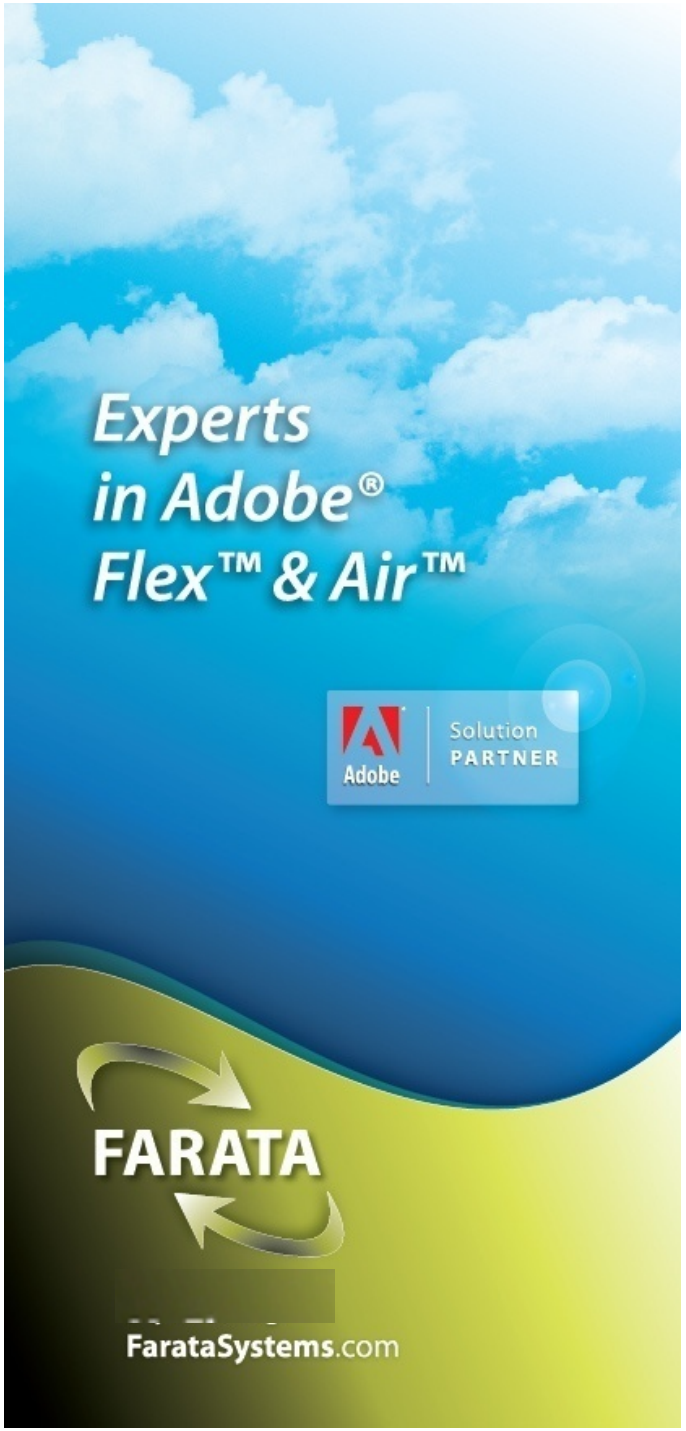
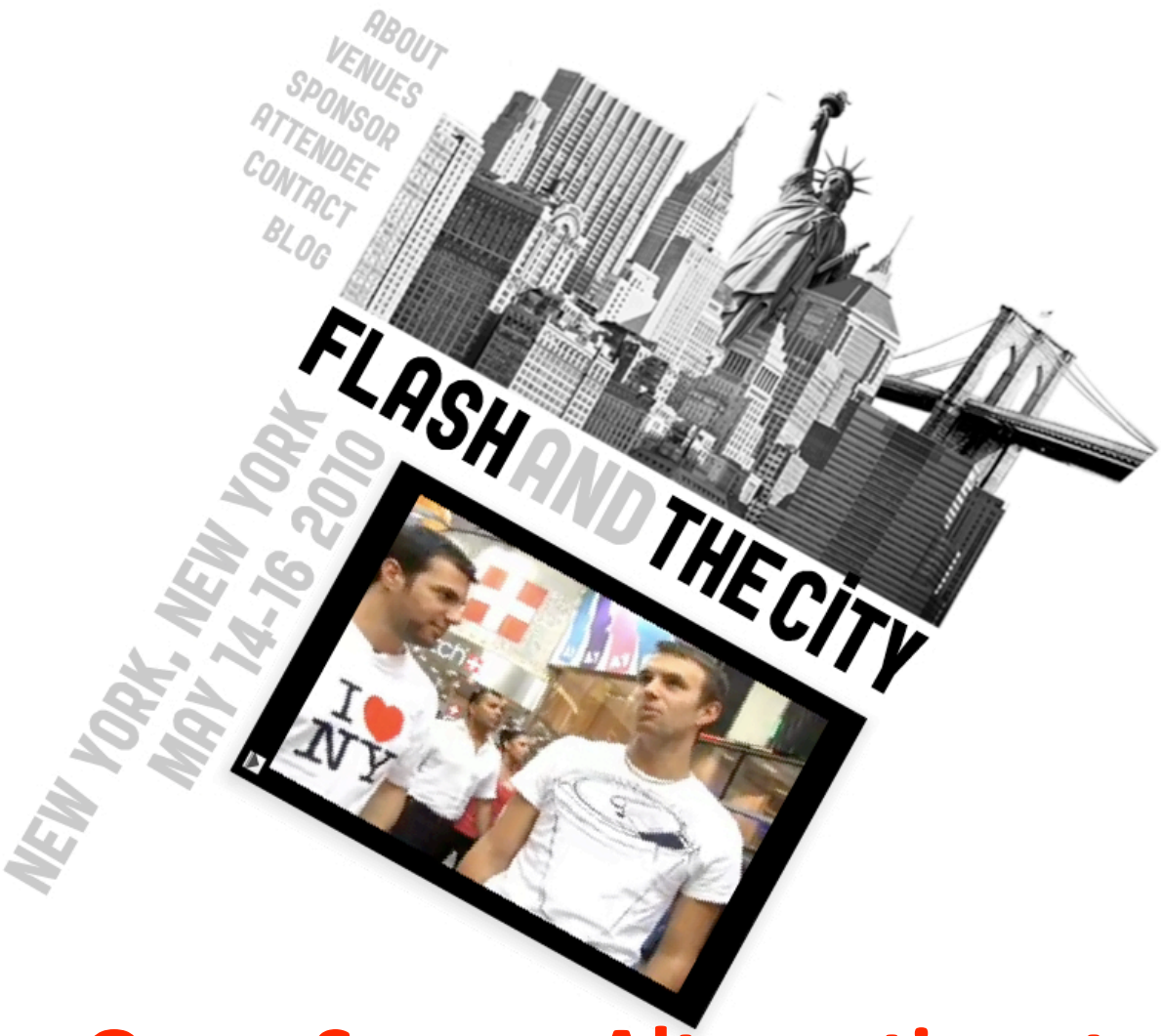
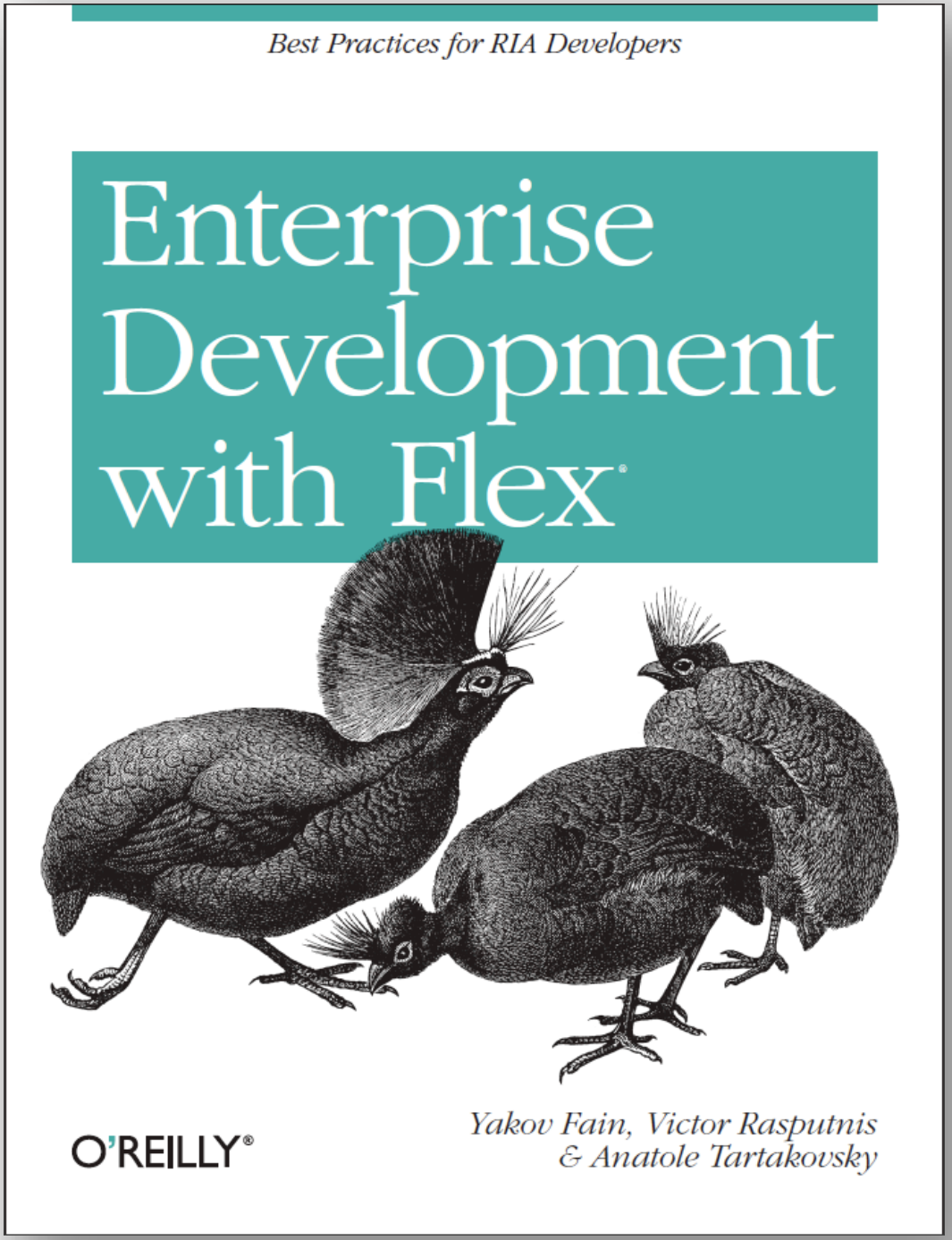




Open Source Alternatives to LCDS: Clear Toolkit

by Yakov Fain a.k.a. @yfain



Farata Systems is always happy to share all technical knowledge we're blessed with.

For a lousy $34.95 you can get this Amazon bestseller as soon as you get online.

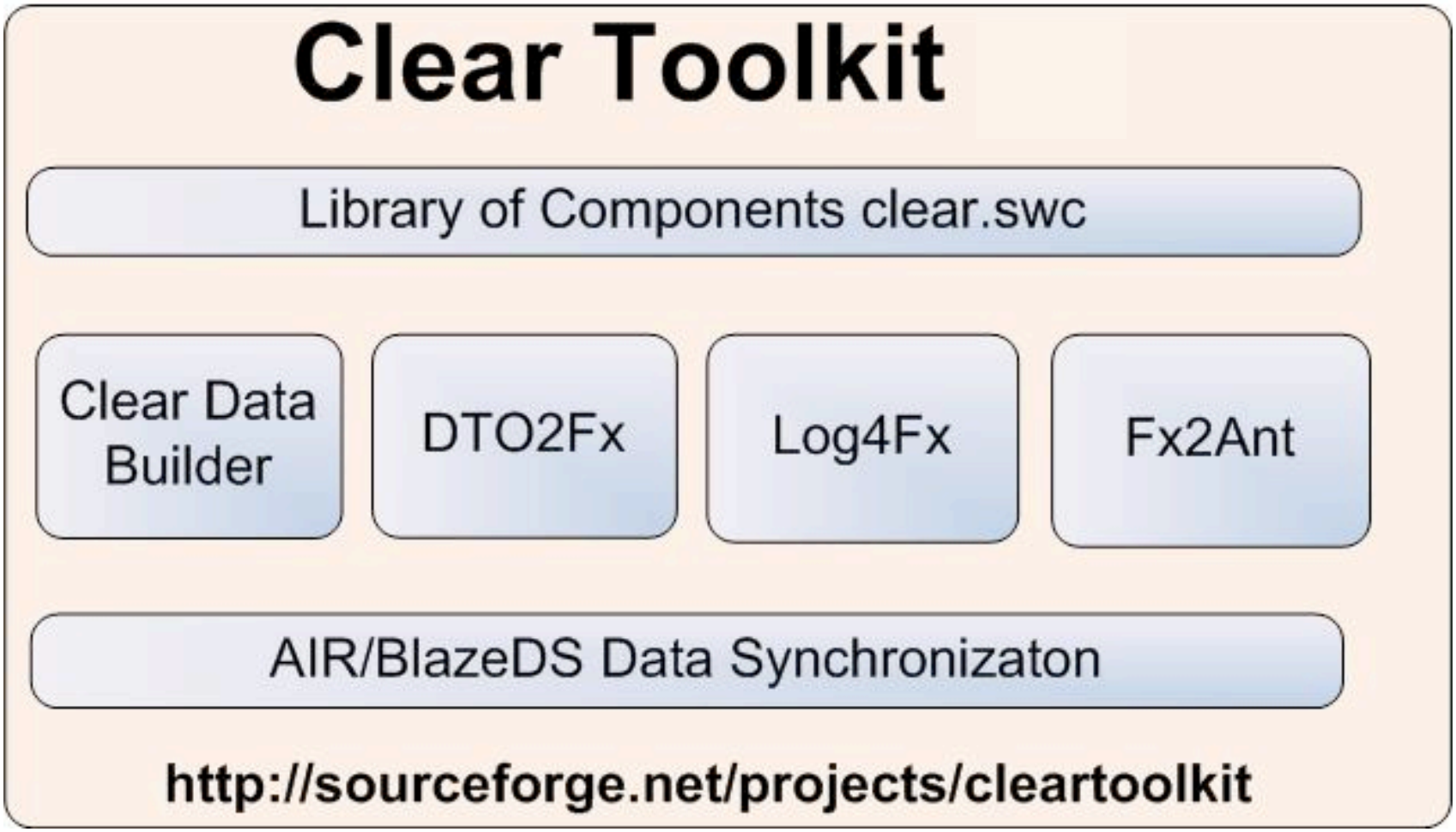
Three of you will get it for free today.

In this preso everything is either already open sourced or getting there

1. Brief overview of Clear Toolkit 3.2 components
2. Data Synchronization with DataCollection and ChangeObject
3. CRUD generation with Clear Data Builder
4. Ant generation with Fx2Ant
5. ActionScript DTO generation with DTO2FX
6. New DataForm component for Flex 4
7. Reliable messaging with BlazeDS

Feature Matrix: Adobe Live Cycle Data Services ES2, BlazeDS 3, Clear Toolkit 3+

Feature	BlazeDS 3	BlazeDS+Clear Toolkit	LCDS ES2
RPC services			
Java remoting/AMF	X	X	X
AJAX to Java	X	X	X
WS/JSON proxy	X	X	X
Messaging			
Servlet-based messaging (hundreds of clients per CPU)	X	X	X
Servlet-based NIO messaging (thousands of clients per CPU)		X*	
Java NIO high-performance messaging (thousands of clients per CPU)			X
Real Time Messaging Protocol (RTMP)			X
Data throttling		v4	X
Reliable communications		v4	X
Data Management			
Transaction (batch processing)		X	X
Data paging		v4	X
Lazy loading (on demand)		v4	X
Hierarchical data collections		X	X
Conflict resolution and synchronization		X	X
SQL adapter		X	X
Hibernate adapter		v4	X
Fiber-aware assembler		v4**	X
Offline Web applications			X
Development and deployment			
Spring integration support	X	v4	X
Adobe Flash® Builder™ modeling plug-in			X
Enterprise support		X	X
RIA PDF generation		X***	X
WSRP portal integration			X
Load/stress testing tool			X
Source code available	X	X	
Edge server			X
Enterprise support plans		X	X
Productivity Tools			
Generator of CRUD application		X	X
Generator of ActionScript data transfer object based on their Java peers		X	X
Generator of ANT script based on the properties of Flash Builder project		X	
Automated Data synchronization of AIR locale cache		X	X
Flash-based Web reporter		v4	
Cost of production deployment			
License type and cost	LGPL v3, Free	MIT, Free	Commercial, about $30K per CPU****



It’s an open source toolkit , that includes a set of Flex Builder plugins and extended Flex components.

Clear Toolkit increases productivity of the enterprise Flex developers.

Clear Toolkit 3.2.1 includes:

Clear.swc library includes a number of enhanced Flex components like Datagrid, ComboBox et al. Supports PDF generation on the client.

Clear Data Builder 3.2.1 is an Eclipse plugin that allows to generate CRUD applications for BlazeDS based on a Java Data Transfer Objects or an SQL statement.

DTO2Fx - automatic generation of ActionScript classes from their Java peers.

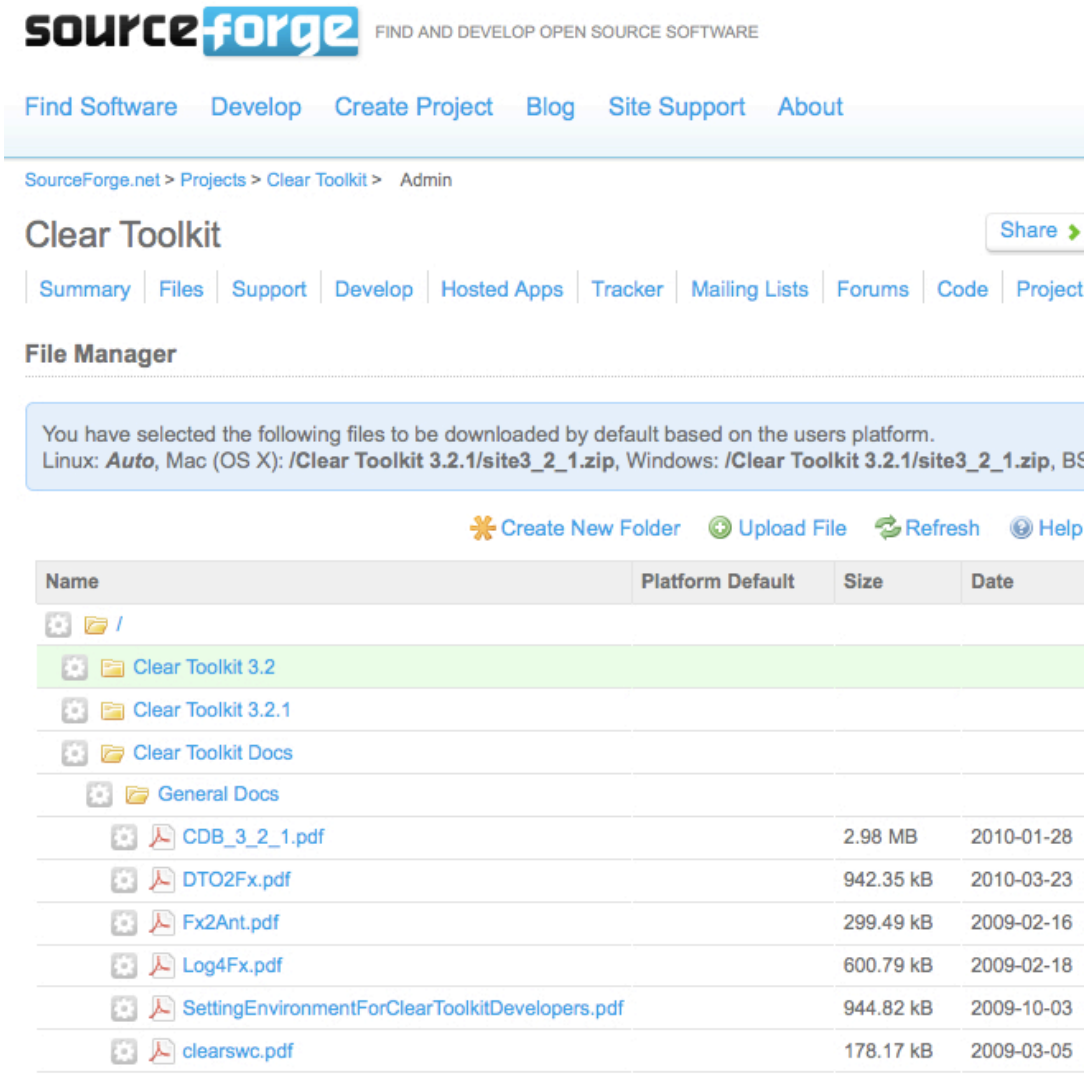
Log4Fx is an advanced logger (Eclipse plugin) that is built on top of Flex logging API. It automates and make the logging process more flexible and user friendly.

Fx2Ant is a generator of optimized ANT build scripts for your Flex Builder projects.

AIR/BlazeDS synchronization solution for occasionally connected applications

PDF generation on the client (described in detail in Chapter 11 of the book)

I like free open source solutions, but I'm sick and tired of working with not documented software. Is there anything to read about your great Clear Toolkit thingy?



New additions in Clear Toolkit 4 (Beta):

DataForm: this Flex 4 compatible component features:

New form layouts: VerticalFormLayout and HorizontalFormLayout

Support of DTO-based dataProvider

Form and FormItem level vaidations

Reliable Messaging with BlazeDS

- Identifying lost packages on the AMF protocol level
- Automatic resends of lost packages
- Dealing with out-of-sequence messages

Data Synchronization with BlazeDS/ using DataCollection

ChangeObject – The Cornerstone of Data Management

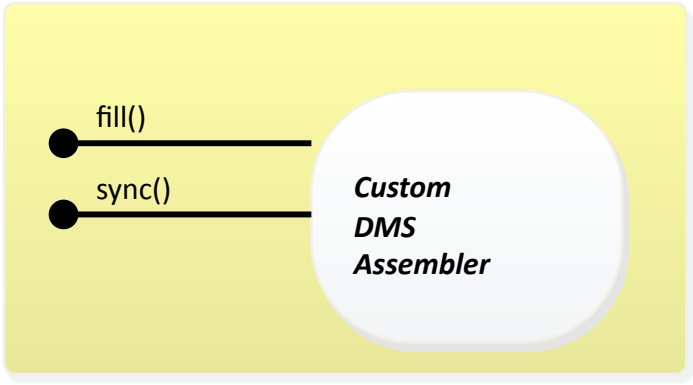
- http://livedocs.adobe.com/livecycle/es/sdkHelp/programmer/lcdsjavadoc/flex/data/ChangeObject.html:
  - getNewVersion(), getPrevVersion()
  - isUpdate(), isCreate(), isDelete()
  - getChangedPropertyNames()
  - setNewVersion()

- All changes are communicated as list of ChangeObjects:
  - Changes from client to server
  - Changes from server to client
  - Response to changes sent by this client (echo)
  - Proliferation of changes sent by other clients (push)

- Only original data set is sent as complete list of records – once!

The Java side of Adobe Data Management Service

- Application may call *commit()* method to ship all changes in one shot.
- Default – autocommit.
- Custom Java objects backing Data Management Services usually provide
  - a method returning the collection, aka *fill* method
  - a method processing the list of changes sent from the client, aka *sync* method



Clear Toolkit Data Management

- Clear Toolkit https://sourceforge.net/projects/cleartoolkit/ supports data synchronization over RemoteObject and Consumer (BlazeDS is enough)
- Supports data pull and server push (changes)
- Enables client-side managed transactions:
  - transactional commit from several array collections
  - transactional batching of any number of remoting operations
  - supports hierarchical nesting of collections
- Major objects:
  - DataCollection
  - BatchService

Clear Toolkit DataCollection Solution

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
xmlns:fx="http://www.faratasystems.com/2008/components"
layout="vertical" creationComplete="sandwiches.fill()">
<fx:DataCollection id="sandwiches" destination="sandwichDAO"
method="getSandwiches" />
<mx:DataGrid dataProvider="{sandwiches}" editable="true">
    <mx:columns>
        <mx:DataGridColumn dataField="sandwichName"
        headerText="Name"/>
        <mx:DataGridColumn dataField="bread" headerText="Bread"/
        >
        <mx:DataGridColumn dataField="meat" headerText="Meat"/>
        <mx:DataGridColumn dataField="spread"
        headerText="Spread"/>
    </mx:columns>
</mx:DataGrid>
</mx:Application>
```

What's DataCollection?

- *com.farata.collections.DataCollection* – part of the clear.swc component library.

- DataCollection is an extension of standard ArrayCollection

- Features:
  - **destination aware** - it knows where to *fill* itself from (on the Java Remoting side)
  - **change tracking** - it records all interactive and programmatic changes
  - **sync-able** – it knows the Java Remoting method to submit all accumulated changes to the server
  - **push-ready** – has internal Consumer object to consume server-pushed changes, if any

DataCollection-inspired derivatives

- *com.farata.remoting.BatchService* – allows batching of changes to multiple DataCollections to be sent to the server for atomic processing
- *CDB plugin:*

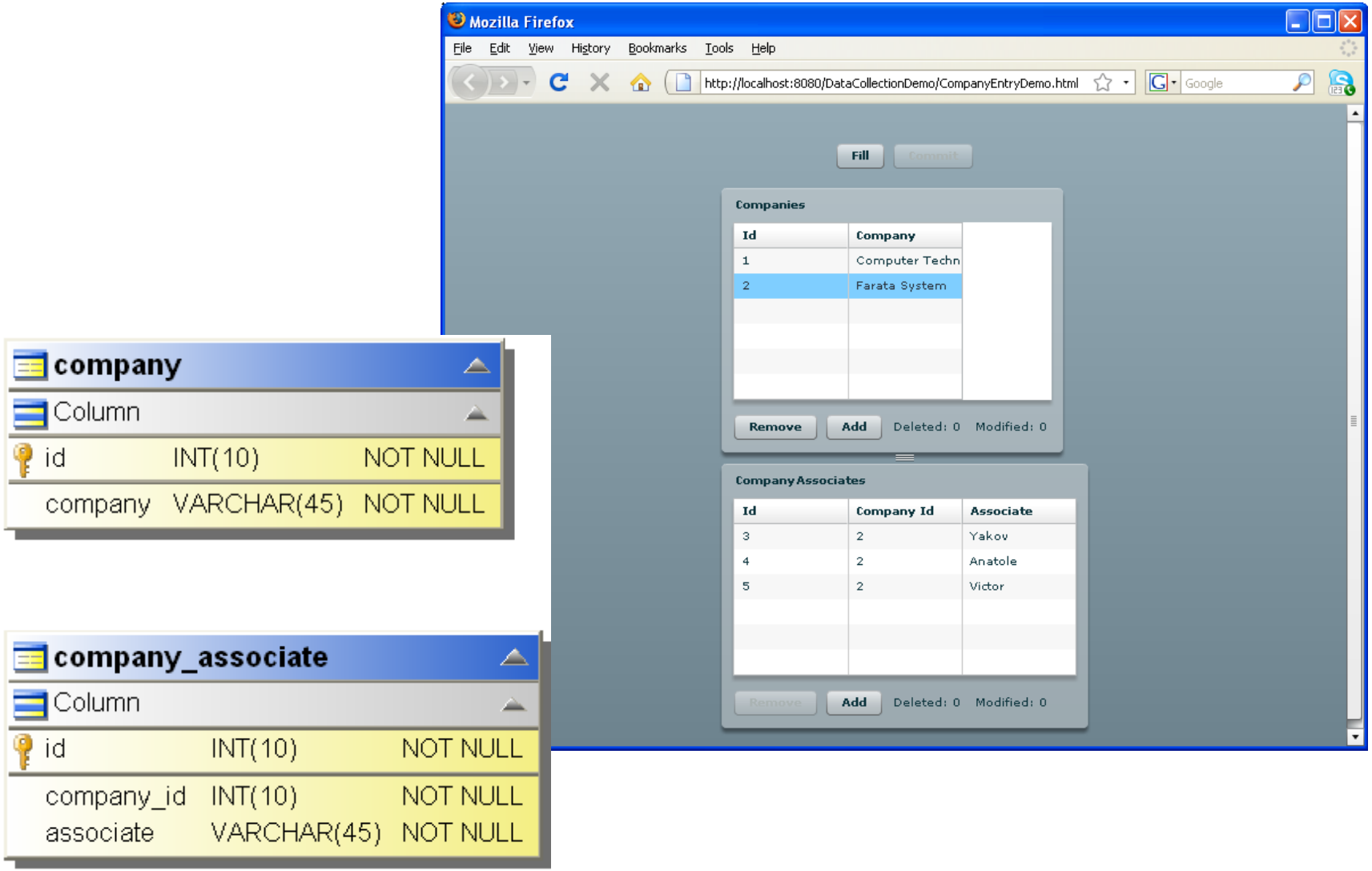
  CRUD code generation for Java and Flex with two starting points:

  a) SQL – all Java will be code generated for you

or

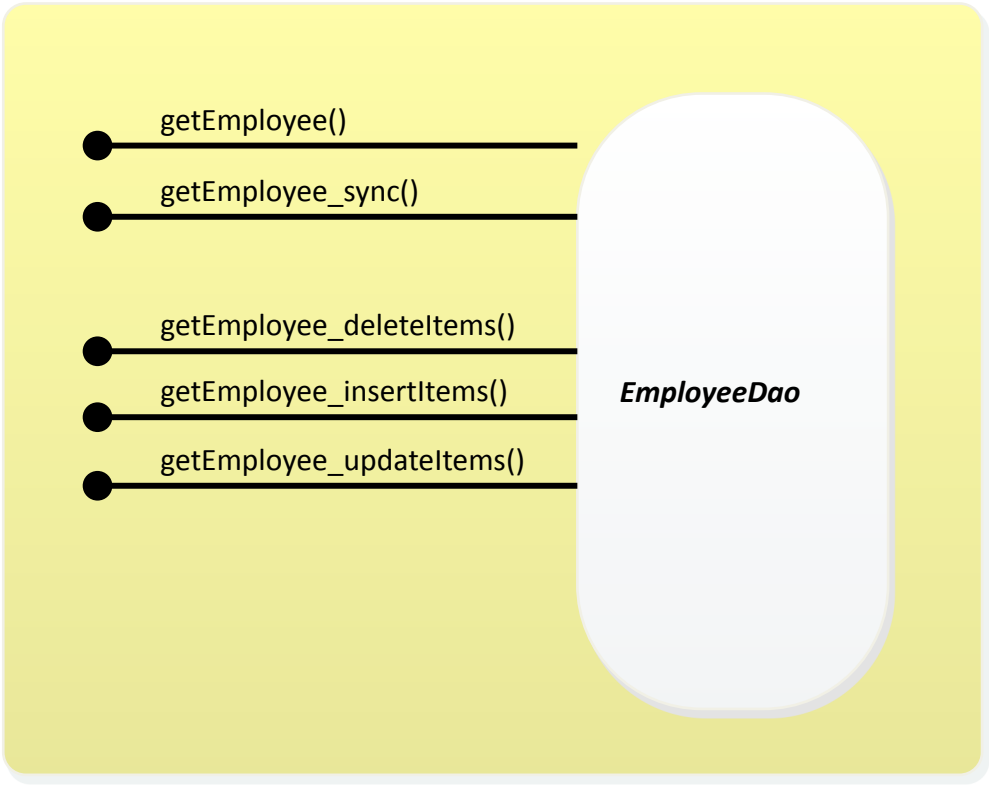
  b) Java DAO and DTO

DataCollection Use Case



How to Program a Client-Managed Transaction

- Have a *fill* method (getEmployee) returning a collection of DTOs
- Have _*sync*() method for simple synchronization;
- Have _deleteItems()/_updateItems()/_insertItems() for batching of changes in a master-detail scenario



BatchService / Batch Gateway Tandem

- *com.farata.remoting.BatchService* is a part of Farata clear.swc
  - Allows to group together changes accumulated by multiple DataCollections:
    batchService.registerCollection(orders, 0); //level 0 - master
    batchService.registerCollection(orderItems,1); //level 1 - details
  - Allows to send the batch of changes to be executed as single JTA transaction by the server:
    var batch:Array = batchService.batchRegisteredCollections();
    batchService.sendBatch(batch)
- *com.farata.remoting.BatchGateway* is a part of Farata daoflex-runtime.jar, pre-installed upon configuring a CDB project
  - Serves preconfigured destination that BatchService sends his batches to. Executes the batch sequentially (by default – as single JTA transaction)

Preparing a Batch on the Client

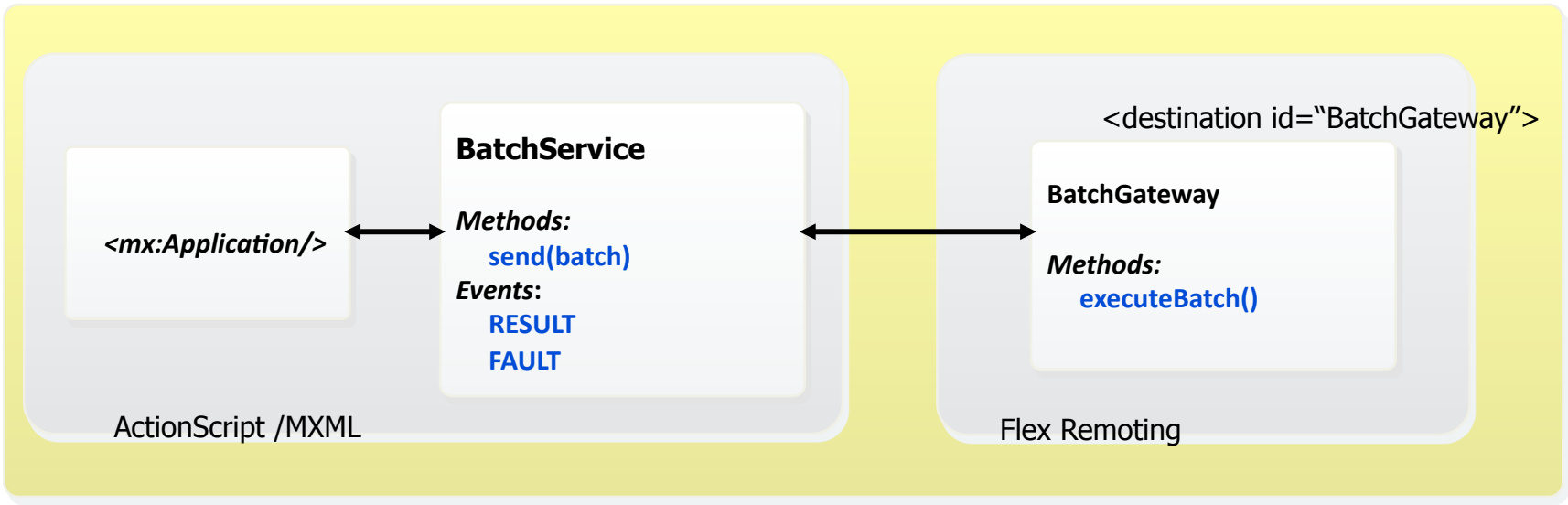
```
public class OrderController {
      public function OrderController(){
         orders = new DataCollection();
         orders.destination="Order";
         orders.method="getOrders";

         orderItems = new DataCollection();
         orderItems.destination="Order";
        orderItems.method="getOrderItems";
         . . . .
        }
        public function fillOrders():void {
         orders.fill();
        }
        public function fillOrderItems(orderId:String):void {
         orderItems.fill(orderId);
        }
}
```

Sending a Batch to the Server

```
public class OrderController {
    public function OrderController(){
            . . . .
      batchService = new BatchService();
      batchService.registerCollection(orders, 0); //master
      batchService.registerCollection(orderItems,1); //details, orders
    }
    public function commit():void {
      var batch:Array =    batchService.batchRegisteredCollections();
      batchService.sendBatch(batch); // transaction, by default
    }
}
```

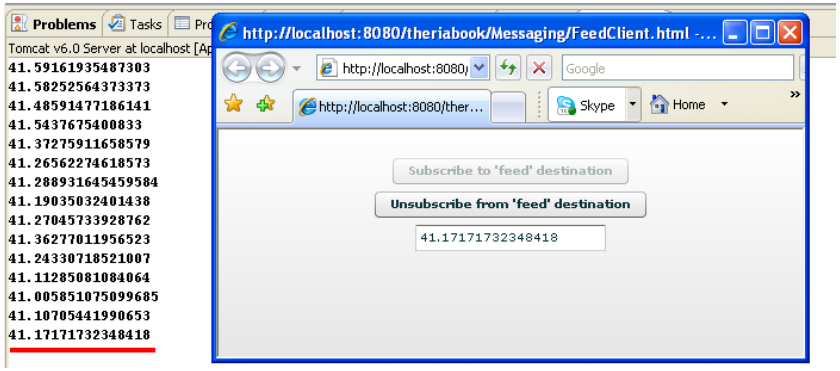
Executing a Batch on the Server

- Pre-built class *com.farata.remoting.BatchGateway*
- Allows to call multiple remote Java methods in a sequence according to the list of *BatchMember*'s prepared in ActionScript.
- Allows to wraps JTA transaction around this sequence
- BatchMember: ("destination", "method", [arguments])
- Use cases: batch updates, batch retrieves



Unraveling AutoSync

Java Server Push to a Flex messaging destination:




```
MessageBroker msgBroker = MessageBroker.getMessageBroker(null);
AsyncMessage msg = new AsyncMessage();
msg.setDestination("feed"); // target destination
msg.setClientId(UUIDUtils.createUUID(false));
msg.setMessageId(UUIDUtils.createUUID(false));
msg.setTimestamp(System.currentTimeMillis());
msg.setBody(“Pushed Message" ); // arbitrary message
msgBroker.routeMessageToService(msg, null);
```

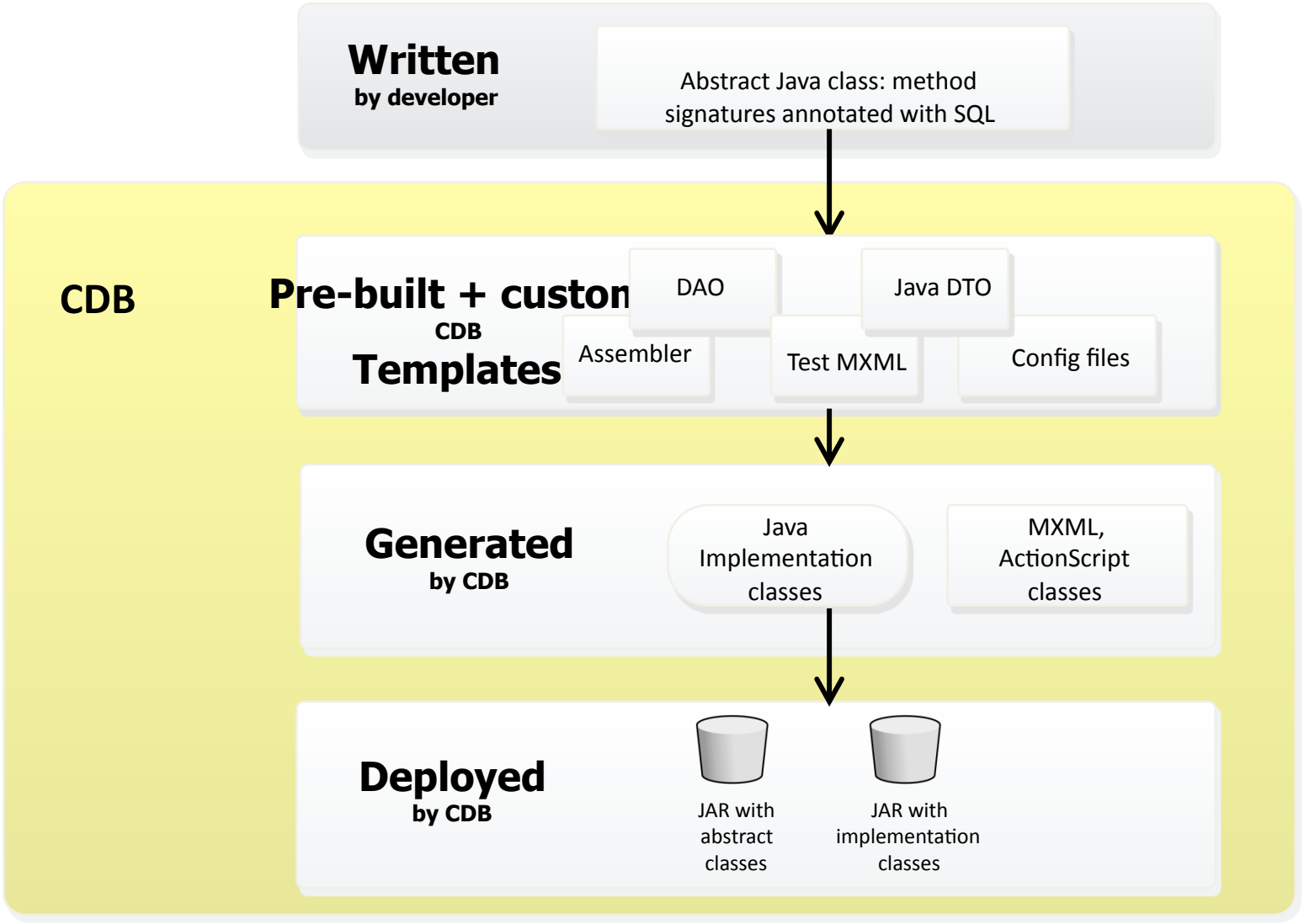
Clear Toolkit 3.2 includes CDB

- Clear Data Builder (CDB) :

http://sourceforge.net/projects/cleartoolkit
(Downloads, documentation, license)

- To get the source code, configure Eclipse CVS :
  - Host: cleartoolkit.cvs.sourceforge.net
  - Repository: /cvsroot/cleartoolkit
  - Protocol: pserver
  - User: anonymous
- To get the Flex 4 compatible version of CDB, get the nightly build at Sourceforge: http://farata.dynalias.org:9080/cruisecontrol/artifacts/ClearToolkit.nightly/

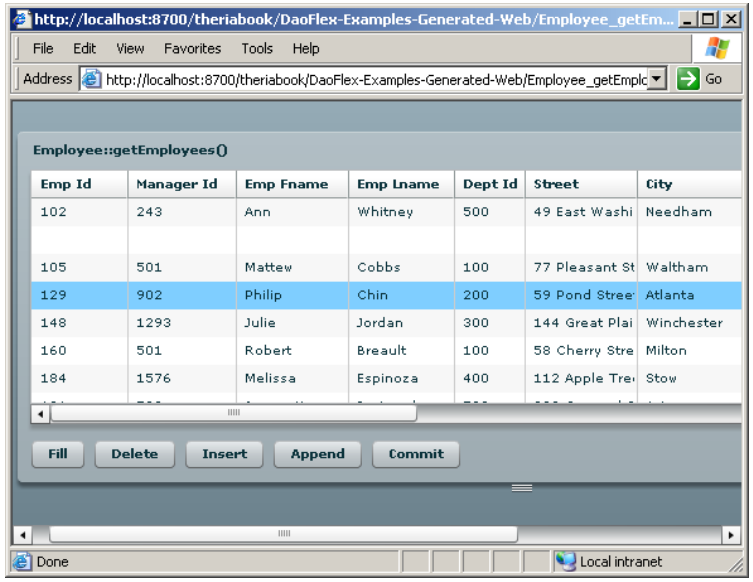
CDB SQL Branch: Complete Automation



CDB SQL Mode

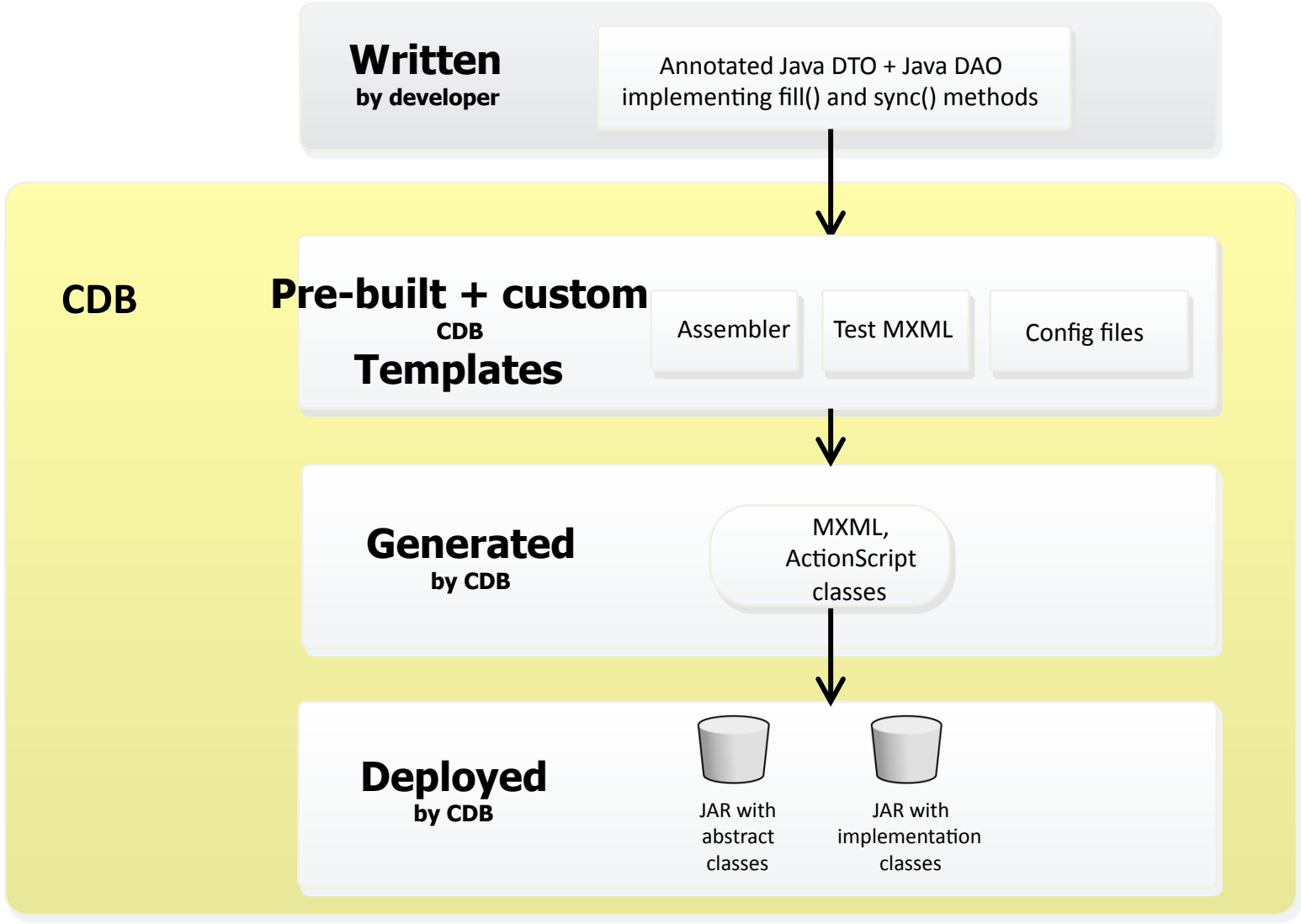
```
package com.theriabook.datasource;
/**
 * @daoflex:webservice
 *   pool=jdbc/theriabook
 */
public abstract class Employee {
/**
* @daoflex:sql
*  sql=  select * from employee where start_date < :startDate
*  transferType=EmployeeDTO[]
*  keyColumns=emp_id
*/
public abstract List getEmployees(Date startDate);
}
```

>



- Code generation solution that completely automates manual Java coding for CRUD (create, retrieve, update, delete) tasks
- Integrates with Eclipse or works standalone as Ant task
- Goes all the way from simple annotated Java class to deployed JARs in the WEB-INF/lib
- Generates all required Java, ActionScript, MXML, XML

CDB Java Mode: Partial Automation



Demo of Clear Data Builder

```
package com.farata.datasource;
import java.util.List;
/**
 * @daoflex:webservice
 *   pool=jdbc/test
 */
public abstract class Employee{
                /**
                * @daoflex:sql
                *  pool=jdbc/test
                *  sql=::  select * from employee
                *  ::
                *  transferType=EmployeeDTO[]
                *  keyColumns=emp_id
                *  updateTable=employee
                */
                public abstract List getEmployees();

                /**
                * @daoflex:sql
                *  sql=:: select * from department
                *  ::
                *  transferType=DepartmentDTO[]
                */
                public abstract List getDepartments();
}
```

Farata DTO2Fx Plugin

http://sourceforge.net/projects/cleartoolkit/files/ read the doc: DTO2Fx.pdf

```
package com.farata.test;
import com.farata.dto2fx.annotations.FXClass;
import com.farata.dto2fx.annotations.FXIgnore;

@FXClass
public class EmployeeDTO {
        public firstName;
        private String sex;
        . . .
        private String uid;

        public long getSex() {
          return sex;
        }
        public void setSex(String sex) {
          this.sex = sex;
        }

        @FXIgnore
        public String getUid() {
          return uid;
        }
        @FXIgnore
        public void setUid(String uid) {
          this.uid = uid;
        }
}
```

>

```
package com.farata.test {
public class _EmployeeDTO extends flash.events.EventDispatcher
        implements mx.core.IPropertyChangeNotifier,
        mx.core.IUID {

@FXClass
public class EmployeeDTO {
        private  var _firstName:String;
        private var _sex:String;
        . . .

        private var uid:String;
 [Bindable(event="propertyChange")]
  public function get firstName():String {
   return _firstName;
  }
  public function set firstName(value:String):void {
   const oldValue:String = this._firstName;
   if (oldValue != value) {
     this._firstName= value;
     dispatchUpdateEvent(“firstName", oldValue, value);
   }
  }
 [Bindable(event="propertyChange")]
  public function get sex():String {
   return _sex;
  }
  public function set sex(value:String):void {
   const oldValue:String = this._sex;
   if (oldValue != value) {
     this._sex = value;
     dispatchUpdateEvent("sex", oldValue, value);
   }
  }
```

Creating ANT scripts with FX2Ant Plugin

http://sourceforge.net/projects/cleartoolkit/files/ read the doc: Fx2Ant.pdf

A modularized Flex application consists of several Flash Builder projects. Right-click at the Flex project and FX2Ant generates the ANT script.

Each of the individual projects should contain a file build.xml to perform build and deployment of this project.

One extra build file is required to run individual projects' builds in an appropriate order and to deploy the entire application in some predefined dir, i.e. c:\serverroot

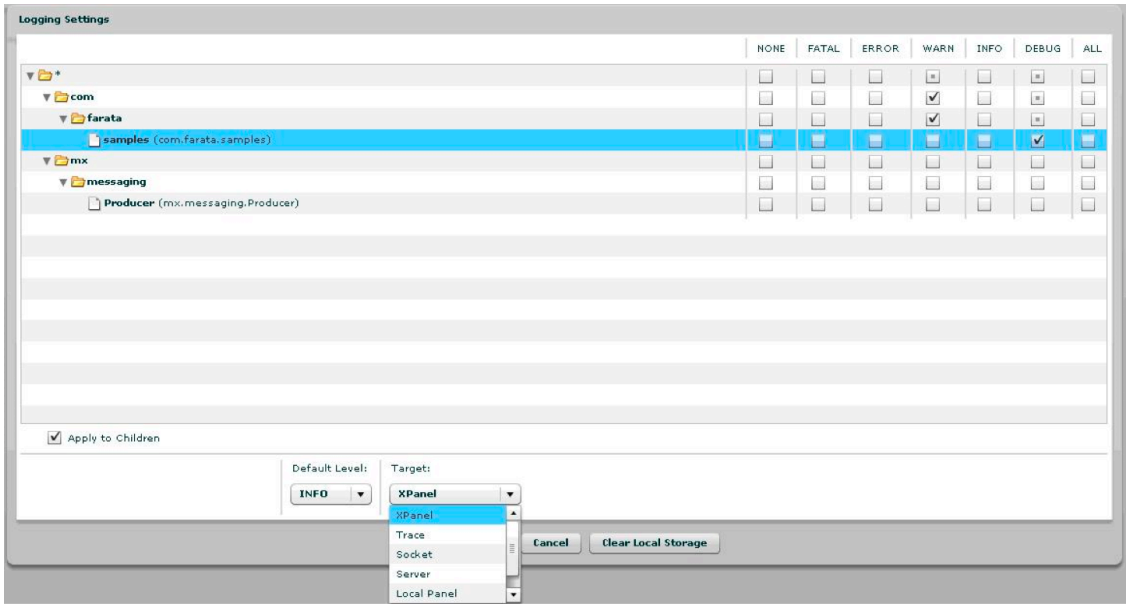
Logging with Log4Fx

```
import mx.logging.Log;
import mx.logging.ILogger;
private var logger:ILogger = Log.getLogger(”MyStockPortfolio”);
```

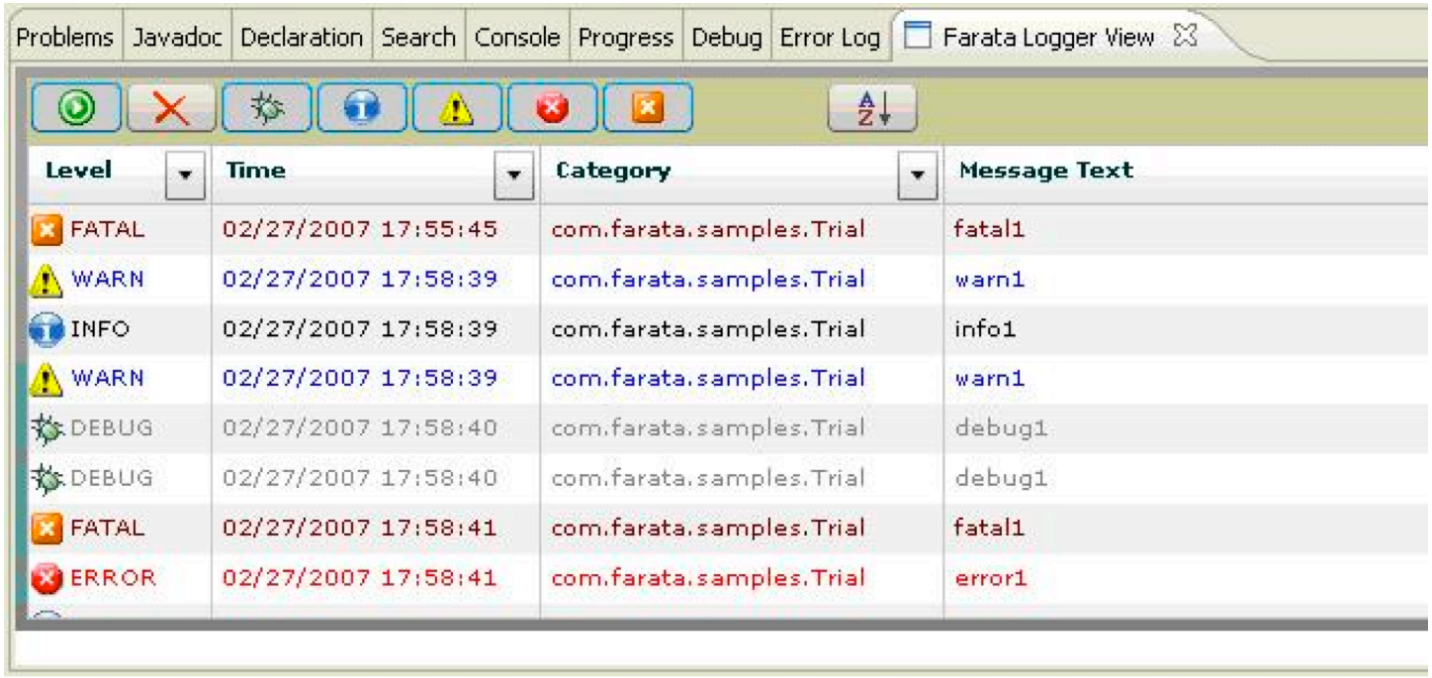
Say, you are considering adding this trace statement in the function getPriceQuotes():

```
if (Log.isDebug()) logger.debug(”Getting price quote for IBM");
```

This message will be sent to the specified destination selected with the Logging Manager.
If a user calls production support complaining about some unexpected behavior,
ask her to press *Ctrl+Shift+Backspace* – the logging manager will pop-up:



Logging with Log4Fx – sample output



You can set the output destination as RemoteLogging to watch the log info from any computer connected to the Internet

Clear Toolkit 4 additions

- Reliable BlazeDS Messaging
- New spark-based components, i.e. DataForm

Clear Toolkit 4: DataForm

- Introduces **VerticalFormLayout** and **HorizontalFormLayout** for auto width adjustment of each **DataFormItem** element and its label.
- Multiline labels are supported
- Supports Data Transfer Object as **dataProvider**
- Features advanced form validation: **validateForm()** and **resetValidation()**
- Offers a form-level **isValid** property..

Demo of DataForm

Unreliable Messaging

- You are developing and testing Flex/BlazeDS in a LAN environment.
- It should work in WAN too

Yeah, right! What if there's a thunderstorm in the area? Will the client receive all network packages?

What if the Ashhole volcano erupts again?

Unreliable Messaging and Common WAN Problems

Problem	Probability (typical for US consumer market)
Lost packages going to server	High (<.2%)
Lost packages going to client	Medium (<.02%)
Messages out of order	Low
Messages duplicated	Low
Message error (corrupted bits)	Very low

Testing on a local / reliable network does not exhibit any WAN problems

Network Problems Simulation

- Software only – Shunra Desktop
- Dedicated hardware appliance – Maxwell mini
- A Custom environment – usually a Linux box with multiple network ports configured as bridge using netem (a Network Emulation functionality for emulating the properties of WAN)

Clear Toolkit 4 will include the test client allowing to emulate problems (will demo today)

How we implemented Reliable Messaging

- We are implementing reliable messaging on the AMF protocol level
- The messages are being held both on client and server till they're confirmed
- Messages are sequentially numbered to process out-of-order messages
- Standard BlazeDS channel/endpoint are subclassed with **ReliableAMFChannel** and **ReliableAMFEndpoint**

Demo of Reliable Messaging

Contact info and useful links

Clear Toolkit Framework: http://sourceforge.net/projects/cleartoolkit/

O'Reilly Book "Enterprise Development with Flex": http://bit.ly/bDg7IR

Email: info@faratasystems.com

Web site: http://www.faratasystems.com

Flex Blog: http://flexblog.faratasystems.com