Comment gérer la sécurité XMLRAD à partir d'une base de données

Par Jean-Philippe BEMPEL

Date de publication : 22 décembre 2004

Cet article explique comment gérer une authentification et une autorisation à partir d'une base de données et non du fichier Security.xml. ♪

Introduction....3
La sécurité intégrée de XMLRAD....3
Personnalisation de la sécurité....4
Personnaliser la procédure de login....5
Personnaliser l'authentification....7
Personnaliser l'autorisation....11
Conclusion....13

Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Introduction

XMLRAD fournit un mécanisme de sécurité dans une application permettant de vérifier qu'un utilisateur a le droit d'accéder à toutes ou certaines parties de l'application. Ce mécanisme est basé sur un fichier Security.xml contenant la liste des utilisateurs, leurs mots de passe (hachés MD5) et les groupes auxquels ils appartiennent. Dans chaque XMLService on va alors pouvoir définir quels sont les utilisateurs ou groupes qui sont autorisés à exécuter le XMLService.

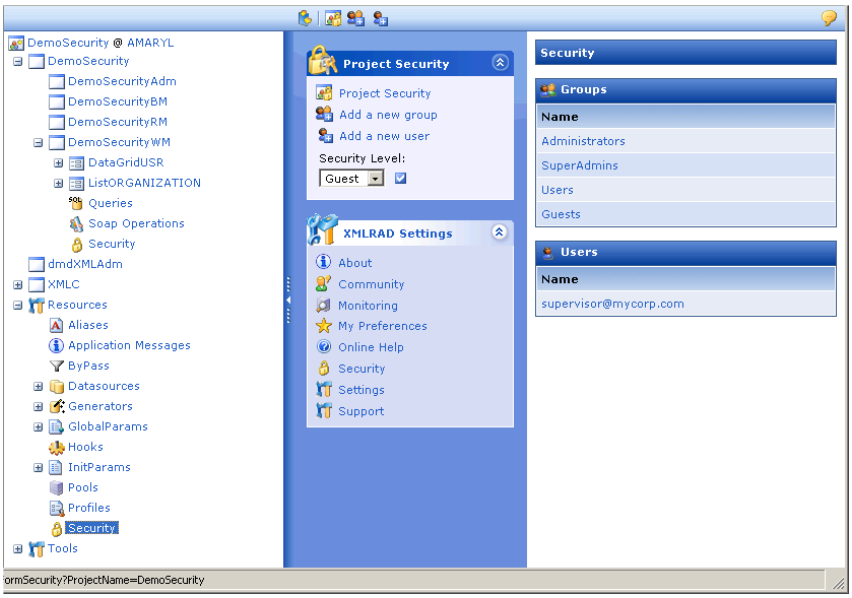
Cependant, on développe souvent une application à partir d'une base de données déjà existante et intégrant une base d'utilisateurs. Il est donc intéressant de réutiliser cette base plutôt que de recréer ces utilisateurs dans le fichier Security.xml. XMLRAD permet de s'affranchir de la mécanique par défaut et de pouvoir réaliser les opérations d'authentification et d'autorisation à l'aide de la base de données ou de toute autre ressource (par exemple un annuaire LDAP).

Nous allons d'abord expliquer comment fonctionne la sécurité intégrée de XMLRAD puis décrire pas à pas comment effectuer une authentification à partir d'un annuaire d'utilisateurs existant dans une base de données.

La sécurité intégrée de XMLRAD

La sécurité intégrée de XMLRAD repose sur un annuaire d'utilisateurs déclaré dans le fichier Security.xml se trouvant dans le répertoire \Bin du projet. Dans cet annuaire on y stocke outre le nom de l'utilisateur, son identifiant, son mot de passe haché en MD5, sa session et l'IP avec laquelle il se connecte.

Pour les autorisations, on a une définition de groupes utilisateur, c'est-à-dire une association utilisateur et groupe d'utilisateurs dans l'annuaire. La définition des permissions se fait au niveau de chaque XMLService par l'ajout soit d'un ou plusieurs groupes soit l'ajout d'un ou plusieurs utilisateurs.



Il existe aussi un niveau de sécurité permettant d'indiquer comment doit être appliquée la sécurité au niveau de l'application. On modifie ce niveau dans le fichier InitParams.xml avec le paramètre XMLC_SecurityLevel. Ce paramètre peut prendre deux valeurs : **Guest** ou **Global**.

Avec le niveau de sécurité **Guest** qui est la valeur par défaut, l'authentification n'est pas obligatoire pour chaque XMLService ou Page de l'application et tout le monde est autorisé à exécuter les XMLServices.

Avec le niveau de sécurité **Global**, l'authentification est obligatoire. On est redirigé vers une boite de login lors du lancement de l'application.

Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

La vérification d'authentification est effectuée à chaque requête HTTP. L'application effectue la vérification dans la méthode CheckAuthentication. La première chose que fait cette méthode c'est de vérifier que l'IP ou le XMLService demandé ne fait pas partie du fichier ByPass.xml. Dans le cas contraire, la requête est exécutée normalement. Si le XMLC_SecurityLevel est à Guest, un identifiant d'utilisateur (XMLC_UserID) est généré pour la conservation du profil et la requête est considérée comme authentifiée. Par contre si le XMLC_SecurityLevel est à la valeur Global, le XML_UserName qui doit être spécifié dans les cookies permet de sélectionner l'utilisateur dans l'annuaire. On confronte alors la session stockée elle aussi dans les cookies (XMLC_Session) avec celle stockée dans le fichier Security.xml. L'expiration et l'IP sont également vérifiées. Si l'une des étapes se passe mal, il y a redirection vers la Form XMLC_FormLogin.

La vérification de l'autorisation se fait en récupérant les permissions au niveau du XMLService. Si on trouve le XMLC_UserID dans les permissions, on autorise. Si les permissions sont à everyone, on autorise. Si le XMLC_Authenticated est à 1, on autorise. Si le groupe utilisateur se trouve parmi les permissions, on autorise. Dans tous les autres cas, on refuse l'accès et une exception XMLC_ACCESS_DENIED est levée.

Le processus de Login est situé dans l'action XMLC_Login dans le module XMLSecurityWM. Cette action est exécutée à la validation du formulaire de XMLC_FormLogin. Le processus de login effectue une recherche dans le fichier Security.xml de l'utilisateur avec le UserName. Si le password dans le fichier est vide, on redirige vers XMLC_FormChangePwd. Si le password transmis (calculé en MD5 côté client) n'est pas le même, une exception ERR_LOGIN_FAILED est levée. On génère une session (XMLC_Session) et on calcule une expiration en fonction de l'InitParams XMLC_LoginTimeOut. Les valeurs XMLC_Session, XMLC_UserID, XMLC_UserName sont stockées dans les cookies pour authentification lors des prochaines requêtes.

Pour que la sécurité soit effective, assurez-vous que le paramètre InitParams XMLC_Design soit à 0.

Personnalisation de la sécurité

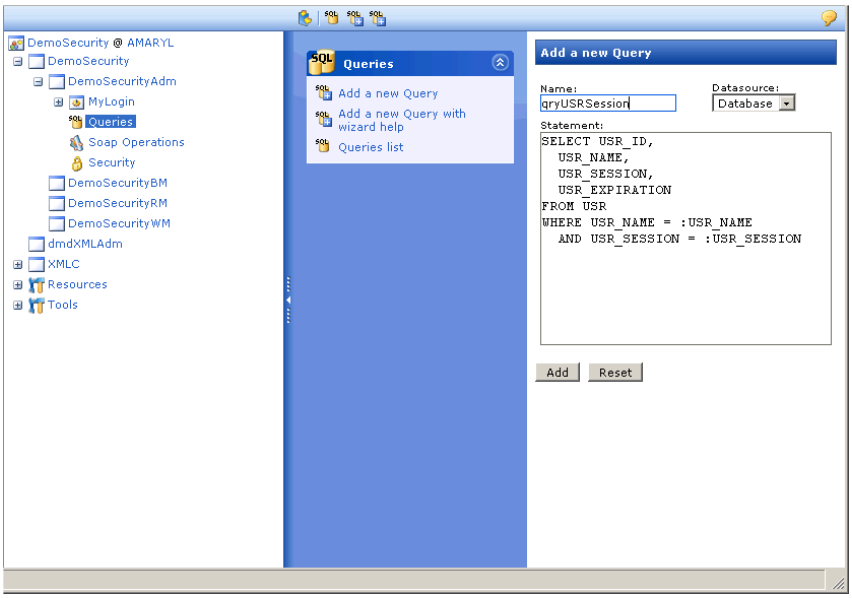
Nous allons partir de l'application de démo Training livrée avec XMLRAD. Cette démo se connecte à une base Access elle aussi livrée. Nous allons rajouter dans cette base une table utilisateur USR avec les champs USR_ID, USR_NAME, USR_PWD, USR_SESSION et USR_EXPIRATION. voici le script SQL permettant de la créer :

```
CREATE TABLE USR (
  USR_ID INTEGER,
  USR_NAME VARCHAR(50),
  USR_PASSWORD VARCHAR(50),
  USR_SESSION VARCHAR(50),
  USR_EXPIRATION FLOAT)
```

Nous devons maintenant modifier l'application pour effectuer notre authentification. Dans XMLRAD, nous allons rajouter trois nouvelles requêtes SQL pour la gestion avec notre base de données. Une requête nommée qryUSRSession permettant de vérifier que la valeur du XMLC_Session transmise avec les cookies est bien celle qui est stockée dans la base de données et que la session n'a pas expiré.

```
SELECT USR_ID,
  USR_NAME,
  USR_SESSION,
  USR_EXPIRATION
FROM USR
WHERE USR_NAME = :USR_NAME
  AND USR_SESSION = :USR_SESSION
```


Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.



Une requête nommée qryUSRPwd permettant de confronter le password transmis lors du login, associé au UserName, avec celui stocké dans la base de données.

```
SELECT USR.USR_ID
FROM USR USR
WHERE USR_NAME = :USR_NAME
  AND USR_PASSWORD = :USR_PASSWORD
```

Enfin, une dernière requête qryUpdateUSRSession pour mettre à jour la nouvelle session calculée lors du login ainsi que son expiration.

```
UPDATE USR
SET
  USR_SESSION = :USR_SESSION,
  USR_EXPIRATION = :USR_EXPIRATION
WHERE USR_ID = :USR_ID
```

Personnaliser la procédure de login

Nous n'allons pas toucher à l'autorisation. Nous allons commencer par nous occuper du processus de login.

Pour détourner l'action XMLC_Login nous allons utiliser les Hooks introduits dans la version 2005 de XMLRAD. Ceci nous permettra de conserver la cinématique de la sécurité de XMLRAD (XMLFormLogin/XMLC_Login).

Un hook permet de remplacer ou de compléter un XMLService existant. Ici le XMLService XMLC_Login fourni par le framework, ne nous convient pas, car il effectue une vérification avec les données se trouvant dans le fichier Security.xml, or nous avons ces informations dans la base de données. C'est pourquoi il nous faut un moyen de détourner l'exécution de ce XMLService au profit de notre propre gestion. Les hooks sont là pour cela. Nous allons remplacer XMLC_Login par le XMLService MyLogin. Pour ce faire nous allons déclarer un Hook sur l'événement BeforeXMLGram de l'action XMLC_Login et qui déclenchera à ce moment-là, le XMLService MyLogin.

On crée donc un nouvel XMLService MyLogin dans le module TrainingAdm. Sur l'événement BeforeXMLGram de ce dernier, on va récupérer les champs **UserName** et **Password** transmis par le formulaire de XMLC_FormLogin pour les transférer dans les champs USR_NAME et USR_PASSWORD qui seront utilisés comme paramètres pour la requête qryUSRPwd.

Si le champ USR_ID résultat de cette requête est à vide, le password est invalide. On lève donc une exception indiquant l'erreur.

Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

On génère une session par exemple en hachant un timestamp avec MD5.

On prépare les valeurs XMLC_UserID, XMLC_UserName et XMLC_Session dans le contexte qui seront stockées dans les cookies par la suite.

On calcule l'expiration de la session avec le paramètre InitParams XMLC_LoginTimeOut, puis on met à jour la session et l'expiration dans la base de données en utilisant la requête qryUpdateUSRSession.

Enfin on positionne la valeur XMLC_SkipLoginAuthentication à 1 pour que les valeurs dans les cookies soient inscrites et que le profile de l'utilisateur soit chargé correctement par le XMLC_Login.

Voici le code :

Delphi

```
procedure TTrainingAdm.MyLoginBeforeXMLGram(XMLGram: IXMLGram; e: TBeforeXMLGramEventArgs);
var
  Session: WideString;
  Expiration: TDateTime;
begin
  Context.SetValue('USR_NAME', Context.GetValue('UserName'));
  Context.SetValue('USR_PASSWORD', Context.GetValue('Password'));
  XMLCollection.DBExtract('qryUSRPwd');
  if Context.GetValue('USR_ID') = '' then
    XMLRequest.RaiseError('Invalid password', 'TTrainingAdm.MyLoginBeforeXMLGram');
  Session := WideUpperCase(MD5(FloatToStr(Now)));
  // Prepare for Cookies
  Context.SetValue('XMLC_UserID', Context.GetValue('USR_ID'));
  Context.SetValue('XMLC_UserName', Context.GetValue('USR_NAME'));
  Context.SetValue('XMLC_Session', Session);
  // Prepare for update DB Session + Expiration
  Context.SetValue('USR_SESSION', Session);
  Expiration := Now+StrToIntDef(XMLApplication.InitParams.GetValue('XMLC_LoginTimeOut'), 720) /
 (24 * 60);
  Context.SetValue('USR_EXPIRATION', FloatToStr(Expiration));
  XMLCollection.DBBatch('qryUpdateUSRSession');
  Context.SetValue('XMLC_SkipLoginAuthentication', '1');
end;
```

C#

```
private void MyLogin_BeforeXMLGram(XMLCLX.IXMLGram XMLGram, XMLComponent.TBeforeXMLGramEventArgs
 e)
{
    Context.SetValue("USR_NAME", Context.GetValue("UserName"));
    Context.SetValue("USR_PASSWORD", Context.GetValue("Password"));
    XMLCollection.DBExtract("qryUSRPwd");
    if (Context.GetValue("USR_ID") == "")
        XMLRequest.RaiseError("Invalid password", "MyLogin_BeforeXMLGram");
    string XMLC_String = DateTime.Now.Ticks.ToString();
    Context.SetValue("XMLC_String", XMLC_String);
    XMLCollection.Execute("XMLC_HashString");
    string Session = Context.GetValue("XMLC_HashedString");
    // Prepare for Cookies
    Context.SetValue("XMLC_UserID", Context.GetValue("USR_ID"));
    Context.SetValue("XMLC_UserName", Context.GetValue("USR_NAME"));
    Context.SetValue("XMLC_Session", Session);
    // Prepare for update DB Session + Expiration
    Context.SetValue("USR_SESSION", Session);
    int LoginTimeOut =
  int.Parse(XMLApp.Units.XMLApp.XMLApplication.InitParams.GetValue("XMLC_LoginTimeOut"));
    DateTime Expiration = DateTime.Now.AddMinutes(LoginTimeOut);
    Context.SetValue("USR_EXPIRATION", Expiration.Ticks.ToString());
    XMLCollection.DBBatch("qryUpdateUSRSession");
    Context.SetValue("XMLC_SkipLoginAuthentication", "1");
}
```

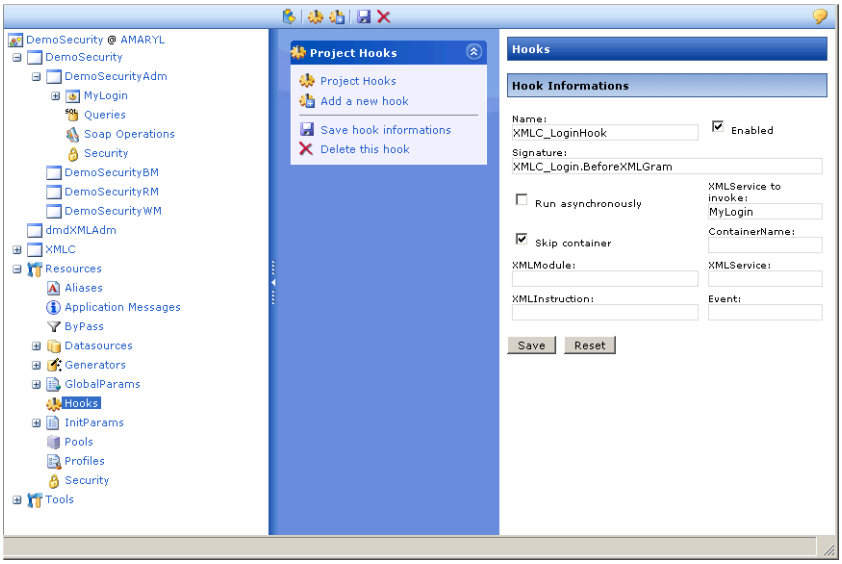

Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Java
```
public void MyLogin_BeforeXMLGram(XMLGram xmlgram, XMLComponent.BeforeXMLGramEventArgs e)
{
    context.setValue("USR_NAME", context.getValue("UserName"));
    context.setValue("USR_PASSWORD", context.getValue("Password"));
    xmlCollection.dbExtract("qryUSRPwd");
    if (context.getValue("USR_ID") == "")
        xmlRequest.raiseError("Invalid password", "MyLogin_BeforeXMLGram");

    String XMLC_String = Long.toString(new Date().getTime());
    context.setValue("XMLC_String", XMLC_String);
    xmlCollection.execute("XMLC_HashString");
    String Session = context.getValue("XMLC_HashedString");
    //      Prepare for Cookies
    context.setValue("XMLC_UserID", context.getValue("USR_ID"));
    context.setValue("XMLC_UserName", context.getValue("USR_NAME"));
    context.setValue("XMLC_Session", Session);
    //      Prepare for update DB Session + Expiration
    context.setValue("USR_SESSION", Session);
    int LoginTimeOut =
 Integer.parseInt(xmlApplication.getInitParams().getValue("XMLC_LoginTimeOut"))*60*1000;
    long Expiration = new Date().getTime() + LoginTimeOut;
    context.setValue("USR_EXPIRATION", Long.toString(Expiration));
    xmlCollection.dbBatch("qryUpdateUSRSession");
    context.setValue("XMLC_SkipLoginAuthentication", "1");
}
```

Il ne nous reste plus qu'à déclarer notre Hook. On va trouver la déclaration des hooks dans les ressources du projet. On crée un nouveau hook nommé XMLC_LoginHook dont la signature sera la suivante :

XMLC_Login.BeforeXMLGram

L'exécution n'est pas asynchrone et le XMLService à invoquer est MyLogin.



Personnaliser l'authentification

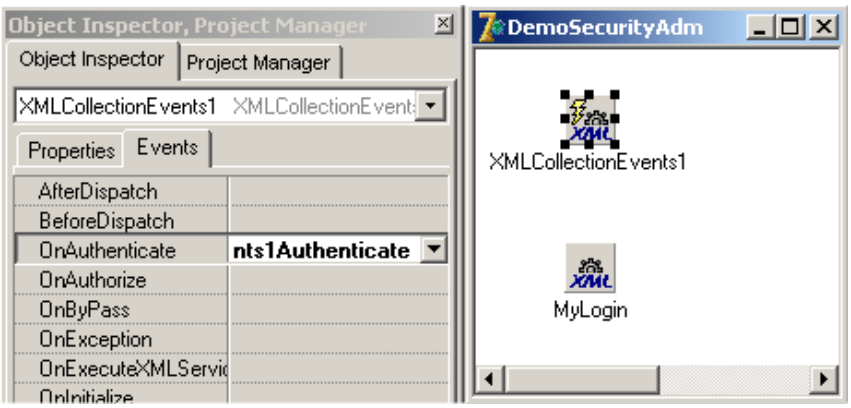
Ouvrons le projet Training.dpr dans Delphi. Dans le module TrainingAdm, posons un composant XMLCollectionEvents. Ce composant permet d'implémenter des gestionnaires d'événement liés à l'accès aux XMLServices. On retrouve dans ce composant des événements tels que BeforeDispatch, OnAuthenticate, OnAuthorize, OnByPass, OnException, OnExecuteXMLService et OnInitialize.

L'implémentation des événements OnAuthenticate et OnAuthorize va nous permettre de court-circuiter la gestion de la sécurité classique de XMLRAD et nous laisser le champ libre sur la manière de la traiter. Mais nous voulons ici

Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

seulement modifier la méthode d'authentification et garder le mécanisme d'autorisation qui lui, est indépendant de la base de données.

Nous allons donc implémenter l'événement OnAuthenticate.



Nous devons d'abord vérifier la connexion à la base de données. En effet, comme on se trouve en dehors du XMLService et du XMLGram, il se peut que la connexion ne soit pas effective. Il faut de même s'assurer qu'une transaction est démarrée.

Comme nous nous occupons de gérer l'authentification, on renvoie dans l'argument du gestionnaire d'événement e.Handled la valeur True.

On récupère les valeurs XMLC_Session et XMLC_UserName transmises par les cookies à partir du Context. On les utilise comme paramètres pour la requête qryUSRSession. On vérifie ainsi que la session transmise est la même que dans la base de données, assurant ainsi que l'utilisateur est toujours authentifié.

Si la requête ne renvoie pas dans le Context le champ USR_ID, l'utilisateur n'est pas authentifié. On lève alors une exception XMLC_AUTHENTICATION_REQUIRED.

Il nous reste à tester si cette session n'a pas expiré. On lèvera la même exception dans le cas contraire.

L'utilisateur est donc bien authentifié et on stocke dans le Context une variable XMLC_Authenticated avec la valeur 1.

Voici le code :

Delphi
```
procédure TTrainingAdm.XMLCollectionEvents1Authenticate(XMLCollection: IXMLCollection; e:
 TAuthenticateEventArgs);
var
  Database: IDacDatabase;
  Session: WideString;
  USR_NAME: WideString;
  Expiration: TDateTime;
begin
  if CompareText(XMLApplication.InitParams.Values['XMLC_SecurityLevel'], 'GUEST') = 0 then
  begin
    Context.SetValue('XMLC_Authenticated', '1');
    Exit;
  end;
  Database := XMLCollection.GetDatabase('Database');
  if Database = nil then
      XMLRequest.RaiseError('Cannot find
 database', 'TTrainingWM.XMLCollectionEvents1Authenticate');
  if not Database.GetConnected then
    Database.Open;
  Database.StartTransaction;
  try
```


Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Delphi

```
    try
      e.Handled := True;
      USR_NAME := Context.GetValue('XMLC_UserName');
      Session := Context.GetValue('XMLC_Session');
      Context.SetValue('USR_NAME', USR_NAME);
      Context.SetValue('USR_SESSION', Session);
      XMLCollection.DBExtract('qryUSRSession');
      if Context.GetValue('USR_ID') = '' then
      begin
        Context.SetValue('XMLC_Authenticated', '0');

 XMLRequest.RaiseError('XMLC_AUTHENTICATION_REQUIRED', 'TTrainingWM.XMLCollectionEvents1Authenticate');
      end;
      Expiration := StrToFloatDef(Context.GetValue('USR_EXPIRATION'), 0);
      if Expiration < Now then
      begin
        Context.SetValue('XMLC_Authenticated', '0');

 XMLRequest.RaiseError('XMLC_AUTHENTICATION_REQUIRED', 'TTrainingWM.XMLCollectionEvents1Authenticate');
      end;
      Context.SetValue('XMLC_Authenticated', '1');
      // mettre ici le code de gestion des groupes et autorisations (cf. Personnaliser
 l'autorisation)
    except
      Database.Rollback;
      raise;
    end;
  finally
    Database.Commit;
  end;
end;
```

C#

```
private void XMLCollectionEvents1_OnAuthenticate(XMLCLX.IXMLCollection XMLCollection,
 XMLComponent.TAuthenticateEventArgs e)
{
    e.Handled = true;

  if (XMLApp.Units.XMLApp.XMLApplication.InitParams.GetValue("XMLC_SecurityLevel").toUpper() == "GUEST")
    {
      Context.SetValue('XMLC_Authenticated', '1');
      return;
    }
    IDacDatabase Database = XMLCollection.GetDatabase("Database");
    if (Database == null)
        XMLRequest.RaiseError("Cannot find database", "XMLCollectionEvents1Authenticate");
    if (!Database.GetConnected())
        Database.Open();
    Database.StartTransaction();
    try
    {
        string USR_NAME = Context.GetValue("XMLC_UserName");
        string Session = Context.GetValue("XMLC_Session");
        Context.SetValue("USR_NAME", USR_NAME);
        Context.SetValue("USR_SESSION", Session);
        XMLCollection.DBExtract("qryUSRSession");
        if (Context.GetValue("USR_ID") == "")
        {
            Context.SetValue("XMLC_Authenticated", "0");

 XMLRequest.RaiseError("XMLC_AUTHENTICATION_REQUIRED", "XMLCollectionEvents1Authenticate");
        }
        string USR_EXPIRATION = Context.GetValue("USR_EXPIRATION");
        double Expiration = double.Parse(USR_EXPIRATION);
        if (Expiration < DateTime.Now.Ticks)
        {
            Context.SetValue("XMLC_Authenticated", "0");

 XMLRequest.RaiseError("XMLC_AUTHENTICATION_REQUIRED", "XMLCollectionEvents1Authenticate");
        }
```


Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

C#

```
        Context.SetValue("XMLC_Authenticated", "1");
        // mettre ici le code de gestion des groupes et autorisations (cf. Personnaliser
 l'autorisation)
    }
    catch
    {
        Database.Rollback();
        throw;
    }
    finally
    {
        Database.Commit();
    }
}
```

Java

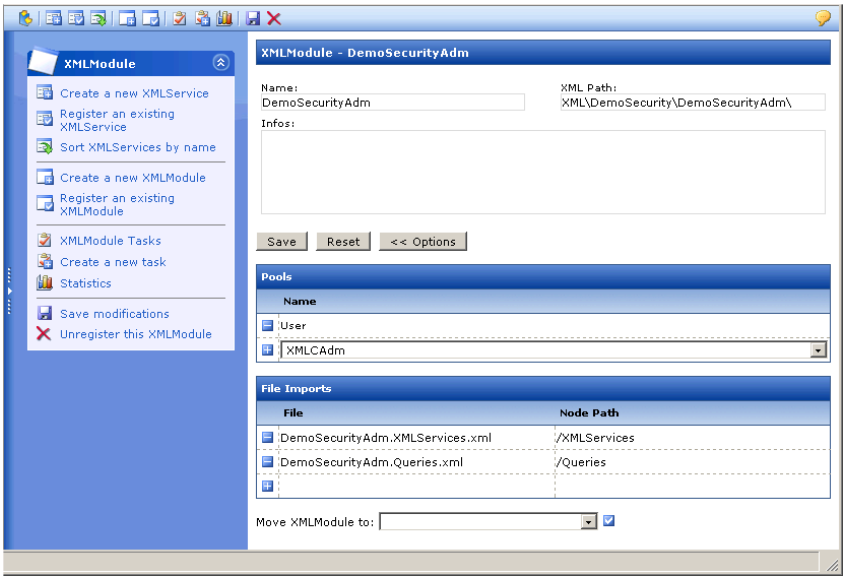
```
public void xmlCollectionEvents1_OnAuthenticate(XMLCollection xmlCollection,
 XMLCollectionEvents.AuthenticateEventArgs e) throws Exception
{
    e.handled = true;
    if (xmlApplication.getInitParams().getValue("XMLC_DefaultAction").equalsIgnoreCase("GUEST"));
    {
        context.setValue("XMLC_Authenticated", "1");
        return;
    }
    DacDatabase Database = xmlCollection.getDatabase("Database");
    if (Database == null)
        xmlRequest.raiseError("Cannot find database", "XMLCollectionEvents1Authenticate");
    if (!Database.getConnected())
        Database.open();
    Database.startTransaction();
    try
    {
        String USR_NAME = context.getValue("XMLC_UserName");
        String Session = context.getValue("XMLC_Session");
        context.setValue("USR_NAME", USR_NAME);
        context.setValue("USR_SESSION", Session);
        xmlCollection.dbExtract("qryUSRSession");
        if (context.getValue("USR_ID").equals(""))
        {
            context.setValue("XMLC_Authenticated", "0");

 xmlRequest.raiseError("XMLC_AUTHENTICATION_REQUIRED", "XMLCollectionEvents1Authenticate");
        }
        String USR_EXPIRATION = context.getValue("USR_EXPIRATION");
        long Expiration = (long) Double.parseDouble(USR_EXPIRATION);
        if (Expiration < new Date().getTime())
        {
            context.setValue("XMLC_Authenticated", "0");

 xmlRequest.raiseError("XMLC_AUTHENTICATION_REQUIRED", "XMLCollectionEvents1Authenticate");
        }
        context.setValue("XMLC_Authenticated", "1");
        //     mettre ici le code de gestion des groupes et autorisations (cf. Personnaliser
 l'autorisation)
    }
    catch (Exception ex)
    {
        Database.rollback();
        throw ex;
    }
    finally
    {
        Database.commit();
    }
}
```


Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Les services d'administration sont actuellement encore sous le contrôle du fichier Security.xml, car ils sont dans un Pool XMLCAdm à part. Si vous voulez inclure ces services d'administration sous le contrôle de votre authentification, vous avez deux solutions.

La première consiste à utiliser le fichier Security.xml en le complétant des utilisateurs superviseurs qui seront autorisés à exécuter ces actions. Il faudra alors dans le processus de login mettre à jour la session générée dans le fichier security.xml. Cette méthode a pour avantage de ne pas requérir une connexion à la base de données pour le Pool XMLCAdm, ce qui permet d'avoir accès à ces actions même si on a un problème de connexion.

La seconde solution est de mettre le module contenant le composant XMLCollectionEvents et le XMLService MyLogin dans le Pool XMLCAdm ainsi que la Datasource. Il faudra aussi bien vous assurer que dans les InitParams, XMLC_PoolRedirect est à 1.



Personnaliser l'autorisation

Il nous reste à personnaliser l'autorisation, c'est-à-dire à associer les utilisateurs de la base de données aux groupes XMLRAD pour que les autorisations mises en place dans l'application s'appliquent correctement. Dans le fichier Security.xml, les groupes sont définis avec un FieldName.

```
<Security>
  <Groups>
    <Group>
      <ID>-411</ID>
      <Name>Administrators</Name>
      <Infos/>
      <FieldName>XMLC_Administrator</FieldName>
      <DirectoryGroupName/>
    </Group>
    <Group>
      <ID>-413</ID>
      <Name>Users</Name>
      <Infos/>
      <FieldName>XMLC_User</FieldName>
      <DirectoryGroupName/>
    </Group>
  </Groups>
</Security>
```

Si un utilisateur authentifié appartient à un de ces groupes, il suffit de positionner la valeur du FieldName dans le Context à 1.

Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

Nous allons rajouter une table GRP et une table USRGRP dans notre base pour stocker l'appartenance des utilisateurs a un ou plusieurs groupes.

```
CREATE TABLE GRP (
  GRP_ID INTEGER,
  GRP_NAME VARCHAR(50),
)
CREATE TABLE USRGRP (
  USRGRP_ID INTEGER,
  GRP_ID INTEGER,
  USR_ID INTEGER
)
```

Il nous faut donc créer une nouvelle requête qryListUSRGRP pour avoir la liste de groupes auxquels appartient un utilisateur.

```
SELECT GRP_ID, GRP_NAME
FROM USRGRP, GRP
WHERE USRGRP.USR_ID = :USR_ID
  AND USRGRP.GRP_ID = GRP.GRP_ID
```

Pour chaque groupe extrait de la liste, on va appeler un XMLService MapUserGroup. Créons ce XMLService (XMLGram only dans Misc.) et dans le BeforeXMLGram, on ajoute le code suivant :

Delphi

```
procédure TTrainingAdm.MapUserGroupBeforeXMLGram(XMLGram: IXMLGram; e: TBeforeXMLGramEventArgs);
begin
  if Context.GetValue('GRP_NAME') = 'Admins' then
      Context.SetValue('XMLC_Administrator', '1');
  if Context.GetValue('GRP_NAME') = 'Users' then
      Context.SetValue('XMLC_User', '1');
end;
```

C#

```
public void MapUserGroup_BeforeXMLGram(IXMLGram XMLGram, XMLComponent.TBeforeXMLGramEventArgs e)
{
    if (Context.GetValue("GRP_NAME") == "Admins")
          Context.SetValue("XMLC_Administrator", "1");
    if (Context.GetValue("GRP_NAME") == "Users")
          Context.SetValue("XMLC_User", "1");
}
```

Java

```
public void MapUserGroup_BeforeXMLGram(XMLGram xmlGram, XMLComponent.BeforeXMLGramEventArgs e)
{
    if (context.getValue("GRP_NAME").equals("Admins"))
          context.setValue("XMLC_Administrator", "1");
    if (context.getValue("GRP_NAME").equals("Users"))
          context.setValue("XMLC_User", "1");
}
```

Nous allons dans le OnAuthenticate et nous allons appeler la méthode DBForEach pour appeler le XMLService MapUserGroup pour chaque groupe extrait.

Delphi

```
  XMLCollection.DBForEach('qryListUSRGRP', 'MapUserGroup');
```

C#

```
  XMLCollection.DBForEach("qryListUSRGRP", "MapUserGroup");
```

Java

```
  xmlCollection.dbForEach("qryListUSRGRP", "MapUserGroup");
```


Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

De cette manière, la méthode CheckAuthorization validera les autorisations pour l'utilisateur. De plus, dans le document XML OutputDoc, dans la section XMLC_Params, les FieldName des groups sont ajoutées :

```
<document>
  <XMLC_Params>
    ...
    <XMLC_Supervisor/>
    <XMLC_Administrator>1<XMLC_Administrator>
    <XMLC_User>1<XMLC_User>
    <XMLC_Guest/>
    ...
  <XMLC_Params>
...
</document>
```

De sorte que, avec le XSL, vous allez pouvoir afficher ou non en fonction de ces autorisations certains éléments de la page HTML.

Conclusion

Nous savons comment fonctionne la sécurité intégrée de XMLRAD et comment la personnaliser en utilisant un annuaire d'utilisateurs se trouvant dans une base de données et non plus dans le fichier Security.xml.

On peut reprendre le principe développé ici pour se connecter à annuaire LDAP par exemple.

Copyright ® 2005 Jean-Philippe BEMPEL. Aucune reproduction, même partielle, ne peut être faite de ce site et de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.