

1 Aufgabenstellung

1.1 Aufgabenstellung Teilbereich A

Für eine Filesystem-Simulation soll eine Java-Web-Applikation erstellt werden, in der folgende Technologien verwendet werden: Spring-Boot, für die Anbindung an die Datenbank Spring-Data-JPA, sowie Thymeleaf für die Erstellung des Frontends.

Programmablauf

Die Web-Seite der Anwendung ist entsprechend der Abbildungen zu gestalten. Bei Programmstart wird der Inhalt vom Root-Verzeichnis C: angezeigt.

Der **up**-Button dient zum Wechseln in das übergeordnete Verzeichnis. Befindet man sich im Wurzelverzeichnis C: so kann nicht mehr in ein übergeordnetes Verzeichnis gewechselt werden.

Rechts neben dem **up**-Button wird der aktuelle Verzeichnispfad angezeigt.

Unter dem **up**-Button wird der Inhalt des aktuellen Verzeichnisses tabellarisch angezeigt: zuerst die Namen aller Verzeichnisse alphabetisch aufsteigend sortiert, dann die Namen der Dateien und Links ebenfalls aufsteigend sortiert. In der zweiten Spalte wird der letzte Änderungszeitpunkt, in der dritten Spalte die Größe der Dateien in KB angezeigt.

Verzeichnisse werden als Button gerendert. Durch Klick auf eine Verzeichnis Button wird der Inhalt dieses Verzeichnis angezeigt.

Unter der Tabelle befindet sich ein Formular mit dessen Hilfe ein neues Verzeichnis oder eine neue Datei (Auswahl erfolgt durch die Radio-Buttons) in das aktuelle Verzeichnis eingefügt wird. Das Textfeld mit dem Namen muss ausgefüllt sein. Das Einfügen erfolgt durch Klick auf den **create**-Button. Die Anzeige der Tabelle wird sofort aktualisiert. Für das Änderungsdatum übernommen wird der aktuelle Zeitpunkt eingesetzt. Beim Erzeugen einer Datei wird für die Größe ein zufälliger Wert zwischen 100 und 10.000 berechnet.

File Explorer

^ up C:

admin	03.03.2016 12:22
home	11.10.2017 01:30

insert ☒ directory ☐ file

Name:

File Explorer

^ up C:\home\users

jon	05.11.2017 14:22	
liz	17.10.2016 10:05	
steve	13.08.2016 08:51	
17.png	18.09.2019 00:09	
Ac.ppt	20.04.2021 04:31	117 KB
Amet.tiff	09.01.2018 05:53	592 KB
DiamInMagna.png	28.10.2020 17:04	203 KB
Nec.ppt	20.03.2019 14:56	98 KB
Nisi.xls	17.12.2019 13:21	303 KB
NislVenenatisLacinia.mp3	27.11.2020 21:13	35 KB
Platea.mp3	15.03.2022 07:53	38 KB
Porta.mp3	12.11.2021 02:15	95 KB
Pretium.ppt	21.03.2018 21:51	263 KB
Quis.mp3	06.09.2019 15:42	68 KB
Ut.tiff	09.07.2020 05:05	167 KB

insert ☒ directory ☐ file

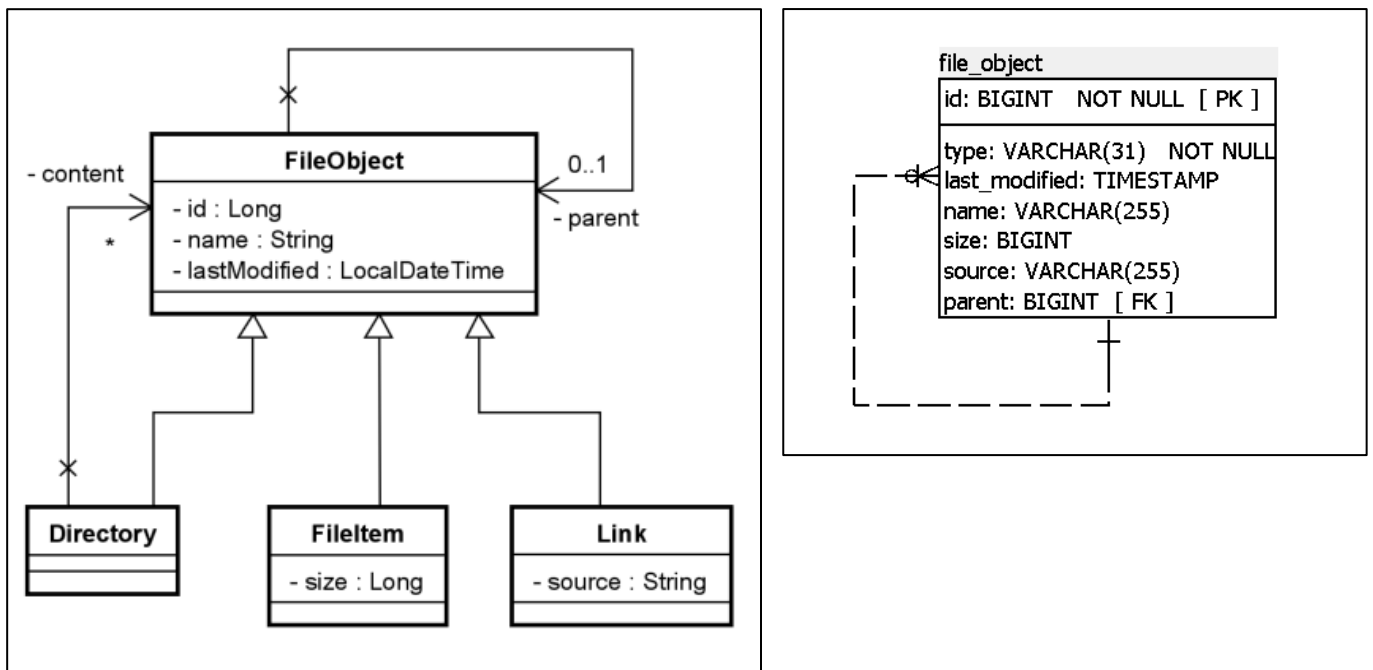
Name:

Datenbankanbindung

Die logische Struktur des simulierten Filesystems entspricht dem eines echten Filesystems: es gibt Verzeichnisse, die als Inhalt weitere Verzeichnisse (directories), Dateien (files) oder Verknüpfungen (links) enthalten können (siehe Klassendiagramm). Es handelt sich also um eine Baumstruktur mit genau einem Wurzelement.

Im Java Backend wird diese Struktur durch eine Klassenhierarchie abgebildet, bei der die Basisklasse **FileObject** alle Werte abbildet, die in allen Klassen vorhanden sind. Die drei abgeleiteten Klassen **Directory**, **FileItem** und **Link** enthalten weitere Elemente als Spezialisierungen.

Alle Daten für das Filesystem werden in der postgres-Datenbank Datenbank 'filesysdb' gespeichert. Die Java Vererbungsstruktur wird in der Datenbank in nur einer einzigen Tabelle abgebildet (siehe ERD). Testdaten befinden sich in der Datei 'filesysdb.sql', deren Inhalt über den pgAdmin in die Datenbank eingefügt wird. Der Zugriff auf die DB erfolgt mit Benutzernamen 'postgres' und dem Passwort 'geheim123'.



Die Diskriminator-Spalte **type** erhält die Werte **DIR**, **FILE** und **LINK**, je nachdem von welcher Klasse ein Objekt erzeugt wurde. Der Primärschlüssel wird automatisch in der Datenbank erzeugt.

Erstelle das OR-Mapping für diese Datenbank auf Basis von Spring-Data-JPA. Dazu sind die Vorgaben aus dem abgebildeten ER- und Klassendiagramm exakt einzuhalten. Verwende ausschließlich Annotationen. Implementiere zu den Klassen alle notwendigen Methoden.

Verwende die **Lombok** Library

Die Datei **application.properties** steht auf dem Prüfungsmoodle zur Verfügung.

Programmbeschreibung:

Die Klasse `pojos.FileObject.java`

- JPA-Mapping entsprechend dem ERD und dem KD.
- Der Primärschlüssel wird in der Datenbank erzeugt.
- `parent` referenziert das Verzeichnis in dem sich das aktuelle `FileObject` befindet. Die Assoziation ist unidirektional.

Die Klasse `pojos.Directory.java`

- Bildet ein Verzeichnis ab.
- JPA-Mapping entsprechend dem ERD und dem KD.
- Verwende für die Diskriminatorspalte den Wert `DIR`.
- Die Liste `content` referenziert alle `FileObject`-Objekte, die sich im aktuellen Verzeichnis befinden.

Die Klasse `pojos.FileItem.java`

- Bildet eine Datei ab.
- JPA-Mapping entsprechend dem ERD und dem KD.
- Verwende für die Diskriminatorspalte den Wert `FILE`.

Die Klasse `pojos.Link.java`

- Bildet eine Verknüpfung ab.
- JPA-Mapping entsprechend dem ERD und dem KD.
- Verwende für die Diskriminatorspalte den Wert `LINK`.

Die Klassen

`database.DirectoryRepository.java`




`database.FileItemRepository.java`

`database.LinkRepository.java`

- Repository Klassen für benötigte CRUD Methoden.
- `getDirectoryByName()` Implementiere die Query-Methode um ein Directory über seinen Namen zu bekommen.

Die Klasse `web.ExplorerController.java`




- Spring-Controller-Klasse für alle Requests.
Enthält die benötigten Session-Attribute.
Der Basispfad für alle Requests ist `"/explorer"`
- `getDirContent()` Aufruf der Methode bei einem `GET`-Request an `"/explorer"`, um den Inhalt eines Verzeichnisses aus der Datenbank zu holen und an das Frontend zu schicken.
Der Verzeichnisname oder der Verzeichnispfad wird als Requestparameter mitgeschickt.
Update aller Modell-Variablen.
Weiterleiten zum Frontend.




-  **updateModel()** Methode um alle Model-Variablen zu aktualisieren.
 Hier erfolgt die Sortierung des Verzeichnisinhaltes: zuerst werden alle Verzeichnisse aufsteigend nach ihrem Namen sortiert, dann alle Dateien und Links ebenfalls aufsteigend nach ihrem Namen sortiert. Verwende dazu einen Streaming-Ausdruck mit Methoden-Referenzen.
 Die Methode wird von allen Request-Methoden verwendet.
-  **getUpDirContent()** Aufruf der Methode bei einem **POST**-Request an `"/explorer/up"`, um den Inhalt des übergeordneten parent-Verzeichnisses aus der Datenbank zu holen und an das Frontend zu schicken.
 Der benötigte Verzeichnisname oder Verzeichnispfad wird als Requestparameter mitgeschickt.
 Update aller Modell-Variablen.
 Weiterleiten zum Frontend.
-  **createContent()** Aufruf der Methode bei einem **POST**-Request an `"/explorer/create"`, um ein neues Verzeichnis oder eine neue Datei einzufügen.
 Der Verzeichnisname oder der Verzeichnispfad bzw. der Typ (Verzeichnis oder File) werden als Requestparameter mitgeschickt.
 Es wird entweder ein neues leeres Directory erzeugt und in die Datenbank eingefügt, oder es wird ein neues File erzeugt und ebenfalls in die Datenbank eingefügt. Als Wert für **lastModified** wird der aktuelle Zeitpunkt verwendet, für **size** wird ein Zufallswert zwischen 100 und 10.000 berechnet.
 Update aller Modell-Variablen.
 Weiterleiten zum Frontend.

Die Datei `templates.ExplorerView.html`

Thymeleaf-Frontend zur Anzeige der Daten und zum Abschicken der Requests.

Verwende für die Schrift einen nicht-proportionalen Schriftsatz, wie z.B. monospace.

-  Ein Klick auf den **up**-Button erzeugt einen **POST**-Request an die URL `"/explorer/up"`.
 Der Directory-Pfad wird als Requestparameter mitgeschickt.
-  Anzeige des aktuellen Verzeichnisses rechts neben dem **up**-Button, wie abgebildet.
-  Anzeige der aktuellen Tabelle: Verzeichnisnamen werden in Buttons angezeigt, die Namen von Dateien und Links als normaler Text.
 Hinweis um in Thymeleaf eine Klassentyp abzufragen:

```
th:if="${animal instanceof T(my.project.Cat)}"
```
-  Bei Klick auf einen Verzeichnis-Button erfolgt ein **GET**-Request an die URL `"/explorer"`.
-  Anzeige der Radio-Buttons und Eingabe-Formular für ein neues Element.
-  Ein Klick auf den **create**-Button erzeugt einen **POST**-Request an die URL `"/explorer/create"`. Es werden der Wert des selektierten Radio-Buttons, das aktuelle Verzeichnis und der Name des neuen Elements im Request mitgeschickt.

Erstellen des IntelliJ Spring-Projekts, der Datei `pom.xml`, sowie Import der Datensätze