Hibernate

Piotr Mazur

Katedra Mikroelektroniki i Technik Informatycznych

Łódź, 8 listopada 2010

HibernateTemplate

HibernateTemplate - klasa udostępniająca interfejs biblioteki Hibernate aplikacji.

Klasa zapewnia dodatkowo obsługę **deklaratywnej transakcyjności** Aby skorzystać z **HibernateTemplate** można

odziedzyczyć po klasie **HibernateDAOSupport** oraz skorzystać z metody **getHibernateTemplate**

Klasa posiada standardowy interfejs do biblioteki Hibernate (metody save, update, get, load), aby z niego skorzystać należy wstrzyknąć obiekt **sessionFactory**

Deklaratywna transakcyjność

Korzystając z deklaratywnej transakcyjności można przenieść odpowiedzialność za tworzenie nowych sesji oraz transakcji na stronę frameworku **Spring**

Aby skorzytać z deklaratywnej transakcyjności niezbędne jest zdefiniowanie managera transakcji

```
<bean id="transactionManager"
    class="org.springframework.orm.hibernate3.HibernateTransactionManager">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
```

Do określenia transakcyjności danej klasy lub metody można zastosować annotację **@Transactional**

@Transactional można przypisać do:

- **Metody** - transakcją zostanie objęta jedna metoda
- **Klasy** - każda metoda w danej klasie będzie oznaczona jako transakcyjna

Aby korzystać z transakcyjności ze wsparciem dla annotacji należy ją dodatkowo włączyć

```
<tx:annotation-driven transaction-manager="transactionManager"/>
```

Uwaga

Zadeklarowanie transakcyjności powoduje automatyczne utworzenie klasy typu proxy. Domyślnie wykorzystywane są do tego **JDK Proxy**
Oznacza to konieczność korzystania z interfejsów przy deklarowaniu klasy, lub konieczność zmiany systemu tworzenia proxy na inny (np: CGLIB)

Można wymusić korzystanie z innej biblioteki do utworzenia proxy dodając parametr do definicji **tx:annotation-driven**

proxy-target-class="true"

Jedną z konsekwencji istnienia proxy jest brak obsługi inwokacji własnych metod **(self-invocation)**

Obsługa transakcyjności

Domyslną akcją przy pomyślnym zakończeniu metody transakcyjnej jest zatwierdzenie transakcji.

Transakcja zostaje automatycznie wycofana jeżeli w trakcie wykonywania metody nastąpi wyjątek typu **RuntimeException**.

Pozostałe wyjątki nie mają wpływu na przebieg transakcji, jest ona zatwierdzana

Transakcyjnośc domyślnie jest zagnieżdzona, co oznacza że wykonanie jednej metody transakcyjnej w obrębie drugiej metody transakcyjnej powoduje wykonanie całego szeregu operacji w **jednej transakcji**

- Zachowanie to można kontrolować parametrem **propagation** annotacji **@Transactional**