Projektstudium

Softwareentwicklung am Beispiel eines internationalen "Big Data"-Projektes

Task: #9 (***)

Meine Aufgabe bestand darin, für das Interface RepositoryChannel einen Dekorierer zu implementieren, der Statistiken über die Anzahl von Lese- und Schreib-Anfragen, die Geschwindigkeit der Abarbeitung dieser Anfragen und die Größe der übertragenen Byte-Blöcke berechnen, speichern und bereitstellen sollte. Ziel war es, mittels des Dekorierers ein besseres Verständnis über die verschiedenen IO-Prozesse, die von RepositoryChannels bearbeitet werden, erlangen zu können und so ggf. nicht richtig funktionierende Anwendungen ausfindig zu machen.

Um ein besser Verständnis über die Aufgabe und die Rolle des RepositoryChannel und des zu entwickelnden Dekorierers in der Dcache-Anwendung zu bekommen, habe ich mich zunächst mit dem Interface und dessen implementierenden Klassen auseinander gesetzt. Resultat davon war die Erkenntnis, dass der RepositoryChannel eine elementare Rolle beim Lesen und Schreiben von Daten von bzw. auf Datenträger der Dcache-Pools spielt. So wird beispielsweise ein RepositoryChannel erzeugt, wenn ein Client eine bestimmte Datei lesen möchte. Dieser ist dann für die Übertragung der Daten an den Client verantwortlich. Wird die Datei wieder geschlossen, so wird auch der RepositoryChannel gelöscht. Bei jeder neuen IO-Anfrage wird jeweils ein neuer RepositoryChannel erzeugt.

Des Weiteren habe ich mich mit dem Entwurfsmuster “Dekorierer“ (engl. Decorator), welches verwendet werden sollte, auseinandergesetzt. Dieses Entwurfsmuster bringt den Vorteil mit sich, dass man flexibel und dynamisch die Funktionalität einer Klasse erweitern kann, ohne dabei die Klasse selbst verändern oder Unterklassen z.B. mittels Vererbung bilden zu müssen. Stattdessen wird eine separate Klasse, auch Dekorierer genannt, programmiert, die dasselbe Interface wie die zu dekorierende Klasse implementiert. Die zusätzliche Funktionalität realisiert der Dekorierer, indem er eine Instanz-Variable vom Typ des Interfaces besitzt. Die zu dekorierende Klasse kann so an den Dekorierer angehängt werden. Wird der Dekorierer dann aufgerufen, gibt dieser den Aufruf entweder verändert oder unverändert weiter.

Bei der Implementierung selber gab es keine außergewöhnlichen Probleme. Wichtig war aber sowohl zuvor als auch während der Entwicklung gründlich zu recherchieren und sich mit der Dcache-Anwendung auseinander zu setzten. Zudem bestand jederzeit die Möglichkeit, per E-Mail und Skype mit den Projekt-Beteiligten in Kontakt zu treten und so bei Fragen Rücksprache zu halten. Dies funktionierte gut und war eine gute Unterstützung bei der Realisierung der Aufgabe. Zudem gab es vor dem letztendlichen “Mergen” des neuen Codes mit der Dcache-Anwendung ein Code-Review und einen Test-Durchlauf, der dazu beigetragen hat, dass die Code Qualität sehr gut war und ich selber mit der Implementierung zufrieden war.

Brief introduction to the OpenID Connect authentication technology

OpenID Connect ist ein offenes Protokoll, das auf OAuth 2.0 Protokoll aufsetzt und dieses durch die Möglichkeit der standardisierten und sicheren Authentifizierung eines Benutzers erweitert. Mit OAuth 2.0 konnte bisher der Endbenutzer einer Anwendung lediglich den Zugriff auf Daten, die von einer anderen Anwendung verwaltet werden, erlauben. Dabei wurden keine Benutzerdaten, eine Benutzer-Id oder Information über den Autorisierungsprozess selber preisgegeben und es war nicht möglich, den Prozess selber zu beeinflussen. Somit fehlten OAuth 2.0 einige wichtige Fähigkeiten, die für das Login eines Benutzers und Single-Sign-On obligatorisch sind. OpenID Connect schließt diese Lücke.

Beteiligte Instanzen bei OpenID Connect sind ähnlich wie bei OAuth 2.0 ein Benutzer, der Daten autorisieren und sich selber authentifizieren kann, eine Relying Party, die die Anwendung darstellt, die Zugriff auf die Daten haben möchte und den Endbenutzer identifizieren will, ein Identity Server, der für die Autorisierung und Authentifizierung verantwortlich ist und ein UserInfo Endpunkt, der eine OAuth-geschützte API für den Zugriff auf Benutzerdaten darstellt.

Der Protokollfluss von OpenId Connect erweitert den von OAuth 2.0. um einige entscheidende Punkte. Dabei gibt es drei verschiedene Arten von Flüssen. Der verbreitetste ist der Authentication-Code-Flow, alternativ gibt es den Implicit-Flow und Hybrid-Flow. Konkreter wird bei dem Authentication-Code-Flow

eine Autorisierungs-Anfrage von der Relying Party an den Benutzer gestellt. Dabei kann die Relying Party der Anfrage den Scope-Wert “openid" hinzufügt und so den Benutzer zu einer zusätzlichen Authentifizierung auffordern. Der Benutzer wird dann zum Identity Server weitergeleitet, der die Autorisierung und Authentifizierung durchführt. Ist die Authentifizierung und Autorisierung erfolgreich, wird der Relying Party ein Autorisation-Grant übergeben. Mit diesem Autorisation-Grant kann sich die Relying Party anschließend beim Identity Server sowohl einen Access Token, als auch einen zusätzlichen, OpenID-spezifischen ID Token abholen. Der ID Token beinhaltet Informationen über die Identität des Benutzers und die durchgeführte Authentifizierung. Dieser Token ist beschränkt auf die Relying Party und darf von keiner anderen Anwendung verwendet werden. Zu den bereits im ID Token enthaltenen Daten kann sich die Relying Party weitere Informationen mittels des Access Tokens beim UserInfo Endpunkt holen.

OpenID Connect profitiert nicht nur von den grundlegenden Eigenschaften von OAuth 2.0 wie z.B. der Einfachheit, Sicherheit und der breiten Unterstützung verschiedenster Anwendungstypen, sondern entkoppelt zudem die Anfrage der Authentifizierung von der verwendeten Authentifizierungs-Methode. So können verschiedene Sicherheitsmaßnahmen zur Authentifizierung eingesetzt werden, ohne dabei Anpassungen bei der Relying Party vornehmen zu müssen. Ein weiterer Vorteil von OpenID Connect ist seine Offenheit und Verwendung von bereits existierenden Standards, wodurch eine Interoperabilität gegeben wird. Des Weiteren ist OpenID Connect erweiterbar, wodurch z.B. die Verschlüsselung der Identitätsdaten oder Session Management möglich wird.

Quellen

Artikel:

Dr. Torsten Lodderstedt: OpenID Connect: Login mit OAuth, Teil 1 – Grundlagen. Online im Internet unter https://www.heise.de/developer/artikel/OpenID-Connect-Login-mit-OAuth-Teil-1-Grundlagen-2218446.html?seite=4 (14.08.1017)

Jens Ihlenfeld: Entwurf für die nächste OpenID-Generation. Online im Internet unter https://www.golem.de/1005/75140.html(14.08.1017)

Sebastian Grüner: OpenID Connect fertiggestellt. Online im Internet unter https://www.golem.de/news/authentifizierung-openid-connect-fertiggestellt-1402-104838.html(14.08.1017)

Offizielle Protokoll Seiten:

http://openid.net/connect/ (14.08.1017)

https://oauth.net/2/ (14.08.1017)

Videos:

https://de.wikipedia.org/wiki/OAuth#Abstrakter_OAuth-2.0-Protokollfluss (14.08.1017)

https://www.youtube.com/watch?v=zEysfgIbqlg 5:48 min (14.08.1017)

https://www.youtube.com/watch?v=6DxRTJN1Ffo 7:22 min (14.08.1017)

https://www.youtube.com/watch?v=Kb56GzQ2pSk 5:42 min (14.08.1017)

https://www.youtube.com/watch?v=CPbvxxslDTU 6:33 min (14.08.1017)