Development Techniques for Native/Hybrid Tizen Apps

Presented by **Kirill Kruchinkin**

st symphony teleca

TIZEN™ DEVELOPER CONFERENCE 2013 SAN FRANCISCO

Agenda

- Introduction and Definitions
- Practices
- Case Studies

Introduction & Definitions

2 App Types

Browser Apps

Installable Apps

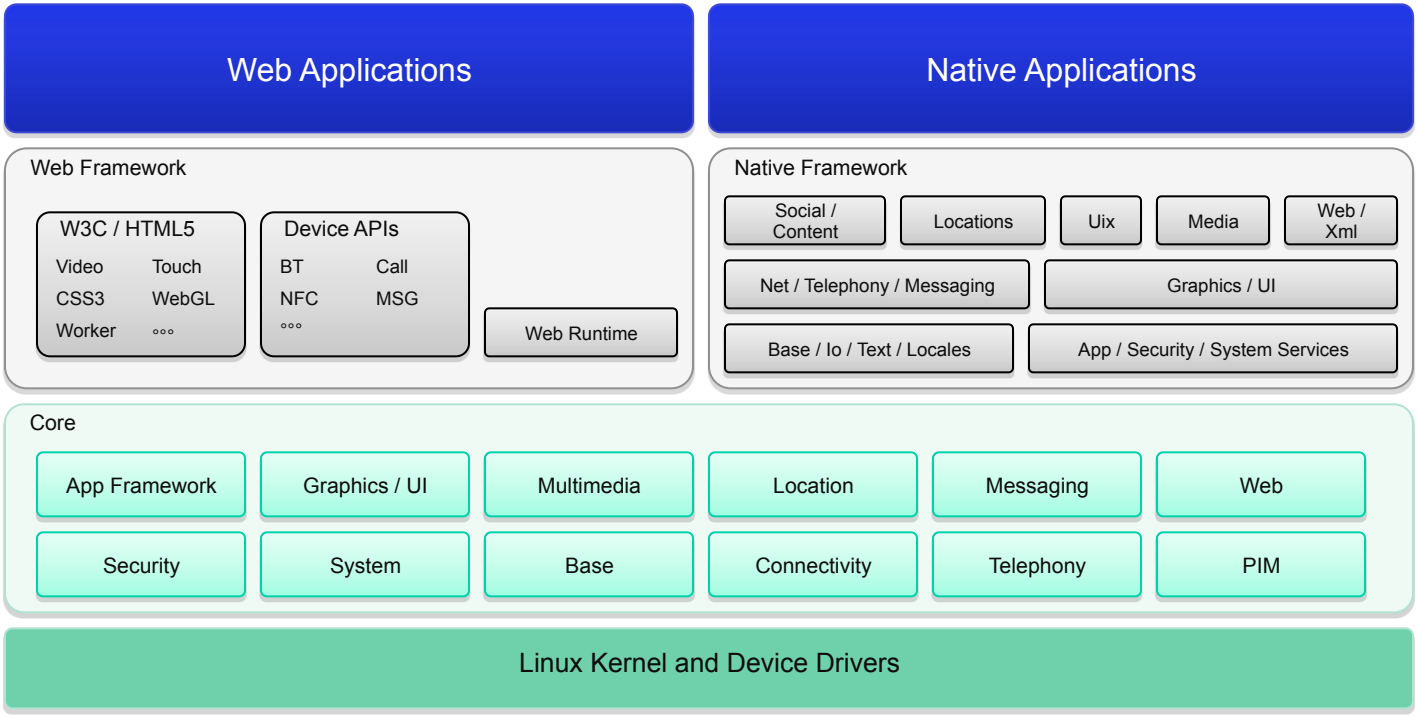
- Native
- WebApps
- Hybrid

Browser Apps

One platform for many devices

- **HTML5 is everywhere**
- **HTML5 is mobile**
- **HTML5 is skillful**
- **HTML5 is open**

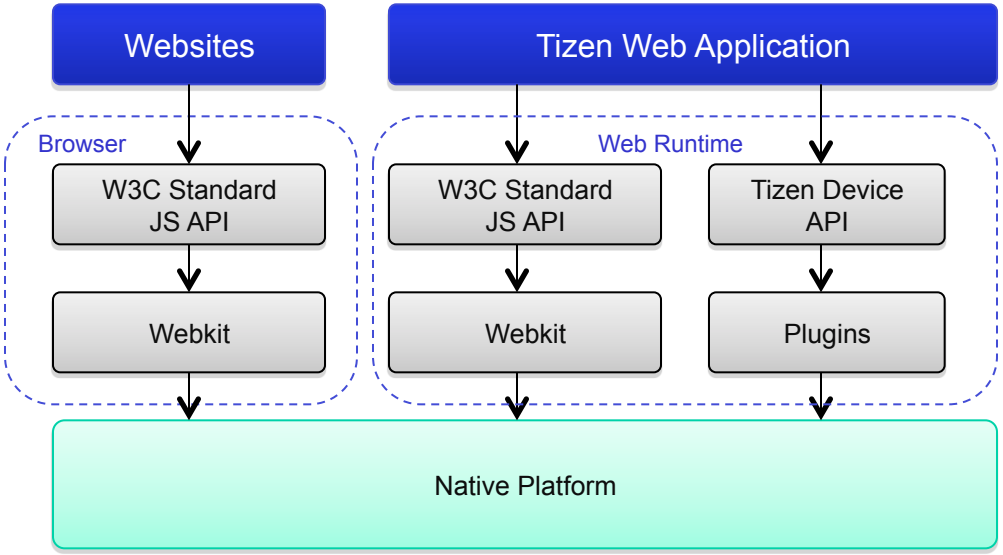
Tizen Application Level



Web Apps

Advantages of Browser Apps with HTML5

- Full offline support
- Device APIs
- Native UI



Native Apps

- Broader access to device hardware
- Better integration with system features
- Faster, smoother and better UI

Hybrid Apps

- Web UI and Native Services
- Rapid development of UI
- Data processing in background
- Share native services between applications
- Native open-source libraries

Always a Challenging Decision

	Native	Web	Hybrid
Rapid UI development	-	+	+
Open standards based	-	+	+
Performance in heavy calculation	+	-	+
Works offline	+	+	+
Background support	+	-	+
Deep HW/platform integration	+	-	+
Low complexity	-	+	-

Practices

Steps to a Hybrid App

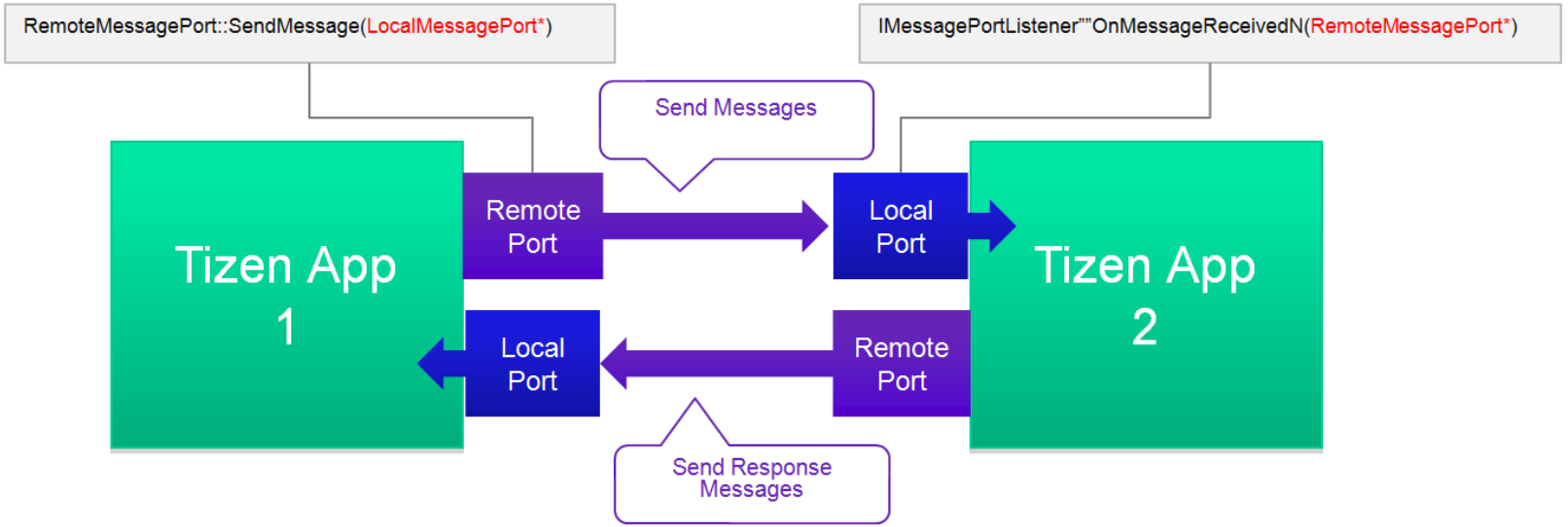
- Create a web application project
- Create a native service project
- Define the reference in the project's properties
- Define a "bridge" between the web and service applications
- Build the application
- Install the application

Bridging Native and JS

- Messaging layer between web and native
- Message Port
- Websocket
- JavaScriptBridge plugin

App Communication through Message Ports

- Implements a communication between two applications
- Platform built-in



Message Porting: Web Step 1

- Define message handler and create a local message port

```
function onReceive(data, remoteMsgPort) {
   console.log('Received data to \'' + remoteMsgPort.name + '\'');
         // Search value with the "response" key
   for(var i in data) {
      if(data[i].key == "response") {
          response = data[i].value;
          break;
      }
   }
}

try {
           localMessagePort = tizen.messageport.requestLocalMessagePort(localMessagePortName);
    localMessagePortWatchId = localMessagePort.addMessagePortListener(function(data, remote) {

              onReceive(data, remote); } );
} catch (e) {
    console.log("Request local message port Error");
}
```

Message Porting: Web Step 2

- **Launch service application**

```
function onSuccess() {
    console.log("Service App launched successfully!");
}

function onError(err) {
    console.log("Service Applaunch failed ="+ err);
}

try {
                tizen.application.launch(appId, onSuccess, onError);
 } catch (e) {
                // handle error
}
```

Message Porting: Web Step 3

- **Open remote port**

```
try {
    remoteMessagePort = tizen.messageport.requestRemoteMessagePort(appId, servicePortName);
} catch (e) {
    console.log("Request remote message port Error ");
}
```

Message Porting: Native Step 1

```
void ServiceChannel::OnMessageReceivedN(RemoteMessagePort* pRemoteMessagePort, IMap* pMessage)
{
    String *pData = static_cast<String *>(pMessage->GetValue(String(L"command")));
     if (pData != null) {
        IJsonValue* pJson = getJsonObj(pData);
        if(null != pJson) {
            // Extract command from pJson in to the pCommand pointer
            ...
            if(null != pCommand) {
                if (pCommand->Equals(DO_COMMAND_1, true)) {
                    // extract additional parameters from pJson and perform requested operation – it is would be better to schedule this operation and
                    // execute it asynchronously (for example, send event to application, set timer, run thread etc,..)
                }
            }
            // Relase pJsonObject
            ...
        }
    // Prepare answer in str variable
    ...

    //Send response in case of bi-direction communication
    HashMap *pMap = new HashMap(SingleObjectDeleter);
    pMap->Construct();
    pMap->Add(new String(L"response"), new String(str));
    pRemoteMessagePort->SendMessage(mLocalMessagePort, pMap);
    // deallocate resources
    ...
}
```

Message Porting: Native Step 2

- **Create local port**

```
mLocalMessagePort = MessagePortManager::RequestLocalMessagePort(mLocalPortName);
mLocalMessagePort->AddMessagePortListener( messagePortListenerImplementation );
```

Message Porting: Native Step 3

- Create remote port

```
mRemoteMessagePort = MessagePortManager::RequestRemoteMessagePort(mRemoteAppID, mRemotePortName);
```

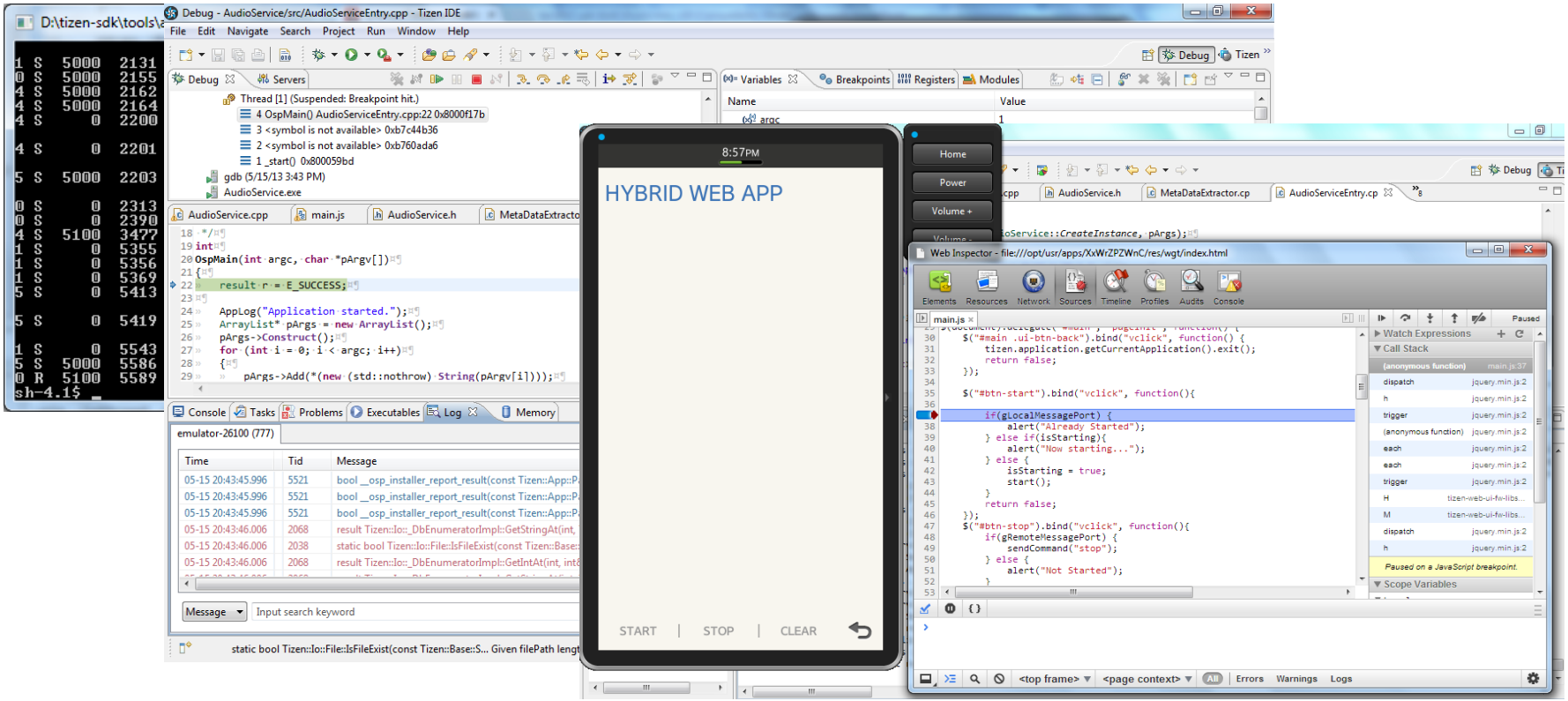
Websocket Approach

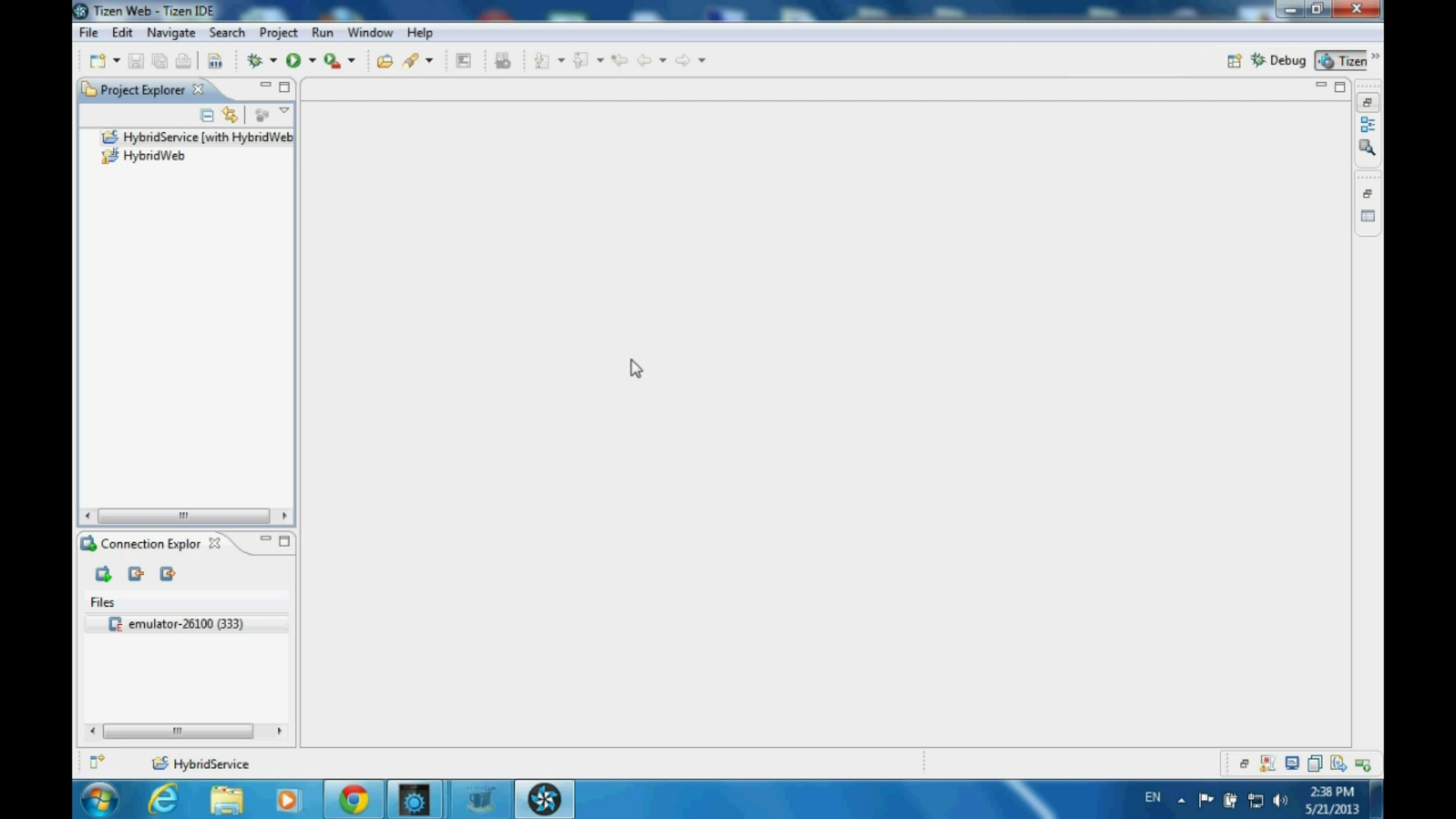
- Works in other platforms
- Works asynchronously
- Doesn’t have a payload limit
- Handshake implementation RFC 6455

Javascriptbridge Plugin

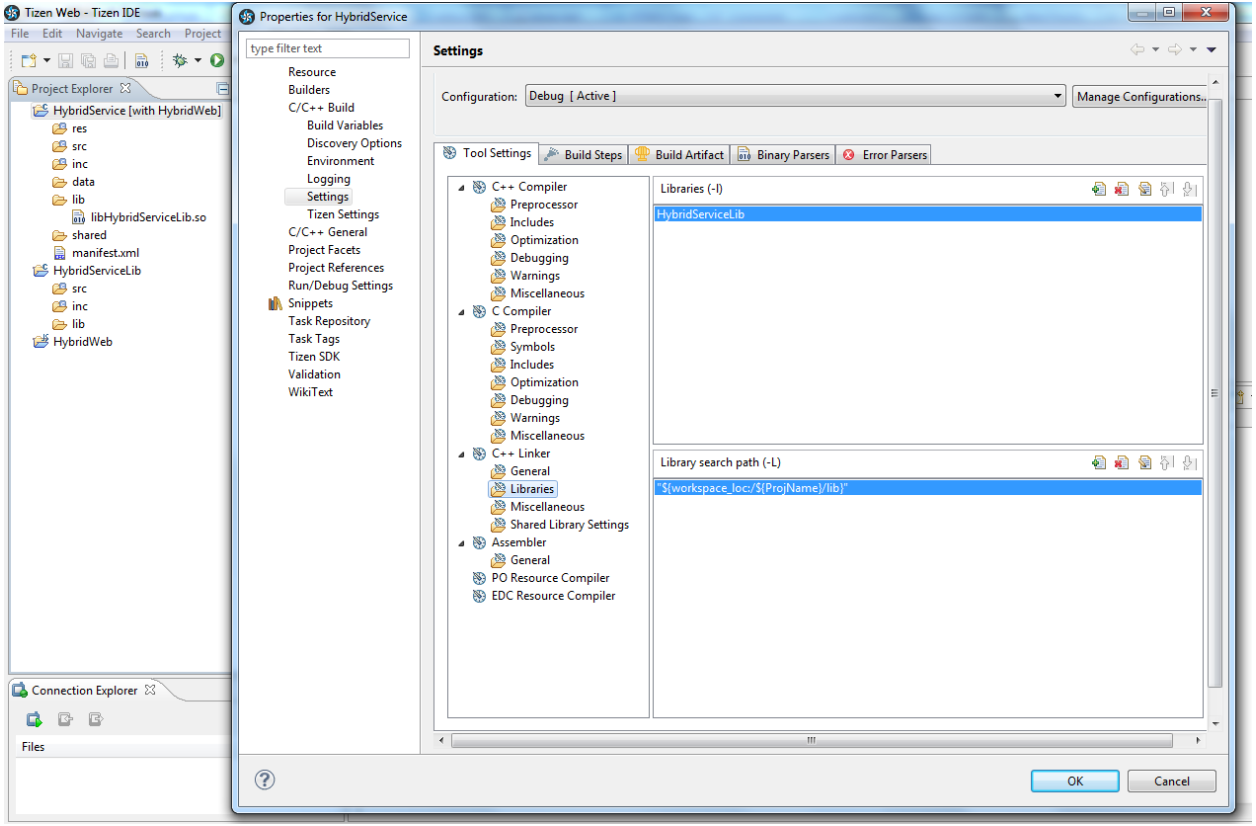
- Include the JavaScriptBridge plugin in a Web page
- Implement a JS method to be called from native
- Implement Tizen::Web::Controls::IJavaScriptBridge on native sode

Debugging




Tizen Web - Tizen IDE
File Edit Navigate Search Project Run Window Help
Debug
Tizen
Project Explorer
HybridService [with HybridWeb
HybridWeb
Connection Explor
Files
emulator-26100 (333)
HybridService
EN
2:38 PM
5/21/2013

3rd Party lib Static Linking



3rd Party lib Dynamic Linking

```
//get a lib Path
wchar_t const * const LIB_NAME = L"lib/libHybridServiceLib.so";
Tizen::App::App *pxApp = Tizen::App::App::GetInstance();
String libPath = pxApp-> GetAppRootPath();
libPath.Append(LIB_NAME);

//get initialized instance of Library
Tizen::Base::Runtime::Library library;
result r = library.Construct(libPath);

//get a function pointer
void *pMethod = library.GetProcAddress(L"CalculateMd5");
pCalculateHash = reinterpret_cast<char* (*)(byte const *, int)>(pMethod);

//use the function
pCalculateHash(pByteArray, byteArrayLength);
```

JSON

- JavaScript Object Notation (JSON) is a text-based open standard designed for human-readable data interchange
- Parsers are available for JavaScript (native JS, jQuery and other frameworks) and a native API
- Human-readable format that is easy for debug
- Web services provide data in JSON format
- Key-value data format is enough to present any data that is required in JavaScript

JSON Usage

- **Web**

```
// parse
 var jsonData = JSON.parse(strJSON);

 // compose
 var strJSON = JSON.stringify(jsonData)
```

- **Native**
  - **Tizen::Web::Json::JsonParser for parsing**
  - **Tizen::Web::Json::JsonWriter for composing**

3rd Party frameworks

- Cordova
- Appcelerator Titanium
- Cocos2d-html5

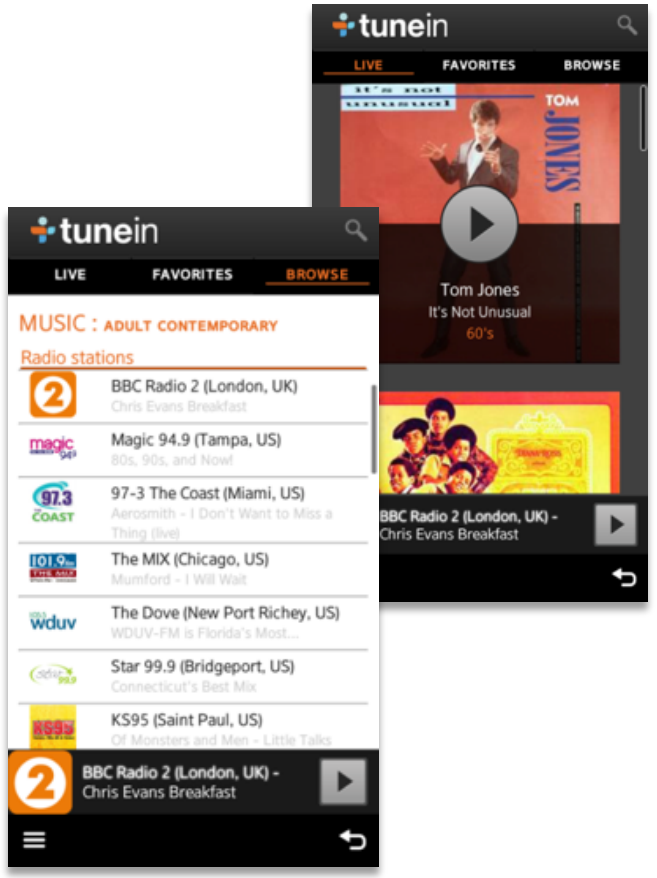
Tips & Tricks

- Use Message Ports as a bridge
- Implement re-try logic to open Message Ports
- Use JSON to transfer data
- Widely use Resource Acquisition Is Initialization (RAII) idiom
- Share assets and layouts between multiple apps in your package
- Specify all privileges needed for the app in the manifest of the main project

Case Studies

TuneIn

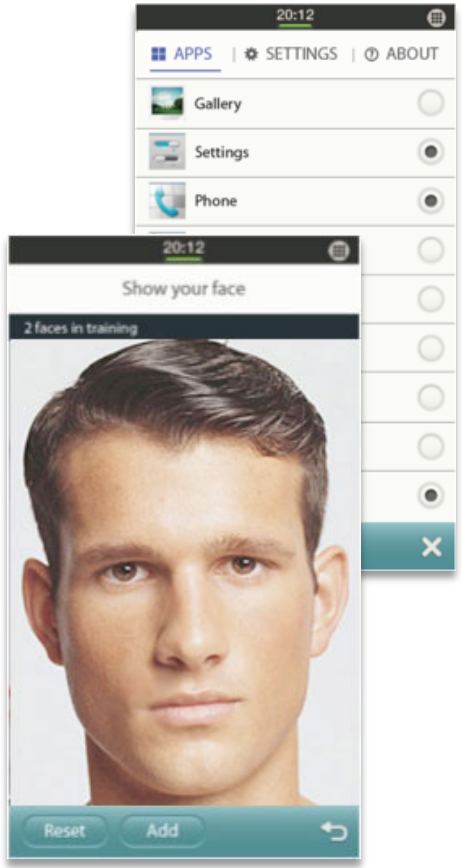
- **TuneIn offers the user the ability to listen to streaming audio from thousands of radio stations**
- **It is a Hybrid App to support more audio formats and resolve stream handling**
- **Web App**
  - UI part
  - TuneIn server communication
- **Native Service**
  - audio playing
  - playlist downloading



TuneIn for Tizen

Peeking Protector: native experience

- **A face recognition application that restricts access to a whole device or selected applications unless the owner's face is recognized.**
- **The application is fully native because the app should:**
  - intercept launching other protecting apps
  - have the top-most visibility
  - hook hard keys like Home and Power
  - utilize native built-in face recognition API
  - use multi-threading to be responsive while the face recognition piece is in progress



Peeking Protector for Tizen

TIZEN™

DEVELOPER CONFERENCE 2013 SAN FRANCISCO