Use JSONB with Hibernate

1. Hibernate does not support JSONB data type

Hibernate does not support PostgreSQL's proprietary *JSONB* data type. But you can change that with a *UserType*.

2. Implement a UserType

You need to implement the UserType interface and to define the SQL type and the Java class of your mapping.

```
public class MyJsonType implements UserType {


    @Override
    public int[] sqlTypes() {
        return new int[]{Types.JAVA_OBJECT};
    }

    @Override
    public Class<MyJson> returnedClass() {
        return MyJson.class;
    }


    …
}
```

Then you need to implement the mapping of the Java object to the JSON document in the *nullSafeSet* method.

```
public class MyJsonType implements UserType {


    @Override
    public void nullSafeSet(PreparedStatement ps, Object value,
                            int idx, SessionImplementor session)
                            throws HibernateException, SQLException {
        if (value == null) {
            ps.setNull(idx, Types.OTHER);
            return;
        }
        try {
            final ObjectMapper mapper = new ObjectMapper();
            final StringWriter w = new StringWriter();
            mapper.writeValue(w, value);
            w.flush();
            ps.setObject(idx, w.toString(), Types.OTHER);
        } catch (final Exception ex) {
            throw new RuntimeException("Failed to convert Invoice to
String: " + ex.getMessage(), ex);
        }
    }
    …
}
```

The mapping of the JSON document into the Java object needs to be implemented in the *nullSafeGet* method.

```
public class MyJsonType implements UserType {


    @Override
    public Object nullSafeGet(ResultSet rs, String[] names,
                SessionImplementor session, final Object owner)
                throws HibernateException, SQLException {
        final String cellContent = rs.getString(names[0]);
        if (cellContent == null) {
            return null;
        }
        try {
            final ObjectMapper mapper = new ObjectMapper();
            return mapper.readValue(cellContent.getBytes("UTF-8"),
                                                returnedClass());
        } catch (final Exception ex) {
            throw new RuntimeException(
                "Failed to convert String to Invoice: "
                + ex.getMessage(), ex);
        }
    }


    …
}
```

3. Register the *UserType*

You need to register the *UserType* with a *@TypeDef* annotation.

```
@org.hibernate.annotations.TypeDef(
        name = "MyJsonType",
        typeClass = MyJsonType.class)

package org.thoughts.on.java.model;
```

Then you can reference it with a *@Type* annotation in the entity mapping.

```
@Entity
public class MyEntity {

        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        @Column(name = "id", updatable = false, nullable = false)
        private Long id;

        @Column
        @Type(type = "MyJsonType")
        private MyJson jsonProperty;

        ...

}
```

4. Register the column type in the Hibernate dialect

The jsonb column type is not supported by PostgreSQL's Hibernate dialect. You can extend an existing dialect and register the column type in the constructor.

```
public class MyPostgreSQL94Dialect extends PostgreSQL94Dialect {


        public MyPostgreSQL94Dialect() {
                this.registerColumnType(Types.JAVA_OBJECT, "jsonb");
        }
}
```