مدرس: دکتر احسان خامس‌پناه علی هدایی، میثاق محقق

MioBook

مقدمه

در این پروژه می‌بایست تمام داده‌های مورد نیاز برنامه را که در فازهای قبلی روی حافظه برنامه ذخیره و استفاده می‌شدند را روی یک پایگاه‌داده که حافظه ماندگار دارد، نگهداری کنید. ابتدا شما باید Schema پایگاه‌داده خود را طراحی کنید و پس از خواندن داده‌ها از این آدرس، آنها را روی پایگاه‌داده نوشته و هنگام استفاده از آن بازیابی کنید. قسمت Front-end پروژه در این فاز تغییری نمی‌کند ولی لازم است داده خوانده شده از پایگاه‌داده را نمایش دهد.

در این پروژه از پایگاه‌داده MySQL استفاده خواهید کرد. در این پایگاه‌داده رابطه‌ای (Relational Database)، داده‌ها در جدول‌هایی با تعداد Attribute ثابت و تایپ مشخص، ذخیره می‌شوند. برای طراحی پایگاه‌داده و ساخت جدول‌ها لازم است از آموخته‌های اولیه خود از درس پایگاه‌داده استفاده کنید و به طور مثال از نمودار Entity-Relationship کمک بگیرید.

شباهت زیادی بین صفت Entity-های پایگاه‌داده و صفت اشیاء دامنه برنامه در طراحی شی‌گرا وجود دارد که موجب راحتی ارتباط و تبدیل بین آن دو می‌شود. برای انجام این تبدیل الزامیست از ORM که در درس با آن آشنا شدید استفاده کنید. دقت کنید که همه داده‌های موجود در برنامه حتی اگر در این فاز از آنها استفاده‌ای نمی‌شوند، باید ذخیره شوند؛ زیرا ممکن است در آینده به آن نیاز داشته باشید و در کل هدف ما این است که داده‌های ما ماندگار باشند. در ادامه Entity-های اصلی برنامه برای درک بهتر توضیح داده می‌شوند. اما دقت کنید که نیاز است جدول‌های دیگری از جمله Relationship-های نمودار بین Entity-های اصلی را نیز تعریف کنید.

موجودیت‌های اصلی برنامه

کاربران

همانطور که در فازهای قبل پروژه پیاده‌سازی کردید، برای هر کاربر سامانه صفت‌های نام کاربری، رمز عبور، ایمیل، آدرس و نقش تعریف می‌شود که باید در جدول کاربران ذخیره شوند. مانند فاز قبل داده را از API مربوطه از آدرس سرویس خارجی خوانده و با دستور مناسب داخل پایگاه‌داده بریزید. هر دو صفت نام کاربری و ایمیل برای کاربران یکتا است؛ پس یکی از آن دو و یا یک ID جدید را به عنوان کلید اصلی (Primary Key) جدول تعریف کنید.
کاربران می‌توانند دو نقش ادمین و مشتری داشته باشند که می‌بایست با سلسله مراتب ISA مخصوص نمودار ER از یکدیگر جداگانه تعریف شوند. این ارث‌بری را می‌توانید توسط JPA پیاده‌سازی کنید.
همچنین آدرس را به جای یک صفت مرکب، یک موجودیت جداگانه تعریف کنید و یک ارتباط بین این دو موجودیت ایجاد کنید.

نویسندگان

در جدول نویسندگان باید کاربر ادمینی که نویسنده را اضافه کرده، نام، نام مستعار، تاریخ به دنیا آمدن، تاریخ فوت و ملیت نویسنده قرار بگیرد. این اطلاعات را از سرویس خارجی دریافت کرده و داخل پایگاه‌داده می‌نویسید. نام نویسنده یکتا است و می‌تواند به عنوان کلید اصلی در نظر گرفته شود.
باید ارتباطی بین رکورد نویسنده و رکورد ادمینی که آن را به سامانه اضافه کرده است ایجاد شود. این ارتباط در SQL به عنوان کلید خارجی (Foreign Key) شناخته می‌شود. کلید خارجی در واقع همان کلید اصلی یک جدول دیگر است و نوع صفت کلید خارجی و کلید اصلی متناظر آن باید یکسان باشد. این ارتباط را به گونه‌ای طراحی کنید که امکان اینکه یک ادمین، چند نویسنده را مدیریت کند نیز وجود داشته باشد. توجه کنید که هر نویسنده دقیقا یک ادمین دارد (OneToMany). باید محدودیتی ایجاد کنید که هنگام اضافه شدن یک نویسنده، ادمین آن قبلا به پایگاه‌داده اضافه شده باشد.

کتاب‌ها

در موجودیت کتاب اطلاعاتی شامل نام، نویسنده، ادمینی که کتاب را اضافه کرده، ناشر، سال نشر، قیمت، ژانرها، خلاصه و محتوای آن ذخیره می‌شود. نویسنده کلید خارجی به رابطه نویسنده و کاربر ادمین کلید خارجی به رابطه ادمین می‌باشد. همچنین نام کتاب به عنوان کلید اصلی می‌تواند در نظر گرفته شود.
می‌توانید برای راحتی کار ژانرها را به عنوان یک استرینگ comma separated در نظر بگیرید ولی اگر رابطه ManyToMany (و در نتیجه جدولی جداگانه) برای آنها تعریف کنید نمره امتیازی دریافت می‌کنید.

کیف پول

در این جدول نام کاربری مشتری و موجودی آن قرار دارد. هر مشتری دقیقا با یک رکورد از این جدول در ارتباط است و هر کیف پول متعلق به یک مشتری است و بدون وجود رکورد مشتری معنی ندارد (OneToOne). در نتیجه می‌توان نام مشتری را که کلید خارجی است را هم کلید اصلی این جدول در نظر گرفت.

کتاب‌های کاربران

در این جدول مشخص می‌شود که هر کاربر چه کتاب‌هایی را خریده است. در آن نام کتاب و نام مشتری به صورت کلید خارجی تعریف می‌شوند. همچنین باید مشخص شود که کتاب قرضی است یا خیر و اگر قرضی است، قرض کاربر چند روزه بوده و از چه تاریخی شروع می‌شود. این موجودیت یک شناسه یکتا نیز لازم دارد که می‌توانید به صورت Auto Increment برای آن تولید کنید.

سبد خرید

به دلیل پیچیدگی و وابسته بودن پیاده‌سازی این جدول به پیاده‌سازی شما در پروژه‌های قبل، می‌توانید سبدهای خرید را همچنان در حافظه اصلی ذخیره کنید ولی بهتر است که در پایگاه‌داده آن را به صورتی که معقول باشد نگهداری کنید.

بازخورد

موجودیت بازخورد شامل اطلاعات مشتری، کتاب، امتیاز داده شده، کامنت و زمان ثبت می‌باشد که در یک جدول ذخیره می‌شود. این موجودیت یک شناسه یکتا نیز لازم دارد که به صورت Auto Increment برای آن تولید کنید. مشتری و کتاب می‌بایست کلید خارجی قرار داده شوند.

توجه کنید که الزاما تناظر یک به یک بین جدول‌های پایگاه‌داده و اشیاء دامنه وجود ندارد و ممکن است که چندین جدول کمکی داشته باشید و یا جدولی که مطرح شده به صورت خودکار ساخته شود.

پایگاه‌داده

JDBC

JDBC یا Java Database Connectivity، پایین‌ترین لایه برای ارتباط بین یک برنامه جاوا و پایگاه‌داده است. این API به برنامه‌نویس اجازه می‌دهد تا پس از اتصال برنامه به پایگاه‌داده، با استفاده از دستورات SQL، مستقیماً روی پایگاه‌داده کوئری اجرا کند و نتایج را دریافت نماید.
در این روش:

- برنامه‌نویس باید به صورت دستی اتصال به پایگاه‌داده را مدیریت کند.
- کوئری‌های SQL باید مستقیماً به صورت استرینگ در کد نوشته شوند.
- نتایج کوئری‌ها به صورت ResultSet بازگردانده می‌شوند و باید با پیمایش Cursor آن، سطرهای جدول به صورت دستی به اشیاء دامنه برنامه تبدیل شوند.

انجام این فرآیند دستی برای همه اشیاء برنامه، منجر به تولید حجم زیادی کد تکراری با نگهداری سخت شده و احتمال رخداد خطا در آن نیز بالا خواهد بود. بنابراین با اینکه JDBC قابلیت استفاده از همه امکانات پایگاه‌داده را دارد، اما ابزارهای دیگری برای داشتن نگاهی سطح بالاتر تولید شده‌اند که این فرآیند را ساده‌تر می‌کنند.
فریم‌ورک‌های ORM یا Object Relational Mapper در جاوا، با استفاده از Annotation-ها و Metadata-ها، نگاشت بین جدول‌های پایگاه‌داده و کلاس‌های دامنه ما را ساده و خودکار می‌کنند.

JPA

JPA یا Java Persistence API یک استاندارد رسمی برای پیاده‌سازی ORM در زبان جاوا است. این استاندارد به برنامه‌نویس اجازه می‌دهد که کلاس‌های جاوا را به جدول‌های پایگاه‌داده نگاشت کند.
JPA با استفاده از Annotation-هایی مانند:

```
@Entity, @Table, @Id, @GeneratedValue, @Basic, @Column, @Enumerated,
@Embeddable, @Embedded, @OneToOne, @OneToMany, @ManyToOne, @ManyToMany,
@JoinColumn, @OnDelete, @Inheritance, @DiscriminatorColumn/Value
```

تعیین می‌کند که هر کلاس جاوا چطور باید به جدول‌های پایگاه‌داده نگاشت شود. شما با این API در کلاس آشنا شدید و دیدید که عملیات‌های CRUD یا Create-Read-Update-Delete و مدیریت تراکنش‌ها نیز از طریق EntityManager انجام‌پذیرند.
در صورت ایجاد لیست OneToMany در یک کلاس و ایجاد فیلد متناظر آن در کلاس دیگر به صورت ManyToOne، با استفاده از mappedBy مشخص می‌کنیم که کلید خارجی آن در کدام جدول قرار می‌گیرد.
توجه داشته باشید که JPA صرفا یک استاندارد است که مشخصات API را مستند کرده و یک پیاده‌سازی نیست.
برای آشنایی با JPA و Annotation-های نام‌برده، مطالب اصلی این لینک را مطالعه کنید.

Hibernate

ابزار Hibernate، یک فریم‌ورک ORM برای جاوا است. Hibernate در واقع یک پیاده‌سازی مشهور از استاندارد JPA است که امکانات پیشرفته‌تری را نیز در اختیار توسعه‌دهنده قرار می‌دهد. استفاده از این ابزار در پروژه شما الزامی است.

Hibernate پس از تنظیم صحیح موجودیت‌ها با Annotation-های JPA، به صورت خودکار از روی کلاس‌های ما، جدول‌های مربوطه در پایگاه‌داده را ایجاد می‌کند، داده‌ها را نگاشت می‌کند، و کوئری‌های موردنیاز را با زبان خاص خودش به نام HQL تولید می‌نماید. این چارچوب به شدت در مدیریت ارتباطات بین موجودیت‌ها، کش کردن، Lazy Loading و بهینه‌سازی پرس‌وجوها کاربرد دارد. در نهایت Hibernate برای ارتباط با پایگاه‌داده از JDBC استفاده می‌کند، ولی این جزئیات را از دید برنامه‌نویس پنهان می‌سازد.

Repository در Spring Data

Repository در واقع یک لایه سطح بالاتر است که توسط فریم‌ورک Spring Data ارائه شده تا کار با JPA و Hibernate را حتی ساده‌تر کند. در این روش، برنامه‌نویس به ازای هر Entity، فقط یک Interface تعریف می‌کند که از یک یا چند تا از Repository-های Spring Data مانند CrudRepository، ListCrudRepository، PagingAndSortingRepository و... ارث‌بری می‌کند. با این کار، متدهای آماده برای ذخیره، ویرایش، حذف و جستجوی موجودیت‌ها را در اختیار خواهد داشت.

مزیت بزرگ Repository این است که دیگر نیازی به نوشتن پیاده‌سازی متدها نیست چون که فریم‌ورک اسپرینگ به صورت خودکار این کار را انجام می‌دهد. یکی از امکانات جذاب این قابلیت اسپرینگ، Query Methods است که می‌توان متدهایی برای جستجو بر اساس نام، تاریخ، شناسه و غیره فقط با نام‌گذاری درست متد ایجاد کرد. به طور مثال، با ساخت یک متد به نام findByUserIdAndDate، کد مربوطه آن به طور خودکار تولید می‌شود.

در Repository، این امکان را داریم که در صورت نیاز، با استفاده از @Query، یک پرس‌وجو خام به پایگاه‌داده را انجام دهیم. همچنین امکان استفاده راحت از JPA Criteria API برای انجام پرس‌وجوهای پیچیده‌تر و اعمال شرط‌های زیاد در لایه پایگاه‌داده را با استفاده از JpaSpecificationExecutor داریم که می‌توانید در مورد آن تحقیق کنید.

این لایه باعث افزایش سرعت توسعه و خوانایی کد می‌شود. برای آشنایی بیشتر، می‌توانید بخش Spring Data JPA در این لینک را مطالعه کنید.

برای افزودن همه قابلیت‌های ذکر شده و Hibernate، تنها اضافه کردن spring-boot-starter-data-jpa و کانکتور MySQL به dependency-های maven کافی می‌باشد.

توجه کنید که در import کردن پکج‌های مربوط به servlet و persistence (که بخشی از Java EE هستند)، از مسیر jakarta و نه javax استفاده کنید.

نکات تکمیلی

- در صورت عدم استفاده از Hibernate بخش اعظمی از نمره شما کسر خواهد شد.
- از آنجا که اجرای پرس‌وجوها توسط DBMS بهینه‌سازی می‌شوند، فرآیندهای جستجو باید در سطح پایگاه‌داده انجام گیرند. مثلاً برای جستجوی یک کتاب نباید همه کتاب‌ها را از پایگاه‌داده خوانده و سپس در حافظه اصلی برنامه یک نام خاص را پیدا کنید. به جستجو در منطق برنامه نمره‌ای تعلق نمی‌گیرد.
- در صورتی که MySQL را روی سیستم خود نصب ندارید، ابتدا آن را نصب کنید و در آن یک دیتابیس بسازید. لازم است که هنگام اجرای برنامه MySQL بالا باشد تا اتصال به پورت مخصوص آن صورت پذیرد.
- دقت کنید تعریف کلید اصلی و کلید خارجی مناسب برای جدول‌هایتان ضروری است.
- برای ذخیره سازی اطلاعات مختلف از Data Type مناسب در شمای رابطه استفاده کنید.
- در صورتی که برای پیاده‌سازی قسمت خاصی از سایت از کدهای آماده موجود در اینترنت استفاده می‌کنید، نحوه کارکرد آنها را نیز یاد بگیرید و هنگام تحویل با آنها آشنایی داشته باشید.
- هنگام خواندن اطلاعات مختلف در شروع برنامه، در صورت وجود موارد جدید آنها را به پایگاه‌داده اضافه کنید و یا در صورت آپدیت شدن اطلاعات، تغییرات لازم را در پایگاه‌داده اعمال کنید. در غیر این صورت، تغییری در پایگاه‌داده ایجاد نکنید.
- هنگام استفاده از EntityManager نیازی به مدیریت اتصال‌ها (Connection Pooling) ندارید.
- استفاده از Repository مربوط به Spring Data بلامانع است.

نکات پایانی

- این تمرین در گروه‌های حداکثر دو نفره انجام می‌شود. برای تحویل آن کافی است که یکی از اعضای گروه، لینک مخزن گیت‌هاب و Hash مربوط به آخرین کامیت پروژه را در سایت درس آپلود کند. پروژه شما بر روی این کامیت مورد ارزیابی قرار می‌گیرد.
- حتما کاربر IE-S04 را به پروژه خود اضافه کنید.
- ساختار مناسب و تمیزی کد برنامه، بخشی از نمره همه پروژه‌های شما خواهد بود. بنابراین در طراحی ساختار برنامه و همچنین خوانایی کد دقت زیادی به خرج دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده شباهت بین کدهای دو گروه، از نمره هر دو گروه مطابق سیاستی که در کلاس گفته شده است کسر خواهد شد.
- سوالات خود را تا حد ممکن در گروه درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آنها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این تمرین ارتباط برقرار کنید.