Prof. Dr.-Ing. R. Roosmann, M. Teuber, H. Plitzner

Aufgabenblatt 6

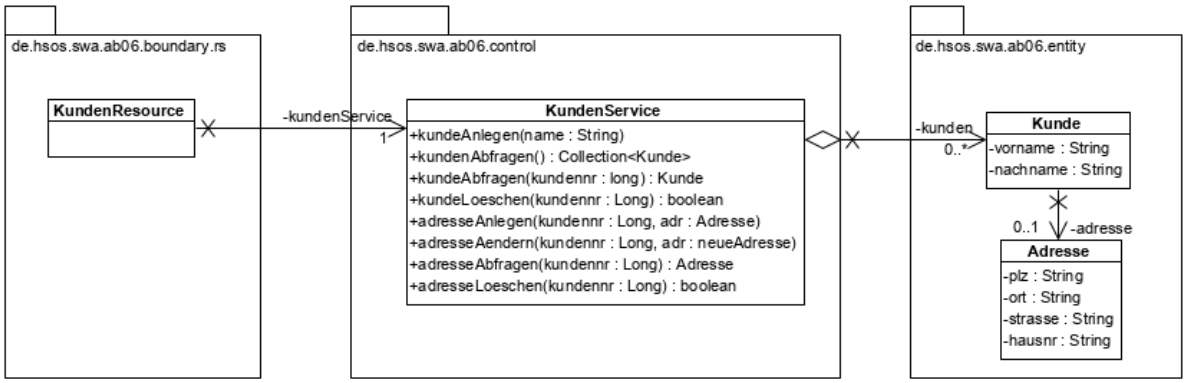
Ziel dieses Aufgabenblatts ist es, die Verwendung von CDI und JPA für eine ressourcen-orientierte Architektur einzusetzen.

Abgabe: Gr.1 09.12.24, Gr.2 12.12.24; Max. Punktzahl: 15; Min. Punktzahl: 8 Punkte

Grundsätzlich ist eine sinnvolle Software-Architektur zur Umsetzung der folgenden Aufgaben zu erstellen und umzusetzen.

Aufgabe 6.1: CDI Built-In Scopes (4 Punkte, 2er-Gruppe)
Umzusetzen ist eine REST-API, über die Kunden verwaltet werden können. Dies bedeutet konkret, dass Kunden hinzugefügt, abgefragt und gelöscht werden können. Kunden können eine Adresse haben. Die Adresse einer Kundin, eines Kunden lassen sich über die Primär-Ressource/kunden/1234 anlegen, wobei 1234 die Kundenummer darstellt. Natürlich soll es möglich sein, die Adresse einer Kundin, eines Kunden zu verändern.

Folgender Software-Architekturentwurf kann bspw. umgesetzt werden.

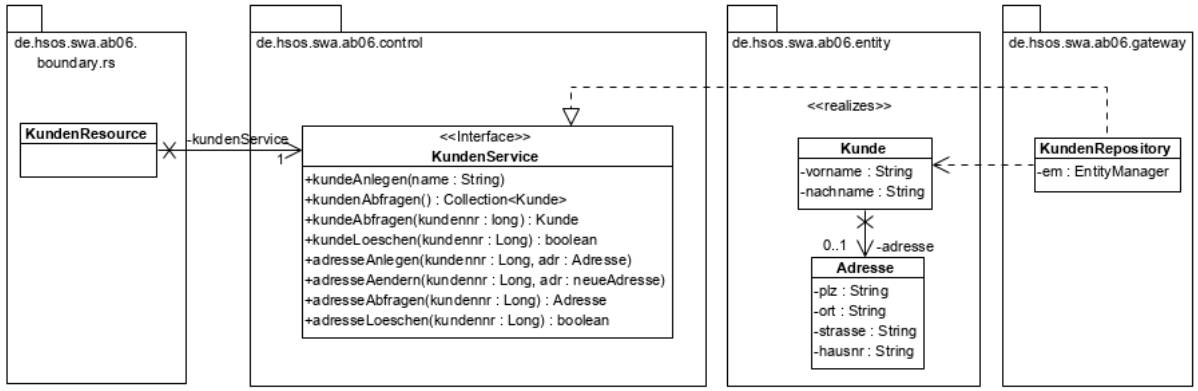


Die Daten sollen transient gehalten werden. Nutzen Sie das CDI Dependency-Injection wann immer möglich und sinnvoll.

Die REST-API soll anschließend über Swagger-UI, curl oder Insomnia aufgerufen werden. Kunden sind dazu anzulegen, Adressen hinzuzufügen und zu ändern. Dazu sollen folgende CDI-Build-in Scopes verwendet werden. Dokumentieren Sie die Zustände a) für einen und b) für zwei oder mehr zeitgleiche Zugriffe:

.boundary.rs	***.control***	***.entity***
@ApplicationScoped	@ApplicationScoped	@Dependent
@RequestScoped	@ApplicationScoped	@Dependent
@ApplicationScoped	@RequestScoped	@Dependent
@RequestScoped	@RequestScoped	@Dependent

Aufgabe 6.2: Persistierung mit JPA und JTA (5 Punkte, 2er-Gruppe)
Die Anwendung aus Aufgabe 6.1 ist so anzupassen, dass die Entity-Klassen persistiert werden. Setzen Sie dazu bspw. folgenden Software-Architekturentwurf um.



Die Daten sollen persistiert werden. Die Entity-Klassen sollen mit @Vetoed annotiert werden, um zu definieren, dass die entsprechenden Objekte nicht vom CDI-Container zu verwalten sind. Nutzen Sie die JPA- und JTA-Annotationen, um das OR-Mapping zu konfigurieren.

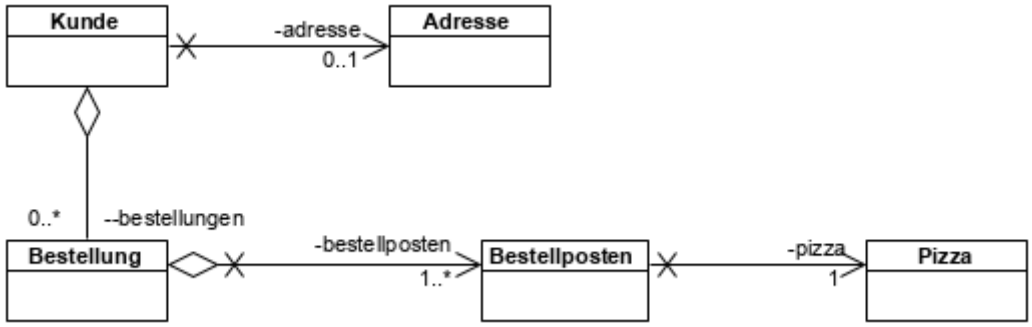
Verwenden Sie auch hier unterschiedliche CDI Build-in Scopes und dokumentieren die Auswirkungen bei einem oder mehreren konkurrierenden Zugriffen:

.boundary.rs	***.control***	***.entity***
@ApplicationScoped	@ApplicationScoped	@Dependent
@RequestScoped	@ApplicationScoped	@Dependent
@ApplicationScopde	@RequestScoped	@Dependent
@RequestScoped	@RequestScoped	@Dependent

Aufgabe 6.3: Pizza4Me REST-API (6 Punkte, 2er-Gruppe)
Mit dieser Aufgabe starten wir ein Projekt, das bis zum Ende des Praktikums umzusetzen ist. Wir gehen iterativ vor und fangen mit dem Entwurf einer REST-API, an.
Umzusetzen sind folgende funktionalen Anforderungen an die REST-API:

- Kunden können über die REST-API alle angebotenen Pizzen abfragen
- Kunden können für eine Pizza die Details abfragen, wie die Beschreibung und den Einzelpreis
- Kunden können Pizzen zu einer Bestellung hinzufügen und die Bestellung verwalten, u.a. indem sie die Mengen einer bestimmten Pizza ändern
- Kunden können Pizzen bestellen.

Das folgende unvollständige Klassendiagramm liegt vor.



Es ist eine Software-Architektur zu entwerfen und zu dokumentieren. Überlegen Sie sich, welche fachlichen Module, technischen Layer, DDD-Patterns usw. zum Einsatz kommen sollen. Orientieren Sie sich dabei an den vorangegangenen Aufgabenblättern. Dokumentieren Sie Ihre Architektur.

Rahmenanforderungen zu Pizza4Me:

- Die Nutzung der Anwendung soll über eine REST-API möglich sein.
- Informationen werden über json übergeben.
- CDI ist zum Abhängigkeitsmanagement zu verwenden, wann immer möglich und sinnvoll.
- Die Objektzustände sind über JPA zu persistieren.
- Die Transaktionssteuerung erfolgt über JTA.

Es ist das API-Design zu erstellen und zu dokumentieren. Beim Design starten Sie mit den Ressourcen und überlegen, welche Ressourcentypen angeboten werden sollen. Listenressourcen werden im Plural, Primärressourcen usw. im Singular beschrieben. Ordnen Sie den Ressourcen http-Methoden / -Verben zu und beschreiben Sie kurz das Ziel dieser Methode. Sollte ein Request-Body übergeben werden, dokumentieren Sie diesen beispielhaft im json-Format. Überlegen Sie sich http-Status Codes, die im Erfolgsfall und bei den Misserfolgsfällen über die Response zurückgegeben werden. Erweitern Sie die Responses um den

Resource	**http-Methode**	**Request: Parameter**	**Request: Body**	**Response: Statuscode**	**Response: Body**
/kunden	GET Ziel: Abfragen sämtlicher Kunden	--	--	200 OK ---------- 204 No Content	Json-Array sämtlicher Kunden ---------- {"info": { "message": "(#1001) Aktuell können keine Kunden übergeben werden",} }
	POST Ziel: Anlegen eines neuen Kunden	…	…	…	…
	PUT	…	…	…	…
	DELETE	…	…	…	…
/kunden/{id}	GET	…	…	…	…

Wichtig: diese Aufgabe wird mit Aufgabenblatt 7 fortgeführt. Sie finden Aufgabenblatt 7 in Ilias (für diejenigen, die die Web-Anwendung möglichst schnell abschließen möchten).

Prof. Dr.-Ing. R. Roosmann, M. Teuber, H. Plitzner

Das Testat der Pizza4Me-App findet am Gr.1 06.01.2024 und Gr.2 09.01.2024.

Hinweise:

- Java EE 8 Tutorial (u.a. Part VIII: Persistence):https://javaee.github.io/tutorial/toc.html
- Weil D. (2015): Java EE 7. Enterprise-Anwendungsentwicklung leicht gemacht, 2. Aufl., entwickler.press
- Ihns O., et al (2011): EJB 3.1 professionell: Grundlagen- und Expertenwissen zu Enterprise Java Beans 3.1, dpunkt.verlag

Viel Erfolg!!