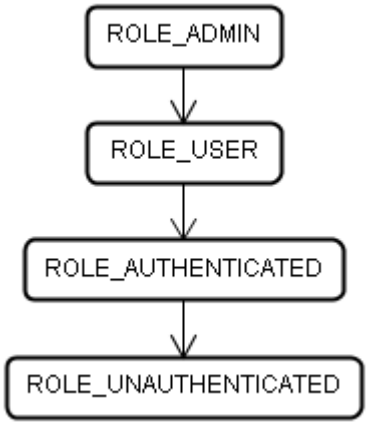
Hierarchical Roles

The concept

Hierarchical roles allow you to define role hierarchies, like the following.



One role in the role hierarchy includes every role lower in the role hierarchy reachable from it and because of that is authorized to do everything it or those reachable lower roles are authorized to do.

So in the above example:

- ROLE_ADMIN includes ROLE_USER, ROLE_AUTHENTICATED and ROLE_UNAUTHENTICATED.
- ROLE_USER includes ROLE_AUTHENTICATED and ROLE_UNAUTHENTICATED.
- ROLE_AUTHENTICATED includes ROLE_UNAUTHENTICATED.

So in conclusion, in a role hierarchy whenever you assign someone a role, you are effectively assigning the role you assign plus all reachable lower roles in the role hierarchy to this person.

The implementation in Acegi

You would configure the role hierarchy from the example above like this:

```
<bean id="roleHierarchy" class="org.acegisecurity.userdetails.hierarchicalroles.RoleHierarchyImpl" >
	<property name="hierarchy">
		<value>
			ROLE_ADMIN > ROLE_USER
			ROLE_USER > ROLE_AUTHENTICATED
			ROLE_AUTHENTICATED > ROLE_UNAUTHENTICATED
		</value>
	</property>
</bean>
```

Take a look at the line "ROLE_ADMIN > ROLE_USER" from the role hierarchy configuration. As you see, one arrow from the role hierarchy picture above directly translates into a greater sign (which also somehow looks like an arrow) in the role hierarchy configuration. (Hint: You can read that greater sign as "includes", e. g. ROLE_ADMIN includes ROLE_USER).

(For the advanced reader: The RoleHierarchyImpl bean is responsible for finding all reachable roles in the role hierarchy when given an GrantedAuthority[] of directly assigned roles.)

Up to now that role hierarchy is isolated and will not yet be used by your application. Therefore you have to also configure a hierarchical roles UserDetailsServiceWrapper like this:

```
<bean id="userDetailsServiceWrapper"
      class="org.acegisecurity.userdetails.hierarchicalroles.UserDetailsServiceWrapper" >
       <property name="roleHierarchy">
              <ref bean="roleHierarchy"/>
       </property>
       <property name="userDetailsService">
              <ref bean="wrappedUserDetailsService"/>
       </property>
</bean>
```

The purpose of the UserDetailsServiceWrapper (which also is an UserDetailsService) is to wrap an existing UserDetailsService and add all reachable lower roles in the configured role hierarchy to any UserDetails the wrapped UserDetailsService returns.

An example for clarification:
Let's suppose your UserDetailsService returns ROLE_ADMIN for the user with username "johndoe". When you wrap your UserDetailsService with the above configuration the UserDetailsServiceWrapper will return ROLE_ADMIN, ROLE_USER, ROLE_AUTHENTICATED and ROLE_UNAUTHENTICATED for that user.

The last step to be able to use hierarchical roles is to use the UserDetailsServiceWrapper anywhere where you previously would have directly used your UserDetailsService (for the advanced reader: the UserDetailsServiceWrapper also is an UserDetailsService because it implements the UserDetailsService interface).

Imagine a very simple use case where you want to state that every logged in person is allowed to logout.

Previously (without hierarchical roles) you had to write something like this:
```
/logout.html=ROLE_ADMIN,ROLE_USER,ROLE_AUTHENTICATED
```

Given the above configuration, now you will be able to write:
```
/logout.html=ROLE_AUTHENTICATED
```

As you see, hierarchical roles not only give you additional power and expressiveness when designing your security model, they also make your intentions clearer, your access rules shorter, more readable and easier to maintain. One could also argue that you don't have to assign and store that many roles per user when you have hierarchical roles.