# San Diego Java Users Group

November 15, 2016

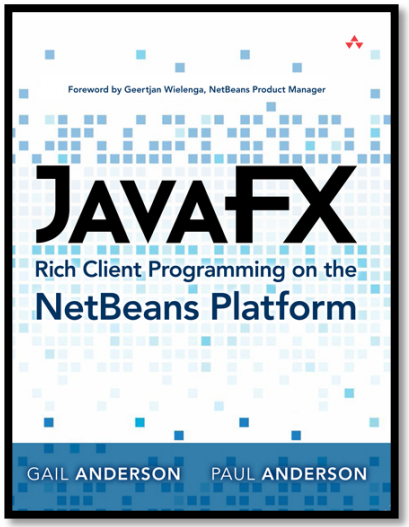
# Java on Mobile: Write Once, Run on IOS & Android

Paul Anderson
Gail Anderson
Anderson Software Group, Inc.
asgteach.com

# So Who Are We?

- Training Company
  - Java 8, JavaFX Courses
- JavaFX Authors
  - JavaFX Rich Client Programming on the NetBeans Platform
- LiveLesson Videos
  - JavaFX Programming
  - Java Reflection





# Agenda

- Why JavaFX for Mobile?
- Gluon Framework
- FXML and Gluon Charm with JavaFX
- Leveraging JavaFX
- Gluon Connect
- Gluon Cloud Authentication
- Afterburner Framework
- Target Platform Tips
- Wrap Up, Q & A

# Why JavaFX On Mobile?

- Critical Goal
  - Platform independent source code
  - “Write Once, Install Everywhere”
- Design Approach
  - Frameworks are a must
  - Hide platform dependencies and messy details
- JavaFX Advantages
  - Java UI, scene graph, nodes, FXML views
  - Properties, listeners, binding, event handlers

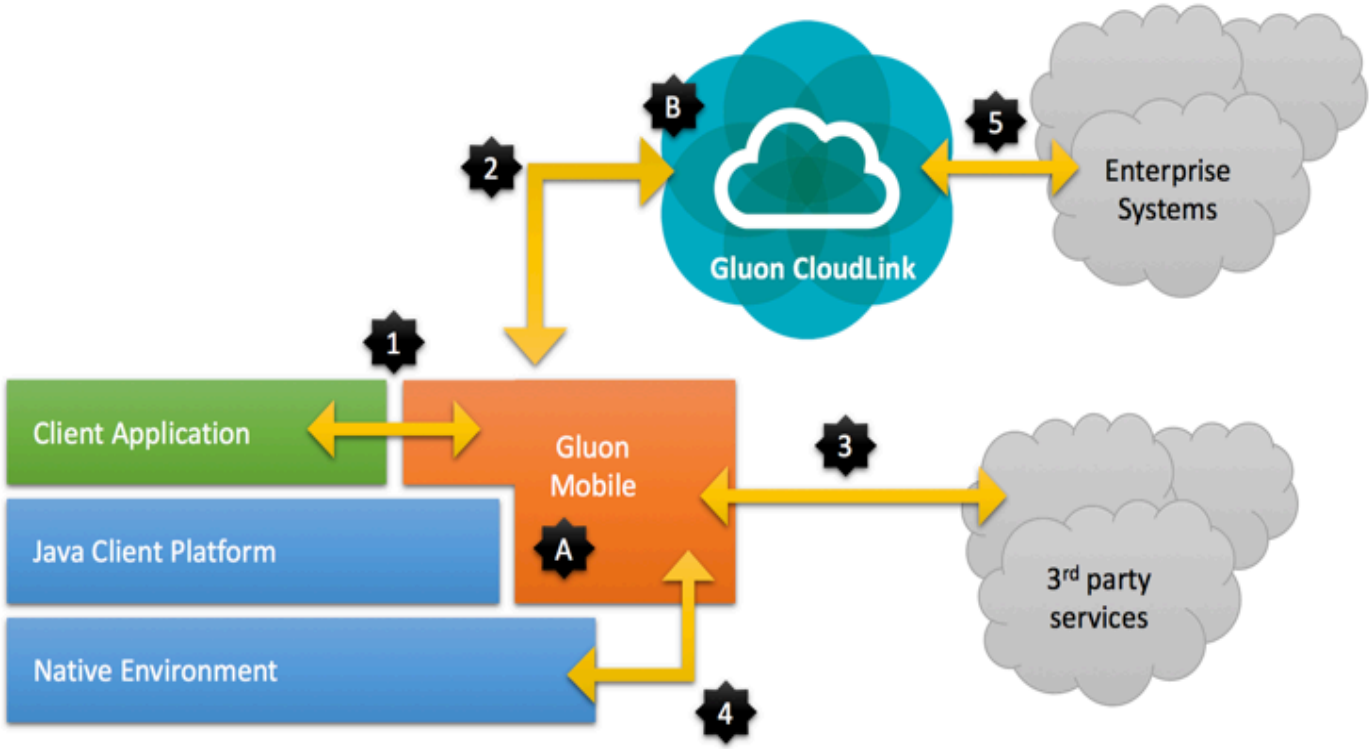
 

# Agenda

- Why JavaFX for Mobile?
- Gluon Framework
- FXML and Gluon Charm with JavaFX
- Leveraging JavaFX
- Gluon Connect
- Gluon Cloud Authentication
- Afterburner Framework
- Target Platform Tips
- Wrap Up, Q & A

# Gluon Framework



# Working with Gluon

- Development Tools
  - NetBeans IDE, Eclipse, IntelliJ IDEA
  - Gluon plug-in with bundled Gradle support
  - Create Gluon project, debug on desktop
  - Gluon licensing (optional)
- Supported Targets
  - IOS and Android
  - Embedded
  - Desktop

# Gluon with IOS

- Prerequisites
  - Mac with MacOS X 10.9+
  - Install and configure Xcode
  - Apple Provisioning
- Launch to IOS
  - Apache Harmony class libraries
  - RoboVM AOT compiler
  - Byte code to native IOS code
  - Deploy to connected IOS device

# Gluon with Android

- Installation
  - Android SDK Manager
  - Build-tools, SDK Platform, Support Library
  - Configure Gradle
- Launch to Android
  - Apache Harmony class libraries
  - Byte code optimized for Android target
  - Deploy to connected Android device
  - Dalvik runtime converts to native code

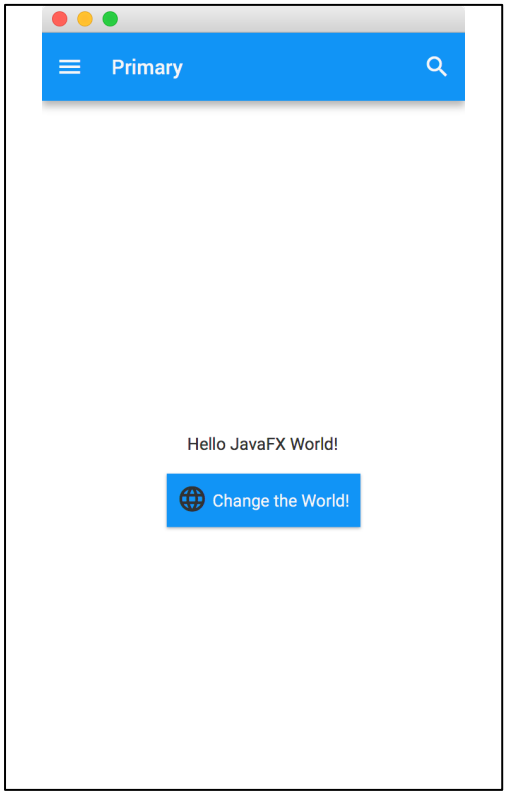
 

# Looking Ahead with Gluon

- Current Limitations
  - Mostly Java 7, lambdas but no streams
  - RoboVM
  - Apache Harmony libraries
- Future Releases
  - Gluon VM
  - OpenJDK 9 Mobile project
  - Replaces RoboVM and Apache Harmony
  - Allows most recent standard Java SDK

# Gluon NetBeans Project



# Gluon Mobile

- Development Tools
  - Scene Builder for layouts, CSS and skins
  - Gradle for builds, IDE independent
- Gluon Library
  - Charm Controls, Gluon Maps
  - Local and cloud storage
  - Material Design for mobile footprint
- Hardware Control
  - Camera, accelerometer, GPS, gestures, …

# Gluon Mobile:  First Look

# Agenda

- Why JavaFX for Mobile?
- Gluon Framework
- FXML and Gluon Charm with JavaFX
- Leveraging JavaFX
- Gluon Connect
- Gluon Cloud Authentication
- Afterburner Framework
- Target Platform Tips
- Wrap Up, Q & A

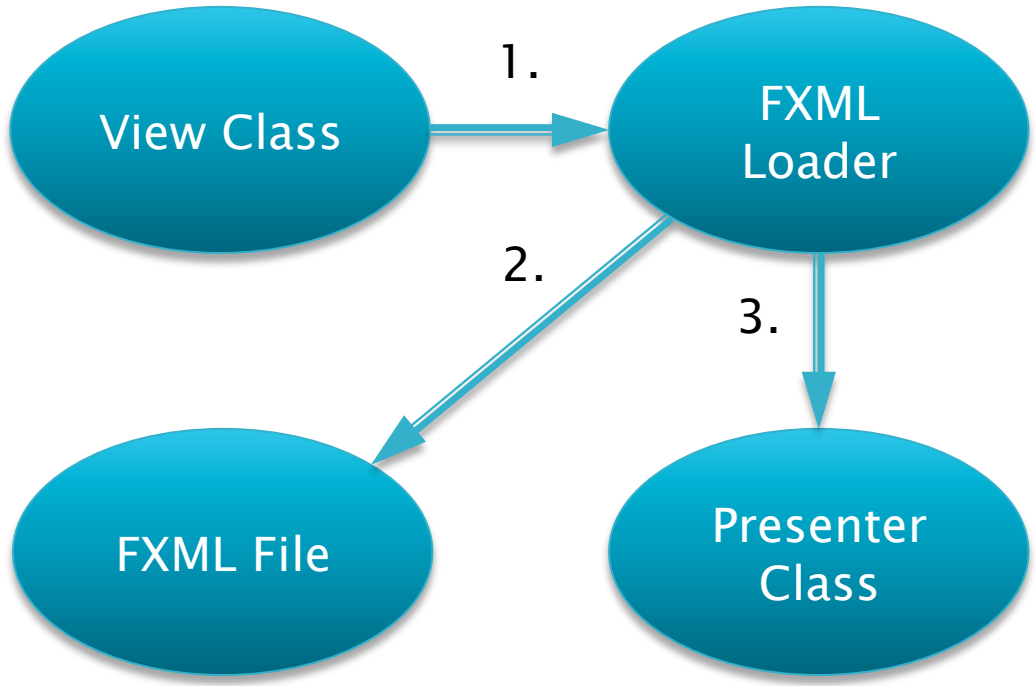
 

# Mobile App Structure

- MobileApplication
  - Main class for JavaFX mobile applications
  - Extends JavaFX Application class
  - Specify views as factories that are called on demand
- Views
  - View class invokes FXMLLoader for FXML
  - Presenter class is the FXML controller class
- Resources
  - fxml, css, images

# Mobile View with FXML




1. View Class invokes FXML Loader

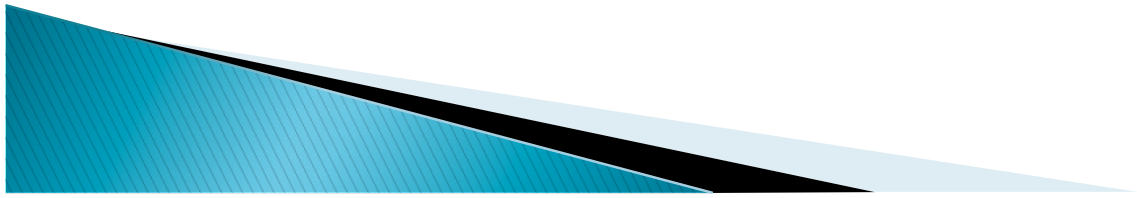
2. FXML Loader parses FXML File and builds scene graph

3. FXML Loader instantiates Presenter and invokes Presenter's `initialize()` method

# Gluon Charm

- UI Controls
  - View, AppBar, SidePopupView
  - MaterialDesignIcon, Avatar, CharmListView
  - FloatingActionButton, ProgressIndicator
- Dialogs
  - Alert, ExceptionDialog
  - DatePicker, TimePicker
- API Library



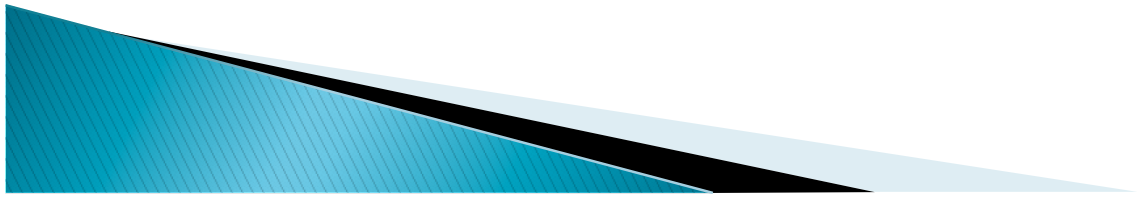
# Agenda

- Why JavaFX for Mobile?
- Gluon Framework
- FXML and Gluon Charm with JavaFX
- **Leveraging JavaFX**
- Gluon Connect
- Gluon Cloud Authentication
- Afterburner Framework
- Target Platform Tips
- Wrap Up, Q & A

# JavaFX Properties

- Why Use Properties?
  - Wraps field value, observable
  - Listeners notified when property updates
  - Used in binding expressions
- Supports
  - Read-write properties
  - Read-only properties
  - Immutable properties



# JavaFX Listeners

- Why Use Listeners?
  - Listens for changes to a property
  - Can be added and removed
  - Listeners notified when property changes
- Invalidation Listener
  - When values become invalid
- Change Listener
  - When values update
  - Access to old and new values

# Implementing Listeners

- Invalidation Listeners

```
myObject.statusProperty().addListener(observable -> {
    // status is invalid...
});
```

- Change Listeners

```
myObject.statusProperty().addListener(
    (observable, oldValue, newValue) -> {
        // status has changed...
});
```

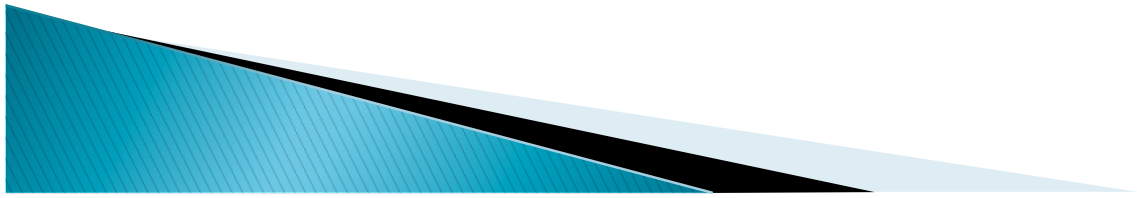
# JavaFX Binding

- What is Binding?
  - Calculates a value from sources
  - Sources are dependencies
  - Observes its dependencies for changes
  - Updates value automatically
- Why Use Binding?
  - Avoids writing listeners
  - More concise, less error prone
  - Keeps UI controls in sync with their model data

# Binding Strategies

- Unidirectional
  - Updates property when dependent property changes
- Bidirectional
  - Property updates in either direction
- Fluent API and Factory Methods
  - Binds properties from libraries of binding expressions
- Custom Binding
  - Specifies property dependencies and compute values



# Gluon Charm: Look and Feel

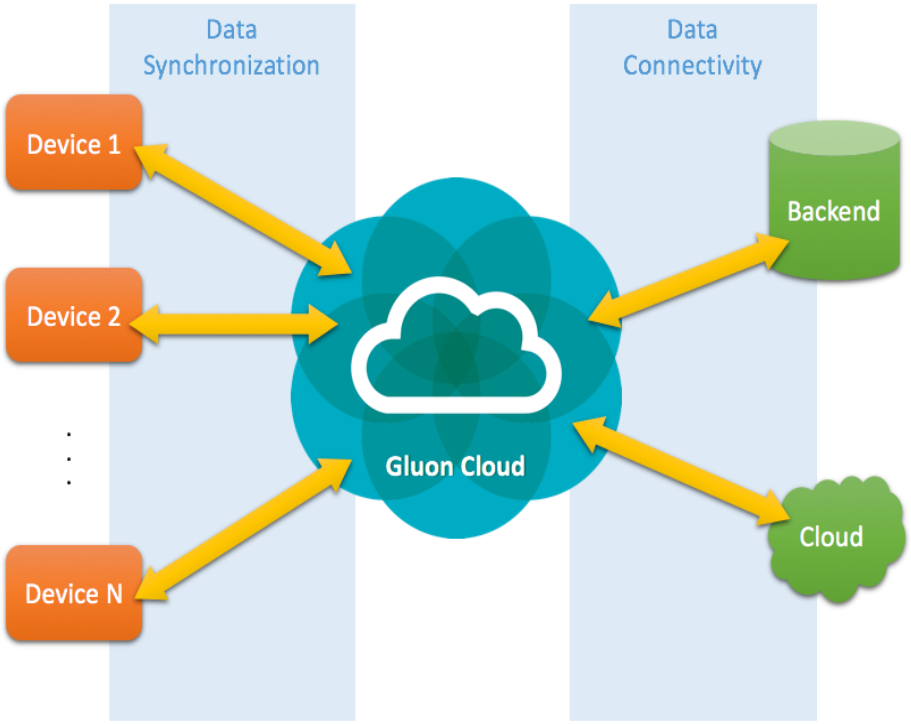

 

# Agenda

- Why JavaFX for Mobile?
- Gluon Framework
- FXML and Gluon Charm with JavaFX
- Leveraging JavaFX
- **Gluon Connect**
- Gluon Cloud Authentication
- Afterburner Framework
- Target Platform Tips
- Wrap Up, Q & A

# Gluon Cloud



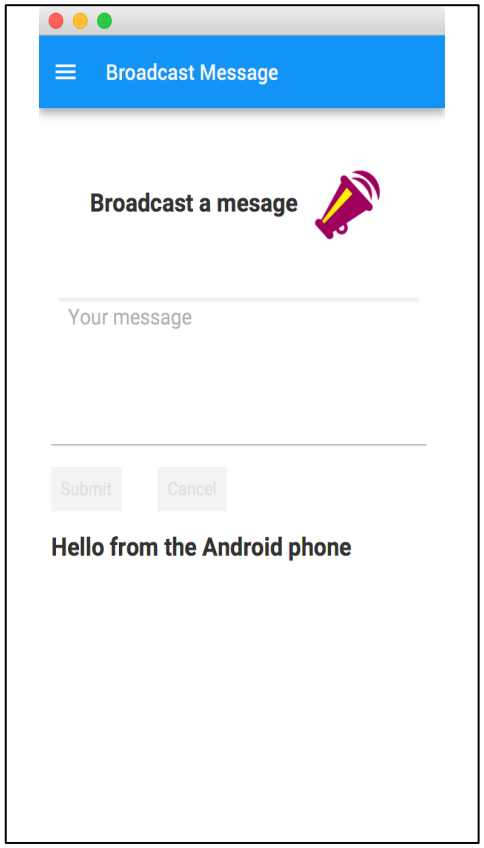

 

# Gluon Connect

- Client Side Library
  - Maps data with observable properties and lists
  - Supports bidirectional data transfers
  - Provides notifications
  - Syncs data automatically
- Supports Common Data Sources
  - Gluon CloudLink
  - File provider
  - REST provider

# Gluon Connect Demo



# Building a GluonClient

- Cloud Storage

```
GluonClient gluonClient =
   GluonClientBuilder.create().credentials(
      new GluonCredentials(APPKEY,
        APPSECRET)).build();
```

- Local Storage

```
GluonClient gluonClient = GluonClientBuilder
  .create().credentials(new GluonCredentials(
     APPKEY, APPSECRET))
       .operationMode(OperationMode.LOCAL_ONLY)
          .build();
```

# Local Binding

- Message Class

```
class Message {
   private final StringProperty text =
      new SimpleStringProperty();
   public StringProperty textProperty()
         { return text; }
}
```

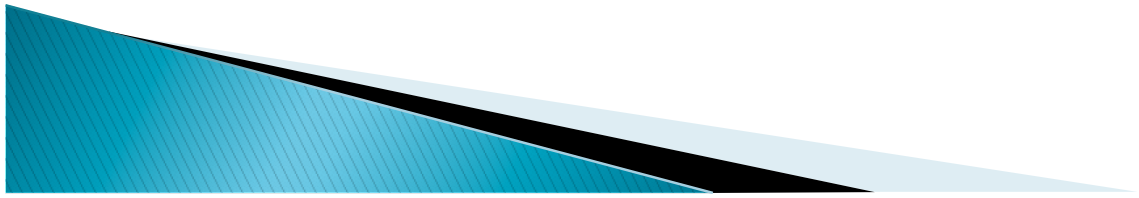
- Bind to UI Control

```
Label msgLabel = new Label();
msgLabel.textProperty().bind(msg.textProperty());
```

# Data Synchronization

- Write Through
  - Update remote copy when local data changes
  - `LIST_WRITE_THROUGH`
  - `OBJECT_WRITE_THROUGH`
- Read Through
  - Update local copy when remote data changes
  - `LIST_READ_THROUGH`
  - `OBJECT_READ_THROUGH`



# Remote Object Binding

- Cloud Object Storage

```
GluonObservableObject<Message> gluonMsg =
    DataProvider.retrieveObject(
      gluonClient.createObjectDataReader("data",
         Message.class, SyncFlag.OBJECT_READ_THROUGH,
           SyncFlag.OBJECT_WRITE_THROUGH));
  . . .

  Label msgLabel = new Label();
  msgLabel.textProperty().bind(
       gluonMsg.get().textProperty());
```

# Local Lists

- Observable UI Control

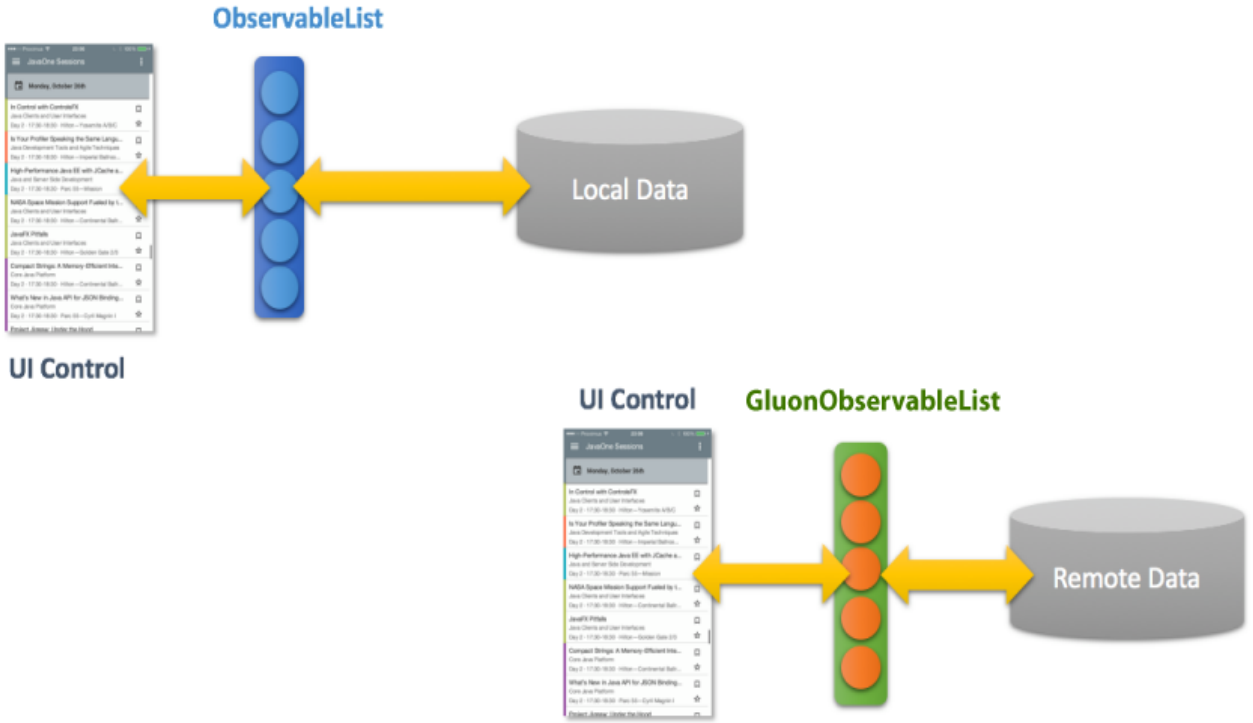
```
ObservableList<Message> messages =
   FXCollections.observableArrayList();
loadMessages(messages);

ListView<Message> listView = new ListView<>();
listView.setItems(messages);
```

- Advantages
  - Add and remove elements in list
  - Updates UI control automatically

# Remote Binding



# Remote List Binding

- Cloud Storage and Synchronization

```
GluonObservableList<Message> gluonList =
   DataProvider.retrieveList(
       gluonClient.createListDataReader("data",
                Message.class));
. . .

ListView<Message> listView = new ListView<>();
listView.setItems(gluonList);
```

# List Synchronizations

- Two Levels
  - Update list for adds and removals
  - Update list elements
- First Level

```
SyncFlag.LIST_WRITE_THROUGH
SyncFlag.LIST_READ_THROUGH
```

- Second Level

```
SyncFlag.OBJECT_WRITE_THROUGH
SyncFlag.OBJECT_READ_THROUGH
```

# List Synchronization

- BPData Local Storage

```
GluonObservableList<BPData> gluonBPData =
    DataProvider.retrieveList(
      gluonClient.createListDataReader(BPDATA,
         BPData.class, SyncFlag.LIST_WRITE_THROUGH,
            SyncFlag.OBJECT_WRITE_THROUGH));

gluonBPData.stateProperty()
  .addListener((obs, ov, nv) -> {
     if (ConnectState.SUCCEEDED.equals(nv)) {
        bpList.set(gluonBPData);
      }
});
```

# Gluon Cloud: Authentication

# Agenda

- Why JavaFX for Mobile?
- Gluon Framework
- FXML and Gluon Charm with JavaFX
- Leveraging JavaFX
- Gluon Connect
- **Gluon Cloud Authentication**
- Afterburner Framework
- Target Platform Tips
- Wrap Up, Q & A

# Demo Apps

- BPMonitor 
  - Four views (Readings, Graph, Stats, Edit)
  - Stores readings locally on device
  - No data sharing with other devices
- BPCloud 
  - Four views (Readings, Graph, Stats, Edit)
  - Writes data to cloud
  - Data tied to authenticated users
  - User can access data from any authenticated device

# Authentication Mode

- Credentials
  - Register application on CloudLink
  - Application keys supplied by Gluon
- Gluon Client

```
GluonClient gluonClient =
   GluonClientBuilder.create().credentials(
      new GluonCredentials(APPKEY, APPSECRET))
         .authenticationMode(AuthenticationMode.USER)
            .build();
```

# Data Provider

- Service Class

```
void getData() {
    GluonObservableList<BPData> gluonBPData =
      DataProvider.retrieveList(
        gluonClient.createListDataReader(
           user.get().getNick() +
              user.get().getNetworkId(), BPData.class,
                 SyncFlag.LIST_READ_THROUGH,
                 SyncFlag.LIST_WRITE_THROUGH,
                 SyncFlag.OBJECT_READ_THROUGH,
                 SyncFlag.OBJECT_WRITE_THROUGH));
   . . .
}
```

# Authentication Timing

- Readings Presenter

```
readings.addEventHandler(LifecycleEvent.SHOWN,
   new EventHandler<LifecycleEvent>() {
        @Override
        public void handle(LifecycleEvent event) {
           readings.removeEventHandler(
              LifecycleEvent.SHOWN, this);
           service.retrieveReadings(); }
 });
```

- Service Class

```
public void retrieveReadings() {
     gluonClient.authenticate(this::getData);
}
```

# Authenticated User Property

- Service Class

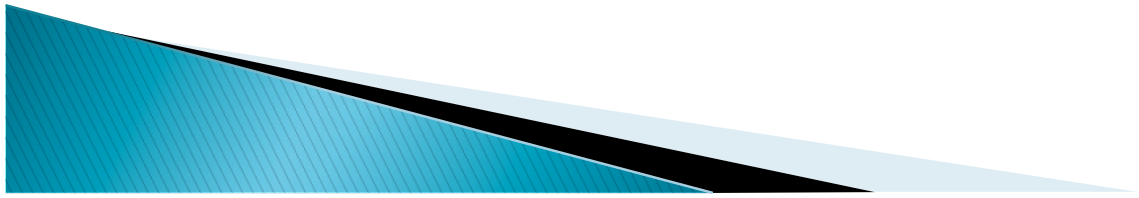
```
private final ObjectProperty<User> user =
        new SimpleObjectProperty<>();
. . .
@PostConstruct
public void postConstruct() {
    gluonClient = GluonClientProvider.getGluonClient();
    user.bind(gluonClient.authenticatedUserProperty());
}
```

# Authenticated Information

- Readings Presenter

```
@Inject
private Service service;
. . .
public void initialize() {
   AppBar appBar =
      MobileApplication.getInstance().getAppBar();
      appBar.setTitleText("BP: " +
         service.getUser.getName());
}
```



# Login Methods

- Supported Platforms
  - Twitter
  - Facebook
  - Google+
- Applications
  - Create on selected platform
  - Platform generates application login keys
  - Copy login keys to Gluon CloudLink portal
  - Apps now provide platform user authentication

# Agenda

- Why JavaFX for Mobile?
- Gluon Framework
- FXML and Gluon Charm with JavaFX
- Leveraging JavaFX
- Gluon Connect
- Gluon Cloud Authentication
- Afterburner Framework
- Target Platform Tips
- Wrap Up, Q & A

# Afterburner Framework

- What is Afterburner?
  - Lightweight framework
  - Provides dependency injection
- Why Use Afterburner?
  - Injects FXML for views and Java objects
  - Generates Java boilerplate code
- Advantages
  - Reduces Java code
  - Safe and easy object sharing among views

# Agenda

- Why JavaFX for Mobile?
- Gluon Framework
- FXML and Gluon Charm with JavaFX
- Leveraging JavaFX
- Gluon Connect
- Gluon Cloud Authentication
- Afterburner Framework
- Target Platform Tips
- Wrap Up, Q & A

# IOS Platform Tips

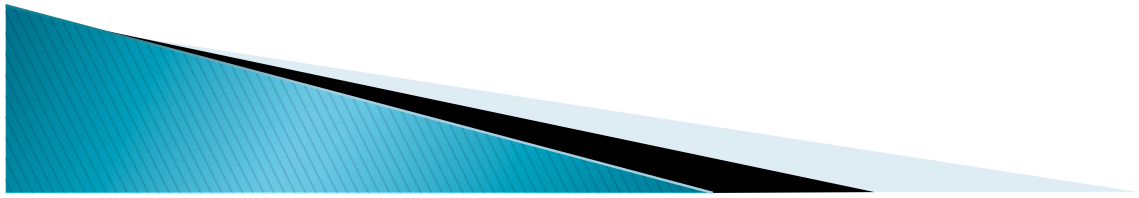
- IOS Platform Files
  - Icons, splash screens, iTunes artwork, IOS settings

```
src/ios/assets
src/ios/Default-Info.plist
```

- Install and Execute on Connected Device

```
Tasks | launch | launchIOSDevice
```

  - Delete app first for “clean” install
  - IDE output window provides runtime feedback



# Android Platform Tips

- Android Platform Files

```
src/android/res
src/android/AndroidManifest.xml
```

- Prepare Device
  - Enable Developer Options and USB debugging
  - Select proper USB configuration: MTP or PTP
- Install and Execute on Connected Device

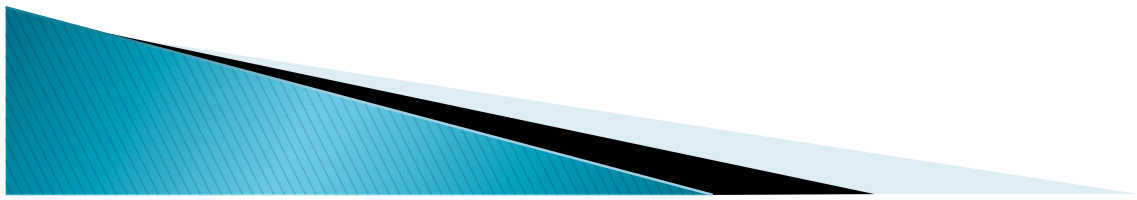
```
Tasks | android | androidInstall
```

  - Delete app first for “clean” install
  - SDK tool `adb` provides runtime feedback

# Summary

- JavaFX Advantages
  - Properties and controls
  - Attach listeners to properties
  - Binding techniques
  - Asynchronous tasks to keep the UI responsive
- Useful Frameworks
  - Gluon/Gradle framework for mobile deployment
  - Afterburner framework for dependency injection
  - Platform/IDE independence



# Wrap Up

- Thanks for Coming!
- Contact Info
  - paul@asgteach.com @paul_asgteach
  - gail@asgteach.com @gail_asgteach
- Session Examples
    - asgteach.com
    - Click to Download
  - Book Give Away
  - Q & A