force.com™
cloud platform

Chatter Workbook

Learn How to Build Cloud Apps with Chatter

30-Minute Developer Tutorials

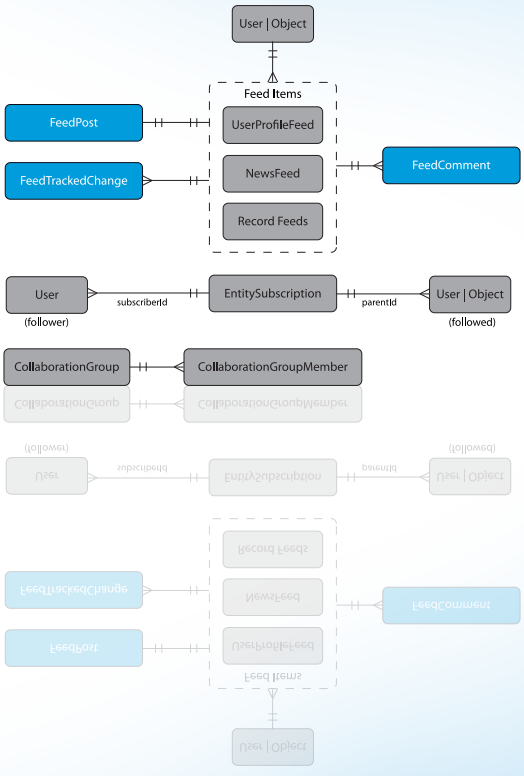


Table of Contents

About the Chatter Workbook

Salesforce Chatter adds a rich suite of collaboration features to any application built on Force.com, allowing you to create applications that transform the way people interact with apps, data, and each other. The Chatter Workbook is an introduction to the major components of Salesforce Chatter, as well as to how you can develop your own applications using Chatter features.

Before You Begin

You will need a Force.com environment that supports both Chatter and Force.com development. These tutorials are designed to work with a Force.com Developer Edition environment, which you can get for free at http://developer.force.com/join.

If you want to complete Tutorial 3, you will need a Java IDE of some kind; we recommend Eclipse, or the Force.com IDE. If you want to complete Tutorial 4, you'll need Microsoft Visual Studio or the free Microsoft Visual Studio Express.

How is the Workbook Organized?

The four tutorials can be completed in any order, though we recommend completing Tutorial 1 and Tutorial 2 first. If you're interested in creating a client that integrates with Chatter, choose between Tutorial 3 and Tutorial 4, depending on your language preference.

- Tutorial 1 explores the fundamentals. It walks you through using the standard Salesforce Chatter application, which introduces you to many of the basic underlying concepts, such as posts, feeds, groups, and comments.
- Tutorial 2 walks you through building a very simple Chatter application on Force.com. You'll learn how to interact with the data model that supports Chatter, which is the starting point of any application built with Chatter.
- Tutorial 3 shows how you can build an external Java client that interacts with the Chatter data model. The application is functionally identical to the application built in Tutorial 2, except that it's built in Java and integrates remotely with a Force.com environment.
- Tutorial 4 is identical to Tutorial 3, although it uses Microsoft .NET as the client implementation language.

Tell Me More....

At the end of each step, there is an optional Tell Me More section. If you like to do things quickly, move on to the next step. However, if you're a smell-the-roses type, there's a lot of useful information here.

- To continue exploring Chatter development, check out the Chatter Cheat Sheet at `http://developer.force.com/cheatsheets`.
- To learn more about Force.com and to access a rich set of resources, visit Developer Force at `http://developer.force.com`.
- For a gentle introduction to developing on Force.com, see the companion Force.com Workbook at `http://developer.force.com/workbook`.

Tutorial #1: Orientation and Setup

Level: Beginner; **Duration:** 25 minutes

Salesforce Chatter provides a rich suite of features, including user profiles, status updates, comments, groups, and feeds. You can use these features to add a collaborative and social dimension to your Force.com applications. Chatter also lets data records play a part in this collaboration—so people not only follow other people, but also data that's important to them. In effect, your applications can bring people and data closer to each other.

In this tutorial you will explore the basic Chatter functionality. Along the way you'll learn about users, posts, status updates, comments, feeds, following users and data, and groups. You'll also learn the names of the objects that model these features in the Chatter data model. All of these concepts are used in later tutorials. There's no programming involved yet, just configuration.

Prerequisite

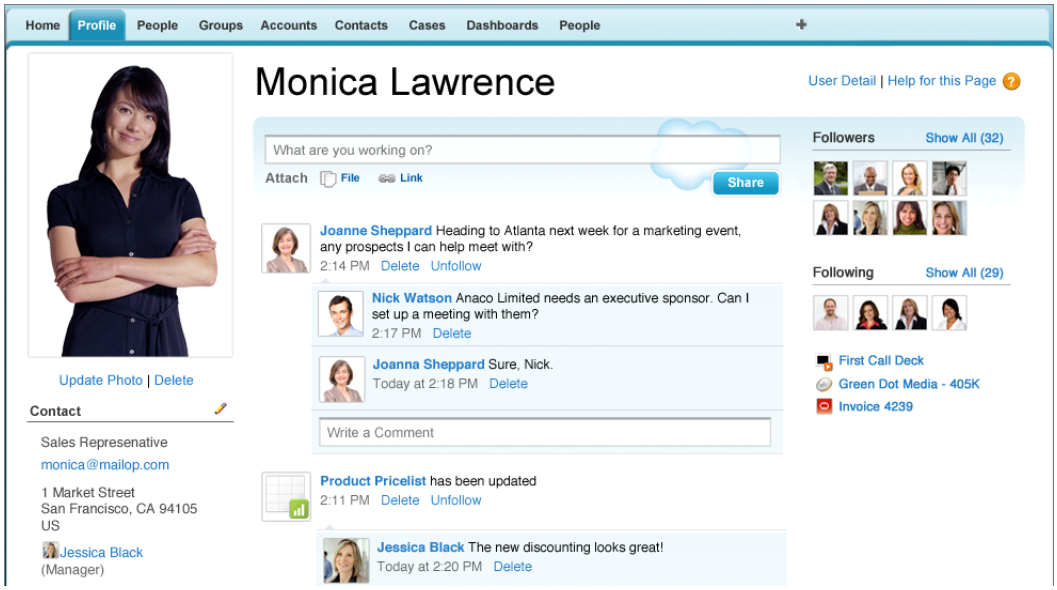
Salesforce Chatter

This workbook requires an environment with Salesforce Chatter enabled. We recommend using a free Developer Edition environment. To sign up, go to `http://developer.force.com/join`.

Step 1: Set Up Your Chatter Profile

In any collaboration environment, it's important to configure your profile so that other users can easily identify and find you. The first step is to log in, find the Salesforce Chatter application, update your profile, and create your first status update.

1. In your browser go to `https://login.salesforce.com`.
2. Enter your username (in the form of an email address) and password.
3. In the top right-hand corner, select **Salesforce Chatter** from the drop-down list of applications.
4. Select the **Profile** tab. It displays posts and record updates that you have made, and posts that other people have made on your profile.



5. Click the pencil icon in the **About Me** area and add your role and interests to the description. Other people can use this description to determine if they want to follow you. Click **Save**.

6. Optionally, select **Update Photo** and upload a photograph of yourself.
7. Finally, make a new status update. Click in the user status field and replace **What are you working on?** with some text, such as `My first chatter post.` Click **Share**.

Congratulations, you have just set up your user profile and made your first status update. In Salesforce Chatter, you can follow data, as well as other users. In the next step you'll follow some data, and in a later step you'll create a user to follow.

Tell Me More....

- The **Profile** tab also lists the users that follow you, as well as users and data records that you are following.
- A post on your user profile has a `UserStatus` type. When you make posts, you have the ability to attach a file, or attach a link. A post on someone else's profile is a `TextPost` type if you don't select these other options when posting. If your post contains a file, it is a `ContentPost`, or if it contains a link, a `LinkPost`. You can also leave a comment on a post, which is stored in the `FeedComment` object.

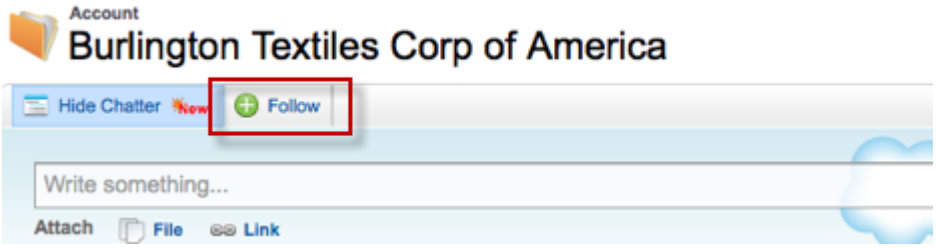
Step 2: Follow Data

In this step, you're going to follow a data record to see how changes to the record are reflected in your Chatter feed. Before you can follow records, you need to enable feed tracking.

1. Click **Setup ➤ Customize ➤ Chatter ➤ Feed Tracking**.
2. Select the **Account** object. Selecting an object displays its tracked fields.
3. Enable the checkbox in the **Employees** field and click **Save**.

To see the effects in action, you need to follow an Account record and modify the Employees field.

1. Select the **Sales** application in the drop-down list.
2. Click the **Accounts** tab.
3. Select **All Accounts** from the view filter, then select any account and click **Follow**.



4. You can make a post to this record, and anyone who follows the record will see the post in their Chatter feed on the Home tab.
5. Now click **Edit** to edit the record.
6. Navigate to the **Employees** field, and insert `200`. Click **Save**.

When you modify a tracked field, all followers see the change you made. To see this:

1. Select the Salesforce Chatter application from the drop-down list.
2. Click the **Home** tab. This tab displays the `NewsFeed` — a list of all posts and updates by users and records you follow, or groups of which you are a member. You'll see a new update, this time from the Account record that you followed, indicating that a change was made to the Employees field.

Chatter lets you follow data, as well as have conversations around a particular data record. Each data record in the default user interface now displays the related Chatter posts (called a record feed), while also providing an indication of who else is following the data.

Tell Me More....

- You can follow both standard objects, such as Accounts, as well as any custom object you create, in exactly the same manner.
- Chatter always indicates the users and records you follow on your Profile tab. Because you now follow a record, the record's name is included in your list of followers.
- Chatter posts can be made by people or by records. Chatter posts by records have the type `TrackedChanges`. Together with `UserStatus`, `TextPosts`, `ContentPosts`, and `LinkPosts`, these cover the various types of posts that can be made in Salesforce Chatter.

Step 3: Follow Users

Chatter is a collaborative platform, so you'll need more than one user if you want to collaborate. In this step, you create a new user, log in as that user, follow the original user and comment on their status update. You need to have your email address handy as user creation involves receiving an email with new user details.

1. Click **Setup ➤ Manage Users ➤ Users** and then click **New User**.
2. Enter the following data in the fields provided:
   - First Name: `Joe`
   - Last Name: `Blogs`
   - Alias: `jblogs`
   - Email: enter a valid email address you can check
   - Username: choose something unique in the form of an email address, such as adding `chatter-` in front of your email address

     Note: A username must be unique and in the form of an email address. You can have multiple usernames that all use the same email address. For example, if your email address is mary@mydomain.com, you could have a few different logins for different purposes, such as chatter-mary@mydomain.com, admin-mary@mydomain.com, standard-user-mary@anotherdomain.org, etc. All of these users have the same actual email address — mary@mydomain.com.

   - User License: select **Salesforce** (not Salesforce Platform)
   - Profile: select **System Administrator**
3. Click **Save** and then click **Logout**.

You'll receive an email indicating the username you selected, and a temporary password. Log in as Joe Blogs:

1. Navigate to `https://login.salesforce.com/`
2. Enter Joe Blogs' username and temporary password.
3. At the prompt, enter a new password and security questions.

Now you're ready to work as Joe Blogs.

1. Select the **Salesforce Chatter** application.
2. Click the **People** tab.
3. Find your profile (the first one you created) and click **Follow**.

4. Click on your name. The list of posts is the `UserProfileFeed`. Find the post you made, click the **Comment** link and type some text. Click **Comment** to save.
5. Click **Logout**.

Now log back in as yourself and follow Joe:

1. Navigate to `https://login.salesforce.com/` and log in with your credentials.
2. Select the Salesforce Chatter app.
3. Click the **Profile** tab. The page now indicates that Joe Blogs is following you, and you can see the comment he left.
4. Select Joe Blogs, by clicking on his name, or image. Click **Follow**. Now you're following Joe as well.

Tell Me More....

- During this step, you probably noticed there are two ways to follow a user — either by looking them up in the People tab, or by clicking **Follow** when viewing their profile.
- If you look at your email inbox, you'll notice an email from Salesforce Chatter informing you that Joe Blogs left a comment on your post. You can toggle the email notifications by going to **Setup ➤ Customize ➤ Chatter ➤ Settings** and modifying the **Email Notification Settings** option.

Step 4: Create a Chatter Group

In a collaboration environment, people often form groups to share knowledge or accomplish a task. In this step, you create and join a Chatter group.

1. Click the **Groups** tab.
2. Click **New Group**.
3. Enter a name for the group, and optionally, a description and click **Save**.
4. Click **Add/Remove** and add Joe Blogs to the group.
5. Now post an update to the group. In the field that displays "Write something..." type `My first group update!` and click **Share**. Everyone in the group (you and Joe Blogs) will see that update.

Tell Me More....

- The `CollaborationGroupFeed` holds all updates to groups.
- Each group member sees an update from the group in their Chatter feed on the Home tab (the `NewsFeed`).
- Groups can be either public or private. If you create a public group, then anyone can view the updates and join. If you create a private group, then you must explicitly approve members before they can join.

Summary

In this tutorial, you learned how to create a Chatter profile, follow users, leave comments, and create groups. This basic Chatter functionality introduced you to the different types of posts that you can create in Chatter.

While you worked through this first tutorial, you might have noticed a number of different views on all the posts - we call these views *feeds*. Your Profile tab lists posts or tracked changes that you've made, or that target you. This is the `UserProfileFeed`. A group has a list of updates that are made to the group, stored in the `CollaborationGroupFeed`. Each record of an object also displays a feed of posts to that record, or tracked changes made to the record, called a record feed. Finally, the Home tab displays your updates, status updates of people you follow, updates to records you follow, and groups that you are a member of; this is called the `NewsFeed`.

Tutorial #2: Building on the Platform

Level: Intermediate; **Duration:** 30 minutes

Salesforce Chatter is built on the Force.com platform. As a result, you can extend the Chatter functionality, or enhance your own applications with Chatter features. In this tutorial, you take some of the concepts from the previous tutorial, the `NewsFeed` and status updates in particular, and build a Visualforce page that lets you programmatically update your status and display it.

Step 1: Create a Visualforce Page

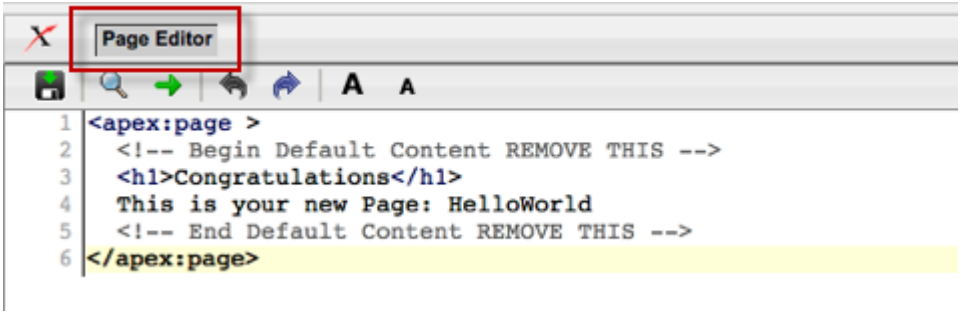
In this step, you'll create a custom user interface using Visualforce. Visualforce is the component-based user interface framework of Force.com, and you can use it to create custom pages. By the end of the tutorial, this page will list your `NewsFeed`, as well as display an input box for updating your status. But before you create the page, it's a good idea to enable Development Mode, which embeds a page editor into Visualforce pages, simplifying development.

1. Click **Setup ➤ My Personal Information ➤ Personal Information ➤ Edit**.
2. Select the **Development Mode** checkbox and click **Save**.
3. In your browser, add the text `/apex/HelloWorld` to the current URL. For example, if your current URL is `https://na1.salesforce.com/005A0000000hFv5`, the new URL will be `https://na1.salesforce.com/apex/HelloWorld`. You will get an error message: "Page HelloWorld does not exist."

   Note: Unsupported browsers do not show this error message. You can create the page by clicking **Setup ➤ Develop ➤ Pages** and clicking **New**. After you create the page, go back to the URL.

4. Click the **Create Page HelloWorld** link to create the new page.
5. Click the **Page Editor** link in the bottom left corner of the browser. The Page Editor tab displays the code and a preview of the new page (which has some default text).



6. You don't really want the heading of the page to say "Congratulations," so change the contents of the heading tag to `News Feed` and remove the comments. While you're there, add an attribute `sidebar="false"` to the `apex:page` tag. Your code will be as follows:

```
<apex:page sidebar="false">
  <h1>News Feed</h1>
</apex:page>
```

Tell Me More....

- Notice that the code for the page looks a lot like standard HTML. That's because a Visualforce page combines HTML tags with Visualforce-specific tags.
- The built-in page editor includes tag completion, as well a comprehensive component reference that indicates how each of the Visualforce tags work.

Step 2: Add User Status Update Functionality

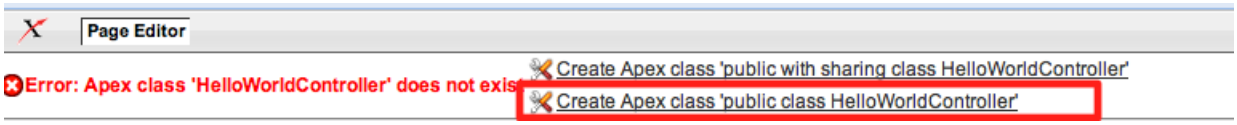
The Model-View-Controller (MVC) design pattern used by Force.com makes it easy to separate the view and its styling from the underlying database and logic. In MVC, the view (the Visualforce page) interacts with a controller. In our case, the controller is an Apex class, which exposes some functionality to the page. For example, the controller might contain the logic that executes when clicking a button. A controller also typically interacts with the model (the database)—exposing data that the view might want to display.

In this step, you modify the Visualforce page to include a text area and button, and add logic to update your Chatter status with the value of the text area after the button is clicked.

1. If the Page Editor isn't open on your Visualforce page, click **Page Editor** in the bottom left corner.
2. Modify your code to include a `HelloWorldController` custom controller by editing the `<apex:page>` tag.

```
<apex:page controller="HelloWorldController" sidebar="false">
```

3. When you click **Save**, the editor notes that the `HelloWorldController` class doesn't exist, and offers to create it for you. Click **Create Apex class 'public class HelloWorldController'**.



Note: If this message doesn't appear, you may be using an unsupported browser. To create the class, click **Setup ➤ Develop ➤ Apex Classes**, and click **New**. In the class, add the text `HelloWorldController { }`, and then return to the URL of your Visualforce page.

4. Just after the closing `</h1>` tag, add a form, together with an input text area and a button:

```
<apex:form>
     <apex:inputText value="{!status}"/>
     <apex:commandButton value="Update" action="{!go}" reRender="recent"/>
</apex:form>
```

5. Click **Save**. The editor informs you that you don't have a property called 'status' and offers to create it for you. Select **Create Apex property 'HelloWorldController.status'**.
6. The editor informs you that you don't have a method called `go()`. Click **Create Apex method 'HelloWorldController.go'**.

By now you might have noticed that a **Controller** tab has appeared, containing the code for the controller. The editor has created a lot of the code for you, and you'll also notice that you now have a form text area, together with a button when viewing the page. The generated code is below:

```
public class HelloWorldController {
   public PageReference go() {
      return null;
```

```
    }
    public String status { get; set; }
}
```

When a value is entered into the text area, the `status` property that you just created is assigned the value. When you click the **Update** button, the Visualforce page calls the `go()` method on the controller. You want this method to update the user's status with the value of the field. That's the next and final task.

1. Modify the contents of the `go()` method to query the User object and locate the currently logged-in user with `UserInfo.getUserId()`.

```
public PageReference go() {
  User u = [SELECT id, CurrentStatus FROM User WHERE id = :UserInfo.getUserId()];
  return null;
}
```

2. Now set its `CurrentStatus` field with the value found in the status property, and then update the `User` record in the database.

```
public PageReference go() {
  User u = [SELECT id, CurrentStatus FROM User WHERE id = :UserInfo.getUserId()];
  u.CurrentStatus = status;
  update u;
  return null;
}
```

Tell Me More....

- The Visualforce `PageReference` type, used in the `go()` method, determines the page that displays after the method completes. By returning `null`, you indicate that no new page displays.
- The way to update someone's Chatter status is to update the `CurrentStatus` field of their `User` record. This creates a new post of type `UserStatus`.

Step 3: Display and Query a News Feed

Currently, you must navigate to the Profile tab to verify your user status update. Instead, let's display your status on the page after each update. The Chatter feed on the Home tab displays your posts, as well as posts by those that you follow. You'll do something similar, but limit it to only display status updates, ignoring comments and so on.

1. Add a new `getRecentStatusUpdates()` method to the `HelloWorldController` that you created in the previous step. After the first line, add the following:

```
public List<NewsFeed> getRecentStatusUpdates() {
}
```

2. Now use a query to retrieve the list of `NewsFeed` items that are `UserStatus` updates. You'll want to order the posts by date, the most recent being first:

```
public List<NewsFeed> getRecentStatusUpdates() {
   List <NewsFeed> aNewsFeed =
        [SELECT  Type, CreatedDate, CreatedBy.name, FeedPost.body
         FROM NewsFeed WHERE Type='UserStatus'
         ORDER BY CreatedDate DESC LIMIT 10];
```

```
    return aNewsFeed;
}
```

3. The code now returns a list of `NewsFeed` items that are `UserStatus` updates. So far so good, but now modify the Visualforce page to display them. Click **Page Editor** to show the panel displaying your Visualforce markup.
4. Underneath the line containing `<apex:commandButton>`, insert the following:

```
<apex:outputPanel id="recent">

</apex:outputPanel>
```

5. You might have noticed that the command button you added earlier has an attribute `reRender="recent"`. This ensures that an AJAX page update is made when the button is clicked—refreshing a block on the page identified by the `"recent"` identifier. Go ahead and put your list within that block. Enter the following between the `outputPanel` tags:

```
<apex:dataList value="{!recentStatusUpdates}" var="update">
    <apex:outputText value="{!update.feedpost.body}"/>
</apex:dataList>
```

6. Click **Save**.
7. Enter some text and click the **Update** button. You'll see the page dynamically list the most recent status updates beneath the query box.

Tell Me More....

- The `<apex:dataList>` component iterates over a list, assigning each item in the list to a variable. In the code shown, it assigns each item (an instance of `NewsFeed`) to a variable named `update`. The `<apex:outputText>` component outputs values. In this case, it grabs the body of a `FeedPost` associated with the `NewsFeed` record. For user status updates, the `body` field holds the text that was posted as part of the user status update.
- The Visualforce page doesn't output all the values that are retrieved from the query. For example, `CreatedDate` and `CreatedBy.Name` are ignored. As an optional exercise, extend the Visualforce page to display these fields as well.

Step 4: Add an Automated Test

While you know the `HelloWorldController` works, automated tests are required before you can deploy the code to a production environment. In this step, you'll add a single test method to the controller to ensure that it behaves as expected. Add the test method as follows:

1. Click **Setup ➤ Develop ➤ Apex Classes**.
2. Click on the name of your class, `HelloWorldController` and click **Edit**.
3. Create a new method before the final closing brace:

```
@isTest
static void testUpdates() {
}
```

4. Between the curly braces, instantiate a new instance of your controller class:

```
HelloWorldController h = new HelloWorldController();
```

5. Assign the status property and then call the `go()` method to post the update.

```
h.status = 'hi';
h.go();
```

6. Return the list of status updates.

```
List<NewsFeed> after = h.getRecentStatusUpdates();
```

7. Now you need to test that the retrieved list has at least one element, and that the most recent update corresponds to the status update you just made. You know the most recent post is first because of the `ORDER BY` in the original query.

```
System.assert(after.size() > 0 );
System.assertEquals(h.status, after[0].feedpost.body);
```

8. Verify your finished code looks like the following and then click **Save**.

```
@isTest
static void testUpdates() {
   HelloWorldController h = new HelloWorldController();
   h.status = 'hi';
   h.go();
   List<NewsFeed> after = h.getRecentStatusUpdates();
   System.assert(after.size() > 0 );
   System.assertEquals(h.status, after[0].feedpost.body);
}
```

9. Click **Run All Tests**. If all goes well, your screen indicates that you ran one test, and that its code coverage is 100%.

Tell Me More....

- Testing is a critical part of code development. This was a very simple example. In a real-world scenario, your tests verify both positive and negative results for many different use cases.
- The `@isTest` annotation indicates that the method is a test method, and that it must be called during testing.

Summary

The data model behind Chatter can be accessed from an application on Force.com, letting you create applications that manipulate or query the data related to the Chatter functionality. In this tutorial you created logic that updated a user's status and queried their news feed. You can imagine embedding some of this logic in your own applications, or extending it using the examples from the Chatter Cheat Sheet at `http://developer.force.com/cheatsheets`.

Tutorial #3: Accessing Chatter From a Java Client

Level: Advanced; **Duration:** 30 minutes

Force.com exposes a number of APIs, one of which is the Force.com Web Services API that lets you access and manipulate data records from outside the platform, using any language that supports SOAP-based Web services. The Chatter data model, described in Tutorial 1 and used in Tutorial 2, can also be accessed in this manner. In this tutorial, you will learn how to use the Force.com Web Services API from Java. The Java application will authenticate with the Force.com platform, update your status, and display the latest items from your news feed. This tutorial is similar to Tutorial 4, which uses .NET to accomplish the same task.

Prerequisites

Java

You will need to have Eclipse or the Force.com IDE, and Java JDK 6 (1.6) installed on your computer. Some knowledge of Java is presumed.

Runtime Libraries

The Force.com Web Service Connector (WSC) is an open source code-generation tool and runtime library for accessing Force.com Web Services from Java. There are two libraries required:

- `wsc-18.jar`—A Web service client for accessing Force.com Web services.
- `partner-18.jar`—A pre-generated Java library using WSC and the Partner WSDL.

Download them from http://code.google.com/p/sfdc-wsc/downloads/list. The tutorial instructions assume these libraries are on your desktop.

Step 1: Create a Java Project

In this step, you create a new Java project in the IDE and configure your Build Path to use the new WSC libraries.

1. In Eclipse, click **New ➤ Project ➤ Java Project**.
2. Enter a project name, such as `Chatter Tutorial`.
3. Ensure that the JRE selected is 1.6.
4. Click **Finish**.

Now you need to modify your project to add the two Java libraries listed in the Prerequisites section to the build path.

1. In the navigation pane, right click the project you just created and choose **Project ➤ Properties ➤ Java Build Path**.
2. Click the **Libraries** tab.
3. Click **Add External JARs** and browse to the location.
4. Select both libraries and click **Open**.
5. Click **OK**.

Tell Me More....

A WSDL (Web Services Description Language) describes a Web service end point. The Force.com Web Services API has two flavors of WSDL:

- The Partner WSDL is generic. Data objects in your Force.com environment are not reflected in any concrete way— they all share a generic "sObject" description. This means that if you write an application that uses the partner WSDL, as you

are doing in this tutorial, it will work within anyone's Force.com environment, given the correct permissions, username and password.

- The Enterprise WSDL is strongly typed. In this WSDL, any data object you have, such as FooBar, will be reflected as an object in Java, FooBar. Of course, this means that the WSDL is specific to your environment (someone else might not have that object), and as a result, this makes the code less generic and portable, but typically more readable.

Step 2: Create a Java Application

Your application consists of a single class that contains all of the application's functionality. To begin creating this application, do the following:

1. Click **File ➤ New ➤ Class**.
2. Enter a package name of `com.developerforce.chatter`.
3. Enter a class name of `ChatterMain` and select the checkbox to create a `main()` method placeholder, **public static void main(String[] args)**.
4. Click **Finish**.
5. The new class opens in the editor. After the package declaration, add the import statements for the required libraries:

```
import com.sforce.soap.partner.*;
import com.sforce.soap.partner.sobject.*;
import com.sforce.ws.*;
import java.io.*;
```

6. After the class declaration, add the static class members to hold your Force.com login credentials and connection object from the Web Services Connector (WSC). You might need to append your security token to the end of the password.

```
static final String USERNAME = "YOUR-SALESFORCE-USERNAME";
static final String PASSWORD = "YOUR-SALESFORCE-PASSWORD-AND-TOKEN";
static PartnerConnection connection;
```

7. Replace the `main()` method in the class with the following code. This method creates a new `PartnerConnection` object (from the WSC library) with your Force.com credentials, and uses it to establish a new connection to the platform. The code has two method calls that are commented out. You'll write those methods in later steps and then uncomment the method calls here.

```
public static void main(String[] args) throws Exception {
    ConnectorConfig config = new ConnectorConfig();
    config.setUsername(USERNAME);
    config.setPassword(PASSWORD);
    connection = Connector.newConnection(config);
    System.out.println("Hello " + connection.getUserInfo().getUserFullName());
    //displayNewsFeed();
    //updateStatus();
}
```

8. Verify that your code looks like the following and then click **Save**.

```
package com.developerforce.chatter;

import com.sforce.soap.partner.*;
import com.sforce.soap.partner.sobject.*;
import com.sforce.ws.*;
import java.io.*;
```

```
public class ChatterMain {
   static final String USERNAME = "jmountjoy@devchatteradmin.org";
   static final String PASSWORD = "password1Ud4VCrSU5YtYf7vbg7NaT1u9j";
   static PartnerConnection connection;

   public static void main(String[] args) throws Exception {
      ConnectorConfig config = new ConnectorConfig();
      config.setUsername(USERNAME);
      config.setPassword(PASSWORD);
      connection = Connector.newConnection(config);
      System.out.println("Hello " + connection.getUserInfo().getUserFullName());
      //displayNewsFeed();
      //updateStatus();
   }
}
```

Tell Me More....

Typically, the first thing you do when interacting with the platform is establish a connection with a Force.com environment. The connection object itself is critical to any further interaction, and contains all the methods you need to interact with the platform. The code in this step uses the `getUserInfo()` method to extract information about the authenticated user. Other methods exist for querying, inserting and updating records, as well as retrieving metadata about objects in your Force.com environment.

Step 3: Run the Application

Now it is time to test your application. If the application connects to your organization successfully, you will see your user's full name printed out.

1. Click **Run ➤ Run As ➤ Java Application**.
2. Click the **Console** tab at the bottom.
3. If the application connects successfully you see, "Hello" followed by your full name.

Tell Me More....

Force.com has a number of built-in security measures— one of which ensures that you need additional security when logging in from a new or untrusted location. If you receive an INVALID_LOGIN exception, make sure your username and password are correct. If they are, you might need to append a security token to your password. To obtain your security token, log into your organization using a Web browser, and navigate to **Setup ➤ My Personal Information ➤ Reset My Security Token**. You will receive an email that contains the string to append to your password.

Step 4: Update Your Status

In this step, you modify the application to update your status with a new method, `updateStatus()`.

1. First, create the `updateStatus()` method:

```
private static void updateStatus() throws Exception {
}
```

2. Between the curly braces, ask the user for a status update:

```
InputStreamReader converter = new InputStreamReader(System.in);
BufferedReader in = new BufferedReader(converter);
System.out.println("What are you working on?");
```

3. Create the `user` sObject to hold the new status update:

```
SObject user = new SObject();
user.setType("User");
user.setId(connection.getUserInfo().getUserId());
user.setField("CurrentStatus", in.readLine());
```

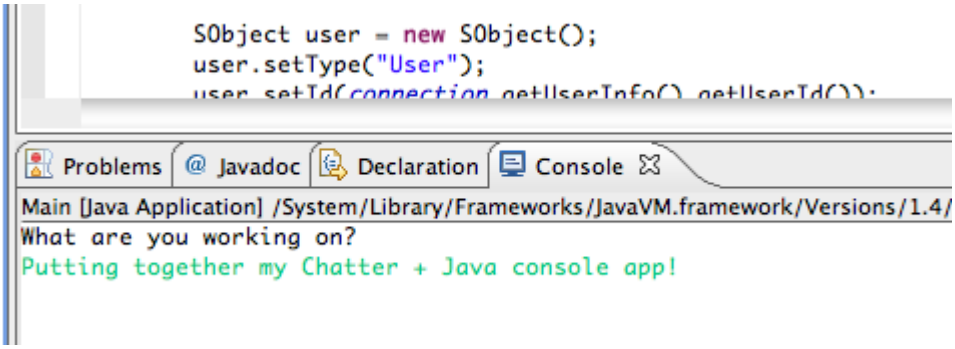
4. Call the `update()` method to update the record:

```
SaveResult[] results = connection.update(new SObject[] { user });
```

5. Check for errors:

```
if (!results[0].isSuccess()) {
   System.out.println("Error updating user status: "
     + results[0].getErrors()[0].getMessage());
   }
   else {
      System.out.println("User status successfully updated.");
   }
}
```

6. Uncomment `updateStatus();` in the `main()` method.
7. Run the application by following Step 3: Run the Application on page 14.
8. This time you'll be prompted to write something. After pressing Enter, your Chatter status will be updated.



Tell Me More....

Because you're using the Partner WSDL, you have to use a generic type called SObject to represent all of the actual objects you have in your Force.com environment. As a result, you have to tell the SObject what type it represents - in the code above this is done with the `setType()` method call. You effectively create a `User` object, provide its Id, and set a field on that object called `CurrentStatus`. This is exactly what you did in Tutorial 2 to update the user status.

Step 5: Retrieve the News Feed

In this step, you query your `NewsFeed` and display the results.

1. Add a new `displayNewsFeed()` method.

```
private static void displayNewsFeed() throws Exception {
   System.out.println("Querying for status updates...");

   QueryResult queryResults = connection.query
     ("SELECT Id, Type, CreatedDate, CreatedBy.name, FeedPost.body
       FROM NewsFeed WHERE Type= 'UserStatus'
       ORDER BY CreatedDate DESC, ID DESC LIMIT 10");

   if (queryResults.getSize() > 0) {
       for (SObject s : queryResults.getRecords()) {
          System.out.println(s.getChild("FeedPost").getField("Body") );
       }
   }
}
```

2. Now uncomment the `displayNewsFeed()` call in the `main()` method.
3. Run the application again and you should see a list of status updates:

```
<terminated> Main [Java Application] /System/Library/Frameworks/JavaVM.framework/V
What are you working on?
One more update!
User status successfully updated.
Querying for status updates...
Id: 0D5A00000004zXuKAI - One more update!
Id: 0D5A00000004zWiKAI - Putting together my Chatter + Java console
```

Tell Me More....

- The code performs a query on the `NewsFeed` object—it's the same query that you used in Tutorial #2: Building on the Platform on page 7. After executing the query, a `QueryResults` object is instantiated with the results of the query. If there are any results, the code iterates over the records that were returned, printing out the body of each feed post.
- Because you're using the Partner WSDL, all record results are of the generic type sObject. As a result, Java can't know which fields are in the object. By using the `getChild()` and `getField()` methods, you can navigate through the generic object and retrieve the data you require.

Summary

In this tutorial, you wrote a Java client application that integrates with the Force.com environment. Your Java application authenticates with the Force.com platform, updates a record (your `User` record, which contains your user status), and performs a query. You can explore the other methods on the connection object to learn what other actions you can perform, or consult the Chatter cheat sheet to find queries to run against the Chatter data model.

Tutorial #4: Accessing Chatter From a .NET Client

Level: Advanced; **Duration:** 30 minutes

Force.com exposes a number of APIs, such as the Force.com Web Services API, that lets you access and manipulate data records from outside the platform using any language that supports SOAP-based Web services. The Chatter data model, described in Tutorial 1 and demonstrated in Tutorial 2, can also be accessed in this manner. In this tutorial, you create a .NET client that authenticates with the Force.com platform, updates your status, and displays the latest items from your news feed. This tutorial is similar to Tutorial 3, which uses Java to accomplish the same task.

Prerequisites

Visual Studio

Familiarity with Microsoft's Visual Studio IDE is presumed.

Step 1: Download the Partner WSDL

Before you begin development, you need to download the Partner WSDL. In the Partner WSDL, data objects in your Force.com environment are not reflected in any concrete way— they all share a generic "sObject" description. This means that if you write an application that uses the partner WSDL, as you are doing in this tutorial, it will work within anyone's Force.com environment, given the correct permissions, username and password.

1. In a Web browser, log into your Force.com environment.
2. Click **Setup ➤ Develop ➤ Generate Partner WSDL**.
3. Save the file to your desktop.

Tell Me More....

For more information on the WSDL, see the Tell Me More.... section in Tutorial #3: Accessing Chatter From a Java Client, Step 1: Create a Java Project on page 12.

Step 2: Create a Console Project in Visual Studio

In this step you create a new C# Console Application and then add the Partner WSDL as a reference.

1. Select **File ➤ New Project**
2. From the templates, choose **Console Application**.
3. Name the project `ChatterConsole` and click **OK**.
4. In the Solution Explorer, right-click **References** and select **Add Service Reference**.
5. Click **Advanced**.
6. In the Service Reference Settings dialog, click **Add Web Reference**.
7. Enter the path to the Partner WSDL.
8. In the Web reference name field, enter `PartnerWSDL` and click **Add Reference**.

Step 3: Create a Login Method

Your application consists of a single class that contains all of the application's functionality. To begin creating this application, do the following:

1. To use the Partner WSDL, add the following `using` statement:

```
using ChatterConsole.PartnerWSDL;
```

2. Next, you need an `SforceService` object to use in all the methods created in this tutorial. You'll also enter your username and password so you don't have to keep re-entering it every time you run the program. If you have a security token, append it to your password. Add the following lines:

```
private SforceService binding;
private static string USERNAME = "YOUR-SALESFORCE-USERNAME";
private static string PASSWORD = "YOUR-SALESFORCE-PASSWORD-AND-TOKEN";
```

3. Add a method signature for the `login()` method.

```
private bool login()
```

4. Next create the service object:

```
binding = new SforceService();
```

5. Try the login, and if successful, go on to bind the `SforceService` object to the correct endpoint, set up the header and session ID. If the login fails, trap and report the error.

```
LoginResult lr;
try {
   Console.WriteLine("Now logging in...");
   lr = binding.login(USERNAME, PASSWORD);

   binding.Url = lr.serverUrl;
   binding.SessionHeaderValue = new SessionHeader();
   binding.SessionHeaderValue.sessionId = lr.sessionId;
   return true;
   }
catch (Exception e){
   // report the fault to the console
   Console.WriteLine(e.Message);
   return false;
}
```

6. This is a good place to pause and make sure that everything works as expected. In the `Main()` that was automatically generated, insert the following code to instantiate the class and then run the login. Also, add two lines at the end of your `Main()` so that the application has to wait for input to close, allowing you to leisurely examine the results of your work:

```
Program chatProgram = new Program();
if (chatProgram.login()) {
   Console.Write("Success!");
   // updateStatus();
   // queryForUpdates();
   Console.WriteLine("\n\nHit enter to exit...");
   Console.ReadLine();
```

```
}  else {
   Console.Write("I couldn't log in.");
}
```

7. Now build and run your project. There should be a message in your console stating `"Success!"`

Tell Me More....

- Typically, the first thing you do when interacting with the platform is establish a connection with a Force.com environment. The connection object itself is critical to any further interaction, and contains all the methods you need to interact with the platform. Other methods exist for querying, inserting and updating records, as well as retrieving metadata about objects in your Force.com environment.
- If you had trouble connecting, see the Tell Me More.... section in Tutorial #3: Accessing Chatter From a Java Client, Step 3: Run the Application on page 14.

Step 4: Create a Status Update Method

In this step, you create a method that asks the user to update their status after logging in.

1. Create a new method for the status update as follows.

```
private void updateStatus() {
}
```

2. Start by asking the user for a status update:

```
Console.Write("\n\nWhat are you working on?\n");
string newStatus = Console.ReadLine();
```

3. Construct an sObject, and give it the correct ID and type:

```
sObject user = new sObject();
user.Id = binding.getUserInfo().userId;
user.type = "User";
```

4. Now create an XML element to store the `CurrentStatus` value.

```
System.Xml.XmlDocument doc = new System.Xml.XmlDocument();
System.Xml.XmlElement statusElement = doc.CreateElement("CurrentStatus");
statusElement.InnerText = newStatus;
```

5. Then add the element to the `user` object.

```
user.Any = new System.Xml.XmlElement[] { statusElement };
```

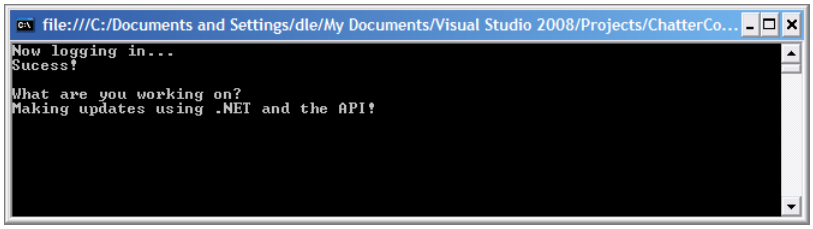
6. Now have Force.com update the record:

```
SaveResult[] results = binding.update(new sObject[] { user });
```

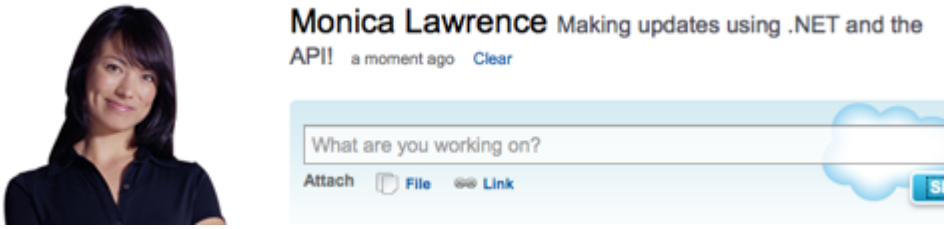
7. Finally, send the results to the console:

```
for (int j = 0; j < results.Length; j++) {
   if (results[j].success)
      Console.WriteLine("Status updated.\n");
   else
      Console.WriteLine("Error updating: " + results[j].errors[0].message + "\n");
}
```

8. In your `Main()` method, add in the call to `updateStatus()` after a successful login.
9. When you run the program again, notice the output in the console.



10. You can verify your status by logging into your Force.com environment and navigating to your Profile tab in the Salesforce Chatter application.



Tell Me More....

The `Any` field of the sObject is an array of `XmlElements` containing information about the fields belonging to the object. When constructing an `XmlElement`, give it the name of the field and fill in its value. The Java code in Tutorial 3 does this slightly differently because the WSC library that it uses hides a lot of this low-level manipulation.

Step 5: Query Status Updates

In this step, you query your `NewsFeed` and display the results.

1. Create another method for the query. Remember to make the call to this method from your `Main()` in the successful login code block.

```
private void queryForUpdates()
```

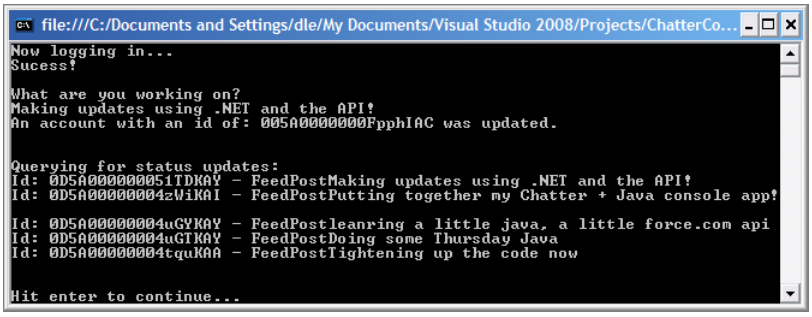
2. First, construct the query and retrieve the results using `SforceService`.

```
QueryResult qr = binding.query
    ("SELECT Id, Type, CreatedDate, CreatedBy.name, FeedPost.body
      FROM NewsFeed WHERE Type='UserStatus'
      ORDER BY CreatedDate DESC, ID DESC LIMIT 5");
```

3. Now iterate through the results to the console:

```
if (qr.size > 0){
   Console.WriteLine("\nQuerying for status updates:");
   for (int index = 0; index < qr.records.Length; index++){
      Console.WriteLine("Id: " + qr.records[index].Id + " - " +
        qr.records[index].Any[4].InnerText);
   }
}
```

4. At this point, you're ready to build and run it, press F5 and go! You should see your last five status updates in the console window.



Tell Me More....

- The code performs a query on the `NewsFeed` object—it's the same query that you used in Tutorial #2: Building on the Platform on page 7. After executing the query, a `QueryResult` object is instantiated with the results of the query. If there are any results, the code iterates over the records that were returned, printing out the body of the feed post.
- Because you're using the Parter WSDL, all records returned are of the generic type SObject. As a result, the included fields are stacked in the array returned by the `Any` field of the SObject. So you need to look at the query you originally made to know which array element to reference in order to retrieve the text string for `FeedPost.body`. In this case, the `innerText` of the `XmlElement` at the 4th array index (beginning at 0): `Any[4].InnerText`.

Summary

In this tutorial, you wrote a .NET application that integrates with the Force.com environment. Your application authenticates with the Force.com platform, updates a record (your `User` record, which contains your user status), and performs a query. You can explore the other methods on the connection object to learn what other actions you can perform, or consult the Chatter cheat sheet to find other queries to run against the Chatter data model.