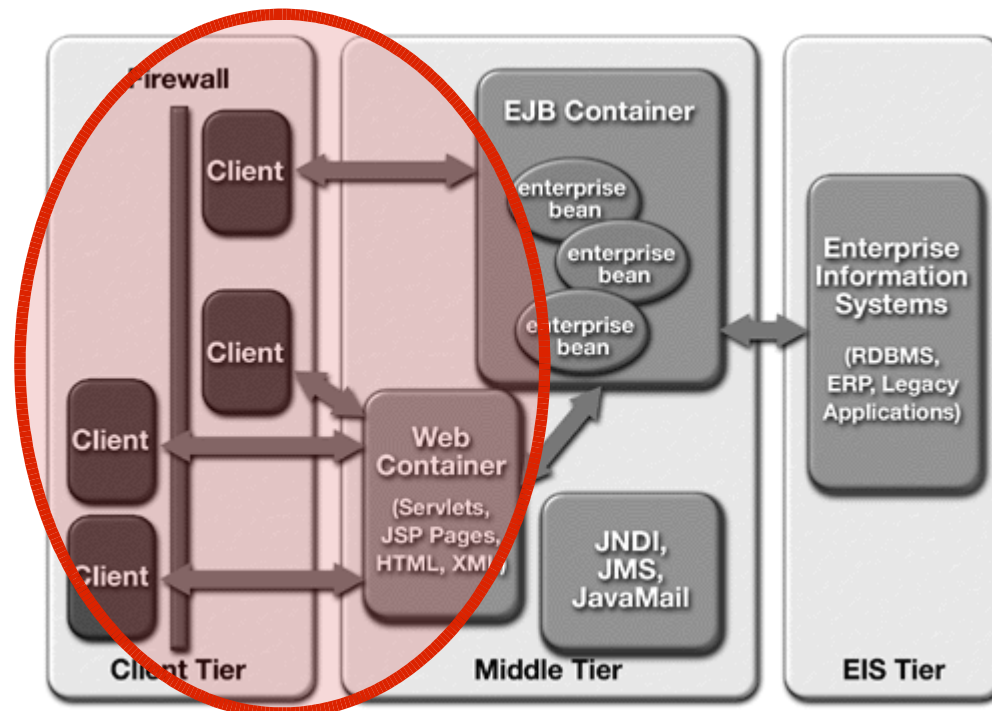
Overview of a 3-tier application



Focus on Front End

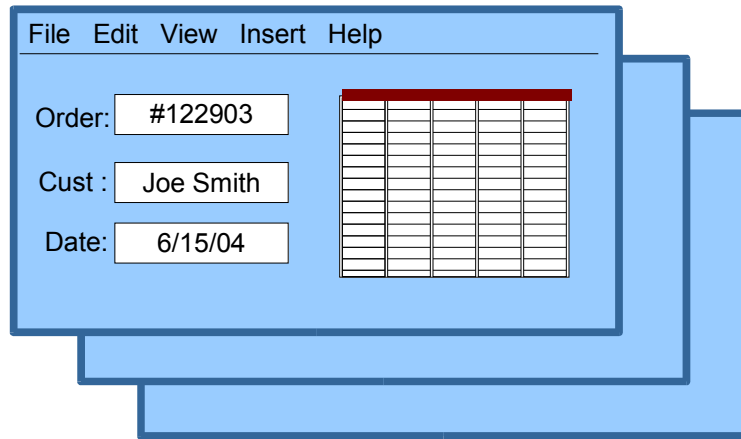
Overview of a 3-tier application

- Focus on binding UI to middle tier
 - Not Object-Relational modelling
 - Not EJB or related technologies



How to Bind UI to Objects?

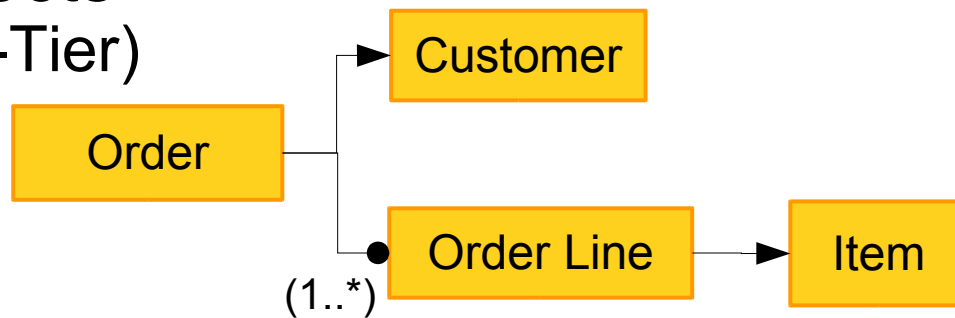
Screens
(Client)



Flat views with multiple objects on each screen



Objects
(Mid-Tier)



Complex object associations

Create View Objects

Combine complex business objects into views

- Entities vs Views
 - Ex: Order + Customer = OrderView
- Useful for 1:1 or N:1 relationships
- Utilize joins if possible

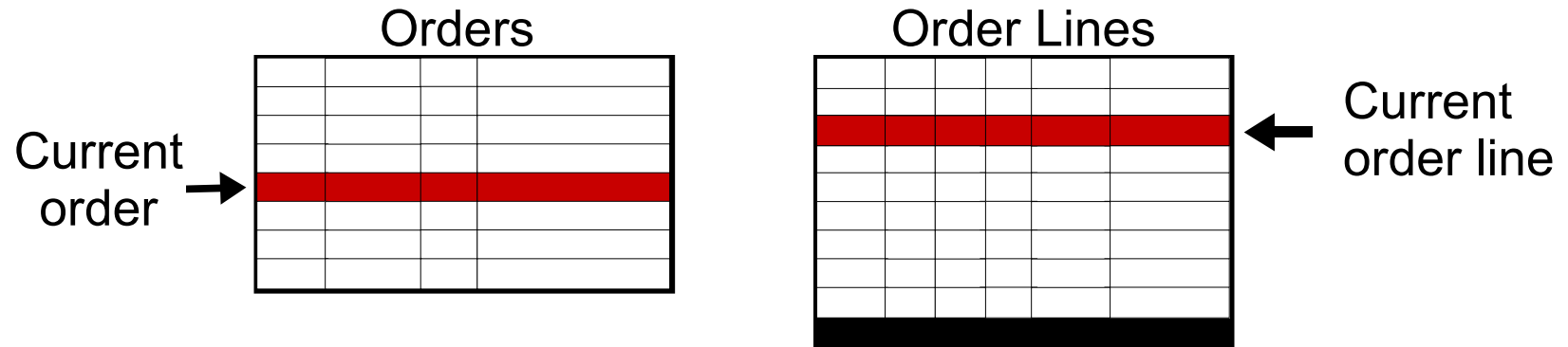
Order Screen	Ship Date:	6/19/04
Order No:	122903	
Order Date:	6/15/04	
Customer:	Sun Microsystems	
Item Count:	3	Total Price: \$129.95

Name comes from Customer entity (F.K.)

Context for Master-Detail Relationships

Flatten hierarchies using context information

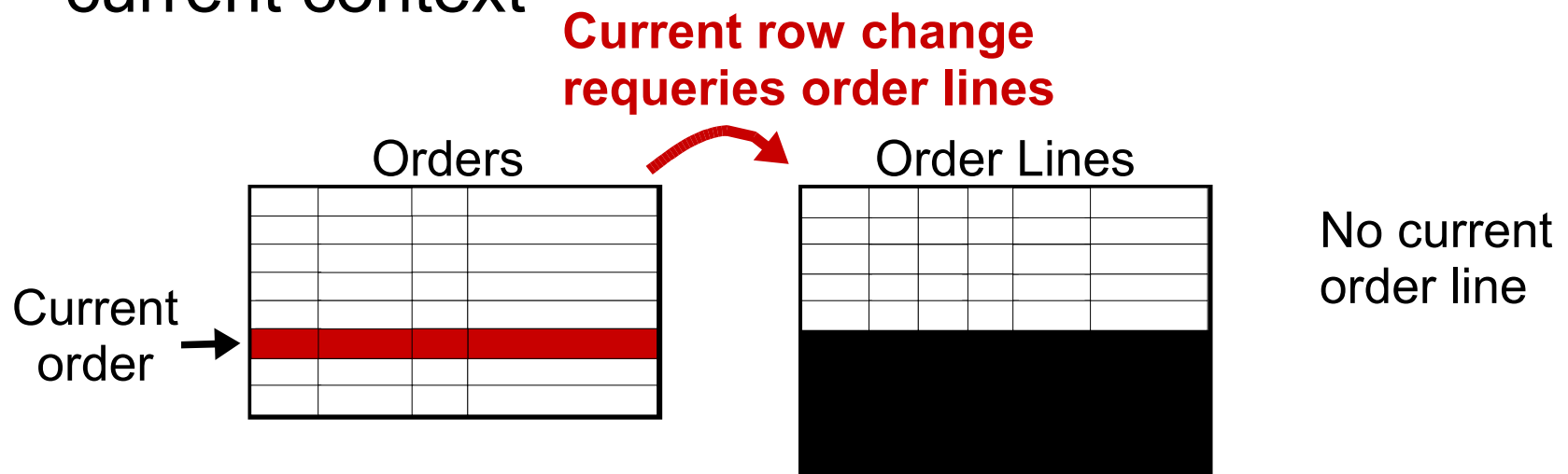
- Define a “current row” for a table, and a “current node” for a tree
- Use current row/node to requery details when master changes
- Set of current rows and nodes becomes the current context



Master-Detail Relationships

Flatten hierarchies using context information

- Define a “current row” for a table, and a “current node” for a tree.
- Use current row/node to requery details when master changes
- Set of current rows and nodes becomes the current context



Map Isomorphic Data Structures

Bind each UI control to its corresponding type

UI Control

Data Type



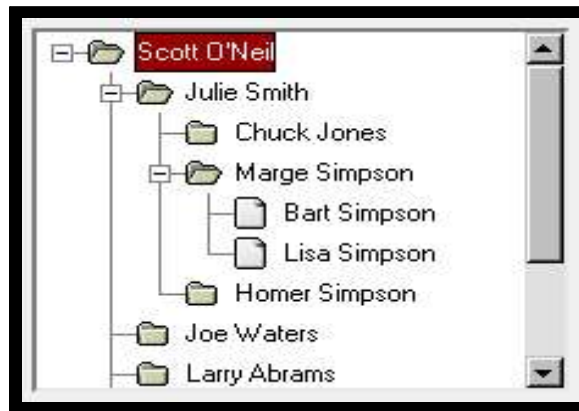
Scalar (String, Boolean)

First Name	Last Name	Phone	email
John	Doe	650 234-5678	jdoe@microsoft.com
Gus	Chi	650 876-9534	gus@vensoft.com
Michael	Sams	650 654.3210	samols@amweb.com
Jim	Marsh	650 887-7723	jim@svbventureone.com

Tabular Data

Uniform rows (data type)

Named and typed cols



Hierarchical Nodes

Each node contains data

Each node has 1 parent

Parents have N children

Agenda

The resurgence of client-side Java

Challenges of building interactive clients

Binding UI controls to remote data

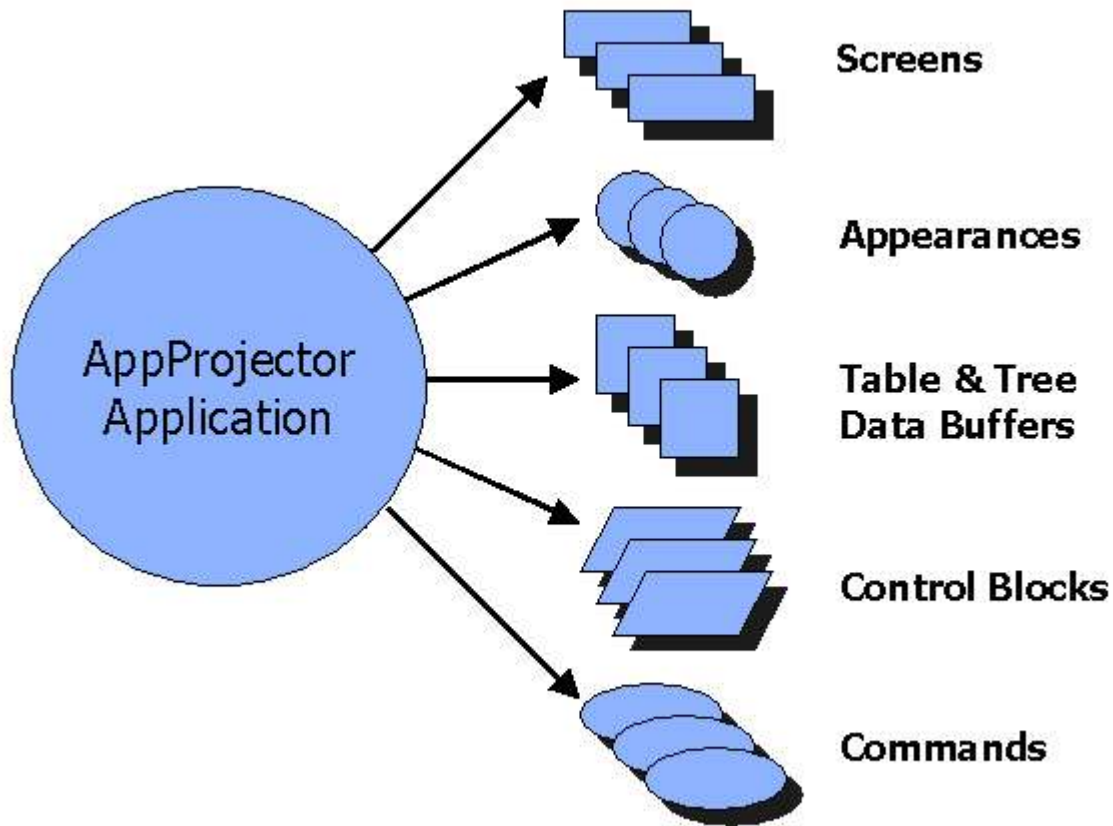
Example Application Model

Optimizing client-server communications

Summary

Example Application Model

Proven model - used in production system 1 yr



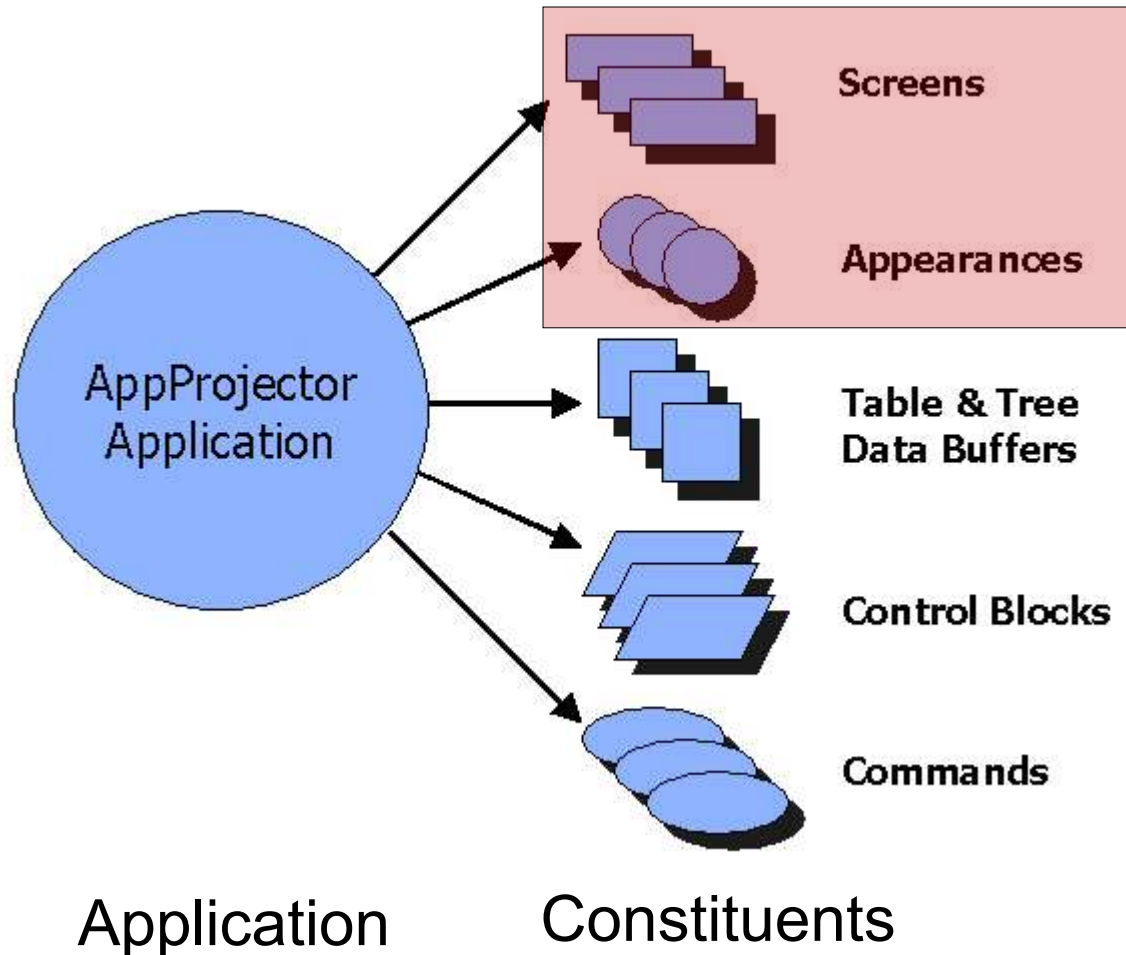
Application

Constituents

- Very general
- Application is a Java interface
- Constituents are all Java interfaces
- Named constituents permit remote reference

Example Application Model

Screens and Appearances



- Very general
- Application is a Java interface
- Constituents are all Java interfaces
- Named constituents permit remote reference

Screens and Appearances

Declarative representation in XML

- Use XML to represent UI
 - Eliminates need to hand-code to component set API
 - Makes it easier to change the UI quickly
 - Separates UI from business logic
- Generic rendering engine
 - Handles all user interfaces with the same code
- Context: Current screen, current appearance
- Appearances are templates for style (like CSS)

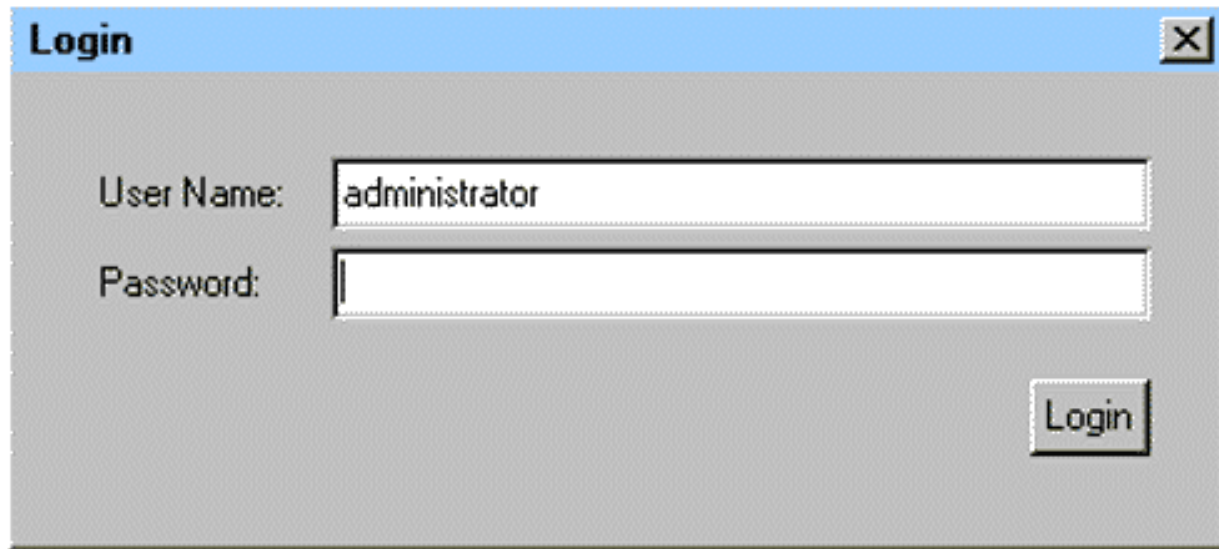
Sample XML Description – Login Dialog

UI components bound to scalar data elements

```
<dialog name="loginDialog"> // Uses GridBagLayout
...
  <label gridx="1" gridy="1" text="User Name:" />
  <textField gridx="2" gridy="1" charsWide="40"
    dataSource="loginParams"
    dataField="username" />
  <label gridx="1" gridy="2" text="Password:" />
  <textField gridx="2" gridy="2" charsWide="40"
    dataSource="loginParams"
    dataField="password" />
  <button gridx="2" gridy="3" align="right"
    text="Login" command="loginCommand" />
...
</dialog>
```

Resulting Screen

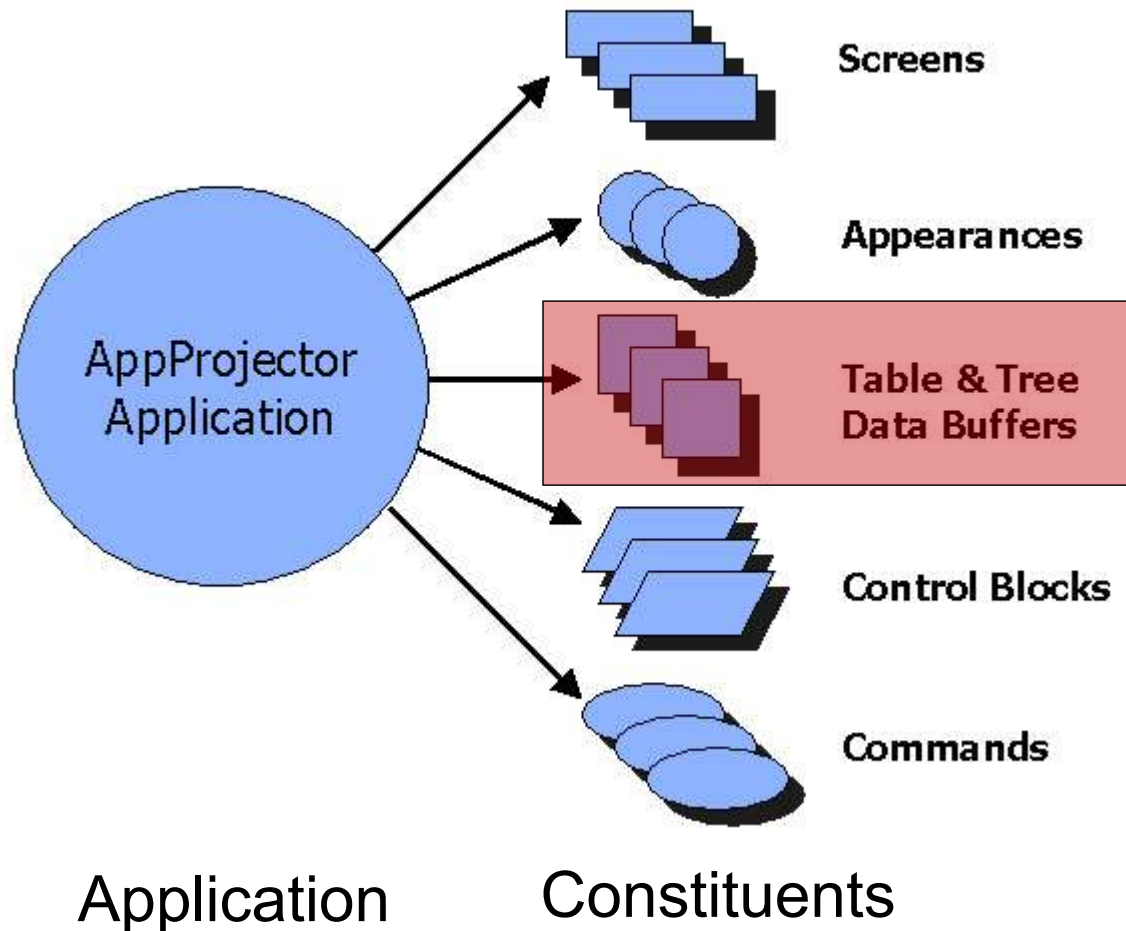
XML converted to GUI widgets



- Data binding ensures text in UI controls kept in synch with data object to which they are bound
- Synchronize data between text field and data object after losing focus on the field.

Example Application Model

Table and Tree Data Buffers



- Very general
- Application is a Java interface
- Constituents are all Java interfaces
- Named constituents permit remote reference

Table Data Buffers

Holds tabular data

- Rows and columns of data
- Named, typed columns
- Current row management
- Emits events when rows inserted, updated, deleted, or when current row is changed

First	Last	Email	Phone
Jack	Smith	jack.smith@sun.com	415.344.3459
Jeff	Smith	jeff.smith@sun.com	415.372.9033
Joe	Smith	joe.smith@sun.com	650.558.0439
John	Smith	john.smith@sun.com	510.273.7292
Joy	Smith	joy.smith@sun.com	408.422.1023

← Current row

Table Data Buffer Interface

Bound to tables and scalar UI components

```
public interface TableDataBuffer extends Named
{
    // Column names and data types
    public TableColumn[] getColumns();

    // Setting and getting values
    public Object getValue(int col); // on current
    public void setValue(int col, Object val); //row
    public Object getValue(int row, int col);

    // Current row management
    public void setCurrentRow(int row);
    public int getCurrentRow();
    public void addCurrentRowListener(CRL l);
    public void removeCurrentRowListener(CRL l);
}
```

Tree Data Buffer Interface

Defines structural aspect of a tree

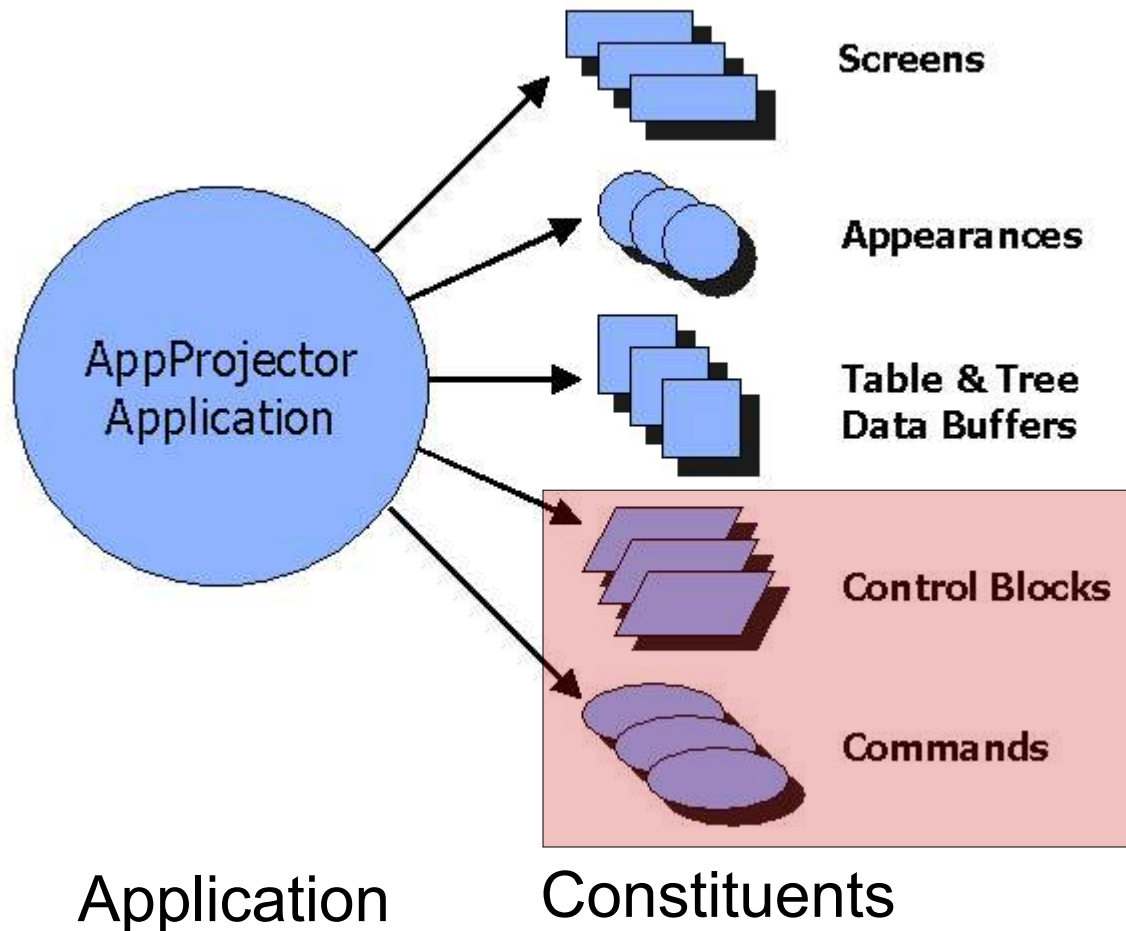
```
public interface TreeDataBuffer extends Named
{
    // Node data management
    public Object getRoot();
    public Object getParent(Object node);
    public int getChildCount(Object parent);
    public Object getChild(Object parent, int i);
    public Object getNodeValue(Object node);

    // Current node management
    public Object getCurrentNode();
    public void addCurrentNodeListener(CNL l);
    public void removeCurrentNodeListener(CNL l);

    // + listeners for node insert, update, delete.
}
```

Example Application Model

Control Blocks and Commands



- Very general
- Application is a Java interface
- Constituents are all Java interfaces
- Named constituents permit remote reference

Control Blocks

Named groups of scalar parameters

```
public interface ControlBlock extends Named
{
    // Parameters and values
    public Enumeration getParameterNames();
    public Object getParameterValue(String name);
    public void setParameterValue(String name,
                                  Object val);

    // Informed after parameter values change
    public void addParameterChangeListener
                (ParameterChangeListener l);
    public void removeParameterChangeListener
                (ParameterChangeListener l);
}
```

Commands

Bound to buttons and menu items

```
public interface Command extends Named
{
    // Enablement
    public boolean isEnabled();
    public void setEnabled(boolean enabled);
    public void
        addEnabledListener(EnabledListener l);
    public void
        removeEnabledListener(EnabledListener l);

    // Execution
    public void execute() throws Exception
    {
        // Command execution code goes here...
    }
}
```

Agenda

The resurgence of client-side Java

Challenges of building interactive clients

Binding UI controls to remote data

The Application Model

Optimizing client-server communications

Summary

Cache Application Data on Client

The key to caching is forgetting!

- Proxy application model on client
- Client-side data cache allows faster response
 - Example: Scroll table to previously viewed rows
- Client can fetch chunks of records as needed to minimize effects of network latency
- Need to keep cache in sync with server
 - e.g. Clear cache when server data clears
- Must forget some cached data to prevent `java.lang.OutOfMemoryException` on client

Bundle Messages

Needed to minimize effects of network latency

- Latency vs. Bandwidth
 - Latency = Delay in sending first byte
 - Bandwidth = Number of bytes per second
 - Latencies up to 300 mS for satellite transmission
- Divide messages into those that can be postponed, and those that must be sent immediately.
- Send sets of messages together to minimize number of round trips.
- Wait a few mS before sending postponable messages to server in case more arrive.

Use Re-entrant Message Processing

Handling responses that generate requests

- Response to client request may generate another request
- Example:
 - 1) client => server: “execute query on table T”
 - 2) server => client: “10 rows inserted in T at index 0”
 - 3) client => server: “Fetch data for T, rows 0-9”
 - 4) server => client: “Row data for T, rows 0-9”
- Should second request from client to server be send while first response is being processed or delayed until the first response is finished?
- Both work, but need to consider design prior to implementing. (HTTP 1.1 limitations)

Demo

The Application
Model in Action



Summary

- Client-side Java is returning to the web
- Need to make it easier to build Java clients
- Challenge is efficiently binding UI to data
- Presented example of an application model that permits easy data-UI binding and clean separation of UI from business logic
- Caching and message bundling can be used to make weakly-connected web clients nearly as responsive as if they were on the desktop

For More Information

- BOF Session #2891
- Asperon Home Page:
 - <http://www.asperon.com>
- AppProjector User's Guide:
 - http://www.asperon.com/AppProjector_UsersGuide.pdf
- White Papers:
 - Asperon AppProjector Presentation Server Technical Overview
 - http://www.asperon.com/AppProjector_whitepaper.pdf
- *Making the Case for Client-Side Java:*
 - Java Developer's Journal Online, Nov 6, 2003.
 - <http://www.sys-con.com/story/?storyid=37819>

Q&A

John T. Kucera

Asperon Corporation



Application Model for Binding Interactive Java Clients to a Remote Server

java.sun.com/javaone/sf

John T. Kucera
Founder & CTO
Asperon Corporation
<http://www.asperon.com>

ASPERON

