PTSLibrary Reference

Table of Contents

PTSLibrary Reference

Namespaces

PTSLibrary Namespace

Classes

Customer[6], PTSAdminFacade[7], PTSClientFacade[13], PTSCustomerFacade[16], PTSSuperFacade[18], Project[20], Task[27], Team[31], TeamLeader[35], User[38]

Enumerations

Status[40]

Customer Class

Represents a customer who has commissioned a project with the company.

System.Object
  PTSLibrary.User[38]
    PTSLibrary.Customer

C#

```
public class Customer : User
```

Remarks

This is a subclass of User superclass./>

Requirements

Namespace:PTSLibrary[6]

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

Customer[6]

Properties

Id[39] (inherited from User), Name[40] (inherited from User)

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

id[39] (inherited from User), name[40] (inherited from User)

Customer Constructor

Constructor.

C#

```
public Customer(
```

```
    string name,
    int id
)
```

Parameters

name

The name of the customer.

id

The identifier .

Remarks

Takes two arguments and assigns them to the inherited variables.

Source code

```
public Customer(string name, int id)
{
    this.name = name;
    this.id = id;
}
```

See Also

Applies to: Customer[6]

PTSAdminFacade Class

The administrator facade.

System.Object
  PTSLibrary.PTSSuperFacade[18]
    PTSLibrary.PTSAdminFacade

C#

```
public class PTSAdminFacade : PTSSuperFacade
```

Remarks

Provides an interface for administrators to access and manage project data.

Requirements

Namespace:PTSLibrary[6]

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

PTSAdminFacade[8]

Methods

Authenticate8, CreateProject10, CreateTask10, Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetListOfCustomers12, GetListOfProjects12, GetListOfTasks19 (inherited from PTSSuperFacade), GetListOfTeams12, GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

dao12

PTSAdminFacade Constructor

Default constructor.

C#

```
public PTSAdminFacade()
```

Remarks

Calls SuperDAO constructor with a new AdminDAO.

Source code

```
public PTSAdminFacade() : base(new DAO.AdminDAO())
{
    dao = (DAO.AdminDAO)base.dao;
}
```

See Also

Applies to: PTSAdminFacade7

PTSAdminFacade.Authenticate Method

Authenticates the administrator.

C#

```
public int Authenticate(
   string username,
   string password
)
```

Parameters

username

The administartor username.

password

The password.

Returns

Administrator Id. Returns '0' if auth fails.

Exceptions

Exception type	Condition
Exception	Thrown when username or password are empty.

Source code

```
public int Authenticate(string username, string password)
{
    if (username == "" || password == "")
    {
        throw new Exception("Missing Data");
    }
    return dao.Authenticate(username, password);
}
```

See Also

Applies to: PTSAdminFacade7

PTSAdminFacade.CreateProject Method

Creates a new project.

C#

```
public void CreateProject(
   string name,
   DateTime startDate,
   DateTime endDate,
   int customerId,
   int administratorId
)
```

Parameters

name

startDate

endDate

customerId

administratorId

Exceptions

Exception type	Condition
Exception	Thrown when name or dates are missing.

Source code

```
 public void CreateProject(string name, DateTime startDate, DateTime endDate,
int customerId, int administratorId)
 {
     if (name == "" || name == null || startDate == null || endDate == null)
     {
         throw new Exception("Missing Project data.");
     }
     dao.CreateProject(name, startDate, endDate, customerId, administratorId);
 }
```

See Also

Applies to: PTSAdminFacade7

PTSAdminFacade.CreateTask Method

Creates a new task within a project.

C#

```
public void CreateTask(
   string name,
   DateTime startDate,
   DateTime endDate,
   int teamId,
   Guid projectId
)
```

Parameters

name

startDate

endDate

teamId

projectId

Exceptions

Exception type	Condition
Exception	Thrown when name or dates are missing.

Source code

```
 public void CreateTask(string name, DateTime startDate, DateTime endDate, int
teamId, Guid projectId)
 {
     if (name == null || name == "" || startDate == null || endDate == null)
     {
         throw new Exception("Missing Data");
     }
     dao.CreateTask(name, startDate, endDate, teamId, projectId);
 }
```

See Also

Applies to: PTSAdminFacade7

PTSAdminFacade.GetListOfCustomers Method

Gets list of customers.

C#

```
public Customer[] GetListOfCustomers()
```

Returns

An array of customer.

Source code

```
 public Customer[] GetListOfCustomers()
 {
     return dao.GetListOfCustomers().ToArray();
 }
```

See Also

Applies to: PTSAdminFacade7

PTSAdminFacade.GetListOfProjects Method

Gets list of projects.

C#

```
public Project[] GetListOfProjects(
   int adminId
)
```

Parameters

adminId

Identifier for the admin.

Returns

An array of projects managed by the administrator.

Source code

```
public Project[] GetListOfProjects(int adminId)
{
    return dao.GetListOfProjects(adminId).ToArray();
}
```

See Also

Applies to: PTSAdminFacade7

PTSAdminFacade.GetListOfTeams Method

Gets list of teams.

C#

```
public Team[] GetListOfTeams()
```

Returns

An array of team.

Source code

```
public Team[] GetListOfTeams()
{
    return dao.GetListOfTeams().ToArray();
}
```

See Also

Applies to: PTSAdminFacade7

dao Field

The administrator dao.

C#

```
private AdminDAO dao
```

Source code

```
private  DAO.AdminDAO dao;
```

See Also

Applies to: PTSAdminFacade[7]

PTSClientFacade Class

The client facade.

System.Object
  PTSLibrary.PTSSuperFacade[18]
    PTSLibrary.PTSClientFacade

C#

```
public class PTSClientFacade : PTSSuperFacade
```

Remarks

Allows a team leader to access project data.

Requirements

Namespace:PTSLibrary[6]

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

PTSClientFacade[13]

Methods

Authenticate[14], Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetListOfProjects[15], GetListOfTasks[19] (inherited from PTSSuperFacade), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

dao[15]

PTSClientFacade Constructor

Default client facade constructor.

C#

```
public PTSClientFacade()
```

Remarks

Takes no arguments. Calls the SuperDAO constructor with a ClientDAO as the argument.

Source code

```
public PTSClientFacade() : base(new DAO.ClientDAO())
{
    dao = (DAO.ClientDAO)base.dao;
}
```

See Also

Applies to: PTSClientFacade[13]

PTSClientFacade.Authenticate Method

Authenticates the team leader.

C#

```
public TeamLeader Authenticate(
   string username,
   string password
)
```

Parameters

username

The username.

password

The password.

Returns

A TeamLeader.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Source code

```
public TeamLeader Authenticate(string username, string password)
{
    if (username == "" || password == "")
    {
        throw new Exception("Missing Data");
    }
    return dao.Authenticate(username, password);
}
```

See Also

Applies to: PTSClientFacade13

PTSClientFacade.GetListOfProjects Method

Gets list of projects.

C#

```
public Project[] GetListOfProjects(
   int teamId
)
```

Parameters

teamId

Identifier for the team.

Returns

An array of project.

Remarks

Gets a list of projects for the team from dao then converts the list to an Array.

Source code

```
public Project[] GetListOfProjects(int teamId)
{
    return (dao.GetListOfProjects(teamId).ToArray());
}
```

See Also

Applies to: PTSClientFacade13

dao Field

The database access object for the Client.

C#

```
private ClientDAO dao
```

Source code

```
private DAO.ClientDAO dao;
```

See Also

Applies to: PTSClientFacade13

PTSCustomerFacade Class

The customer facade.

System.Object
  PTSLibrary.PTSSuperFacade[18]
    PTSLibrary.PTSCustomerFacade

C#

```
public class PTSCustomerFacade : PTSSuperFacade
```

Remarks

Interface used by customers to access data.

Requirements

Namespace:PTSLibrary[6]

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

PTSCustomerFacade[16]

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetListOfProjects[17], GetListOfTasks[19] (inherited from PTSSuperFacade), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

dao[17]

PTSCustomerFacade Constructor

Default constructor.

C#

```
public PTSCustomerFacade()
```

Remarks

Takes no arguments .

Source code

```
public PTSCustomerFacade() : base(new DAO.CustomerDAO())
{
    dao = (DAO.CustomerDAO)base.dao;
}
```

See Also

Applies to: PTSCustomerFacade16

PTSCustomerFacade.GetListOfProjects Method

Gets list of projects.

C#

```
public Project[] GetListOfProjects(
   int customerId
)
```

Parameters

customerId

Identifier for the customer.

Returns

An array of projects.

Remarks

Gets the projects commissioned by the customer. The dao returns a list which is converted to an Array.

Source code

```
public Project[] GetListOfProjects(int customerId)
{
    return (dao.GetListOfProjects(customerId)).ToArray();
}
```

See Also

Applies to: PTSCustomerFacade16

dao Field

The database access object.

C#

```
private CustomerDAO dao
```

Source code

```
private DAO.CustomerDAO dao;
```

See Also

Applies to: PTSCustomerFacade16

PTSSuperFacade Class

The super facade.

System.Object
  PTSLibrary.PTSSuperFacade
    PTSLibrary.PTSAdminFacade7
    PTSLibrary.PTSClientFacade13
    PTSLibrary.PTSCustomerFacade16

C#

```
public class PTSSuperFacade
```

Remarks

Provides a public interface to the business component.

Requirements

Namespace:PTSLibrary6

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

PTSSuperFacade18

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetListOfTasks19, GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

dao19

PTSSuperFacade Constructor

Constructor.

C#

```
public PTSSuperFacade(
   SuperDAO dao
)
```

Parameters

dao

The DAO to be used.

Remarks

Creates a SuperFacade that can be used by all users.

Source code

```
public PTSSuperFacade(DAO.SuperDAO dao)
{
    this.dao = dao;
}
```

See Also

Applies to: PTSSuperFacade[18]

PTSSuperFacade.GetListOfTasks Method

Gets list of tasks.

C#

```
public Task[] GetListOfTasks(
   Guid projectId
)
```

Parameters

projectId

Identifier for the project.

Returns

An array of tasks.

Remarks

Method to retrieve tasks for a specific project.

Source code

```
public Task[] GetListOfTasks(Guid projectId)
{
    return (dao.GetListOfTasks(projectId)).ToArray();
}
```

See Also

Applies to: PTSSuperFacade[18]

dao Field

The SuperDAO used for data access.

C#

```
protected SuperDAO dao
```

Source code

```
protected DAO.SuperDAO dao;
```

See Also

Applies to: PTSSuperFacade[18]

Project Class

A project.

System.Object
PTSLibrary.Project

C#

```
public class Project
```

Remarks

Represents a project that has been commissioned with the company. Note that there is no default constructor.

Requirements

Namespace:PTSLibrary[6]

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

Project[21]

Properties

ExpectedEndDate[25], ExpectedStartDate[25], Name[25], ProjectId[26], Tasks[26], TheCustomer[26]

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

expectedEndDate[25], expectedStartDate[25], name[25], projectId[26], tasks[26], theCustomer[26]

Project(String, DateTime, DateTime, Guid, List<Task>) Constructor

Second Constructor.

C#

```
public Project(
    string name,
    DateTime startDate,
    DateTime endDate,
    Guid projectId,
    List<Task> tasks
)
```

Parameters

name

The project name.

startDate

The expected start date.

endDate

The expected end date.

projectId

The identifier of the project.

tasks

The list of tasks within the project.

Remarks

Has an extra parameter <param cref="tasks" />

Source code

```
 public Project(string name, DateTime startDate, DateTime endDate, Guid
projectId, List<Task> tasks)
 {
     this.name = name;
     this.expectedStartDate = startDate;
     this.expectedEndDate = endDate;
     this.projectId = projectId;
     this.tasks = tasks;
 }
```

See Also

Applies to: Project20

Project(String, DateTime, DateTime, Guid, Customer) Constructor

Constructor.

C#

```
public Project(
    string name,
    DateTime startDate,
    DateTime endDate,
```

```
    Guid projectId,
    Customer customer
)
```

Parameters

name

The project name.

startDate

The expected start date.

endDate

The expected end date.

projectId

The identifier of the project.

customer

The customer to who the project belongs.

Source code

```
 public Project(string name, DateTime startDate, DateTime endDate, Guid
projectId, Customer customer)
 {
     this.name = name;
     this.expectedStartDate = startDate;
     this.expectedEndDate = endDate;
     this.projectId = projectId;
     this.theCustomer = customer;
 }
```

See Also

Applies to: Project[20]

Project.ExpectedEndDate Property

Gets or sets the expected end date.

C#

```
public DateTime ExpectedEndDate {get; set;}
```

Property Value

The expected end date.

Source code

```
 public DateTime ExpectedEndDate { get => expectedEndDate; set =>
expectedEndDate = value; }
```

See Also

Applies to: Project20

Project.ExpectedStartDate Property

Gets or sets the expected start date.

C#

```
public DateTime ExpectedStartDate {get; set;}
```

Property Value

The expected start date.

Source code

```
 public DateTime ExpectedStartDate { get => expectedStartDate; set =>
expectedStartDate = value; }
```

See Also

Applies to: Project20

Project.Name Property

Gets or sets the name of the project.

C#

```
public string Name {get; set;}
```

Property Value

The name.

Source code

```
 public string Name { get => name; set => name = value; }
```

See Also

Applies to: Project20

Project.ProjectId Property

Gets the identifier of the project.

C#

```
public Guid ProjectId {get;}
```

Property Value

The identifier of the project.

Remarks

The projectId cannot be changed.

Source code

```
public Guid ProjectId { get => projectId; }
```

See Also

Applies to: Project20

Project.Tasks Property

Gets or sets the tasks.

C#

```
public List<Task> Tasks {get; set;}
```

Property Value

The list of tasks within the project.

Source code

```
public List<Task> Tasks { get => tasks; set => tasks = value; }
```

See Also

Applies to: Project20

Project.TheCustomer Property

Gets or sets the Customer.

C#

```
public Customer TheCustomer {get; set;}
```

Property Value

The customer who commissioned the project.

Source code

```
public Customer TheCustomer { get => theCustomer; set => theCustomer = value; }
```

See Also

Applies to: Project20

expectedEndDate Field

C#

```
private DateTime expectedEndDate
```

Source code

```
private DateTime expectedEndDate;
```

See Also

Applies to: Project20

expectedStartDate Field

C#

```
private DateTime expectedStartDate
```

Source code

```
private DateTime expectedStartDate;
```

See Also

Applies to: Project20

name Field

C#

```
private string name
```

Source code

```
private string name;
```

See Also

Applies to: Project20

projectId Field

C#

```
private Guid projectId
```

Source code

```
private Guid projectId;
```

See Also

Applies to: Project20

tasks Field

C#

```
private List<Task> tasks
```

Source code

```
private List<Task> tasks;
```

See Also

Applies to: Project20

theCustomer Field

C#

```
private Customer theCustomer
```

Source code

```
private Customer theCustomer;
```

See Also

Applies to: Project[20]

Task Class

Represents a task within a project. A project may have more than one task.

System.Object
  PTSLibrary.Task

C#

```
public class Task
```

Remarks

All properties are public.

Requirements

Namespace:PTSLibrary[6]

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

Task[27]

Properties

Name[30], NameAndStatus[28], TaskId[30], TheStatus[29]

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

name[30], status[30], taskId[30]

Task Constructor

The only Constructor.

C#

```
public Task(
   Guid id,
   string name,
   Status status
)
```

Parameters

id

The identifier.

name

The name of the task.

status

The status.

Remarks

Sets the basic properties of the task.

Source code

```
public Task(Guid id, string name, Status status)
{
    this.taskId = id;
    this.name = name;
    this.status = status;
}
```

See Also

Applies to: Task[27]

Task.Name Property

Gets or sets the name.

C#

```
public string Name {get; set;}
```

Property Value

The name of the task.

Source code

```
public string Name
{
    get { return name; }
    set { name = value; }
}
```

See Also

Applies to: Task[27]

Task.NameAndStatus Property

Gets the name and status.

C#

```
public string NameAndStatus {get;}
```

Property Value

The name and status formatted as one string.

Source code

```
public string NameAndStatus
{
    get { return name + " - " + status;  }
}
```

See Also

Applies to: Task[27]

Task.TaskId Property

Gets or sets the id.

C#

```
public Guid TaskId {get; set;}
```

Property Value

The identifier of the task.

Source code

```
public Guid TaskId
{
    get { return taskId; }
    set { taskId = value; }
}
```

See Also

Applies to: Task[27]

Task.TheStatus Property

Gets or sets the status.

C#

```
public Status TheStatus {get; set;}
```

Property Value

The status.

Source code

```
public Status TheStatus
{
    get { return status; }
    set { status = value; }
}
```

See Also

Applies to: Task27

name Field

C#

```
private string name
```

Source code

```
private string name;
```

See Also

Applies to: Task27

status Field

C#

```
private Status status
```

Source code

```
private Status status;
```

See Also

Applies to: Task27

taskId Field

C#

```
private Guid taskId
```

Source code

```
private Guid taskId;
```

See Also

Applies to: Task[27]

Team Class

Represents a team.

System.Object
PTSLibrary.Team

C#

```
public class Team
```

Remarks

Both internal and external teams are represented using this class.

Requirements

Namespace:PTSLibrary[6]

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

Team[31]

Properties

Leader[35], Location[35], Name[35], TeamId[34]

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

id[34], leader[35], location[35], name[35]

Team Constructor

Constructor.

C#

```
public Team(
    int id,
    string location,
    string name,
    TeamLeader leader
)
```

Parameters

id

The identifier.

location

The location where the team is based.

name

The name of the team.

leader

The team leader.

Remarks

Sets the properties of the team.

Source code

```
public Team(int id, string location, string name, TeamLeader leader)
{
    this.location = location;
    this.name = name;
    this.id = id;
    this.leader = leader;
}
```

See Also

Applies to: Team[31]

Team.Leader Property

Gets or sets the leader.

C#

```
public TeamLeader Leader {get; set;}
```

Property Value

The team leader.

Source code

```
public TeamLeader Leader
```

```
{
    get { return leader; }
    set { leader = value; }
}
```

See Also

Applies to: Team[31]

Team.Location Property

Gets or sets the location.

C#

```
public string Location {get; set;}
```

Property Value

The location where the team is based.

Source code

```
public string Location
{
    get { return location; }
    set { location = value; }
}
```

See Also

Applies to: Team[31]

Team.Name Property

Gets or sets the name.

C#

```
public string Name {get; set;}
```

Property Value

The name of the team.

Source code

```
public string Name
{
    get { return name; }
```

```
    set { name = value; }
}
```

See Also

Applies to: Team[31]

Team.TeamId Property

Gets or sets the identifier of the team.

C#

```
public int TeamId {get; set;}
```

Property Value

The identifier of the team.

Source code

```
public int TeamId
{
    get { return id; }
    set { id = value; }
}
```

See Also

Applies to: Team[31]

id Field

C#

```
private int id
```

Source code

```
private int id;
```

See Also

Applies to: Team[31]

leader Field

C#

```
private TeamLeader leader
```

Source code

```
private TeamLeader leader;
```

See Also

Applies to: Team31

location Field

C#

```
private string location
```

Source code

```
private string location, name;
```

See Also

Applies to: Team31

name Field

C#

```
private string name
```

Source code

```
private string location, name;
```

See Also

Applies to: Team31

TeamLeader Class

A team leader.

System.Object
  PTSLibrary.User38
    PTSLibrary.TeamLeader

C#

```
public class TeamLeader : User
```

Remarks

This is a subclass of the User superclass.

Requirements

Namespace:PTSLibrary6

Assembly: PTSLibrary (in PTSLibrary.dll)

Constructors

TeamLeader36

Properties

Id39 (inherited from User), Name40 (inherited from User), TeamId37

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

id39 (inherited from User), name40 (inherited from User), teamId37

TeamLeader Constructor

Constructor.

C#

```
public TeamLeader(
   string name,
   int id,
   int teamId
)
```

Parameters

name

The name of the team.

id

The user identifier of the team leader.

teamId

The identifier of the team led by the leader.

Remarks

Generates a TeamLeader object with the provided properties.

Source code

```
public TeamLeader(string name, int id, int teamId)
{
    this.name = name;
    this.id = id;
    this.teamId = teamId;
}
```

See Also

Applies to: TeamLeader[35]

TeamLeader.TeamId Property

Gets or sets the id of the team.

C#

```
public int TeamId {get; set;}
```

Property Value

The identifier of the leader's team.

Source code

```
public int TeamId
{
    get { return teamId; }
    set { teamId = TeamId; }
}
```

See Also

Applies to: TeamLeader[35]

teamId Field

C#

```
private int teamId
```

Source code

```
private int teamId;
```

See Also

Applies to: TeamLeader[35]

User Class

User of the system.

System.Object
  PTSLibrary.User
    PTSLibrary.Customer[6]
    PTSLibrary.TeamLeader[35]

C#

```
public class User
```

Remarks

Represents any person who interacts with the PTS System. This class acts as the base class for Customer and TeamLeader.

Requirements

Namespace:PTSLibrary[6]

Assembly: PTSLibrary (in PTSLibrary.dll)

Properties

Id[39], Name[40]

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

Fields

id[39], name[40]

User.Id Property

Gets the identifier.

C#

```
public int Id {get;}
```

Property Value

The user identifier.

Remarks

Id is a readonly property.

Source code

```
public int Id
{
    get { return id; }
}
```

See Also

Applies to: User[38]

User.Name Property

Gets the name.

C#

```
public string Name {get;}
```

Property Value

The username.

Remarks

Username is a readonly property.

Source code

```
public string Name
{
    get { return name; }
}
```

See Also

Applies to: User[38]

id Field

The user identifier.

C#

```
protected int id
```

Source code

```
protected int id;
```

See Also

Applies to: User38

name Field

The username.

C#

```
protected string name
```

Source code

```
protected string name;
```

See Also

Applies to: User38

Status Enumeration

Enumeration of values that represent status.

Constant	Value	Description
ReadyToStart	1	Task is ready but not commenced.
InProgress	2	Task is being executed.
Completed	3	Task is finished.
WaitingForPredecessor	4	Task is waiting for another to be completed.

Requirements

Namespace:PTSLibrary6

Assembly: PTSLibrary (in PTSLibrary.dll)

PTSLibrary.DAO Namespace

Classes

AdminDAO41, ClientDAO49, CustomerDAO52, SuperDAO55

AdminDAO Class

An admin database access object.

System.Object
  PTSLibrary.DAO.SuperDAO55
    PTSLibrary.DAO.AdminDAO

C#

```
internal class AdminDAO : SuperDAO
```

Remarks

This is a subclass of SuperDAO that enables administrators to access the restricted database records.

Requirements

Namespace:PTSLibrary.DAO41

Assembly: PTSLibrary (in PTSLibrary.dll)

Methods

Authenticate41, CreateProject42, CreateTask44, Equals (inherited from Object), Finalize (inherited from Object), GetCustomer56 (inherited from SuperDAO), GetHashCode (inherited from Object), GetListOfCustomers48, GetListOfProjects48, GetListOfTasks57 (inherited from SuperDAO), GetListOfTeams48, GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

AdminDAO.Authenticate Method

Authenticates an administrator.

C#

```
public int Authenticate(
   string username,
   string password
)
```

Parameters

username

The username.

password

The password.

Returns

The administrator user id, or '0' if authentication fails.

Exceptions

Exception type	Condition
Exception	Thrown when an sql exception error condition occurs.

Remarks

The method checks the IsAdministrator flag in Users table.

Source code

```
 public int Authenticate(string username, string password)
 {
     string sql;
     SqlConnection cn;
     SqlCommand cmd;
     SqlDataReader dr;

     sql = String.Format("SELECT UserId FROM Person WHERE IsAdministrator = 1
AND Username='{0}' " +
                             "AND Password='{1}'", username, password);
     cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
     cmd = new SqlCommand(sql, cn);
     int id = 0;
     try
     {
         cn.Open();
         dr = cmd.ExecuteReader(CommandBehavior.SingleRow);
         if (dr.Read())
         {
             id = (int)dr["UserId"];
         }
         dr.Close();
     }
     catch (SqlException ex)
     {
         throw new Exception("Error Accessing Database", ex);
     }
     finally
     {
         cn.Close();
     }
     return id;
 }
```

See Also

Applies to: AdminDAO[41]

AdminDAO.CreateProject Method

Creates a new project and stores it in the database.

C#

```
public void CreateProject(
   string name,
   DateTime startDate,
   DateTime endDate,
   int customerId,
   int administratorId
)
```

Parameters

name

The name.

startDate

The expected start date.

endDate

The expected end date.

customerId

Identifier for the customer.

administratorId

Identifier for the project administrator.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Remarks

This method registers a new project.

Source code

```
 public void CreateProject(string name, DateTime startDate, DateTime endDate,
int customerId, int administratorId)
 {
     string sql;
     SqlConnection cn;
     SqlCommand cmd;
     Guid projectId = Guid.NewGuid();

     sql = "INSERT INTO Project (ProjectId, Name, ExpectedStartDate,
ExpectedEndDate, CustomerId, AdministratorId)";
     sql += String.Format("VALUES ( '{0}', '{1}', '{2}', '{3}', {4}, {5})",
                          projectId, name, startDate, endDate, customerId,
administratorId);
     cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
     cmd = new SqlCommand(sql, cn);

     try
     {
         cn.Open();
```

```
            cmd.ExecuteNonQuery();
        }
        catch (SqlException ex)
        {
            throw new Exception("Error Inserting", ex);
        }
        finally
        {
            cn.Close();
        }
    }
```

See Also

Applies to: AdminDAO[41]

AdminDAO.CreateTask Method

Creates a new task.

C#

```
public void CreateTask(
   string name,
   DateTime startDate,
   DateTime endDate,
   int teamId,
   Guid projectId
)
```

Parameters

name

The name of the task.

startDate

The expected start date.

endDate

The expected end date.

teamId

Identifier for the team assigned the task.

projectId

Identifier for the project.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Remarks

Registers a new task or subtask for a project.

Source code

```
 public void CreateTask(string name, DateTime startDate, DateTime endDate, int
teamId, Guid projectId)
 {
     string sql;
     SqlConnection cn;
     SqlCommand cmd;
     Guid taskId = Guid.NewGuid();
     sql = "INSERT INTO Task (TaskId, Name, ExpectedDateStarted,
ExpectedDateCompleted, ProjectId, TeamId, StatusId)";
     sql += String.Format("VALUES ( '{0}', '{1}', '{2}', '{3}', '{4}', {5},
{6})", taskId, name,
                              startDate, endDate, projectId, teamId, 1);
     cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
     cmd = new SqlCommand(sql, cn);
     try
     {
         cn.Open();
         cmd.ExecuteNonQuery();
     }
     catch (SqlException ex)
     {
         throw new Exception("Error Inserting", ex);
     }
     finally
     {
         cn.Close();
     }
 }
```

See Also

Applies to: AdminDAO[41]

AdminDAO.GetListOfCustomers Method

Gets list of customers.

C#

```
public List<Customer> GetListOfCustomers()
```

Returns

The list of all customers.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Remarks

Gets the list of customers who have commissioned projects.

Source code

```
public List<Customer> GetListOfCustomers()
{
    string sql;
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataReader dr;
    List<Customer> customers;
    customers = new List<Customer>();
    sql = "SELECT * FROM Customer";
    cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
    cmd = new SqlCommand(sql, cn);
    try
    {
        cn.Open();
        dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            Customer c = new Customer(dr["Name"].ToString(),
(int)dr["CustomerId"]);
            customers.Add(c);
        }
        dr.Close();
    }
    catch (SqlException ex)
    {
        throw new Exception("Error Getting list", ex);
    }
    finally
    {
        cn.Close();
    }
    return customers;
}
```

See Also

Applies to: AdminDAO[41]

AdminDAO.GetListOfProjects Method

Gets list of projects.

C#

```
public List<Project> GetListOfProjects(
   int adminId
)
```

Parameters

adminId

User Id for the admin.

Returns

The list of projects for an administrator.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Remarks

This method queries the database for all projects that match a given administrator.

Source code

```
public List<Project> GetListOfProjects(int adminId)
{
    string sql;
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataReader dr;
    List<Project> projects;
    projects = new List<Project>();
    sql = "SELECT * FROM Project WHERE AdministratorId = " + adminId;
    cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
    cmd = new SqlCommand(sql, cn);
    try
    {
        cn.Open();
        dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            Customer cust = GetCustomer((int)dr["CustomerId"]);
            Project p = new Project(dr["Name"].ToString(),
(DateTime)dr["ExpectedStartDate"],
           (DateTime)dr["ExpectedEndDate"], (Guid)dr["ProjectId"], cust);
            projects.Add(p);
        }
        dr.Close();
    }
    catch (SqlException ex)
    {
        throw new Exception("Error Getting list", ex);
    }
    finally
    {
        cn.Close();
    }
    return projects;
}
```

See Also

Applies to: AdminDAO41

AdminDAO.GetListOfTeams Method

Gets list of teams.

C#

```
public List<Team> GetListOfTeams()
```

Returns

The list of all teams.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Remarks

Gets all teams without filtering by any criterion.

Source code

```
public List<Team> GetListOfTeams()
{
    string sql;
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataReader dr;
    List<Team> teams;
    teams = new List<Team>();
    sql = "SELECT * FROM Team";
    cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
    cmd = new SqlCommand(sql, cn);
    try
    {
        cn.Open();
        dr = cmd.ExecuteReader();
        while (dr.Read())
        {
            Team t = new Team((int)dr["TeamId"], dr["Location"].ToString(),
           dr["Name"].ToString(), null);
            teams.Add(t);
        }
        dr.Close();
    }
    catch (SqlException ex)
    {
        throw new Exception("Error getting team list", ex);
    }
    finally
    {
        cn.Close();
    }
    return teams;
}
```

See Also

Applies to: AdminDAO[41]

ClientDAO Class

A client database access object.

System.Object
  PTSLibrary.DAO.SuperDAO[55]
    PTSLibrary.DAO.ClientDAO

C#

```
internal class ClientDAO : SuperDAO
```

Remarks

A client refers to a team leader (Not a Customer). This DAO allows a team leader to manage projects assigned to their team.

Requirements

Namespace:PTSLibrary.DAO[41]

Assembly: PTSLibrary (in PTSLibrary.dll)

Methods

Authenticate[49], Equals (inherited from Object), Finalize (inherited from Object), GetCustomer[56] (inherited from SuperDAO), GetHashCode (inherited from Object), GetListOfProjects[50], GetListOfTasks[57] (inherited from SuperDAO), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

ClientDAO.Authenticate Method

Authenticates the team leader.

C#

```
public TeamLeader Authenticate(
   string username,
   string password
)
```

Parameters

username

The username.

password

The password.

Returns

A TeamLeader.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Remarks

This method authenticates users only if they are team leaders.

Source code

```
 public TeamLeader Authenticate(string username, string password)
 {
     string sql;
     SqlConnection cn;
     SqlCommand cmd;
     SqlDataReader dr;
     TeamLeader teamLeader = null;
     sql = String.Format("SELECT DISTINCT Person.Name, UserId, TeamId FROM
Person " +
                         "INNER JOIN Team ON (Team.TeamLeaderId = Person.UserId)
" +
                         "WHERE  Username='{0}' AND Password='{1}'", username,
password);
     cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
     cmd = new SqlCommand(sql, cn);
     try
     {
         cn.Open();
         dr = cmd.ExecuteReader(CommandBehavior.SingleRow);
         if (dr.Read())
         {
             teamLeader = new TeamLeader(dr["Name"].ToString(),
(int)dr["UserId"], (int)dr["TeamId"]);
         }
         dr.Close();
     }
     catch (SqlException ex)
     {
         throw new Exception("Error Accessing Database", ex);
     }
     finally
     {
         cn.Close();
     }
     return teamLeader;
 }
```

See Also

Applies to: ClientDAO49

ClientDAO.GetListOfProjects Method

Gets list of projects.

C#

```
public List<Project> GetListOfProjects(
   int teamId
)
```

Parameters

teamId

Identifier for the team.

Returns

The list of projects.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Remarks

Gets the list of projects for a particular team.

Source code

```
 public List<Project> GetListOfProjects(int teamId)
 {
     string sql;
     SqlConnection cn;
     SqlCommand cmd;
     SqlDataReader dr;
     List<Project> projects;
     projects = new List<Project>();
     sql = "SELECT P.* FROM Project AS P INNER JOIN Task AS T ON (P.ProjectId =
T.ProjectId) WHERE T.TeamId = " + teamId;
     cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
     cmd = new SqlCommand(sql, cn);
     try
     {
         cn.Open();
         dr = cmd.ExecuteReader();
         while (dr.Read())
         {
             Customer cust = GetCustomer((int)dr["CustomerId"]);
             Project p = new Project(dr["Name"].ToString(),
(DateTime)dr["ExpectedStartDate"],
            (DateTime)dr["ExpectedEndDate"], (Guid)dr["ProjectId"], cust);
             projects.Add(p);
         }
         dr.Close();
     }
     catch (SqlException ex)
     {
         throw new Exception("Error Getting list", ex);
     }
```

```
    finally
    {
        cn.Close();
    }
    return projects;
}
```

See Also

Applies to: ClientDAO[49]

CustomerDAO Class

A customer database access object.

System.Object
  PTSLibrary.DAO.SuperDAO[55]
    PTSLibrary.DAO.CustomerDAO

C#

```
internal class CustomerDAO : SuperDAO
```

Remarks

Provides an interface through which a customer can access the database.

Requirements

Namespace:PTSLibrary.DAO[41]

Assembly: PTSLibrary (in PTSLibrary.dll)

Methods

Authenticate[52], Equals (inherited from Object), Finalize (inherited from Object), GetCustomer[56] (inherited from SuperDAO), GetHashCode (inherited from Object), GetListOfProjects[54], GetListOfTasks[57] (inherited from SuperDAO), GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

CustomerDAO.Authenticate Method

Authenticates a customer.

C#

```
public int Authenticate(
   string username,
   string password
)
```

Parameters

username

The username.

password

The customer password.

Returns

The customer id but when authentication fails, returns '0'.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Source code

```
 public int Authenticate(string username, string password)
 {
     string sql;
     SqlConnection cn;
     SqlCommand cmd;
     SqlDataReader dr;

     sql = String.Format("SELECT CustomerId FROM Customer WHERE Username='{0}' "
+
                              "AND Password='{1}'", username, password);

     cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
     cmd = new SqlCommand(sql, cn);
     int id = 0;
     try
     {
         cn.Open();
         dr = cmd.ExecuteReader(CommandBehavior.SingleRow);
         if (dr.Read())
         {
             id = (int)dr["CustomerId"];
         }
         dr.Close();
     }
     catch (SqlException ex)
     {
         throw new Exception("Error Accessing Database", ex);
     }
     finally
     {
         cn.Close();
     }
     return id;
 }
```

See Also

Applies to: CustomerDAO[52]

CustomerDAO.GetListOfProjects Method

Gets list of projects.

C#

```
public List<Project> GetListOfProjects(
   int customerId
)
```

Parameters

customerId

Identifier for the customer.

Returns

The list of projects.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs.

Remarks

This method queries the database for projects that belong to a customer.

Source code

```
 public List<Project> GetListOfProjects(int customerId)
 {

     string sql;
     SqlConnection cn;
     SqlCommand cmd;
     SqlDataReader dr;
     List<Project> projects;
     projects = new List<Project>();

     sql = "SELECT * FROM Project WHERE CustomerId = " + customerId.ToString();
     cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
     cmd = new SqlCommand(sql, cn);

     try
     {
         cn.Open();
         dr = cmd.ExecuteReader();
         SqlConnection cn2; SqlCommand cmd2; SqlDataReader dr2;// string sql2;
//custom
         while (dr.Read())
         {
             List<Task> tasks = new List<Task>();
             sql = "SELECT * FROM Task WHERE ProjectId = '" +
dr["ProjectId"].ToString() + "'";
             //sql2 = "SELECT * FROM Task WHERE ProjectId = '" +
dr["ProjectId"].ToString() + "'";
```

```
            cn2 = new
SqlConnection(Properties.Settings.Default.ConnectionString);
            cmd2 = new SqlCommand(sql, cn2);
            cn2.Open();
            dr2 = cmd2.ExecuteReader();
            while (dr2.Read())
            {
                Task t = new Task((Guid)dr2["TaskId"], dr2["Name"].ToString(),
                                  (Status)dr2["StatusId"]);
                tasks.Add(t);
            }
            dr2.Close();
            Project p = new Project(dr["Name"].ToString(),
(DateTime)dr["ExpectedStartDate"],
                                    (DateTime)dr["ExpectedEndDate"],
(Guid)dr["ProjectId"], tasks);
            projects.Add(p);
            cn2.Close();
        }
        dr.Close();
    }
    catch (SqlException ex)
    {
        throw new Exception("Error Getting list", ex);
    }
    finally
    {
        cn.Close();
    }
    return projects;
 }
```

See Also

Applies to: CustomerDAO[52]

SuperDAO Class

The super DAO.

System.Object
  PTSLibrary.DAO.SuperDAO
    PTSLibrary.DAO.AdminDAO[41]
    PTSLibrary.DAO.ClientDAO[49]
    PTSLibrary.DAO.CustomerDAO[52]

C#

```
public class SuperDAO
```

Remarks

Acts as a base class for data access objects. DAO's provide an interface through which applications access the database indirectly.

Requirements

Namespace:PTSLibrary.DAO[41]

Assembly: PTSLibrary (in PTSLibrary.dll)

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetCustomer[56], GetHashCode (inherited from Object), GetListOfTasks[57], GetType (inherited from Object), MemberwiseClone (inherited from Object), ToString (inherited from Object)

SuperDAO.GetCustomer Method

Gets the customer.

C#

```
protected Customer GetCustomer(
   int custId
)
```

Parameters

custId

Identifier for the customer.

Returns

The customer.

Exceptions

Exception type	Condition
Exception	Thrown when an exception error condition occurs during sql query.

Remarks

This method queries the database for customer details using a given Customer (User) Id.

Source code

```
protected Customer GetCustomer(int custId)
{
    string sql;
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataReader dr;
    Customer cust;

    sql = "SELECT * FROM Customer WHERE CustomerId = " + custId;
    cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
    cmd = new SqlCommand(sql, cn);
    try
    {
        cn.Open();
```

```
            dr = cmd.ExecuteReader(CommandBehavior.SingleRow);
            dr.Read();
            cust = new Customer(dr["Name"].ToString(), (int)dr["CustomerId"]);
            dr.Close();
        }
        catch (SqlException ex)
        {
            throw new Exception("Error Getting Customer", ex);
        }
        finally
        {
            cn.Close();
        }
        return cust;
    }
```

See Also

Applies to: SuperDAO[55]

SuperDAO.GetListOfTasks Method

Gets list of tasks for a given project.

C#

```
public List<Task> GetListOfTasks(
   Guid projectId
)
```

Parameters

projectId

Identifier for the project.

Returns

The list of tasks.

Exceptions

Exception type	Condition
Exception	Thrown when an sql exception error condition occurs.

Remarks

Gets the tasks saved in the database for one project.

Source code

```
public List<Task> GetListOfTasks(Guid projectId)
{
    string sql;
```

```
    SqlConnection cn;
    SqlCommand cmd;
    SqlDataReader dr;
    List<Task> tasks;
    tasks = new List<Task>();

    sql = "SELECT * FROM Task WHERE ProjectId = '" + projectId + "'";
    cn = new SqlConnection(Properties.Settings.Default.ConnectionString);
    cmd = new SqlCommand(sql, cn);

    try
    {
        cn.Open();
        dr = cmd.ExecuteReader();
        while(dr.Read())
        {
            Task t = new Task((Guid)dr["TaskId"], dr["Name"].ToString(),
(Status)((int)dr["StatusId"]));
            tasks.Add(t);
        }
        dr.Close();
    }
    catch (SqlException ex)
    {
        throw new Exception("Error getting tasks list", ex);
    }
    finally
    {
        cn.Close();
    }
    return tasks;
 }
```

See Also

Applies to: SuperDAO[55]

PTSLibrary.Properties Namespace

Classes

Settings[59]

Settings Class

System.Object
  System.Configuration.SettingsBase
    System.Configuration.ApplicationSettingsBase
      PTSLibrary.Properties.Settings

C#

```
[global::System.Runtime.CompilerServices.CompilerGenerated()]
[global::System.CodeDom.Compiler.GeneratedCode("Microsoft.VisualStudio.Editors.S
ettingsDesigner.SettingsSingleFileGenerator", "15.7.0.0")]
internal sealed class Settings : ApplicationSettingsBase
```

Requirements

Namespace:PTSLibrary.Properties[59]

Assembly: PTSLibrary (in PTSLibrary.dll)

Properties

ConnectionString[60], Context (inherited from ApplicationSettingsBase), Default[61], IsSynchronized (inherited from SettingsBase), Item (inherited from ApplicationSettingsBase), Properties (inherited from ApplicationSettingsBase), PropertyValues (inherited from ApplicationSettingsBase), Providers (inherited from ApplicationSettingsBase), SettingsKey (inherited from ApplicationSettingsBase)

Methods

Equals (inherited from Object), Finalize (inherited from Object), GetHashCode (inherited from Object), GetPreviousVersion (inherited from ApplicationSettingsBase), GetType (inherited from Object), Initialize (inherited from SettingsBase), MemberwiseClone (inherited from Object), OnPropertyChanged (inherited from ApplicationSettingsBase), OnSettingChanging (inherited from ApplicationSettingsBase), OnSettingsLoaded (inherited from ApplicationSettingsBase), OnSettingsSaving (inherited from ApplicationSettingsBase), Reload (inherited from ApplicationSettingsBase), Reset (inherited from ApplicationSettingsBase), Save (inherited from ApplicationSettingsBase), ToString (inherited from Object), Upgrade (inherited from ApplicationSettingsBase)

Events

PropertyChanged (inherited from ApplicationSettingsBase), SettingChanging (inherited from ApplicationSettingsBase), SettingsLoaded (inherited from ApplicationSettingsBase), SettingsSaving (inherited from ApplicationSettingsBase)

Fields

defaultInstance[61]

Settings.ConnectionString Property

C#

```
[global::System.Configuration.ApplicationScopedSetting()]
[global::System.Diagnostics.DebuggerNonUserCode()]
[global::System.Configuration.SpecialSetting(global::System.Configuration.Specia
lSetting.ConnectionString)]
[global::System.Configuration.DefaultSettingValue("Data Source=XPLICIT;Initial
Catalog=wm75;Integrated Security=True")]
public string ConnectionString {get;}
```

Source code

```
 [global::System.Configuration.ApplicationScopedSettingAttribute()]
 [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Configuration.SpecialSettingAttribute(global::System.Configurati
on.SpecialSetting.ConnectionString)]
 [global::System.Configuration.DefaultSettingValueAttribute("Data
Source=XPLICIT;Initial Catalog=wm75;Integrated Security=True")]


 public string ConnectionString {
     get {
         return ((string)(this["ConnectionString"]));
     }
 }
```

See Also

Applies to: Settings[59]

Settings.Default Property

C#

```
public static Settings Default {get;}
```

Source code

```
 public static Settings Default {
     get {
         return defaultInstance;
     }
 }
```

See Also

Applies to: Settings[59]

defaultInstance Field

C#

```
new private static Settings defaultInstance
```

Source code

```
 private static Settings defaultInstance =
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(ne
w Settings())));
```

See Also

Applies to: Settings[59]

Index

tasks Field 26
teamId Field 37
theCustomer Field 26