ImageShop BusinessLogic-Documentation für PL-Dev

*[dieses Dokument gilt gleichzeitig als **Contract** zwischen BO-Dev und PL-Dev]*

Bevor es los geht:

1. Im Folgenden steht [MA] steht ManagementAccess – ManagementAccess stellt ein Objekt dar, dass beim Inititalisieren ein User Objekt als Parameter (im Konstruktor) erwartet. Nur wenn der User über ein bestmmtes AccessLevel verfügt, kann ein ManagementAccess Objekt erzeugt werden. Das ManagementAccess Objekt seinerseits stellt diverse Funktionen zur Verfügung, die Objekte sämtlicher Typen (ISUser, ISFolder, ISProductType, ISOrder, ISImage) zurückgeben kann. Funktionen und SETs für Properties, die im folgenden Text mit [MA] gekennzeichnet sind, sind nur auf Objekte anwendbar, die über ManagementAccess abgefragt wurden. Dies soll als Sicherheitsmaßnahme dienen.

2. Objekte die in zweiter Stufe über solche durch ManagementAccess erzeugten Objekte angesprochen werden erben den ManagementAccess.
Beispiel:
BasicFunctions.login(User,Password); gibt einen normalen ISUser zurück.
Dieser ISUser kann keinesfalls die Funktion User.Folders[0].delete(); aufrufen (es wird eine Exception geworfen!)
Auch kann dieser ISUser nicht User.Folders[0].Images[0].delete(); aufrufen. Wenn der User allerdings ein ManagementAccess Objekt erzeugen kann, können diese beiden Funktionen über dieses ManagementAccess Objekt aufgerufen werden:
ManagementAccess(User).getUserById(10).Folders[0].delete();
Genauso: ManagementAccess(User).getUserById(10).Folders[0].Images[0].delete();

3. Es wird dringend empfohlen alle Operationen (Methodenaufrufe, abrufen und setzen von properties) mittels try/catch auszuführen, da intern sehr komplexe Berechtigungs- und Integritätsüberprüfungen stattfinden, um Fehlerhafte und Unerlaubte Vorgänge im System zu vermeiden. Mittels Exception.Message können die jeweiligen Fehlermeldungen beim catchen abgerufen werden, die schildern was schief gelaufen ist und auch dem User ausgegeben werden können.

Es werden nun die einzelnen Objekttypen (Klassen) und deren Eigenschaften/Methoden geschildert, wie sie für den PL Programmierer sichtbar/wichtig sind. An dieser Stelle wird es emfohlen sich bereits vorab mittels der Dokumente „ObjektHierachie“ und „ManagementAccess“ einen visuellen Eindruck über die hierarchische Ordnung der einzelnen Objekttypen (Klassen) zu verschaffen.

Die Eigenschaft _ID

Diese Eigenschaft wird von der Klasse BaseObject an die Objekttypen ISUser, ISFolder, ISProductType, ISResolution und ISImage vererbt und hilft diese eindeutig identifizieren zu können.

String _ID (get)

Class/Object ISUser

Eigenschaften die mit dem jeweiligen DatenBank Objekt verknüpft sind (immer mit „_“ voran):

String _Username (set[MA]/get)
Int _PermissionLevel (set[MA]/get)
String _FirstName (set[MA]/get)
String _LastName (set[MA]/get)
String _Email (set[MA]/get)
String _Password (set[MA]/get)
String _ID (get)

Access auf zugeordnete/übergeordnete/untergordnete Objekte:

ReadOnlyCollection<ISOrder> ISUser.Orders *[z.B. alle Orders eines Users auflisten)*
ReadOnlyCollection<ISFolder> ISUser.Folders *[z.B. alle Folder eines Users auflisten]*
ISFolder ISUser.Folder(id) *[z.B. um alle Images eines ganz bestimmten Ordners zu bekommen, auf den der User geklickt hat und diese dann anzuzeigen]*

Methoden:

bool delete()[MA], **bool** addFolder(ISFolder)[MA], **bool** removeFolder(ISFolder)[MA], **Order** addOrder(ISImage, ISProducttype)

Class/Object ISFolder

Eigenschaften die mit dem jeweiligen DatenBank Objekt verknüpft sind (immer mit „_“ voran):

String _Foldername (set[MA]/get)
Int _FolderType (set[MA]/get)

Access auf zugeordnete/übergeordnete/untergeordnete Objekte:

ReadOnlyCollection<ISImage> ISFolder.Images
ISImage ISFolder.Image(id) *[z.B. auch um zu prüfen ob ein ganz bestimmte Image im Ordner überhaupt vorhanden ist. Sonst → Exception]*
ReadOnlyCollection<ISResolution> ISFolder.Resolutions
ISResolution ISFolder.Resolution(id) *[z.B. um zu prüfen ob eine ganz bestimmte Resolution im Ordner überhaupt vorhanden ist. Sonst → Exception]*
ReadOnlyCollection<ISProductType> ISFolder.ProductTypes
ISProductType ISFolder.ProductType(id) *[z.B. um zu prüfen ob ein ganz bestimmter ProductType im Ordner überhaupt vorhanden ist. Sonst → Exception]*
ReadOnlyCollection<ISUser> ISFolder.Users
ISUser ISFolder.User(id) *[z.B. um zu prüfen ob ein ganz bestimmter ProductType im Ordner überhaupt vorhanden ist. Sonst → Exception]*

Methoden:
bool delete()[MA], **bool** addImage(System.IO.Stream)[MA], **bool** addUser(ISUser)[MA], **bool** removeUser(ISUser)[MA], **String** getZIPDownloadPath(ISResolution), **bool** addProductType(ISProductType)[MA], **bool** removeProductType(ISProductTyoe)[MA] **bool** addResolution(ISResolution)[MA], **bool** removeResolution(ISResolution)[MA]

Class/Object ISImage

Eigenschaften die mit dem jeweiligen DatenBank Objekt verknüpft sind (immer mit „_“ voran):

String _Name (set[MA]/get)

Access auf zugeordnete/übergeordnete/untergeordnete Objekte:

ISFolder ISImage.Folder

Methoden:
bool deleteImage()[MA], **Image** getImageInResolution(ISResolution)

Class/Object ISProductType

Eigenschaften die mit dem jeweiligen DatenBank Objekt verknüpft sind (immer mit „_“ voran):

String _Name (set[MA]/get)
Float _Price (set[MA]/get)

Access auf zugeordnete/übergeordnete/untergeordnete Objekte:

List <ISFolder> ISProductType.Folders
ISFolder ISProductType.Folder(id)

Methoden:
bool deleteProductType()[MA]

Class/Object ISResolution

Eigenschaften die mit dem jeweiligen DatenBank Objekt verknüpft sind (immer mit „_“ voran):

int _Width (set[MA]/get)
int _Height(set[MA]/get)

Access auf zugeordnete/übergeordnete/untergeordnete Objekte:

List <ISFolder> ISResolution.Folders
ISFolder ISResolution.Folder(id)

Methoden:

bool deleteResolution()[MA]

Class/Object ISOrder

Ein **ISOrder** Objekt besteht immer aus einem **ISImage** und einem **ISProducttype**. Eine Liste aller Order-Objekte kann vom User mittels **ReadOnlyCollection<Order>** ISUser.Orders ausgelesen werden. Die Liste **ReadOnlyCollection<Order>** ISUser.Orders kommt daher einem Warenkorb gleich, ein einzelnes **ISOrder** Objekt kommt einem Produkt im Warenkorb gleich. **ReadOnlyCollection<Order>** ISUser.Orders Listen immer rein User und Session bezogen und werden NICHT in der Datenbank gespeichert. Beim Login eines Users wird die Liste **ReadOnlyCollection<Order>** ISUser.Orders am **ISUser** erzeugt. Mit der Methode

Order ISUser.addOrder(Image, Producttype);

kann ein neues Orderobjekt zu dieser Liste hinzugefügt werden. Einzelne Order Objekte in dieser Liste können mittels Index (User.Orders[index]) angesprochen werden.

Eigenschaften:

ISProductType ProductType(get)
ISImage Image(get)

Methoden:

bool removeOrder(), **void** setCount(int count), **int** getCount()

Class/Object ISManagementAccess

Eine Erklärung zum Objekt **ISManagementAccess** ist am Beginn des Dokuments zu finden
Ein Objekt dieser Klasse muss mit einem Konstruktor erzeugt werden:
new ISManagementAccess (ISUser);
Der übergebene User muss die Berechtigung haben, ein ManagementAccess Operationen (in diesem Dokument mit [MA] gekennzeichnet), sonst wird das Objekt nicht erzeugt sondern eine Exception geworfen.

Eigenschaften:

ISProductType ProductType (get)
ISImage ProductType (get)

Access auf verwaltbare Objekte:

ReadOnlyCollection<ISUser> ISManagementAccess.getUserByID(id);
ISUser ISManagementAccess.getAllUsers;
ReadOnlyCollection<ISFolder> ISManagementAccess.getFolderByID(id);
I**SFolder** ISManagementAccess.getAllFolders;

ReadOnlyCollection<ISProductType> ISManagementAccess.getProductTypeByID(id);
ISProductType ISManagementAccess.getAllProductTypes;
ReadOnlyCollection<ISResolution> ISManagementAccess.getProductTypeByID(id);
ISResolution ISManagementAccess.getAllResolutions;
ISFolder ISManagementAccess.createFolder(String FolderName, int FolderType); *[für int Foldertype gilt: 1 macht einen normalen Folder, jeder andere Wert erzeugt einen public Folder]*
ISProductType ISManagementAccess.createProductype(String ProductTypeName);
ISResolution ISManagementAccess.createResolution(String ResolutionName);
ISUser ISManagementAccess.createUser(String Username, String EMail, String FirstName, String LastName, String Password, optional int PermissionLevel=0);