Probleme bei der Nutzung der Bean Validation in JPA

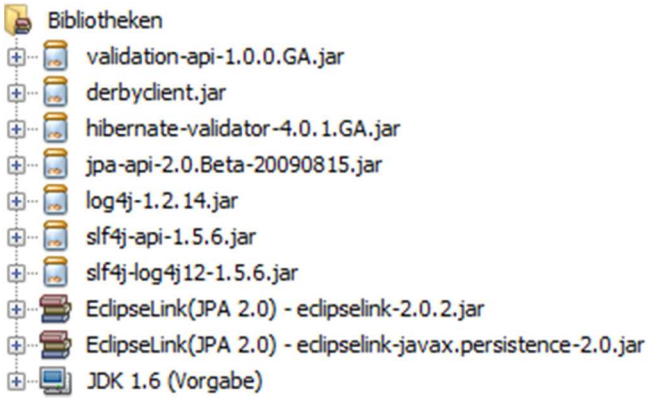
Die aktuelle Netbeans-Installation 6.9.1 bietet als JPA-Realisierung EclipseLink in der Version 2.0.2, die in der Zusammenarbeit mit der Referenzimplementierung der Bean Validation anscheinend Probleme macht. Das Problem soll mit folgendem Beispiel erläutert werden. Gegeben sei die folgende Klasse Produkt, die JPA und Bean Validation-Annotationen zusammen nutzt.

```
package entitaeten;

import java.io.Serializable;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Version;
import javax.validation.constraints.Min;
import javax.validation.constraints.Pattern;

@Entity
public class Produkt implements Serializable {

    @Id
    @GeneratedValue
    private int produktnr;

    @Pattern(regexp = "[A-Z].*", message = "Beginn mit Großbuchstaben")
    private String name;

    @Min(value = 1, message = "positiver Preis")
    private int verkaufspreis;

    @OneToMany(cascade = {CascadeType.PERSIST, CascadeType.MERGE})
    private Set<Bestellposten> posten = new HashSet<Bestellposten>();

    @Version
    private long version;

    public Produkt() {
    }

    public Produkt(String name, int verkaufspreis) {
        this.name = name;
        this.verkaufspreis = verkaufspreis;
    }

    @Override
    public String toString() {
        StringBuffer ergebnis = new StringBuffer(produktnr + ":" + name
            + " Preis:" + verkaufspreis + "\n");
        for (Bestellposten b : posten) {
            ergebnis.append(" " + b + "\n");
        }
        return ergebnis.toString();
    }

    public void addPosten(Bestellposten b) {
        posten.add(b);
    }

    public void deletePosten(Bestellposten b) {
        posten.remove(b);
    }

    // get- und set-Methoden folgen ...
}
```

Generell gibt es bei der Nutzung der Validierung zwei Ansätze. Der erste Ansatz nutzt die Validierungsmöglichkeiten direkt, d. h. die Validierungsmethoden werden aufgerufen und die erhaltenen Ergebnisse verarbeitet. Der Ansatz hat den großen Vorteil, dass Daten unmittelbar wenn sie entstehen, validiert werden können und so unerwünschte Daten nicht in das System „eindringen“. Der zweite Ansatz nutzt aus, dass JPA prüft, ob eine Bean Validation-Realisierung vorliegt und diese bei der Nutzung von persist(.) aufruft. Die Validierung wirft dann gegebenenfalls eine javax.validation.ConstraintViolationException, die dann analysiert werden kann und zum Rollback der Datenbank-Transaktion führen sollte. Beide Ansätze werden in folgendem Programm in der Methode rein() genutzt.

```
package Main;

import entitaeten.Produkt;
import java.util.Set;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.validation.ConstraintViolation;
import javax.validation.Validation;
import javax.validation.Validator;
import javax.validation.ValidatorFactory;

public class Minimal {

    private EntityManagerFactory emf = Persistence.createEntityManagerFactory("AufgabePU");
    private EntityManager em = emf.createEntityManager();

    public void analyse(Produkt o) {
        System.out.println("Analyse von " + o);
        ValidatorFactory factory =
                Validation.buildDefaultValidatorFactory();
        Validator validator = factory.getValidator();
        Set<ConstraintViolation<Produkt>> cv = validator.validate(o);
        for (ConstraintViolation<Produkt> c : cv) {
            System.out.println(" :: " + c.getMessage());
        }
    }

    public void rein() {
        Produkt p1 = new Produkt("ei", -23);
        analyse(p1); // erster Ansatz, Validierung selbst aufrufen
        try {
            System.out.println("TEST: ");
            em.getTransaction().begin();
            em.persist(p1); // zweiter Ansatz mit impliziter Nutzung
            em.getTransaction().commit();
        } catch (Exception e) {
            System.out.println(e); // weitere Behandlung fehlt
        } finally{
            if(em.isOpen())
                em.close();
            if(emf.isOpen());
                emf.close();
        }
    }

    public static void main(String[] args) {
        new Minimal().rein();
    }
}
```

Hat man dann, wie bisher beschrieben das Projekt eingerichtet und die Bibliotheken zur Bean Validation-Nutzung eingebunden, sieht das Bibliotheksverzeichnis vom Inhalt her wie folgt aus.

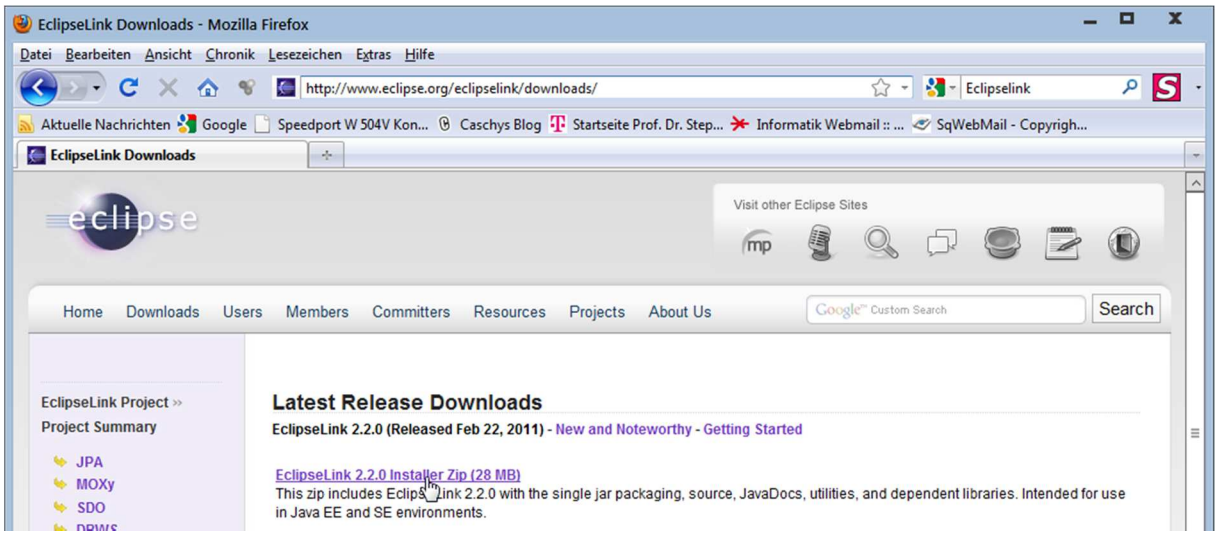



Lässt man jetzt das Programm laufen, erhält man neben vielen Logging-Informationen die folgende Ausgabe.

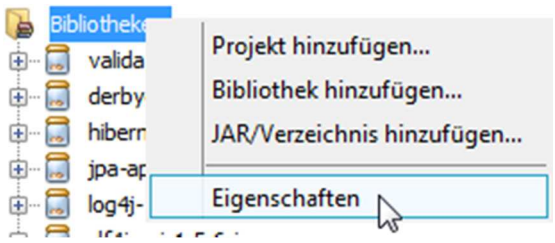
```
Analyse von 0:ei Preis:-23

 :: positiver Preis
 :: Beginn mit Großbuchstaben
TEST:
```

Man erkennt, dass die aufgerufene Validierung funktioniert, der bei der Persistierung erwartete Aufruf, aber nicht stattfindet. Ein Blick in die Datenbank zeigt auch, dass das Objekt persistiert wurde.

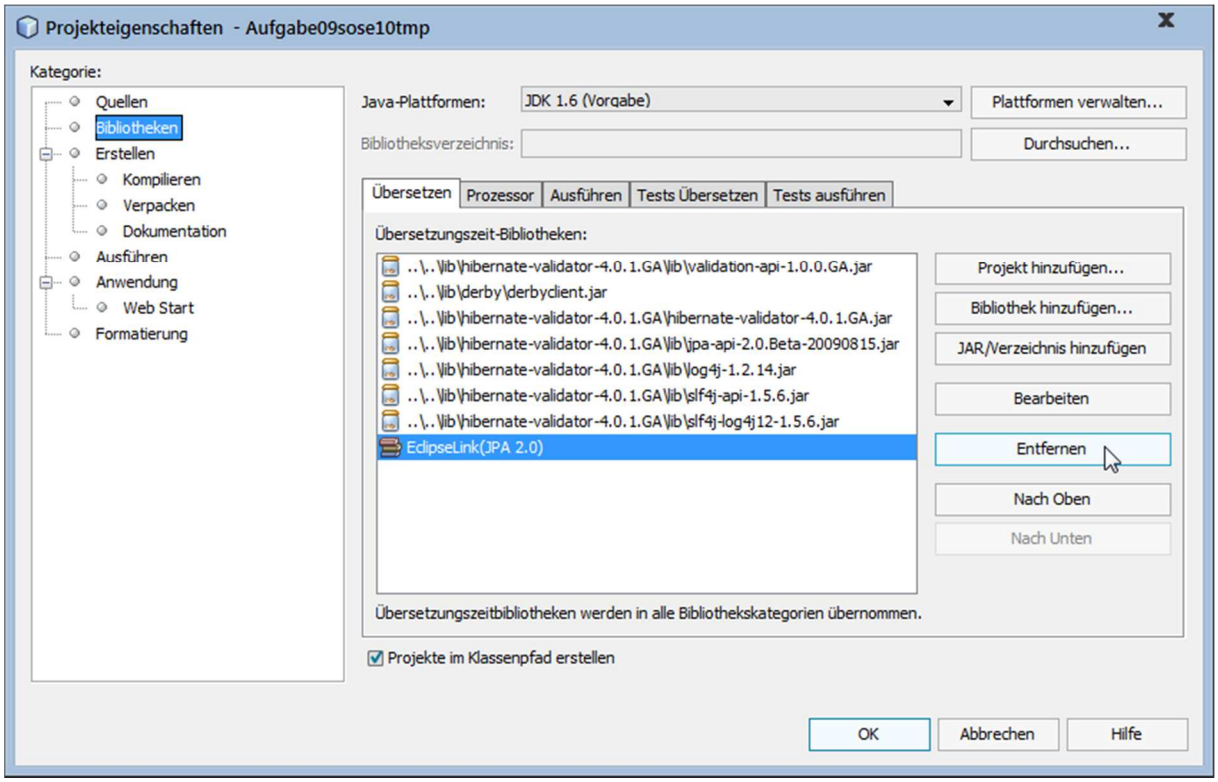

Das Problem scheint bei der genutzten EclipseLink-Version zu liegen, frühere Varianten liefen problemlos, zum Glück neuere Varianten auch. Aus diesem Grund wird jetzt eine neuere Variante von der Webseite http://www.eclipse.org/eclipselink/downloads/ geladen.



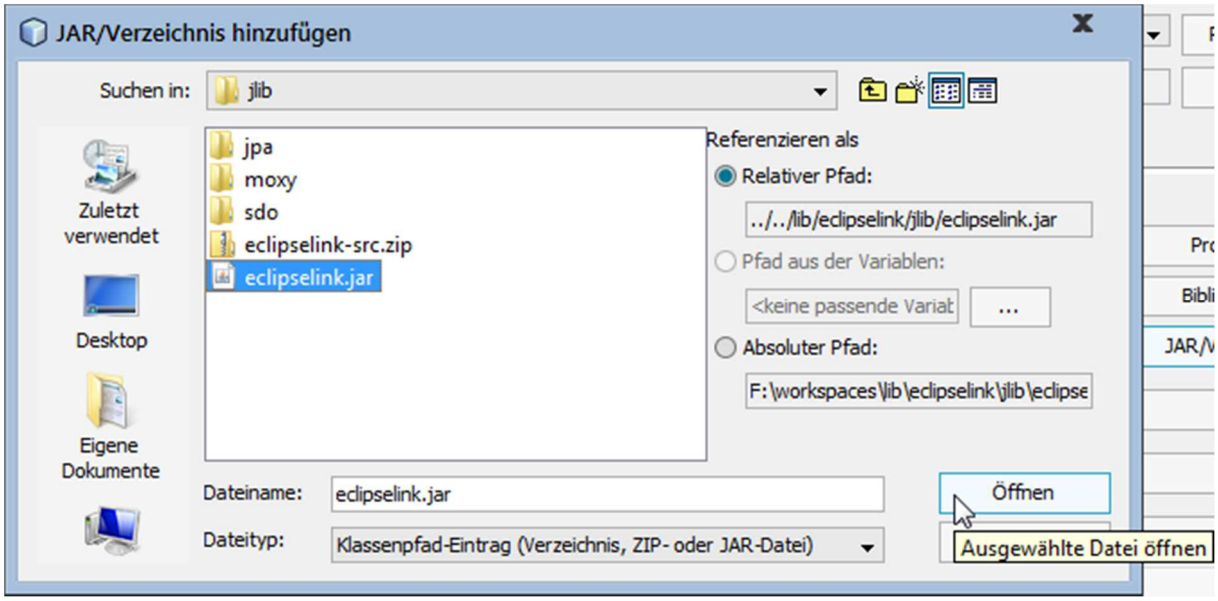

Die erhaltene Datei wird irgendwo, z. B. in einem lib-Verzeichnis in der Nähe der Netbeans-Projekte ausgepackt. Nun werden die alten EclipseLink-Dateien entfernt, wozu man z. B. einen Rechtsklick auf dem Bibliotheksverzeichnis machen kann und dann „Eigenschaften“ wählt.



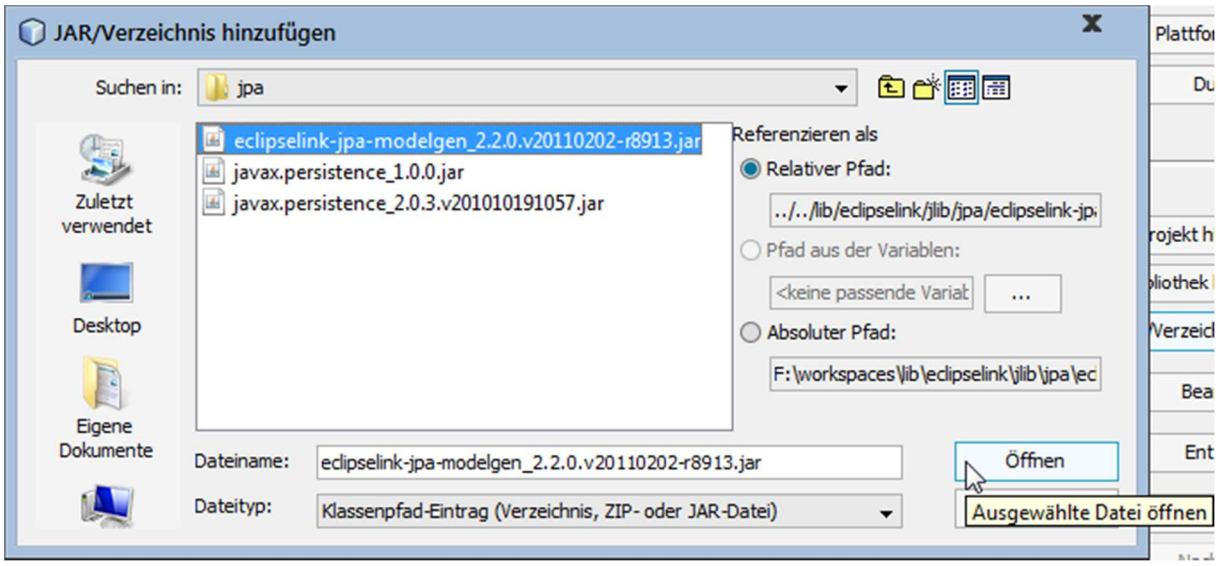

Hier klickt man jetzt den Eintrag „EclipseLink (JPA 2.0)“ und dann rechts auf „Entfernen“.




Danach wird der Knopf „JAR/Verzeichnis hinzufügen“ gedrückt und im ausgepackten neuen Eclipselink in das Verzeichnis jlib gewechselt, um dann eclipselink.jar auszuwählen.




Im nächsten Schritt wird eine zweite Bibliothek eclipselink-jpa-modelgen_2.2.0.v20110202-r8913.jar hinzugefügt, die sich im Verzeichnis jlib/jpa befindet.

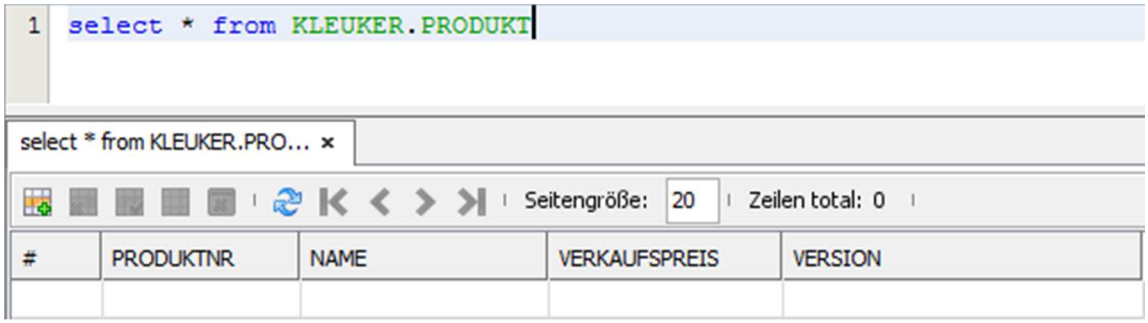



Hat man jetzt die EclipseLink-Dateien importiert, kann das Projekt wie folgt aussehen.


Führt man das Programm jetzt aus, erhält man als kleinen zentralen Unterschied die Information, dass ein Exception aufgetreten ist. Die Ausgabe sieht wie folgt aus:

```
Analyse von 0:ei Preis:-23

 :: positiver Preis
 :: Beginn mit Großbuchstaben
TEST:
javax.validation.ConstraintViolationException: Bean Validation constraint(s) violated while
executing Automatic Bean Validation on callback event:'prePersist'. Please refer to embedded
ConstraintViolations for details.
```

Eine Kontrolle des Datenbankinhalts zeigt auch, dass das Objekt nicht persistiert wurde.




Hinweis: Kommentare und alternative Lösungen sind willkommen.