Übungsplattform 2.0

12. März 2014

Dokumentation zur Logikschicht

Inhaltsverzeichnis

1 Einführung

Die Logikschicht stellt die Verbindung zwischen der Nutzerschnittstelle, dem Dateisystem und der Datenbank dar. Sie reagiert auf Anfragen von der Nutzerschnittstelle und gibt die entsprechenden Informationen zurück. Um diese Informationen zusammenzustellen, werden eigene Anfragen an die Datenbank und gegebenenfalls an das Dateisystem gestellt.

Der Informationsaustausch zwischen den verschiedenen Schichten erfolgt sowohl über Parameter in der Anfrage-URL, als auch über den Body der Anfrage – dieser stellt eine JSON-codierte Zeichenkette dar.

2 Struktur

Die Logik untergliedert sich in einen Controller und mehrere Komponenten. Anfragen, die die Logik bearbeiten soll, werden immer an den Controller gesendet. Dieser entscheidet anhand der Anfrage-URL, zu welcher Komponente er die Anfrage weiterleitet. Die jeweilige Komponente bearbeitet folgend die entsprechende Anfrage. Auch sie schickt ihre neu erstellten Anfragen und ihre Antwort immer an den Controller.

Diese Vorgehensweise hat mehrere Vorteile. Zum einen muss die Nutzerschnittstelle nur Wissen, wie die Adresse des Controllers ist. An diese sendet Sie ihre Anfragen. Sie benötigt jedoch kein Wissen darüber, auf welchem Server die eigentliche Komponente erreichbar ist. Auch die einzelnen Komponenten benötigen kein Wissen darüber, wo die Datenbank oder das Dateisystem liegt. Diese senden ebenso ihre kompletten Anfragen über den Controller.

Weiterhin ist das System durch diese Struktur leichter erweiter- und änderbar. Mehr dazu im Abschnitt 4.

Etwas negativ kann sich durch diese Struktur jedoch die Laufzeit entwickeln. Je komplexer die Informationen, die zu einer Anfrage zurückgegeben werden sollen, sind, desto länger dauert auch das zusammenstellen dieser. Dieser Effekt tritt stärker auf, wenn die einzelnen Schichten und der Controller von den Komponenten physisch getrennt sind. Läuft dagegen alles auf einem Server, treten keine relevanten Verzögerungen auf.

2.1 Die Arbeitsweise des Controllers

Der Controller nimmt eine Anfrage der Nutzerschnittstelle entgegen. Diese besteht aus einer URL, dem Header, dem Body und einer Methode.

Die URL (z. B. *http://100.200.3.1/uebungsplattform/logic/controller/user/id/1*) wird ab der Stelle nach dem Controller betrachtet. Im Beispiel also */user/id/1*. Anhand des Prefixes dieses URL-Stücks (hier */user*) entscheidet der Controller, wohin die Anfrage weitergeleitet werden muss. Dabei gibt es folgende mögliche Prefixe:

- */DB* Diese Anfrage wird an die Datenbank weitergeleitet.
- */FS* Diese Anfrage wird an das Dateisystem (Filesystem) weitergeleitet.

- */komponente1* Diese Anfrage wird an die Komponente mit dem Namen *Komponente1* weitergeleitet.

Bei dem Weiterleiten ersetzt der Controller seine eigene Adresse durch die Adresse der Zielkomponente. Ist die User-Komponente z. B. unter der Adresse *http://abc.de/uebungsplattform/logic/user* erreichbar, so leitet der Controller im obigen Beispiel die Anfrage an *http://abc.de/uebungsplattform/logic/user/id/1* weiter. Dabei übernimmt er die Methode sowie den Header und den Body der Anfrage.

2.2 Die Arbeitsweise einer Komponente

Komponenten können Unterschiedliche Funktionalität aufweisen. Zum einen leiten sie spezielle Anfragen nur weiter. Zum anderen geben Sie Informationen zur Darstellung einer Seite zurück oder leiten das Abspeichern von Daten und Dateien ein.

Eine Komponente nimmt Anfragen vom Controller entgegen. Je nach Aufrufmethode und URL wird durch diese Anfrage eine Funktion der Komponente aufgerufen. Beim ausführen dieser Funktion werden wesentliche Schritte ausgeführt um die Anfrage zu erfüllen. Gegebenenfalls werden Informationen zurückgegeben.

Benötigt die Komponente Informationen aus der Datenbank, so erstellt sie eine neue Anfrage an die Adresse des Controllers mit dem Prefix */DB*. Analog der Prefix */FS* für Anfragen an das Dateisystem.

Eine Übersicht aller Komponenten und zugehöriger Funktionen ist hier zu finden.

3 spezifische Bestandteile

3.1 .htaccess Dateien

Zu jeder Komponente gehört eine .htaccess Datei. Diese wird aktiv, sobald der Ordner, in dem sie gespeichert ist, über eine URL aufgerufen wird. In ihr wird definiert, an welche Datei der Aufruf gerichtet wird. Ein Beispiel hierfür ist in dem Ordner *Musterkomponente* zu finden.

3.2 CConfig.json Dateien

Die jeweilige Komponente ist in der Lage, aus dieser Datei die Informationen zu lesen, die sie zum versenden von Anfragen benötigt. Sie besteht zum einen aus Daten, die die Komponente selbst betreffen und zum anderen aus den Daten, die sie benötigt, um anderen Komponenten etwas zu senden:

- *id* Die interne Identifikationsnummer der Komponente.
- *name* Der Name der Komponente.
- *address* Die Adresse der Komponente.
- *prefix* Der Prefix dieser Komponente.

- *links* Einer Liste von diesen Informationen zu alle anderen Komponenten, mit denen diese Komponente kommuniziert.

Der Vorteil von der Verwendung von diesen Dateien liegt in der einfachen Änderbarkeit. Die CConfig Dateien zu allen Komponenten werden in der Datenbank erstellt und verwaltet. Ändert sich z. B. die Adresse des Controllers, so muss das nur einmal in der Datenbank geändert werden. Anschließend werden die aktualisierten CConfig Dateien an alle Komponenten verschickt. Ein Eintragen der neuen Adresse im Quellcode jeder einzelne Komponente ist dadurch nicht nötig.

Mehr zur Erzeugung dieser Dateien und der Verwaltung der dafür nötigen Daten ist in der Dokumentation zur Datenbank zu finden.

Ein Beispiel für eine solche Konfigurationsdatei ist in dem Ordner *Musterkomponente* zu finden.

3.3 GetSite-Komponente

Die GetSite-Komponente nimmt eine besondere Stellung für das Gesamtsystem ein. Sie wird dafür benutzt, die Daten, welche zur Darstellung der Seiten der Benutzeroberfläche nötig sind, zusammen zu stellen. Hierfür genügt eine einzige GET-Anfrage vom UI an einen bestimmten Prefix. Die entsprechende Funktion innerhalb der GetSite-Komponente führt daraufhin eine oder eine Abfolge von Anfragen an die Datenbank aus. Diese Daten stellt sie anschließend in der speziell für die jeweilige Seite entwickelten Datenstruktur zur Verfügung.

3.4 LFileHandler-Class

Um Dateien im System abzuspeichern sind mehrere Schritte nötig. Die LFileHandler Klasse enthält Funktionen zum Hinzufügen und Löschen einer Datei.

Zu einer Datei gehören nicht nur Daten die diese selbst betreffen, sondern auch Informationen darüber, um was für eine Datei es sich handelt (z. B. ein Attachment).

3.4.1 Datei speichern

Zur Speicherung aller Informationen sind folgende Schritte notwendig:

1. Speicherung der eigentlichen Datei im Dateisystem.
   → Rückgabe: Informationen zur Dateispeicherung (z. B. Speicheradresse, hash-Wert, Größe)

2. Speicherung der Informationen, die direkt diese Datei betreffen (z. B. Speicheradresse, hash-Wert, Größe), in der DBFile Tabelle.
   → Rückgabe: unter anderem die File ID, welche die Datei eindeutig identifiziert.

3. Speicherung der Informationen zu diesem Attachment. Dazu gehört die File ID und zusätzliche Daten wie z. B. die ExerciseID. Also die Identifikationsnummer der Übungsaufgabe, zu der dieser Anhang gehört.

Um dieses Vorgehen zu vereinfachen, kann die LFileHandler Klasse verwendet werden. der Aufruf *LFileHandler::add($file)* übernimmt die Schritte *1* und *2* und liefert ein File-Objekt zurück. Im Anschluss muss nur noch der Schritt *3* ausgeführt werden.

3.4.2 Datei löschen

Zum Löschen sind in umgedrehter Reihenfolge die Informationen der jeweiligen Schritte aus Abschnitt 3.4.1 zu entfernen. Hierbei müssen jedoch einige Bedingungen beachtet werden. Es können beispielsweise verschiedene Attachments auf die selbe FileID verweisen. Ist das der Fall, darf natürlich die eigentliche Datei nicht aus dem Dateisystem entfernt werden. Dies würde zu inkonsistenzen führen. Um solche Fehler zu vermeiden, kann die Funktion *LFileHandler::delete($file)* benutzt werden. Dieser wird ein File-Objekt übergeben.

Beispielsweise müssen erst alle Informationen zu einem Attachment gelöscht werden (Umkehrung Schritt *3*). Danach kann *LFileHandler::delete($file)* aufgerufen werden, um das File ordnungsgemäß zu entfernen (Umkehrung Schritte *1* und *2*).

4 Hinzufügen einer neuen Komponente

— — — Muster zu finden im Ordner Musterkomponente — — —