ADF Code Corner

76. Extending ADF Security to check ADF BC Entity attribute insert permissions

twitter.com/adfcodecorner

Abstract:

Using ADF Security, ADF Business Components entity attributes can be protected against unauthorized update. However, what if you want to distinguish between update and update while new? This article shows you how the ADF Business Components framework can be extended to also authorize the insert while new case, so that users may have no privilege at all to update an attribute, the privilege to update the attribute if the entity row is new or full update permissions. To ensure minimal impct to the framework behavior, the solution uses a custom Resource Permission you create declaratively in ADF Security.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
08-MAR-2011

Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

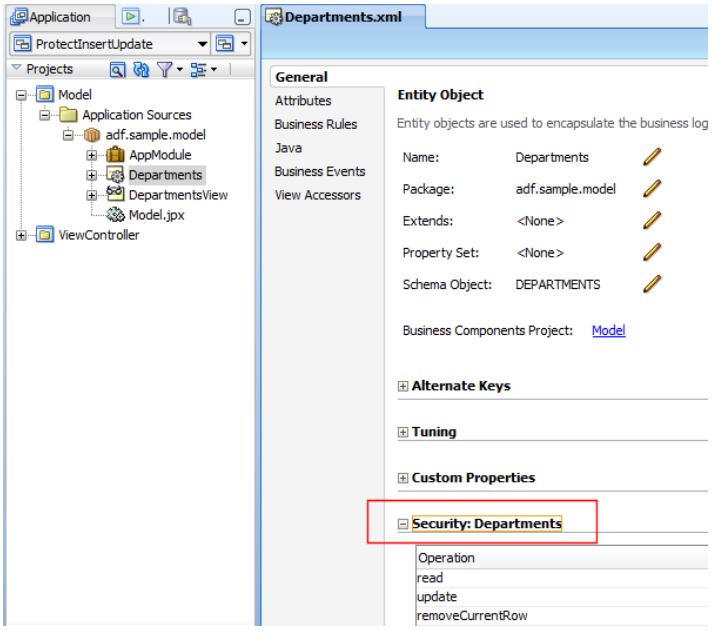
Introduction

Entity attributes in ADF Business Components can be protected against unauthorized update using ADF Security. To do so, you enable ADF Security for your Oracle ADF application as explained in this ADF Insider recording:

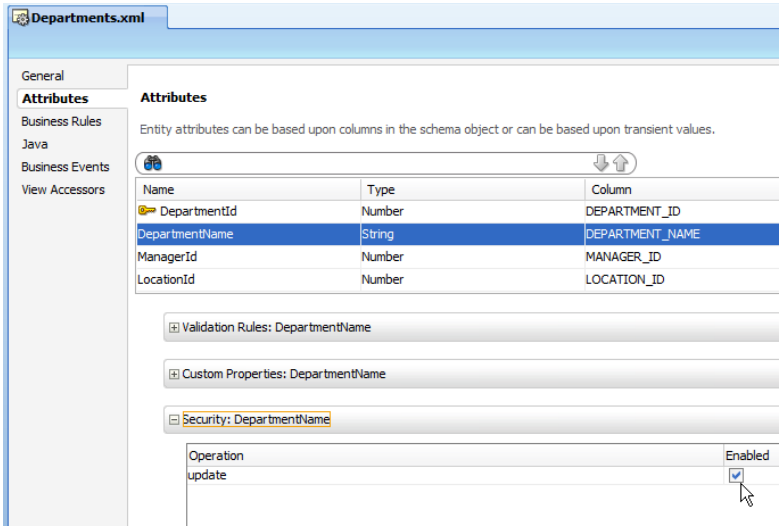
http://download.oracle.com/otn_hosted_doc/jdeveloper/11gdemos/adfinsidersecurity/adfinsidersecurity.html

With ADF Security enabled you then configure entity attribute security. Unlike pages and task flows, which are protected when you enable ADF Security, securing ADF Business Components is a two step process:

- flagging an entity or entity attribute to require authorization
- authorizing users for the protected entity and attribute

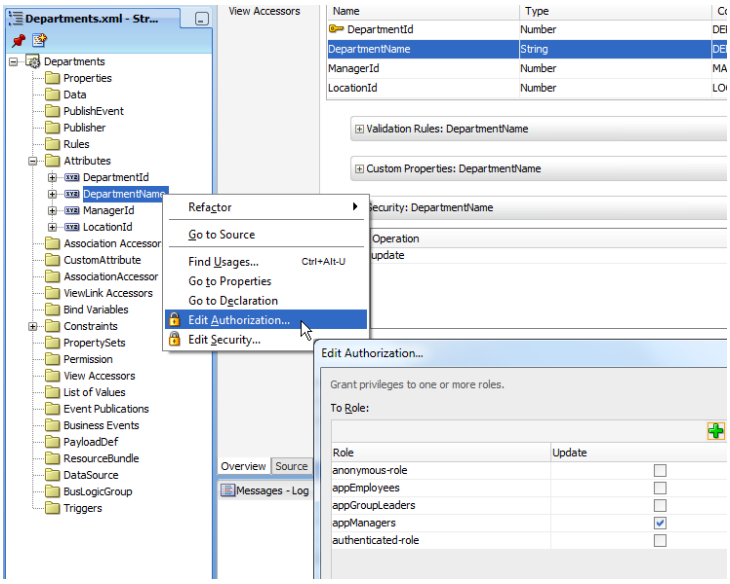


To enforce authorization on an entity or entity attribute, you double click the entity to protect in the Application Navigator and select the security option for the entity or one of its attribites, as shown in the images above and below.



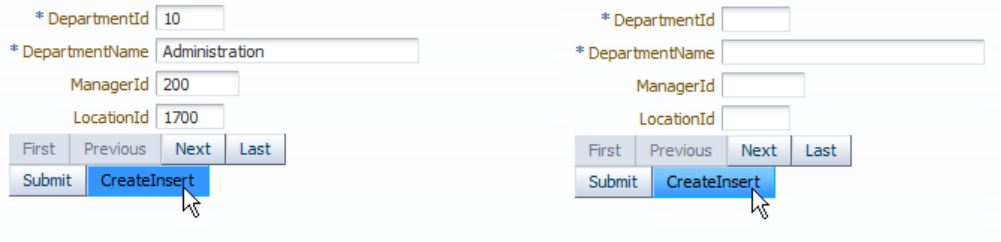
To grant authorization, you select the entity or entity attribute in the Structure Window and choose *Edit Authorization* from the context menu. The *Edit Authorization* dialog lists all application roles defined for the application.

The image below shows authorization configured for an entity attribute, in which case only *update* can be protected by default. For entities, the options are *read, update and delete.*

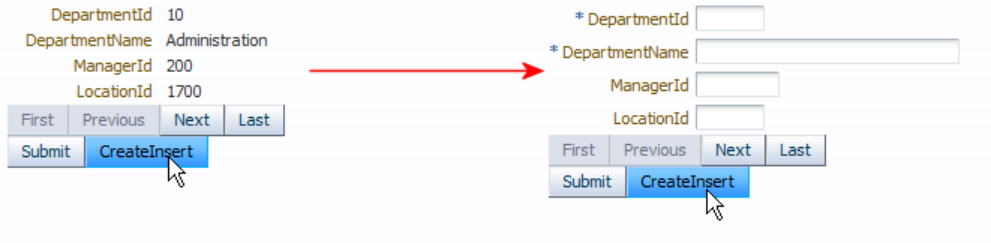


In this article however, I explain a solution that extends the default ADF Business Components framework behavior to further distinguish the entity attribute update protection between *update* and *update when new*. The runtime behavior for different users is shown below:

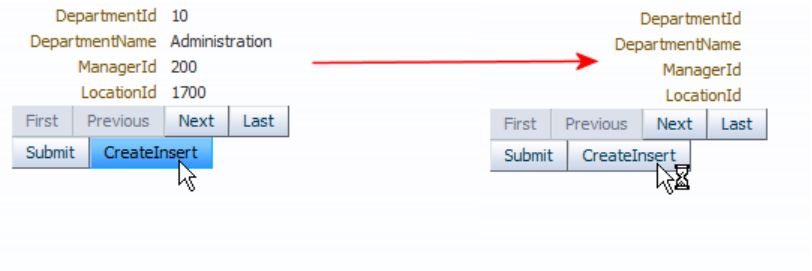
1) Update privilege



2) Update privilege only row is new



3) No update privilege at all



For the 3rd option, when no update privileges are available, you would hide the CreateInsert button using ADF Securit expressions like

Creating and using ResourcePermissions

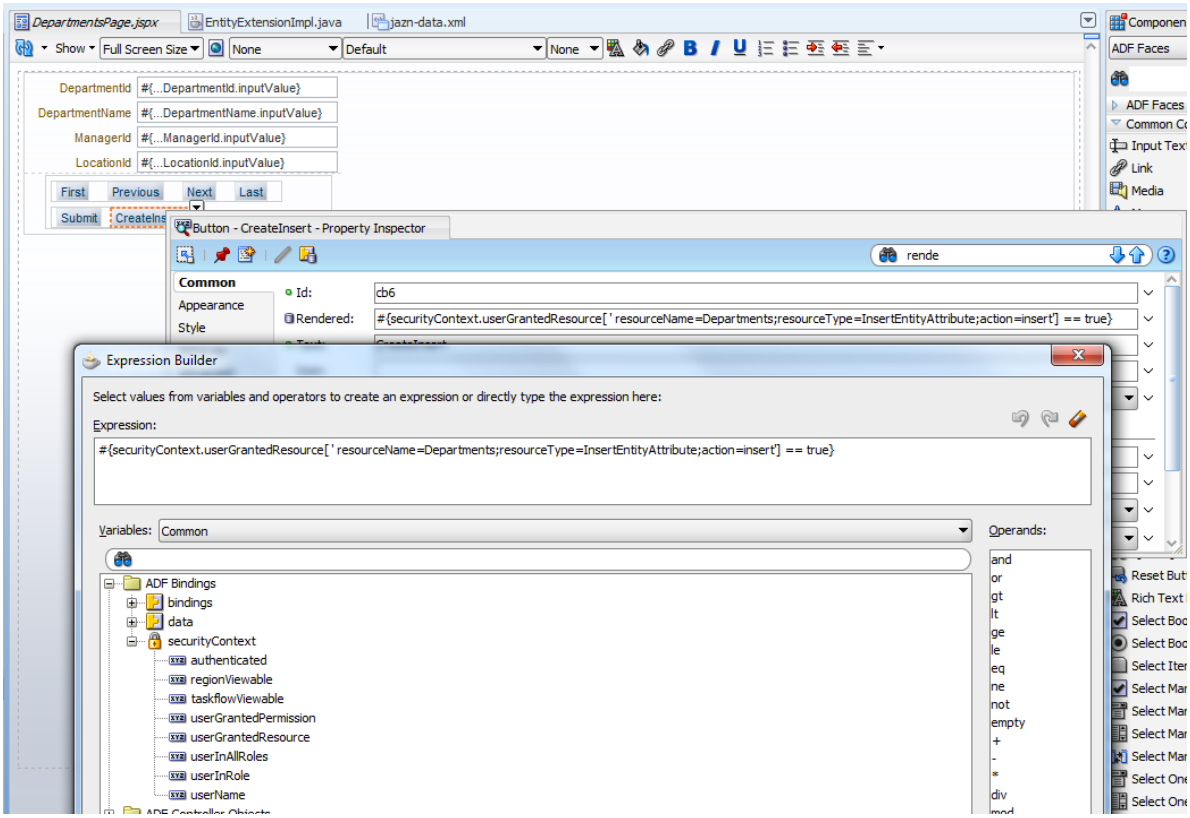
The solution to protect updates for new rows separate from the overall *attribute update* is with the help of the OPSS `ResourcesPermission`, a multi-purpose JAAS permission that you use to declaratively define your own resource protection that then you can check using Expression Language or Java, as documented in this blog:

http://blogs.oracle.com/jdevotnharvest/2011/02/how_to_protect_ui_components_using_opss_resource_permissions.html

The solution explained in this article extends the `EntityImpl` class with a `ResourcePermission` check in Java so the outcome of the authorization check surfaces through the binding layer in a call to *hints.updateable*.

For enabling or disabling the *CreateInsert* button, security Expressions must be used to tell whether the authenticated user is allowed to update the department entity for new entity rows.

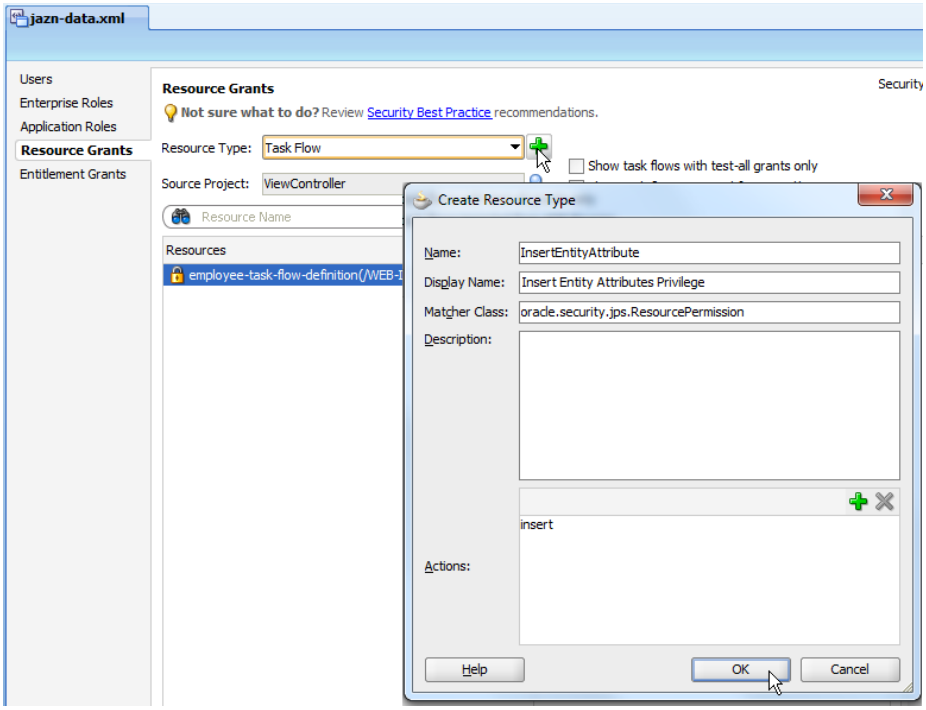
```
#{securityContext.userGrantedResource['resourceName=Departments;resourceType=
InsertEntityAttribute;action=insert']== true}
```



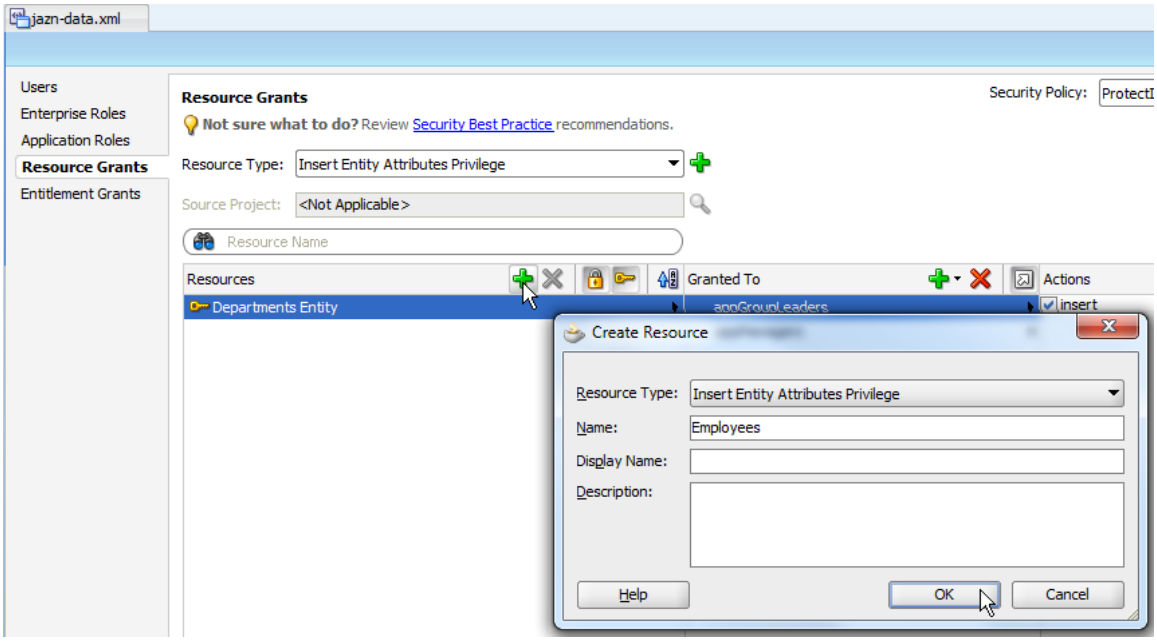
As shown in the image above, the *userGrantedResource* expression is set on the *Rendered* property of the button. It can be set using the Expression Language editor in Oracle JDeveloper.

To create a new `ResourcePermission` configuration, choose **Application | Secure | Resource Grants** from the Oracle JDeveloper menu. Click the green plus icon next to the *Resource Type* label. In the

opened *Create Resource Type* dialog provide a name for the new resource. In my example, the resource is named *InsertEntityAttribute*. The permission class is automatically set to `ResourcePermission`. The action is defined as *insert*. If your resource allows more than a single user action, then you list all of the possible actions so they can be granted to users.



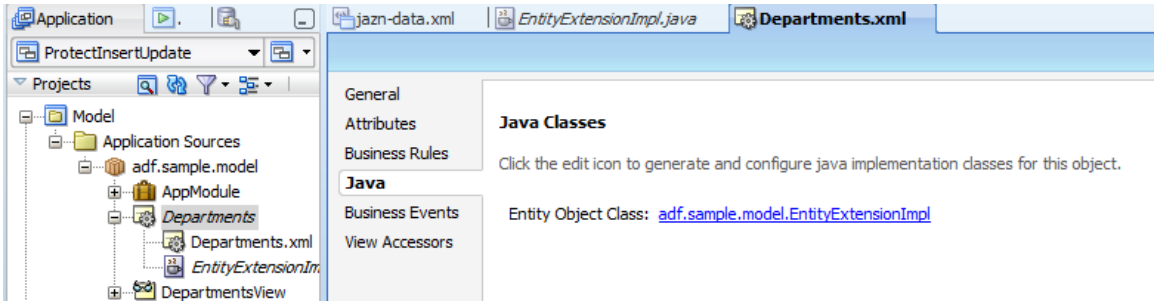
Next, click the green plus icon next to the *Resources* section label to create a new resource to protect with the `InsertEntityAttribute` resource permission type. In the example below I create a resource for the *Employees* entity. When the resource is created, the *insert* action can be granted to users and application roles. I did the same when defining the protection resource for the Departments entity.



Note: In this example I am not going fine granular in that I define a resource for each attribute in an entity. Instead I assume that users either have the right to insert all attributes of a new row or not. If you need this with finer granularity, then this too is possible. Just create a resource type with the entity name in it and then define resources for each attribute.

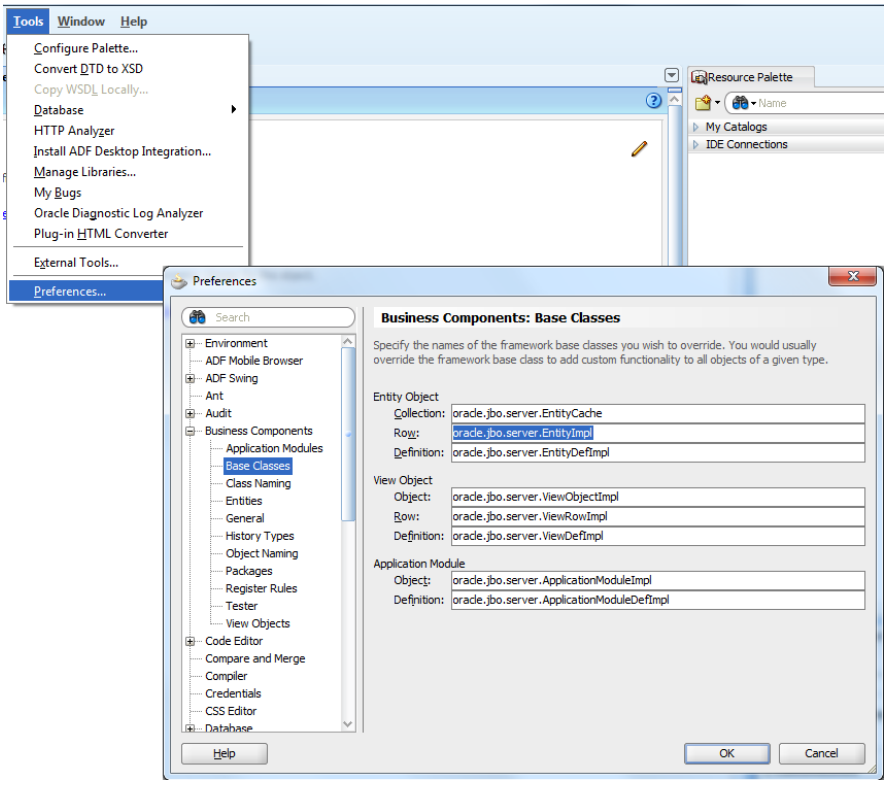
Extending the ADF BC entity classes to use ResourcePermissions

ADF Business Components entities can be extended with a custom entity class that extends the Oracle default `EntityImpl` class for individual entity objects, using the *Java* category in the entity dialog, or globally by configuration on the model project or in the IDE preferences.



In this example, the security extension is applied for a single entity using the *Java* category in the ADF Business Components editing dialog.

If you wanted to do the same for all new entities build within the IDE, you would configure the class on the IDE level as shown below



The entity extension used in this example is shown below. It is a generic solution that you can easily modify, for example to protect *"update when new"* based on individual attribute resource permissions.

```
package adf.sample.model;

import oracle.adf.share.ADFContext;
import oracle.adf.share.security.SecurityContext;
import oracle.adf.share.security.binding.BindingPermissionDef;
import oracle.jbo.DataSecurityProvider;
import oracle.jbo.server.AttributeDefImpl;
import oracle.jbo.server.DBTransactionImpl;
import oracle.jbo.server.EntityCache;
import oracle.jbo.server.EntityImpl;
import oracle.jbo.server.security.DataSecurityProviderManager;
import oracle.jbo.server.security.PermissionHelper;

import oracle.security.jps.ResourcePermission;

public class EntityExtensionImpl extends EntityImpl {
  public EntityExtensionImpl() {
     super();
  }

  @Override
  public boolean isAttributeUpdateable(int i) {

    DBTransactionImpl dbtransaction =
                       (DBTransactionImpl)this.getDBTransaction();
    DataSecurityProvider provider = null;
    provider = new DataSecurityProviderManager(dbtransaction)
                   .getDataSecurityProvider();
    EntityCache ec = getEntityCache();
    AttributeDefImpl attrDef =
                      (AttributeDefImpl) ec.getAttributeDef(i);
    BindingPermissionDef permDef = attrDef.getPermissionDef();

    String privToCheck = permDef == null ? null :
           permDef.findPrivilege(PermissionHelper.UPDATE_ACTION);


   //if one of the following is true, then no security has been enabled
   //on the entity attribute. Security is enabled by choosing the Edit
   //Security option on the attribute context menu in the Structure
   //Window
    if (provider == null || permDef == null || privToCheck == null)
    {
       return true;
    }
```

```
    //check if attribute is new (insert case)
    if (getPostState() == STATUS_NEW ||
        getPostState() == STATUS_INITIALIZED)
    {
        //build ResourcePermission
        //type = InsertEntityAttribute, Target =
        //InsertEntityAttribute, Action = insert
        String type = "InsertEntityAttribute";
        String entityName = this.getEntityDef().getName();
        String action = "insert";
        SecurityContext securityCtx =
            ADFContext.getCurrent().getSecurityContext();
        ResourcePermission resourcePermission =
                    new ResourcePermission(type, entityName,action);
        boolean userHasPermission =
                    securityCtx.hasPermission(resourcePermission);
        if (userHasPermission){
          return true;
        }
        return false;
     }

    //it is an update of an existing attribute value. So let's have
    //the default implementation handling this
    return super.isAttributeUpdateable(i);
  }
}
```

The way this solution works is that it does not require additional configuration other than the creation of a custom `ResourcePermission` configuration. If an attribute is configured for ADF Security *update* protection, then the "*update when new*" privilege is checked too. If an entity attribute is not security enabled then the custom "*update when new*" security check is not performed either.

Note: I think this is exactly what framework extensions are supposed to do: change the framework default behavior with no additional configuration. This also means that undoing the entity extension will automatically use the default behavior of handling attribute updates only.

Important: The extension class above assume you have a `ResourcePermission` configuration "InsertEntityAttribute" in your jazn-data.xml file. It also assumes that the entity name is the name of the resource to protect and that the action to authorize is "insert". If any of these assumptions is different in your implementation, then you need to update the source code accordingly.

Implementing Security on the UI

Because the security is checked on the entity level, there is no need to check the permission again on the ADF Faces component level. Instead you only need to set the component *ReadOnly* property to check whether an attribute is updateable or not.

```
<af:inputText value="#{bindings.DepartmentName.inputValue}"
              …
              readOnly="#{!bindings.DepartmentName.hints.updateable}">
              …
</af:inputText>
```

Note that **all View Object** instances that use the entity object will automatically apply this attribute authorization, making security configuration consistent and easy to maintain.

Download

The sample workspace described in this article can be downloaded from the ADF Code Corner website:

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

There it is example #76, which requires you to configure the model against a database with the HR schema enabled. User names defined are

- sking/welcome1 – has the privilege to always update attributes
- ahunold/welcome1 – can only update attributes if entity is new
- dfaviet/welcome1 – not allowed to update entity attributes, even when they are for a new row. In this case the CreateInsert button is hidden.

RELATED DOCOMENTATION

☒	ADF Insider Recording about ADF Security: http://download.oracle.com/otn_hosted_doc/jdeveloper/11gdemos/adfinsidersecurity/adfinsidersecurity.html
☒	Oracle® Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework11g Release 1 http://download.oracle.com/docs/cd/E17904_01/web.1111/b31974/adding_security.htm#BGBGJEAH
☒	Ch 21 – Fusion Developer Guide, Frank Nimphius, Lynn Munsinger (McGraw Hill) – http://www.mhprofessional.com/product.php?cat=112&isbn=0071622543
☒	Resource Permission blog entry http://blogs.oracle.com/jdevotnharvest/2011/02/how_to_protect_ui_components_using_opss_resource_permissions.html