Merkblatt Hausarbeit

Auf diesem Merkblatt sollen einmal kurz die wichtigsten Tipps und Tricks für die Hausarbeit im Vorlesungsfach „Internet-Anwendungsarchitektur“ gegeben werden. Bitte sehen Sie diese paar Seiten als kleine Hilfestellung, die keinen Anspruch auf Vollständigkeit haben. Es geht einfach nur darum, ein paar wichtige Hinweise zu geben, um Probleme oder unnötigen Mehraufwand zu vermeiden.

Grundsätzliches

Bei der Hausarbeit liegt das Hauptaugenmerk auf einer sauberen, durchdachten Architektur, der korrekten und effizienten Anwendung der einzelnen Technologien sowie auf der vollständigen und qualitativ guten Umsetzung der funktionalen Anforderungen. Das „hübsche Aussehen“ ist zweitrangig und nicht bewertungsrelevant. Sie sollten sich deshalb zuerst um die Funktion und dann um das GUI-Design kümmern.

Sie bearbeiten die Hausarbeit in Gruppen - teilen Sie sich also auf, wobei eine Aufteilung nach technologischen Gesichtspunkten am einfachsten ist. Es ist deutlich einfacher, nur eine Technologie richtig zu beherrschen als alle drei. Wichtig ist allerdings, dass Sie auch in den anderen Technologien grundlegend verstehen und wiedergeben können, was dort passiert.

Der Einsatz der besprochenen Technologien Struts 2/Angular – Spring – Hibernate/JPA ist verpflichtend. Denken Sie daran, dass eine möglichst einfache Installation möglich sein sollte (die müssen Sie schließlich auch beschreiben). Überlegen Sie also, welchen Installationsaufwand eventuell eingesetzte Zusatztechnologien bedeuten. Dies gilt insbesondere bei einer Abweichung im Bereich der Infrastrukturkomponenten wie bspw. Server oder Datenbank. Die hier in der Vorlesung eingesetzten Technologien sind für die Hausarbeit ausreichend.

Wir haben in der Hausarbeit quasi eine Kunde-Lieferant-Beziehung. Dabei stellen wir als Prüfer die Kunden da. Wenn Sie also Anpassungen/Erweiterungen an der Anwendung planen oder Unklarheiten haben, besprechen Sie dies mit dem Kunden.

Ein wichtiger Tipp: Wir haben zum Ende des 6. Semesters und im Repetitorium eine funktionsfähige Anwendung mit den oben genannten Technologien gebaut. Nehmen Sie diese Anwendung als Grundlage, außer Sie haben vor, die komplette Konfiguration und das Zusammenstellen der notwendigen Bibliotheken aufs Neue durchzuführen.

Anforderungen

Die Anforderungen an die Hausarbeit haben wir zum Ende des 6. Semesters vorgestellt. Hier noch einmal eine kurze Zusammenfassung:

- Abdeckung der Anwendungsfälle (optional gekennzeichnete sind davon ausgenommen)
- Keine 400er-/500er-Fehlermeldungen, keine Stacktraces
- Implementierung unter Anwendung einer Drei-Schichten-Architektur

- Verwendung der Frameworks Struts 2 oder Angular (gem. Aufgabenstellung), Spring und Hibernate
- Adäquater Einsatz von Session- und Request-Scope (sprich: Session-Scope nur dann, wenn nötig)
- Transaktionsbehandlung
- Explizite Fehler- und Ausnahmebehandlung
- Keine Compilerfehler oder -warnungen (evtl. müssen hierfür Compileroptionen angeben werden)
- Sinnvolle Codeformatierung und nicht mehr als 120 Zeichen pro Zeile
- Der Autor ist im Quellcode anzugeben
- Eine Quellcodedokumentation hat unter Nutzung von Javadoc zu erfolgen (ausgenommen sind Getter- und Setter-Methoden, die keinerlei Logik beinhalten)
- Tasklog pro Teilnehmer (Tag, Aufgabe, Stunden, Gesamtsumme)
- Teilnahme an der Feedbackrunde

Wichtige Implementierungshinweise

Nachfolgend noch ein paar allgemeine und ein paar technologiespezifische Hinweise für die Implementierung.

Allgemein

Es gibt ein paar grundsätzliche Fehler, die sehr gerne in Java gemacht werden, dabei allerdings eventuell nur sehr schwer wiedergefunden werden:

- Strukturieren Sie Ihre Klassen durch eine saubere Paketstruktur. Bitte nicht alle Klassen in nur einem Paket erstellen. Erstellen Sie bspw. Unterpakete für die einzelnen Schichten ("web", "service", "data") oder Technologien ("action", "controller", "service", "dao", "model").
- Legen Sie gemeinsame Namenskonventionen fest und einigen Sie sich auf eine Sprache. Nichts ist schlimmer als deutsch-englische Klassennamen. Sie werden damit definitiv durcheinanderkommen.
- Beachten Sie die Java-Namenskonventionen:
  - Paketnamen immer klein
  - Klassennamen in Camel-Case mit großem Anfangsbuchstaben (*ExampleService*)
  - Attribut- und Methodennamen in Camel-Case mit kleinem Anfangsbuchstaben (*exampleMethod*)
  - Nur Singular
- Dokumentieren Sie Ihre Klassen!! Wenigstens die Public API muss ausreichend dokumentiert sein (JavaDoc).
- Lassen Sie die Zugriffsmethoden (Getter und Setter) durch die IDE generieren.
- Wenn Sie Objekte auf Gleichheit prüfen wollen, nehmen Sie *equals()*. Der Vergleich von zwei Strings mit „==“ wird je nach Situation falsche Ergebnisse zurückgeben.
- Passen Sie auf *null*-Objekte auf. Auf *null* können Sie nichts aufrufen, auch kein *equals()*. Wenn Sie sich also unsicher sind, ob ein String eventuell *null* sein könnte, dann prüfen Sie zuerst

darauf. Alternative: Drehen Sie den Ausdruck um (aus '*s.equals("wert")*' wird '*"wert".equals(s)*').

- Wenn Sie die Methode *equals()* überschreiben, müssen Sie auch die Methode *hashCode()* überschreiben … immer! Die Fehler, die sich ansonsten ergeben, sind extrem schwer zu finden. Auch hier bietet die IDE Generatoren an.
- Wenn die IntelliJ IDEA Compiler-Fehler melden sollten oder viele Klassen auf nicht existente Importe verweisen, dann liegt es höchstwahrscheinlich daran, dass es fehlende Bibliotheken gibt. Wenn Sie Maven einsetzen und die benötigten Bibliotheken in der pom.xml angegeben wurden, hilft es meistens, die Maven-Bibliotheken zu importieren. Gehen Sie dazu in das Maven-Tool-Window (im Standard befindet es sich in der rechten Seitenleiste) und klicken Sie auf den Doppelpfeil, der den Import erneut ausführt. Je nach Internetverbindung und Rechner kann dies eine gewisse Zeit brauchen. Danach sollte das Kompilieren funktionieren.
- Wenn Sie manchmal das Gefühl haben, dass irgendwas nicht richtig aktualisiert wird, dann starten Sie den Server (bspw. Apache Tomcat) neu.

Angular

Ein paar Hinweise für das Angular-Framework:

- Wir empfehlen dringend den Einsatz von Angular CLI – wie in der Vorlesung. Das macht die Arbeit mit Angular deutlich einfacher und hilft, klassische Fehler in der Konfiguration zu vermeiden.
- Um Programmierfehler zu verhindern, ist es sinnvoll, den Compiler von Typescript in den "strict"-Modus zu bringen. So wird bspw. verhindert, dass Instanzvariablen uninitialisiert bleiben. Die Konfiguration erfolgt in der Datei tsconfig.json.
- Wenn etwas nicht wie gewünscht funktionieren sollte oder die IntelliJ bspw. eine Komponente oder Direktive nicht findet und Fehler meldet, hängt es ggf. nur damit zusammen, dass das notwendige Modul nicht importiert wurde.
- Bei nicht zu findenden Compilerfehlern, gerade innerhalb von eingesetzten Bibliotheken, hilft es manchmal, das "node_modules"-Verzeichnis zu löschen und anschließend die verwendeten Bibliotheken mit "npm install" neu zu laden.
- Je nach Vorliebe kann neben npm bspw. auch yarn als Paketmanager eingesetzt werden. Beides ist zugelassen, nur mischen sollte man beides innerhalb eines Projektes nicht. Das kann zu Problemen führen.
- Die Versionsangaben von Bibliotheken sollten möglichst spezifisch sein. Sonst kann es unter Umständen vorkommen, dass nach Abgabe der Hausarbeit eine neue Version einer Bibliothek zu Compiler- oder Laufzeitfehlern führt. Aus diesem Grund bitte auch die Datei package-lock.json bzw. yarn.lock mit ins Repository einchecken.

Einige der Hinweise im Struts-Bereich (s.u.) gelten aber auch für das Angular-Framework. Ansonsten gilt, dass man gerne Fragen stellen darf, wenn man an einer Stelle hängt und auch nach Recherche nicht weiterkommt. Wir werden zwar keine direkten Lösungswege bereitstellen, aber manchmal reicht ja schon ein Hinweis, um eine Idee für das richtige Vorgehen zu bekommen.

Struts

Nun ein paar Struts-spezifische Hinweise:

- Wenn Sie einmal nicht mehr wissen, wieso etwas nicht so funktioniert, wie Sie es geplant haben, denken Sie einmal im Kopf den HTTP-Mechanismus durch. Struts hängt sehr stark an den Prinzipien dieses Protokolls, und manchmal hilft es einfach, den Ablauf noch mal im Kopf durchzuspielen.
- Pro Action-Aufruf (= HTTP-Request) gibt es in der Regel auch nur eine Action-Klasse. Die Action enthält in der Regel also alles, was Sie zum Server schicken wollen, aber auch das, was auf der Folgeseite angezeigt werden muss.
- Die beiden oberen Punkte lassen sich häufig verhindern, wenn Sie vor dem Beginn der Entwicklungsarbeiten genau planen, welche Seite welche andere aufruft und welche Actions verwendet werden sollen. Malen Sie das auf. Dann haben Sie eigentlich schon die komplette Struktur für die Action-Konfiguration fertig. Fangen Sie bitte nicht erst beim Implementieren der Action damit an, sich zu überlegen, was eigentlich in der Action durchgeführt werden muss.
- Wenn Sie eine Validierung auf ein bestimmtes Textmuster in der Anwendung machen müssen, dann brauchen Sie entweder einen eigenen Validator (den Sie bauen müssen), oder Sie nehmen stattdessen den Regexp-Validator, der auf reguläre Ausdrücke prüfen kann. Häufig ist das die einfachere Variante.
- Wenn Sie eine feldübergreifende Validierung machen müssen, dann denken Sie daran, dass Sie auch in der Action Validierungen durchführen können. Und dort haben Sie in der Regel alle benötigten Daten zur Verfügung.
- Wichtig: Die Validierungen funktionieren nur, wenn die Action von der Klasse ActionSupport abgeleitet wurde.
- Wenn Sie das Gefühl haben, dass das gesamte Struts-Framework irgendwie nicht so richtig das macht, was es machen sollte, schauen Sie in der Struts-Konfiguration einmal nach, ob das "package"-Element in der Konfiguration vom Package "struts-default" oder "tiles-default" abgeleitet wurde. Ist dies nicht der Fall, fehlen wichtige Grundfunktionen.

Spring

Bei Spring gibt es eigentlich gar nicht so viel zu sagen:

- Teilen Sie Beans immer in Implementierungsklasse und Interface auf. Das ist nicht nur sauber, sondern nur so können Sie die aspektorientierten Transaktionsmechanismen nutzen.
- Nehmen Sie die Konfiguration von Hibernate und Transaktionen möglichst gleich zu Anfang vor.
- Stimmen Sie möglichst früh mit Ihrem Frontend-Kollegen ab, welche Service-Klassen mit welchen Methoden benötigt werden. Das ist der Vertrag zwischen den beiden Schichten, und beide Seiten müssen dies möglichst frühzeitig wissen, um mit der Implementierung anfangen zu können.
- Trennen Sie immer die Geschäftslogik vom Datenbankzugriff, sprich: Es gibt Service-Beans und DAO-Beans. Die Frontend-Schicht arbeitet nur mit den Services, DAOs werden nur Spring-intern von den Service-Klassen verwendet.

Hibernate

Letztlich auch noch ein paar Hinweise zu Hibernate:

- Denken Sie nicht in SQL. Hibernate ist ein Persistenzframework und keine Klassenbibliothek, über die man einfacher SQL-Statements absetzen kann. Schauen Sie sich den Graphen aus der Vorlesung an und verdeutlichen Sie sich die unterschiedlichen Objektzustände. Wenn ein Objekt persistent ist, werden alle Änderungen an diesem Objekt in die Datenbank geschrieben, egal ob Sie *persist()* oder *merge()* aufrufen!
- Wenn Sie eine *LazyInitializationException* bekommen, dann hat Ihr Frontend-Kollege höchstwahrscheinlich auf ein Attribut zugegriffen, dass von Hibernate noch nicht geladen wurde. Da in der Präsentationsschicht kein Persistenz-Kontext mehr existiert, funktioniert das Lazy Loading nicht. Lösung: Sie müssen die Daten schon im Backend der Anwendung initialisieren. Schauen Sie in die Vorlesungsfolien, dort sind unterschiedliche Möglichkeiten beschrieben.
- Setzen Sie bidirektionale Assoziationen nur ein, wenn Sie sie wirklich brauchen. Unidirektional ist deutlich einfacher. Muss es jedoch bidirektional sein, dann denken Sie daran, dass sie beide Seiten der Assoziation synchronisieren müssen (siehe Vorlesungsfolien). Denken Sie auch daran, dass Sie mit JPQL/HQL im Endeffekt auch bidirektionale Assoziationen aufbauen können, und zwar abfragespezifisch.
- Lassen Sie die Datenbank durch Hibernate generieren. Das spart deutlich Arbeit, und Sie haben auf jeden Fall ein effizientes und zum Objektmodell passendes Datenbankmodell.
- Nutzen Sie die Möglichkeit, Datenbankstatements automatisch durch Hibernate beim Start durchzuführen. Sie benötigen dafür eine Datei namens *import.sql* im Wurzelverzeichnis des Klassenpfads. Hierüber können Sie beispielsweise Testdaten erzeugen, da auch nach dem Neugenerieren der Datenbank wieder angelegt werden.

Datenbank

Als Datenbank würden wir die H2-Datenbank empfehlen, die wir auch schon in den Übungen verwendet haben. Sie macht die Installation eines Datenbankservers überflüssig und erleichtert durch die dateibasierte Datenhaltung die Zusammenarbeit in Gruppen.

Informationen zur Datenbank und die jeweils aktuelle Version gibt es auf der Homepage: http://www.h2database.com

Wenn Sie die Datenbank herunterladen und installieren, gibt es anschließend auch einen Eintrag im Startmenü (für die Windows-Benutzer), der einen Aufruf der webbasierten Konsole ermöglicht. Denken Sie daran, dass auf die Datenbank immer nur ein Prozess gleichzeitig zugreifen kann, sofern sie nicht im Server-Modus gestartet wird. D.h. Ihr Tomcat-Server muss heruntergefahren sein, bevor Sie mit der Konsole auf die Datenbank zugreifen können.

Hilfreiche Ressourcen im Internet

Zuletzt noch eine Auflistung von hilfreichen Internetadressen:

- http://struts.apache.org – Die Homepage von Struts
- http://www.angular.io – Die Homepage von Angular
- http://www.spring.io – Die Homepage vom Spring-Framework
- http://www.hibernate.org – Die Homepage vom Hibernate-Framework
- http://www.oracle.com/technetwork/java/index.html – Die Java-Homepage, über die hilfreiche Tutorials abrufbar sind

- http://www.h2database.com – Die Homepage der H2-Datenbank
- http://tomcat.apache.org – Die Homepage des Tomcat-Servers
- http://commons.apache.org – Hilfreiche Java-Bibliotheken für alle möglichen Einsatzgebiete

Schauen Sie auch die Vorlesungsunterlagen noch mal durch. Alles, was für die Implementierung an Technologie benötigt wird, haben wir besprochen. Und in der Regel finden Sie auch immer die entsprechenden Beispiele in den Folien.

Sollten Sie wirklich einmal nicht mehr weiterwissen, und auch die Dokumentation und die Internetrecherche konnten keine Antwort liefern, dann sind wir unter der Nordakademie-Email-Adresse stefan.reichert@nordakademie.de bzw. stephan.anft@nordakademie.de zu erreichen. Bitte gewähren Sie uns aber eine gewisse Zeit bis zur Antwort, da wir die Mailbox zwar regelmäßig, aber nicht täglich checken.