1 Einleitung

Autor: Jonas Picker

Hier wird der Entwurf des Bibliotheksmanagementsystems BiBi beschrieben. Das Dokument bezieht sich auf das Lastenheft von Christian Bachmaier und Armin Größlinger und stellt eine technische Vertiefung unseres Pflichtenhefts dar, welches im Folgenden wiederholt referenziert wird.

2 Systemarchitektur

Autor: Ivan Charviakou

3 Klassendiagramm

Autor: Mohamad Najjar

4 JSF-Dialoge

Autor: León Liehr

5 Systemfunktionen

Autor: Jonas Picker

5.1 Technische Systemsicherheit

+Session-Hijacking **Kommunikationsverschlüsselung:** Durch das vorrausgesetzte SSL-Zertifikat des Tomcat-Servers wird der Anwendung die Kommunikation mit dem HTTPS-Protokoll ermöglicht. Dies ist eine Ergänzung der HTTP-Kommunikation um eine Transportverschlüsselung. Zusätzlich zur Vereitelung von Abhörversuchen der an Ihren Server gesendeten Anfragen signalisieren Sie den Klienten so auch die Vertrauenswürdigkeit Ihrer Institution durch die CA-Zertifizierung.
Nutzerberechtigungen: Um zu Verhindern, dass Nutzer ohne ausreichende Berechtigungen auf zugangsbeschränkte Bereiche des Webspaces zugreifen können, werden entsprechende Maßnahmen ergriffen: Um den direkten Aufruf von Seiten, die für den Nutzer nicht zugänglich sein sollten, zu unterbinden, verwenden wir einen sogenannten Phaselistener. Wie jede JSF-Anwendung durchläuft auch unsere nach Erhalt einer Anfrage eine Reihe von Verarbeitungsphasen, bevor die Antwort an den Klienten fertiggestellt und verschickt wird. Ein Phaselistener kann während dieses Prozesses zusätzliche Integritätsbedingungen abprüfen und gegebenenfalls die Antwort auf die Anfrage verändern/verhindern. Zusätzlich wird durch das in JSF eingebaute Session-Tracking[1] sichergestellt, dass die verschiedenen Nutzerrollen eine unterschiedliche Version der gleichen Seiten angezeigt bekommen, deren Funktionalitäten den Berechtigungen der Rollen entsprechen.

[1] Das Verfolgen der Klientenverbindung zwischen den eigentlich verbindungslosen HTTP-Anfragen

Vorbeugen von Angriffen: Durch geschicktes Einschleusen von SQL[2]-Code in ungesicherte Formularfelder könnten Unbefugte direkten Zugriff auf die darunterliegende Datenbank erhalten. Der Datenbankzugriff in unserem System erfolgt via JDBC[3], welche bereits über einen Mechanismus zur Verhinderung von diesen SQL-Injections verfügt. Hierbei wird verhindert, dass von Nutzern eingegebener Text vom Managementsystem der Datenbank interpretiert wird. Ein weiterer Angriffsvektor stellt das sogenannte Cross-Site-Scripting dar. Dabei wird versucht, nutzergenerierte Teile von Websiten mit vom Browser interpretierbaren Codeabschnitten zu füllen, welche dann beim Anzeigen des Inhalts bei anderen Klienten ausgeführt werden. Es gibt verschiedene Versionen dieser Angriffsart, von denen nicht alle am Server verhindert werden können. Jedoch wird in der Anwendungslogik jede Nutzereingabe auf interpretierbaren HTML (und eingebundenen JavaScript) Code untersucht und gegebenenfalls entschärft.

5.2 Fehlerbehandlung und Logging

Von uns vorhergesehene Fehler und einige im Hintergrund ablaufende Prozesse werden durch einen selbst erstellten Logging-Mechanismus dokumentiert. Da dieser Mechanismus während des Entwicklungsprozesses ebenfalls nützlich ist und auf feinster Einstellung für den Anwender uninteressant sein könnte, wird es eine Option geben, die den Detailgrad und die Anzahl der ausgegebenen Meldungen festlegt. Alle Meldungen werden in eine seperate Log-Datei geschrieben und können wahlweise auf der Konsole angezeigt werden. Vorhergesehene Fehler in Abläufen werden von der Anwendung bestmöglich abgehandelt und wie erwähnt dokumentiert. Sollte jedoch ein nicht vorhergesehener Fehler auftreten, wird der Grund dafür in From der Java-eigenen Exceptions auf die Konsole ausgegeben. Durch umfangreiches Testen der Anwendung vor Auslieferung versuchen wir, Sie vor einem solchen Fall zu bewahren.

5.3 Selbstständige Prozesse

Im System gibt es verschiedene Fristen (Abholungsmarkierung, Ausleihrückgabe [global, pro Medium, pro Nutzer], Gültigkeit des Verifizierungs-/Zurücksetzungslinks, Mahnungszeitversatz) und damit verbundene Aktionen. Um diese Aktionen ausführen zu können, muss das System in gewissen Abständen selbstständig auf die Datenbank zugreifen. In der Klasse ???? ist dafür ein eigener Wartungsthread implementiert.

5.4 Starten und Stoppen der Anwendung

Start: Beim Systemstart wird zunächst der oben beschriebene Logging-Mechanismus initialisiert. Im Anschluss liest das System die Konfigurationsdatei ein und versucht die dort angegebene Verbindung zur Datenbank. Bei erfolgreicher Verbindung wird überprüft, ob die benötigten Tabellennamen alle existieren bzw. legt diese bei deren Fehlen an. Sollte die Struktur der Tabellen fehlerhaft sein, wird eine Fehlermeldung geloggt und die Tabellen werden nicht vom System benutzt. Wenn die Datenbank fehlerhaft/nicht verbunden ist, werden eingehende Anfragen mit einer Fehlerseite beantwortet. Nach erfolgreichem Ablauf der vorhin genannten Prozesse wird als nächstes der Wartungsthread angestoßen, um fristenabhängige Aktionen durchzuführen. Danach ist das System vollständig betriebsbereit.

[2]Structured Query Language, eine weit verbreitete Datenbankanfragesprache
[3]Java Database Connectivity, eine universelle Datenbankschnittstelle

Stop: Beim planmäßigen Herunterfahren des Systems werden zunächst die noch offenen Datenbankverbindungen geschlossen und danach die Anwendung gestoppt. Dda dies sehr schnell passiert, wird währenddessen auf eine extra Umleitung der Nutzer auf eine Fehlerseite verzichtet. Da unsere SQL-Transaktionen dem ACID[4]-Prinzip folgen, wird die Datenbank jedoch auch beim Herunterfahren während eines Zugriffs in konsistentem Zustand hinterlassen. Nutzersessions überleben das Ausschalten des Servers nicht, d.h. alle eingeloggten Nutzer sind nach einem Neustart des Systems ausgeloggt. Wenn die Java Laufzeitumgebung des Systems plötzlich beendet wird und das System abstürzt, wird mittels einer sogenannten ShutdownHook trotzdem noch versucht, offene Datenbankverbindungen zu schließen. Dies ist bei einem Stromausfall allerdings unmöglich.

6 Datenfluss

Autor: Sergei Pravdin

7 ER-Modell

Autor: Jonas Picker

7.1 Legende

Jede Entität wird im Folgenden zusammen mit ihren nicht offensichtlichen Attributen und Relationen kurz beschrieben.
Medium: Diese Entität modelliert die in der Bibliothek gehaltenen Medien. Zusätzlich zum Primärschlüssel ist auch das Attribut 'Rückgabefrist' und 'Icon' nicht vom Administrator entfernbar, die restlichen Attribute (kursiv) stellen den modifizierbaren Standartsatz der Medienattribute dar (PfHft /D020/). Erwähnenswert ist hier noch das als mehrwertig markierte Attribut 'Autoren', diese Markierung kann auch zu benutzerdefinierten Attributen hinzugefügt werden (PfHft. /F380/).
Kategorie: Die mit dieser Entität verbundenen Relationen ordnen jedem Medium maximal eine Kategorie zu und modellieren die Kategoriehierarchie (PfHft. /W440/) durch die Selbstbeziehung.
Exemplar: Von jedem Medium muss mindestens ein Exemplar vorhanden sein. Auch kann ein bestimmtes Exemplar von genau einem Nutzer zur Abholung markiert werden, diese Aktion (genau wie die Abholung oder Rückgabe) ändert den Verfügbarkeitsstatus und die dazugehörige Operationsfrist dementsprechend.
Benutzer: Ob ein Benutzer die Ausleihfunktion benutzen kann, wird durch das Attribut 'Ausleihstatus' modelliert. 'Accountstatus' zeigt hingegen an, ob der Nutzeraccount bereits den Verifizierungsprozess durchlaufen hat (PfHft. /W70/). Das Passwort wird in gehashter Form abgespeichert.
Linkgenerator: Diese Entität kapselt einen befristet gültigen URL-Suffix aus dem der Verifizierungs- oder Passwortzurücksetzungslink (je nach Zweck) für genau einen Account erstellt wird.
Anwendung: Hier werden die setzbaren globalen Variablen und Anwendungseinstellungen gespeichert. Da die Tabelle nur einen Eintrag hat, wird auf einen Primärschlüssel verzichtet. 'Anonymer Zugang' modelliert die Berechtigungen anonymer Nutzer beim Besuchen des Webspaces (PfHft. /F10/), während 'Registrierungsstatus' eine nutzerfreundlichere Alternative als den RegEx zum Sperren der Registrierung bietet (PfHft. /F20/). 'Ausleihstatus' steht für das Umschalten des Sys-

[4]Atomicity, Consistency, Isolation, Durability

tems zur manuellen Freischaltung der Ausleihfunktion für registrierter Nutzer.

8 Installationsanleitung

Autor: Jonas Picker

8.1 Java Laufzeitumgebung

Um dem Tomcat das finden der richtigen Laufzeitumgebung zu erleichter, sollte diese bereits vor dessen Installation auf Ihrem Server zu finden sein. Zum Zeitpunkt der Entwicklung ist aktuellste Version 16 noch nicht voll Kompatibel mit allen von uns verwendeten Werkzeugen. Wir verwenden daher die JDK 15.0.2. Wählen Sie bitte die passende Version für Ihr Betriebssystem aus dieser Liste aus und laden Sie sie herunter. Da der Installationsprozess der offiziellen Java-Version der Firma Oracle sich etwas von dem der OpenJDK (welche ohne ausführliche Installationsanweisungen daherkommt) unterscheidet, sind hier Anleitungen von externen Quellen für die Betriebssysteme Windows Linux und macOS. Erwähnenswert ist noch die Tatsache, dass der Tomcat als letztes Mittel die Umgebungsvariable JAVA_HOME zum finden der JDK benutzt. Näheres entnehmen Sie bitte den Links im nun folgenden Abschnitt.

8.2 Apache Tomcat

Die Version 10.0.x des Apache Tomcat muss auf dem Server installiert sein. Die richtige Installationsanleitung für das auf dem Server installierte Betriebsystem ist unter diesem Link zu finden. Sie müssen die entsprechende Datei dann hier herunterladen. Achten Sie besonders auf die Abschnitte, die das Setzen der Umgebungsvariablen CATALINA_HOME und CATALINA_BASE beschreiben.

8.3 Registrieren des SSL-Zertifikats

Um verschlüsselte Kundenkommunikation zu ermöglichen, muss auf dem Tomcat nun das geforderte SSL-Zertifikat initialisiert werden. Der genaue Ablauf ist mit weiteren Informationen zusammen unter diesem Link zu finden.

8.4 Aufspielen der Anwendung

Unsere Applikation wird im .war-Format ausgeliefert, welches wie hier beschrieben auf verschiedenen Wegen auf den Tomcat Server aufgespielt ('deployed') werden kann. Wir empfehlen einen Weg zu wählen, der die Applikation nicht auf einen laufenden Tomcat aufspielt, da sonst wegen der noch leeren config.txt (siehe nächster Abschnitt) die Datenbankverbindung fehlschlagen kann und ein Neustart erforderlich ist.

8.5 Verbindung zur Datenbank

Unsere Applikation sollte nun unter dem Tomcat Installationsverzeichnis in einem Ordner zu finden sein. Navigieren Sie vor dem Start der Anwendung in den Ordner /WEB-INF und öffnen Sie das Textdokument config.properties. Sie finden dort Zeilen mit den Einträgen DB_HOST, DB_PORT,

DB_NAME, DB_USER und DB_PASSWORD. Befüllen Sie diese Zeilen nach dem Doppelpunkt mit einem Leerzeichen und folgenden Daten:

- DB_HOST: Die URL unter der der Datenbankserver erreichbar ist
- DB_PORT: Der Port über den Anfragen an den Datenbankserver geleitet werden
- DB_NAME: Der Name der Datenbank am Server
- DB_USER: Der Name des Benutzers der Datenbank
- DB_PASSWORD: Das Passwort zum Benutzernamen

8.6 Starten der Webanwendung

Nun sollten alle nötigen Schritte zur Installation abgeschlossen sein und Sie können die Webanwendung z.B. auf dem direkten Weg starten, indem Sie im Verzeichnis der gesetzten Umgebungsvariable CATALINA_HOME den Ordner /bin ansteuern und startup.sh aufrufen. Zum Abschalten des Tomcat wählen Sie dann shutdown.sh im gleichen Ordner.