The IdentifierBundle - who is requesting authorization?

The policy interface has a single method, and looks like this:

```
public interface PolicyIface {
    public PolicyDecision isAuthorized(IdentifierBundle whoToAuth, RequestedAction whatToAuth);
}
```

The nature of whatToAuth is covered in Creating a VIVO authorization policy - an example and A more elaborate authorization policy. This page is about whoToAuth.

The challenge of identity and authorization

A user's level of authorization may depend on a variety of information:

- are they logged in?
- what is their role?
- do they have a profile page?
- what information is in their profile page?
- do they have "proxy authorization" to edit additional pages?

These questions are made more complex because this information is stored in multiple data models. Also, the policy does not have access to the current request or session, so it is not always easy to obtain information.

The IdentifierBundle to the rescue

Notice that the isAuthorized method receives an argument of the type IdentifierBundle. This consists of many Identifier objects, and each Id entifier contains a small piece of information about the current user.

You can see the contents of this bundle (as well as many other things) by directing your browser to /vivo/admin/showAuth. This screen shot shows information about an anonymous (not logged in) session:

Current user Not logged in

Identifiers:

	HasPermission[DisplayByRolePermission['Public']]	
	$Has {\tt Permission} [Simple {\tt Permission} '' java: edu. cornell. mannlib.vitro. we bapp. auth. permissions. Simple {\tt Permission} \# {\tt Page Viewablam} auth. permissions (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Permission} \# {\tt Page Viewablam} auth. permission (Simple {\tt Page Viewablam} auth. permission (Simple {\tt Page Viewablam} auth. permission) auth. permission (Simple {\tt Page Viewablam} auth. permission (Simple {\tt Page Viewablam} auth. permission) auth. permission (Simple$	lePublic']]
	$Has {\tt Permission[SimplePermission['java:edu.cornell.mannlib.vitro.webapp.auth.permissions.SimplePermission#QueryFullMonths and the state of the s$	odel']]

Associated Individuals: (match by http://vivoweb.org/ontology/core#scopusId)



And this one shows information about a user who is logged in as a self-editor.

Current user

URI:	http://vivo.mydomain.edu/individual/u6627
First name:	Able
Last name:	Baker
Email Address:	abaker@mydomain.edu
External Auth ID:	abaker
Login count:	5
Role:	http://vitro.mannlib.cornell.edu/ns/vitro/authorization#SELF_EDITOR

Identifiers:

HasPermissionSet[Self Editor]	
HasPermission[DisplayByRolePermission['Public']]	
$Has {\tt Permission} [Simple {\tt Permission} ['java:edu.cornell.mannlib.vitro.webapp.auth.permissions.Simple {\tt Permission} {\tt #DoFrontEncorrect} and {\tt Permission} {\tt Perm$	lEditing']]
$Has {\tt Permission} [Simple {\tt Permission} ['java:edu.cornell.mannlib.vitro.webapp.auth.permissions.Simple {\tt Permission} \# EditOwn Action {\tt Permission} = {\tt Permission} + {\tt$	count']]
$Has {\tt Permission[SimplePermission[']ava:edu.cornell.mannlib.vitro.webapp.auth.permissions.SimplePermission\#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaptions.SimplePermission#ManageOwnebaption#ManageOwnebaptions.SimplePermission#ManageOwnebaption#ManageOwne$	nProxies']]
$Has {\sf Permission} [Simple {\sf Permission} ['java:edu.cornell.mannlib.vitro.webapp.auth.permissions.Simple {\sf Permission} # {\sf Page Viewallow} ('java:edu.cornell.mannlib.vitro.webapp.auth.permissions.Simple {\sf Permission} # {\sf Page Viewallow} ('$	oleLoggedIn']]
$Has {\tt Permission} [Simple {\tt Permission} ['java:edu.cornell.mannlib.vitro.webapp.auth.permissions.Simple {\tt Permission} {\tt Permissin} {\tt Permission} {\tt Permission} {\tt P$	olePublic']]
$Has {\sf Permission} [Simple {\sf Permission} ['java:edu.cornell.mannlib.vitro.webapp.auth.permissions.Simple {\sf Permission} # {\sf QueryFullMonth}] " {\sf QueryFullMonth} ['java:edu.cornell.mannlib.vitro.webapp.auth.permissions.Simple {\sf Permission} # {\sf QueryFullMonth}] " {\sf$	odel']]
$Has {\tt Permission[SimplePermission[']ava:edu.cornell.mannlib.vitro.webapp.auth.permissions.SimplePermission#QueryUserMetricS$	AccountsModel']]
$Has {\sf Permission} [Simple {\sf Permission}]' java: edu. cornell. mannlib.vitro. we bapp. auth. permissions. Simple {\sf Permission} # Use {\sf BasicAjicAjicAjicAjicAjicAjicAjicAjicAjicAj$	axControllers']]
$Has {\sf Permission} [Simple {\sf Permission}"] iava: edu. cornell.mannlib.vitro.webapp.auth.permissions. Simple {\sf Permission} {\sf #Use Miscellapping and the second $	aneousPages']]
HasProfile[http://vivo.mydomain.edu/individual/n8155]	
IsUser[http://vivo.mydomain.edu/individual/u6627]	

Associated Individuals: (match by http://vivoweb.org/ontology/core#scopusId)

http://vivo.mydomain.edu/individual/n8155 May edit

Your policy has access to these Identifier objects, and the Identifier classes have static methods that make it easier to find the information you want.

For example, in edu.cornell.mannlib.vitro.webapp.auth.identifier.common.IsUser

```
String userUri = null;
Collection<String> userUris = IsUser.getUserUris(whoToAuth);
if (!userUris.isEmpty()) {
    userUri = userUris.iterator().next();
}
// null means not logged in.
// Non-null is the URI of the user account.
```

And, in edu.cornell.mannlib.vitro.webapp.auth.identifier.common.HasProfile

```
String profileUri = null;
Collection<String> profileUris = HasProfile.getProfileUris(whoToAuth);
if (!profileUris.isEmpty()) {
    profileUri = profileUris.iterator().next();
}
// null means either not logged in, or no profile.
// Non-null is the URI of the profile page.
```

In most cases, the policy is more interested in the URI of the profile page, rather than the URI of the user account. However, either one might come in handy.

It might be worth noting that HasProfile and HasProxyEditingRights are both subclasses of HasAssociatedIndividual. That means that you can easily distinguish between them, or not, according to the needs of your particular policy.