Ćwiczenie 1:

Komponent StudentListService powinien listę studentów weryfikować z zapisem w bazie danych. W tym celu zdefiniuj połączenie do bazy danych w pliku konfiguracyjnym oraz wstrzyknij je do odpowiedniego komponentu. Metoda isStudent() powinna zostać zmodyfikowana, aby łączyła się do bazy danych. Kontrakt metody isStudent() nie może zostać zmieniony – tzn nie mogą zostać zwracane jakiekolwiek wyjątki.

Fragment SQL definiujący bazę danych:

```
create  table students (
   id bigint generated by default as identity,
  name varchar(255),
  primary  key (id)
)

insert intostudents values(1, 'Smith')
```

Instrukcja rozwiązania

1. Aby korzystać z bazy danych, należy dołączyć dwie bilbioletki: spring-jdbc oraz spring-tx, biblioteki obsługujące połączenie: commons-dbcp i commons-pool, a także sam sterownik bazy danych hsqldb.jar
2. W pliku konfiguracyjnym definiujemy komponent BasicDataSource, używający plikowej bazy danych typu embedded.
3. W komponencie StudentListService, za pomocą adnotacji @Autowired wstrzykujemy komponent DataSource i tworzymy komponent JdbcTemplate. Jako że kontrakt musi zostać zachowany, nie chcemy zwracać ani jawnie obsługiwać wyjątków SQL.
4. Modyfikujemy metodę isStudent, aby za pomocą JDBC pobrała z bazy studenta o zadanym imieniu. Jeżeli nie ma w bazie rekordu o takiej metoda zwraca false.

Rozwiązanie

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
      destroy-method="close">
      <property name="driverClassName" value="org.hsqldb.jdbcDriver" />
      <property name="url"
            value ="jdbc:hsqldb:file:studentdb" />
      <property name="username" value="sa" />
      <property name="password" value="" />
</bean>
```

```
@Component
public  class StudentListService {

	private JdbcTemplate jdbc;

	@Autowired
	public void setJdbcTemplate(DataSource ds) {
		jdbc = new JdbcTemplate(ds);
	}

	public boolean isStudent(String name) {
		boolean isStudent = false;

		SqlRowSet rowSet = jdbc.
			queryForRowSet("SELECT * FROM STUDENTS WHERE name=?", name);

		if (rowSet.next()) {
			isStudent = true;
		}

		return isStudent;
	}
}
```

Ćwiczenie 2:

Zmodyfikuj powyższe ćwiczenia, aby zamiast JdbcTemplate użyć Hibernate.

Instrukcja rozwiązania

1. Dodaj do projektu biblioteki: spring-orm, hibernate (hibernate-core, hibernate-jpa-2.0-api, hibernate-annotations, hibernate-commons-annotations), jta, dom4j, slf4j-api, slf4j-jcl, commons-collections oraz javassist
2. W pliku konfiguracyjnym inicjujemy SesstionFactory
3. Dadaj klasę modelu Student, będącą encją w rozumieniu Hibernate'a
4. Zmodyfikuj komponent StudentListService aby korzystał z Hibernatea

Rozwiązanie

```
<bean id="mySessionFactory"
class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
	<property name="dataSource" ref="dataSource" />
	<property name="packagesToScan" value="pl.devcastzone.spring.model" />
	<property name="hibernateProperties">
		<value>
			hibernate.dialect=org.hibernate.dialect.HSQLDialect
		</value>
	</property>
</bean>
```

```
@Entity
@Table (name = "students")
public  class Student {

	@Id
	private long id;

	private String name;

	public long getId() {
		return id;
	}

	public void setId(long id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

}
```

```
@Component
public  class StudentListService {

	@Resource
	private SessionFactory sessionFactory;

	public boolean isStudent(String name) {
		boolean isStudent = false;

		List<?> list = sessionFactory.openSession()
			.createCriteria(Student.class)
			.add(Restrictions.eq("name", name))
			.list();

		if (list.size() > 0) {
			isStudent = true;
		}

		return isStudent;
	}

}
```