dataManagement

Class MySqlDatabase

java.lang.Object
dataManagement.MySqlDatabase

All Implemented Interfaces:

IDataBase

```
public class MySqlDatabase
extends java.lang.Object
implements IDataBase
```

Dies ist die konkrete Implementierung der Schnittstelle zur Datenbank. Dazu implementiert diese Klasse das Interface IDataBase, welches eine abstrakte Definition der Schnittstelle von Java zu einer beliebigen Datenbank vorgibt. Bei der konkret in dieser Klasse verwendeten Datenbank handelt es sich um MySql

Constructor Summary

Constructors

Constructor and Description
MySqlDatabase(java.lang.String url, java.lang.String name, java.lang.String pw) Der Konstruktor lädt den jdbc Treiber und richtet danach eine Datenbankverbindung mittels der übergebenen Daten ein, indem er das Attribut con instanziiert.

Method Summary

Methods

Modifier and Type	Method and Description
void	**createBook**(java.lang.String bookName) Diese Methode legt ein neues Buch in der Tabelle "Books" an
void	**createChapter**(java.lang.String chapterName, int bookID) Diese Methode legt ein neues Kapitel in der Tabelle "Chapter" an
void	**createContent**(byte[] contentBytes, int contentNumber, int pageID) Diese Methode legt einen neuen Seiteninhalt in der Tabelle "Content" an
void	**createPage**(**PageInformation** pageInfo, int chapterID) Diese Methode legt eine neue Seite in der Tabelle "Pages" an

void	**deleteBook**(int bookID) Diese Methode löscht ein Buch aus der Tabelle "Books"
void	**deleteChapter**(int chapterID) Diese Methode löscht ein Kapitel aus der Tabelle "Chapter"
void	**deleteContent**(int contentID) Diese Methode löscht einen Seiteninhalt aus der Tabelle "Content"
void	**deletePage**(int pageID) Diese Methode löscht eine Seite aus der Tabelle "Pages"
boolean	**existsBook**(java.lang.String bookName) Diese Methode gibt zurück, ob es in der Tabelle "Books" ein Buch mit dem Namen "bookName" gibt
boolean	**existsChapter**(java.lang.String chapterName, int bookID) Diese Methode gibt zurück, ob es in der Tabelle "Chapter" ein Kapitel mit dem Namen "chapterName" gibt
boolean	**existsContent**(int contentNumber, int pageID) Diese Methode gibt zurück, ob es in der Tabelle "Content" ein Eintrag mit der Nummer "contentNumber" gibt
boolean	**existsPage**(java.lang.String pageName, int chapterID) Diese Methode gibt zurück, ob es in der Tabelle "Pages" eine Seite mit dem Namen "pageName" gibt
java.lang.String[]	**getAllBooks**() Diese Methode liefert den Namen aller Bücher welche in der Datenbank angelegt sind
java.lang.String[]	**getBookChildren**(int bookID) Diese Methode liefert zur id eines Buches alle Kapitel welche zu diesem Buch gehören
int	**getBookID**(java.lang.String bookName) Diese Methode liefert zu dem Namen eines Buches die entsprechende id in der Datenbank
java.lang.String[]	**getChapterChildren**(int chapterID) Diese Methode liefert zur id eines Kapitels alle Seiten welche zu diesem Kapitel gehören
int	**getChapterID**(java.lang.String chapterName, int bookID) Diese Methode liefert zu dem Namen eines Kapitels die entsprechende id in der Datenbank
int	**getContentID**(int ContentNumber, int pageID) Diese Methode liefert zu der Nummer eines Seiteinhalts die entsprechende id in der Datenbank
int[]	**getPageChildren**(int pageID) Diese Methode liefert zur id einer Seite alle Inhalte welche zu dieser Seite gehören
int	**getPageID**(java.lang.String pageName, int chapterID) Diese Methode liefert zu dem Namen einer Seite die entsprechende id in der Datenbank
byte[]	**loadContent**(int contentID) Diese Methode läd einen Seiteninhalt aus der Tabelle "Content"
PageInformation	**loadPageInformation**(int pageID) Diese Methode lädt die Seiteninformation aus der Tabelle "Pages"
void	**saveContent**(byte[] contentBytes, int contentID) Diese Methode speichert einen Seiteninhalt in der Tabelle "Content"

void	**savePage**(**PageInformation** pageInfo, int chapterID, int pageID) Diese Methode speichert eine Seite in der Tabelle "Pages"

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MySqlDatabase

```
public MySqlDatabase(java.lang.String url,
             java.lang.String name,
             java.lang.String pw)
              throws java.lang.ClassNotFoundException,
                     java.sql.SQLException
```

Der Konstruktor lädt den jdbc Treiber und richtet danach eine Datenbankverbindung mittels der übergebenen Daten ein, indem er das Attribut con instanziiert.

Parameters:

`url` - der Pfad zur Datenbank

`name` - der Loginname zur Datenbank

`pw` - das Passwort zur Datenbank

Throws:

`java.lang.ClassNotFoundException` - Fehler beim Laden des jdbc Treibers

`java.sql.SQLException` - Datenbankfehler

Method Detail

createBook

```
public void createBook(java.lang.String bookName)
                throws java.sql.SQLException
```

Diese Methode legt ein neues Buch in der Tabelle "Books" an

Specified by:

`createBook` in interface `IDataBase`

Parameters:

`bookName` - der Name des anzulegenden Buches

Throws:

`java.sql.SQLException` - Datenbankfehler

createChapter

```
public void createChapter(java.lang.String chapterName,
                  int bookID)
                    throws java.sql.SQLException
```

Diese Methode legt ein neues Kapitel in der Tabelle "Chapter" an

Specified by:

`createChapter` in interface `IDataBase`

Parameters:

`chapterName` - der Name des anzulegenden Kapitels

`bookID` - die id des Buches zu dem das anzulegende Kapitel gehören soll

Throws:

`java.sql.SQLException` - Datenbankfehler

createPage

```
public void createPage(PageInformation pageInfo,
               int chapterID)
                 throws java.sql.SQLException
```

Diese Methode legt eine neue Seite in der Tabelle "Pages" an

Specified by:

`createPage` in interface `IDataBase`

Parameters:

`pageInfo` - informationen zur Seite

`chapterID` - die id des Kapitels zu dem die anzulegende Seite gehören soll

Throws:

`java.sql.SQLException` - Datenbankfehler

createContent

```
public void createContent(byte[] contentBytes,
                  int contentNumber,
                  int pageID)
                    throws java.sql.SQLException
```

Diese Methode legt einen neuen Seiteninhalt in der Tabelle "Content" an

Specified by:

`createContent` in interface `IDataBase`

Parameters:

`contentBytes` - das serialisierte Objekt

`contentNumber` - die nummer des Seiteninhalts innerhalb seiner Seite

`pageID` - die id der Seite zu welcher der Inhalt gehören soll

Throws:

`java.sql.SQLException` - Datenbankfehler

deleteBook

```
public void deleteBook(int bookID)
                throws java.sql.SQLException
```

Diese Methode löscht ein Buch aus der Tabelle "Books"

Specified by:

`deleteBook` in interface `IDataBase`

Parameters:

`bookID` - Die id des Buches, welches gelöscht werden soll

Throws:

`java.sql.SQLException` - Datenbankfehler

deleteChapter

```
public void deleteChapter(int chapterID)
                  throws java.sql.SQLException
```

Diese Methode löscht ein Kapitel aus der Tabelle "Chapter"

Specified by:

`deleteChapter` in interface `IDataBase`

Parameters:

`chapterID` - Die id des Kapitels, welches gelöscht werden soll

Throws:

`java.sql.SQLException` - Datenbankfehler

deletePage

```
public void deletePage(int pageID)
```

```
                throws java.sql.SQLException
```

Diese Methode löscht eine Seite aus der Tabelle "Pages"

Specified by:

`deletePage` in interface `IDataBase`

Parameters:

`pageID` - Die id der Seite, welche gelöscht werden soll

Throws:

`java.sql.SQLException` - Datenbankfehler

deleteContent

```
public void deleteContent(int contentID)
                   throws java.sql.SQLException
```

Diese Methode löscht einen Seiteninhalt aus der Tabelle "Content"

Specified by:

`deleteContent` in interface `IDataBase`

Parameters:

`contentID` - Die id des Seiteninhaltes, welcher gelöscht werden soll

Throws:

`java.sql.SQLException` - Datenbankfehler

saveContent

```
public void saveContent(byte[] contentBytes,
                int contentID)
                  throws java.sql.SQLException
```

Diese Methode speichert einen Seiteninhalt in der Tabelle "Content"

Specified by:

`saveContent` in interface `IDataBase`

Parameters:

`contentBytes` - das serialisierte Objekt

`contentID` - die innerhalb der Tabelle "Content"

Throws:

`java.sql.SQLException` - Datenbankfehler

savePage

```
public void savePage(PageInformation pageInfo,
            int chapterID,
            int pageID)
              throws java.sql.SQLException
```

Diese Methode speichert eine Seite in der Tabelle "Pages"

Specified by:

`savePage` in interface `IDataBase`

Parameters:

`pageInfo` - die Information zur Seite

`chapterID` - das Kapitel zu welchem die Seite gehört

`pageID` - die id der Seite innerhalb der Tabelle "Pages"

Throws:

`java.sql.SQLException` - Datenbankfehler

loadContent

```
public byte[] loadContent(int contentID)
                  throws java.sql.SQLException
```

Diese Methode läd einen Seiteninhalt aus der Tabelle "Content"

Specified by:

`loadContent` in interface `IDataBase`

Parameters:

`contentID` - die id des angefragten Seiteninhalts

Returns:

das serialisierte Objekt zur angefragten id

Throws:

`java.sql.SQLException` - Datenbankfehler

loadPageInformation

```
public PageInformation loadPageInformation(int pageID)
                                    throws java.sql.SQLException
```

Diese Methode lädt die Seiteninformation aus der Tabelle "Pages"

Specified by:

`loadPageInformation` in interface `IDataBase`

Parameters:

`pageID` - die id der angefragten Seite

Returns:

die Seiteninformation zur angefragten id

Throws:

`java.sql.SQLException` - Datenbankfehler

existsBook

```
public boolean existsBook(java.lang.String bookName)
                   throws java.sql.SQLException
```

Diese Methode gibt zurück, ob es in der Tabelle "Books" ein Buch mit dem Namen "bookName" gibt

Specified by:

`existsBook` in interface `IDataBase`

Parameters:

`bookName` - der Name des Buchs

Returns:

zum angefragten Buch existiert ein Eintrag in der Datenbank

Throws:

`java.sql.SQLException` - Datenbankfehler

existsChapter

```
public boolean existsChapter(java.lang.String chapterName,
                     int bookID)
                       throws java.sql.SQLException
```

Diese Methode gibt zurück, ob es in der Tabelle "Chapter" ein Kapitel mit dem Namen "chapterName" gibt

Specified by:

`existsChapter` in interface `IDataBase`

Parameters:

`chapterName` - der Name des Kapitels

`bookID` - die id des Buches zu welchem das Kapitel gehört

Returns:

zum angefragten Kapitel existiert ein Eintrag in der Datenbank

Throws:

`java.sql.SQLException` - Datenbankfehler

existsPage

```
public boolean existsPage(java.lang.String pageName,
                  int chapterID)
                    throws java.sql.SQLException
```

Diese Methode gibt zurück, ob es in der Tabelle "Pages" eine Seite mit dem Namen "pageName" gibt

Specified by:

`existsPage` in interface `IDataBase`

Parameters:

`pageName` - der Name der Seite

`chapterID` - die id des Kapitels zu welchem die Seite gehört

Returns:

zur angefragten Seite existiert ein Eintrag in der Datenbank

Throws:

`java.sql.SQLException` - Datenbankfehler

existsContent

```
public boolean existsContent(int contentNumber,
                     int pageID)
                       throws java.sql.SQLException
```

Diese Methode gibt zurück, ob es in der Tabelle "Content" ein Eintrag mit der Nummer "contentNumber" gibt

Specified by:

`existsContent` in interface `IDataBase`

Parameters:

`contentNumber` - die Nummer des Seiteninhalts

`pageID` - die id der Seite zu welcher der Inhalt gehört

Returns:

zum angefragten Seiteninhalt existiert ein Eintrag in der Datenbank

Throws:

`java.sql.SQLException` - Datenbankfehler

getBookID

```
public int getBookID(java.lang.String bookName)
              throws java.sql.SQLException
```

Diese Methode liefert zu dem Namen eines Buches die entsprechende id in der Datenbank

Specified by:

`getBookID` in interface `IDataBase`

Parameters:

`bookName` - Der Name des angefragten Buches

Returns:

die id des angefragten Buches

Throws:

`java.sql.SQLException` - Datenbankfehler

getChapterID

```
public int getChapterID(java.lang.String chapterName,
                int bookID)
                  throws java.sql.SQLException
```

Diese Methode liefert zu dem Namen eines Kapitels die entsprechende id in der Datenbank

Specified by:

`getChapterID` in interface `IDataBase`

Parameters:

`chapterName` - Der Name des angefragten Kapitels

`bookID` - die id des Buches zu welchem das Kapitel gehört

Returns:

die id des angefragten Kapitels

Throws:

`java.sql.SQLException` - Datenbankfehler

getPageID

```
public int getPageID(java.lang.String pageName,
            int chapterID)
              throws java.sql.SQLException
```

Diese Methode liefert zu dem Namen einer Seite die entsprechende id in der Datenbank

Specified by:

`getPageID` in interface `IDataBase`

Parameters:

`pageName` - Der Name der angefragten Seite

`chapterID` - die id des Kapitels zu welchem die Seite gehört

Returns:

die id der angefragten Seite

Throws:

`java.sql.SQLException` - Datenbankfehler

getContentID

```
public int getContentID(int ContentNumber,
               int pageID)
                 throws java.sql.SQLException
```

Diese Methode liefert zu der Nummer eines Seiteinhalts die entsprechende id in der Datenbank

Specified by:

`getContentID` in interface `IDataBase`

Parameters:

`ContentNumber` - Die Nummer des angefragten Seiteninhalts

`pageID` - die id der Seite zu welchem dir Inhalt gehört

Returns:

die id ders angefragten Seiteinhalts

Throws:

`java.sql.SQLException` - Datenbankfehler

getBookChildren

```
public java.lang.String[] getBookChildren(int bookID)
                                   throws java.sql.SQLException
```

Diese Methode liefert zur id eines Buches alle Kapitel welche zu diesem Buch gehören

Specified by:

`getBookChildren` in interface `IDataBase`

Parameters:

`bookID` - Die id des angefragten Buches

Returns:

der Name aller Kapitel, welche zum angefragten Buch gehören

Throws:

`java.sql.SQLException` - Datenbankfehler

getChapterChildren

```
public java.lang.String[] getChapterChildren(int chapterID)
                                      throws java.sql.SQLException
```

Diese Methode liefert zur id eines Kapitels alle Seiten welche zu diesem Kapitel gehören

Specified by:

`getChapterChildren` in interface `IDataBase`

Parameters:

`chapterID` - Die id des angefragten Kapitels

Returns:

der Name aller Seiten, welche zum angefragten Kapitel gehören

Throws:

`java.sql.SQLException` - Datenbankfehler

getPageChildren

```
public int[] getPageChildren(int pageID)
                      throws java.sql.SQLException
```

Diese Methode liefert zur id einer Seite alle Inhalte welche zu dieser Seite gehören

Specified by:

`getPageChildren` in interface `IDataBase`

Parameters:

`pageID` - Die id der angefragten Seite

Returns:

der Name aller Inhalte, welche zur angefragten Seite gehören

Throws:

`java.sql.SQLException` - Datenbankfehler

getAllBooks

```
public java.lang.String[] getAllBooks()
                                 throws java.sql.SQLException
```

Diese Methode liefert den Namen aller Bücher welche in der Datenbank angelegt sind

Specified by:

`getAllBooks` in interface `IDataBase`

Returns:

ein String[] mit den Namen aller angelegten Büchern

Throws:

`java.sql.SQLException` - Datenbankfehler

Overview Package Class Use Tree Deprecated Index Help

Prev Class **Next Class** Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method