Creating Secure Force.com Applications

Access Control in Force.com

Brian Soby
Platform Security Director
bsoby@salesforce.com
Involved Concepts

- General Web Application Security
- Force.com Programmatic Enforcement
- Declarative Security Configuration
General Web Application Security

Big topic.

Go to Brendan O’Connor’s talk at 3pm:

*Lock It Down: Secure Your App*
Declarative Security Configuration (For Access Control)

- CRUD (Create-Read-Update-Delete)
- FLS (Field Level Security)
- Permissions
- Sharing
### Sharing vs CRUD vs FLS

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Title</th>
<th>Mailing Street</th>
<th>Mailing City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lenny</td>
<td>Howe</td>
<td>Lawyer</td>
<td>2320 Sand Hill Road</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>Aidan</td>
<td>Plante</td>
<td>Sr. Sales Director</td>
<td>12 Spear Street</td>
<td>San Francisco</td>
</tr>
<tr>
<td>Laurie</td>
<td>Darby</td>
<td>VP Sales</td>
<td>22 5th Ave</td>
<td>New York</td>
</tr>
<tr>
<td>Jim</td>
<td>Steele</td>
<td></td>
<td>421 Main Street</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>Gavin</td>
<td>Fontana</td>
<td></td>
<td>P.O. Box 420</td>
<td>Minneapolis</td>
</tr>
<tr>
<td>Andy</td>
<td>Rother</td>
<td></td>
<td>1132 Westchester Ave</td>
<td>White Plains</td>
</tr>
<tr>
<td>Jason</td>
<td>Price</td>
<td>Sr. Sales Director</td>
<td>1445 Lawton Lane</td>
<td></td>
</tr>
<tr>
<td>Felix</td>
<td>Frye</td>
<td></td>
<td>1111 Westran Blvd</td>
<td>White Plains</td>
</tr>
<tr>
<td>Sophie</td>
<td>Kostos</td>
<td>Purchasing Rep</td>
<td>1515 Broadway</td>
<td>New York</td>
</tr>
<tr>
<td>Paul</td>
<td>Huxtable</td>
<td>District Manager</td>
<td>P.O. Box 3-740</td>
<td>Shamburg</td>
</tr>
<tr>
<td>Francis</td>
<td>Buchner</td>
<td>Account Manager</td>
<td>P.O. Box A-455</td>
<td>Shamburg</td>
</tr>
<tr>
<td>Mandy</td>
<td>Hall</td>
<td>Account Executive</td>
<td>122 Chestnut Street</td>
<td>San Francisco</td>
</tr>
<tr>
<td>Jay</td>
<td>Price</td>
<td>Sales Supervisor</td>
<td>2455 Paces Ferry Road</td>
<td>Atlanta</td>
</tr>
<tr>
<td>Jack</td>
<td>Fallon</td>
<td></td>
<td>Forest Road</td>
<td>Middlesex</td>
</tr>
<tr>
<td>Rick</td>
<td>Lykor</td>
<td></td>
<td>Reisholzer Werfstrasse 38-42</td>
<td>Dusseldorf</td>
</tr>
<tr>
<td>George</td>
<td>Fiss</td>
<td>Sr. Sales Director</td>
<td>1111 Westchester Ave</td>
<td>White Plains</td>
</tr>
<tr>
<td>Robert</td>
<td>Samps</td>
<td>Account Executive</td>
<td>Postfach 2103</td>
<td>Weisbaden</td>
</tr>
<tr>
<td>Zoe</td>
<td>Kramer</td>
<td>VP of Sales</td>
<td>100 Abbott Park Rd.</td>
<td>Abboot Park</td>
</tr>
<tr>
<td>Edward</td>
<td>Samos</td>
<td>President and CEO</td>
<td>10 Mair Rd.</td>
<td>New York City</td>
</tr>
<tr>
<td>Leanne</td>
<td>Tomlin</td>
<td>VP Customer Support</td>
<td>10 Mair Rd.</td>
<td>New York</td>
</tr>
<tr>
<td>Jen</td>
<td>Jacobs</td>
<td></td>
<td>101 California Street</td>
<td>San Francisco</td>
</tr>
</tbody>
</table>

**Sharing** refers to the collaborative access control of data. **CRUD** stands for Create, Read, Update, and Delete, which are the basic operations in a database. **FLS** stands for Field-Level Security, allowing granular control over which features and data are accessible by users based on their roles and permissions.
CRUD

- Profile-based assignment
- Governs access on an object type basis
FLS

- Profile-based assignment
- Governs read/write for fields on an object type basis
Permissions

- Profile-based assignment
- Special access rights
- May take precedent over CRUD, FLS, or Sharing
Sharing

- Governs access on an per-record basis
- Defaults based on object’s “sharing model”
- Additive only
Lots of Ways to Share

- User/Owner
- Groups
- Criteria
- Apex Managed
- Role Hierarchy
- Manual record sharing
- Territory
- Account/Contact (High Volume Portal User)
Custom Settings

- Public vs Protected
Best Practices

- Object-level Modify/View All are administrative privileges, use sharing rules first
- Install objects with private sharing models
- Always assume objects have private sharing models
- Use Least Privilege in assigning sharing, CRUD, FLS, and permissions
Involved Concepts

- General Web Application Security
- Force.com Programmatic Enforcement
- Declarative Security Configuration
Force.com Programmatic Enforcement

- Sharing
- CRUD/FLS
- Permissions and Special Access
Sharing

```java
public with sharing class ContactController {
    public List<Contact> getContacts() {
        return [Select Id,Name,Email,LastName,Phone from Contact limit 20];
    }
}
```
Sharing – Split Model

```java
public with sharing class ContactController {
    public List<Contact> getContacts() {
        return [Select Id,Name,Email,LastName,Phone from Contact limit 20];
    }
}

public without sharing class ScoreGuardian {
    public Double averageScore() {
        AggregateResult[] res = [Select AVG(Score__c) aver From Contact];
        return Double.valueOf(res[0].get('aver'));
    }
}

public Double getAverageScore() {
    return new ScoreGuardian().averageScore();
}
```
Sharing – Updates and Deletes

```java
public with sharing class ContactController {
    public PageReference deleteContact() {
        delete [Select Id From Contact limit 1];
        return null;
    }
}
```
Sharing – Updates and Deletes

```java
public with sharing class ContactController {
    public PageReference deleteContact() {
        delete [Select Id From Contact limit 1];
        return null;
    }
}
```

Delete failed. First exception on row 0 with id 0038000000Tu968AAB; first error: 
INSUFFICIENT_ACCESS_OR_READONLY, insufficient access rights on object id: []

An unexpected error has occurred. Your development organization has been notified.
Sharing – Updates and Deletes

```java
public with sharing class ContactController {
    public PageReference deleteContact() {
        try {
            delete [Select Id From Contact limit 1];
        } catch (System.DmlException DMLEx) {
            // Error handling logic
        }
        return null;
    }
}
```
What happens?

```java
public with sharing class ContactController {
    public PageReference deleteContact() {
        delete [Select Id From Contact limit 1];
        return null;
    }
}
```

(assuming the user has delete rights via sharing [Full Access])
public with sharing class ContactController {

    public PageReference deleteContact() {

        if (!Schema.sObjectType.Contact.isDeletable()) {
            // Handle no-access case
            return null;
        }

        try {
            delete [Select Id From Contact limit 1];
        } catch (System.DmlException DMLEx) {
            // Error handling logic
        }

        return null;
    }
}

What happens?

global with sharing class ContactWebService {
    webservice static List<Contact> getContacts() {
        return [Select Id, Name, Email, LastName, Phone from Contact limit 20];
    }
}

(assume no FLS access either)
global with sharing class ContactWebService {
    private static List<String> getVisibleFields() {
        String[] queryFields = new String[]{};
        Map<String, Schema.SObjectField> fmap = Schema.SObjectType.Contact.fields.getMap();
        for (String fieldName : fmap.keySet()) {
            if (fmap.get(fieldName).getDescribe().isAccessible())
                queryFields.add(fieldName);
        }
        return queryFields;
    }

    webservice static List<Contact> getContacts() {
        String soql = 'Select ' + Util.stringJoin(',', getVisibleFields()) + '
                      From Contact limit 20';
        soql += ' From Contact limit 20';
        return Database.query(soql);
    }
}
global with sharing class ContactWebService {

  webservice static Boolean updateEmail(String contactId, String newEmail) {
    if (!Schema.SObjectType.Contact.fields.Email.isUpdateable()) {
      // Missing CRUD-Update or FLS-Write on the Email field
      return False;
    } else {
      Contact c = [Select Id From Contact Where Id=: contactId];
      c.Email = newEmail;
      update c;
      return True;
    }
  }
}
CRUD/FLS – Creating Objects and Fields

- Same is updating
- Call isCreateable() on fields
CRUD/FLS – The Easy Way

- Let VisualForce do it for you!

```java
public with sharing class ContactController {
    public Contact getMyCon () {
        return [Select Email From Contact limit 1];
    }
}

<apex:page controller="ContactController">
    <apex:outputField value="{!myCon.Email}" />
</apex:page>
```
CRUD/FLS – The Easy Way

```java
public with sharing class ContactController {
    public Contact c {get;set;}
    public ContactController () {
        c = [Select Email From Contact limit 1];
    }
}

<apex:page controller="ContactController">
    <apex:form>
        <apex:inputField value="{!c.Email}" />
    </apex:form>
</apex:page>
```
What happens?

- Assume no CRUD-Read and no FLS

```apex
<apex:page controller="ContactController">
    <apex:outputText value="{!c.Email}" />
</apex:page>
```
What happens?

- Assume no CRUD-Read and no FLS

```apex
<apex:page controller="ContactController">
  <h1>{!c.Email}</h1>
</apex:page>
```
What happens?

- Assume no CRUD-Create/Update and FLS is Not Visible or Read Only

```apex
<apex:page controller="ContactController">
  <apex:form>
    <apex:inputText value="{!c.Email}" />
  </apex:form>
</apex:page>
```
What happens?

- Assume no CRUD-Create/Update and FLS is Not Visible or Read Only

public with sharing class ContactController {
    public String getTheEmail() {
        return [Select Email From Contact limit 1].Email;
    }
}

<apex:page controller="ContactController">
    {!theEmail}
</apex:page>
Visualforce Enforcement Guide

- “Naked” `{!obj.Field}` enforces CRUD/FLS readability
- `<apex:*Text>` does **NOT** enforce CRUD/FLS
- `<apex:*Field>` does enforce CRUD/FLS
Permissions – Common Sense Approach

- Check whatever admin perms make sense for your functionality

```java
public with sharing class UserChanger {
    public PageReference changeRandomEmail() {
        Profile prof = [Select PermissionsManageUsers From Profile Where Id=:UserInfo.getProfileId()];
        if (!prof.PermissionsManageUsers)
            return null; // Throw error or something

        User u = [Select Email From User Limit 1];
        u.Email = 'blah@blah.com';
        update u;
        return null;
    }
}
```
Best Practices

- Use “with sharing” on classes with external interfaces
- Reference SObject fields directly in VF with `<apex:*Field> tags`
- Manually check CRUD/FLS when operating on SObjects or field data directly in Apex
Force.com Programmatic Enforcement

Declarative Security Configuration

General Web Application Security