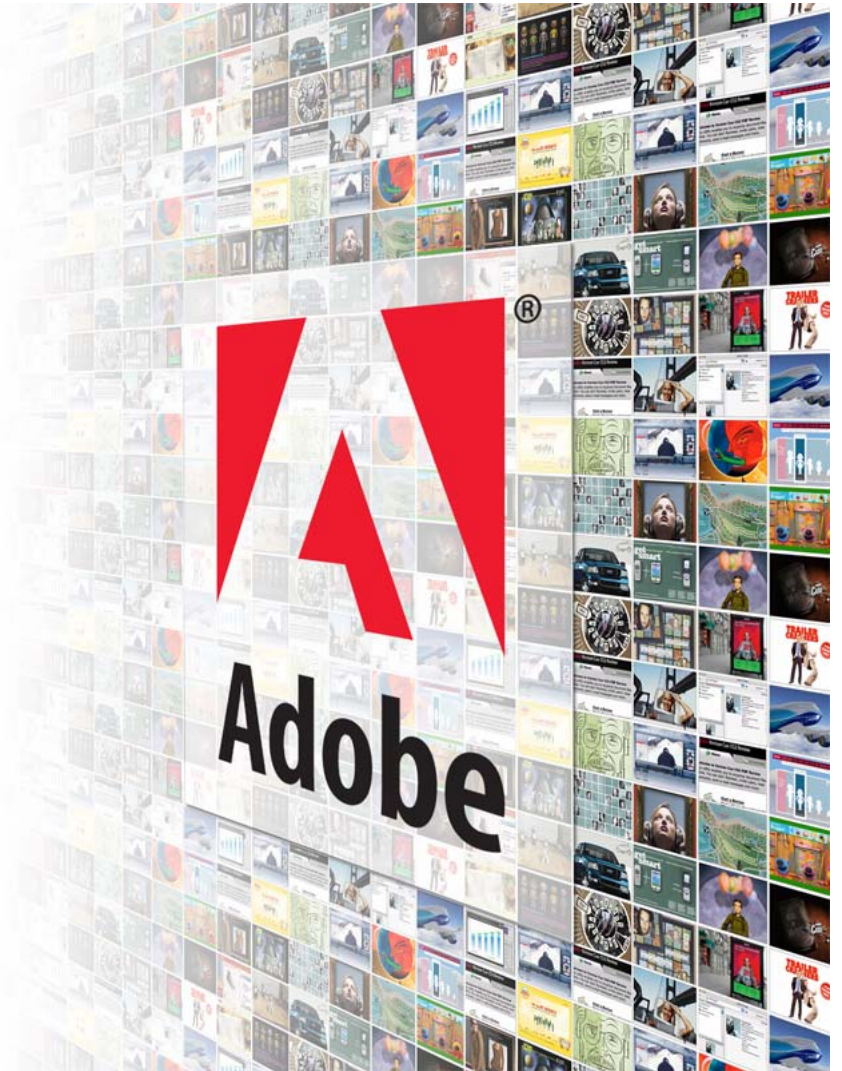


# Flex Data Services

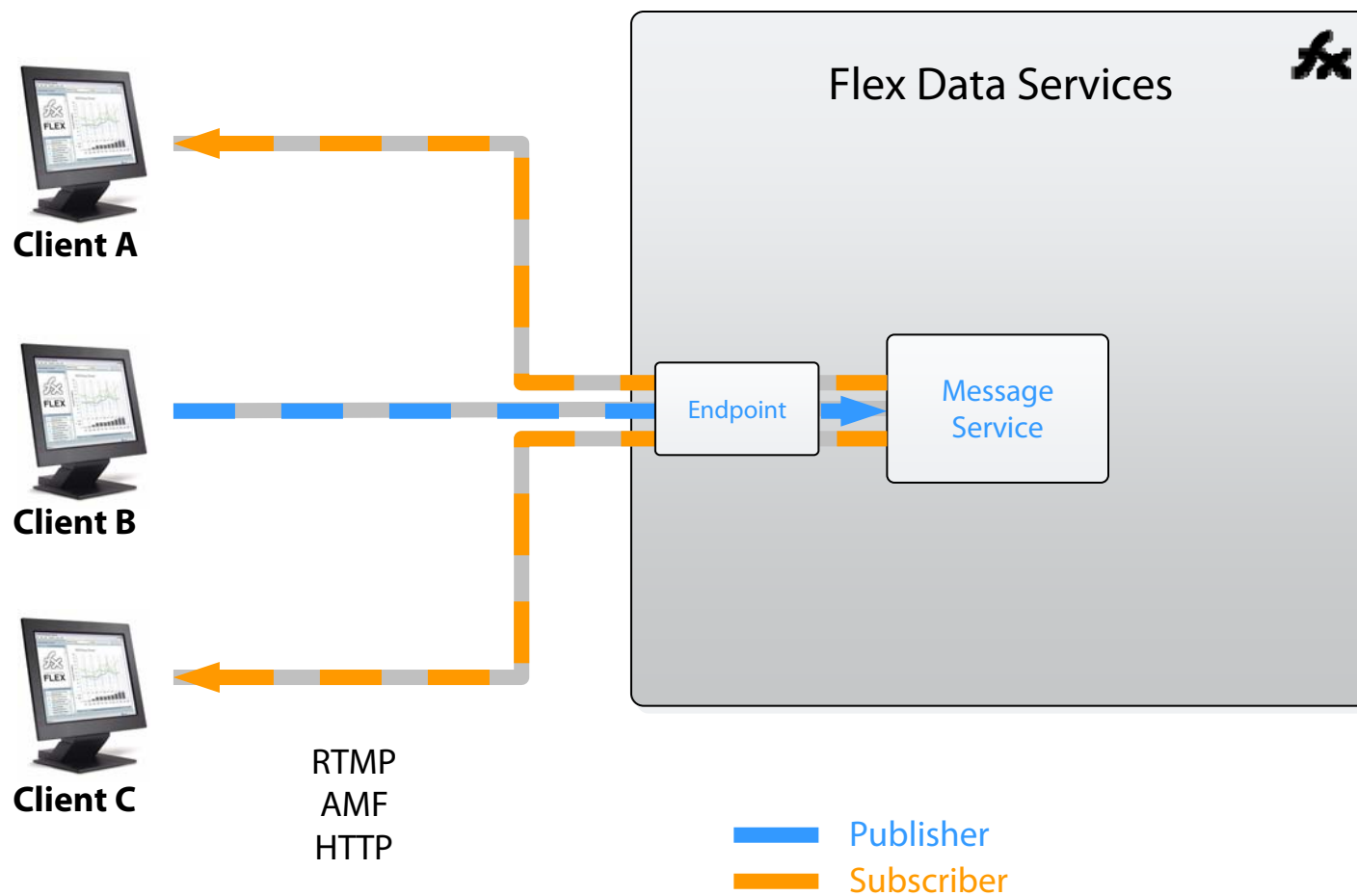
Christophe Coenraets  
Adobe Systems



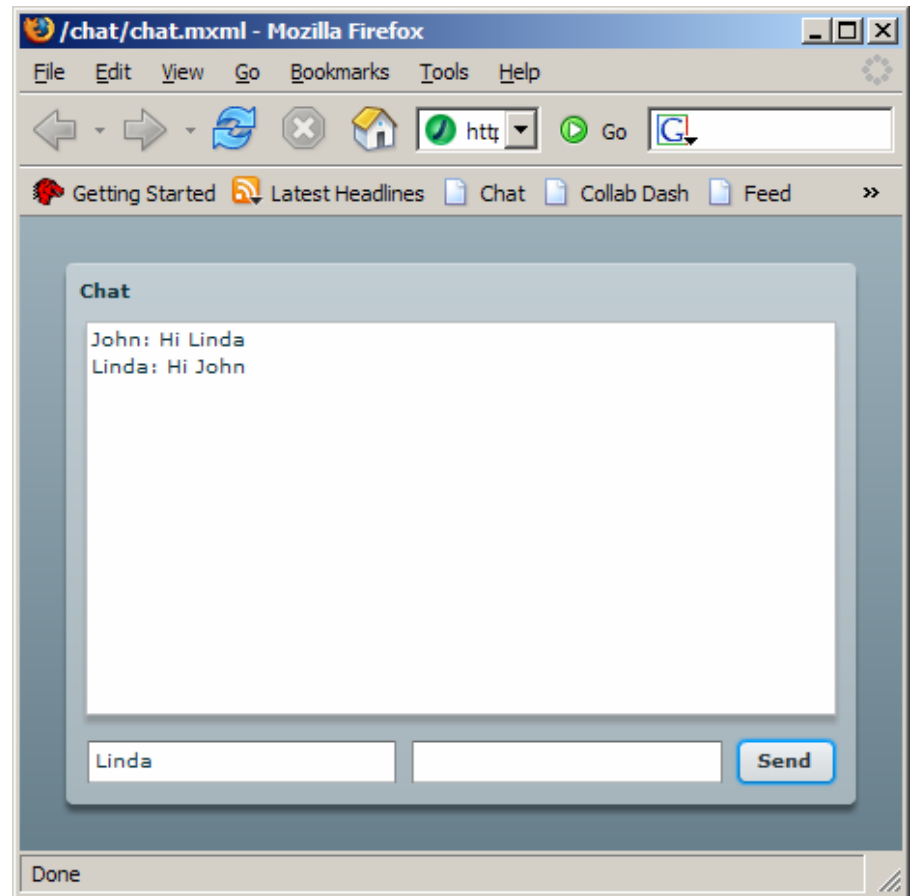
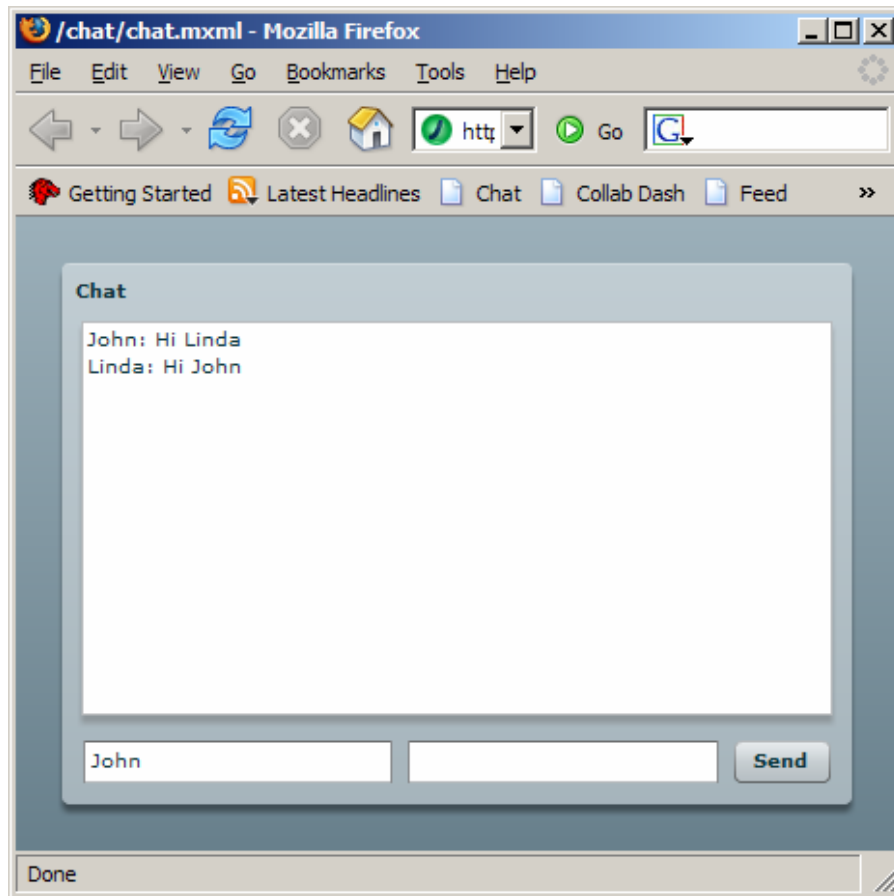
# Building Data-Rich Internet Applications

- In current RIAs, the user interface is fluid, but the data is not
- Traditional limitations:
  - Lack of publish/subscribe messaging
  - Data synchronization between tiers is complex

# Flex 2.0 Publish/Subscribe Messaging Simple Use Case



# Demo: Chat



# Publishing a Message

- Setting up a Producer

```
<mx:Producer id="producer" destination="chat"/>
```

- Sending the message

```
public function send():void
{
    var message:AsyncMessage = new AsyncMessage();
    message.body.userId = userId.text;
    message.body.msg = msg.text;
    producer.send(message);
}
```

# Subscribing to a Destination

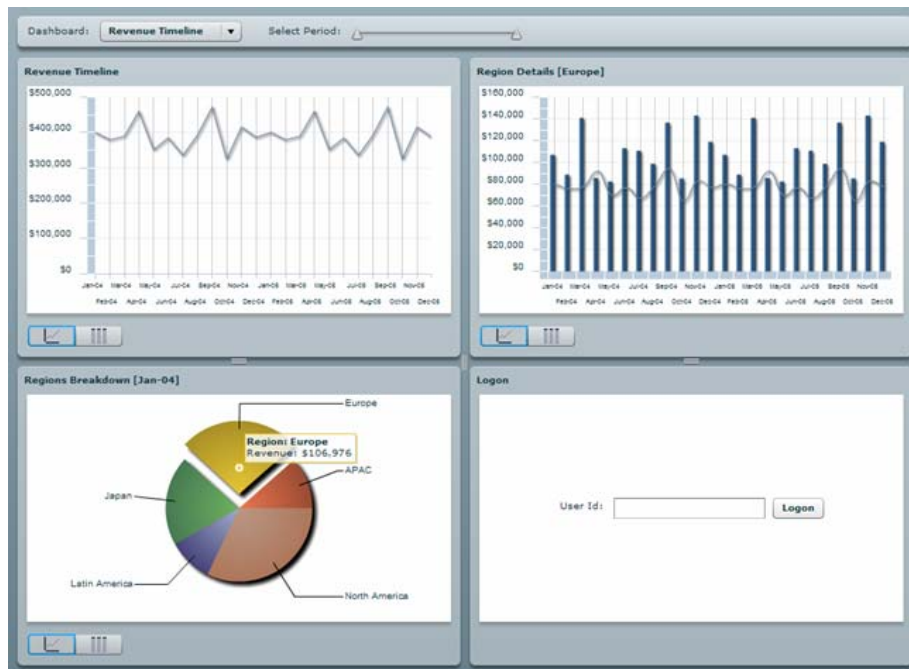
- Setting up a Consumer

```
<mx:Consumer id="consumer" destination="chat"  
    message="messageHandler(event)"/>
```

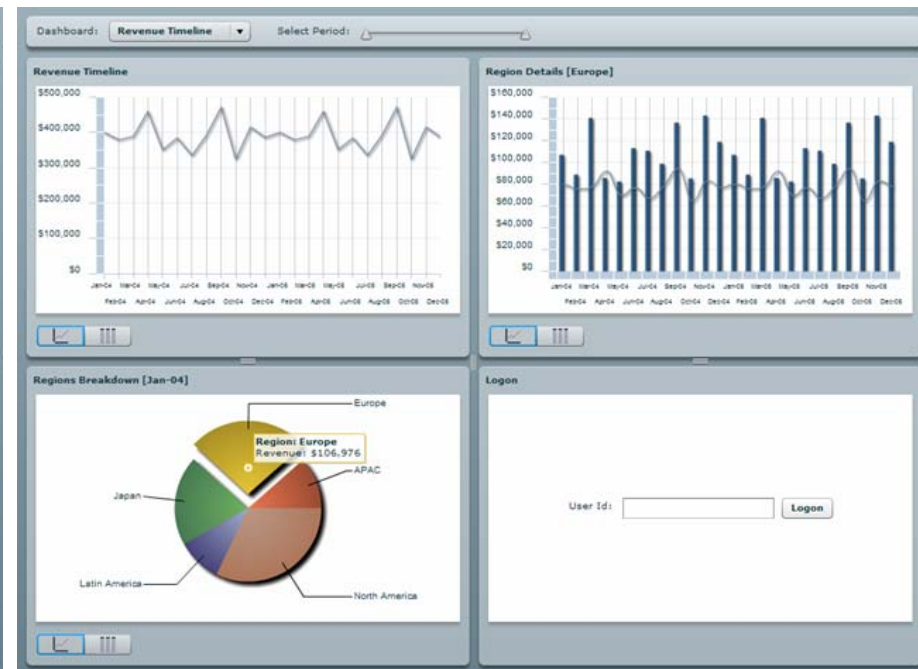
- Handling messages

```
public function messageHandler(event:MessageEvent):void  
{  
    log.text += event.message.body.userId + ": " + event.message.body.msg + "\n";  
}
```

# Demo: Collaboration Dashboard Application Sharing

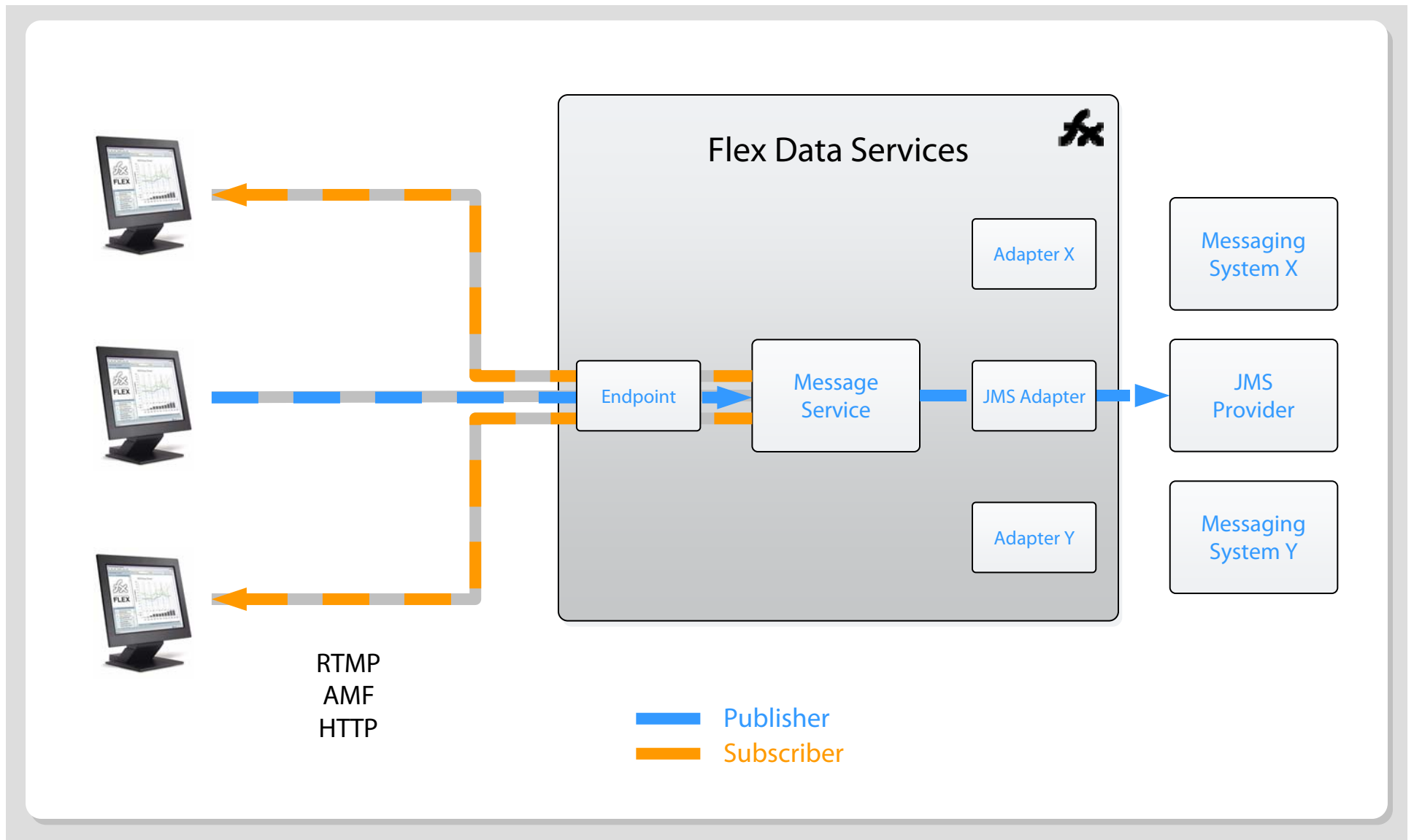


John's session



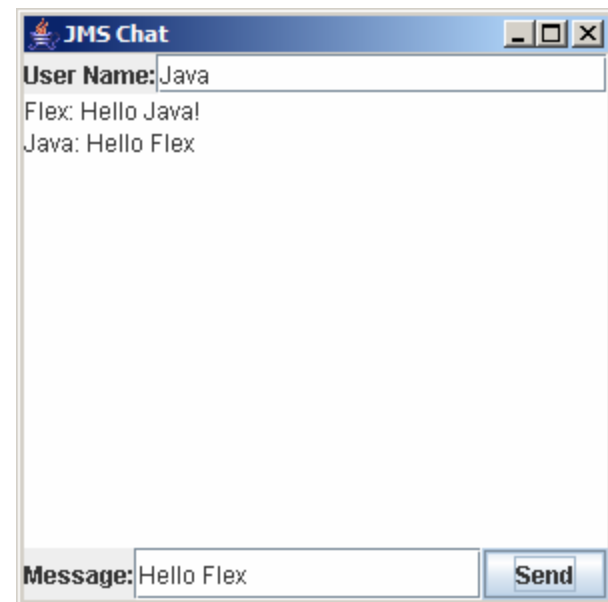
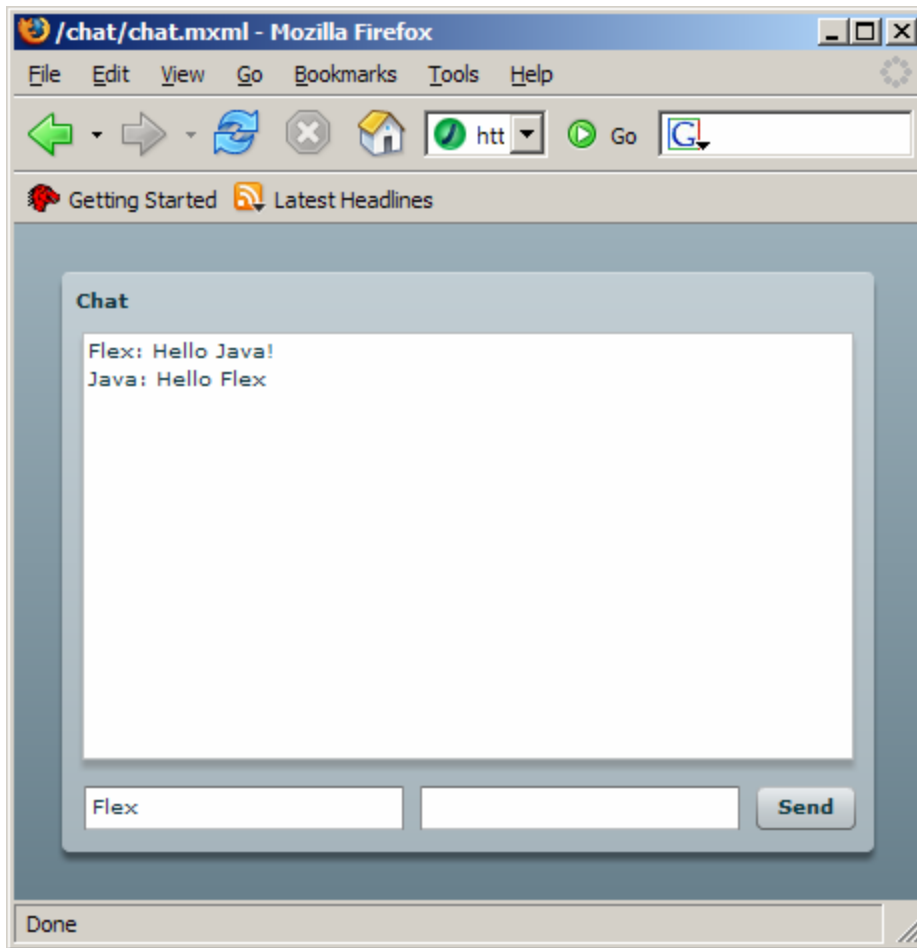
Linda's session

# Flex 2.0 Messaging

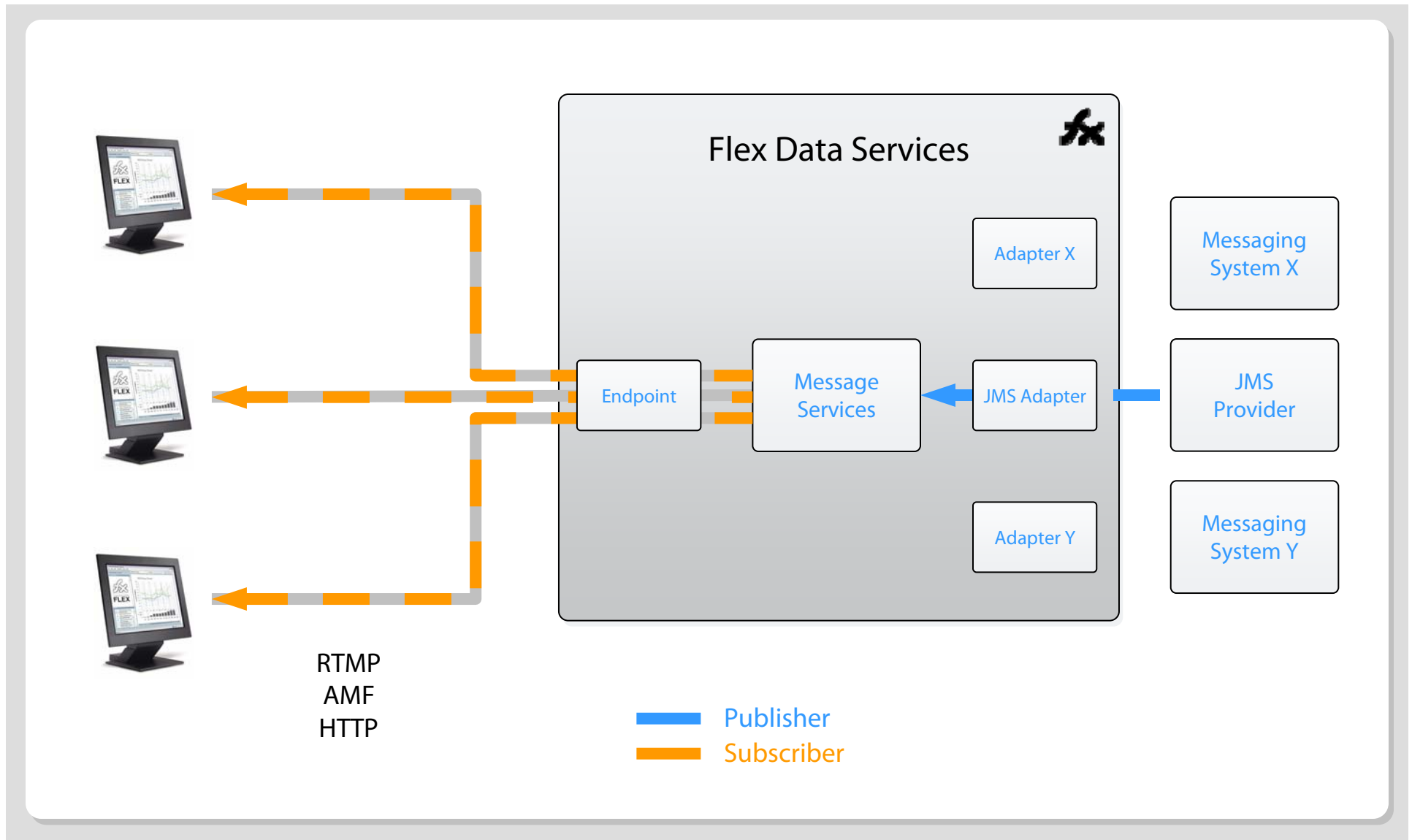




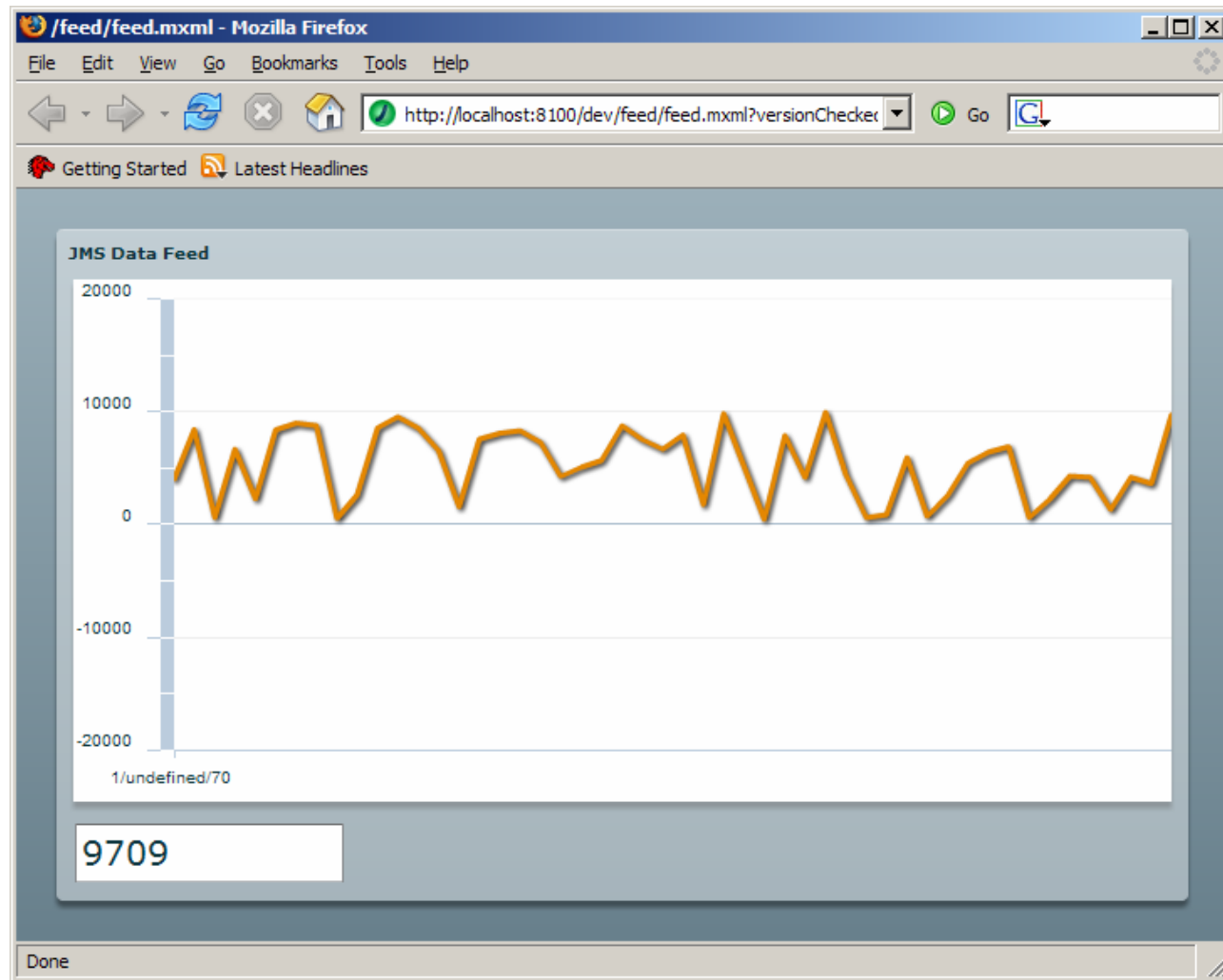
# Demo: JMS Chat



# Flex 2.0 Messaging Data Push



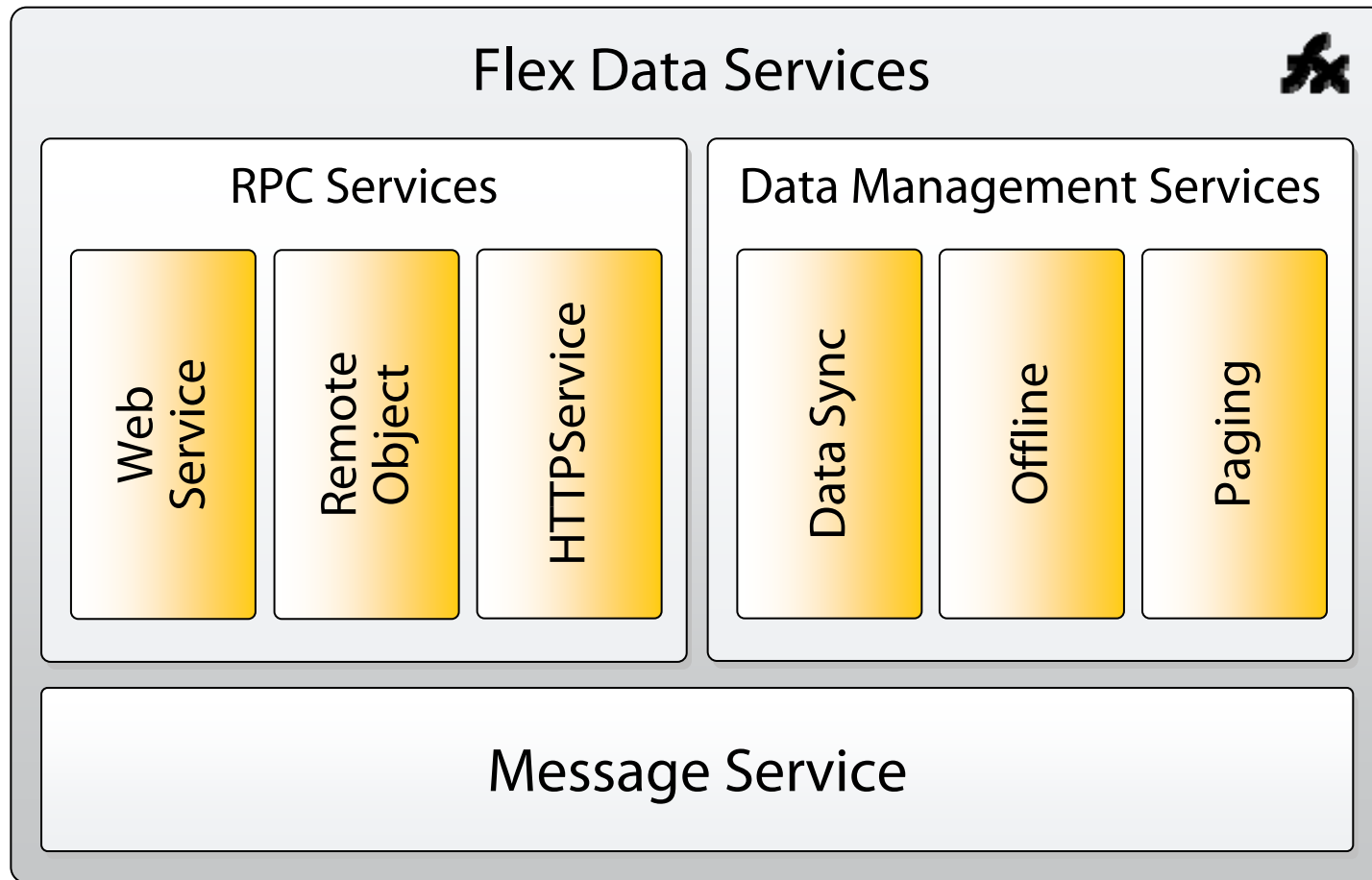
# Demo: Real Time Data Feed (JMS)



# Messaging Summary

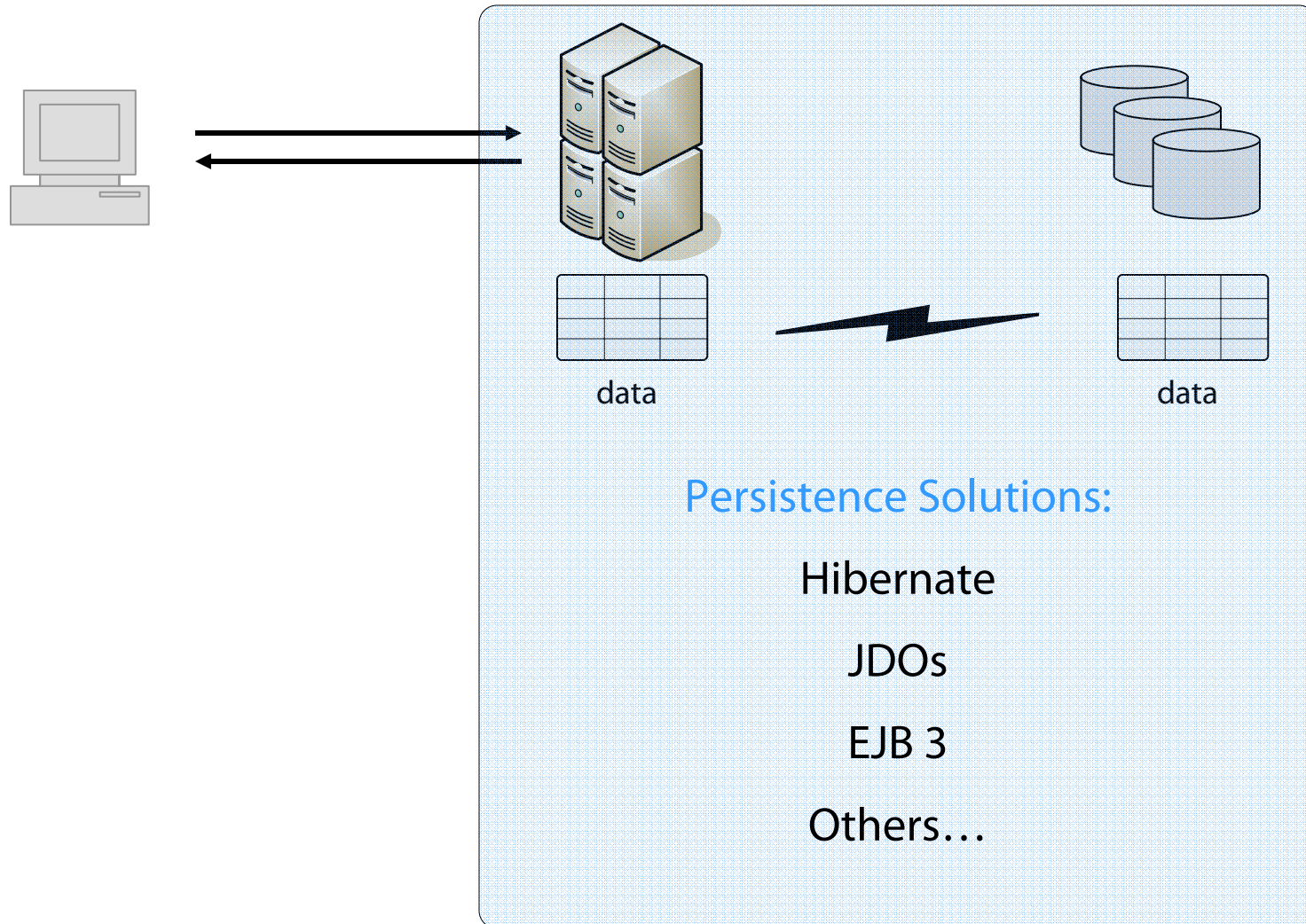
- Flex messaging enables new types of applications
  - Data push
  - Collaboration
  - Data Synchronization
- Destinations are configured in xml file
- Messages and channels are decoupled
- Destinations are available to Flex and non-flex applications

# Data Services



## Working with Data

### Traditional Page-Centric HTML Model

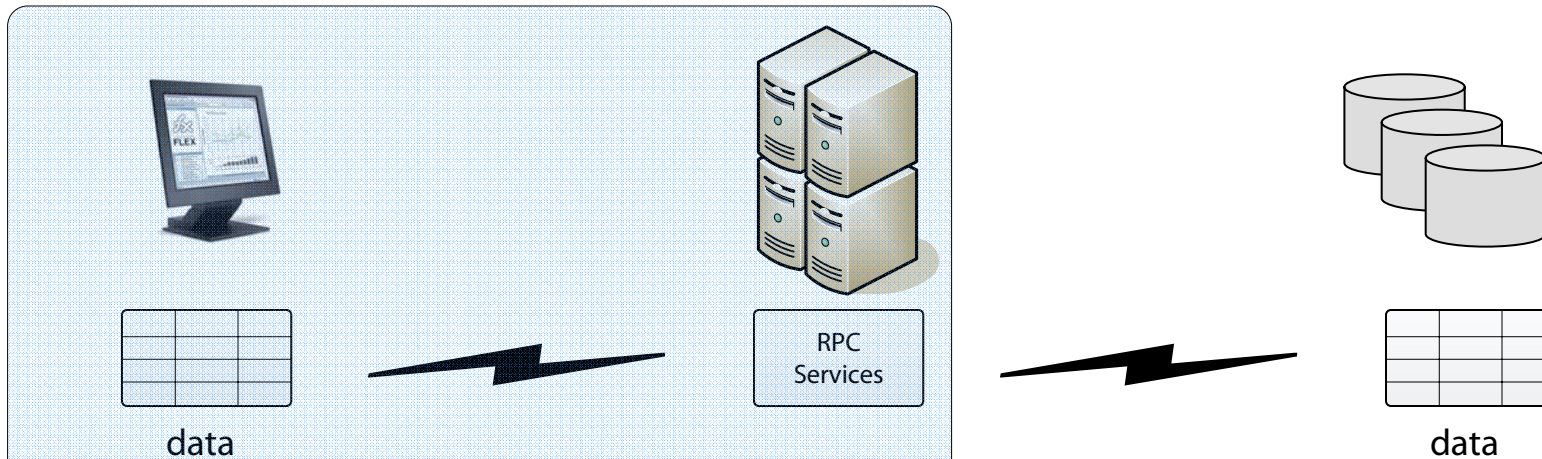


# RIA Model



- Client owns a version of the data
- Rich client-side data manipulation
- Data needs to be synchronized between **client-tier and middle-tier** and between middle-tier and data store

# RPC Approach

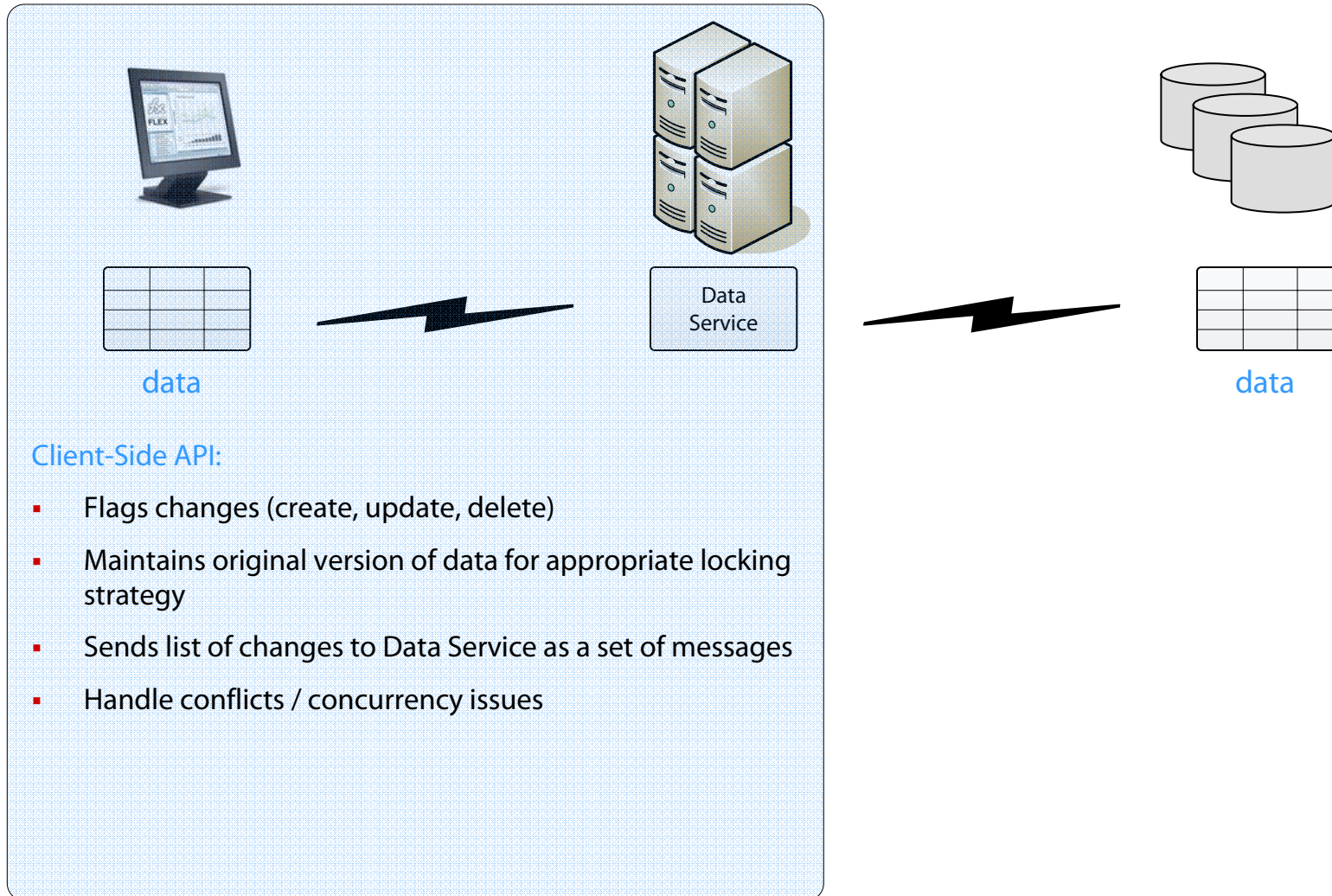


## Developer's Responsibilities:

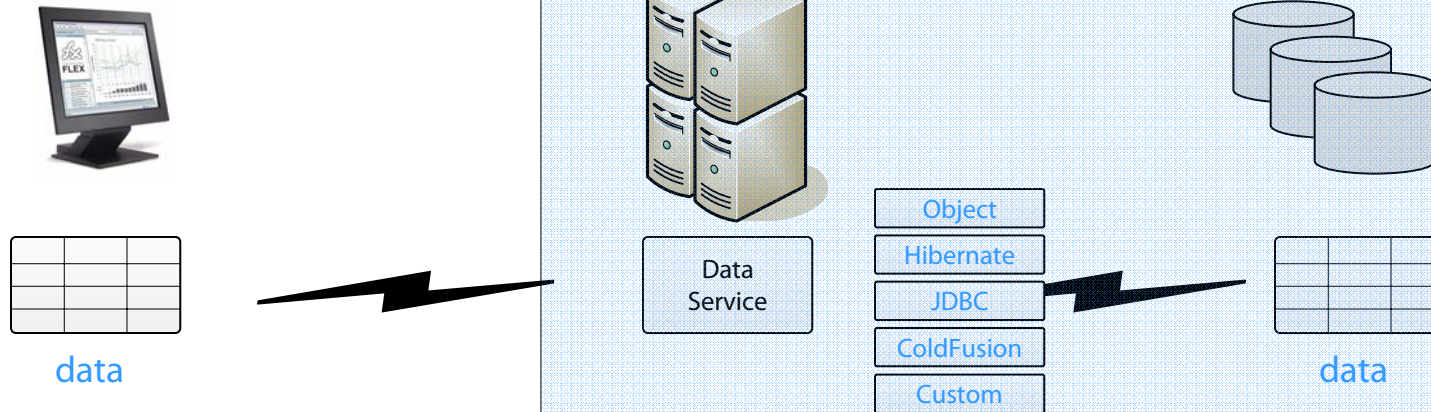
- Flag changes (create, update, delete)
- Maintain original version of data for appropriate locking strategy
- Make Remote Procedure Calls for creating, updating, deleting items
- Identify and handle conflicts / concurrency issues



# Data Services Approach



# Data Services Approach



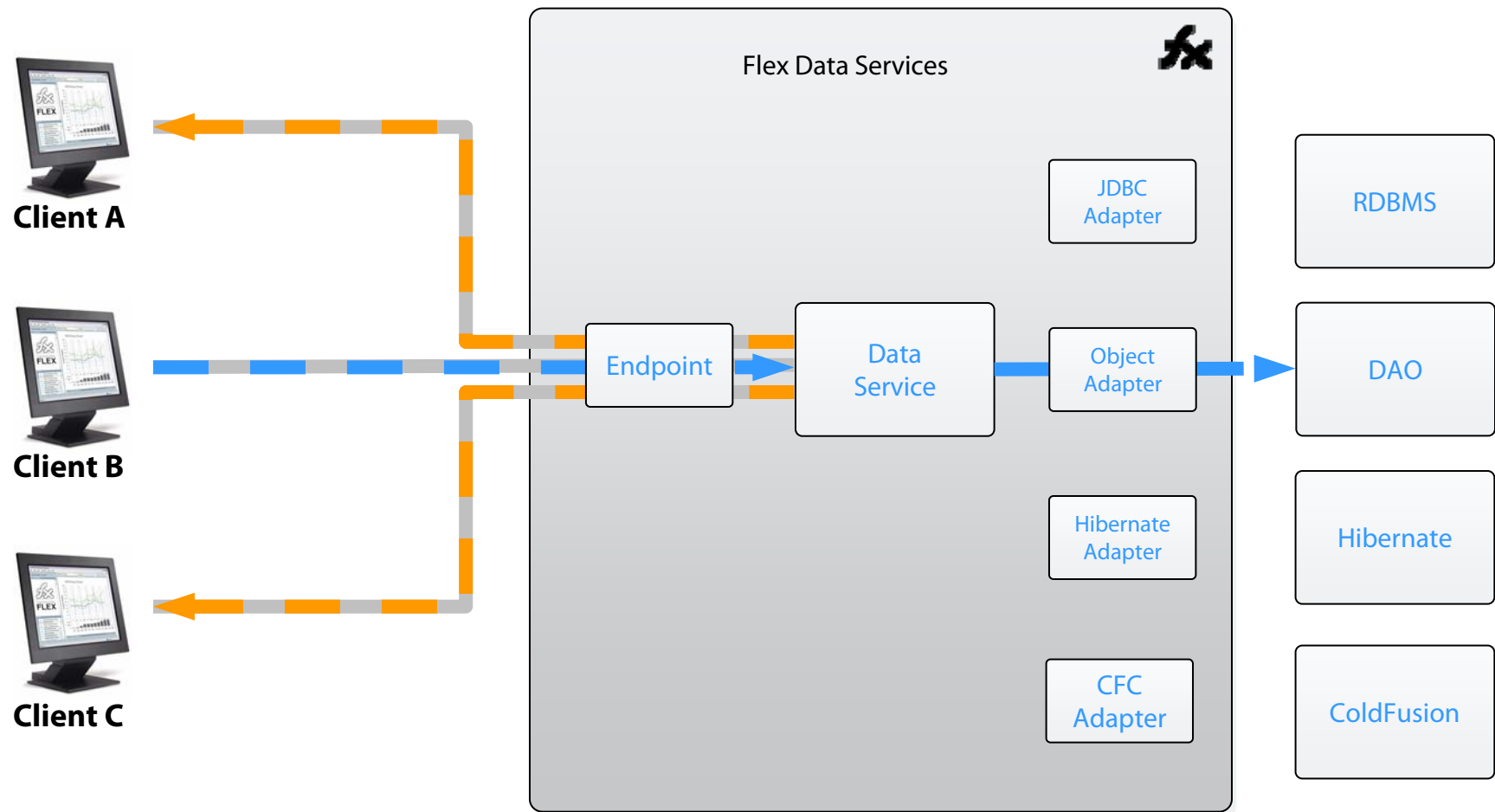
## Data Service:

- Deserialize changes coming from client
- Serializes data before sending to client
- Passes list of changes to configured adapter
- Notifies clients who subscribed to destination

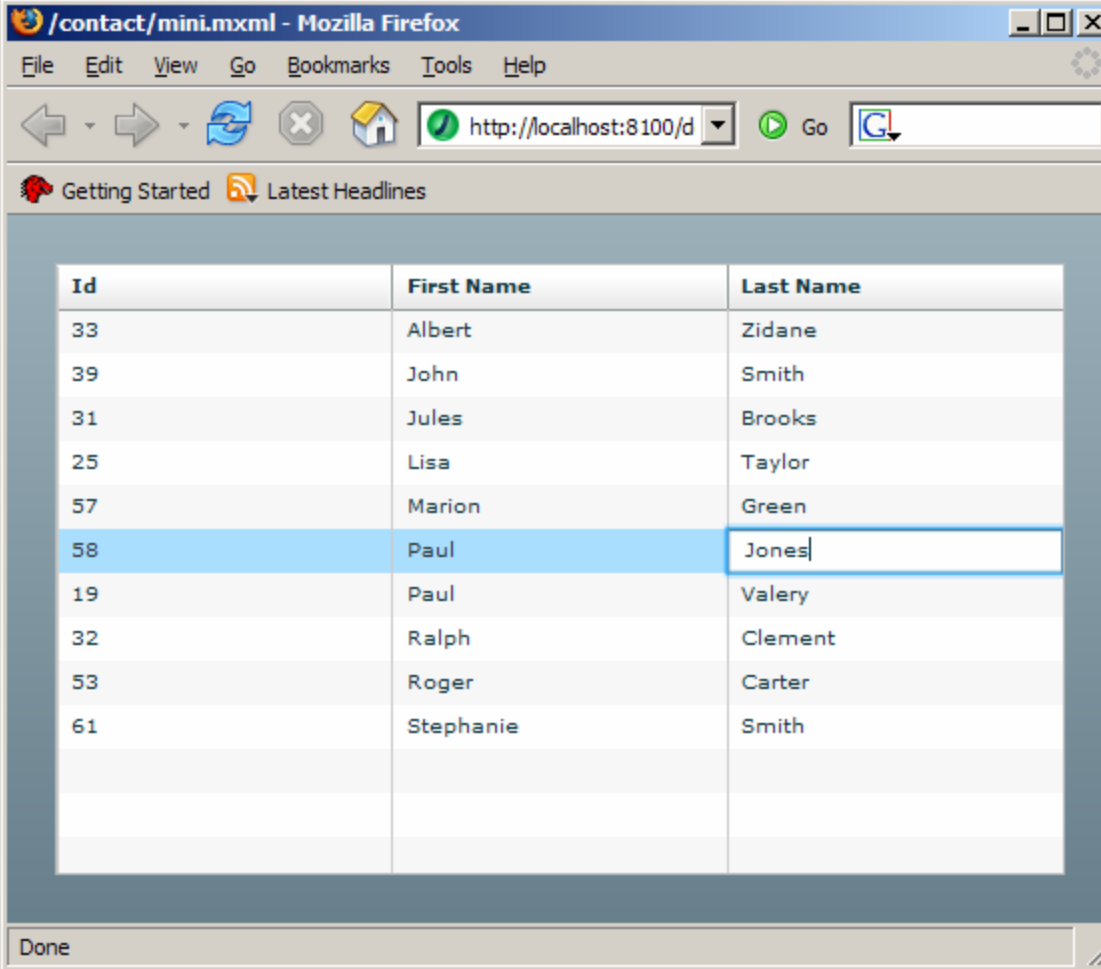
## Adapter:

- Processes changes
- Identifies conflicts
- Passes results to Data Service

# Data Services



# Demo: A Simple Data Services Application



The screenshot shows a Mozilla Firefox browser window with the title bar "/contact/mini.mxml - Mozilla Firefox". The address bar contains "http://localhost:8100/d". The page displays a table with three columns: "Id", "First Name", and "Last Name". The table contains 12 rows of data. The row with "Id" 58 is highlighted in blue, and the "Last Name" cell for this row, "Jones", is selected with a text cursor. The status bar at the bottom shows "Done".

Id	First Name	Last Name
33	Albert	Zidane
39	John	Smith
31	Jules	Brooks
25	Lisa	Taylor
57	Marion	Green
58	Paul	Jones
19	Paul	Valery
32	Ralph	Clement
53	Roger	Carter
61	Stephanie	Smith

# Demo: A Simple Data Services Application

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.macromedia.com/2005/mxml" creationComplete="initApp()">
<mx:Script>
    public var ds:DataService;
    [Bindable]
    public var contacts:ArrayCollection;
    public function initApp() {
        contacts = new ArrayCollection();
        ds = new DataService("contact");
        ds.fill(contacts);
    }
</mx:Script>
<mx:DataGrid dataProvider="{contacts}" editable="true"/>
</mx:Application>
```

# Demo: Contact Application

**Contact List**

Id	First Name	Last Name
33	Albert	Zidane
39	John	Smith
31	Jules	Brooks
25	Lisa	Taylor
57	Marion	Green
58	Paul	Jones
19	Paul	Valery
32	Ralph	Clement
53	Roger	Carter
61	Stephanie	Smith

**Stephanie Smith**

First Name \*

Last Name \*

Address

City

State

Zip

Phones

Number	Description

Done

# flex-services.xml

```
<destination id="contact">
  <adapter ref="java-dao" />
  <properties>
    <metadata>
      <identity property="contactId"/>
    </metadata>
    <server>
      <assembler>
        <class>samples.contact.ContactAssembler</class>
      </assembler>
      <fill-method>
        <name>loadContacts</name>
      </fill-method>
      <sync-method>
        <name>syncContacts</name>
      </sync-method>
    </server>
  </properties>
</destination>
```

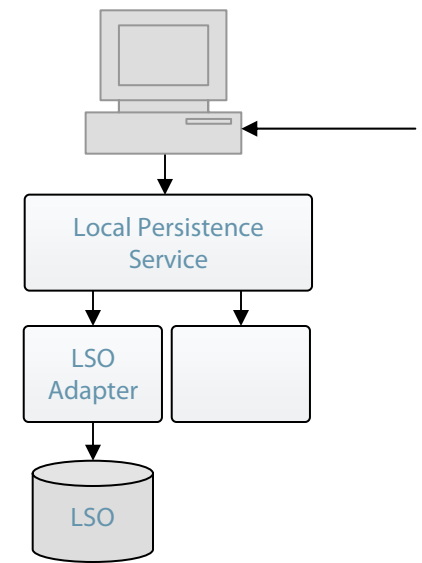
# ContactAssembler.java

```
public class ContactAssembler {  
    public List loadContacts() {  
        ContactDAO dao = new ContactDAO();  
        return dao.getContacts();  
    }  
    public List syncContacts(List changes) {  
        Iterator iterator = changes.iterator();  
        ChangeObject co;  
        while (iterator.hasNext()) {  
            co = (ChangeObject) iterator.next();  
            if (co.isCreate()) {  
                // invoke logic to create item  
            } else if (co.isUpdate()) {  
                // invoke logic to update item  
            } else if (co.isDelete()) {  
                // invoke logic to delete item  
            }  
        }  
        return changes;  
    }  
}
```



# Local Persistence Service

- Allows offline data access
- Provides common API for creating, reading, updating, and deleting local data
- Adapters are used for different types of local data storage devices
- Local Shared Objects adapter is available in Flex 2.0
- Offline data synchronization supported through combination of local persistence and messaging services



# Paging

- Maximum message size can be specified on destination
- If size exceeds maximum value, multiple message batches are used
- Client reassembles separate messages
- Enables asynchronous data paging across the network
- User interface elements can display portions of the collection without waiting for the entire collection to load

# Summary

- Flex Messaging enables new categories of applications including push and collaboration
- Data Services simplify the development and maintenance of data driven application

**Thank you.**