Question
Eduard Lebedyuk · Feb 7, 2016

Best practices to store user information/settings in Caché

I'm interested in different approaches on how to store user data in Caché. I'm assuming that application uses Caché security/Caché users and not a self-made authentication system.

Several approaches, I'm familiar with:

1. Store data in Security.Users:Attributes property. Sample methods:

```
/// Set settings for User equal to Settings
ClassMethod SetSettings(Settings As %String = "", User As %String = {$Username}) As %
Status
{
 New $Namespace
 Set $Namespace = "%SYS"
 Set Properties("Attributes", "Settings") = $LB(Config)
 Set st = ##class(Security.Users).Modify(User, .Properties)
 Return st
}

/// Get settings for User
ClassMethod GetSettings(User As %String = {$Username}) As %String
{
 New $Namespace
 Set $Namespace = "%SYS"
 Set st = ##class(Security.Users).Get(User, .Properties)
 Return $ListGet($Get(Properties("Attributes", "Settings")))
}
```

Advantages:

- Easy to implement
- Easy to retrieve some system information about user

Disadvantages:

- Requires resources

  `%DB_CACHESYS:RW, %Admin_Secure:U`

2. Reference Security.Users as a property from your own class. Sample class:

```
Class Utils.UserInfo Extends %Persistent
{
```

```
Property User As Security.Users;

Property Settings As %String;

Index UserIndex On User [ IdKey, PrimaryKey, Unique ];

/// Set settings for User equal to Settings
ClassMethod SetSettings(Settings As %String = "", User As %String = {$Username}) As %
Status
{
 If ..%ExistsId(User) {
  Set settingsObj = ..%OpenId(User)
 } Else {
  Set settingsObj = ..%New()
  Set userObj = ##class(Security.Users).%OpenId(User)
  Set settingsObj.User = userObj
 }
 Set settingsObj.Settings = Settings
 Return settingsObj.%Save()
}

/// Get settings for User
ClassMethod GetSettings(User As %String = {$Username}) As %String
{
 Return ..SettingsGetStored(User)
}
}
```

Advantages:

- Does not need to open an object to retrieve settings

Disadvantages:

- Requires mapping of Security.Users class to an application namespace

3. Reference User as a string property. About the same as the previous option. Sample class:

```
Class Utils.UserInfo Extends %Persistent
{

Property User As %String;

Property Settings As %String;

Index UserIndex On User [ IdKey, PrimaryKey, Unique ];

/// Set settings for User equal to Settings
ClassMethod SetSettings(Settings As %String = "", User As %String = {$Username}) As %
Status
{

 If ..%ExistsId(User) {
  Set settingsObj = ..%OpenId(User)
 } Else {
  Set settingsObj = ..%New()
```

```
  Set settingsObj.User = User
 }
 Set settingsObj.Settings = Settings
 Return settingsObj.%Save()
}

/// Get settings for User
ClassMethod GetSettings(User As %String = {$Username}) As %String
{
 Return ..SettingsGetStored(User)
}
}
```

Advantages:

- Does not need to open an object to retrieve settings

Disadvantages:

- Invalid data may be entered, the check that username is correct would again require some form of access to Security.Users

4. Store settings in global. Sample methods:

```
/// Set settings for User equal to Settings
ClassMethod SetConfig(Config As %String = "", User As %String = {$Username}) As %Stat
us
{
 Set ^SettingsGlobal(User) = Config
 Return $$$OK
}

/// Get settings for User
ClassMethod GetConfig(App As %String, User As %String = {$Username}) As %String
{
 Return $Get(^SettingsGlobal(User))
}
```

Advantages:

- Easy to set up
- Does not need to open an object to set or retrieve settings

Disadvantages:

- Invalid data may be entered, the check that username is correct would again require some form of access to Security.Users
- No object or SQL access

So, are there any other solutions? Moreover, what is the best approach?

#Caché #Security #Authentication #Access control

Source URL:https://community.intersystems.com/post/best-practices-store-user-informationsettings-cach%C3%A9