

Praktikum "Daten- und Systemintegration" – WS 2009/2010

Praktikumsaufgabe 1

Labor : WI-Labor D14/1.11

Gruppen: 1 Praktikumsgruppe mit max. 16 Teilnehmern
in 2er-Teams

Prof. Dr. F. Bühler, Prof. Dr. G. Turetschek
Praktikumsunterstützung: Jan Horneck (B.Sc.)

Inhaltsverzeichnis

Inhaltsverzeichnis

1 Praktikum 1 – Web-Services.....	3
1.1 Ausführung der Beispielanwendung.....	3
1.1.1 Öffnen des Projektes.....	3
1.1.2 Kompilierung und Deployment der Anwendung.....	4
1.1.3 Ausführen der Anwendung.....	6
1.1.4 Analyse der Anwendung.....	8
1.2 Bereitstellen von Session-Beans als Webservices.....	9
1.2.1 Annotationen zum Erzeugen von Web-Services.....	9
Ausprobieren der Web-Services.....	11
1.3 Erstellen eines neuen Webservices.....	13
1.3.1 Anlegen des Projektes.....	13
1.3.2 Erstellen des Web-Service.....	14
1.3.3 Konsumieren der Webservices.....	16
1.3.4 Öffnen des WS-Client-Projektes.....	16
1.3.5 Generieren der Web-Service-Clients.....	17
1.3.6 Aufruf der Webservices (Erstellung der WS-Client-Anwendung).....	19
2 Abgabe.....	21
I. Anhang.....	22
Der Bestellprozess - Fachliche Sicht.....	22
Der Bestellprozess - Technische Sicht.....	23
Weitere Quellen.....	23
Häufige Fragen.....	24

1 Praktikum 1 – Web-Services

Ziel des ersten Praktikums ist das *Wrapping* einer *Legacy*-Anwendung mittels Web-Services. Die Anwendungsfunktionen, welche über *Session-Beans* zur Verfügung stehen, sollen anderen Anwendungen/Komponenten als Web-Services angeboten werden können.

Vorteile, die man sich hiervon verspricht, sind die leichtere Wiederverwendbarkeit der Funktionen auch außerhalb der *JavaEE*-Plattform, sowie die Möglichkeit mit Hilfe von BPM-Tools Geschäftsprozesse flexibler zu gestalten.

Aus fachlicher Sicht handelt es sich bei der Anwendung um die Implementierung eines Bestellvorgangs. Ein Kunde bestellt eine bestimmte Anzahl Exemplare eines Artikels. Diese Bestellung muss zunächst validiert und danach ausgeführt werden. Zunächst ist zu überprüfen, ob genügend Exemplare im Lager vorhanden sind. Falls nicht, erfolgt eine Nachbestellung. Anschließend werden die Artikel aus dem Lager ausgebucht. Schließlich wird der Preis und die Mehrwertsteuer der Bestellung berechnet. Der genaue Ablauf des Prozesses (fachliche und technische Sicht) ist im Anhang dargestellt.

In der *Legacy*-Anwendung ist der Bestellprozess durch eine Reihe von EJB-Komponenten (*Session-Beans* und *Entity Classes*) implementiert, welche von EJB-Client-Anwendungen angesprochen werden können. Die Verwendung von Web-Services soll die Möglichkeit bieten, dass Anwendungen, die für andere Plattformen / in anderen Programmiersprachen entwickelt wurden, auch diese Funktionen nutzen können. Da Web-Services standardmäßig das HTTP-Protokoll verwenden, ist nach dem *Wrapping* auch eine Kommunikation über Firewalls hinweg möglich.

1.1 Ausführung der Beispielanwendung

Der erste Schritt ist das Öffnen der *Legacy*-Anwendung in *Netbeans*, sowie das Kompilieren, *Deployment* und die Ausführung der Anwendung. Nachdem erste Erfahrungen im Umgang mit der Client-Anwendung gesammelt wurden, soll die Funktionsweise der Anwendung analysiert werden.

1.1.1 Öffnen des Projektes

Um das Projekt zu öffnen, wählen sie in der Menue-Leiste **File** → **Open Project** aus. Wechseln sie dann ins entsprechende Verzeichnis (**D:\DSI\Projekte\DSI-Praktikum-1\ausgang**) und wählen sie das Projekt **DSI1-ausgang** aus. Die Checkboxen **Open as Main Project** und **Open Required Projects** müssen aktiviert werden (siehe Abbildung 1).

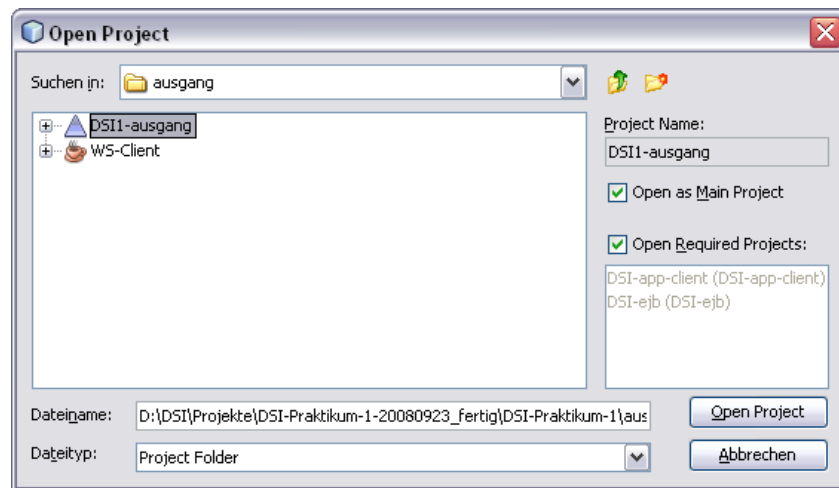


Abbildung 1: "Open Required Projects" auswählen

Nach dem Drücken auf den **Open Project**-Knopf werden 3 Projekte geöffnet.

DSI1-ausgang	Enterprise-Application-Projekt
DSI-app-client	EJB-Client
DSI-ejb	EJB-Komponenten

Das **DSI-ejb Projekt** enthält die *Legacy*-Anwendung in Form von *Enterprise Java Beans* und weitere Hilfsklassen. Hier wird die Geschäftslogik des Bestellprozesses abgebildet. Die Beans werden auf dem Applikationsserver in einem EJB-Container ausgeführt.

Das **DSI-app-client Projekt** implementiert den Applikations-Client. Es enthält die grafische Benutzeroberfläche sowie Hilfsklassen für die Anzeige von Daten. Es verwendet das DSI-ejb Projekt als Bibliothek, da dieses die Schnittstellen für die *Session Beans*, sowie gemeinsam verwendete Datentypen (Klassen) enthält. Dieses Projekt wird auf dem Client ausgeführt. Es handelt sich somit um einen Applikations-Client, der Zugriff auf Dienste und Funktionalitäten des Applikationsservers hat.

Das **DSI1-ausgang Projekt** ist ein sog. Enterprise-Application Projekt. Als solches ist es eine Art „Elternprojekt“ für die anderen beiden Projekte. Es ist eine Sammlung von verschiedenen EJB- und Web-Applikations-Komponenten. Die *Enterprise Application* enthält keinen Quellcode, sondern Konfigurationsdateien, welche das Zusammenspiel der einzelnen Komponenten beschreiben. Beim Aufruf von **Build** werden zunächst die beteiligten Komponenten kompiliert und dann ein Paket aus allen Komponenten, zusammen mit einigen Konfigurationsdateien, geschnürt. Dieses Enterprise Archiv (Dateiendung: .ear) wird dann auf dem Applikationsserver *deployed*.

1.1.2 Kompilierung und Deployment der Anwendung

Bevor die Anwendung ausgeführt werden kann, muss sie zunächst *kompiliert* und *deployed* werden. *Deployment* beschreibt die Auslieferung von Software auf ein Zielsystem, in unserem Fall die Auslieferung der Anwendung auf den Applikationsserver. Der Applikationsserver muss zuvor gestartet werden.

Um den Applikationsserver *Glassfish* zu starten, wird in die **Services**-Sicht gewechselt. Hier kann im Kontext-Menue des Glassfish-Knotens *Glassfish* gestartet werden (siehe Abbildung 2).

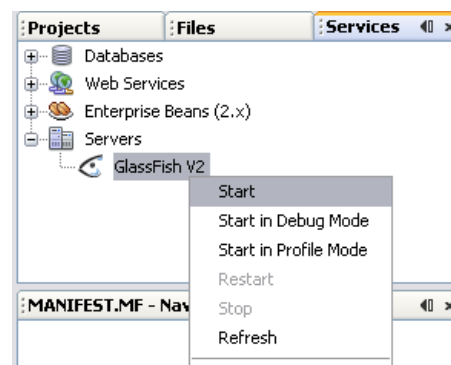


Abbildung 2: Starten des Glassfish-Servers

Das Starten des Servers nimmt einige Zeit in Anspruch. Im unteren Bereich von Netbeans, werden im Output-Reiter unter **Glassfish v2** die Ausgaben des Servers angezeigt.

Sofern Sie sich als Admin auf dem Applikationsserver einloggen müssen, lautet der Benutzername **admin** und das Passwort **adminadmin**.

Um die Anwendung zu kompilieren, wird wieder in die **Projects**-Sicht gewechselt, und im Kontextmenue von **DSII-ausgang** der Punkt **Build** ausgewählt. Sofern kein Fehler auftritt, wird in dem Ausgabefenster der Text **BUILD SUCCESSFUL** angezeigt.

Als letzter Punkt fehlt nun nur noch das *Deployment*, welches unter dem Punkt **Undeploy and Deploy** im selben Menue zu finden ist. Hierbei wird eventuell eine Version der Anwendung - welche bereits *deployed* wurde - wieder aus dem Applikationsserver entfernt und das kompilierte und neu zusammengeschnürte Archiv auf dem Server installiert.

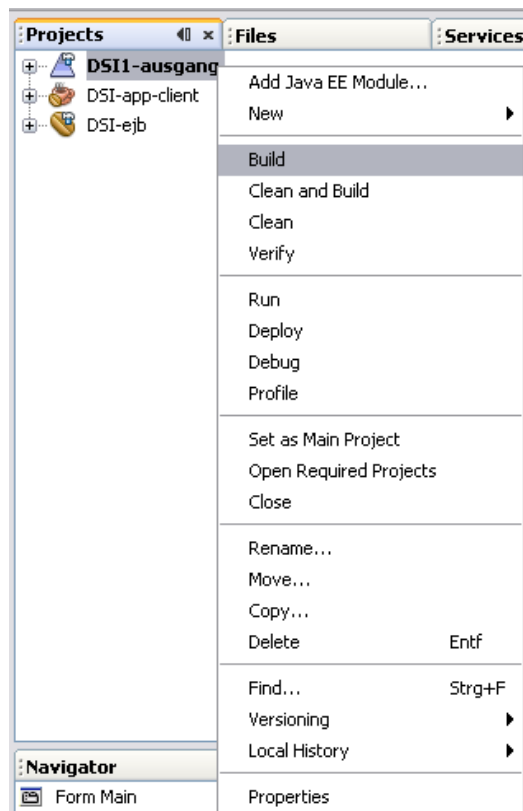



Abbildung 3: Kompilieren, Deployen und Starten der Anwendung

1.1.3 Ausführen der Anwendung

Mit einem Klick auf **Run** im selben Kontext-Menue wird die Anwendung gestartet und zeigt folgende Eingabemasken (siehe Abbildung 4 und Abbildung 5).



Rufen Sie **Build**, **Deploy** und **Run** immer aus dem Kontext-Menue von **DSI1-ausgang** ( **DSI1-ausgang**) aus auf. Der Aufruf von DSI-app-client funktioniert nicht, da der Client dann nicht die Beans kennt¹.

¹ Um die Komplexität von JNDI zu umgehen, wird Java-Webstart verwendet um die Beans auf den Client per Dependency-Injection einzufügen.

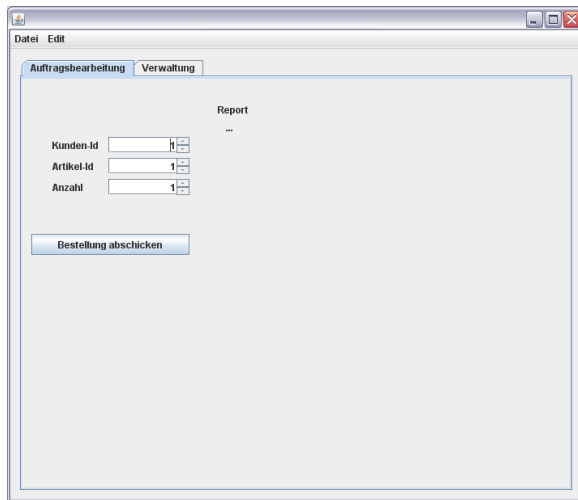


Abbildung 4: Die Auftragsbearbeitung

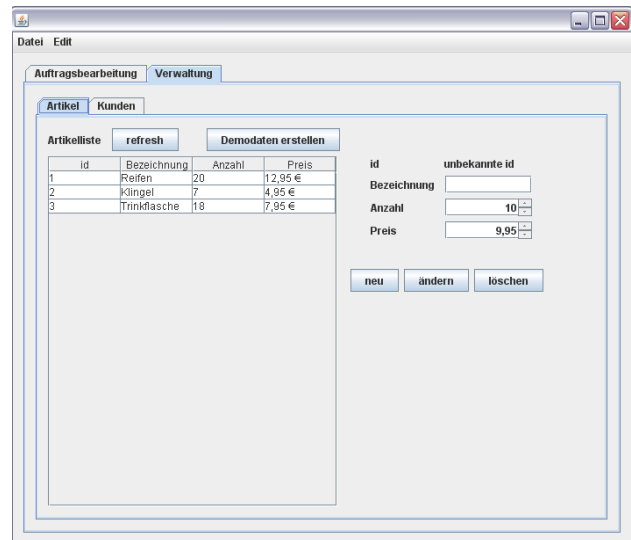


Abbildung 5: Die Verwaltung der Daten

Im GUI-Bereich **Auftragsbearbeitung** können Artikel bestellt werden. Jede Bestellung besteht aus einer **Kunden-Id**, welche einen Kunden-Datensatz referenziert, einer **Artikel-Id**, welche sich auf einen Artikel-Datensatz bezieht und der gewünschten **Anzahl** des zu bestellenden Artikels.

Die Artikel- und Kundendaten lassen sich im Reiter **Verwaltung** bearbeiten. Zunächst sollten Sie im Bereich **Verwaltung** jeweils in der Kunden- und Artikel-Ansicht den **Demodaten erstellen** Knopf betätigen, um die Datenbank zu füllen.

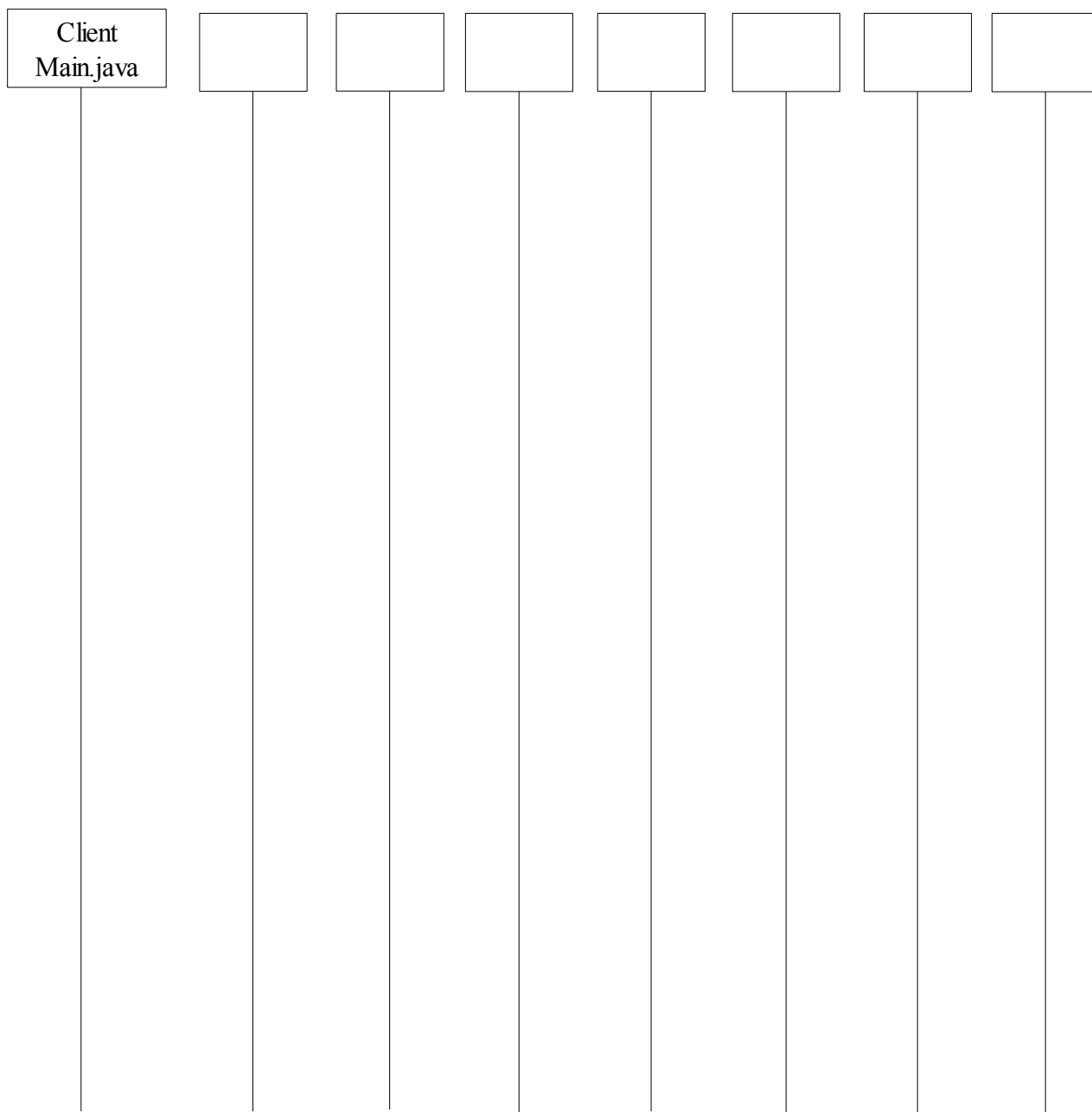
Im Bereich **Auftragsbearbeitung** können anschließend Artikel bestellt werden. Falls Artikel- und Kundennummer existieren, wird eine Erfolgsmeldung zurückgegeben, anderenfalls eine Fehlermeldung.

In der **Verwaltungs-Ansicht** können nun die Änderungen an der Datenbank, die das Durchführen einer Bestellung bewirkt hat, angesehen werden. Hierzu muss der **Refresh**-Button gedrückt werden. Wenn durch eine Bestellung die Anzahl eines Artikels kleiner als Null werden würde, wird zunächst nachbestellt, und zwar so viele Exemplare, dass sich nach der Durchführung noch mindestens 10 Stück im Lager befinden.

1.1.4 Analyse der Anwendung

Nachdem die Anwendung ausgeführt wurde und bevor es an die Erstellung der Web-Services geht, sollen Sie zunächst die bestehende Anwendung analysieren. Erstellen sie hierzu ein **UML-Sequenzdiagramm, das die Kommunikation der einzelnen Komponenten darstellt. Ausgangspunkt** ist das Client-Programm **Main.java** und die Methode **jButtonArtikelAbschickenActionPerformed()**. Diese GUI-Methode ruft die *bestelleArtikel*-Methode der *auftragsAbwicklungBean* auf. Stellen sie dar wie der Bestellprozess im Einzelnen abläuft. Vervollständigen Sie hierzu das nachfolgende Diagramm:

Tipp: Es muss nicht jedes argument eingetragen werden, die verwendeten Beans und deren aufgerufenen Methoden allerdings schon.



1.2 Bereitstellen von Session-Beans als Webservices

Als nächster Schritt sollen die Methoden der Session-Beans als Web-Services bereitgestellt werden. Da für eine später geplante Prozessintegrationsschicht nicht alle Bean-Methoden als Web-Services benötigt werden, reicht es aus, wenn folgende Session-Beans berücksichtigt werden:

- BestellBean
- ArtikelManagerBean
- LagerBean
- LieferantBean

Die Infrastruktur zum Bereitstellen von Web-Services wird von *Glassfish* bereits mitgeliefert². Der Entwickler hat die Aufgabe, die Klassen und Methoden kenntlich zu machen, für die ein Web-Service generiert werden soll. Hierfür müssen die Klassen und Methoden mit Meta-Daten versehen werden, die diese als Web-Service kennzeichnen. Die Meta-Daten werden mit Hilfe von Annotationen festgelegt.

1.2.1 Annotationen zum Erzeugen von Web-Services

Wie bereits bei der Entwicklung von *Enterprise Java Beans*, so werden auch bei der Deklaration von Web-Services Annotationen verwendet. Laut Wikipedia³ ist eine Annotation in Java ein Sprachelement welches die Einbindung von Metadaten in den Quellcode erlaubt.

Die für uns relevanten Annotationen sind:

@WebService	Wird vor die Klassendeklaration geschrieben. Bewirkt, dass alle Methoden mit der Sichtbarkeit <i>public</i> als Web-Service bereitgestellt werden.
@WebMethod	Wird vor eine Methodendeklaration geschrieben. Falls in einer mit @WebService annotierten Klasse, eine Methode mit @WebMethod annotiert ist, werden nur die Methoden als Web-Service bereitgestellt, die diese Annotation aufweisen.
@WebParam(name = "name")	Wird in der Methodendeklaration vor den entsprechenden Parameter geschrieben. Benennt den Parameter für den Web-Service Aufruf. Falls nicht vorhanden, werden <i>param0</i> , <i>param1</i> ... als Parameter-Namen verwendet.
@WebResult(name="name")	Wird in der Methodendeklaration vor den Rückgabe-Datentyp geschrieben. Benennt den Rückgabewert für den Webservice-Aufruf. Falls nicht vorhanden wird <i>return</i> verwendet.

Auch bei Annotationen ist, wie in Java üblich, auf die Groß-/Kleinschreibung zu achten. Die mit **Strg+Leertaste** zu aktivierende Code-Vervollständigung hilft auch beim Einfügen von Annotationen.

² Hierbei wird der Metro Webservice Stack verwendet, welcher die JAX-WS API implementiert.
<https://metro.dev.java.net/> Stand: 2008-07-16

³ [http://de.wikipedia.org/wiki/Annotation_\(Java\)](http://de.wikipedia.org/wiki/Annotation_(Java)) Stand: 2009-01-14

nen. Nach Hinzufügen der `@WebService`-Annotation wird auch automatisch die notwendige Import-Deklaration (`import javax.jws.WebService;`) hinzugefügt.

Es folgt ein Beispiel, wie ein Ausschnitt der Datei **BestellBean.java** nach dem Einfügen der oben beschriebenen Annotationen aussehen könnte (siehe Abbildung 6). Bei den anderen Klassen ist analog dazu vorzugehen.

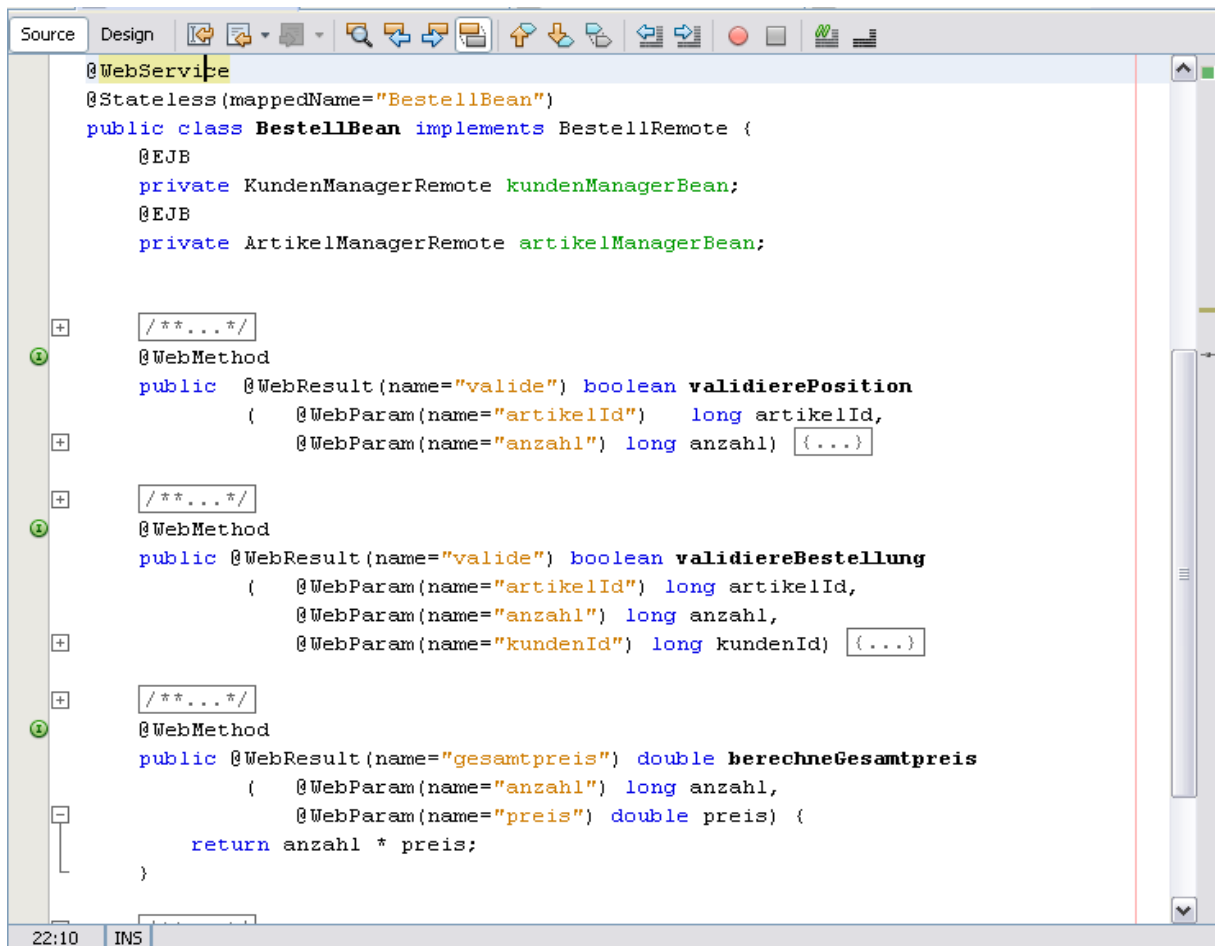


Abbildung 6: Web-Service-Annotationen der BestellBean-Klasse

Verwendung der Netbeans Web-Service Design-Ansicht

Sobald eine Klasse mit der `@WebService` Annotation markiert wurde, lässt sich der Web-Service in der **Design-Ansicht** im Paket **DSI-ejb\Web Services** öffnen (siehe Abbildung 7). Hier können mit Hilfe eines grafischen Editors zum Beispiel die Parameter-Namen geändert werden oder neue Operationen erzeugt werden. Der Name des Rückgabe-Wertes lässt sich allerdings nicht bearbeiten.

Es ist möglich, dass die Design-Ansicht nicht immer korrekt funktioniert. Jedoch stellt sie eine bequeme Methode zum erstellen von Web-Services bereit.

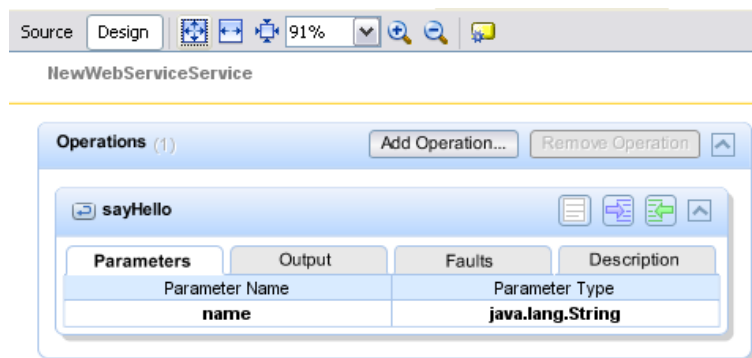


Abbildung 7: Design-Ansicht eines Web-Services

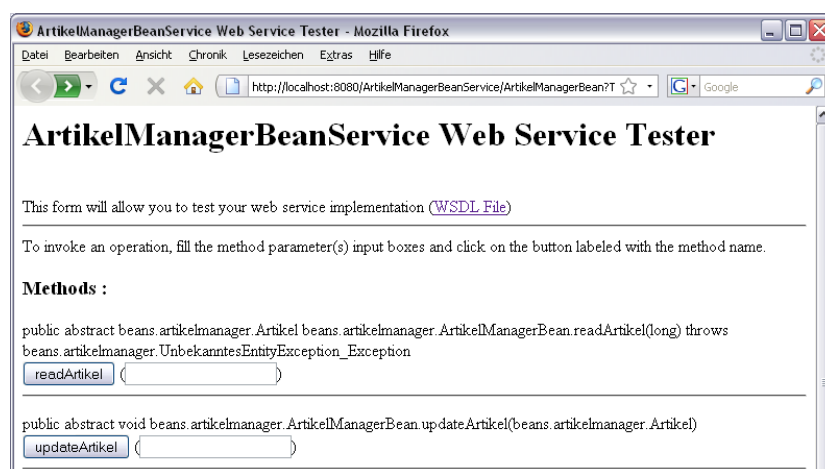


Abbildung 8: Das Webformular zum Testen des Webservices

Ausprobieren der Web-Services

Nachdem die Web-Services erstellt wurden und die Anwendung erneut *kompiliert* und *deployed* wurde, können die Web-Services über eine Web-Oberfläche des Applikationsservers ausprobiert werden.



Hierbei ist wieder daran zu denken **Build, Deploy** und **Run** immer aus dem Kontext-Menue von **DSI1-ausgang** (**DSI1-ausgang**) aufzurufen.

Hierzu öffnet man im **DSI-ejb Projekt** (**DSI-ejb**) den Knoten **Web-Services** und öffnet mit einem Rechtsklick auf den entsprechenden Web-Service das Kontext-Menue. In diesem wählt man den Menüpunkt **Test Web Service** aus (siehe Abbildung 12).

Der Webbrowser wird daraufhin geöffnet und bietet ein Formular an, mittels dem Web-Service-Operationen (welche keine komplexen Datentypen als Eingabe-Parameter besitzen) ausgeführt wer-

den können (siehe Abbildung 9). Nach der Ausführung bekommt man die gesendeten SOAP-Nachrichten angezeigt.

Auf der Webseite zum Testen der Web-Services befindet sich auch ein Link zur Anzeige der automatisch generierten **WSDL-Datei**.

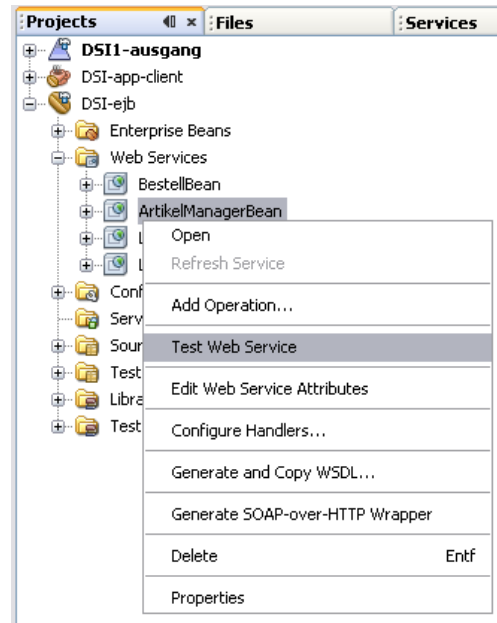


Abbildung 9: Testen des Web-Services

Stellen Sie sicher, dass Sie für alle genannten *Beans* die Web-Services erstellt und getestet haben.

Wie bereits beschrieben, wurde bei der Nach-Bestellung eine Mindestmenge von 10 Exemplaren fest implementiert. Zur Verbesserung dieser Situation soll ein neuer Web-Service entwickelt werden, der eine Mindestlagermenge pro Artikel als Ergebnis zurück liefert.

In den folgenden Abschnitten wird beschrieben, wie Sie einen neuen Web-Service erstellen.

1.3 Erstellen eines neuen Webservices

Zusätzlich zu den EJB-Methoden, welche nun als Web-Service-Operationen bereitgestellt wurden, soll ein weiterer Web-Service von Grund auf erstellt werden. Dieser Schritt zeigt, auf welche Weise Web-Services ohne die Nutzung von *Enterprise Java Beans* erstellt werden können. Hierzu muss ein eigenes Projekt angelegt werden.

Der zu erstellende Web-Service soll die Funktion bereitstellen, zu einer übermittelten Artikel-Id die Mindestmenge der nachzubestellenden Artikel zurückzugeben. Die Mindestmenge war zuvor in der Klasse *AuftragsAbwicklungBean* auf den Wert 10 festgelegt. Zur Ermittlung der Mindestmenge soll es genügen, eine einfache *If-Abfrage* zu verwenden (d.h. es muss keine DB-Anbindung entwickelt werden). Alle Artikel mit einer Artikel-Id < 5 bekommen die Mindestmenge 100 zugewiesen, der Rest die Mindestmenge 20.

1.3.1 Anlegen des Projektes

Für den Webservice wird mittels **File** → **New Project** ein Projekt des Typs **Java Web** → **Web Application** erstellt. Als Projekt-Name wird *DSI-Andere-Webservices* vorgeschlagen. Es sollte im selben Verzeichnis wie die anderen Projekte angelegt werden (**D:\DSI\Projekte\DSI-Praktikum-1\ausgang**) (siehe Abbildung 10).

Im **Server and Settings** Dialog des Wizards soll es zur bestehenden Enterprise-Application *DSI-1-ausgang* hinzugefügt werden. Es werden keine *Frameworks* benötigt.

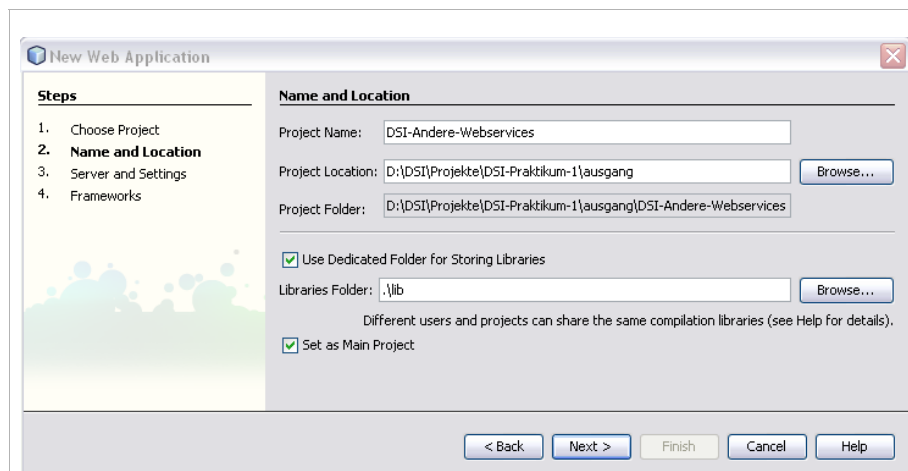


Abbildung 10: Anlegen des Projektes für den Webservice - 1

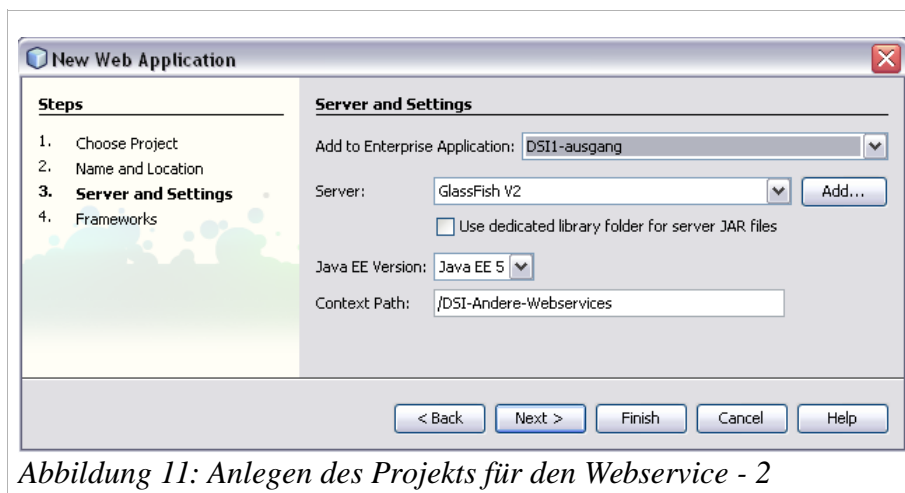


Abbildung 11: Anlegen des Projekts für den Webservice - 2

1.3.2 Erstellen des Web-Service

Standardmäßig wird ein Projekt erzeugt, das im Browser eine „Hello World“-Seite präsentiert. Um das Skelett des Web-Services zu generieren, genügt die Auswahl des Punktes **New** → **Web Service** aus dem Kontext-Menue des Web Application-Projekts **DSI-Andere-Webservices** (siehe Abbildung 12).

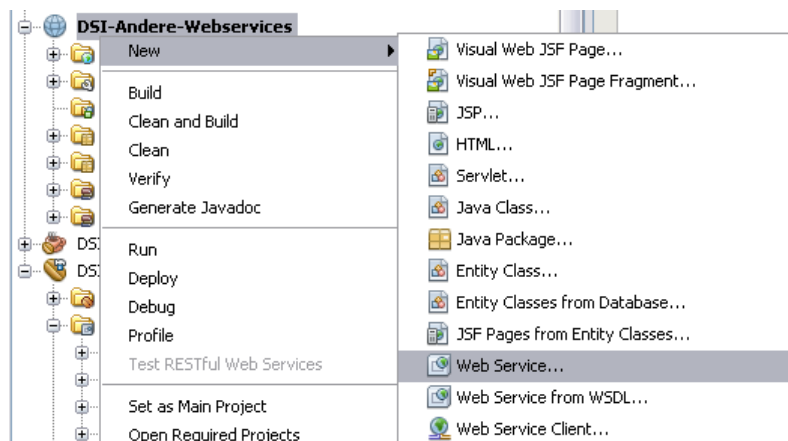


Abbildung 12: Erstellen des Web-Services-1

Im folgenden Dialog wird als Web-Service-Name **Mindestmenge** und als Package-Name **ws** eingegeben. Die entsprechende Java-Klasse wird erzeugt und kann nun im Projektbaum unter **Source Packages** zum Bearbeiten geöffnet werden (siehe Abbildung 14).

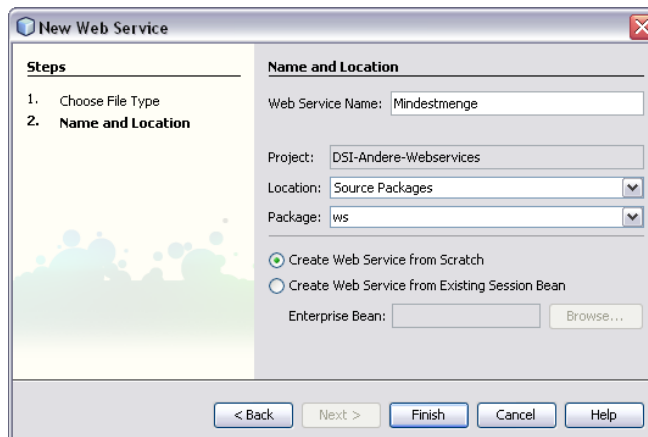


Abbildung 13: Erstellen des Webservices - 2

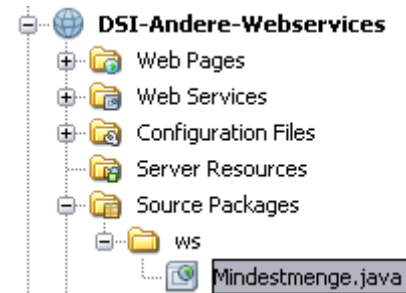


Abbildung 14: Erstellen des Web-Services - 3

Der Web-Service wird als normale Klassen-Methode implementiert, welche mit Annotationen versehen wird (siehe Abbildung 15).

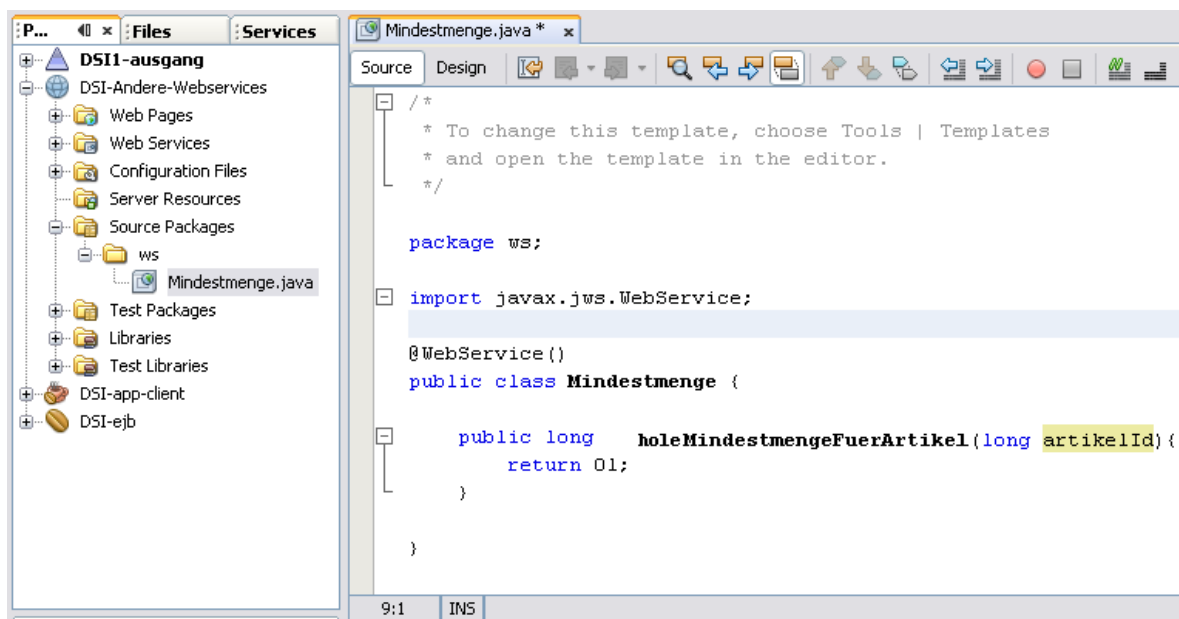



Abbildung 15: Implementieren des Web-Service (Ausschnitt – entspricht nicht der vollständigen Implementierung).

Anschließend können Sie die geforderte Funktionalität der Web-Service-Operation **holeMindestmengeFuerArtikel()** implementieren und nach *Build/Deployment* testen.



Auch Hierbei ist wieder daran zu denken **Build, Deploy** und **Run** immer aus dem Kontext-Menue von **DSI1-ausgang** ( **DSI1-ausgang**) aufzurufen.

1.3.3 Konsumieren der Webservices

Der nächste und letzte Schritt ist das Konsumieren der Web-Services. Hierfür steht ein Client bereit, welcher eine vereinfachte Version der GUI des EJB-Clients besitzt. Durch den Aufruf der Web-Services soll der Bestellvorgang im Client implementiert werden.

1.3.4 Öffnen des WS-Client-Projektes

Öffnen sie zunächst das Projekt (**File** → **Open Project**) mit dem Namen *WS-Client*. Es enthält eine grafische Oberfläche ähnlich der des EJB-Clients (ohne die Masken zum Anzeigen und Bearbeiten der Kunden- sowie Artikel-Daten).

Der Bestellprozess soll nun Client-seitig mittels **Aufruf der Web-Services** implementiert werden. Hierfür steht die Klasse *Bestellung* bereit. In dieser ist eine Methode *fuehreBestellungDurch* vorhanden, welche bereits von der GUI aufgerufen wird.

Die Anwendung lässt sich bereits ausführen, wobei die Methode *fuehreBestellungAus* eine Exception wirft, welche in einer Dialogbox angezeigt wird (siehe Abbildung 16).

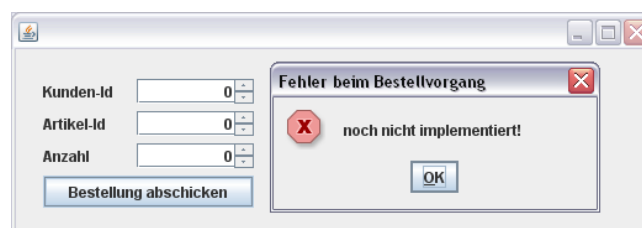


Abbildung 16: Noch wurde der Bestellvorgang nicht implementiert



Die folgenden Schritte betreffend der **JAX-WS-2.1** Bibliothek, können ignoriert werden, da diese bereits dem Projekt hinzugefügt wurde. Bei eigenen Experimenten sollte jedoch nicht vergessen werden, die Bibliothek hinzuzufügen.

Bevor es daran geht, den Bestellvorgang zu implementieren, muss noch die *JAX-WS 2.1* Bibliothek zum Projekt hinzugefügt werden. Diese Bibliothek, welche die Verwendung von Web-Services auf XML-Basis ermöglicht, wird beim JDK zwar bereits mitgeliefert, allerdings in der Version 2.0. Der Quellcode, der für den Aufruf der Web-Services generiert wird, benötigt allerdings die Version 2.1 dieser Bibliothek.

Sofern diese Bibliothek fehlt, kann die sie mit einem Rechtsklick auf **Libraries** und der Auswahl von **Add Library** aus dem Kontext-Menue hinzugefügt werden. Falls sie sich in der Liste befindet muss sie (**JAX-WS 2.1**) ausgewählt werden, um mit einem Klick auf den **Add Library**-Button zum Projekt hinzugefügt zu werden. Falls sie nicht in der Liste vorhanden ist, muss sie zunächst über den Dialog der sich nach betätigen des **Import...**-Buttons öffnet, importiert werden (siehe Abbildung 17).

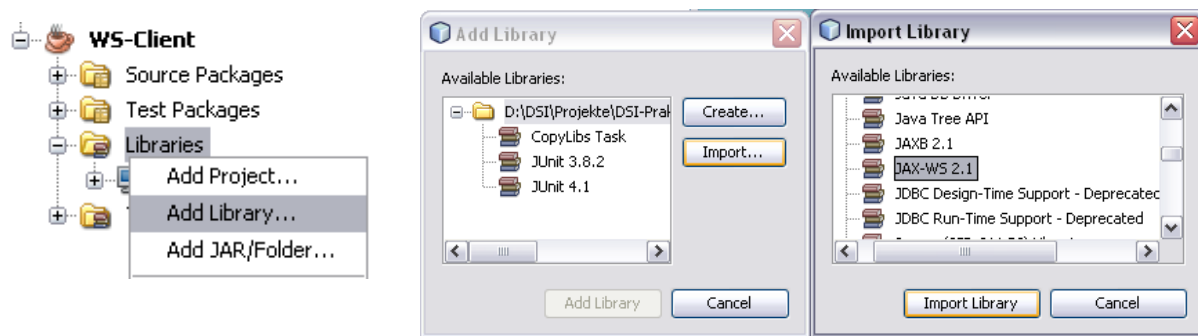


Abbildung 17: Importieren der JAX-WS 2.1 Bibliothek

1.3.5 Generieren der Web-Service-Clients

Um aus einem Java-Programm auf die Web-Services zugreifen zu können, müssen zunächst *Web-Service Clients* erstellt werden. Hierbei handelt es sich um *Wrapper*-Klassen, welche den Aufruf der Web-Services vor dem Entwickler verbergen. Für jeden Web-Service muss ein solcher Client generiert werden.

Der Web-Service Client wird aus der entsprechenden WSDL-Datei generiert und enthält auch sämtliche komplexen Datentypen, die als Parameter benötigt werden. Der Client wird mit **New** → **Web Service Client** generiert (siehe Abbildung 18).

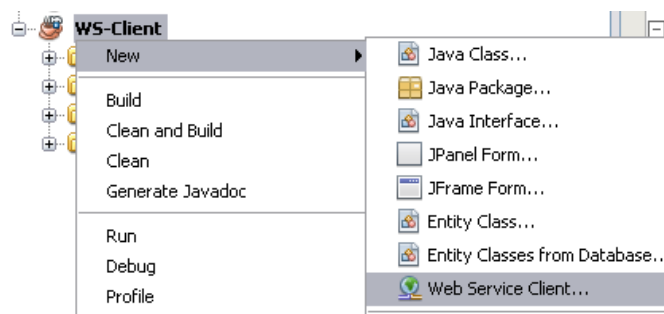


Abbildung 18: Erstellung eines Web-Service Clients

Im sich öffnenden Dialog muss die WSDL-Datei ausgewählt werden, aus der der Client generiert wird. Diese Datei kann zum einen über *Projects* gewählt werden, wo die WSDL-Dateien der geöffneten Projekte angeboten werden (siehe Abbildung 20), oder über **WSDL URL**, bei dem die URL zur WSDL-Datei direkt eingegeben werden kann (siehe Abbildung 19).

Die URL der WSDL-Datei eines Web-Services kann ermittelt werden, indem wie oben beschrieben der Web-Service getestet wird und auf der Formular-Seite dem Link *WSDL-File* gefolgt wird. Für jeden erstellten Web-Service muss dieser Schritt wiederholt werden.

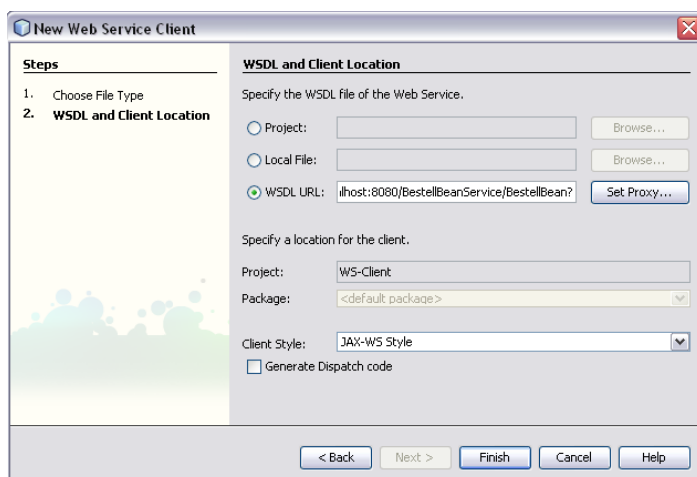


Abbildung 19: Dialog zur Erstellung eines Webservice Clients

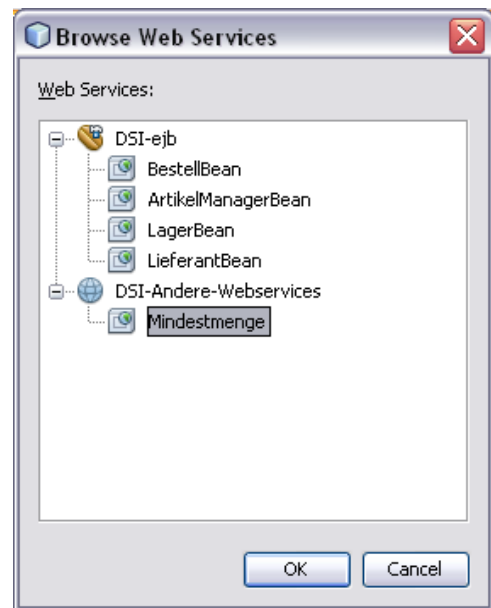


Abbildung 20: Project-Dialog zur Erstellung eines Webservice Clients

Der Quellcode der generierten Klassen lässt sich in der Files-Ansicht unter **build** → **generated** → **wsimport** → **client** betrachten (siehe Abbildung 21). Man kann leicht sehen, dass bereits für einen einfachen Web-Service ohne komplexe Datentypen eine Vielzahl von Klassen generiert werden.

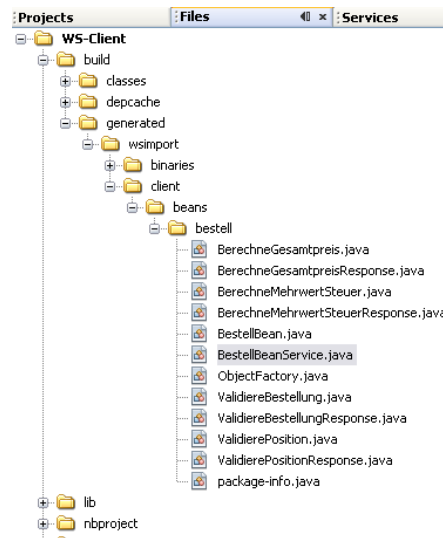


Abbildung 21: Die generierten Klassen des Bestell-Webservices

Erstellen Sie nun für sämtliche Web-Services jeweils einen *Web-Service Client*.

1.3.6 Aufruf der Webservices (Erstellung der WS-Client-Anwendung)

Wenn nun sämtliche Web-Service-Clients generiert wurden, kann es letztendlich an die Nutzung dieser Clients gehen. Hierfür wird nun die Klasse *Bestellung* (**Bestellung.java**) zum Bearbeiten geöffnet.

In dieser Klasse befindet sich zum Einen die Methode *fuehreBestellungDurch*, welche noch mit Funktionalität zu füllen ist. Des Weiteren sind eine Reihe von auskommentierten Methoden vorhanden. Die Methode *erstelleReport* liefert eine HTML-Darstellung der Bestelldaten zurück, sofern sie mit den richtigen Argumenten aufgerufen wird. Die anderen Methoden sind dazu gedacht, die Web-Service-Aufrufe zu *wrappen*.

Um einen Web-Service aufzurufen, muss an der entsprechenden Stelle im Quellcode mit einem Rechtsklick das Kontext-Menue geöffnet werden und **Web Service Client Resources** → **Call Web Service Operation** ausgewählt werden. Hier kann dann die gewünschte Webservice-Operation ausgewählt werden (siehe Abbildung 23). Da hierbei wieder einige Zeilen Code pro Aufruf generiert werden, sollten um die Übersicht zu behalten, die Web-Service-Aufrufe in den bereits vorgesehenen Methodenrumpfen stattfinden (siehe Abbildung 22).

Weiterhin empfiehlt es sich, dass eventuelle *Exceptions* nicht einfach (mittels des generierten Codes) gefangen werden, sondern weiter bis in die GUI geworfen werden, wo sie im Fehlerfall angezeigt werden (siehe Abbildung 24).

```
private boolean validiereBestellung(long artikelId, long anzahl, long kundenId) {  
  
    try { // Call Web Service Operation  
        beans.bestell.BestellBeanService service = new beans.bestell.BestellBeanService();  
        beans.bestell.BestellBean port = service.getBestellBeanPort();  
        // TODO initialize WS operation arguments here  
        long artikelId = 0L;  
        long anzahl = 0L;  
        long kundenId = 0L;  
        // TODO process result here  
        boolean result = port.validiereBestellung(artikelId, anzahl, kundenId);  
        System.out.println("Result = " + result);  
    } catch (Exception ex) {  
        // TODO handle custom exceptions here  
    }  
  
}
```

Abbildung 22: Der generierte Code zum Aufruf des Web-Services

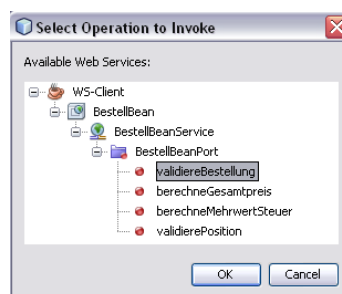


Abbildung 23: Auswahl der aufzurufenden Web-Service-Operation

```
private boolean validiereBestellung(long artikelId, long anzahl, long kundenId) throws Exception{  
    beans.bestell.BestellBeanService service = new beans.bestell.BestellBeanService();  
    beans.bestell.BestellBean port = service.getBestellBeanPort();  
    return port.validiereBestellung(artikelId, anzahl, kundenId);  
}
```

Abbildung 24: Diese Befehle genügen, um den Web-Service aufzurufen

Nun müssen in der **fuehreBestellungDurch**-Methode die Wrapper-Methoden aufgerufen werden⁴.

The screenshot shows a web application window with a title bar. Inside, there's a form with three input fields labeled 'Kunden-Id', 'Artikel-Id', and 'Anzahl', each containing the value '1'. Below these fields is a button labeled 'Bestellung abschicken'. To the right of the input fields, the text 'Bestellung' is displayed in bold. Below this, the order details are listed: 'Kunden-Nr: 1', 'Artikel', 'Artikel-Nr: 1 Bezeichnung: Reifen', 'Bestellmenge: 1', and 'Preis: 12,95€ + 2,46€ MwSt = 15,41€'.

Abbildung 25: Eine erfolgreich durchgeführte Bestellung

Wenn alles korrekt implementiert wurde, sollte bei der Ausführung der Anwendung beispielsweise folgende Ausgabe erscheinen (siehe Abbildung 25).

Vor Aufruf der WS-Client-Anwendung sollten vorher die entsprechenden Datensätze mit dem EJB-Client angelegt werden.

⁴ Das Aktivitätsdiagramm im Anhang verdeutlicht den zu implementierenden Kontrollfluss.

2 Abgabe

Pro Gruppe muss abgegeben werden:

- UML-Sequenzdiagramm
- Quellcode der Datei Bestellung.java

Bitte mailen Sie die entsprechenden Dateien unter Angabe der Gruppenmitglieder an f.buehler@fbi.h-da.de und g.turetschek@fbi.h-da.de.

I. Anhang

Der Bestellprozess - Fachliche Sicht

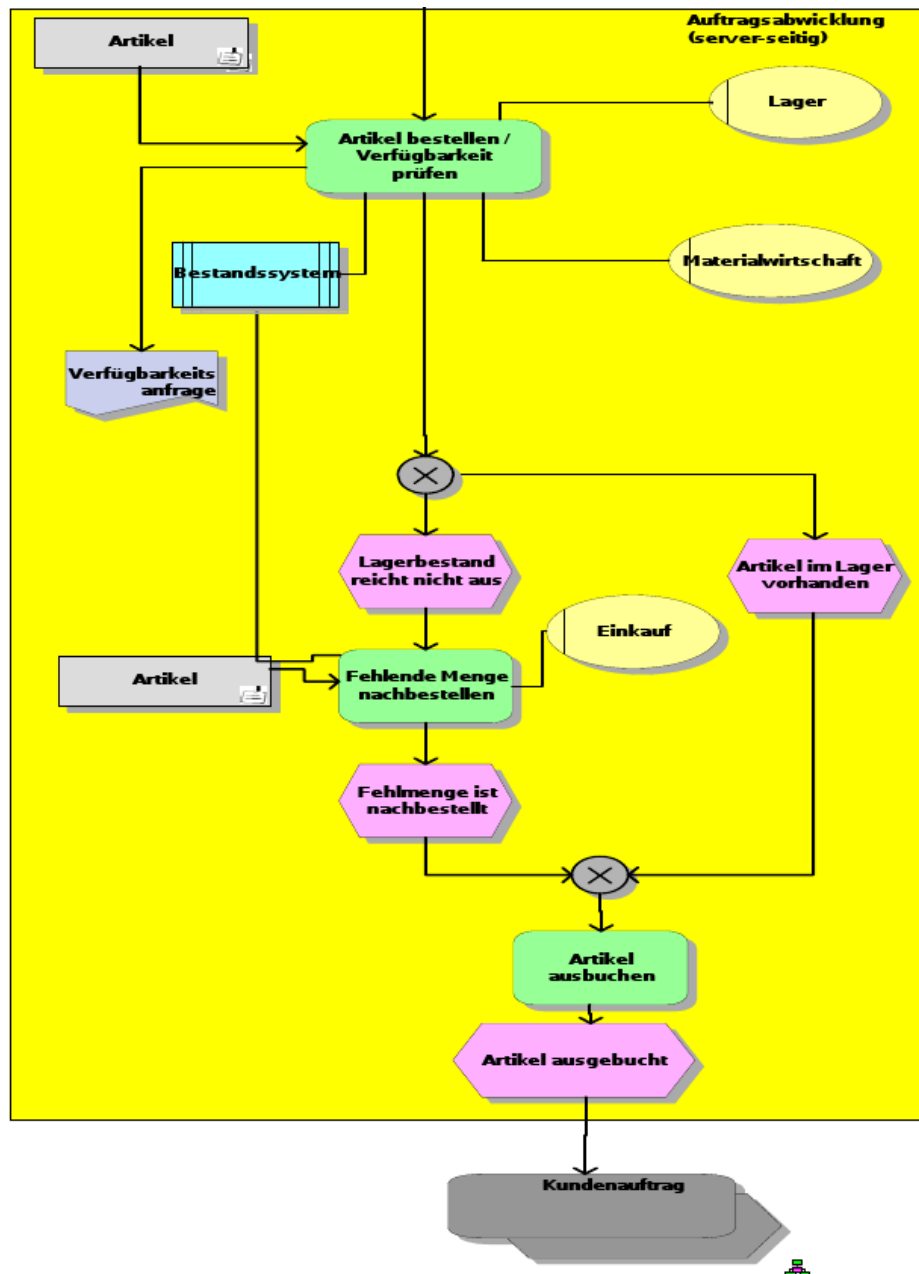


Abbildung 26: Bestellprozess aus fachlicher Sicht

Der Bestellprozess - Technische Sicht

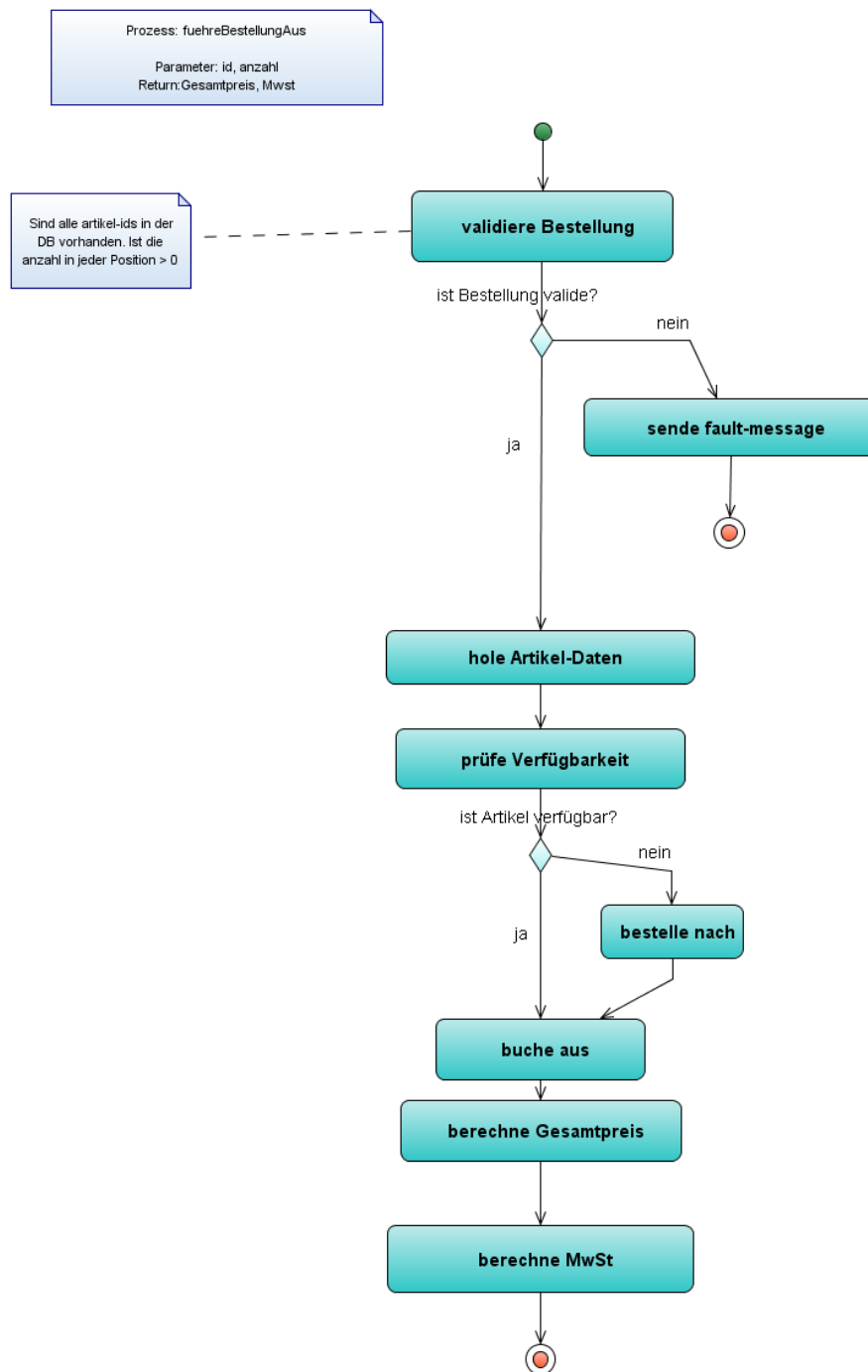


Abbildung 27: Bestellprozess aus technischer Sicht

Weitere Quellen

- Web Services Business Process Execution Language Version 2.0 – Primer. <http://www.oasis-open.org/committees/download.php/23974/wsbpel-v2.0-primer.pdf>. Stand 2008-06-04

- XML Schema Part 0: Primer. <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/>. Stand 2008-06-06

Häufige Fragen

1. **Wie lautet das Login für den Glassfish-Server?**
Name: admin
Passwort: adminadmin
2. **Ich finde dem Menue-Eintrag „Web Service“ nicht im Kontextmenue!**
Dann müssen sie den Menue-Punkt „Other“ wählen, und dort unter „Web Services“ den Punkt „Web Service“ auswählen. Der Menue-Punkt wird sodann ins Kontext-Menue hinzugefügt.
3. **Was sind Annotationen?**
Annotationen sind Meta-Daten die in den Quelltext eingebettet werden. Sie können vom Compiler oder auch zur Laufzeit ausgewertet werden.
Siehe: [http://de.wikipedia.org/wiki/Annotation_\(Java\)](http://de.wikipedia.org/wiki/Annotation_(Java)) Stand: 2008-07-14