ASPNETDB Database

Table of Contents

ASPNETDB Database

Description

ASP.NET 2.0 Provider Database

Remarks

The ASPNETDB is used by ASP.NET 2.0 providers to persist state in SQL Server. This database is typically created using the aspnet_regsql.exe tool that comes with ASP.NET.

ASP.NET 2.0 incluides the following types of providers:

- Membership
- Role Management
- Site Map
- Profile
- Session State
- Web Events
- Web Parts Personalization
- Protected Configuration

For more information about ASP.NET 2.0 Providers, please visit the following MSDN article:

Microsoft ASP.NET 2.0 Providers

See Also

Tables | Views | Procedures

Tables: ASPNETDB

Tables

Name	Description
aspnet_Applications	Used by ASP.NET features to provide an application scope for data.
aspnet_Membership	Used by the SQL Membership Provider to store membership data.
aspnet_Paths	Used by the SQL Personalization Provider to store the path for which Web Parts personalization state has been saved.
aspnet_PersonalizationAllUsers	Used by the SQL Personalization Provider to store shared personalization data.
aspnet_PersonalizationPerUser	Used by the SQL Personalization Provider to store per-user personalization data.
aspnet_Profile	Used by the SQL Profile Provider to store individual instances of property values.
aspnet_Roles	Used by the SQL Role Provider to store role data.
aspnet_SchemaVersions	Used to track the versions of schemas required by ASP.NET features.
aspnet_Users	Used to store information regarding users, including user names and IDs.
aspnet_UsersInRoles	Used by the SQL Role Provider to map roles to users.
aspnet_WebEvent_Events	Used by the SQL Web Event Provider to log event data.

Table: aspnet_Applications

Description

Used by ASP.NET features to provide an application scope for data.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationName	nvarchar(256)	Yes		Application name
LoweredApplicationName	nvarchar(256)	Yes		Application name (lowercase)
🔑 ApplicationId	uniqueidentifier	Yes	(newid())	Application ID
Description	nvarchar(256)	No		Application description

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Applications	ApplicationId	aspnet_Users	ApplicationId
aspnet_Applications	ApplicationId	aspnet_Membership	ApplicationId
aspnet_Applications	ApplicationId	aspnet_Roles	ApplicationId
aspnet_Applications	ApplicationId	aspnet_Paths	ApplicationId

Indexes

Name	Type	Columns
aspnet_Applications_Index	Non-unique, Clustered	LoweredApplicationName
PK__aspnet_Applicati__7E6CC920	Unique	ApplicationId
UQ__aspnet_Applicati__00551192	Unique	ApplicationName
UQ__aspnet_Applicati__7F60ED59	Unique	LoweredApplicationName

Referencing Views

Name
vw_aspnet_Applications

Definition

```
CREATE TABLE [aspnet_Applications]
(
    [ApplicationName] nvarchar(256) NOT NULL,
    [LoweredApplicationName] nvarchar(256) NOT NULL,
    [ApplicationId] uniqueidentifier NOT NULL DEFAULT ((newid())),
    [Description] nvarchar(256) NULL
```

Definition

```
    CONSTRAINT [PK_aspnet_Applications] PRIMARY KEY
    (
        [ApplicationId]
    )
)
```

Table: aspnet_Membership

Description

Used by the SQL Membership Provider to store membership data.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationId	uniqueidentifier	Yes		Application ID
🔑 UserId	uniqueidentifier	Yes		User ID
Password	nvarchar(128)	Yes		Password (plaintext, hashed, or encrypted; base-64-encoded if hashed or encrypted)
PasswordFormat	int	Yes	((0))	Password format (0=Plaintext, 1=Hashed, 2=Encrypted)
PasswordSalt	nvarchar(128)	Yes		Randomly generated 128-bit value used to salt password hashes; stored in base-64-encoded form
MobilePIN	nvarchar(16)	No		User's mobile PIN (currently not used)
Email	nvarchar(256)	No		User's e-mail address
LoweredEmail	nvarchar(256)	No		User's e-mail address (lowercase)
PasswordQuestion	nvarchar(256)	No		Password question
PasswordAnswer	nvarchar(128)	No		Answer to password question
IsApproved	bit	Yes		1=Approved, 0=Not approved
IsLockedOut	bit	Yes		1=Locked out, 0=Not locked out
CreateDate	datetime	Yes		Date and time this account was created
LastLoginDate	datetime	Yes		Date and time of this user's last login
LastPasswordChangedDate	datetime	Yes		Date and time this user's password was last changed
LastLockoutDate	datetime	Yes		Date and time this user was last locked out
FailedPasswordAttemptCount	int	Yes		Number of consecutive failed login attempts
FailedPasswordAttemptWindowStart	datetime	Yes		Date and time of first failed login if FailedPasswordAttemptCount is nonzero
FailedPasswordAnswerAttemptCount	int	Yes		Number of consecutive failed password answer attempts
FailedPasswordAnswerAttemptWindowStart	datetime	Yes		Date and time of first failed password answer if FailedPasswordAnswerAttemptCount is nonzero
Comment	ntext	No		Additional text

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Applications	ApplicationId	aspnet_Membership	ApplicationId
aspnet_Users	UserId	aspnet_Membership	UserId

Indexes

Name	Type	Columns
aspnet_Membership_index	Non-unique, Clustered	ApplicationId, LoweredEmail
PK__aspnet_Membershi__1367E606	Unique	UserId

Referencing Views

Name
vw_aspnet_MembershipUsers

Definition

```
CREATE TABLE [aspnet_Membership]
(
    [ApplicationId] uniqueidentifier NOT NULL,
    [UserId] uniqueidentifier NOT NULL,
    [Password] nvarchar(128) NOT NULL,
    [PasswordFormat] int NOT NULL DEFAULT (((0))),
    [PasswordSalt] nvarchar(128) NOT NULL,
    [MobilePIN] nvarchar(16) NULL,
    [Email] nvarchar(256) NULL,
    [LoweredEmail] nvarchar(256) NULL,
    [PasswordQuestion] nvarchar(256) NULL,
    [PasswordAnswer] nvarchar(128) NULL,
    [IsApproved] bit NOT NULL,
    [IsLockedOut] bit NOT NULL,
    [CreateDate] datetime NOT NULL,
    [LastLoginDate] datetime NOT NULL,
    [LastPasswordChangedDate] datetime NOT NULL,
    [LastLockoutDate] datetime NOT NULL,
    [FailedPasswordAttemptCount] int NOT NULL,
    [FailedPasswordAttemptWindowStart] datetime NOT NULL,
    [FailedPasswordAnswerAttemptCount] int NOT NULL,
    [FailedPasswordAnswerAttemptWindowStart] datetime NOT NULL,
    [Comment] ntext NULL
    CONSTRAINT [PK_aspnet_Membership] PRIMARY KEY
    (
        [UserId]
    )
)
```

Table: aspnet_Paths

Description

Used by the SQL Personalization Provider to store the path for which Web Parts personalization state has been saved.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationId	uniqueidentifier	Yes		Application ID
PathId	uniqueidentifier	Yes	(newid())	Path ID
Path	nvarchar(256)	Yes		Path name
LoweredPath	nvarchar(256)	Yes		Path name (lowercase)

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Applications	ApplicationId	aspnet_Paths	ApplicationId
aspnet_Paths	PathId	aspnet_PersonalizationAllUsers	PathId
aspnet_Paths	PathId	aspnet_PersonalizationPerUser	PathId

Indexes

Name	Type	Columns
aspnet_Paths_index	Unique, Clustered	ApplicationId, LoweredPath
PK__aspnet_Paths__44FF419A	Unique	PathId

Referencing Views

Name
vw_aspnet_WebPartState_Paths

Definition

```
CREATE TABLE [aspnet_Paths]
(
    [ApplicationId] uniqueidentifier NOT NULL,
    [PathId] uniqueidentifier NOT NULL DEFAULT ((newid())),
    [Path] nvarchar(256) NOT NULL,
    [LoweredPath] nvarchar(256) NOT NULL
    CONSTRAINT [PK_aspnet_Paths] PRIMARY KEY
    (
        [PathId]
    )
)
```

Table: aspnet_PersonalizationAllUsers

Description

Used by the SQL Personalization Provider to store shared personalization data.

Columns

Name	Type	Required?	Defaults To	Description
🔑 PathId	uniqueidentifier	Yes		ID of the virtual path to which this state pertains
PageSettings	image	Yes		Serialized personalization state
LastUpdatedDate	datetime	Yes		Date and time state was saved

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Paths	PathId	aspnet_PersonalizationAllUsers	PathId

Indexes

Name	Type	Columns
PK__aspnet_Personali__4AB81AF0	Unique, Clustered	PathId

Referencing Views

Name
vw_aspnet_WebPartState_Shared

Definition

```
CREATE TABLE [aspnet_PersonalizationAllUsers]
(
   [PathId] uniqueidentifier NOT NULL,
   [PageSettings] image NOT NULL,
   [LastUpdatedDate] datetime NOT NULL
   CONSTRAINT [PK_aspnet_PersonalizationAllUsers] PRIMARY KEY
   (
      [PathId]
   )
)
```

Table: aspnet_PersonalizationPerUser

Description

Used by the SQL Personalization Provider to store per-user personalization data.

Columns

Name	Type	Required?	Defaults To	Description
🔑 Id	uniqueidentifier	Yes	(newid())	ID of this record
PathId	uniqueidentifier	No		ID of the virtual path to which this state pertains
UserId	uniqueidentifier	No		ID of the user to which this state pertains
PageSettings	image	Yes		Serialized personalization state
LastUpdatedDate	datetime	Yes		Date and time state was saved

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Paths	PathId	aspnet_PersonalizationPerUser	PathId
aspnet_Users	UserId	aspnet_PersonalizationPerUser	UserId

Indexes

Name	Type	Columns
aspnet_PersonalizationPerUser_index1	Unique, Clustered	PathId, UserId
aspnet_PersonalizationPerUser_ncindex2	Unique	UserId, PathId
PK__aspnet_Personali__4D94879B	Unique	Id

Referencing Views

Name
vw_aspnet_WebPartState_User

Definition

```
CREATE TABLE [aspnet_PersonalizationPerUser]
(
   [Id] uniqueidentifier NOT NULL DEFAULT ((newid())),
   [PathId] uniqueidentifier NULL,
   [UserId] uniqueidentifier NULL,
   [PageSettings] image NOT NULL,
   [LastUpdatedDate] datetime NOT NULL
```

Definition

```
    CONSTRAINT [PK_aspnet_PersonalizationPerUser] PRIMARY KEY
    (
        [Id]
    )
)
```

Table: aspnet_Profile

Description

Used by the SQL Profile Provider to store individual instances of property values.

Columns

Name	Type	Required?	Defaults To	Description
🔑 UserId	uniqueidentifier	Yes		ID of the user to which this profile data pertains
PropertyNames	ntext	Yes		Names of all property values stored in this profile
PropertyValuesString	ntext	Yes		Values of properties that could be persisted as text
PropertyValuesBinary	image	Yes		Values of properties that were configured to use binary serialization
LastUpdatedDate	datetime	Yes		Date and time this profile was last updated

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Users	UserId	aspnet_Profile	UserId

Indexes

Name	Type	Columns
PK__aspnet_Profile__286302EC	Unique, Clustered	UserId

Referencing Views

Name
vw_aspnet_Profiles

Definition

```
CREATE TABLE [aspnet_Profile]
(
    [UserId] uniqueidentifier NOT NULL,
    [PropertyNames] ntext NOT NULL,
    [PropertyValuesString] ntext NOT NULL,
    [PropertyValuesBinary] image NOT NULL,
    [LastUpdatedDate] datetime NOT NULL
    CONSTRAINT [PK_aspnet_Profile] PRIMARY KEY
    (
        [UserId]
    )
)
```

Table: aspnet_Roles

Description

Used by the SQL Role Provider to store role data.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationId	uniqueidentifier	Yes		Application ID
🔑 RoleId	uniqueidentifier	Yes	(newid())	Role ID
RoleName	nvarchar(256)	Yes		Role name
LoweredRoleName	nvarchar(256)	Yes		Role name (lowercase)
Description	nvarchar(256)	No		Role description (currently unused)

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Applications	ApplicationId	aspnet_Roles	ApplicationId
aspnet_Roles	RoleId	aspnet_UsersInRoles	RoleId

Indexes

Name	Type	Columns
aspnet_Roles_index1	Unique, Clustered	ApplicationId, LoweredRoleName
PK__aspnet_Roles__31EC6D26	Unique	RoleId

Referencing Views

Name
vw_aspnet_Roles

Definition

```
CREATE TABLE [aspnet_Roles]
(
   [ApplicationId] uniqueidentifier NOT NULL,
   [RoleId] uniqueidentifier NOT NULL DEFAULT ((newid())),
   [RoleName] nvarchar(256) NOT NULL,
   [LoweredRoleName] nvarchar(256) NOT NULL,
   [Description] nvarchar(256) NULL
   CONSTRAINT [PK_aspnet_Roles] PRIMARY KEY
   (
      [RoleId]
   )
)
```

Table: aspnet_SchemaVersions

Description

Used to track the versions of schemas required by ASP.NET features.

Columns

Name	Type	Required?	Defaults To	Description
🔑 Feature	nvarchar(128)	Yes		Name of the application feature
🔑 CompatibleSchemaVersion	nvarchar(128)	Yes		Schema version required for compatibility
IsCurrentVersion	bit	Yes		1=Current version, 0=Not current version

Indexes

Name	Type	Columns
PK__aspnet_SchemaVer__08EA5793	Unique, Clustered	Feature, CompatibleSchemaVersion

Definition

```
CREATE TABLE [aspnet_SchemaVersions]
(
   [Feature] nvarchar(128) NOT NULL,
   [CompatibleSchemaVersion] nvarchar(128) NOT NULL,
   [IsCurrentVersion] bit NOT NULL
   CONSTRAINT [PK_aspnet_SchemaVersions] PRIMARY KEY
   (
      [Feature],
      [CompatibleSchemaVersion]
   )
)
```

Table: aspnet_Users

Description

Used to store information regarding users, including user names and IDs.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationId	uniqueidentifier	Yes		Application ID
🔑 UserId	uniqueidentifier	Yes	(newid())	User ID
UserName	nvarchar(256)	Yes		User name
LoweredUserName	nvarchar(256)	Yes		User name (lowercase)
MobileAlias	nvarchar(16)	No	(NULL)	User's mobile alias (currently not used)
IsAnonymous	bit	Yes	((0))	1=Anonymous user, 0=Not an anonymous user
LastActivityDate	datetime	Yes		Date and time of last activity by this user

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Applications	ApplicationId	aspnet_Users	ApplicationId
aspnet_Users	UserId	aspnet_Membership	UserId
aspnet_Users	UserId	aspnet_Profile	UserId
aspnet_Users	UserId	aspnet_UsersInRoles	UserId
aspnet_Users	UserId	aspnet_PersonalizationPerUser	UserId

Indexes

Name	Type	Columns
aspnet_Users_Index	Unique, Clustered	ApplicationId, LoweredUserName
aspnet_Users_Index2	Non-unique	ApplicationId, LastActivityDate
PK__aspnet_Users__03317E3D	Unique	UserId

Referencing Views

Name
vw_aspnet_MembershipUsers
vw_aspnet_Users

Definition

```
CREATE TABLE [aspnet_Users]
(
   [ApplicationId] uniqueidentifier NOT NULL,
   [UserId] uniqueidentifier NOT NULL DEFAULT ((newid())),
   [UserName] nvarchar(256) NOT NULL,
   [LoweredUserName] nvarchar(256) NOT NULL,
   [MobileAlias] nvarchar(16) NULL DEFAULT ((NULL)),
   [IsAnonymous] bit NOT NULL DEFAULT (((0))),
   [LastActivityDate] datetime NOT NULL
   CONSTRAINT [PK_aspnet_Users] PRIMARY KEY
   (
      [UserId]
   )
)
```

Table: aspnet_UsersInRoles

Description

Used by the SQL Role Provider to map roles to users.

Columns

Name	Type	Required?	Defaults To	Description
🔑 UserId	uniqueidentifier	Yes		User ID
🔑 RoleId	uniqueidentifier	Yes		Role ID

Relationships

Primary Table	Primary Key	Foreign Table	Foreign Key
aspnet_Users	UserId	aspnet_UsersInRoles	UserId
aspnet_Roles	RoleId	aspnet_UsersInRoles	RoleId

Indexes

Name	Type	Columns
aspnet_UsersInRoles_index	Non-unique	RoleId
PK__aspnet_UsersInRo__35BCFE0A	Unique, Clustered	UserId, RoleId

Referencing Views

Name
vw_aspnet_UsersInRoles

Definition

```
CREATE TABLE [aspnet_UsersInRoles]
(
   [UserId] uniqueidentifier NOT NULL,
   [RoleId] uniqueidentifier NOT NULL
   CONSTRAINT [PK_aspnet_UsersInRoles] PRIMARY KEY
   (
      [UserId],
      [RoleId]
   )
)
```

Table: aspnet_WebEvent_Events

Description

Used by the SQL Web Event Provider to log event data.

Columns

Name	Type	Required?	Defaults To	Description
🔑 EventId	char(32)	Yes		Event ID (from WebBaseEvent.EventId)
EventTimeUtc	datetime	Yes		UTC time at which the event was fired (from WebBaseEvent.EventTimeUtc)
EventTime	datetime	Yes		Local time at which the event was fired (from WebBaseEvent.EventTime)
EventType	nvarchar(256)	Yes		Event type (for example, WebFailureAuditEvent)
EventSequence	decimal(19)	Yes		Event sequence number (from WebBaseEvent.EventSequence)
EventOccurrence	decimal(19)	Yes		Event occurrence count (from WebBaseEvent.EventOccurrence)
EventCode	int	Yes		Event code (from WebBaseEvent.EventCode)
EventDetailCode	int	Yes		Event detail code (from WebBaseEvent.EventDetailCode)
Message	nvarchar(1024)	No		Event message (from WebBaseEvent.EventMessage)
ApplicationPath	nvarchar(256)	No		Physical path of the application that generated the Web event (for example, C:\Websites\MyApp)
ApplicationVirtualPath	nvarchar(256)	No		Virtual path of the application that generated the event (for example, /MyApp)
MachineName	nvarchar(256)	Yes		Name of the machine on which the event was generated
RequestUrl	nvarchar(1024)	No		URL of the request that generated the Web event
ExceptionType	nvarchar(256)	No		If the Web event is a WebBaseErrorEvent, type of exception recorded in the ErrorException property; otherwise, DBNull
Details	ntext	No		Text generated by calling ToString on the Web event

Indexes

Name	Type	Columns
PK__aspnet_WebEvent___5FB337D6	Unique, Clustered	EventId

Definition

```
CREATE TABLE [aspnet_WebEvent_Events]
(
    [EventId] char(32) NOT NULL,
    [EventTimeUtc] datetime NOT NULL,
    [EventTime] datetime NOT NULL,
    [EventType] nvarchar(256) NOT NULL,
    [EventSequence] decimal(19) NOT NULL,
    [EventOccurrence] decimal(19) NOT NULL,
    [EventCode] int NOT NULL,
    [EventDetailCode] int NOT NULL,
    [Message] nvarchar(1024) NULL,
    [ApplicationPath] nvarchar(256) NULL,
    [ApplicationVirtualPath] nvarchar(256) NULL,
    [MachineName] nvarchar(256) NOT NULL,
    [RequestUrl] nvarchar(1024) NULL,
    [ExceptionType] nvarchar(256) NULL,
    [Details] ntext NULL
    CONSTRAINT [PK_aspnet_WebEvent_Events] PRIMARY KEY
    (
        [EventId]
    )
)
```

Views: ASPNETDB

Views

Name	Description
vw_aspnet_Applications	Displays information for all applications.
vw_aspnet_MembershipUsers	Displays a list of ASP.NET membership users associated with the unique identifier for the user.
vw_aspnet_Profiles	Displays user profile information.
vw_aspnet_Roles	Displays role information.
vw_aspnet_Users	Displays a list of users per application.
vw_aspnet_UsersInRoles	Displays which users are associated with which roles by the unique identifiers for the user and the role.
vw_aspnet_WebPartState_Paths	Displays Web Parts state path information.
vw_aspnet_WebPartState_Shared	Displays Web Parts state information.
vw_aspnet_WebPartState_User	Displays Web Parts user information.

View: vw_aspnet_Applications

Description

Displays information for all applications.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationName	nvarchar	Yes		Application name
LoweredApplicationName	nvarchar	Yes		Application name (lowercase)
ApplicationId	uniqueidentifier	Yes		Application ID
Description	nvarchar	No		Application description

Tables Referenced

Name
aspnet_Applications

Definition

```
SELECT [dbo].[aspnet_Applications].[ApplicationName],
[dbo].[aspnet_Applications].[LoweredApplicationName], [dbo].[aspnet_Applications].[ApplicationId],
[dbo].[aspnet_Applications].[Description]
  FROM [dbo].[aspnet_Applications]
```

View: vw_aspnet_MembershipUsers

Description

Displays a list of ASP.NET membership users associated with the unique identifier for the user.

Columns

Name	Type	Required?	Defaults To	Description
UserId	uniqueidentifier	Yes		User ID
PasswordFormat	int	Yes		Password format (0=Plaintext, 1=Hashed, 2=Encrypted)
MobilePIN	nvarchar	No		User's mobile PIN (currently not used)
Email	nvarchar	No		User's e-mail address
LoweredEmail	nvarchar	No		User's e-mail address (lowercase)
PasswordQuestion	nvarchar	No		Password question
PasswordAnswer	nvarchar	No		Answer to password question
IsApproved	bit	Yes		1=Approved, 0=Not approved
IsLockedOut	bit	Yes		1=Locked out, 0=Not locked out
CreateDate	datetime	Yes		Date and time this account was created
LastLoginDate	datetime	Yes		Date and time of this user's last login
LastPasswordChangedDate	datetime	Yes		Date and time this user's password was last changed
LastLockoutDate	datetime	Yes		Date and time this user was last locked out
FailedPasswordAttemptCount	int	Yes		Number of consecutive failed login attempts
FailedPasswordAttemptWindowStart	datetime	Yes		Date and time of first failed login if FailedPasswordAttemptCount is nonzero
FailedPasswordAnswerAttemptCount	int	Yes		Number of consecutive failed password answer attempts
FailedPasswordAnswerAttemptWindowStart	datetime	Yes		Date and time of first failed password answer if FailedPasswordAnswerAttemptCount is nonzero
Comment	ntext	No		Additional text
ApplicationId	uniqueidentifier	Yes		Application ID
UserName	nvarchar	Yes		User name
MobileAlias	nvarchar	No		User's mobile alias (currently not used)
IsAnonymous	bit	Yes		1=Anonymous user, 0=Not an anonymous user

Columns

Name	Type	Required?	Defaults To	Description
LastActivityDate	datetime	Yes		Date and time of last activity by this user

Tables Referenced

Name
aspnet_Membership
aspnet_Users

Definition

```
SELECT [dbo].[aspnet_Membership].[UserId],
            [dbo].[aspnet_Membership].[PasswordFormat],
            [dbo].[aspnet_Membership].[MobilePIN],
            [dbo].[aspnet_Membership].[Email],
            [dbo].[aspnet_Membership].[LoweredEmail],
            [dbo].[aspnet_Membership].[PasswordQuestion],
            [dbo].[aspnet_Membership].[PasswordAnswer],
            [dbo].[aspnet_Membership].[IsApproved],
            [dbo].[aspnet_Membership].[IsLockedOut],
            [dbo].[aspnet_Membership].[CreateDate],
            [dbo].[aspnet_Membership].[LastLoginDate],
            [dbo].[aspnet_Membership].[LastPasswordChangedDate],
            [dbo].[aspnet_Membership].[LastLockoutDate],
            [dbo].[aspnet_Membership].[FailedPasswordAttemptCount],
            [dbo].[aspnet_Membership].[FailedPasswordAttemptWindowStart],
            [dbo].[aspnet_Membership].[FailedPasswordAnswerAttemptCount],
            [dbo].[aspnet_Membership].[FailedPasswordAnswerAttemptWindowStart],
            [dbo].[aspnet_Membership].[Comment],
            [dbo].[aspnet_Users].[ApplicationId],
            [dbo].[aspnet_Users].[UserName],
            [dbo].[aspnet_Users].[MobileAlias],
            [dbo].[aspnet_Users].[IsAnonymous],
            [dbo].[aspnet_Users].[LastActivityDate]
  FROM [dbo].[aspnet_Membership] INNER JOIN [dbo].[aspnet_Users]
      ON [dbo].[aspnet_Membership].[UserId] = [dbo].[aspnet_Users].[UserId]
```

View: vw_aspnet_Profiles

Description

Displays user profile information.

Columns

Name	Type	Required?	Defaults To	Description
UserId	uniqueidentifier	Yes		ID of the user to which this profile data pertains
LastUpdatedDate	datetime	Yes		Date and time this profile was last updated
DataSize	int	No		Size of the profile data

Tables Referenced

Name
aspnet_Profile

Definition

```
SELECT [dbo].[aspnet_Profile].[UserId], [dbo].[aspnet_Profile].[LastUpdatedDate],
      [DataSize]=  DATALENGTH([dbo].[aspnet_Profile].[PropertyNames])
                 + DATALENGTH([dbo].[aspnet_Profile].[PropertyValuesString])
                 + DATALENGTH([dbo].[aspnet_Profile].[PropertyValuesBinary])
  FROM [dbo].[aspnet_Profile]
```

View: vw_aspnet_Roles

Description

Displays role information.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationId	uniqueidentifier	Yes		Application ID
RoleId	uniqueidentifier	Yes		Role ID
RoleName	nvarchar	Yes		Role name
LoweredRoleName	nvarchar	Yes		Role name (lowercase)
Description	nvarchar	No		Role description (currently unused)

Tables Referenced

Name
aspnet_Roles

Definition

```
SELECT [dbo].[aspnet_Roles].[ApplicationId], [dbo].[aspnet_Roles].[RoleId], [dbo].[aspnet_Roles].[RoleName],
[dbo].[aspnet_Roles].[LoweredRoleName], [dbo].[aspnet_Roles].[Description]
  FROM [dbo].[aspnet_Roles]
```

View: vw_aspnet_Users

Description

Displays a list of users per application.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationId	uniqueidentifier	Yes		Application ID
UserId	uniqueidentifier	Yes		User ID
UserName	nvarchar	Yes		User name
LoweredUserName	nvarchar	Yes		User name (lowercase)
MobileAlias	nvarchar	No		User's mobile alias (currently not used)
IsAnonymous	bit	Yes		1=Anonymous user, 0=Not an anonymous user
LastActivityDate	datetime	Yes		Date and time of last activity by this user

Tables Referenced

Name
aspnet_Users

Definition

```
SELECT [dbo].[aspnet_Users].[ApplicationId], [dbo].[aspnet_Users].[UserId], [dbo].[aspnet_Users].[UserName],
[dbo].[aspnet_Users].[LoweredUserName], [dbo].[aspnet_Users].[MobileAlias],
[dbo].[aspnet_Users].[IsAnonymous], [dbo].[aspnet_Users].[LastActivityDate]
  FROM [dbo].[aspnet_Users]
```

View: vw_aspnet_UsersInRoles

Description

Displays which users are associated with which roles by the unique identifiers for the user and the role.

Columns

Name	Type	Required?	Defaults To	Description
UserId	uniqueidentifier	Yes		User ID
RoleId	uniqueidentifier	Yes		Role ID

Tables Referenced

Name
aspnet_UsersInRoles

Definition

```
SELECT [dbo].[aspnet_UsersInRoles].[UserId], [dbo].[aspnet_UsersInRoles].[RoleId]
  FROM [dbo].[aspnet_UsersInRoles]
```

View: vw_aspnet_WebPartState_Paths

Description

Displays Web Parts state path information.

Columns

Name	Type	Required?	Defaults To	Description
ApplicationId	uniqueidentifier	Yes		Application ID
PathId	uniqueidentifier	Yes		Path ID
Path	nvarchar	Yes		Path name
LoweredPath	nvarchar	Yes		Path name (lowercase)

Tables Referenced

Name
aspnet_Paths

Definition

```
SELECT [dbo].[aspnet_Paths].[ApplicationId], [dbo].[aspnet_Paths].[PathId], [dbo].[aspnet_Paths].[Path],
[dbo].[aspnet_Paths].[LoweredPath]
  FROM [dbo].[aspnet_Paths]
```

View: vw_aspnet_WebPartState_Shared

Description

Displays Web Parts state information.

Columns

Name	Type	Required?	Defaults To	Description
PathId	uniqueidentifier	Yes		ID of the virtual path to which this state pertains
DataSize	int	No		Size of the data
LastUpdatedDate	datetime	Yes		Date and time state was saved

Tables Referenced

Name
aspnet_PersonalizationAllUsers

Definition

```
SELECT [dbo].[aspnet_PersonalizationAllUsers].[PathId],
[DataSize]=DATALENGTH([dbo].[aspnet_PersonalizationAllUsers].[PageSettings]),
[dbo].[aspnet_PersonalizationAllUsers].[LastUpdatedDate]
  FROM [dbo].[aspnet_PersonalizationAllUsers]
```

View: vw_aspnet_WebPartState_User

Description

Displays Web Parts user information.

Columns

Name	Type	Required?	Defaults To	Description
PathId	uniqueidentifier	No		ID of the virtual path to which this state
UserId	uniqueidentifier	No		ID of the user to which this state pertains
DataSize	int	No		Size of the user-scoped data
LastUpdatedDate	datetime	Yes		Date and time state was saved

Tables Referenced

Name
aspnet_PersonalizationPerUser

Definition

```
SELECT [dbo].[aspnet_PersonalizationPerUser].[PathId], [dbo].[aspnet_PersonalizationPerUser].[UserId],
[DataSize]=DATALENGTH([dbo].[aspnet_PersonalizationPerUser].[PageSettings]),
[dbo].[aspnet_PersonalizationPerUser].[LastUpdatedDate]
  FROM [dbo].[aspnet_PersonalizationPerUser]
```

Procedures: ASPNETDB

Procedures

Name	Description
aspnet_AnyDataInTables	Checks to see if there is any data in the specified tables.
aspnet_Applications_CreateApplication	Adds a new application to the aspnet_Application table.
aspnet_CheckSchemaVersion	Checks the compatibility of the schema version for the given feature.
aspnet_Membership_ChangePasswordQuestionAndAnswer	Changes the specified user's password question and answer.
aspnet_Membership_CreateUser	Adds a new membership user to the membership database. Records the user in the aspnet_Users and aspnet_Membership tables and, if necessary, adds a new application to the aspnet_Applications table.
aspnet_Membership_FindUsersByEmail	Retrieves records from aspnet_Membership table with email addresses matching the specified pattern and with the specified application ID.
aspnet_Membership_FindUsersByName	Retrieves records from aspnet_Membership table with user names matching the specified pattern and with the specified application ID.
aspnet_Membership_GetAllUsers	Retrieves all users from the aspnet_Membership table with the specified application ID.
aspnet_Membership_GetNumberOfUsersOnline	Gets the number of users currently online (those whose last activity dates.
aspnet_Membership_GetPassword	Gets the specified user's password data from the database. Used for retrieving passwords with a user-supplied password answer.
aspnet_Membership_GetPasswordWithFormat	Gets the specified user's password from the database. Used by the provider to retrieve passwords for performing password comparisons (for example, when ValidateUser needs to validate a password).
aspnet_Membership_GetUserByEmail	Given an e-mail address and application ID, retrieves the corresponding record from the aspnet_Membership table.
aspnet_Membership_GetUserByName	Given a user name and application ID, retrieves the corresponding record from the aspnet_Membership table.
aspnet_Membership_GetUserByUserId	Given a user ID and application ID, retrieves the corresponding record from the aspnet_Membership table.
aspnet_Membership_ResetPassword	Resets the specified user's password based on a password answer.
aspnet_Membership_SetPassword	Sets the specified user's password to the password input to the stored procedure.
aspnet_Membership_UnlockUser	Restores login privileges for the specified user by setting the user's IsLockedOut bit to 0.
aspnet_Membership_UpdateUser	Updates the user's last activity date in the aspnet_Users table and e-mail address, comment, isapproved status, and last login date in the aspnet_Membership table.
aspnet_Membership_UpdateUserInfo	Updates account locking data for the specified user in the aspnet_Users and aspnet_Membership tables. Used in conjunction with provider methods that track bad password and bad password-answer attempts.
aspnet_Paths_CreatePath	Retrieves a path ID from the aspnet_Paths table, or creates a new one if the specified path doesn't exist.

Procedures

Name	Description
aspnet_Personalization_GetApplicationId	Converts the application name input to it into an application ID.
aspnet_PersonalizationAdministration_DeleteAllState	Deletes all records from aspnet_PersonalizationAllUsers or aspnet_PersonalizationPerUser corresponding to the specified application ID.
aspnet_PersonalizationAdministration_FindState	Retrieves profile data from aspnet_PersonalizationAllUsers or aspnet_PersonalizationPerUser meeting several input criteria.
aspnet_PersonalizationAdministration_GetCountOfState	Returns a count of records in the aspnet_PersonalizationAllUsers table with path names matching the specified pattern, or a count of records in the aspnet_PersonalizationPerUser table meeting several input criteria.
aspnet_PersonalizationAdministration_ResetSharedState	Resets shared state for the specified page, by deleting the corresponding record from the aspnet_PersonalizationAllUsers table.
aspnet_PersonalizationAdministration_ResetUserState	Resets per-user state for the specified user and the specified page, by deleting the corresponding record from the aspnet_PersonalizationPerUser table. Can also delete records, based on the user's last activity date if it falls on or before the specified date.
aspnet_PersonalizationAllUsers_GetPageSettings	Retrieves shared state for the specified page from the aspnet_PersonalizationAllUsers table.
aspnet_PersonalizationAllUsers_ResetPageSettings	Resets shared state for the specified page, by deleting the corresponding record from the aspnet_PersonalizationAllUsers table.
aspnet_PersonalizationAllUsers_SetPageSettings	Saves shared state for the specified page in the aspnet_PersonalizationAllUsers table.
aspnet_PersonalizationPerUser_GetPageSettings	Retrieves per-user state for the specified page and the specified user from the aspnet_PersonalizationPerUser table.
aspnet_PersonalizationPerUser_ResetPageSettings	Resets per-user state for the specified page and the specified user, by deleting the corresponding record from the aspnet_PersonalizationPerUser table.
aspnet_PersonalizationPerUser_SetPageSettings	Saves per-user state for the specified page and the specified user in the aspnet_PersonalizationPerUser table.
aspnet_Profile_DeleteInactiveProfiles	Deletes profile data from the aspnet_Profile table for users whose last activity dates in the aspnet_Users table fall on or before the specified date.
aspnet_Profile_DeleteProfiles	Deletes profile data from the aspnet_Profile table for the specified users.
aspnet_Profile_GetNumberOfInactiveProfiles	Queries the aspnet_Profile table to get a count of profiles whose last activity dates (in the aspnet_Users table) fall on or before the specified date.
aspnet_Profile_GetProfiles	Retrieves profile data from the aspnet_Profile table for users who match the criteria input to the stored procedure.
aspnet_Profile_GetProperties	Retrieves profile data for the specified user.
aspnet_Profile_SetProperties	Saves profile data for the specified user.
aspnet_RegisterSchemaVersion	Registers the compatible schema required for the given feature.
aspnet_Roles_CreateRole	Adds a role to the aspnet_Roles table and, if necessary, adds a new application to the aspnet_Applications table.

Procedures

Name	Description
aspnet_Roles_DeleteRole	Removes a role from the aspnet_Roles table. Optionally deletes records referencing the deleted role from the aspnet_UsersInRoles table.
aspnet_Roles_GetAllRoles	Retrieves all roles with the specified application ID from the aspnet_Roles table.
aspnet_Roles_RoleExists	Checks the aspnet_Roles table to determine whether the specified role exists.
aspnet_Setup_RemoveAllRoleMembers	Removes all roles from the given SQL account.
aspnet_Setup_RestorePermissions	Restores permissions to the given SQL account.
aspnet_UnRegisterSchemaVersion	Unregisters the schema version for the given feature.
aspnet_Users_CreateUser	Adds a user to the aspnet_Users table. Called by aspnet_Membership_CreateUser.
aspnet_Users_DeleteUser	Deletes a user from the aspnet_Membership table and optionally from other SQL provider tables, including aspnet_Users.
aspnet_UsersInRoles_AddUsersToRoles	Adds the specified users to the specified roles by adding them to the aspnet_UsersInRoles table.
aspnet_UsersInRoles_FindUsersInRole	Queries the aspnet_UsersInRoles table for all users belonging to the specified role whose user names match the specified pattern.
aspnet_UsersInRoles_GetRolesForUser	Queries the aspnet_UsersInRoles table for all roles assigned to a specified user.
aspnet_UsersInRoles_GetUsersInRoles	Queries the aspnet_UsersInRoles table for all users belonging to the specified role.
aspnet_UsersInRoles_IsUserInRole	Checks the aspnet_UsersInRoles table to determine whether the specified user belongs to the specified role.
aspnet_UsersInRoles_RemoveUsersFromRoles	Removes the specified users from the specified roles by deleting the corresponding records from the aspnet_UsersInRoles table.
aspnet_WebEvent_LogEvent	Records a Web event in the aspnet_WebEvents_Events table.

Procedure: aspnet_AnyDataInTables

Description

Checks to see if there is any data in the specified tables.

Parameters

Name	Type	Direction
@TablesToCheck	int	Input

Definition

```
CREATE PROCEDURE [dbo].aspnet_AnyDataInTables
    @TablesToCheck int
AS
BEGIN
    -- Check Membership table if (@TablesToCheck & 1) is set
    IF ((@TablesToCheck & 1) <> 0 AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'vw_aspnet_MembershipUsers') AND
(type = 'V'))))
    BEGIN
        IF (EXISTS(SELECT TOP 1 UserId FROM dbo.aspnet_Membership))
        BEGIN
            SELECT N'aspnet_Membership'
            RETURN
        END
    END

    -- Check aspnet_Roles table if (@TablesToCheck & 2) is set
    IF ((@TablesToCheck & 2) <> 0  AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'vw_aspnet_Roles') AND (type = 'V'))) )
    BEGIN
        IF (EXISTS(SELECT TOP 1 RoleId FROM dbo.aspnet_Roles))
        BEGIN
            SELECT N'aspnet_Roles'
            RETURN
        END
    END

    -- Check aspnet_Profile table if (@TablesToCheck & 4) is set
    IF ((@TablesToCheck & 4) <> 0  AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'vw_aspnet_Profiles') AND (type = 'V'))) )
    BEGIN
        IF (EXISTS(SELECT TOP 1 UserId FROM dbo.aspnet_Profile))
        BEGIN
            SELECT N'aspnet_Profile'
            RETURN
        END
    END

    -- Check aspnet_PersonalizationPerUser table if (@TablesToCheck & 8) is set
    IF ((@TablesToCheck & 8) <> 0  AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'vw_aspnet_WebPartState_User') AND
(type = 'V'))) )
    BEGIN
        IF (EXISTS(SELECT TOP 1 UserId FROM dbo.aspnet_PersonalizationPerUser))
        BEGIN
            SELECT N'aspnet_PersonalizationPerUser'
```

Definition

```
            RETURN
        END
    END

    -- Check aspnet_PersonalizationPerUser table if (@TablesToCheck & 16) is set
    IF ((@TablesToCheck & 16) <> 0  AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'aspnet_WebEvent_LogEvent') AND (type
= 'P'))) )
    BEGIN
        IF (EXISTS(SELECT TOP 1 * FROM dbo.aspnet_WebEvent_Events))
        BEGIN
            SELECT N'aspnet_WebEvent_Events'
            RETURN
        END
    END

    -- Check aspnet_Users table if (@TablesToCheck & 1,2,4 & 8) are all set
    IF ((@TablesToCheck & 1) <> 0 AND
        (@TablesToCheck & 2) <> 0 AND
        (@TablesToCheck & 4) <> 0 AND
        (@TablesToCheck & 8) <> 0 AND
        (@TablesToCheck & 32) <> 0 AND
        (@TablesToCheck & 128) <> 0 AND
        (@TablesToCheck & 256) <> 0 AND
        (@TablesToCheck & 512) <> 0 AND
        (@TablesToCheck & 1024) <> 0)
    BEGIN
        IF (EXISTS(SELECT TOP 1 UserId FROM dbo.aspnet_Users))
        BEGIN
            SELECT N'aspnet_Users'
            RETURN
        END
        IF (EXISTS(SELECT TOP 1 ApplicationId FROM dbo.aspnet_Applications))
        BEGIN
            SELECT N'aspnet_Applications'
            RETURN
        END
    END
END
```

Procedure: aspnet_Applications_CreateApplication

Description

Adds a new application to the aspnet_Application table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@ApplicationId	uniqueidentifier	Input/Output

Definition

```
CREATE PROCEDURE [dbo].aspnet_Applications_CreateApplication
    @ApplicationName      nvarchar(256),
    @ApplicationId        uniqueidentifier OUTPUT
AS
BEGIN
    SELECT  @ApplicationId = ApplicationId FROM dbo.aspnet_Applications WHERE
LOWER(@ApplicationName) = LoweredApplicationName

    IF(@ApplicationId IS NULL)
    BEGIN
        DECLARE @TranStarted   bit
        SET @TranStarted = 0

        IF( @@TRANCOUNT = 0 )
        BEGIN
                BEGIN TRANSACTION
                SET @TranStarted = 1
        END
        ELSE
            SET @TranStarted = 0

        SELECT  @ApplicationId = ApplicationId
        FROM dbo.aspnet_Applications WITH (UPDLOCK, HOLDLOCK)
        WHERE LOWER(@ApplicationName) = LoweredApplicationName

        IF(@ApplicationId IS NULL)
        BEGIN
            SELECT  @ApplicationId = NEWID()
            INSERT  dbo.aspnet_Applications (ApplicationId, ApplicationName, LoweredApplicationName)
            VALUES  (@ApplicationId, @ApplicationName, LOWER(@ApplicationName))
        END


        IF( @TranStarted = 1 )
        BEGIN
            IF(@@ERROR = 0)
            BEGIN
                SET @TranStarted = 0
                COMMIT TRANSACTION
            END
            ELSE
            BEGIN
                SET @TranStarted = 0
                ROLLBACK TRANSACTION
            END
```

Definition

```
        END
    END
END
```

Procedure: aspnet_CheckSchemaVersion

Description

Checks the compatibility of the schema version for the given feature.

Parameters

Name	Type	Direction
@Feature	nvarchar	Input
@CompatibleSchemaVersion	nvarchar	Input

Definition

```
CREATE PROCEDURE [dbo].aspnet_CheckSchemaVersion
    @Feature                   nvarchar(128),
    @CompatibleSchemaVersion   nvarchar(128)
AS
BEGIN
    IF (EXISTS( SELECT  *
                FROM    dbo.aspnet_SchemaVersions
                WHERE   Feature = LOWER( @Feature ) AND
                        CompatibleSchemaVersion = @CompatibleSchemaVersion ))
        RETURN 0

    RETURN 1
END
```

Procedure: aspnet_Membership_ChangePasswordQuestionAndAnswer

Description

Changes the specified user's password question and answer.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@NewPasswordQuestion	nvarchar	Input
@NewPasswordAnswer	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_ChangePasswordQuestionAndAnswer
    @ApplicationName       nvarchar(256),
    @UserName              nvarchar(256),
    @NewPasswordQuestion   nvarchar(256),
    @NewPasswordAnswer     nvarchar(128)
AS
BEGIN
    DECLARE @UserId uniqueidentifier
    SELECT  @UserId = NULL
    SELECT  @UserId = u.UserId
    FROM    dbo.aspnet_Membership m, dbo.aspnet_Users u, dbo.aspnet_Applications a
    WHERE   LoweredUserName = LOWER(@UserName) AND
            u.ApplicationId = a.ApplicationId  AND
            LOWER(@ApplicationName) = a.LoweredApplicationName AND
            u.UserId = m.UserId
    IF (@UserId IS NULL)
    BEGIN
        RETURN(1)
    END

    UPDATE dbo.aspnet_Membership
    SET    PasswordQuestion = @NewPasswordQuestion, PasswordAnswer = @NewPasswordAnswer
    WHERE  UserId=@UserId
    RETURN(0)
END
```

Procedure: aspnet_Membership_CreateUser

Description

Adds a new membership user to the membership database. Records the user in the aspnet_Users and aspnet_Membership tables and, if necessary, adds a new application to the aspnet_Applications table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@Password	nvarchar	Input
@PasswordSalt	nvarchar	Input
@Email	nvarchar	Input
@PasswordQuestion	nvarchar	Input
@PasswordAnswer	nvarchar	Input
@IsApproved	bit	Input
@CurrentTimeUtc	datetime	Input
@CreateDate	datetime	Input
@UniqueEmail	int	Input
@PasswordFormat	int	Input
@UserId	uniqueidentifier	Input/Output

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_CreateUser
    @ApplicationName                          nvarchar(256),
    @UserName                                 nvarchar(256),
    @Password                                 nvarchar(128),
    @PasswordSalt                             nvarchar(128),
    @Email                                    nvarchar(256),
    @PasswordQuestion                         nvarchar(256),
    @PasswordAnswer                           nvarchar(128),
    @IsApproved                               bit,
    @CurrentTimeUtc                           datetime,
    @CreateDate                               datetime = NULL,
    @UniqueEmail                              int      = 0,
    @PasswordFormat                           int      = 0,
    @UserId                                   uniqueidentifier OUTPUT
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL

    DECLARE @NewUserId uniqueidentifier
    SELECT @NewUserId = NULL

    DECLARE @IsLockedOut bit
    SET @IsLockedOut = 0

    DECLARE @LastLockoutDate  datetime
```

Definition

```
    SET @LastLockoutDate = CONVERT( datetime, '17540101', 112 )

    DECLARE @FailedPasswordAttemptCount int
    SET @FailedPasswordAttemptCount = 0

    DECLARE @FailedPasswordAttemptWindowStart  datetime
    SET @FailedPasswordAttemptWindowStart = CONVERT( datetime, '17540101', 112 )

    DECLARE @FailedPasswordAnswerAttemptCount int
    SET @FailedPasswordAnswerAttemptCount = 0

    DECLARE @FailedPasswordAnswerAttemptWindowStart  datetime
    SET @FailedPasswordAnswerAttemptWindowStart = CONVERT( datetime, '17540101', 112 )

    DECLARE @NewUserCreated bit
    DECLARE @ReturnValue   int
    SET @ReturnValue = 0

    DECLARE @ErrorCode     int
    SET @ErrorCode = 0

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
            BEGIN TRANSACTION
            SET @TranStarted = 1
    END
    ELSE
          SET @TranStarted = 0

    EXEC dbo.aspnet_Applications_CreateApplication @ApplicationName, @ApplicationId OUTPUT

    IF( @@ERROR <> 0 )
    BEGIN
       SET @ErrorCode = -1
       GOTO Cleanup
    END

    SET @CreateDate = @CurrentTimeUtc

    SELECT  @NewUserId = UserId FROM dbo.aspnet_Users WHERE LOWER(@UserName) =
LoweredUserName AND @ApplicationId = ApplicationId
    IF ( @NewUserId IS NULL )
    BEGIN
       SET @NewUserId = @UserId
       EXEC @ReturnValue = dbo.aspnet_Users_CreateUser @ApplicationId, @UserName, 0, @CreateDate,
@NewUserId OUTPUT
       SET @NewUserCreated = 1
    END
    ELSE
    BEGIN
       SET @NewUserCreated = 0
       IF( @NewUserId <> @UserId AND @UserId IS NOT NULL )
       BEGIN
          SET @ErrorCode = 6
          GOTO Cleanup
       END
    END
```

Definition

```
IF( @@ERROR <> 0 )
BEGIN
   SET @ErrorCode = -1
   GOTO Cleanup
END

IF( @ReturnValue = -1 )
BEGIN
   SET @ErrorCode = 10
   GOTO Cleanup
END

IF ( EXISTS ( SELECT UserId
          FROM   dbo.aspnet_Membership
          WHERE  @NewUserId = UserId ) )
BEGIN
   SET @ErrorCode = 6
   GOTO Cleanup
END

SET @UserId = @NewUserId

IF (@UniqueEmail = 1)
BEGIN
   IF (EXISTS (SELECT *
           FROM  dbo.aspnet_Membership m WITH ( UPDLOCK, HOLDLOCK )
           WHERE ApplicationId = @ApplicationId AND LoweredEmail = LOWER(@Email)))
   BEGIN
      SET @ErrorCode = 7
      GOTO Cleanup
   END
END

IF (@NewUserCreated = 0)
BEGIN
   UPDATE dbo.aspnet_Users
   SET    LastActivityDate = @CreateDate
   WHERE  @UserId = UserId
   IF( @@ERROR <> 0 )
   BEGIN
      SET @ErrorCode = -1
      GOTO Cleanup
   END
END

INSERT INTO dbo.aspnet_Membership
         ( ApplicationId,
           UserId,
           Password,
           PasswordSalt,
           Email,
           LoweredEmail,
           PasswordQuestion,
           PasswordAnswer,
           PasswordFormat,
           IsApproved,
           IsLockedOut,
           CreateDate,
           LastLoginDate,
```

Definition

```
                LastPasswordChangedDate,
                LastLockoutDate,
                FailedPasswordAttemptCount,
                FailedPasswordAttemptWindowStart,
                FailedPasswordAnswerAttemptCount,
                FailedPasswordAnswerAttemptWindowStart )
        VALUES ( @ApplicationId,
                @UserId,
                @Password,
                @PasswordSalt,
                @Email,
                LOWER(@Email),
                @PasswordQuestion,
                @PasswordAnswer,
                @PasswordFormat,
                @IsApproved,
                @IsLockedOut,
                @CreateDate,
                @CreateDate,
                @CreateDate,
                @LastLockoutDate,
                @FailedPasswordAttemptCount,
                @FailedPasswordAttemptWindowStart,
                @FailedPasswordAnswerAttemptCount,
                @FailedPasswordAnswerAttemptWindowStart )

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    IF( @TranStarted = 1 )
    BEGIN
            SET @TranStarted = 0
            COMMIT TRANSACTION
    END

    RETURN 0

Cleanup:

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
            ROLLBACK TRANSACTION
    END

    RETURN @ErrorCode

END
```

Procedure: aspnet_Membership_FindUsersByEmail

Description

Retrieves records from aspnet_Membership table with email addresses matching the specified pattern and with the specified application ID.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@EmailToMatch	nvarchar	Input
@PageIndex	int	Input
@PageSize	int	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_FindUsersByEmail
    @ApplicationName       nvarchar(256),
    @EmailToMatch          nvarchar(256),
    @PageIndex             int,
    @PageSize              int
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM dbo.aspnet_Applications WHERE
LOWER(@ApplicationName) = LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN 0

    -- Set the page bounds
    DECLARE @PageLowerBound int
    DECLARE @PageUpperBound int
    DECLARE @TotalRecords   int
    SET @PageLowerBound = @PageSize * @PageIndex
    SET @PageUpperBound = @PageSize - 1 + @PageLowerBound

    -- Create a temp table TO store the select results
    CREATE TABLE #PageIndexForUsers
    (
        IndexId int IDENTITY (0, 1) NOT NULL,
        UserId uniqueidentifier
    )

    -- Insert into our temp table
    IF( @EmailToMatch IS NULL )
        INSERT INTO #PageIndexForUsers (UserId)
            SELECT u.UserId
            FROM   dbo.aspnet_Users u, dbo.aspnet_Membership m
            WHERE  u.ApplicationId = @ApplicationId AND m.UserId = u.UserId AND m.Email IS NULL
            ORDER BY m.LoweredEmail
    ELSE
        INSERT INTO #PageIndexForUsers (UserId)
            SELECT u.UserId
            FROM   dbo.aspnet_Users u, dbo.aspnet_Membership m
            WHERE  u.ApplicationId = @ApplicationId AND m.UserId = u.UserId AND m.LoweredEmail LIKE
```

Definition

```
LOWER(@EmailToMatch)
            ORDER BY m.LoweredEmail

    SELECT  u.UserName, m.Email, m.PasswordQuestion, m.Comment, m.IsApproved,
            m.CreateDate,
            m.LastLoginDate,
            u.LastActivityDate,
            m.LastPasswordChangedDate,
            u.UserId, m.IsLockedOut,
            m.LastLockoutDate
    FROM   dbo.aspnet_Membership m, dbo.aspnet_Users u, #PageIndexForUsers p
    WHERE  u.UserId = p.UserId AND u.UserId = m.UserId AND
           p.IndexId >= @PageLowerBound AND p.IndexId <= @PageUpperBound
    ORDER BY m.LoweredEmail

    SELECT  @TotalRecords = COUNT(*)
    FROM    #PageIndexForUsers
    RETURN @TotalRecords
END
```

Procedure: aspnet_Membership_FindUsersByName

Description

Retrieves records from aspnet_Membership table with user names matching the specified pattern and with the specified application ID.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserNameToMatch	nvarchar	Input
@PageIndex	int	Input
@PageSize	int	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_FindUsersByName
    @ApplicationName       nvarchar(256),
    @UserNameToMatch       nvarchar(256),
    @PageIndex             int,
    @PageSize              int
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM dbo.aspnet_Applications WHERE
LOWER(@ApplicationName) = LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN 0

    -- Set the page bounds
    DECLARE @PageLowerBound int
    DECLARE @PageUpperBound int
    DECLARE @TotalRecords   int
    SET @PageLowerBound = @PageSize * @PageIndex
    SET @PageUpperBound = @PageSize - 1 + @PageLowerBound

    -- Create a temp table TO store the select results
    CREATE TABLE #PageIndexForUsers
    (
        IndexId int IDENTITY (0, 1) NOT NULL,
        UserId uniqueidentifier
    )

    -- Insert into our temp table
    INSERT INTO #PageIndexForUsers (UserId)
        SELECT u.UserId
        FROM   dbo.aspnet_Users u, dbo.aspnet_Membership m
        WHERE  u.ApplicationId = @ApplicationId AND m.UserId = u.UserId AND u.LoweredUserName LIKE
LOWER(@UserNameToMatch)
        ORDER BY u.UserName


    SELECT  u.UserName, m.Email, m.PasswordQuestion, m.Comment, m.IsApproved,
            m.CreateDate,
            m.LastLoginDate,
```

Definition

```
            u.LastActivityDate,
            m.LastPasswordChangedDate,
            u.UserId, m.IsLockedOut,
            m.LastLockoutDate
    FROM   dbo.aspnet_Membership m, dbo.aspnet_Users u, #PageIndexForUsers p
    WHERE  u.UserId = p.UserId AND u.UserId = m.UserId AND
           p.IndexId >= @PageLowerBound AND p.IndexId <= @PageUpperBound
    ORDER BY u.UserName

    SELECT  @TotalRecords = COUNT(*)
    FROM    #PageIndexForUsers
    RETURN @TotalRecords
END
```

Procedure: aspnet_Membership_GetAllUsers

Description

Retrieves all users from the aspnet_Membership table with the specified application ID.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@PageIndex	int	Input
@PageSize	int	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_GetAllUsers
    @ApplicationName       nvarchar(256),
    @PageIndex             int,
    @PageSize              int
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM dbo.aspnet_Applications WHERE
LOWER(@ApplicationName) = LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN 0


    -- Set the page bounds
    DECLARE @PageLowerBound int
    DECLARE @PageUpperBound int
    DECLARE @TotalRecords   int
    SET @PageLowerBound = @PageSize * @PageIndex
    SET @PageUpperBound = @PageSize - 1 + @PageLowerBound

    -- Create a temp table TO store the select results
    CREATE TABLE #PageIndexForUsers
    (
        IndexId int IDENTITY (0, 1) NOT NULL,
        UserId uniqueidentifier
    )

    -- Insert into our temp table
    INSERT INTO #PageIndexForUsers (UserId)
    SELECT u.UserId
    FROM   dbo.aspnet_Membership m, dbo.aspnet_Users u
    WHERE  u.ApplicationId = @ApplicationId AND u.UserId = m.UserId
    ORDER BY u.UserName

    SELECT @TotalRecords = @@ROWCOUNT

    SELECT u.UserName, m.Email, m.PasswordQuestion, m.Comment, m.IsApproved,
            m.CreateDate,
            m.LastLoginDate,
            u.LastActivityDate,
            m.LastPasswordChangedDate,
            u.UserId, m.IsLockedOut,
```

Definition

```
          m.LastLockoutDate
    FROM   dbo.aspnet_Membership m, dbo.aspnet_Users u, #PageIndexForUsers p
    WHERE  u.UserId = p.UserId AND u.UserId = m.UserId AND
          p.IndexId >= @PageLowerBound AND p.IndexId <= @PageUpperBound
    ORDER BY u.UserName
    RETURN @TotalRecords
END
```

Procedure: aspnet_Membership_GetNumberOfUsersOnline

Description

Gets the number of users currently online (those whose last activity dates.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@MinutesSinceLastInActive	int	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_GetNumberOfUsersOnline
    @ApplicationName            nvarchar(256),
    @MinutesSinceLastInActive   int,
    @CurrentTimeUtc             datetime
AS
BEGIN
    DECLARE @DateActive datetime
    SELECT  @DateActive = DATEADD(minute,  -(@MinutesSinceLastInActive), @CurrentTimeUtc)

    DECLARE @NumOnline int
    SELECT  @NumOnline = COUNT(*)
    FROM    dbo.aspnet_Users u(NOLOCK),
            dbo.aspnet_Applications a(NOLOCK),
            dbo.aspnet_Membership m(NOLOCK)
    WHERE   u.ApplicationId = a.ApplicationId                  AND
            LastActivityDate > @DateActive                     AND
            a.LoweredApplicationName = LOWER(@ApplicationName) AND
            u.UserId = m.UserId
    RETURN(@NumOnline)
END
```

Procedure: aspnet_Membership_GetPassword

Description

Gets the specified user's password data from the database. Used for retrieving passwords with a user-supplied password answer.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@MaxInvalidPasswordAttempts	int	Input
@PasswordAttemptWindow	int	Input
@CurrentTimeUtc	datetime	Input
@PasswordAnswer	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_GetPassword
    @ApplicationName                nvarchar(256),
    @UserName                       nvarchar(256),
    @MaxInvalidPasswordAttempts     int,
    @PasswordAttemptWindow          int,
    @CurrentTimeUtc                 datetime,
    @PasswordAnswer                 nvarchar(128) = NULL
AS
BEGIN
    DECLARE @UserId                                 uniqueidentifier
    DECLARE @PasswordFormat                         int
    DECLARE @Password                               nvarchar(128)
    DECLARE @passAns                                nvarchar(128)
    DECLARE @IsLockedOut                            bit
    DECLARE @LastLockoutDate                        datetime
    DECLARE @FailedPasswordAttemptCount             int
    DECLARE @FailedPasswordAttemptWindowStart       datetime
    DECLARE @FailedPasswordAnswerAttemptCount       int
    DECLARE @FailedPasswordAnswerAttemptWindowStart datetime

    DECLARE @ErrorCode     int
    SET @ErrorCode = 0

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
            BEGIN TRANSACTION
            SET @TranStarted = 1
    END
    ELSE
        SET @TranStarted = 0

    SELECT  @UserId = u.UserId,
        @Password = m.Password,
        @passAns = m.PasswordAnswer,
```

Definition

```
            @PasswordFormat = m.PasswordFormat,
            @IsLockedOut = m.IsLockedOut,
            @LastLockoutDate = m.LastLockoutDate,
            @FailedPasswordAttemptCount = m.FailedPasswordAttemptCount,
            @FailedPasswordAttemptWindowStart = m.FailedPasswordAttemptWindowStart,
            @FailedPasswordAnswerAttemptCount = m.FailedPasswordAnswerAttemptCount,
            @FailedPasswordAnswerAttemptWindowStart = m.FailedPasswordAnswerAttemptWindowStart
    FROM    dbo.aspnet_Applications a, dbo.aspnet_Users u, dbo.aspnet_Membership m WITH ( UPDLOCK )
    WHERE   LOWER(@ApplicationName) = a.LoweredApplicationName AND
            u.ApplicationId = a.ApplicationId    AND
            u.UserId = m.UserId AND
            LOWER(@UserName) = u.LoweredUserName

    IF ( @@rowcount = 0 )
    BEGIN
        SET @ErrorCode = 1
        GOTO Cleanup
    END

    IF( @IsLockedOut = 1 )
    BEGIN
        SET @ErrorCode = 99
        GOTO Cleanup
    END

    IF ( NOT( @PasswordAnswer IS NULL ) )
    BEGIN
        IF( ( @passAns IS NULL ) OR ( LOWER( @passAns ) <> LOWER( @PasswordAnswer ) ) )
        BEGIN
            IF( @CurrentTimeUtc > DATEADD( minute, @PasswordAttemptWindow,
@FailedPasswordAnswerAttemptWindowStart ) )
            BEGIN
                SET @FailedPasswordAnswerAttemptWindowStart = @CurrentTimeUtc
                SET @FailedPasswordAnswerAttemptCount = 1
            END
            ELSE
            BEGIN
                SET @FailedPasswordAnswerAttemptCount = @FailedPasswordAnswerAttemptCount + 1
                SET @FailedPasswordAnswerAttemptWindowStart = @CurrentTimeUtc
            END

            BEGIN
                IF( @FailedPasswordAnswerAttemptCount >= @MaxInvalidPasswordAttempts )
                BEGIN
                    SET @IsLockedOut = 1
                    SET @LastLockoutDate = @CurrentTimeUtc
                END
            END

            SET @ErrorCode = 3
        END
        ELSE
        BEGIN
            IF( @FailedPasswordAnswerAttemptCount > 0 )
            BEGIN
                SET @FailedPasswordAnswerAttemptCount = 0
                SET @FailedPasswordAnswerAttemptWindowStart = CONVERT( datetime, '17540101', 112 )
            END
        END
```

Definition

```
        UPDATE dbo.aspnet_Membership
        SET IsLockedOut = @IsLockedOut, LastLockoutDate = @LastLockoutDate,
            FailedPasswordAttemptCount = @FailedPasswordAttemptCount,
            FailedPasswordAttemptWindowStart = @FailedPasswordAttemptWindowStart,
            FailedPasswordAnswerAttemptCount = @FailedPasswordAnswerAttemptCount,
            FailedPasswordAnswerAttemptWindowStart = @FailedPasswordAnswerAttemptWindowStart
        WHERE @UserId = UserId

        IF( @@ERROR <> 0 )
        BEGIN
            SET @ErrorCode = -1
            GOTO Cleanup
        END
    END

    IF( @TranStarted = 1 )
    BEGIN
             SET @TranStarted = 0
             COMMIT TRANSACTION
    END

    IF( @ErrorCode = 0 )
        SELECT @Password, @PasswordFormat

    RETURN @ErrorCode

Cleanup:

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
             ROLLBACK TRANSACTION
    END

    RETURN @ErrorCode

END
```

Procedure: aspnet_Membership_GetPasswordWithFormat

Description

Gets the specified user's password from the database. Used by the provider to retrieve passwords for performing password comparisons (for example, when ValidateUser needs to validate a password).

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@UpdateLastLoginActivityDate	bit	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_GetPasswordWithFormat
    @ApplicationName                nvarchar(256),
    @UserName                       nvarchar(256),
    @UpdateLastLoginActivityDate    bit,
    @CurrentTimeUtc                 datetime
AS
BEGIN
    DECLARE @IsLockedOut                        bit
    DECLARE @UserId                             uniqueidentifier
    DECLARE @Password                           nvarchar(128)
    DECLARE @PasswordSalt                       nvarchar(128)
    DECLARE @PasswordFormat                     int
    DECLARE @FailedPasswordAttemptCount         int
    DECLARE @FailedPasswordAnswerAttemptCount   int
    DECLARE @IsApproved                         bit
    DECLARE @LastActivityDate                   datetime
    DECLARE @LastLoginDate                      datetime

    SELECT  @UserId          = NULL

    SELECT  @UserId = u.UserId, @IsLockedOut = m.IsLockedOut, @Password=Password,
@PasswordFormat=PasswordFormat,
            @PasswordSalt=PasswordSalt, @FailedPasswordAttemptCount=FailedPasswordAttemptCount,
                        @FailedPasswordAnswerAttemptCount=FailedPasswordAnswerAttemptCount,
@IsApproved=IsApproved,
            @LastActivityDate = LastActivityDate, @LastLoginDate = LastLoginDate
    FROM    dbo.aspnet_Applications a, dbo.aspnet_Users u, dbo.aspnet_Membership m
    WHERE   LOWER(@ApplicationName) = a.LoweredApplicationName AND
            u.ApplicationId = a.ApplicationId    AND
            u.UserId = m.UserId AND
            LOWER(@UserName) = u.LoweredUserName

    IF (@UserId IS NULL)
        RETURN 1

    IF (@IsLockedOut = 1)
        RETURN 99

    SELECT   @Password, @PasswordFormat, @PasswordSalt, @FailedPasswordAttemptCount,
             @FailedPasswordAnswerAttemptCount, @IsApproved, @LastLoginDate, @LastActivityDate
```

Definition

```
    IF (@UpdateLastLoginActivityDate = 1 AND @IsApproved = 1)
    BEGIN
        UPDATE  dbo.aspnet_Membership
        SET     LastLoginDate = @CurrentTimeUtc
        WHERE   UserId = @UserId

        UPDATE  dbo.aspnet_Users
        SET     LastActivityDate = @CurrentTimeUtc
        WHERE   @UserId = UserId
    END


    RETURN 0
END
```

Procedure: aspnet_Membership_GetUserByEmail

Description

Given an e-mail address and application ID, retrieves the corresponding record from the aspnet_Membership table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@Email	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_GetUserByEmail
    @ApplicationName  nvarchar(256),
    @Email            nvarchar(256)
AS
BEGIN
    IF( @Email IS NULL )
        SELECT  u.UserName
        FROM    dbo.aspnet_Applications a, dbo.aspnet_Users u, dbo.aspnet_Membership m
        WHERE   LOWER(@ApplicationName) = a.LoweredApplicationName AND
                u.ApplicationId = a.ApplicationId    AND
                u.UserId = m.UserId AND
                m.LoweredEmail IS NULL
    ELSE
        SELECT  u.UserName
        FROM    dbo.aspnet_Applications a, dbo.aspnet_Users u, dbo.aspnet_Membership m
        WHERE   LOWER(@ApplicationName) = a.LoweredApplicationName AND
                u.ApplicationId = a.ApplicationId    AND
                u.UserId = m.UserId AND
                LOWER(@Email) = m.LoweredEmail

    IF (@@rowcount = 0)
        RETURN(1)
    RETURN(0)
END
```

Procedure: aspnet_Membership_GetUserByName

Description

Given a user name and application ID, retrieves the corresponding record from the aspnet_Membership table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@CurrentTimeUtc	datetime	Input
@UpdateLastActivity	bit	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_GetUserByName
    @ApplicationName      nvarchar(256),
    @UserName             nvarchar(256),
    @CurrentTimeUtc       datetime,
    @UpdateLastActivity   bit = 0
AS
BEGIN
    DECLARE @UserId uniqueidentifier

    IF (@UpdateLastActivity = 1)
    BEGIN
        -- select user ID from aspnet_users table
        SELECT TOP 1 @UserId = u.UserId
        FROM    dbo.aspnet_Applications a, dbo.aspnet_Users u, dbo.aspnet_Membership m
        WHERE    LOWER(@ApplicationName) = a.LoweredApplicationName AND
                u.ApplicationId = a.ApplicationId    AND
                LOWER(@UserName) = u.LoweredUserName AND u.UserId = m.UserId

        IF (@@ROWCOUNT = 0) -- Username not found
            RETURN -1

        UPDATE   dbo.aspnet_Users
        SET      LastActivityDate = @CurrentTimeUtc
        WHERE    @UserId = UserId

        SELECT m.Email, m.PasswordQuestion, m.Comment, m.IsApproved,
                m.CreateDate, m.LastLoginDate, u.LastActivityDate, m.LastPasswordChangedDate,
                u.UserId, m.IsLockedOut, m.LastLockoutDate
        FROM    dbo.aspnet_Applications a, dbo.aspnet_Users u, dbo.aspnet_Membership m
        WHERE  @UserId = u.UserId AND u.UserId = m.UserId
    END
    ELSE
    BEGIN
        SELECT TOP 1 m.Email, m.PasswordQuestion, m.Comment, m.IsApproved,
                m.CreateDate, m.LastLoginDate, u.LastActivityDate, m.LastPasswordChangedDate,
                u.UserId, m.IsLockedOut,m.LastLockoutDate
        FROM    dbo.aspnet_Applications a, dbo.aspnet_Users u, dbo.aspnet_Membership m
        WHERE    LOWER(@ApplicationName) = a.LoweredApplicationName AND
                u.ApplicationId = a.ApplicationId    AND
                LOWER(@UserName) = u.LoweredUserName AND u.UserId = m.UserId
```

Definition

```
        IF (@@ROWCOUNT = 0) -- Username not found
            RETURN -1
    END

    RETURN 0
END
```

Procedure: aspnet_Membership_GetUserByUserId

Description

Given a user ID and application ID, retrieves the corresponding record from the aspnet_Membership table.

Parameters

Name	Type	Direction
@UserId	uniqueidentifier	Input
@CurrentTimeUtc	datetime	Input
@UpdateLastActivity	bit	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_GetUserByUserId
    @UserId               uniqueidentifier,
    @CurrentTimeUtc       datetime,
    @UpdateLastActivity   bit = 0
AS
BEGIN
    IF ( @UpdateLastActivity = 1 )
    BEGIN
        UPDATE   dbo.aspnet_Users
        SET      LastActivityDate = @CurrentTimeUtc
        FROM     dbo.aspnet_Users
        WHERE    @UserId = UserId

        IF ( @@ROWCOUNT = 0 ) -- User ID not found
            RETURN -1
    END

    SELECT  m.Email, m.PasswordQuestion, m.Comment, m.IsApproved,
            m.CreateDate, m.LastLoginDate, u.LastActivityDate,
            m.LastPasswordChangedDate, u.UserName, m.IsLockedOut,
            m.LastLockoutDate
    FROM    dbo.aspnet_Users u, dbo.aspnet_Membership m
    WHERE   @UserId = u.UserId AND u.UserId = m.UserId

    IF ( @@ROWCOUNT = 0 ) -- User ID not found
       RETURN -1

    RETURN 0
END
```

Procedure: aspnet_Membership_ResetPassword

Description

Resets the specified user's password based on a password answer.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@NewPassword	nvarchar	Input
@MaxInvalidPasswordAttempts	int	Input
@PasswordAttemptWindow	int	Input
@PasswordSalt	nvarchar	Input
@CurrentTimeUtc	datetime	Input
@PasswordFormat	int	Input
@PasswordAnswer	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_ResetPassword
    @ApplicationName             nvarchar(256),
    @UserName                    nvarchar(256),
    @NewPassword                 nvarchar(128),
    @MaxInvalidPasswordAttempts  int,
    @PasswordAttemptWindow       int,
    @PasswordSalt                nvarchar(128),
    @CurrentTimeUtc              datetime,
    @PasswordFormat              int = 0,
    @PasswordAnswer              nvarchar(128) = NULL
AS
BEGIN
    DECLARE @IsLockedOut                            bit
    DECLARE @LastLockoutDate                        datetime
    DECLARE @FailedPasswordAttemptCount             int
    DECLARE @FailedPasswordAttemptWindowStart       datetime
    DECLARE @FailedPasswordAnswerAttemptCount       int
    DECLARE @FailedPasswordAnswerAttemptWindowStart datetime

    DECLARE @UserId                                 uniqueidentifier
    SET     @UserId = NULL

    DECLARE @ErrorCode     int
    SET @ErrorCode = 0

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
            BEGIN TRANSACTION
            SET @TranStarted = 1
    END
    ELSE
```

Definition

```
            SET @TranStarted = 0

    SELECT  @UserId = u.UserId
    FROM    dbo.aspnet_Users u, dbo.aspnet_Applications a, dbo.aspnet_Membership m
    WHERE   LoweredUserName = LOWER(@UserName) AND
            u.ApplicationId = a.ApplicationId  AND
            LOWER(@ApplicationName) = a.LoweredApplicationName AND
            u.UserId = m.UserId

    IF ( @UserId IS NULL )
    BEGIN
        SET @ErrorCode = 1
        GOTO Cleanup
    END

    SELECT @IsLockedOut = IsLockedOut,
           @LastLockoutDate = LastLockoutDate,
           @FailedPasswordAttemptCount = FailedPasswordAttemptCount,
           @FailedPasswordAttemptWindowStart = FailedPasswordAttemptWindowStart,
           @FailedPasswordAnswerAttemptCount = FailedPasswordAnswerAttemptCount,
           @FailedPasswordAnswerAttemptWindowStart = FailedPasswordAnswerAttemptWindowStart
    FROM dbo.aspnet_Membership WITH ( UPDLOCK )
    WHERE @UserId = UserId

    IF( @IsLockedOut = 1 )
    BEGIN
        SET @ErrorCode = 99
        GOTO Cleanup
    END

    UPDATE dbo.aspnet_Membership
    SET    Password = @NewPassword,
           LastPasswordChangedDate = @CurrentTimeUtc,
           PasswordFormat = @PasswordFormat,
           PasswordSalt = @PasswordSalt
    WHERE  @UserId = UserId AND
           ( ( @PasswordAnswer IS NULL ) OR ( LOWER( PasswordAnswer ) = LOWER( @PasswordAnswer ) ) )

    IF ( @@ROWCOUNT = 0 )
        BEGIN
            IF( @CurrentTimeUtc > DATEADD( minute, @PasswordAttemptWindow,
@FailedPasswordAnswerAttemptWindowStart ) )
            BEGIN
                SET @FailedPasswordAnswerAttemptWindowStart = @CurrentTimeUtc
                SET @FailedPasswordAnswerAttemptCount = 1
            END
            ELSE
            BEGIN
                SET @FailedPasswordAnswerAttemptWindowStart = @CurrentTimeUtc
                SET @FailedPasswordAnswerAttemptCount = @FailedPasswordAnswerAttemptCount + 1
            END

            BEGIN
                IF( @FailedPasswordAnswerAttemptCount >= @MaxInvalidPasswordAttempts )
                BEGIN
                    SET @IsLockedOut = 1
                    SET @LastLockoutDate = @CurrentTimeUtc
                END
            END
```

Definition

```
            SET @ErrorCode = 3
        END
    ELSE
        BEGIN
            IF( @FailedPasswordAnswerAttemptCount > 0 )
            BEGIN
                SET @FailedPasswordAnswerAttemptCount = 0
                SET @FailedPasswordAnswerAttemptWindowStart = CONVERT( datetime, '17540101', 112 )
            END
        END

    IF( NOT ( @PasswordAnswer IS NULL ) )
    BEGIN
        UPDATE dbo.aspnet_Membership
        SET IsLockedOut = @IsLockedOut, LastLockoutDate = @LastLockoutDate,
            FailedPasswordAttemptCount = @FailedPasswordAttemptCount,
            FailedPasswordAttemptWindowStart = @FailedPasswordAttemptWindowStart,
            FailedPasswordAnswerAttemptCount = @FailedPasswordAnswerAttemptCount,
            FailedPasswordAnswerAttemptWindowStart = @FailedPasswordAnswerAttemptWindowStart
        WHERE @UserId = UserId

        IF( @@ERROR <> 0 )
        BEGIN
            SET @ErrorCode = -1
            GOTO Cleanup
        END
    END

    IF( @TranStarted = 1 )
    BEGIN
            SET @TranStarted = 0
            COMMIT TRANSACTION
    END

    RETURN @ErrorCode

Cleanup:

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
            ROLLBACK TRANSACTION
    END

    RETURN @ErrorCode

END
```

Procedure: aspnet_Membership_SetPassword

Description

Sets the specified user's password to the password input to the stored procedure.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@NewPassword	nvarchar	Input
@PasswordSalt	nvarchar	Input
@CurrentTimeUtc	datetime	Input
@PasswordFormat	int	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_SetPassword
    @ApplicationName  nvarchar(256),
    @UserName         nvarchar(256),
    @NewPassword      nvarchar(128),
    @PasswordSalt     nvarchar(128),
    @CurrentTimeUtc   datetime,
    @PasswordFormat   int = 0
AS
BEGIN
    DECLARE @UserId uniqueidentifier
    SELECT  @UserId = NULL
    SELECT  @UserId = u.UserId
    FROM    dbo.aspnet_Users u, dbo.aspnet_Applications a, dbo.aspnet_Membership m
    WHERE   LoweredUserName = LOWER(@UserName) AND
            u.ApplicationId = a.ApplicationId  AND
            LOWER(@ApplicationName) = a.LoweredApplicationName AND
            u.UserId = m.UserId

    IF (@UserId IS NULL)
        RETURN(1)

    UPDATE dbo.aspnet_Membership
    SET Password = @NewPassword, PasswordFormat = @PasswordFormat, PasswordSalt = @PasswordSalt,
        LastPasswordChangedDate = @CurrentTimeUtc
    WHERE @UserId = UserId
    RETURN(0)
END
```

Procedure: aspnet_Membership_UnlockUser

Description

Restores login privileges for the specified user by setting the user's IsLockedOut bit to 0.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_UnlockUser
    @ApplicationName                         nvarchar(256),
    @UserName                                nvarchar(256)
AS
BEGIN
    DECLARE @UserId uniqueidentifier
    SELECT  @UserId = NULL
    SELECT  @UserId = u.UserId
    FROM    dbo.aspnet_Users u, dbo.aspnet_Applications a, dbo.aspnet_Membership m
    WHERE   LoweredUserName = LOWER(@UserName) AND
            u.ApplicationId = a.ApplicationId  AND
            LOWER(@ApplicationName) = a.LoweredApplicationName AND
            u.UserId = m.UserId

    IF ( @UserId IS NULL )
        RETURN 1

    UPDATE dbo.aspnet_Membership
    SET IsLockedOut = 0,
        FailedPasswordAttemptCount = 0,
        FailedPasswordAttemptWindowStart = CONVERT( datetime, '17540101', 112 ),
        FailedPasswordAnswerAttemptCount = 0,
        FailedPasswordAnswerAttemptWindowStart = CONVERT( datetime, '17540101', 112 ),
        LastLockoutDate = CONVERT( datetime, '17540101', 112 )
    WHERE @UserId = UserId

    RETURN 0
END
```

Procedure: aspnet_Membership_UpdateUser

Description

Updates the user's last activity date in the aspnet_Users table and e-mail address, comment, isapproved status, and last login date in the aspnet_Membership table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@Email	nvarchar	Input
@Comment	ntext	Input
@IsApproved	bit	Input
@LastLoginDate	datetime	Input
@LastActivityDate	datetime	Input
@UniqueEmail	int	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_UpdateUser
    @ApplicationName      nvarchar(256),
    @UserName             nvarchar(256),
    @Email                nvarchar(256),
    @Comment              ntext,
    @IsApproved           bit,
    @LastLoginDate        datetime,
    @LastActivityDate     datetime,
    @UniqueEmail          int,
    @CurrentTimeUtc       datetime
AS
BEGIN
    DECLARE @UserId uniqueidentifier
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @UserId = NULL
    SELECT  @UserId = u.UserId, @ApplicationId = a.ApplicationId
    FROM    dbo.aspnet_Users u, dbo.aspnet_Applications a, dbo.aspnet_Membership m
    WHERE   LoweredUserName = LOWER(@UserName) AND
            u.ApplicationId = a.ApplicationId  AND
            LOWER(@ApplicationName) = a.LoweredApplicationName AND
            u.UserId = m.UserId

    IF (@UserId IS NULL)
        RETURN(1)

    IF (@UniqueEmail = 1)
    BEGIN
        IF (EXISTS (SELECT *
                    FROM  dbo.aspnet_Membership WITH (UPDLOCK, HOLDLOCK)
                    WHERE ApplicationId = @ApplicationId  AND @UserId <> UserId AND LoweredEmail =
LOWER(@Email)))
        BEGIN
            RETURN(7)
```

Definition

```
        END
    END

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
            BEGIN TRANSACTION
            SET @TranStarted = 1
    END
    ELSE
          SET @TranStarted = 0

    UPDATE dbo.aspnet_Users WITH (ROWLOCK)
    SET
         LastActivityDate = @LastActivityDate
    WHERE
       @UserId = UserId

    IF( @@ERROR <> 0 )
        GOTO Cleanup

    UPDATE dbo.aspnet_Membership WITH (ROWLOCK)
    SET
         Email            = @Email,
         LoweredEmail     = LOWER(@Email),
         Comment          = @Comment,
         IsApproved       = @IsApproved,
         LastLoginDate    = @LastLoginDate
    WHERE
       @UserId = UserId

    IF( @@ERROR <> 0 )
        GOTO Cleanup

    IF( @TranStarted = 1 )
    BEGIN
          SET @TranStarted = 0
          COMMIT TRANSACTION
    END

    RETURN 0

Cleanup:

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
          ROLLBACK TRANSACTION
    END

    RETURN -1
END
```

Procedure: aspnet_Membership_UpdateUserInfo

Description

Updates account locking data for the specified user in the aspnet_Users and aspnet_Membership tables. Used in conjunction with provider methods that track bad password and bad password-answer attempts.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@IsPasswordCorrect	bit	Input
@UpdateLastLoginActivityDate	bit	Input
@MaxInvalidPasswordAttempts	int	Input
@PasswordAttemptWindow	int	Input
@CurrentTimeUtc	datetime	Input
@LastLoginDate	datetime	Input
@LastActivityDate	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Membership_UpdateUserInfo
    @ApplicationName                nvarchar(256),
    @UserName                       nvarchar(256),
    @IsPasswordCorrect              bit,
    @UpdateLastLoginActivityDate    bit,
    @MaxInvalidPasswordAttempts     int,
    @PasswordAttemptWindow          int,
    @CurrentTimeUtc                 datetime,
    @LastLoginDate                  datetime,
    @LastActivityDate               datetime
AS
BEGIN
    DECLARE @UserId                                 uniqueidentifier
    DECLARE @IsApproved                             bit
    DECLARE @IsLockedOut                            bit
    DECLARE @LastLockoutDate                        datetime
    DECLARE @FailedPasswordAttemptCount             int
    DECLARE @FailedPasswordAttemptWindowStart       datetime
    DECLARE @FailedPasswordAnswerAttemptCount       int
    DECLARE @FailedPasswordAnswerAttemptWindowStart datetime

    DECLARE @ErrorCode     int
    SET @ErrorCode = 0

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
            BEGIN TRANSACTION
            SET @TranStarted = 1
    END
    ELSE
```

Definition

```
          SET @TranStarted = 0

    SELECT  @UserId = u.UserId,
            @IsApproved = m.IsApproved,
            @IsLockedOut = m.IsLockedOut,
            @LastLockoutDate = m.LastLockoutDate,
            @FailedPasswordAttemptCount = m.FailedPasswordAttemptCount,
            @FailedPasswordAttemptWindowStart = m.FailedPasswordAttemptWindowStart,
            @FailedPasswordAnswerAttemptCount = m.FailedPasswordAnswerAttemptCount,
            @FailedPasswordAnswerAttemptWindowStart = m.FailedPasswordAnswerAttemptWindowStart
    FROM    dbo.aspnet_Applications a, dbo.aspnet_Users u, dbo.aspnet_Membership m WITH ( UPDLOCK )
    WHERE   LOWER(@ApplicationName) = a.LoweredApplicationName AND
            u.ApplicationId = a.ApplicationId    AND
            u.UserId = m.UserId AND
            LOWER(@UserName) = u.LoweredUserName

    IF ( @@rowcount = 0 )
    BEGIN
        SET @ErrorCode = 1
        GOTO Cleanup
    END

    IF( @IsLockedOut = 1 )
    BEGIN
        GOTO Cleanup
    END

    IF( @IsPasswordCorrect = 0 )
    BEGIN
        IF( @CurrentTimeUtc > DATEADD( minute, @PasswordAttemptWindow,
@FailedPasswordAttemptWindowStart ) )
        BEGIN
            SET @FailedPasswordAttemptWindowStart = @CurrentTimeUtc
            SET @FailedPasswordAttemptCount = 1
        END
        ELSE
        BEGIN
            SET @FailedPasswordAttemptWindowStart = @CurrentTimeUtc
            SET @FailedPasswordAttemptCount = @FailedPasswordAttemptCount + 1
        END

        BEGIN
            IF( @FailedPasswordAttemptCount >= @MaxInvalidPasswordAttempts )
            BEGIN
                SET @IsLockedOut = 1
                SET @LastLockoutDate = @CurrentTimeUtc
            END
        END
    END
    ELSE
    BEGIN
        IF( @FailedPasswordAttemptCount > 0 OR @FailedPasswordAnswerAttemptCount > 0 )
        BEGIN
            SET @FailedPasswordAttemptCount = 0
            SET @FailedPasswordAttemptWindowStart = CONVERT( datetime, '17540101', 112 )
            SET @FailedPasswordAnswerAttemptCount = 0
            SET @FailedPasswordAnswerAttemptWindowStart = CONVERT( datetime, '17540101', 112 )
            SET @LastLockoutDate = CONVERT( datetime, '17540101', 112 )
        END
    END
```

Definition

```
    IF( @UpdateLastLoginActivityDate = 1 )
    BEGIN
        UPDATE  dbo.aspnet_Users
        SET     LastActivityDate = @LastActivityDate
        WHERE   @UserId = UserId

        IF( @@ERROR <> 0 )
        BEGIN
            SET @ErrorCode = -1
            GOTO Cleanup
        END

        UPDATE  dbo.aspnet_Membership
        SET     LastLoginDate = @LastLoginDate
        WHERE   UserId = @UserId

        IF( @@ERROR <> 0 )
        BEGIN
            SET @ErrorCode = -1
            GOTO Cleanup
        END
    END


    UPDATE dbo.aspnet_Membership
    SET IsLockedOut = @IsLockedOut, LastLockoutDate = @LastLockoutDate,
  FailedPasswordAttemptCount = @FailedPasswordAttemptCount,
        FailedPasswordAttemptWindowStart = @FailedPasswordAttemptWindowStart,
        FailedPasswordAnswerAttemptCount = @FailedPasswordAnswerAttemptCount,
        FailedPasswordAnswerAttemptWindowStart = @FailedPasswordAnswerAttemptWindowStart
    WHERE @UserId = UserId

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    IF( @TranStarted = 1 )
    BEGIN
         SET @TranStarted = 0
         COMMIT TRANSACTION
    END

    RETURN @ErrorCode

Cleanup:

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
         ROLLBACK TRANSACTION
    END

    RETURN @ErrorCode

END
```

Procedure: aspnet_Paths_CreatePath

Description

Retrieves a path ID from the aspnet_Paths table, or creates a new one if the specified path doesn't exist.

Parameters

Name	Type	Direction
@ApplicationId	uniqueidentifier	Input
@Path	nvarchar	Input
@PathId	uniqueidentifier	Input/Output

Definition

```
CREATE PROCEDURE dbo.aspnet_Paths_CreatePath
    @ApplicationId UNIQUEIDENTIFIER,
    @Path           NVARCHAR(256),
    @PathId         UNIQUEIDENTIFIER OUTPUT
AS
BEGIN
    BEGIN TRANSACTION
    IF (NOT EXISTS(SELECT * FROM dbo.aspnet_Paths WHERE LoweredPath = LOWER(@Path) AND
ApplicationId = @ApplicationId))
    BEGIN
        INSERT dbo.aspnet_Paths (ApplicationId, Path, LoweredPath) VALUES (@ApplicationId, @Path,
LOWER(@Path))
    END
    COMMIT TRANSACTION
    SELECT @PathId = PathId FROM dbo.aspnet_Paths WHERE LOWER(@Path) = LoweredPath AND
ApplicationId = @ApplicationId
END
```

Procedure: aspnet_Personalization_GetApplicationId

Description

Converts the application name input to it into an application ID.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@ApplicationId	uniqueidentifier	Input/Output

Definition

```
CREATE PROCEDURE dbo.aspnet_Personalization_GetApplicationId (
   @ApplicationName NVARCHAR(256),
   @ApplicationId UNIQUEIDENTIFIER OUT)
AS
BEGIN
   SELECT @ApplicationId = ApplicationId FROM dbo.aspnet_Applications WHERE
LOWER(@ApplicationName) = LoweredApplicationName
END
```

Procedure: aspnet_PersonalizationAdministration_DeleteAllState

Description

Deletes all records from aspnet_PersonalizationAllUsers or aspnet_PersonalizationPerUser corresponding to the specified application ID.

Parameters

Name	Type	Direction
@AllUsersScope	bit	Input
@ApplicationName	nvarchar	Input
@Count	int	Input/Output

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationAdministration_DeleteAllState (
    @AllUsersScope bit,
    @ApplicationName NVARCHAR(256),
    @Count int OUT)
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
    IF (@ApplicationId IS NULL)
        SELECT @Count = 0
    ELSE
    BEGIN
        IF (@AllUsersScope = 1)
            DELETE FROM aspnet_PersonalizationAllUsers
            WHERE PathId IN
               (SELECT Paths.PathId
                FROM dbo.aspnet_Paths Paths
                WHERE Paths.ApplicationId = @ApplicationId)
        ELSE
            DELETE FROM aspnet_PersonalizationPerUser
            WHERE PathId IN
               (SELECT Paths.PathId
                FROM dbo.aspnet_Paths Paths
                WHERE Paths.ApplicationId = @ApplicationId)

        SELECT @Count = @@ROWCOUNT
    END
END
```

Procedure: aspnet_PersonalizationAdministration_FindState

Description

Retrieves profile data from aspnet_PersonalizationAllUsers or aspnet_PersonalizationPerUser meeting several input criteria.

Parameters

Name	Type	Direction
@AllUsersScope	bit	Input
@ApplicationName	nvarchar	Input
@PageIndex	int	Input
@PageSize	int	Input
@Path	nvarchar	Input
@UserName	nvarchar	Input
@InactiveSinceDate	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationAdministration_FindState (
    @AllUsersScope bit,
    @ApplicationName NVARCHAR(256),
    @PageIndex              INT,
    @PageSize               INT,
    @Path NVARCHAR(256) = NULL,
    @UserName NVARCHAR(256) = NULL,
    @InactiveSinceDate DATETIME = NULL)
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
    IF (@ApplicationId IS NULL)
        RETURN

    -- Set the page bounds
    DECLARE @PageLowerBound INT
    DECLARE @PageUpperBound INT
    DECLARE @TotalRecords   INT
    SET @PageLowerBound = @PageSize * @PageIndex
    SET @PageUpperBound = @PageSize - 1 + @PageLowerBound

    -- Create a temp table to store the selected results
    CREATE TABLE #PageIndex (
        IndexId int IDENTITY (0, 1) NOT NULL,
        ItemId UNIQUEIDENTIFIER
    )

    IF (@AllUsersScope = 1)
    BEGIN
        -- Insert into our temp table
        INSERT INTO #PageIndex (ItemId)
        SELECT Paths.PathId
        FROM dbo.aspnet_Paths Paths,
             ((SELECT Paths.PathId
               FROM dbo.aspnet_PersonalizationAllUsers AllUsers, dbo.aspnet_Paths Paths
```

Definition

```
               WHERE Paths.ApplicationId = @ApplicationId
                      AND AllUsers.PathId = Paths.PathId
                      AND (@Path IS NULL OR Paths.LoweredPath LIKE LOWER(@Path))
              ) AS SharedDataPerPath
              FULL OUTER JOIN
              (SELECT DISTINCT Paths.PathId
               FROM dbo.aspnet_PersonalizationPerUser PerUser, dbo.aspnet_Paths Paths
               WHERE Paths.ApplicationId = @ApplicationId
                      AND PerUser.PathId = Paths.PathId
                      AND (@Path IS NULL OR Paths.LoweredPath LIKE LOWER(@Path))
              ) AS UserDataPerPath
              ON SharedDataPerPath.PathId = UserDataPerPath.PathId
             )
        WHERE Paths.PathId = SharedDataPerPath.PathId OR Paths.PathId = UserDataPerPath.PathId
        ORDER BY Paths.Path ASC

        SELECT @TotalRecords = @@ROWCOUNT

        SELECT Paths.Path,
               SharedDataPerPath.LastUpdatedDate,
               SharedDataPerPath.SharedDataLength,
               UserDataPerPath.UserDataLength,
               UserDataPerPath.UserCount
        FROM dbo.aspnet_Paths Paths,
             ((SELECT PageIndex.ItemId AS PathId,
                      AllUsers.LastUpdatedDate AS LastUpdatedDate,
                      DATALENGTH(AllUsers.PageSettings) AS SharedDataLength
               FROM dbo.aspnet_PersonalizationAllUsers AllUsers, #PageIndex PageIndex
               WHERE AllUsers.PathId = PageIndex.ItemId
                     AND PageIndex.IndexId >= @PageLowerBound AND PageIndex.IndexId <= @PageUpperBound
              ) AS SharedDataPerPath
              FULL OUTER JOIN
              (SELECT PageIndex.ItemId AS PathId,
                      SUM(DATALENGTH(PerUser.PageSettings)) AS UserDataLength,
                      COUNT(*) AS UserCount
               FROM aspnet_PersonalizationPerUser PerUser, #PageIndex PageIndex
               WHERE PerUser.PathId = PageIndex.ItemId
                     AND PageIndex.IndexId >= @PageLowerBound AND PageIndex.IndexId <= @PageUpperBound
               GROUP BY PageIndex.ItemId
              ) AS UserDataPerPath
              ON SharedDataPerPath.PathId = UserDataPerPath.PathId
             )
        WHERE Paths.PathId = SharedDataPerPath.PathId OR Paths.PathId = UserDataPerPath.PathId
        ORDER BY Paths.Path ASC
    END
    ELSE
    BEGIN
        -- Insert into our temp table
        INSERT INTO #PageIndex (ItemId)
        SELECT PerUser.Id
        FROM dbo.aspnet_PersonalizationPerUser PerUser, dbo.aspnet_Users Users, dbo.aspnet_Paths Paths
        WHERE Paths.ApplicationId = @ApplicationId
              AND PerUser.UserId = Users.UserId
              AND PerUser.PathId = Paths.PathId
              AND (@Path IS NULL OR Paths.LoweredPath LIKE LOWER(@Path))
    AND (@UserName IS NULL OR Users.LoweredUserName LIKE LOWER(@UserName))
              AND (@InactiveSinceDate IS NULL OR Users.LastActivityDate <= @InactiveSinceDate)
        ORDER BY Paths.Path ASC, Users.UserName ASC

        SELECT @TotalRecords = @@ROWCOUNT
```

Definition

```
        SELECT Paths.Path, PerUser.LastUpdatedDate, DATALENGTH(PerUser.PageSettings),
Users.UserName, Users.LastActivityDate
        FROM dbo.aspnet_PersonalizationPerUser PerUser, dbo.aspnet_Users Users, dbo.aspnet_Paths Paths,
#PageIndex PageIndex
        WHERE PerUser.Id = PageIndex.ItemId
            AND PerUser.UserId = Users.UserId
            AND PerUser.PathId = Paths.PathId
            AND PageIndex.IndexId >= @PageLowerBound AND PageIndex.IndexId <= @PageUpperBound
        ORDER BY Paths.Path ASC, Users.UserName ASC
    END

    RETURN @TotalRecords
END
```

Procedure: aspnet_PersonalizationAdministration_GetCountOfState

Description

Returns a count of records in the aspnet_PersonalizationAllUsers table with path names matching the specified pattern, or a count of records in the aspnet_PersonalizationPerUser table meeting several input criteria.

Parameters

Name	Type	Direction
@Count	int	Input/Output
@AllUsersScope	bit	Input
@ApplicationName	nvarchar	Input
@Path	nvarchar	Input
@UserName	nvarchar	Input
@InactiveSinceDate	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationAdministration_GetCountOfState (
   @Count int OUT,
   @AllUsersScope bit,
   @ApplicationName NVARCHAR(256),
   @Path NVARCHAR(256) = NULL,
   @UserName NVARCHAR(256) = NULL,
   @InactiveSinceDate DATETIME = NULL)
AS
BEGIN

   DECLARE @ApplicationId UNIQUEIDENTIFIER
   EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
   IF (@ApplicationId IS NULL)
      SELECT @Count = 0
   ELSE
      IF (@AllUsersScope = 1)
         SELECT @Count = COUNT(*)
         FROM dbo.aspnet_PersonalizationAllUsers AllUsers, dbo.aspnet_Paths Paths
         WHERE Paths.ApplicationId = @ApplicationId
               AND AllUsers.PathId = Paths.PathId
               AND (@Path IS NULL OR Paths.LoweredPath LIKE LOWER(@Path))
      ELSE
         SELECT @Count = COUNT(*)
         FROM dbo.aspnet_PersonalizationPerUser PerUser, dbo.aspnet_Users Users, dbo.aspnet_Paths
Paths
         WHERE Paths.ApplicationId = @ApplicationId
               AND PerUser.UserId = Users.UserId
               AND PerUser.PathId = Paths.PathId
               AND (@Path IS NULL OR Paths.LoweredPath LIKE LOWER(@Path))
               AND (@UserName IS NULL OR Users.LoweredUserName LIKE LOWER(@UserName))
               AND (@InactiveSinceDate IS NULL OR Users.LastActivityDate <= @InactiveSinceDate)
END
```

Procedure: aspnet_PersonalizationAdministration_ResetSharedState

Description

Resets shared state for the specified page, by deleting the corresponding record from the aspnet_PersonalizationAllUsers table.

Parameters

Name	Type	Direction
@Count	int	Input/Output
@ApplicationName	nvarchar	Input
@Path	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationAdministration_ResetSharedState (
    @Count int OUT,
    @ApplicationName NVARCHAR(256),
    @Path NVARCHAR(256))
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
    IF (@ApplicationId IS NULL)
        SELECT @Count = 0
    ELSE
    BEGIN
        DELETE FROM dbo.aspnet_PersonalizationAllUsers
        WHERE PathId IN
            (SELECT AllUsers.PathId
             FROM dbo.aspnet_PersonalizationAllUsers AllUsers, dbo.aspnet_Paths Paths
             WHERE Paths.ApplicationId = @ApplicationId
                   AND AllUsers.PathId = Paths.PathId
                   AND Paths.LoweredPath = LOWER(@Path))

        SELECT @Count = @@ROWCOUNT
    END
END
```

Procedure: aspnet_PersonalizationAdministration_ResetUserState

Description

Resets per-user state for the specified user and the specified page, by deleting the corresponding record from the aspnet_PersonalizationPerUser table. Can also delete records, based on the user's last activity date if it falls on or before the specified date.

Parameters

Name	Type	Direction
@Count	int	Input/Output
@ApplicationName	nvarchar	Input
@InactiveSinceDate	datetime	Input
@UserName	nvarchar	Input
@Path	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationAdministration_ResetUserState (
    @Count              int             OUT,
    @ApplicationName        NVARCHAR(256),
    @InactiveSinceDate      DATETIME          = NULL,
    @UserName               NVARCHAR(256)      = NULL,
    @Path               NVARCHAR(256)      = NULL)
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
    IF (@ApplicationId IS NULL)
        SELECT @Count = 0
    ELSE
    BEGIN
        DELETE FROM dbo.aspnet_PersonalizationPerUser
        WHERE Id IN (SELECT PerUser.Id
                     FROM dbo.aspnet_PersonalizationPerUser PerUser, dbo.aspnet_Users Users, dbo.aspnet_Paths
Paths
                     WHERE Paths.ApplicationId = @ApplicationId
                           AND PerUser.UserId = Users.UserId
                           AND PerUser.PathId = Paths.PathId
                           AND (@InactiveSinceDate IS NULL OR Users.LastActivityDate <= @InactiveSinceDate)
                           AND (@UserName IS NULL OR Users.LoweredUserName = LOWER(@UserName))
                           AND (@Path IS NULL OR Paths.LoweredPath = LOWER(@Path)))

        SELECT @Count = @@ROWCOUNT
    END
END
```

Procedure: aspnet_PersonalizationAllUsers_GetPageSettings

Description

Retrieves shared state for the specified page from the aspnet_PersonalizationAllUsers table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@Path	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationAllUsers_GetPageSettings (
    @ApplicationName  NVARCHAR(256),
    @Path              NVARCHAR(256))
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    DECLARE @PathId UNIQUEIDENTIFIER

    SELECT @ApplicationId = NULL
    SELECT @PathId = NULL

    EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
    IF (@ApplicationId IS NULL)
    BEGIN
        RETURN
    END

    SELECT @PathId = u.PathId FROM dbo.aspnet_Paths u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredPath = LOWER(@Path)
    IF (@PathId IS NULL)
    BEGIN
        RETURN
    END

    SELECT p.PageSettings FROM dbo.aspnet_PersonalizationAllUsers p WHERE p.PathId = @PathId
END
```

Procedure: aspnet_PersonalizationAllUsers_ResetPageSettings

Description

Resets shared state for the specified page, by deleting the corresponding record from the aspnet_PersonalizationAllUsers table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@Path	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationAllUsers_ResetPageSettings (
    @ApplicationName  NVARCHAR(256),
    @Path             NVARCHAR(256))
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    DECLARE @PathId UNIQUEIDENTIFIER

    SELECT @ApplicationId = NULL
    SELECT @PathId = NULL

    EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
    IF (@ApplicationId IS NULL)
    BEGIN
        RETURN
    END

    SELECT @PathId = u.PathId FROM dbo.aspnet_Paths u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredPath = LOWER(@Path)
    IF (@PathId IS NULL)
    BEGIN
        RETURN
    END

    DELETE FROM dbo.aspnet_PersonalizationAllUsers WHERE PathId = @PathId
    RETURN 0
END
```

Procedure: aspnet_PersonalizationAllUsers_SetPageSettings

Description

Saves shared state for the specified page in the aspnet_PersonalizationAllUsers table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@Path	nvarchar	Input
@PageSettings	image	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationAllUsers_SetPageSettings (
    @ApplicationName  NVARCHAR(256),
    @Path             NVARCHAR(256),
    @PageSettings     IMAGE,
    @CurrentTimeUtc   DATETIME)
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    DECLARE @PathId UNIQUEIDENTIFIER

    SELECT @ApplicationId = NULL
    SELECT @PathId = NULL

    EXEC dbo.aspnet_Applications_CreateApplication @ApplicationName, @ApplicationId OUTPUT

    SELECT @PathId = u.PathId FROM dbo.aspnet_Paths u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredPath = LOWER(@Path)
    IF (@PathId IS NULL)
    BEGIN
        EXEC dbo.aspnet_Paths_CreatePath @ApplicationId, @Path, @PathId OUTPUT
    END

    IF (EXISTS(SELECT PathId FROM dbo.aspnet_PersonalizationAllUsers WHERE PathId = @PathId))
        UPDATE dbo.aspnet_PersonalizationAllUsers SET PageSettings = @PageSettings, LastUpdatedDate =
@CurrentTimeUtc WHERE PathId = @PathId
    ELSE
        INSERT INTO dbo.aspnet_PersonalizationAllUsers(PathId, PageSettings, LastUpdatedDate) VALUES
(@PathId, @PageSettings, @CurrentTimeUtc)
    RETURN 0
END
```

Procedure: aspnet_PersonalizationPerUser_GetPageSettings

Description

Retrieves per-user state for the specified page and the specified user from the aspnet_PersonalizationPerUser table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@Path	nvarchar	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationPerUser_GetPageSettings (
    @ApplicationName  NVARCHAR(256),
    @UserName         NVARCHAR(256),
    @Path             NVARCHAR(256),
    @CurrentTimeUtc   DATETIME)
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    DECLARE @PathId UNIQUEIDENTIFIER
    DECLARE @UserId UNIQUEIDENTIFIER

    SELECT @ApplicationId = NULL
    SELECT @PathId = NULL
    SELECT @UserId = NULL

    EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
    IF (@ApplicationId IS NULL)
    BEGIN
        RETURN
    END

    SELECT @PathId = u.PathId FROM dbo.aspnet_Paths u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredPath = LOWER(@Path)
    IF (@PathId IS NULL)
    BEGIN
        RETURN
    END

    SELECT @UserId = u.UserId FROM dbo.aspnet_Users u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredUserName = LOWER(@UserName)
    IF (@UserId IS NULL)
    BEGIN
        RETURN
    END

    UPDATE   dbo.aspnet_Users WITH (ROWLOCK)
    SET      LastActivityDate = @CurrentTimeUtc
    WHERE    UserId = @UserId
    IF (@@ROWCOUNT = 0) -- Username not found
        RETURN
```

Definition

```
    SELECT p.PageSettings FROM dbo.aspnet_PersonalizationPerUser p WHERE p.PathId = @PathId AND
p.UserId = @UserId
END
```

Procedure: aspnet_PersonalizationPerUser_ResetPageSettings

Description

Resets per-user state for the specified page and the specified user, by deleting the corresponding record from the aspnet_PersonalizationPerUser table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@Path	nvarchar	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationPerUser_ResetPageSettings (
   @ApplicationName  NVARCHAR(256),
   @UserName         NVARCHAR(256),
   @Path             NVARCHAR(256),
   @CurrentTimeUtc   DATETIME)
AS
BEGIN
   DECLARE @ApplicationId UNIQUEIDENTIFIER
   DECLARE @PathId UNIQUEIDENTIFIER
   DECLARE @UserId UNIQUEIDENTIFIER

   SELECT @ApplicationId = NULL
   SELECT @PathId = NULL
   SELECT @UserId = NULL

   EXEC dbo.aspnet_Personalization_GetApplicationId @ApplicationName, @ApplicationId OUTPUT
   IF (@ApplicationId IS NULL)
   BEGIN
      RETURN
   END

   SELECT @PathId = u.PathId FROM dbo.aspnet_Paths u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredPath = LOWER(@Path)
   IF (@PathId IS NULL)
   BEGIN
      RETURN
   END

   SELECT @UserId = u.UserId FROM dbo.aspnet_Users u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredUserName = LOWER(@UserName)
   IF (@UserId IS NULL)
   BEGIN
      RETURN
   END

   UPDATE   dbo.aspnet_Users WITH (ROWLOCK)
   SET      LastActivityDate = @CurrentTimeUtc
   WHERE    UserId = @UserId
   IF (@@ROWCOUNT = 0) -- Username not found
      RETURN
```

Definition

```
    DELETE FROM dbo.aspnet_PersonalizationPerUser WHERE PathId = @PathId AND UserId = @UserId
    RETURN 0
END
```

Procedure: aspnet_PersonalizationPerUser_SetPageSettings

Description

Saves per-user state for the specified page and the specified user in the aspnet_PersonalizationPerUser table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@Path	nvarchar	Input
@PageSettings	image	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_PersonalizationPerUser_SetPageSettings (
    @ApplicationName  NVARCHAR(256),
    @UserName         NVARCHAR(256),
    @Path             NVARCHAR(256),
    @PageSettings     IMAGE,
    @CurrentTimeUtc   DATETIME)
AS
BEGIN
    DECLARE @ApplicationId UNIQUEIDENTIFIER
    DECLARE @PathId UNIQUEIDENTIFIER
    DECLARE @UserId UNIQUEIDENTIFIER

    SELECT @ApplicationId = NULL
    SELECT @PathId = NULL
    SELECT @UserId = NULL

    EXEC dbo.aspnet_Applications_CreateApplication @ApplicationName, @ApplicationId OUTPUT

    SELECT @PathId = u.PathId FROM dbo.aspnet_Paths u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredPath = LOWER(@Path)
    IF (@PathId IS NULL)
    BEGIN
        EXEC dbo.aspnet_Paths_CreatePath @ApplicationId, @Path, @PathId OUTPUT
    END

    SELECT @UserId = u.UserId FROM dbo.aspnet_Users u WHERE u.ApplicationId = @ApplicationId AND
u.LoweredUserName = LOWER(@UserName)
    IF (@UserId IS NULL)
    BEGIN
        EXEC dbo.aspnet_Users_CreateUser @ApplicationId, @UserName, 0, @CurrentTimeUtc, @UserId
OUTPUT
    END

    UPDATE   dbo.aspnet_Users WITH (ROWLOCK)
    SET      LastActivityDate = @CurrentTimeUtc
    WHERE    UserId = @UserId
    IF (@@ROWCOUNT = 0) -- Username not found
        RETURN

    IF (EXISTS(SELECT PathId FROM dbo.aspnet_PersonalizationPerUser WHERE UserId = @UserId AND
```

Definition

```
PathId = @PathId))
        UPDATE dbo.aspnet_PersonalizationPerUser SET PageSettings = @PageSettings, LastUpdatedDate =
@CurrentTimeUtc WHERE UserId = @UserId AND PathId = @PathId
    ELSE
        INSERT INTO dbo.aspnet_PersonalizationPerUser(UserId, PathId, PageSettings, LastUpdatedDate)
VALUES (@UserId, @PathId, @PageSettings, @CurrentTimeUtc)
    RETURN 0
END
```

Procedure: aspnet_Profile_DeleteInactiveProfiles

Description

Deletes profile data from the aspnet_Profile table for users whose last activity dates in the aspnet_Users table fall on or before the specified date.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@ProfileAuthOptions	int	Input
@InactiveSinceDate	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Profile_DeleteInactiveProfiles
   @ApplicationName        nvarchar(256),
   @ProfileAuthOptions     int,
   @InactiveSinceDate      datetime
AS
BEGIN
   DECLARE @ApplicationId uniqueidentifier
   SELECT  @ApplicationId = NULL
   SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
   IF (@ApplicationId IS NULL)
   BEGIN
      SELECT  0
      RETURN
   END

   DELETE
   FROM    dbo.aspnet_Profile
   WHERE   UserId IN
         (   SELECT  UserId
             FROM    dbo.aspnet_Users u
             WHERE   ApplicationId = @ApplicationId
                     AND (LastActivityDate <= @InactiveSinceDate)
                     AND (
                            (@ProfileAuthOptions = 2)
                         OR (@ProfileAuthOptions = 0 AND IsAnonymous = 1)
                         OR (@ProfileAuthOptions = 1 AND IsAnonymous = 0)
                        )
         )

   SELECT  @@ROWCOUNT
END
```

Procedure: aspnet_Profile_DeleteProfiles

Description

Deletes profile data from the aspnet_Profile table for the specified users.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserNames	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Profile_DeleteProfiles
    @ApplicationName        nvarchar(256),
    @UserNames              nvarchar(4000)
AS
BEGIN
    DECLARE @UserName     nvarchar(256)
    DECLARE @CurrentPos   int
    DECLARE @NextPos      int
    DECLARE @NumDeleted   int
    DECLARE @DeletedUser  int
    DECLARE @TranStarted  bit
    DECLARE @ErrorCode    int

    SET @ErrorCode = 0
    SET @CurrentPos = 1
    SET @NumDeleted = 0
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
        BEGIN TRANSACTION
        SET @TranStarted = 1
    END
    ELSE
          SET @TranStarted = 0

    WHILE (@CurrentPos <= LEN(@UserNames))
    BEGIN
        SELECT @NextPos = CHARINDEX(N',', @UserNames,  @CurrentPos)
        IF (@NextPos = 0 OR @NextPos IS NULL)
            SELECT @NextPos = LEN(@UserNames) + 1

        SELECT @UserName = SUBSTRING(@UserNames, @CurrentPos, @NextPos - @CurrentPos)
        SELECT @CurrentPos = @NextPos+1

        IF (LEN(@UserName) > 0)
        BEGIN
            SELECT @DeletedUser = 0
            EXEC dbo.aspnet_Users_DeleteUser @ApplicationName, @UserName, 4, @DeletedUser OUTPUT
            IF( @@ERROR <> 0 )
            BEGIN
                SET @ErrorCode = -1
                GOTO Cleanup
            END
            IF (@DeletedUser <> 0)
```

Definition

```
                SELECT @NumDeleted = @NumDeleted + 1
        END
    END
    SELECT @NumDeleted
    IF (@TranStarted = 1)
    BEGIN
            SET @TranStarted = 0
            COMMIT TRANSACTION
    END
    SET @TranStarted = 0

    RETURN 0

Cleanup:
    IF (@TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
            ROLLBACK TRANSACTION
    END
    RETURN @ErrorCode
END
```

Procedure: aspnet_Profile_GetNumberOfInactiveProfiles

Description

Queries the aspnet_Profile table to get a count of profiles whose last activity dates (in the aspnet_Users table) fall on or before the specified date.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@ProfileAuthOptions	int	Input
@InactiveSinceDate	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Profile_GetNumberOfInactiveProfiles
    @ApplicationName        nvarchar(256),
    @ProfileAuthOptions     int,
    @InactiveSinceDate      datetime
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
    BEGIN
        SELECT 0
        RETURN
    END

    SELECT  COUNT(*)
    FROM    dbo.aspnet_Users u, dbo.aspnet_Profile p
    WHERE   ApplicationId = @ApplicationId
        AND u.UserId = p.UserId
        AND (LastActivityDate <= @InactiveSinceDate)
        AND (
                (@ProfileAuthOptions = 2)
                OR (@ProfileAuthOptions = 0 AND IsAnonymous = 1)
                OR (@ProfileAuthOptions = 1 AND IsAnonymous = 0)
            )
END
```

Procedure: aspnet_Profile_GetProfiles

Description

Retrieves profile data from the aspnet_Profile table for users who match the criteria input to the stored procedure.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@ProfileAuthOptions	int	Input
@PageIndex	int	Input
@PageSize	int	Input
@UserNameToMatch	nvarchar	Input
@InactiveSinceDate	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Profile_GetProfiles
    @ApplicationName        nvarchar(256),
    @ProfileAuthOptions     int,
    @PageIndex              int,
    @PageSize               int,
    @UserNameToMatch        nvarchar(256) = NULL,
    @InactiveSinceDate      datetime      = NULL
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN

    -- Set the page bounds
    DECLARE @PageLowerBound int
    DECLARE @PageUpperBound int
    DECLARE @TotalRecords   int
    SET @PageLowerBound = @PageSize * @PageIndex
    SET @PageUpperBound = @PageSize - 1 + @PageLowerBound

    -- Create a temp table TO store the select results
    CREATE TABLE #PageIndexForUsers
    (
        IndexId int IDENTITY (0, 1) NOT NULL,
        UserId uniqueidentifier
    )

    -- Insert into our temp table
    INSERT INTO #PageIndexForUsers (UserId)
        SELECT  u.UserId
        FROM    dbo.aspnet_Users u, dbo.aspnet_Profile p
        WHERE   ApplicationId = @ApplicationId
            AND u.UserId = p.UserId
            AND (@InactiveSinceDate IS NULL OR LastActivityDate <= @InactiveSinceDate)
```

Definition

```
            AND (     (@ProfileAuthOptions = 2)
                   OR (@ProfileAuthOptions = 0 AND IsAnonymous = 1)
                   OR (@ProfileAuthOptions = 1 AND IsAnonymous = 0)
                 )
            AND (@UserNameToMatch IS NULL OR LoweredUserName LIKE LOWER(@UserNameToMatch))
        ORDER BY UserName

    SELECT  u.UserName, u.IsAnonymous, u.LastActivityDate, p.LastUpdatedDate,
            DATALENGTH(p.PropertyNames) + DATALENGTH(p.PropertyValuesString) +
DATALENGTH(p.PropertyValuesBinary)
    FROM    dbo.aspnet_Users u, dbo.aspnet_Profile p, #PageIndexForUsers i
    WHERE   u.UserId = p.UserId AND p.UserId = i.UserId AND i.IndexId >= @PageLowerBound AND i.IndexId
<= @PageUpperBound

    SELECT COUNT(*)
    FROM   #PageIndexForUsers

    DROP TABLE #PageIndexForUsers
END
```

Procedure: aspnet_Profile_GetProperties

Description

Retrieves profile data for the specified user.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Profile_GetProperties
   @ApplicationName      nvarchar(256),
   @UserName             nvarchar(256),
   @CurrentTimeUtc       datetime
AS
BEGIN
   DECLARE @ApplicationId uniqueidentifier
   SELECT  @ApplicationId = NULL
   SELECT  @ApplicationId = ApplicationId FROM dbo.aspnet_Applications WHERE
LOWER(@ApplicationName) = LoweredApplicationName
   IF (@ApplicationId IS NULL)
      RETURN

   DECLARE @UserId uniqueidentifier
   SELECT  @UserId = NULL

   SELECT @UserId = UserId
   FROM   dbo.aspnet_Users
   WHERE  ApplicationId = @ApplicationId AND LoweredUserName = LOWER(@UserName)

   IF (@UserId IS NULL)
      RETURN
   SELECT TOP 1 PropertyNames, PropertyValuesString, PropertyValuesBinary
   FROM        dbo.aspnet_Profile
   WHERE       UserId = @UserId

   IF (@@ROWCOUNT > 0)
   BEGIN
      UPDATE dbo.aspnet_Users
      SET    LastActivityDate=@CurrentTimeUtc
      WHERE  UserId = @UserId
   END
END
```

Procedure: aspnet_Profile_SetProperties

Description

Saves profile data for the specified user.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@PropertyNames	ntext	Input
@PropertyValuesString	ntext	Input
@PropertyValuesBinary	image	Input
@UserName	nvarchar	Input
@IsUserAnonymous	bit	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Profile_SetProperties
    @ApplicationName        nvarchar(256),
    @PropertyNames          ntext,
    @PropertyValuesString   ntext,
    @PropertyValuesBinary   image,
    @UserName               nvarchar(256),
    @IsUserAnonymous        bit,
    @CurrentTimeUtc         datetime
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL

    DECLARE @ErrorCode     int
    SET @ErrorCode = 0

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
       BEGIN TRANSACTION
       SET @TranStarted = 1
    END
    ELSE
            SET @TranStarted = 0

    EXEC dbo.aspnet_Applications_CreateApplication @ApplicationName, @ApplicationId OUTPUT

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    DECLARE @UserId uniqueidentifier
    DECLARE @LastActivityDate datetime
```

Definition

```
    SELECT  @UserId = NULL
    SELECT  @LastActivityDate = @CurrentTimeUtc

    SELECT @UserId = UserId
    FROM   dbo.aspnet_Users
    WHERE  ApplicationId = @ApplicationId AND LoweredUserName = LOWER(@UserName)
    IF (@UserId IS NULL)
        EXEC dbo.aspnet_Users_CreateUser @ApplicationId, @UserName, @IsUserAnonymous,
@LastActivityDate, @UserId OUTPUT

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    UPDATE dbo.aspnet_Users
    SET    LastActivityDate=@CurrentTimeUtc
    WHERE  UserId = @UserId

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    IF (EXISTS( SELECT *
           FROM   dbo.aspnet_Profile
           WHERE  UserId = @UserId))
        UPDATE dbo.aspnet_Profile
        SET    PropertyNames=@PropertyNames, PropertyValuesString = @PropertyValuesString,
               PropertyValuesBinary = @PropertyValuesBinary, LastUpdatedDate=@CurrentTimeUtc
        WHERE  UserId = @UserId
    ELSE
        INSERT INTO dbo.aspnet_Profile(UserId, PropertyNames, PropertyValuesString, PropertyValuesBinary,
LastUpdatedDate)
             VALUES (@UserId, @PropertyNames, @PropertyValuesString, @PropertyValuesBinary,
@CurrentTimeUtc)

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    IF( @TranStarted = 1 )
    BEGIN
            SET @TranStarted = 0
            COMMIT TRANSACTION
    END

    RETURN 0

Cleanup:

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
            ROLLBACK TRANSACTION
    END
```

Definition

```
    RETURN @ErrorCode
END
```

Procedure: aspnet_RegisterSchemaVersion

Description

Registers the compatible schema required for the given feature.

Parameters

Name	Type	Direction
@Feature	nvarchar	Input
@CompatibleSchemaVersion	nvarchar	Input
@IsCurrentVersion	bit	Input
@RemoveIncompatibleSchema	bit	Input

Definition

```
CREATE PROCEDURE [dbo].aspnet_RegisterSchemaVersion
    @Feature                   nvarchar(128),
    @CompatibleSchemaVersion   nvarchar(128),
    @IsCurrentVersion          bit,
    @RemoveIncompatibleSchema  bit
AS
BEGIN
    IF( @RemoveIncompatibleSchema = 1 )
    BEGIN
        DELETE FROM dbo.aspnet_SchemaVersions WHERE Feature = LOWER( @Feature )
    END
    ELSE
    BEGIN
        IF( @IsCurrentVersion = 1 )
        BEGIN
            UPDATE dbo.aspnet_SchemaVersions
            SET IsCurrentVersion = 0
            WHERE Feature = LOWER( @Feature )
        END
    END

    INSERT  dbo.aspnet_SchemaVersions( Feature, CompatibleSchemaVersion, IsCurrentVersion )
    VALUES( LOWER( @Feature ), @CompatibleSchemaVersion, @IsCurrentVersion )
END
```

Procedure: aspnet_Roles_CreateRole

Description

Adds a role to the aspnet_Roles table and, if necessary, adds a new application to the aspnet_Applications table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@RoleName	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Roles_CreateRole
    @ApplicationName  nvarchar(256),
    @RoleName         nvarchar(256)
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL

    DECLARE @ErrorCode     int
    SET @ErrorCode = 0

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
        BEGIN TRANSACTION
        SET @TranStarted = 1
    END
    ELSE
        SET @TranStarted = 0

    EXEC dbo.aspnet_Applications_CreateApplication @ApplicationName, @ApplicationId OUTPUT

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    IF (EXISTS(SELECT RoleId FROM dbo.aspnet_Roles WHERE LoweredRoleName = LOWER(@RoleName)
AND ApplicationId = @ApplicationId))
    BEGIN
        SET @ErrorCode = 1
        GOTO Cleanup
    END

    INSERT INTO dbo.aspnet_Roles
            (ApplicationId, RoleName, LoweredRoleName)
     VALUES (@ApplicationId, @RoleName, LOWER(@RoleName))

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
```

Definition

```
        GOTO Cleanup
    END

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
        COMMIT TRANSACTION
    END

    RETURN(0)

Cleanup:

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
        ROLLBACK TRANSACTION
    END

    RETURN @ErrorCode

END
```

Procedure: aspnet_Roles_DeleteRole

Description

Removes a role from the aspnet_Roles table. Optionally deletes records referencing the deleted role from the aspnet_UsersInRoles table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@RoleName	nvarchar	Input
@DeleteOnlyIfRoleIsEmpty	bit	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Roles_DeleteRole
    @ApplicationName            nvarchar(256),
    @RoleName                   nvarchar(256),
    @DeleteOnlyIfRoleIsEmpty    bit
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN(1)

    DECLARE @ErrorCode     int
    SET @ErrorCode = 0

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
        BEGIN TRANSACTION
        SET @TranStarted = 1
    END
    ELSE
        SET @TranStarted = 0

    DECLARE @RoleId   uniqueidentifier
    SELECT  @RoleId = NULL
    SELECT  @RoleId = RoleId FROM dbo.aspnet_Roles WHERE LoweredRoleName = LOWER(@RoleName)
AND ApplicationId = @ApplicationId

    IF (@RoleId IS NULL)
    BEGIN
        SELECT @ErrorCode = 1
        GOTO Cleanup
    END
    IF (@DeleteOnlyIfRoleIsEmpty <> 0)
    BEGIN
        IF (EXISTS (SELECT RoleId FROM dbo.aspnet_UsersInRoles  WHERE @RoleId = RoleId))
        BEGIN
            SELECT @ErrorCode = 2
```

Definition

```
            GOTO Cleanup
        END
    END


    DELETE FROM dbo.aspnet_UsersInRoles  WHERE @RoleId = RoleId

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    DELETE FROM dbo.aspnet_Roles WHERE @RoleId = RoleId  AND ApplicationId = @ApplicationId

    IF( @@ERROR <> 0 )
    BEGIN
        SET @ErrorCode = -1
        GOTO Cleanup
    END

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
        COMMIT TRANSACTION
    END

    RETURN(0)

Cleanup:

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
        ROLLBACK TRANSACTION
    END

    RETURN @ErrorCode
END
```

Procedure: aspnet_Roles_GetAllRoles

Description

Retrieves all roles with the specified application ID from the aspnet_Roles table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Roles_GetAllRoles (
    @ApplicationName           nvarchar(256))
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN
    SELECT RoleName
    FROM   dbo.aspnet_Roles WHERE ApplicationId = @ApplicationId
    ORDER BY RoleName
END
```

Procedure: aspnet_Roles_RoleExists

Description

Checks the aspnet_Roles table to determine whether the specified role exists.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@RoleName	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_Roles_RoleExists
    @ApplicationName  nvarchar(256),
    @RoleName         nvarchar(256)
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN(0)
    IF (EXISTS (SELECT RoleName FROM dbo.aspnet_Roles WHERE LOWER(@RoleName) =
LoweredRoleName AND ApplicationId = @ApplicationId ))
        RETURN(1)
    ELSE
        RETURN(0)
END
```

Procedure: aspnet_Setup_RemoveAllRoleMembers

Description

Removes all roles from the given SQL account.

Parameters

Name	Type	Direction
@name	nvarchar	Input

Definition

```
CREATE PROCEDURE [dbo].aspnet_Setup_RemoveAllRoleMembers
   @name   sysname
AS
BEGIN
   CREATE TABLE #aspnet_RoleMembers
   (
      Group_name      sysname,
      Group_id        smallint,
      Users_in_group  sysname,
      User_id         smallint
   )

   INSERT INTO #aspnet_RoleMembers
   EXEC sp_helpuser @name

   DECLARE @user_id smallint
   DECLARE @cmd nvarchar(500)
   DECLARE c1 cursor FORWARD_ONLY FOR
      SELECT User_id FROM #aspnet_RoleMembers

   OPEN c1

   FETCH c1 INTO @user_id
   WHILE (@@fetch_status = 0)
   BEGIN
      SET @cmd = 'EXEC sp_droprolemember ' + '''' + @name + ''', ''' + USER_NAME(@user_id) + ''''
      EXEC (@cmd)
      FETCH c1 INTO @user_id
   END

   CLOSE c1
   DEALLOCATE c1
END
```

Procedure: aspnet_Setup_RestorePermissions

Description

Restores permissions to the given SQL account.

Parameters

Name	Type	Direction
@name	nvarchar	Input

Definition

```
CREATE PROCEDURE [dbo].aspnet_Setup_RestorePermissions
    @name   sysname
AS
BEGIN
    DECLARE @object sysname
    DECLARE @protectType char(10)
    DECLARE @action varchar(60)
    DECLARE @grantee sysname
    DECLARE @cmd nvarchar(500)
    DECLARE c1 cursor FORWARD_ONLY FOR
        SELECT Object, ProtectType, [Action], Grantee FROM #aspnet_Permissions where Object = @name

    OPEN c1

    FETCH c1 INTO @object, @protectType, @action, @grantee
    WHILE (@@fetch_status = 0)
    BEGIN
        SET @cmd = @protectType + ' ' + @action + ' on ' + @object + ' TO [' + @grantee + ']'
        EXEC (@cmd)
        FETCH c1 INTO @object, @protectType, @action, @grantee
    END

    CLOSE c1
    DEALLOCATE c1
END
```

Procedure: aspnet_UnRegisterSchemaVersion

Description

Unregisters the schema version for the given feature.

Parameters

Name	Type	Direction
@Feature	nvarchar	Input
@CompatibleSchemaVersion	nvarchar	Input

Definition

```
CREATE PROCEDURE [dbo].aspnet_UnRegisterSchemaVersion
    @Feature                   nvarchar(128),
    @CompatibleSchemaVersion   nvarchar(128)
AS
BEGIN
    DELETE FROM dbo.aspnet_SchemaVersions
        WHERE   Feature = LOWER(@Feature) AND @CompatibleSchemaVersion = CompatibleSchemaVersion
END
```

Procedure: aspnet_Users_CreateUser

Description

Adds a user to the aspnet_Users table. Called by aspnet_Membership_CreateUser.

Parameters

Name	Type	Direction
@ApplicationId	uniqueidentifier	Input
@UserName	nvarchar	Input
@IsUserAnonymous	bit	Input
@LastActivityDate	datetime	Input
@UserId	uniqueidentifier	Input/Output

Definition

```
CREATE PROCEDURE [dbo].aspnet_Users_CreateUser
    @ApplicationId    uniqueidentifier,
    @UserName         nvarchar(256),
    @IsUserAnonymous  bit,
    @LastActivityDate DATETIME,
    @UserId           uniqueidentifier OUTPUT
AS
BEGIN
    IF( @UserId IS NULL )
        SELECT @UserId = NEWID()
    ELSE
    BEGIN
        IF( EXISTS( SELECT UserId FROM dbo.aspnet_Users
                    WHERE @UserId = UserId ) )
            RETURN -1
    END

    INSERT dbo.aspnet_Users (ApplicationId, UserId, UserName, LoweredUserName, IsAnonymous,
LastActivityDate)
    VALUES (@ApplicationId, @UserId, @UserName, LOWER(@UserName), @IsUserAnonymous,
@LastActivityDate)

    RETURN 0
END
```

Procedure: aspnet_Users_DeleteUser

Description

Deletes a user from the aspnet_Membership table and optionally from other SQL provider tables, including aspnet_Users.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@TablesToDeleteFrom	int	Input
@NumTablesDeletedFrom	int	Input/Output

Definition

```
CREATE PROCEDURE [dbo].aspnet_Users_DeleteUser
    @ApplicationName  nvarchar(256),
    @UserName         nvarchar(256),
    @TablesToDeleteFrom int,
    @NumTablesDeletedFrom int OUTPUT
AS
BEGIN
    DECLARE @UserId               uniqueidentifier
    SELECT  @UserId               = NULL
    SELECT  @NumTablesDeletedFrom = 0

    DECLARE @TranStarted   bit
    SET @TranStarted = 0

    IF( @@TRANCOUNT = 0 )
    BEGIN
            BEGIN TRANSACTION
            SET @TranStarted = 1
    END
    ELSE
        SET @TranStarted = 0

    DECLARE @ErrorCode   int
    DECLARE @RowCount    int

    SET @ErrorCode = 0
    SET @RowCount  = 0

    SELECT  @UserId = u.UserId
    FROM    dbo.aspnet_Users u, dbo.aspnet_Applications a
    WHERE   u.LoweredUserName       = LOWER(@UserName)
        AND u.ApplicationId         = a.ApplicationId
        AND LOWER(@ApplicationName) = a.LoweredApplicationName

    IF (@UserId IS NULL)
    BEGIN
        GOTO Cleanup
    END

    -- Delete from Membership table if (@TablesToDeleteFrom & 1) is set
```

Definition

```
    IF ((@TablesToDeleteFrom & 1) <> 0 AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'vw_aspnet_MembershipUsers') AND
(type = 'V'))))
    BEGIN
        DELETE FROM dbo.aspnet_Membership WHERE @UserId = UserId

        SELECT @ErrorCode = @@ERROR,
               @RowCount = @@ROWCOUNT

        IF( @ErrorCode <> 0 )
            GOTO Cleanup

        IF (@RowCount <> 0)
            SELECT  @NumTablesDeletedFrom = @NumTablesDeletedFrom + 1
    END

    -- Delete from aspnet_UsersInRoles table if (@TablesToDeleteFrom & 2) is set
    IF ((@TablesToDeleteFrom & 2) <> 0  AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'vw_aspnet_UsersInRoles') AND (type =
'V'))) )
    BEGIN
        DELETE FROM dbo.aspnet_UsersInRoles WHERE @UserId = UserId

        SELECT @ErrorCode = @@ERROR,
                @RowCount = @@ROWCOUNT

        IF( @ErrorCode <> 0 )
            GOTO Cleanup

        IF (@RowCount <> 0)
            SELECT  @NumTablesDeletedFrom = @NumTablesDeletedFrom + 1
    END

    -- Delete from aspnet_Profile table if (@TablesToDeleteFrom & 4) is set
    IF ((@TablesToDeleteFrom & 4) <> 0  AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'vw_aspnet_Profiles') AND (type = 'V'))) )
    BEGIN
        DELETE FROM dbo.aspnet_Profile WHERE @UserId = UserId

        SELECT @ErrorCode = @@ERROR,
                @RowCount = @@ROWCOUNT

        IF( @ErrorCode <> 0 )
            GOTO Cleanup

        IF (@RowCount <> 0)
            SELECT  @NumTablesDeletedFrom = @NumTablesDeletedFrom + 1
    END

    -- Delete from aspnet_PersonalizationPerUser table if (@TablesToDeleteFrom & 8) is set
    IF ((@TablesToDeleteFrom & 8) <> 0  AND
        (EXISTS (SELECT name FROM sysobjects WHERE (name = N'vw_aspnet_WebPartState_User') AND
(type = 'V'))) )
    BEGIN
        DELETE FROM dbo.aspnet_PersonalizationPerUser WHERE @UserId = UserId

        SELECT @ErrorCode = @@ERROR,
                @RowCount = @@ROWCOUNT

        IF( @ErrorCode <> 0 )
```

Definition

```
            GOTO Cleanup

        IF (@RowCount <> 0)
            SELECT  @NumTablesDeletedFrom = @NumTablesDeletedFrom + 1
    END

    -- Delete from aspnet_Users table if (@TablesToDeleteFrom & 1,2,4 & 8) are all set
    IF ((@TablesToDeleteFrom & 1) <> 0 AND
        (@TablesToDeleteFrom & 2) <> 0 AND
        (@TablesToDeleteFrom & 4) <> 0 AND
        (@TablesToDeleteFrom & 8) <> 0 AND
        (EXISTS (SELECT UserId FROM dbo.aspnet_Users WHERE @UserId = UserId)))
    BEGIN
        DELETE FROM dbo.aspnet_Users WHERE @UserId = UserId

        SELECT @ErrorCode = @@ERROR,
                @RowCount = @@ROWCOUNT

        IF( @ErrorCode <> 0 )
            GOTO Cleanup

        IF (@RowCount <> 0)
            SELECT  @NumTablesDeletedFrom = @NumTablesDeletedFrom + 1
    END

    IF( @TranStarted = 1 )
    BEGIN
            SET @TranStarted = 0
            COMMIT TRANSACTION
    END

    RETURN 0

Cleanup:
    SET @NumTablesDeletedFrom = 0

    IF( @TranStarted = 1 )
    BEGIN
        SET @TranStarted = 0
            ROLLBACK TRANSACTION
    END

    RETURN @ErrorCode

END
```

Procedure: aspnet_UsersInRoles_AddUsersToRoles

Description

Adds the specified users to the specified roles by adding them to the aspnet_UsersInRoles table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserNames	nvarchar	Input
@RoleNames	nvarchar	Input
@CurrentTimeUtc	datetime	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_UsersInRoles_AddUsersToRoles
	@ApplicationName  nvarchar(256),
	@UserNames		  nvarchar(4000),
	@RoleNames		  nvarchar(4000),
	@CurrentTimeUtc   datetime
AS
BEGIN
	DECLARE @AppId uniqueidentifier
	SELECT  @AppId = NULL
	SELECT  @AppId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName) =
LoweredApplicationName
	IF (@AppId IS NULL)
		RETURN(2)
	DECLARE @TranStarted   bit
	SET @TranStarted = 0

	IF( @@TRANCOUNT = 0 )
	BEGIN
		BEGIN TRANSACTION
		SET @TranStarted = 1
	END

	DECLARE @tbNames  table(Name nvarchar(256) NOT NULL PRIMARY KEY)
	DECLARE @tbRoles  table(RoleId uniqueidentifier NOT NULL PRIMARY KEY)
	DECLARE @tbUsers  table(UserId uniqueidentifier NOT NULL PRIMARY KEY)
	DECLARE @Num		int
	DECLARE @Pos		int
	DECLARE @NextPos	int
	DECLARE @Name		nvarchar(256)

	SET @Num = 0
	SET @Pos = 1
	WHILE(@Pos <= LEN(@RoleNames))
	BEGIN
		SELECT @NextPos = CHARINDEX(N',', @RoleNames,  @Pos)
		IF (@NextPos = 0 OR @NextPos IS NULL)
			SELECT @NextPos = LEN(@RoleNames) + 1
		SELECT @Name = RTRIM(LTRIM(SUBSTRING(@RoleNames, @Pos, @NextPos - @Pos)))
		SELECT @Pos = @NextPos+1

		INSERT INTO @tbNames VALUES (@Name)
```

Definition

```
			SET @Num = @Num + 1
		END

		INSERT INTO @tbRoles
		  SELECT RoleId
		  FROM   dbo.aspnet_Roles ar, @tbNames t
		  WHERE  LOWER(t.Name) = ar.LoweredRoleName AND ar.ApplicationId = @AppId

		IF (@@ROWCOUNT <> @Num)
		BEGIN
			SELECT TOP 1 Name
			FROM   @tbNames
			WHERE  LOWER(Name) NOT IN (SELECT ar.LoweredRoleName FROM dbo.aspnet_Roles
ar,  @tbRoles r WHERE r.RoleId = ar.RoleId)
			IF( @TranStarted = 1 )
				ROLLBACK TRANSACTION
			RETURN(2)
		END

		DELETE FROM @tbNames WHERE 1=1
		SET @Num = 0
		SET @Pos = 1

		WHILE(@Pos <= LEN(@UserNames))
		BEGIN
			SELECT @NextPos = CHARINDEX(N',', @UserNames,  @Pos)
			IF (@NextPos = 0 OR @NextPos IS NULL)
				SELECT @NextPos = LEN(@UserNames) + 1
			SELECT @Name = RTRIM(LTRIM(SUBSTRING(@UserNames, @Pos, @NextPos - @Pos)))
			SELECT @Pos = @NextPos+1

			INSERT INTO @tbNames VALUES (@Name)
			SET @Num = @Num + 1
		END

		INSERT INTO @tbUsers
		  SELECT UserId
		  FROM   dbo.aspnet_Users ar, @tbNames t
		  WHERE  LOWER(t.Name) = ar.LoweredUserName AND ar.ApplicationId = @AppId

		IF (@@ROWCOUNT <> @Num)
		BEGIN
			DELETE FROM @tbNames
			WHERE LOWER(Name) IN (SELECT LoweredUserName FROM dbo.aspnet_Users au,
@tbUsers u WHERE au.UserId = u.UserId)

			INSERT dbo.aspnet_Users (ApplicationId, UserId, UserName, LoweredUserName,
IsAnonymous, LastActivityDate)
			  SELECT @AppId, NEWID(), Name, LOWER(Name), 0, @CurrentTimeUtc
			  FROM   @tbNames

			INSERT INTO @tbUsers
			  SELECT  UserId
			  FROMdbo.aspnet_Users au, @tbNames t
			  WHERE   LOWER(t.Name) = au.LoweredUserName AND au.ApplicationId = @AppId
		END

		IF (EXISTS (SELECT * FROM dbo.aspnet_UsersInRoles ur, @tbUsers tu, @tbRoles tr WHERE
tu.UserId = ur.UserId AND tr.RoleId = ur.RoleId))
		BEGIN
```

Definition

```
		SELECT TOP 1 UserName, RoleName
		FROM		dbo.aspnet_UsersInRoles ur, @tbUsers tu, @tbRoles tr, aspnet_Users u,
aspnet_Roles r
		WHERE		u.UserId = tu.UserId AND r.RoleId = tr.RoleId AND tu.UserId =
ur.UserId AND tr.RoleId = ur.RoleId

		IF( @TranStarted = 1 )
			ROLLBACK TRANSACTION
		RETURN(3)
	END

	INSERT INTO dbo.aspnet_UsersInRoles (UserId, RoleId)
	SELECT UserId, RoleId
	FROM @tbUsers, @tbRoles

	IF( @TranStarted = 1 )
		COMMIT TRANSACTION
	RETURN(0)
END
```

Procedure: aspnet_UsersInRoles_FindUsersInRole

Description

Queries the aspnet_UsersInRoles table for all users belonging to the specified role whose user names match the specified pattern.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@RoleName	nvarchar	Input
@UserNameToMatch	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_UsersInRoles_FindUsersInRole
    @ApplicationName  nvarchar(256),
    @RoleName         nvarchar(256),
    @UserNameToMatch  nvarchar(256)
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN(1)
     DECLARE @RoleId uniqueidentifier
     SELECT  @RoleId = NULL

     SELECT  @RoleId = RoleId
     FROM    dbo.aspnet_Roles
     WHERE   LOWER(@RoleName) = LoweredRoleName AND ApplicationId = @ApplicationId

     IF (@RoleId IS NULL)
         RETURN(1)

    SELECT u.UserName
    FROM   dbo.aspnet_Users u, dbo.aspnet_UsersInRoles ur
    WHERE  u.UserId = ur.UserId AND @RoleId = ur.RoleId AND u.ApplicationId = @ApplicationId AND
LoweredUserName LIKE LOWER(@UserNameToMatch)
    ORDER BY u.UserName
    RETURN(0)
END
```

Procedure: aspnet_UsersInRoles_GetRolesForUser

Description

Queries the aspnet_UsersInRoles table for all roles assigned to a specified user.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_UsersInRoles_GetRolesForUser
    @ApplicationName  nvarchar(256),
    @UserName         nvarchar(256)
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN(1)
    DECLARE @UserId uniqueidentifier
    SELECT  @UserId = NULL

    SELECT  @UserId = UserId
    FROM    dbo.aspnet_Users
    WHERE   LoweredUserName = LOWER(@UserName) AND ApplicationId = @ApplicationId

    IF (@UserId IS NULL)
        RETURN(1)

    SELECT r.RoleName
    FROM   dbo.aspnet_Roles r, dbo.aspnet_UsersInRoles ur
    WHERE  r.RoleId = ur.RoleId AND r.ApplicationId = @ApplicationId AND ur.UserId = @UserId
    ORDER BY r.RoleName
    RETURN (0)
END
```

Procedure: aspnet_UsersInRoles_GetUsersInRoles

Description

Queries the aspnet_UsersInRoles table for all users belonging to the specified role.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@RoleName	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_UsersInRoles_GetUsersInRoles
    @ApplicationName  nvarchar(256),
    @RoleName         nvarchar(256)
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN(1)
     DECLARE @RoleId uniqueidentifier
     SELECT  @RoleId = NULL

     SELECT  @RoleId = RoleId
     FROM    dbo.aspnet_Roles
     WHERE   LOWER(@RoleName) = LoweredRoleName AND ApplicationId = @ApplicationId

     IF (@RoleId IS NULL)
         RETURN(1)

    SELECT u.UserName
    FROM   dbo.aspnet_Users u, dbo.aspnet_UsersInRoles ur
    WHERE  u.UserId = ur.UserId AND @RoleId = ur.RoleId AND u.ApplicationId = @ApplicationId
    ORDER BY u.UserName
    RETURN(0)
END
```

Procedure: aspnet_UsersInRoles_IsUserInRole

Description

Checks the aspnet_UsersInRoles table to determine whether the specified user belongs to the specified role.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserName	nvarchar	Input
@RoleName	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_UsersInRoles_IsUserInRole
    @ApplicationName  nvarchar(256),
    @UserName         nvarchar(256),
    @RoleName         nvarchar(256)
AS
BEGIN
    DECLARE @ApplicationId uniqueidentifier
    SELECT  @ApplicationId = NULL
    SELECT  @ApplicationId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName)
= LoweredApplicationName
    IF (@ApplicationId IS NULL)
        RETURN(2)
    DECLARE @UserId uniqueidentifier
    SELECT  @UserId = NULL
    DECLARE @RoleId uniqueidentifier
    SELECT  @RoleId = NULL

    SELECT  @UserId = UserId
    FROM    dbo.aspnet_Users
    WHERE   LoweredUserName = LOWER(@UserName) AND ApplicationId = @ApplicationId

    IF (@UserId IS NULL)
        RETURN(2)

    SELECT  @RoleId = RoleId
    FROM    dbo.aspnet_Roles
    WHERE   LoweredRoleName = LOWER(@RoleName) AND ApplicationId = @ApplicationId

    IF (@RoleId IS NULL)
        RETURN(3)

    IF (EXISTS( SELECT * FROM dbo.aspnet_UsersInRoles WHERE  UserId = @UserId AND RoleId =
@RoleId))
        RETURN(1)
    ELSE
        RETURN(0)
END
```

Procedure: aspnet_UsersInRoles_RemoveUsersFromRoles

Description

Removes the specified users from the specified roles by deleting the corresponding records from the aspnet_UsersInRoles table.

Parameters

Name	Type	Direction
@ApplicationName	nvarchar	Input
@UserNames	nvarchar	Input
@RoleNames	nvarchar	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_UsersInRoles_RemoveUsersFromRoles
        @ApplicationName  nvarchar(256),
        @UserNames              nvarchar(4000),
        @RoleNames              nvarchar(4000)
AS
BEGIN
        DECLARE @AppId uniqueidentifier
        SELECT  @AppId = NULL
        SELECT  @AppId = ApplicationId FROM aspnet_Applications WHERE LOWER(@ApplicationName) =
LoweredApplicationName
        IF (@AppId IS NULL)
                RETURN(2)


        DECLARE @TranStarted   bit
        SET @TranStarted = 0

        IF( @@TRANCOUNT = 0 )
        BEGIN
                BEGIN TRANSACTION
                SET @TranStarted = 1
        END

        DECLARE @tbNames  table(Name nvarchar(256) NOT NULL PRIMARY KEY)
        DECLARE @tbRoles  table(RoleId uniqueidentifier NOT NULL PRIMARY KEY)
        DECLARE @tbUsers  table(UserId uniqueidentifier NOT NULL PRIMARY KEY)
        DECLARE @Num            int
        DECLARE @Pos            int
        DECLARE @NextPos  int
        DECLARE @Name           nvarchar(256)
        DECLARE @CountAll int
        DECLARE @CountU         int
        DECLARE @CountR         int


        SET @Num = 0
        SET @Pos = 1
        WHILE(@Pos <= LEN(@RoleNames))
        BEGIN
                SELECT @NextPos = CHARINDEX(N',', @RoleNames,  @Pos)
                IF (@NextPos = 0 OR @NextPos IS NULL)
                        SELECT @NextPos = LEN(@RoleNames) + 1
```

Definition

```
                SELECT @Name = RTRIM(LTRIM(SUBSTRING(@RoleNames, @Pos, @NextPos - @Pos)))
                SELECT @Pos = @NextPos+1

                INSERT INTO @tbNames VALUES (@Name)
                SET @Num = @Num + 1
        END

        INSERT INTO @tbRoles
          SELECT RoleId
          FROM   dbo.aspnet_Roles ar, @tbNames t
          WHERE  LOWER(t.Name) = ar.LoweredRoleName AND ar.ApplicationId = @AppId
        SELECT @CountR = @@ROWCOUNT

        IF (@CountR <> @Num)
        BEGIN
                SELECT TOP 1 N'', Name
                FROM   @tbNames
                WHERE  LOWER(Name) NOT IN (SELECT ar.LoweredRoleName FROM dbo.aspnet_Roles
ar,  @tbRoles r WHERE r.RoleId = ar.RoleId)
                IF( @TranStarted = 1 )
                        ROLLBACK TRANSACTION
                RETURN(2)
        END


        DELETE FROM @tbNames WHERE 1=1
        SET @Num = 0
        SET @Pos = 1


        WHILE(@Pos <= LEN(@UserNames))
        BEGIN
                SELECT @NextPos = CHARINDEX(N',', @UserNames,  @Pos)
                IF (@NextPos = 0 OR @NextPos IS NULL)
                        SELECT @NextPos = LEN(@UserNames) + 1
                SELECT @Name = RTRIM(LTRIM(SUBSTRING(@UserNames, @Pos, @NextPos - @Pos)))
                SELECT @Pos = @NextPos+1

                INSERT INTO @tbNames VALUES (@Name)
                SET @Num = @Num + 1
        END

        INSERT INTO @tbUsers
          SELECT UserId
          FROM   dbo.aspnet_Users ar, @tbNames t
          WHERE  LOWER(t.Name) = ar.LoweredUserName AND ar.ApplicationId = @AppId

        SELECT @CountU = @@ROWCOUNT
        IF (@CountU <> @Num)
        BEGIN
                SELECT TOP 1 Name, N''
                FROM   @tbNames
                WHERE  LOWER(Name) NOT IN (SELECT au.LoweredUserName FROM dbo.aspnet_Users
au,  @tbUsers u WHERE u.UserId = au.UserId)

                IF( @TranStarted = 1 )
                        ROLLBACK TRANSACTION
                RETURN(1)
        END
```

Definition

```
	SELECT  @CountAll = COUNT(*)
	FROM  dbo.aspnet_UsersInRoles ur, @tbUsers u, @tbRoles r
	WHERE   ur.UserId = u.UserId AND ur.RoleId = r.RoleId

	IF (@CountAll <> @CountU * @CountR)
	BEGIN
		SELECT TOP 1 UserName, RoleName
		FROM		@tbUsers tu, @tbRoles tr, dbo.aspnet_Users u, dbo.aspnet_Roles r
		WHERE		u.UserId = tu.UserId AND r.RoleId = tr.RoleId AND
					tu.UserId NOT IN (SELECT ur.UserId FROM dbo.aspnet_UsersInRoles
ur WHERE ur.RoleId = tr.RoleId) AND
					tr.RoleId NOT IN (SELECT ur.RoleId FROM dbo.aspnet_UsersInRoles
ur WHERE ur.UserId = tu.UserId)
		IF( @TranStarted = 1 )
			ROLLBACK TRANSACTION
		RETURN(3)
	END

	DELETE FROM dbo.aspnet_UsersInRoles
	WHERE UserId IN (SELECT UserId FROM @tbUsers)
	  AND RoleId IN (SELECT RoleId FROM @tbRoles)
	IF( @TranStarted = 1 )
		COMMIT TRANSACTION
	RETURN(0)
END
```

Procedure: aspnet_WebEvent_LogEvent

Description

Records a Web event in the aspnet_WebEvents_Events table.

Parameters

Name	Type	Direction
@EventId	char	Input
@EventTimeUtc	datetime	Input
@EventTime	datetime	Input
@EventType	nvarchar	Input
@EventSequence	decimal	Input
@EventOccurrence	decimal	Input
@EventCode	int	Input
@EventDetailCode	int	Input
@Message	nvarchar	Input
@ApplicationPath	nvarchar	Input
@ApplicationVirtualPath	nvarchar	Input
@MachineName	nvarchar	Input
@RequestUrl	nvarchar	Input
@ExceptionType	nvarchar	Input
@Details	ntext	Input

Definition

```
CREATE PROCEDURE dbo.aspnet_WebEvent_LogEvent
        @EventId         char(32),
        @EventTimeUtc    datetime,
        @EventTime       datetime,
        @EventType       nvarchar(256),
        @EventSequence   decimal(19,0),
        @EventOccurrence decimal(19,0),
        @EventCode       int,
        @EventDetailCode int,
        @Message         nvarchar(1024),
        @ApplicationPath nvarchar(256),
        @ApplicationVirtualPath nvarchar(256),
        @MachineName    nvarchar(256),
        @RequestUrl      nvarchar(1024),
        @ExceptionType   nvarchar(256),
        @Details         ntext
AS
BEGIN
    INSERT
        dbo.aspnet_WebEvent_Events
        (
            EventId,
            EventTimeUtc,
            EventTime,
```

Definition

```
            EventType,
            EventSequence,
            EventOccurrence,
            EventCode,
            EventDetailCode,
            Message,
            ApplicationPath,
            ApplicationVirtualPath,
            MachineName,
            RequestUrl,
            ExceptionType,
            Details
        )
    VALUES
    (
        @EventId,
        @EventTimeUtc,
        @EventTime,
        @EventType,
        @EventSequence,
        @EventOccurrence,
        @EventCode,
        @EventDetailCode,
        @Message,
        @ApplicationPath,
        @ApplicationVirtualPath,
        @MachineName,
        @RequestUrl,
        @ExceptionType,
        @Details
    )
END
```

Index