**public** **with** **sharing** **class** AccessSecurity {

**public** Enum OPERATION\_TYPE {DB\_INSERT, DB\_UPSERT,DB\_UPDATE,DB\_DELETE,DB\_SELECT}

**private** **static** **Set**<String> systemAndRefFields = **new** **Set**<String>{'Id','IsDeleted','CreatedBy','CreatedById','CreatedDate',

'LastModifiedBy','LastModifiedById','LastModifiedDate','SystemModstamp',

'Account','Contact','Who','What','Owner','RecordType','Attachments','Profile','Territory1'};

/\* Method: updateAsUser()

\* Purpose: This methods calls CRUDEnforcementCheck for checking the update accessibility of the requested object.

\* if it passes the CRUDEnforcementCheck, then it update the object

\*/

**public** **static** Database.SaveResult[] updateAsUser(SObject [] devObjs, **Set**<String> fieldSet,boolean arrayAllOrNoneParam) {

**if**(devObjs==**null** || devObjs[0]==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObjs[0] , AccessSecurity.OPERATION\_TYPE.DB\_UPDATE,fieldSet);

**return** Database.**update**(devObjs, arrayAllOrNoneParam);

}

**public** **static** **void** updateAsUser(SObject [] devObjs, **Set**<String> fieldSet) {

**if**(devObjs==**null** || devObjs[0]==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObjs[0] , AccessSecurity.OPERATION\_TYPE.DB\_UPDATE,fieldSet);

**update** devObjs;

}

**public** **static** **void** updateAsUser(SObject devObj, **set**<String> fieldSet) {

**if**(devObj==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObj , AccessSecurity.OPERATION\_TYPE.DB\_UPDATE,fieldSet);

**update** devObj;

}

/\* Method: upsertAsUser()

\* Purpose: This methods calls CRUDEnforcementCheck for checking the upsert accessibility of the requested object.

\* if it passes the CRUDEnforcementCheck, then it upsert the object

\*/

**public** **static** **void** upsertAsUser(SObject [] devObjs, **Set**<String> fieldSet) {

**if**(devObjs==**null** || devObjs[0]==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObjs[0] , AccessSecurity.OPERATION\_TYPE.DB\_UPSERT,fieldSet);

**upsert** devObjs;

}

**public** **static** **void** upsertAsUser(SObject devObj, **set**<String> fieldSet) {

**if**(devObj==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObj , AccessSecurity.OPERATION\_TYPE.DB\_UPSERT,fieldSet);

**upsert** devObj;

}

**public** **static** Database.UpsertResult[] upsertAsUser(SObject [] devObjs, **Set**<String> fieldSet,boolean arrayAllOrNoneParam) {

**if**(devObjs==**null** || devObjs[0]==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObjs[0] , AccessSecurity.OPERATION\_TYPE.DB\_UPSERT,fieldSet);

**return** Database.**upsert**(devObjs, arrayAllOrNoneParam);

}

/\* Method: insertAsUser()

\* Purpose: This methods calls CRUDEnforcementCheck for checking the insert accessibility of the requested object.

\* if it passes the CRUDEnforcementCheck, then it inserts the object

\*/

**public** **static** **void** insertAsUser(SObject devObj, **Set**<String> fieldSet) {

**if**(devObj==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObj , AccessSecurity.OPERATION\_TYPE.DB\_INSERT,fieldSet);

**insert** devObj;

}

**public** **static** Database.SaveResult[] insertAsUser(SObject [] devObjs, **Set**<String> fieldSet,boolean arrayAllOrNoneParam) {

**if**(devObjs==**null** || devObjs[0]==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObjs[0] , AccessSecurity.OPERATION\_TYPE.DB\_INSERT,fieldSet);

**return** Database.**insert**(devObjs, arrayAllOrNoneParam);

}

**public** **static** **void** insertAsUser(SObject [] devObjs, **Set**<String> fieldSet) {

**if**(devObjs==**null** || devObjs[0]==**null**){

**throw** **new** CustomException('NULL record found!');

}

CRUDEnforcementCheck(devObjs[0] , AccessSecurity.OPERATION\_TYPE.DB\_INSERT,fieldSet);

**insert** devObjs;

}

/\* Method: deleteAsUser()

\* Purpose: This methods calls CRUDEnforcementCheck for checking the delete accessibility of the requested object.

\* if it passes the CRUDEnforcementCheck, then it deletes the object

\*/

**public** **static** **void** deleteAsUser(SObject devObj) {

**if**(devObj==**null**){

**throw** **new** CustomException('NULL record found!');

}

**if** (devObj.getSObjectType().getDescribe().isAccessible() && devObj.getSObjectType().getDescribe().isDeletable()){

**delete** devObj;

}**else**{

noAccessException(devObj.getSObjectType(),'',OPERATION\_TYPE.DB\_DELETE);

}

}

**public** **static** Database.DeleteResult[] deleteAsUser(SObject [] devObjs,boolean arrayAllOrNoneParam) {

**if**(devObjs==**null** || devObjs[0]==**null**){

**throw** **new** CustomException('NULL record found!');

}

**if** (devObjs[0].getSObjectType().getDescribe().isAccessible() &&

devObjs[0].getSObjectType().getDescribe().isDeletable()){

**return** Database.**delete**(devObjs, arrayAllOrNoneParam);

}**else**{

noAccessException(devObjs[0].getSObjectType(),'',OPERATION\_TYPE.DB\_DELETE);

}

**return** **null**;

}

**public** **static** **void** deleteAsUser(SObject [] devObjs) {

**if**(devObjs==**null** || devObjs.size()==0){

**throw** **new** CustomException('NULL record found!');

}

**if** (devObjs[0].getSObjectType().getDescribe().isAccessible() &&

devObjs[0].getSObjectType().getDescribe().isDeletable()){

**delete** devObjs;

}**else**{

noAccessException(devObjs[0].getSObjectType(),'',OPERATION\_TYPE.DB\_DELETE);

}

}

**public** **static** **void** CRUDEnforcementCheck(sObject sObj,AccessSecurity.OPERATION\_TYPE operation,**Set**<String> fieldSet) {

**if**(sObj != **null**){

SObjectType objType = sObj.getSObjectType();

**if**(fieldSet == **null** || fieldSet.size() <= 0){

fieldSet = sObj.getPopulatedFieldsAsMap().keySet();

}

**if** (objType != **null**) {

Map < String, Schema.SObjectField > m = objType.getDescribe().**fields**.getMap();

**if** (operation == **null**) {

**return**;

}

**else** **if** (operation == OPERATION\_TYPE.DB\_INSERT){

**if**(objType.getDescribe().isCreateable()){

**for** (String fieldToCheck: fieldSet){

**if**(!systemAndRefFields.contains(fieldToCheck) && !fieldToCheck.contains('\_\_r')){

**if** (m.**get**(fieldToCheck).getDescribe().isPermissionable() &&

!m.**get**(fieldToCheck).getDescribe().isAutoNumber() &&

!m.**get**(fieldToCheck).getDescribe().isCalculated() &&

!m.**get**(fieldToCheck).getDescribe().isCreateable()){

noAccessException(objType, fieldToCheck, operation);

}

}

}

}**else**{

noAccessException(objType,'', operation);

}

}**else** **if** (operation == OPERATION\_TYPE.DB\_UPDATE) {

**if**(objType.getDescribe().isUpdateable()){

**for** (String fieldToCheck: fieldSet) {

**if**(!systemAndRefFields.contains(fieldToCheck) && !fieldToCheck.contains('\_\_r')){

**if** (m.**get**(fieldToCheck).getDescribe().isPermissionable() &&

!m.**get**(fieldToCheck).getDescribe().isAutoNumber() &&

!m.**get**(fieldToCheck).getDescribe().isCalculated() &&

!m.**get**(fieldToCheck).getDescribe().isUpdateable()){

noAccessException(objType, fieldToCheck, operation);

}

}

}

}**else**{

noAccessException(objType,'', operation);

}

}**else** **if** (operation == OPERATION\_TYPE.DB\_SELECT) {

**if**(objType.getDescribe().isAccessible()){

**for** (String fieldToCheck: m.keySet()){

**if** (!m.**get**(fieldToCheck).getDescribe().isAccessible()) {

noAccessException(objType, fieldToCheck, operation);

}

}

}**else**{

noAccessException(objType,'', operation);

}

}**else** **if** (operation == OPERATION\_TYPE.DB\_UPSERT) {

**if**(objType.getDescribe().isCreateable() && objType.getDescribe().isUpdateable()){

**for** (String fieldToCheck: fieldSet){

**if**(!systemAndRefFields.contains(fieldToCheck) && !fieldToCheck.contains('\_\_r')){

**if** (m.**get**(fieldToCheck).getDescribe().isPermissionable() &&

!m.**get**(fieldToCheck).getDescribe().isAutoNumber() &&

!m.**get**(fieldToCheck).getDescribe().isCalculated() &&

(!m.**get**(fieldToCheck).getDescribe().isCreateable() ||

!m.**get**(fieldToCheck).getDescribe().isUpdateable())){

noAccessException(objType, fieldToCheck, operation);

}

}

}

}**else** {

noAccessException(objType,'', operation);

}

}**else** **if** (operation == OPERATION\_TYPE.DB\_DELETE) {

**if** (!objType.getDescribe().isAccessible() && !objType.getDescribe().isDeletable()){

noAccessException(objType,'' , operation);

}

}

}

}**else**{

noAccessToPerform(sObj);

}

}

**private** **static** **void** noAccessException(SObjectType sobjType, String field, OPERATION\_TYPE operation) {

**throw** **new** CustomException((operation != **null**? operation + ': ' : '' ) +'No access to object ' + sobjType + (field != **null**? ' (' + field + ')' : '') );

}

**private** **static** **void** noSuchObjectException(String sobjType) {

**throw** **new** CustomException('No access to object ' + sobjType);

}

**private** **static** **void** noAccessToPerform(SObject sobj) {

**throw** **new** CustomException('No access to perform this operation on ' + sobj);

}

}