



Xamarin Evolve 2014

Integrating with Salesforce

Craig Dunn
craig@xamarin.com

 Xamarin University

SaaS On Your Phone

Xamarin University

Agenda

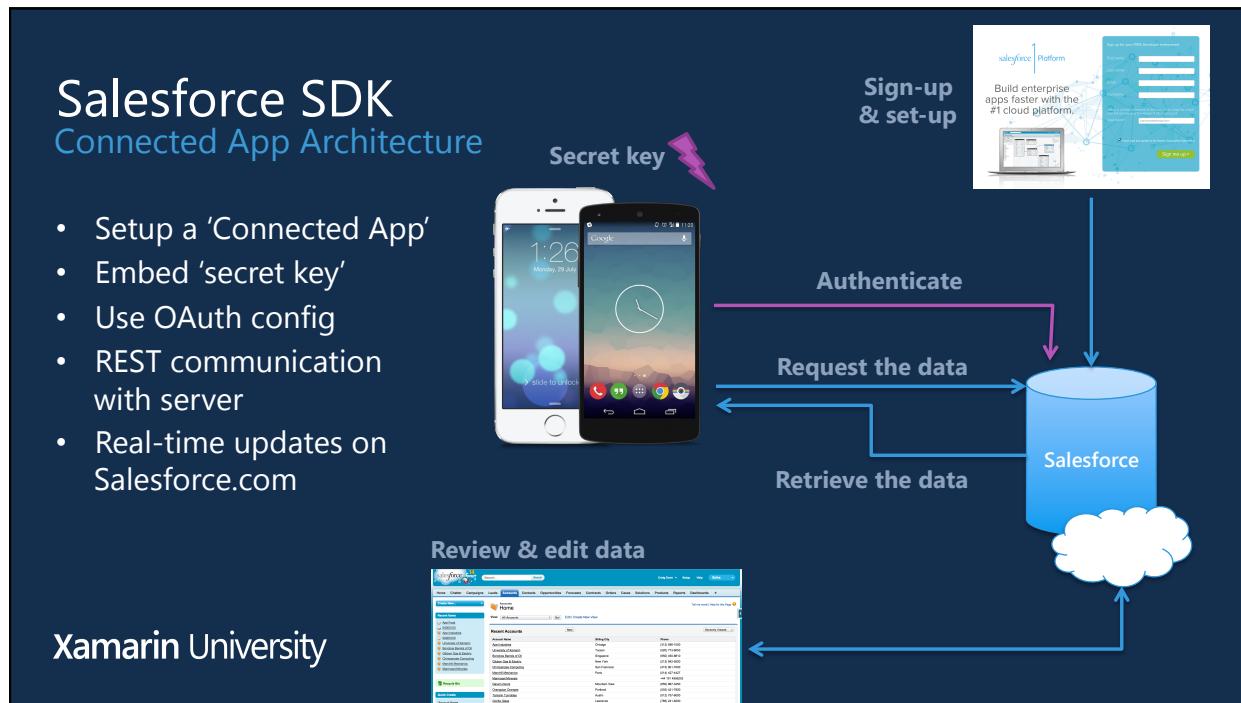
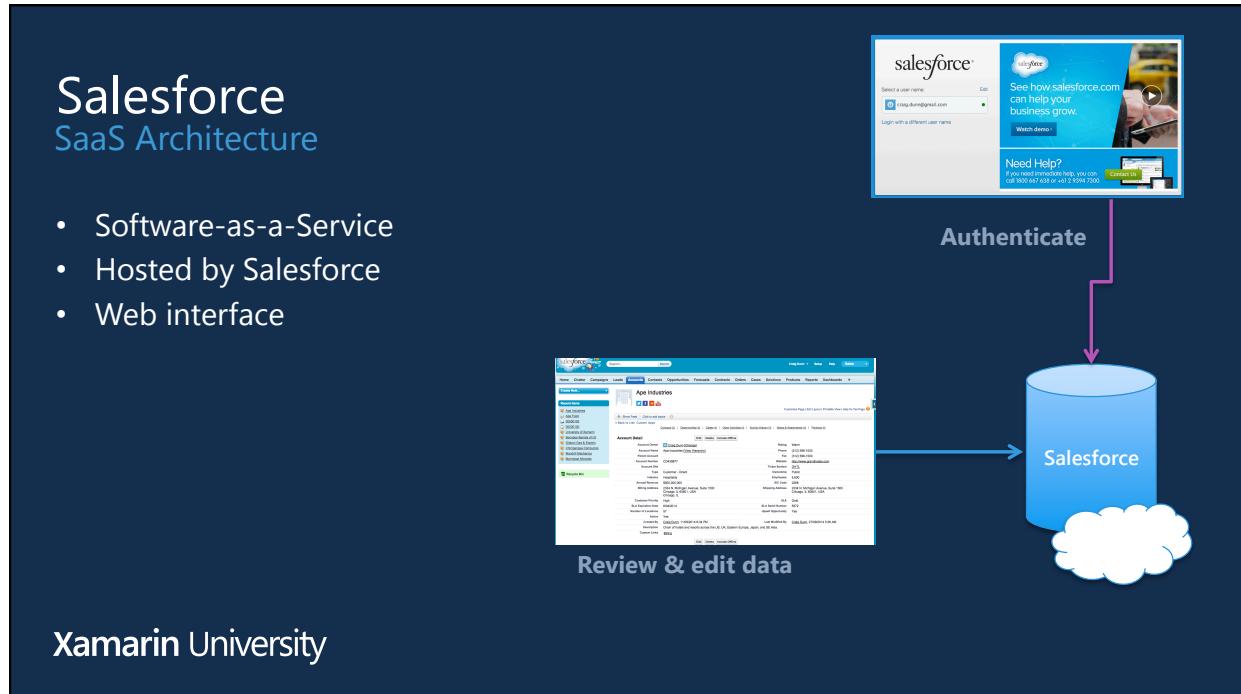
1. Salesforce SDK
2. Getting Started
3. Customization

Xamarin University



Salesforce SDK

Xamarin University



OAuth 2.0 Authentication protocol

- Client ID
- Client Secret
- Redirect URL

The diagram illustrates the OAuth 2.0 authentication protocol flow. It shows four main components: a Client Application (represented by a server icon), a User (represented by a person using a laptop), a Resource Server (represented by three server icons labeled 'na1.salesforce.com'), and an Authorization Server (represented by a server icon labeled 'login.salesforce.com'). The flow starts with the User interacting with the Client Application. The Client Application then interacts with the Authorization Server to 'Authenticates, Grants Access'. The Authorization Server then 'Issues Tokens' to the Client Application. Finally, the Client Application 'Accesses Data' from the Resource Server.

Xamarin University

image: Salesforce

OAuth 2.0 Authentication protocol

- Client ID
- Client Secret
- Redirect URL

The screenshot shows two sequential screens from a mobile application. The first screen is an 'Authenticate' screen for 'salesforce' with fields for email ('craig.dunn@gmail.com') and password, and buttons for 'Log in to Salesforce', 'Remember User Name', and 'Forgot your password?'. The second screen is a permission request dialog from 'XamEvolve14' asking for access to basic information, manage data, and perform requests on behalf of the user. It includes 'Allow' and 'Deny' buttons. Below the screenshots, the same system architecture diagram is shown, mapping the mobile steps to the 'User', 'Authorization Server (login.salesforce.com)', and 'Resource Server (na1.salesforce.com)' components.

Xamarin University

REST

Representational State Transfer

- Resources represented by URLs
- HTTP verbs GET POST PUT DELETE (PATCH)
- HTTP status codes
- HTTP headers for authentication
- JSON payloads

Xamarin University

http://en.wikipedia.org/wiki/Representational_state_transfer

REST

When to use REST?

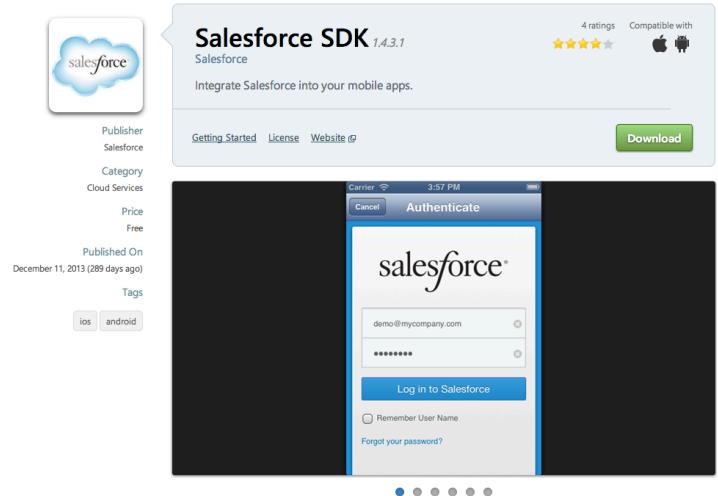
API Name	What It's For	When to Use It	Protocol	Data Format
REST API	Accessing objects in your organization using REST.	You want to leverage the REST architecture to integrate with your organization. No WSDL requirement. Well-suited for browser-based applications, mobile apps, and highly-interactive social applications.	REST	JSON, XML

Xamarin University

Client Libraries

Salesforce SDK Component

- Xamarin Component Store
 - iOS
 - Android

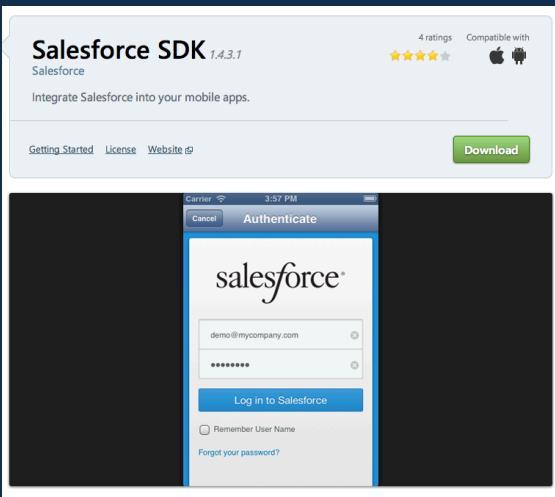


Xamarin University

Client Libraries

Salesforce SDK Component

- Create, Update, and Delete SObjects
- Query data with SOQL
- Search with SOSL
- Consistent C# API across iOS & Android



Xamarin University

Getting Started

Xamarin University

developer.salesforce.com

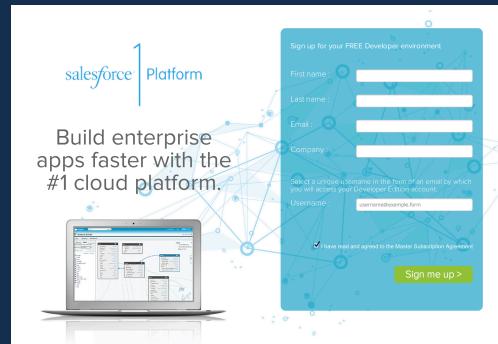
- Sign-up for a free account
- Configure a 'connected app'
- Run Salesforce SDK sample

Xamarin University



Sign-up

- It's easy for anyone to sign up as a developer
- If your organization already uses Salesforce, use a different email address for playing around

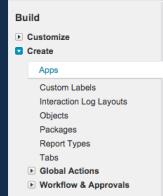


<http://developer.salesforce.com/>

Xamarin University

Connected App

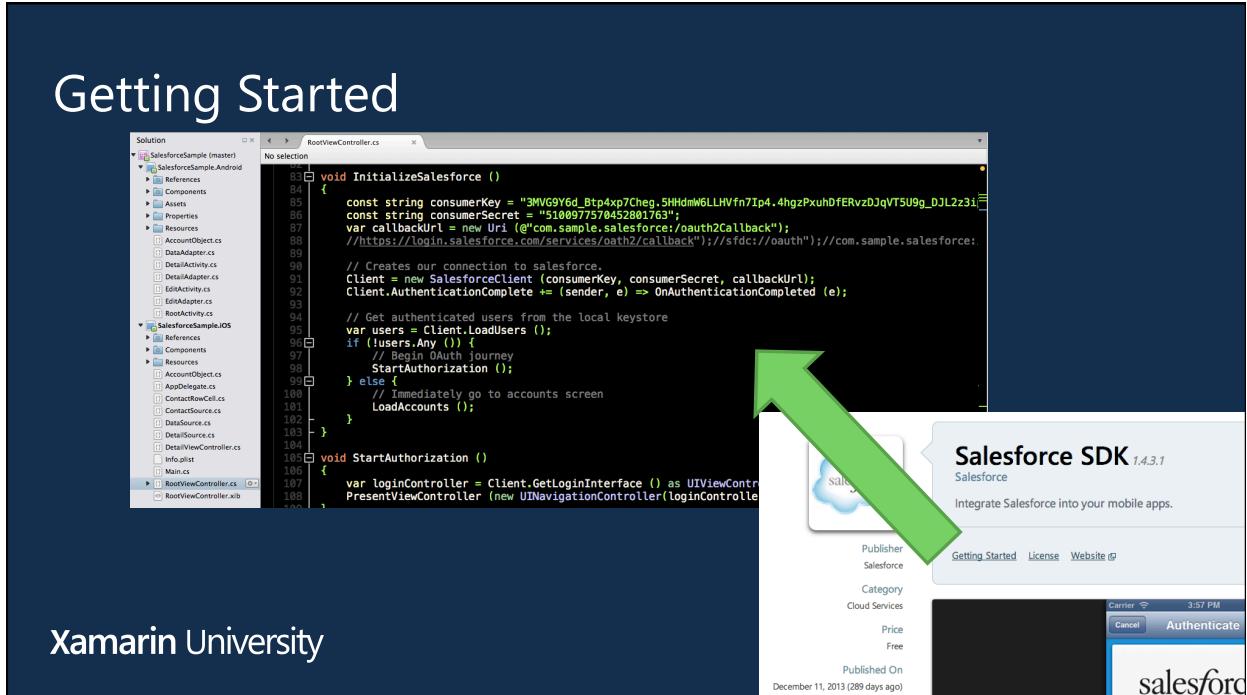
- Create new Connected App



A screenshot of the Salesforce 'Basic Information' screen for creating a new Connected App. The app name is set to 'XamEvolve14'. The 'Save' and 'Cancel' buttons are at the top right. The form includes fields for API Name, Contact Email, Contact Phone, Logo Image URL, Icon URL, Info URL, and Description. A large green arrow points from the bottom left towards the 'API (Enable OAuth Settings)' section at the bottom of the screen.

Connected App Name	XamEvolve14
API Name	XamEvolve14
Contact Email	craig.dunn@gmail.com
Contact Phone	
Logo Image URL	(Upload logo image or Choose one of our sample logos)
Icon URL	(Choose one of our sample logos)
Info URL	
Description	Welcome to the Salesforce SDK

API (Enable OAuth Settings)
Enable OAuth Settings



Getting Started
Connected App settings

API (Enable OAuth Settings)

Consumer Key	3MVG9Y6d_Btp4xp7Cheg.5HHdmW6LLHFn7Ip4.4hg2PxuhDfERvzDjqVT5U9g_DJL2z3ipo9YXNlxdIwmEhb	Consumer Secret	5100977570452801763
Selected OAuth Scopes	Access your basic information (id, profile, email, address, phone) Access and manage your data (api) Perform requests on your behalf at any time (refresh_token, offline_access)	Callback URL	com.sample.salesforce:/oauth2Callback

```
const string consumerKey = "3MVG9Y6d_Btp4xp7Cheg.5HHdmW6L";
const string consumerSecret = "5100977570452801763";
var callbackUrl =
  new Uri ("com.sample.salesforce:/oauth2Callback");
```

Xamarin University

Connected App

Android and iOS configuration

The screenshot shows the configuration interface for a Connected App. On the left, there are icons for Android (green) and iOS (pink). The main area has two sections: 'Required permissions' for Android and 'URL Types' for iOS.

Required permissions (Android):

- InternalSystemWindow
- Internet
- KillBackgroundProcesses
- ManageAccounts
- ManageAppTokens
- MasterClear

URL Types (iOS):

Identifier: com.sample.salesforce URL Schemes: com.sample.salesforce
Icon: (58x58) Role: None

Xamarin University

Getting Started

Connect and authenticate

```
// Creates our connection to salesforce.  
Client = new SalesforceClient (consumerKey, consumerSecret, callbackUrl);  
  
// Get authenticated users from the local keystore  
var users = Client.LoadUsers ();  
  
if (!users.Any ())  
{  
    client.AuthenticationComplete += (sender, e) => OnAuthenticationComplete (e);  
  
    // Starts the Salesforce login process.  
    var loginUI = client.GetLoginInterface ();  
    DisplayThe (loginUI); // PLATFORM SPECIFIC !
```

Xamarin University

Getting Started

Load Account data

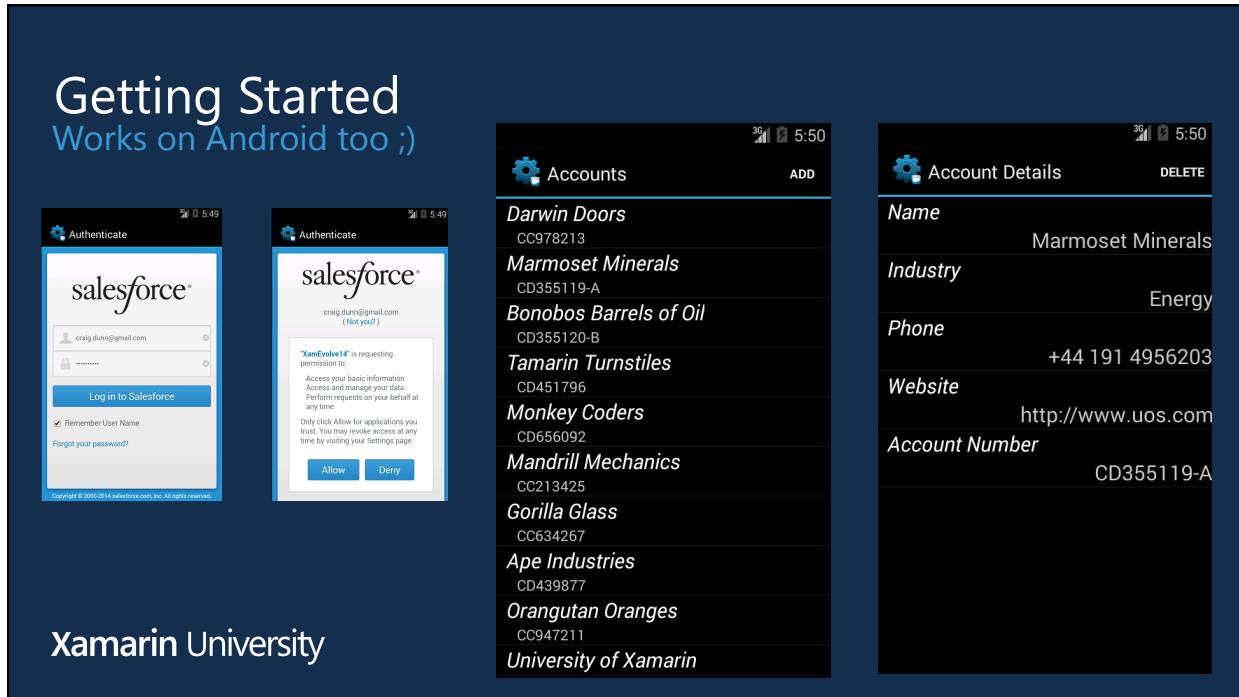
```
    }
else
{
    // We're ready to fetch some data!
    // Let's grab some sales accounts to display.
    IEnumerable<SObject> results =
        await client.ReadAsync (
            "SELECT Id, Name, AccountNumber FROM Account");

    DoSomethingAmazingWith (results); // SObjects
}
```

Xamarin University

Demo

Xamarin University



Customization

- Xamarin.Forms
- Custom SObjects & Queries
- Editing
- Search



Xamarin University

Xamarin.Forms

Using a component with Xamarin.Forms

- Xamarin.Forms
 - iOS & Android
- Shared Project
 - #if directives for OAuth
- Use the same credentials as the previous sample

The left screenshot shows the 'Accounts' screen with a list of accounts. The right screenshot shows a detailed view of the account 'Tamarin Turnstiles'.

Accounts Screen Data:

Name	Industry	Modified
Darwin Doors	General	2014-09-26 07:36:30 PM
Marmoset Minerals	Electronics	
Bonobos Barrels of Oil		
Tamarin Turnstiles		
Ape Industries		
Orangutan Oranges		
University of Gorilla Glass		
Gibbon Gas		

Account Detail Screen Data:

General
Name: Tamarin Turnstiles
Industry: Electronics
Modified: 2014-09-26 07:36:30 PM
Contact
Phone: (512) 757-6000
Website
Website: http://edgecomm.com

Xamarin University

Queries

Getting data out of Salesforce with SOQL

- SELECT Id in queries "SELECT Id, Name, ... "
- FROM objects in Salesforce "SELECT Id, Name, ... FROM Account"
- WHERE, IN, LIKE '%' for filtering "SELECT Id, Name, ... FROM Account WHERE Id = 'asdf' "
- ORDER BY, GROUP BY, HAVING
- LIMIT, OFFSET for paging

Xamarin University

Queries

Getting data out of Salesforce

- Client.QueryAsync

```
IEnumerable<SObject> response = await Client.QueryAsync  
    (@"SELECT Id, Name, AccountNumber, Phone, Website,  
     Industry, LastModifiedDate FROM Account");  
  
response.Select (s => s.As<AccountObject> ()).ToList();
```

Xamarin University

Custom SObjects

Creating SObject classes

- Subclass SObject
 - ResourceName getter
 - PreparingForUpdate event
 - GetOption and SetOption

```
public class ProductObject : SObject
{
    public ProductObject ()
    {
        this.PreparingUpdateRequest += PreparingUpdateRequest;
    }
    public override string ResourceName {
        get { return "Product2"; }
    }
    public string Name
    {
        get { return GetOption ("Name"); }
        set { SetOption ("Name", value); }
    }
    // remaining fields...
```

Xamarin University

Editing SObjects

Modifying SObjects on the server

```
// Create
string newId = await Client.CreateAsync (mySObject);

// Update
await Client.UpdateAsync (mySObject);

// Delete
bool wasDeleted = await Client.DeleteAsync (mySObject);
```

Xamarin University



Editing SObjects
Modifying SObjects on the server

Carrier 3:15 PM Products

C# shirt (green)
CSGREEN

C# shirt (navy)
CSNAVY

C# shirt (light blue)
CSBLUE

F# shirt (yellow)
FSYELLOW

F# shirt (red)
FSRED

F# shirt (purple)
FSPURPLE

Monkey (black)
MONKEYBLACK

Monkey (brown)
MONKEYBROWN

Monkey (tamarin)
MONKEYTAMARIN

Banana
BANANA

Accounts Products

Carrier 3:15 PM < Products

C# shirt (green)
CSGREEN

it's not easy

Save

Accounts Products

Xamarin University

Search

Search using SOSL

- FIND using {} to contain the search term
"FIND {query}"
- IN Salesforce-defined groups
"FIND {query} IN NAME FIELDS"
- RETURNING specifies objects and optionally fields
"FIND {query}
IN NAME FIELDS
RETURNING Account (Id, Name, AccountNumber,
Website, Industry, LastModifiedDate)"
- * and ? wildcards

Xamarin University

Search

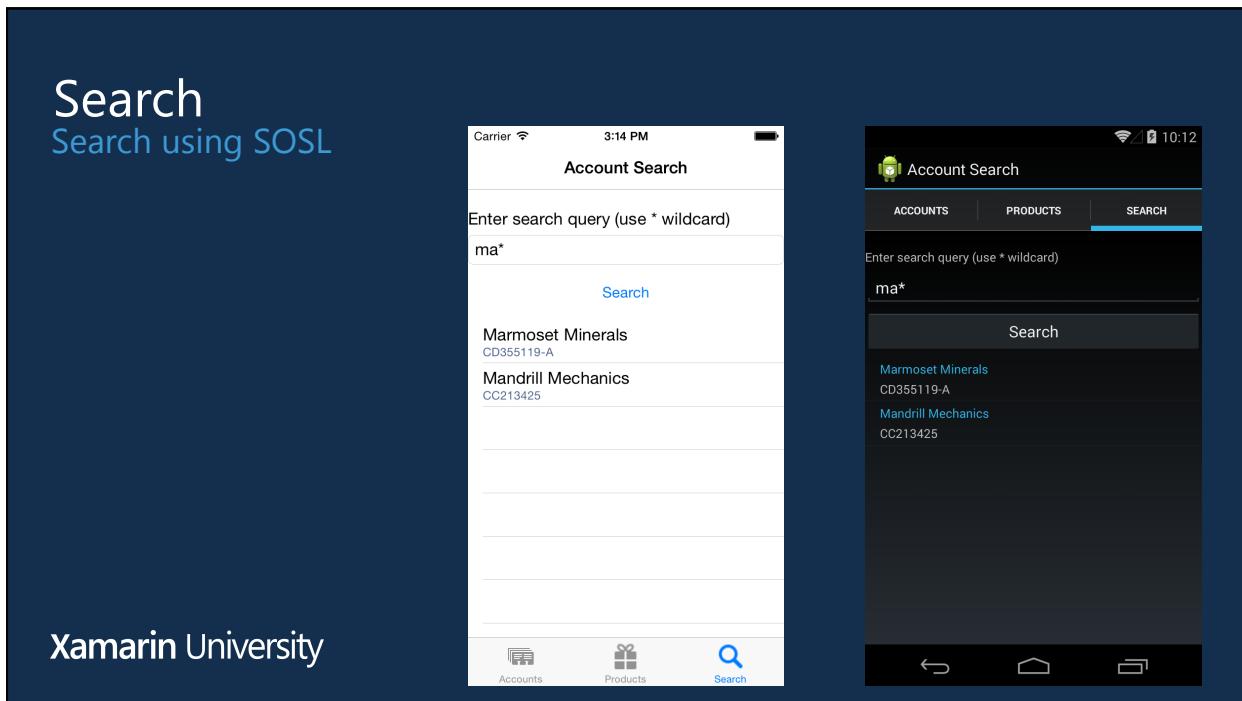
Search using SOSL

```
var sosl =
    " FIND {" + query + "} "+
    " IN NAME FIELDS "+
    " RETURNING Account ";
IEnumerable<SearchResult> searchResponse =
    await Client.SearchAsync (sosl);
// HACK: parse out all the Id values into a string

// now pull the 'found' SObjects
var soql =
    "SELECT Id, Name, AccountNumber, Phone, Website,
    "FROM Account "+
    "WHERE Id IN (" + ids + ")";

IEnumerable<SObject> queryResponse =
    await Client.QueryAsync (soql);
```

Xamarin University



Summary

- Salesforce is easily integrated into Xamarin apps
- Custom SObjects are easy to create, edit, save and search
- Works in Xamarin.Forms!

Xamarin University

Xamarin Evolve 2014

Integrating with Salesforce

Craig Dunn
craig@xamarin.com

