ANDROID CROSS PLATFORM DEVELOPMENT

MOTIVATION

- We develop on many native platforms
  - iOS
  - Android
  - Mac
- Minimize duplicated efforts
- Better collaboration across teams

CONSOLIDATION

» Model / Domain objects

» Interfaces to platform APIs

» Business logic

SOLUTIONS

- Webviews
  - Sencha, PhoneGap, etc
- Cross compilation / Code Generation
  - Xamarin, Corona, J2OBJ-C
- Low level
  - C, C++, etc

CHALLENEGES

» Performance

» Massive dependencies

» Bindings

» Learning curve

C++

» C++ 11 is much more approachable

» Performant

» Great tooling and libraries

C++ ON ANDROID

» Compiled via the NDK

» Android tooling support is lacking but steadily improving

» A new language to learn

» JNI wrappers are ugly and difficult to maintain

DJINNI

» Developed by Dropbox

» A tool for generating cross platform bindings

» Allows connecting C++ to either Java or Objectve-C

IDL

- Djinni generates interfaces using an interface description file
  - Enums
  - Records (Model/Domain objects)
  - Interfaces
    - Some implemented in Java and Obj-C
    - Some implemented in C++ and called from Java and Obj-C

USER PROFILE EXAMPLE

DEFINE OUR STORAGE INTERFACE

```
IKeyValueStorage = interface +j +o {
  writeString(key: string, value: string);
  writeInt32(key: string, value: i32);
  writeBool(key: string, value: bool);

  remove(key: string);

  readString(key: string) : optional<string>;
  readInt32(key: string) : i32;
  readBool(key: string) : bool;
}
```

DEFINE OUR SHARED SERVICE

```
IUserProfileService = interface +c {
  static createUserProfileService(keyValueStorage: IKeyValueStorage) : IUserProfileService;

  updateUserProfile(userProfile: UserProfile) : UserProfile;
  getUserProfile() : optional<UserProfile>;
  removeUserProfile();

  addListener(listener: IUserProfileServiceListener);
  removeListener(listener: IUserProfileServiceListener);
}

IUserProfileServiceListener = interface +o +j {
  userProfileUpdated(userProfile: UserProfile);
  userProfileRemoved();
}
```

DEFINE OUR MODEL

```
UserProfile = record {
  firstName : string;
  lastName : string;
}
```

C++

```cpp
class IKeyValueStorage {
public:
    virtual ~IKeyValueStorage() {}

    virtual void writeString(const std::string & key, const std::string & value) = 0;

    virtual void writeInt32(const std::string & key, int32_t value) = 0;

    virtual void writeBool(const std::string & key, bool value) = 0;

    virtual void writeBinary(const std::string & key, const std::vector<uint8_t> & value) = 0;

    virtual void remove(const std::string & key) = 0;

    virtual boost::optional<std::string> readString(const std::string & key) = 0;

    virtual int32_t readInt32(const std::string & key) = 0;

    virtual bool readBool(const std::string & key) = 0;

    virtual boost::optional<std::vector<uint8_t>> readBinary(const std::string & key) = 0;
};
```

JAVA

```java
public abstract class IKeyValueStorage {
    public abstract void writeString(String key, String value);

    public abstract void writeInt32(String key, int value);

    public abstract void writeBool(String key, boolean value);

    public abstract void writeBinary(String key, byte[] value);

    public abstract void remove(String key);

    public abstract String readString(String key);

    public abstract int readInt32(String key);

    public abstract boolean readBool(String key);

    public abstract byte[] readBinary(String key);
}
```

OBJ-C

```objc
@protocol FLIKeyValueStorage

- (void)writeString:(nonnull NSString *)key
              value:(nonnull NSString *)value;

- (void)writeInt32:(nonnull NSString *)key
             value:(int32_t)value;

- (void)writeBool:(nonnull NSString *)key
            value:(BOOL)value;

- (void)writeBinary:(nonnull NSString *)key
              value:(nonnull NSData *)value;

- (void)remove:(nonnull NSString *)key;

- (nullable NSString *)readString:(nonnull NSString *)key;

- (nullable NSNumber *)readInt32:(nonnull NSString *)key;

- (nullable NSNumber *)readBool:(nonnull NSString *)key;

- (nullable NSData *)readBinary:(nonnull NSString *)key;

@end
```

SO WE HAVE OUR INTERFACES

LETS TIE THEM TOGETHER

IMPLEMENT INTERFACES IN JAVA AND OBJ-C

» `IKeyValueStorage`

JAVA IMPL

```
private final SharedPreferences sharedPreferences;

public KeyValueStore(SharedPreferences sharedPreferences) {
    this.sharedPreferences = sharedPreferences;
}
```

```java
@Override
public void writeString(String key, String value) {
    sharedPreferences.edit().putString(key, value).apply();
}

@Override
public void writeInt32(String key, int value) {
    sharedPreferences.edit().putInt(key, value).apply();
}

@Override
public void writeBool(String key, boolean value) {
    sharedPreferences.edit().putBoolean(key, value).apply();
}

@Override
public void writeBinary(String key, byte[] value) {
    throw new RuntimeException("Unable to store binary data");
}
```

```java
@Override
public void remove(String key) {
    sharedPreferences.edit().remove(key).apply();
}

@Override
public String readString(String key) {
    return sharedPreferences.getString(key, null);
}

@Override
public int readInt32(String key) {
    return sharedPreferences.getInt(key, 0);
}

@Override
public boolean readBool(String key) {
    return sharedPreferences.getBoolean(key, false);
}

@Override
public byte[] readBinary(String key) {
    throw new RuntimeException("Unable to get binary data");
}
```

IMPLEMENT A PROFILE SERVICE IN C++

```
namespace xplatform {

    static std::string const kProfileKey = "profile";
    static std::string const kProfileFirstNameKey = kProfileKey + "." + "firstName";
    static std::string const kProfileLastNameKey = kProfileKey + "." + "lastName";

    std::shared_ptr<IUserProfileService> IUserProfileService::createUserProfileService(
        const std::shared_ptr<IKeyValueStorage> & keyValueStorage)
    {
        return std::make_shared<UserProfileServiceImpl>(keyValueStorage);
    }

    UserProfileServiceImpl::UserProfileServiceImpl(
        const std::shared_ptr<IKeyValueStorage> & keyValueStorage) : keyValueStorage(keyValueStorage)
    {

    }

}
```

“

IMPLEMENT GET AND REMOVE

```
    UserProfile UserProfileServiceImpl::updateUserProfile(const UserProfile & userProfile)
    {
        keyValueStorage->writeString(kProfileFirstNameKey, userProfile.firstName);
        keyValueStorage->writeString(kProfileLastNameKey, userProfile.lastName);

        notifyListenersProfileUpdated(*(getUserProfile()));
        return userProfile;
    }

    boost::optional<UserProfile> UserProfileServiceImpl::getUserProfile()
    {
        auto firstName = keyValueStorage->readString(kProfileFirstNameKey);
        //If we can't read the first name, we don't have a profile.
        if (firstName) {
            auto lastName = keyValueStorage->readString(kProfileLastNameKey);
            return UserProfile(*firstName, *lastName);
        }
        boost::optional<UserProfile> userProfile;
        return userProfile;
    }

    void UserProfileServiceImpl::removeUserProfile()
    {
        keyValueStorage->remove(kProfileFirstNameKey);
        keyValueStorage->remove(kProfileLastNameKey);

        notifyListenersProfileRemoved();
    }
```

IMPLEMENT THE LISTENERS

```
std::vector<UserProfileServiceImpl::UserProfileListenerPtr>::iterator
 UserProfileServiceImpl::findListener(UserProfileListenerPtr listener)
{
    return std::find(listeners.begin(), listeners.end(), listener);
}

void UserProfileServiceImpl::addListener(
    const std::shared_ptr<IUserProfileServiceListener> & listener)
{
    if (!isListenerRegistered(listener)) {
        listeners.push_back(listener);
    }
}

void UserProfileServiceImpl::removeListener(
    const std::shared_ptr<IUserProfileServiceListener> & listener)
{
    auto it = findListener(listener);
    if (it != listeners.end()) {
        listeners.erase(it);
    }
}
```

EXAMPLE PROJECT

» https://github.com/zsiegel/android-xplatform

QUESTIONS