dataManagement

Class Model

java.lang.Object
    dataManagement.Model

```
public class Model
extends java.lang.Object
```

Die Klasse Model dient der Kapselung der Datenhaltung und ist Teil des MVC-Design. Das Model verwaltet die Schnittstelle zur Datenbank und implementiert die nötige Verarbeitungslogik im Umgang mit der Datenbank. Dazu überprüft es die Existenz von Verzeichnissen und wirft die passenden Exceptions, falls es zu Konflikten in der Verarbeitung kommt. Das Model kann aus dem Pfad eines Elementes die id in der Datenbank bestimmen und diese zur weiteren Verarbeitung nutzen. Zielsetzung dieser Klasse ist ein komfortabler Zugriff auf die Daten, welcher lediglich über Pfadnamen operiert. Die Komplexität der Datenbank soll somit gekapselt werden und vom Controller aus nicht mehr ersichtlich sein. Die Schnittstelle zum Controller soll minimal gehalten sein. Das Model serialisiert und deserialisiert den Seiteninhalt, welcher in der Datenbank abgelegt werden soll.

Constructor Summary

Constructors

Constructor and Description
`Model()` Instanziiert die Attribute zum Serializer und db Pfad und Logininformationen zur Datenbank sind übergangsweise "hard coded"

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`createBook(java.lang.String bookName)` Überprüft ob das zu erstellende Buch bereits existiert, falls nicht leitet diese Methode den Auftrag ein Buch zu erstellen an die Datenbankschnittstelle weiter
`void`	`createChapter(java.lang.String chapterName, java.lang.String bookName)` Die id des Buches in welchem das Kapitel gespeichert werden soll wird geladen Überprüft ob das zu erstellende Kapitel bereits existiert, falls nicht leitet diese Methode den Auftrag ein Kapitel zu erstellen an die Datenbankschnittstelle weiter

Page	**createPage**(java.lang.String pageName, java.lang.String chapterName, java.lang.String bookName) Die id des Kapitels in welchem die Seite gespeichert werden soll wird geladen Überprüft ob die zu erstellende Seite bereits existiert, falls nicht leitet diese Methode den Auftrag die Seite zu erstellen an die Datenbankschnittstelle weiter Die Attribute der Seite werden mit Defaultwerten gefüllt
void	**deleteBook**(java.lang.String bookName) Diese Methode lädt die id des zu löschenden Buches.
void	**deleteChapter**(java.lang.String chapterName, java.lang.String bookName) Diese Methode lädt die id des zu löschenden Kapitels.
void	**deletePage**(java.lang.String pageName, java.lang.String chapterName, java.lang.String bookName) Diese Methode lädt die id der zu löschenden Seite.
Tree	**getTree**() Der Verzeichnisbaum wird aus der Datenbank abgerufen und als Objekt der Klasse Tree zurückgegeben.
Page	**loadPage**(java.lang.String pageName, java.lang.String chapterName, java.lang.String bookName) Es wird überprüft ob der angegebene Pfad gültig ist.
void	**savePage**(**Page** page) Diese Methode speichert eine Seite.

Methods inherited from class java.lang.Object

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

Constructor Detail

Model

```
public Model()
      throws java.lang.ClassNotFoundException,
             java.sql.SQLException
```

Instanziiert die Attribute zum Serializer und db Pfad und Logininformationen zur Datenbank sind übergangsweise "hard coded"

Throws:

`java.lang.ClassNotFoundException`

`java.sql.SQLException`

Method Detail

createBook

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

Überprüft ob das zu erstellende Buch bereits existiert, falls nicht leitet diese Methode den Auftrag ein Buch zu erstellen an die Datenbankschnittstelle weiter

Parameters:

`bookName` - der Name des Buches

Throws:

`DirectoryAlreadyExistsException` - das Buch existiert bereits

`java.sql.SQLException` - Datenbankfehler

createChapter

```
public void createChapter(java.lang.String chapterName,
                  java.lang.String bookName)
                    throws DirectoryAlreadyExistsException,
                           java.sql.SQLException,
                           DirectoryDoesNotExistsException
```

Die id des Buches in welchem das Kapitel gespeichert werden soll wird geladen Überprüft ob das zu erstellende Kapitel bereits existiert, falls nicht leitet diese Methode den Auftrag ein Kapitel zu erstellen an die Datenbankschnittstelle weiter

Parameters:

`bookName` - der Name des Buches

`chapterName` - der Name des Kapitels

Throws:

`DirectoryDoesNotExistsException` - das Buch existiert nicht

`DirectoryAlreadyExistsException` - das Kapitel existiert bereits

`java.sql.SQLException` - Datenbankfehler

createPage

```
public Page createPage(java.lang.String pageName,
               java.lang.String chapterName,
               java.lang.String bookName)
                 throws DirectoryAlreadyExistsException,
                        java.sql.SQLException,
                        DirectoryDoesNotExistsException
```

Die id des Kapitels in welchem die Seite gespeichert werden soll wird geladen Überprüft ob die zu erstellende Seite bereits existiert, falls nicht leitet diese Methode den Auftrag die Seite zu erstellen an die

Datenbankschnittstelle weiter Die Attribute der Seite werden mit Defaultwerten gefüllt

Parameters:

`bookName` - der Name des Buches

`chapterName` - der Name des Kapitels

`pageName` - der Name der Seite

Throws:

`DirectoryDoesNotExistsException` - das Buch oder Kapitel existiert nicht

`DirectoryAlreadyExistsException` - die Seite existiert bereits

`java.sql.SQLException` - Datenbankfehler

deleteBook

```
public void deleteBook(java.lang.String bookName)
                throws java.sql.SQLException,
                       DirectoryDoesNotExistsException
```

Diese Methode lädt die id des zu löschenden Buches. Falls das Buch nicht existiert wird eine Exception geworfen. Falls das Buch existiert wird es gelöscht. Alle Unterverzeichnisse dieses Buches werden durch das dmbs mittels ON DELETE CASCATE gelöscht.

Parameters:

`bookName` - Der Name des Buches

Throws:

`java.sql.SQLException` - Datenbankfehler

`DirectoryDoesNotExistsException` - das Buch existiert nicht

deleteChapter

```
public void deleteChapter(java.lang.String chapterName,
                java.lang.String bookName)
                  throws java.sql.SQLException,
                         DirectoryDoesNotExistsException
```

Diese Methode lädt die id des zu löschenden Kapitels. Falls das Buch oder Kapitel nicht existiert wird eine Exception geworfen. Falls das Kapitel existiert wird es gelöscht. Alle Unterverzeichnisse dieses Kapitels werden durch das dmbs mittels ON DELETE CASCATE gelöscht.

Parameters:

`bookName` - Der Name des Buches

`chapterName` - Der Name des Kapitels

Throws:

`java.sql.SQLException` - Datenbankfehler

`DirectoryDoesNotExistsException` - das Buch oder Kapitel existiert nicht

deletePage

```
public void deletePage(java.lang.String pageName,
              java.lang.String chapterName,
              java.lang.String bookName)
                throws java.sql.SQLException,
                       DirectoryDoesNotExistsException
```

Diese Methode lädt die id der zu löschenden Seite. Falls das Buch, Kapitel oder Seite nicht existiert wird eine Exception geworfen. Falls die Seite existiert wird sie gelöscht. Alle Unterverzeichnisse dieser Seite werden durch das dmbs mittels ON DELETE CASCATE gelöscht.

Parameters:

`bookName` - Der Name des Buches

`chapterName` - Der Name des Kapitels

`pageName` - Der Name der Seite

Throws:

`java.sql.SQLException` - Datenbankfehler

`DirectoryDoesNotExistsException` - das Buch oder Kapitel existiert nicht

savePage

```
public void savePage(Page page)
              throws DirectoryDoesNotExistsException,
                     java.sql.SQLException,
                     java.io.IOException
```

Diese Methode speichert eine Seite. Alle Pfadangaben liegen in der PageInformation. Die PageInformation

beinhalten ebenfalls alle Informationen zum Speichern der Seite. Die Schnittstelle zum Controller besteht nur aus dieser Methode, das Speichern der Seiteninformation wird bewusst gekapselt und in dieser Methode ebenfalls ausgeführt. Jede Seite speichert alle ungespeicherten Änderungen, welche an ihr vorgenommen werden. In dieser Methode wird über die Liste der änderungen iteriert und abhängig von der ContentInstruction der Seiteninhalt serialisiert und erstellt, gespeichert oder gelöscht.

Parameters:

`page` - die zu speichernde Seite

Throws:

`DirectoryDoesNotExistsException` - die Seite oder ein Seiteninhalt existiert nicht

`java.sql.SQLException` - Datenbankfehler

`java.io.IOException` - Fehler beim Serialisieren

loadPage

```
public Page loadPage(java.lang.String pageName,
            java.lang.String chapterName,
            java.lang.String bookName)
              throws java.sql.SQLException,
                     DirectoryDoesNotExistsException,
                     java.lang.ClassNotFoundException,
                     java.io.IOException
```

Es wird überprüft ob der angegebene Pfad gültig ist. Das Laden einer Seite besteht aus zwei Schritten. Zuerst wird die PageInformation aus der Datenbank abgerufen und ein Objekt der Klasse Page erstellt. Danach wird der Seiteninhalt zu dieser Seite geladen, deserialisiert und der Seite zugefügt.

Parameters:

`pageName` - der Name der Seite

`chapterName` - der Name des Kapitels

`bookName` - der Name des Buches#

Returns:

die angefragte Seite

Throws:

`java.sql.SQLException` - Datenbankfehler

`DirectoryDoesNotExistsException` - der Pfad zur Seite ist falsch

`java.lang.ClassNotFoundException` - Fehler beim deserialisieren

`java.io.IOException` - Fehler beim deserialisieren

getTree

```
public Tree getTree()
             throws java.sql.SQLException
```

Der Verzeichnisbaum wird aus der Datenbank abgerufen und als Objekt der Klasse Tree zurückgegeben. Dazu wird iterativ mittels 3 Schleifen immer zuerst das obere Level des Baumes erstellt, und dann zu jedem Element dieses Levels die Kindbäume geladen

Returns:

der Verzeichnisbaum

Throws:

`java.sql.SQLException` - Datenbankfehler