Security for Developers Cheatsheet

salesforce force.com

Overview

The Force.com platform offers built-in security features and protections that can be utilized by developers including object-level access to CRUD operations and field-level security. Also see the Security Cheat Sheet for Administrators.

Sharing Keywords – Force.com

Controls record-level security of data. These keywords are used in Apex class declarations.

`with sharing`	Operate with the calling user's sharing rights. *Recommended.*
`without sharing`	Operate without the calling user's sharing rights. Generally only recommended for classes doing reporting or data aggregation.
`<unspecified sharing>`	Inherit sharing from calling class. Not recommended for Visualforce controllers or Web services.

CRUD (Create, Read, Update, Delete) – Force.com

Controls object-level security of data. These are standard sObject and field methods.

`isCreateable()`	Returns TRUE if instances of this object can be created by the current user, false otherwise.
`isAccessible()`	Returns true if the current user can see instances of this object type, false otherwise.
`isUpdateable()`	Returns true if instances of this object can be updated by the current user, false otherwise.
`isDeleteable()`	Returns true if instances of this object can be deleted by the current user, false otherwise.

FLS (Field Level Security) Describe Call – Force.com

Controls access to object fields. These are standard sObject and field methods.

`isCreateable()`	Returns true if the field can be created by the current user, false otherwise.
`isAccessible()`	Returns true if the current user can see this field, false otherwise.
`isUpdateable()`	Returns true if the field can be edited by the current user, false otherwise.

Visualfore Escaping Functions – Force.com

Server-side functions to escape data to prevent cross-site scripting.

Example:

```
<html><head><title>
   {!HTMLENCODE($Request.title)}
</title></head></html>
```

`JSENCODE`	Escapes data for use in JavaScript quoted strings.
`JSINHTMLENCODE`	Escapes data for use in JavaScript quoted strings that will be used in HTML tags.
`HTMLENCODE`	Escapes data for use in HTML tags.
`URLENCODE`	Escapes data for use in URLs according to RFC 3986 syntax.

ESAPI Functions – Force.com

ESAPI security library for Force.com available at http://code.google.com/p/force-dot-com-esapi.

SFDCAccessController Class
Provides access control functionality to enforce CRUD/FLS and sharing in the Force.com platform.

`setSharingMode()`	Configures the library to operate with sharing, without sharing, or to inherit sharing.
`setOperationMode()`	Configures the library to require all operations be successful or to omit changes for which the user does not have access.
`insertAsUser()`	Insert objects while respecting the user's access rights.
`updateAsUser()`	Update objects while respecting the user's access rights.
`deleteAsUser()`	Delete objects while respecting the user's access rights.
`getViewableFields()`	Return a list of object fields that are viewable by the current user.
`getUpdateableFields()`	Return a list of object fields that are updateable by the current user.
`getCreatableFields()`	Return a list of object fields that are creatable by the current user.

Security for Developers Cheatsheet

salesforce force.com

ESAPI Functions – Force.com cont.

`isAuthorizedToView()`	Returns whether or not the current user is authorized to view a given list of fields of a given object.
`isAuthorizedToCreate()`	Returns whether or not the current user is authorized to create a given list of fields of a given object.
`isAuthorizedToUpdate()`	Returns whether or not the current user is authorized to update a given list of fields of a given object.
`isAuthorozedToDelete()`	Returns whether or not the current user is authorized to delete a given object.
SFDCEncoder Class Provides text escaping functions for Force.com.	
`SFDC_JSENCODE`	Escapes data for use in JavaScript quoted strings.
`SFDC_JSINHTMLENCODE`	Escapes data for use in JavaScript quoted strings that will be used in HTML tags.
`SFDC_HTMLENCODE`	Escapes data for use in HTML tags.
`SFDC_URLENCODE`	Escapes data for use in URLs according to RFC 3986 syntax.

Custom Setting Methods

Special objects and field methods that support a "protected" mode for storing sensitive information like encryption keys.

`getAll()`	Returns a map of the data sets defined for the custom setting. List custom settings only.
`getInstance()`	Returns the "lowest level" custom setting data set for the specified dataset name, user ID, Profile ID, or current user (depending on parameters and setting type). Identical to getValues() for List custom settings.
`getValues()`	Returns only the custom setting data set for the specified dataset name, user ID, Profile ID, or current user (depending on parameters and setting type).
`getOrgDefaults()`	Returns the custom setting data set for the organization. Hierarchy custom settings only.

Crypto Class – Force.com

Provides standard algorithms for creating digests, message authentication codes, and signatures, as well as encrypting and decrypting information using AES. Encryption keys should be stored securely within a Protected Custom Setting.

`encrypt()`	Encrypts the blob clearText using the specified algorithm, private key, and initialization vector. Use this method when you want to specify your own initialization vector.
`encryptWithManagedIV()`	Encrypts the blob clearText using the specified algorithm and private key. Use this method when you want salesforce. com to generate the initialization vector for you.
`decrypt()`	Decrypts the blob cipherText using the specified algorithm, private key, and initialization vector.
`decryptWithManagedIV()`	Decrypts the blob IVAndCipherText using the specified algorithm and private key. Use this method to decrypt blobs encrypted using the encryptWithManagedIV method.
`generateAesKey()`	Generates an AES key of the specified size.
`generateDigest()`	Computes a one-way hash digest based on the input string and algorithm.
`generateMac()`	Computes a message authentication code (MAC) for the input string, using the private key and the specified algorithm.
`getRandomInteger()`	Returns a random Integer.
`getRandomLong()`	Returns a random Long.
`sign()`	Computes a unique digital signature for the input string, using the supplied private key and the specified algorithm.
`signWithCertificate()`	Computes a unique digital signature for the input string, using the specified algorithm and the supplied certificate and key pair.
`signXML()`	Envelops the signature into an XML document.