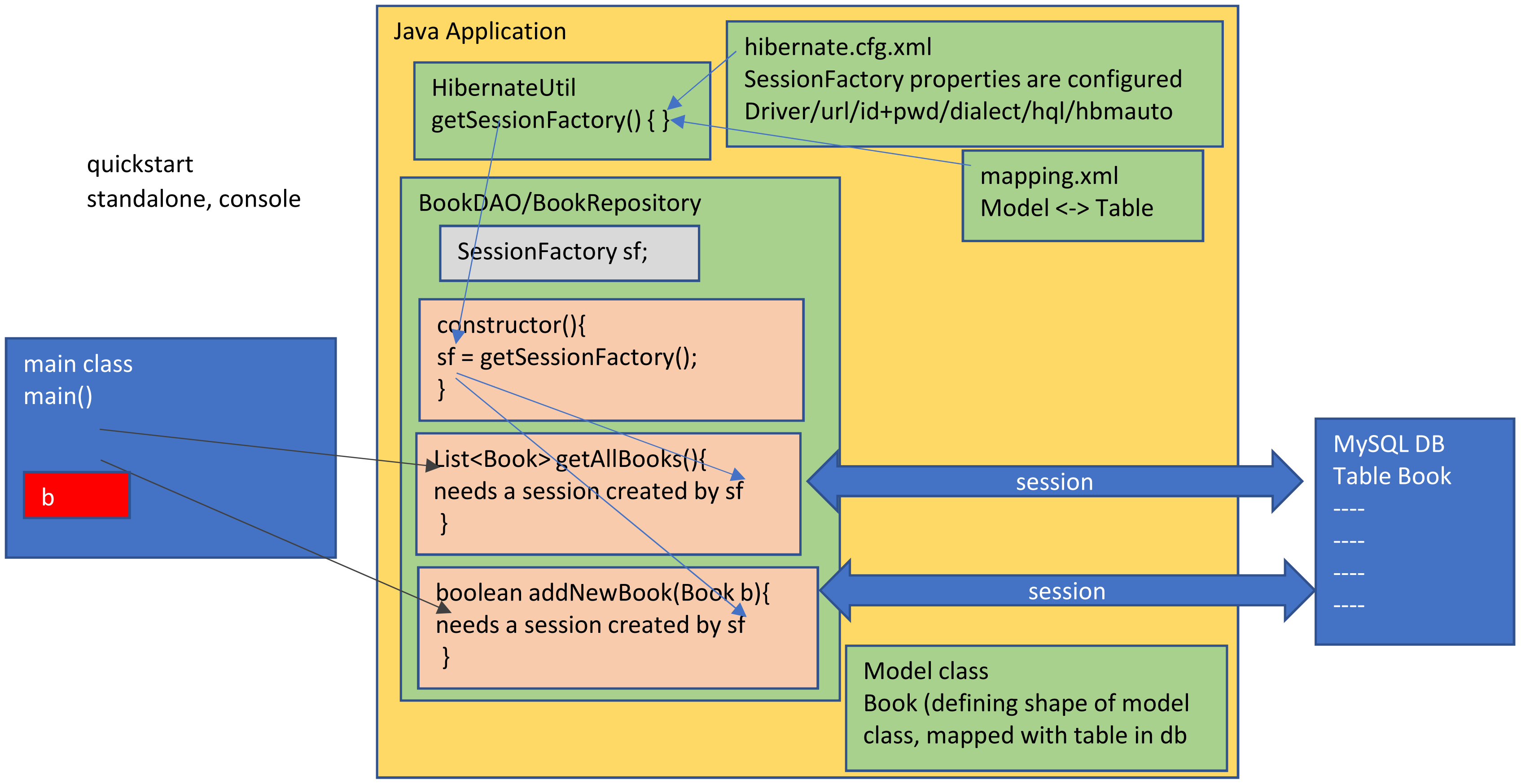
quickstart
standalone, console

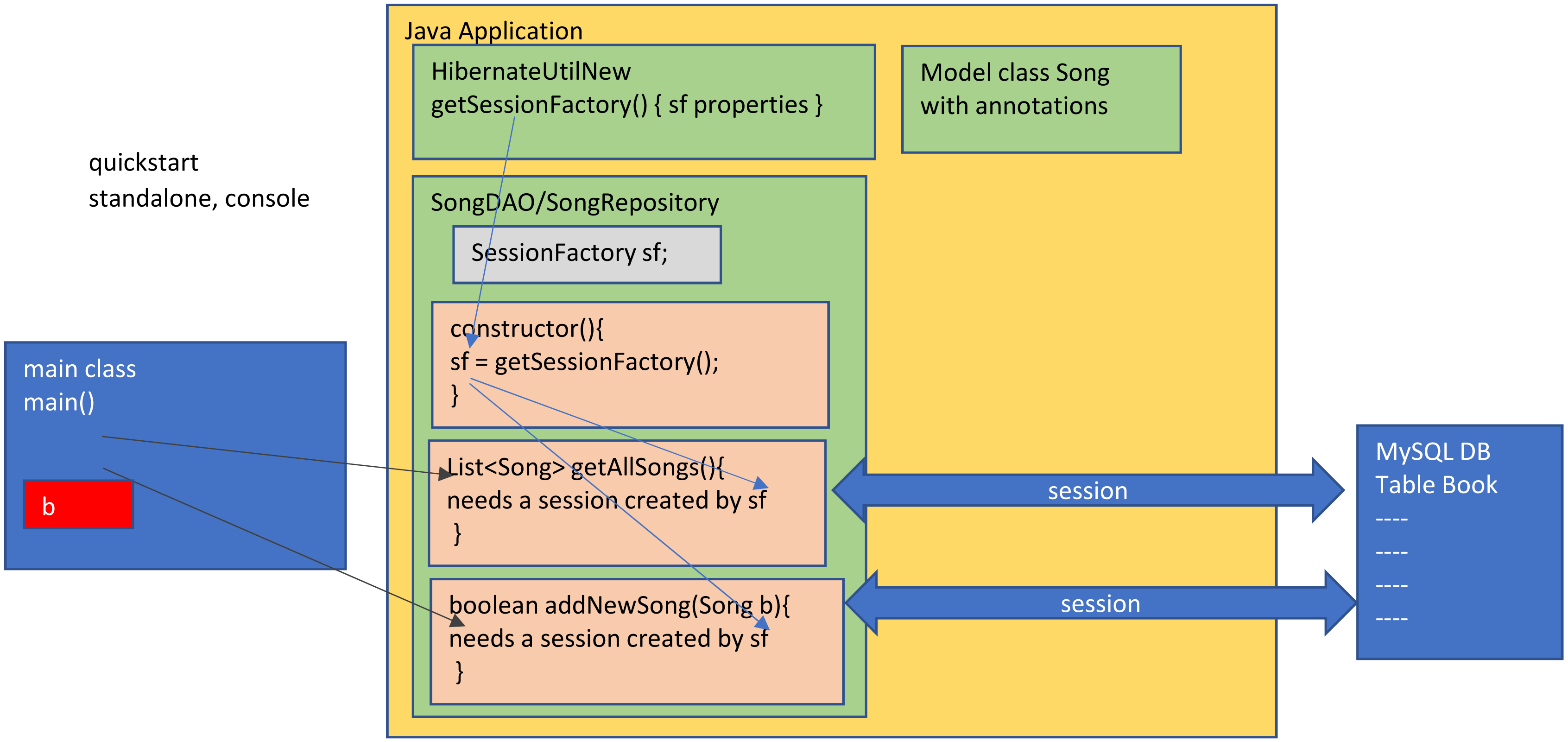

Java Application
hibernate.cfg.xml
SessionFactory properties are configured
Driver/url/id+pwd/dialect/hql/hbmauto
HibernateUtil
getSessionFactory() { }
mapping.xml
Model <-> Table
BookDAO/BookRepository
SessionFactory sf;
constructor(){
sf = getSessionFactory();
}
List<Book> getAllBooks(){
needs a session created by sf
}
boolean addNewBook(Book b){
needs a session created by sf
}
main class
main()
b
session
session
MySQL DB
Table Book

Model class
Book (defining shape of model class, mapped with table in db

In the above example

SessionFactory configuration is done xml format
Model class mapping is done xml format
HibernateUtil.getSessionFactory() loading the configurations

quickstart
standalone, console



Changes
No xml files
sessionFactory configuration is done withing UTIL class method only
Model class is maped with table using annotations withing model

What is annotation
Used to provide additional info to compiler about
method/variable/param/class/interface/..
@Override

1 Create model class

class <-> table

which member variable of class representing primary key table

@Entity

Used at class level

Used to map java class with DB table

@Id

Used at member variable level within entity class

Used to map with primary key in DB table

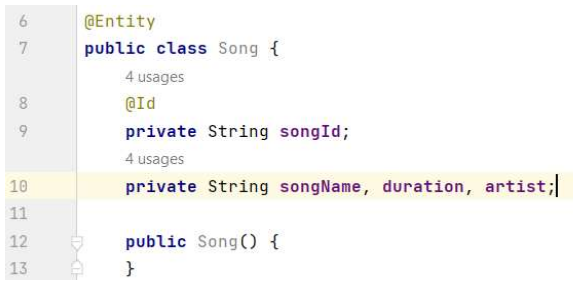
class Song

    songId

    songName, duration, artist

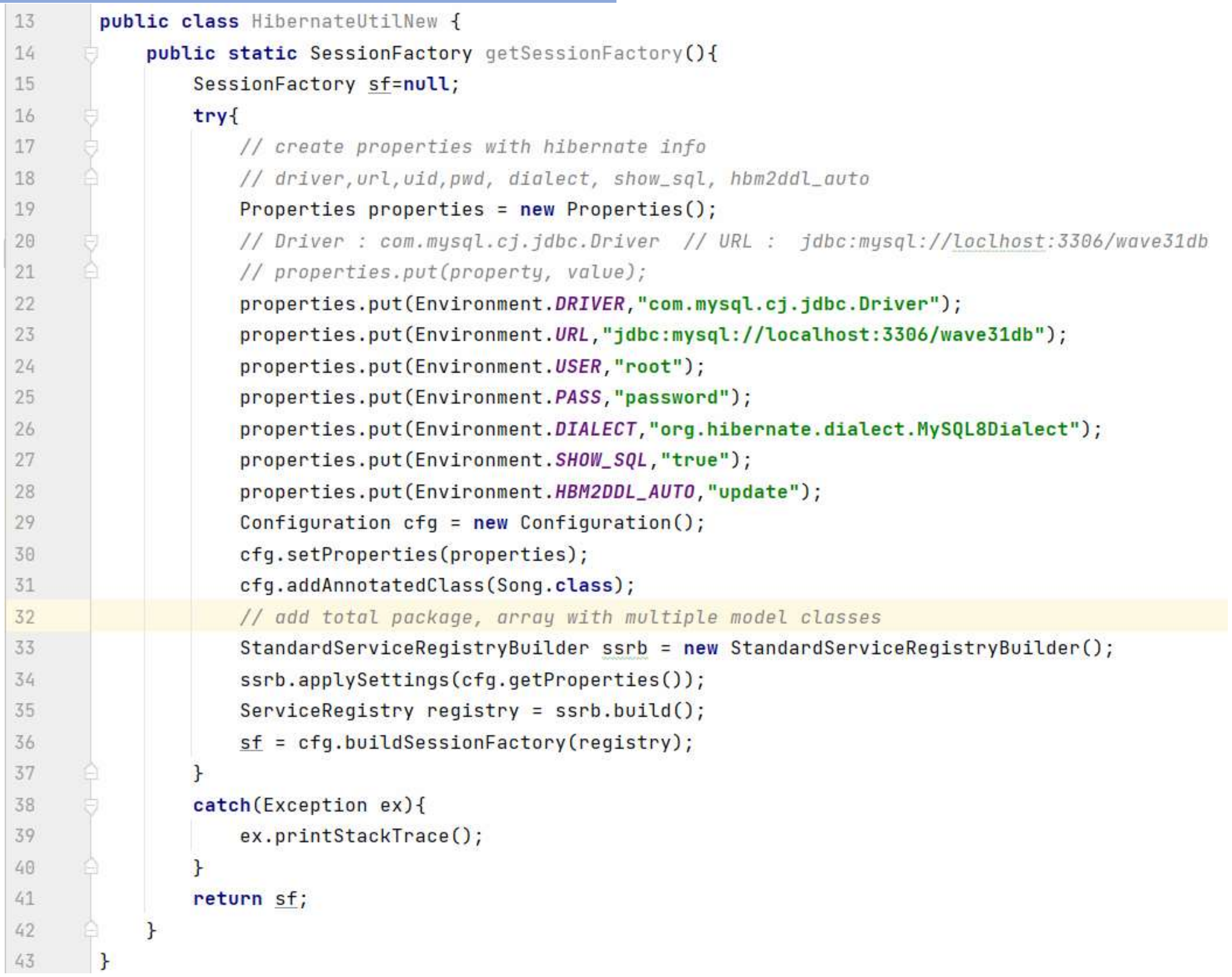
Here, Song class must be annotated with @Entity

songId variable must be annotated with @Id



```
@Entity
public class Song {
    4 usages
    @Id
    private String songId;
    4 usages
    private String songName, duration, artist;

    public Song() {
    }
```

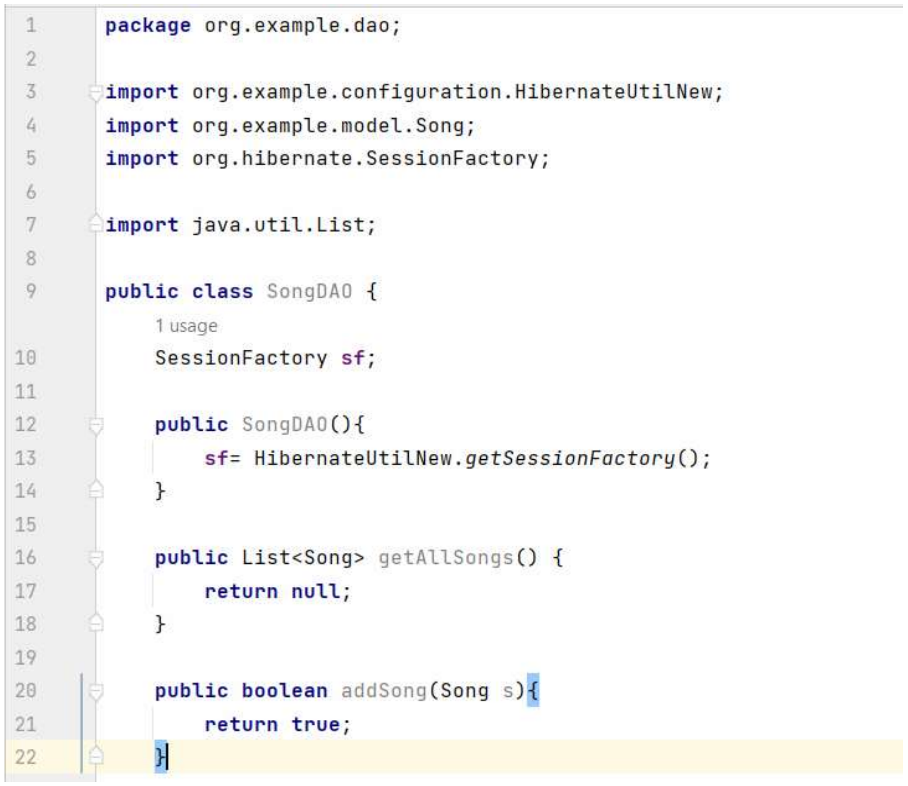
2 Define sessionFactory in UTIL class method



```java
public class HibernateUtilNew {
    public static SessionFactory getSessionFactory(){
        SessionFactory sf=null;
        try{
            // create properties with hibernate info
            // driver,url,uid,pwd, dialect, show_sql, hbm2ddl_auto
            Properties properties = new Properties();
            // Driver : com.mysql.cj.jdbc.Driver  // URL :  jdbc:mysql://loclhost:3306/wave31db
            // properties.put(property, value);
            properties.put(Environment.DRIVER,"com.mysql.cj.jdbc.Driver");
            properties.put(Environment.URL,"jdbc:mysql://localhost:3306/wave31db");
            properties.put(Environment.USER,"root");
            properties.put(Environment.PASS,"password");
            properties.put(Environment.DIALECT,"org.hibernate.dialect.MySQL8Dialect");
            properties.put(Environment.SHOW_SQL,"true");
            properties.put(Environment.HBM2DDL_AUTO,"update");
            Configuration cfg = new Configuration();
            cfg.setProperties(properties);
            cfg.addAnnotatedClass(Song.class);
            // add total package, array with multiple model classes
            StandardServiceRegistryBuilder ssrb = new StandardServiceRegistryBuilder();
            ssrb.applySettings(cfg.getProperties());
            ServiceRegistry registry = ssrb.build();
            sf = cfg.buildSessionFactory(registry);
        }
        catch(Exception ex){
            ex.printStackTrace();
        }
        return sf;
    }
}
```

3 Define DAO class, get sf object

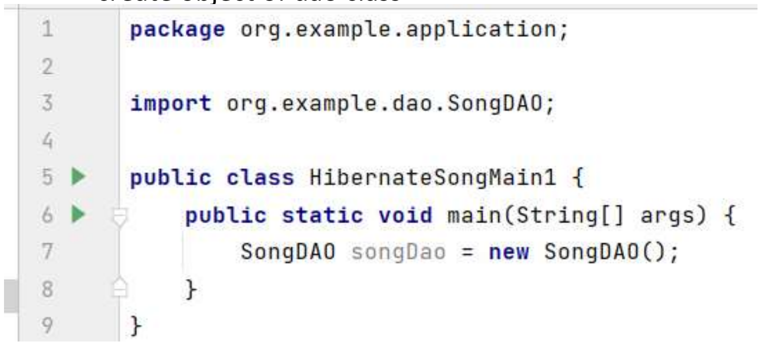
HibernatUtilNew.getSessionFactory()



```java
package org.example.dao;

import org.example.configuration.HibernateUtilNew;
import org.example.model.Song;
import org.hibernate.SessionFactory;

import java.util.List;

public class SongDAO {
    1 usage
    SessionFactory sf;

    public SongDAO(){
        sf= HibernateUtilNew.getSessionFactory();
    }

    public List<Song> getAllSongs() {
        return null;
    }

    public boolean addSong(Song s){
        return true;
    }
```

4 Define main class with main()

create object of dao class



```java
package org.example.application;

import org.example.dao.SongDAO;

public class HibernateSongMain1 {
    public static void main(String[] args) {
        SongDAO songDao = new SongDAO();
    }
}
```