

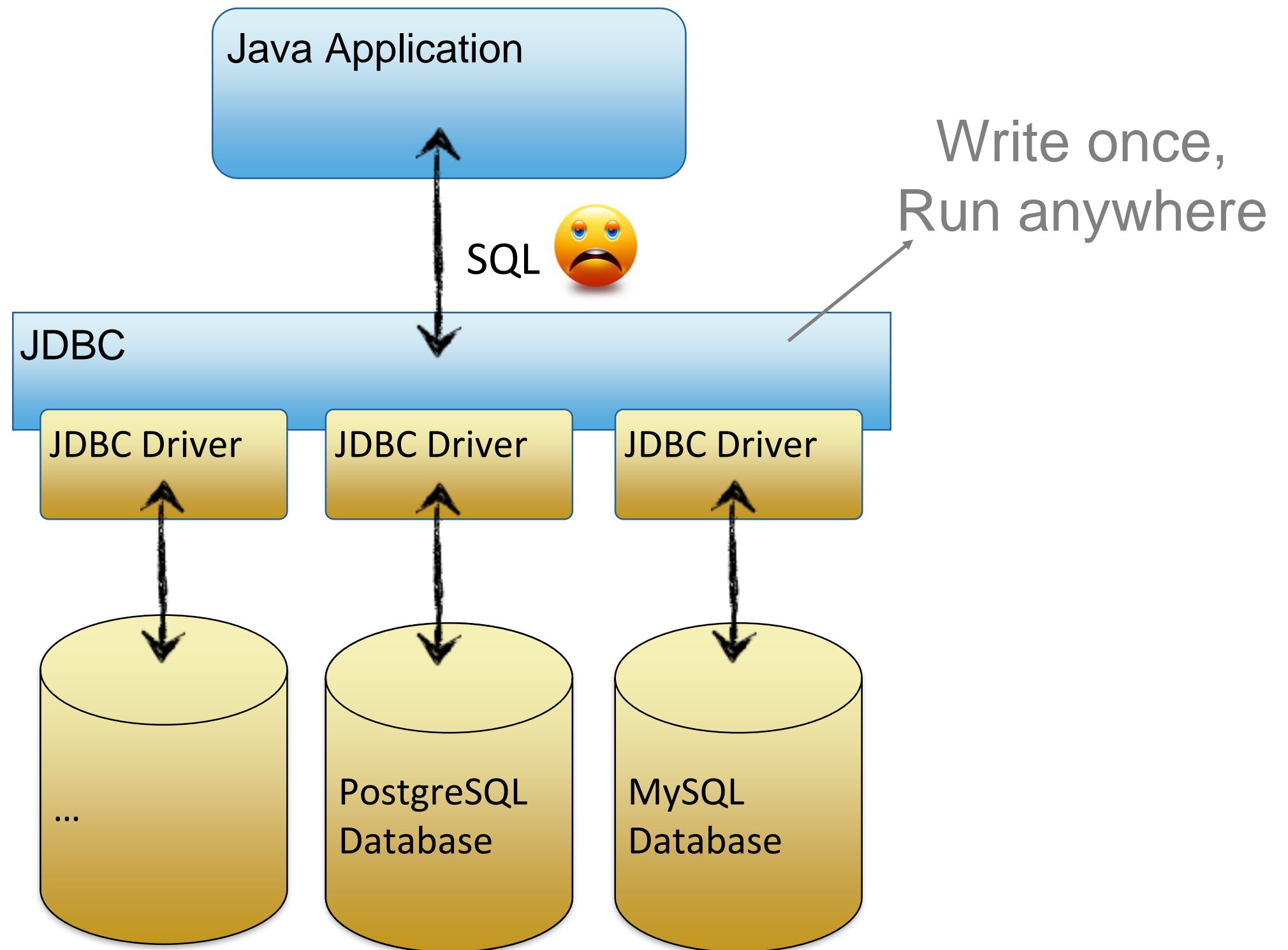
JPA

Intro



JDBC

JAVA DATABASE CONNECTIVITY



'OBJECT-RELATIONAL IMPEDANCE MISMATCH'

- Object graph # tabular format:
 - # Granularity
 - # Subtypes (inheritance)
 - # Identity
 - # Associations
 - # Data navigation



ORM

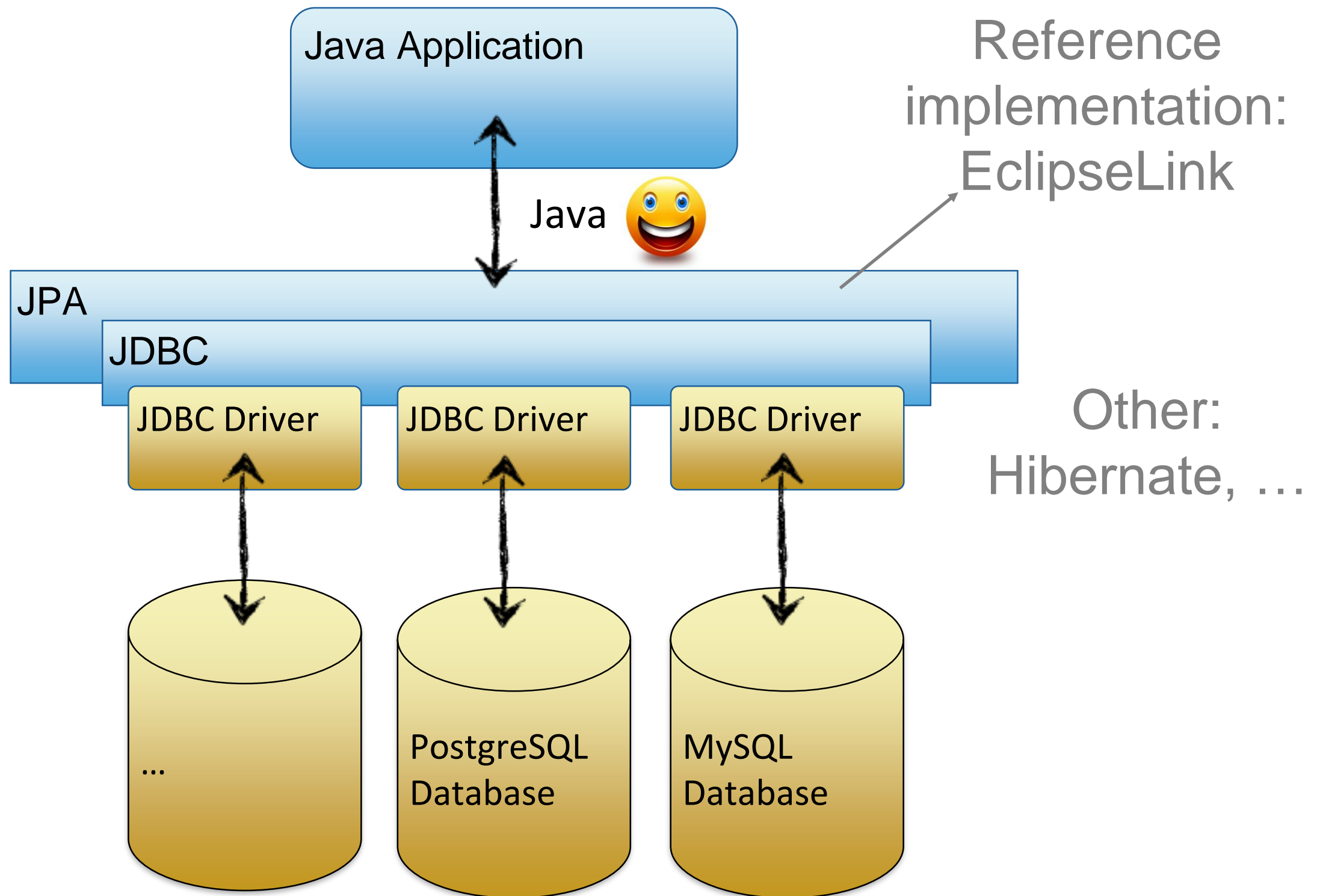
OBJECT RELATIONAL MAPPING

"... is a programming technique for converting data between incompatible type systems in object-oriented programming languages."

Wikipedia

JPA

JAVA PERSISTENCE API



javafx.persistence (Java(TM) x

docs.oracle.com/javaee/7/api/

AppsStartpaginaGoogle MapsWikipediaFotoosInformaticaKLHInfo»Other Bookmarks

OverviewPackageClassUseTreeDeprecatedIndexHelp

Prev PackageNext PackageFramesNo Frames

Package javax.persistence

Java Persistence is the API for the management for persistence and object/relational mapping.

See: Description

Interface Summary

Interface	Description
AttributeConverter <X,Y>	A class that implements this interface can be used to convert entity attribute state into database column representation and back again.
AttributeNode <T>	Represents an attribute node of an entity graph.
Cache	Interface used to interact with the second-level cache.
EntityGraph <T>	This type represents the root of an entity graph that will be used as a template to define the attribute nodes and boundaries of a graph of entities and entity relationships.
EntityManager	Interface used to interact with the persistence context.
EntityManagerFactory	Interface used to interact with the entity manager factory for the persistence unit.
EntityTransaction	Interface used to control transactions on resource-local entity managers.
Parameter <T>	Type for query parameter objects.
PersistenceUnitUtil	Utility interface between the application and the persistence provider managing the persistence unit.
PersistenceUtil	Utility interface between the application and the persistence provider(s).

javax.persistence

Interfaces

AttributeConverterAttributeNodeCacheEntityGraphEntityManagerEntityManagerFactoryEntityTransactionParameterPersistenceUnitUtilPersistenceUtilQueryStoredProcedureQuerySubgraphTupleTupleElementTypedQuery

Classes

Persistence



Model:

- ☐ Write database class
- ☐ Modify entity class
- ☐ Write persistence.xml
- ☐ Add libraries

☑ Write database class

```
public class CountryRelationalDatabase implements CountryDatabase{  
    ...  
  
    public void add(Country object) {  
        manager.getTransaction().begin();  
        manager.persist(object);  
        manager.getTransaction().commit();  
    }  
  
    public Country get(Long id) {  
        Country country = manager.find(Country.class, id);  
        return country;  
    }  
    ...  
}
```

→ Transaction for updates

→ Save Object

→ Retrieve Object


Write database class


```
public class CountryRelationalDatabase implements CountryDatabase{
    private EntityManagerFactory factory;
    private EntityManager manager;
    private String name;

    public void openConnection(String name){
        factory = Persistence.createEntityManagerFactory(name);
        manager = factory.createEntityManager();
    }

    ...

    public void closeConnection() throws DatabaseException {
        try {
            manager.close();
            factory.close();
        } catch (Exception e) {
            throw new DatabaseException(e.getMessage(), e);
        }
    }
}
```

 Does all the work

 Close!

- `entityManager.persist(object);`
- `entityManager.remove(object);`
- `entityManager.merge(object);`
- `entityManager.find(class, object);`
- `entityManager.createQuery(String);`

- `entityManager.getTransaction().begin();`
- `entityManager.getTransaction().commit();`
- `entityManager.flush();`



Model:

- ☒ Write database class
- ☐ Modify entity class
- ☐ Write persistence.xml
- ☐ Add libraries

Modify entity class

```
import javax.persistence.*;
```

```
@Entity
```

```
public class Country {
```

```
    @Id
```

```
    @GeneratedValue
```

```
    private long id;
```

```
    private String name;
```

```
    private int numberInhabitants;
```

 **Annotations**



Model:

- ☒ Write database class
- ☒ Modify entity class
- ☐ Write persistence.xml
- ☐ Add libraries

✓ Write persistence.xml

Configuration file



NOK for webapp!
TBD in 3TI



```
<persistence version="2.1"
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
    http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="TourismPU" transaction-type="RESOURCE_LOCAL">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>org.ucll.demo.domain.Country</class>
    <properties>
      <property name="javax.persistence.jdbc.url"
        value="jdbc:derby://localhost:1527/tourism"/>
      <property name="javax.persistence.jdbc.user" value="app"/>
      <property name="javax.persistence.jdbc.password" value="app"/>
      <property name="javax.persistence.jdbc.driver"
        value="org.apache.derby.jdbc.ClientDriver"/>
      <property name="javax.persistence.schema-generation.database.action"
        value="drop-and-create"/>
    </properties>
  </persistence-unit>
</persistence>
```



Model:

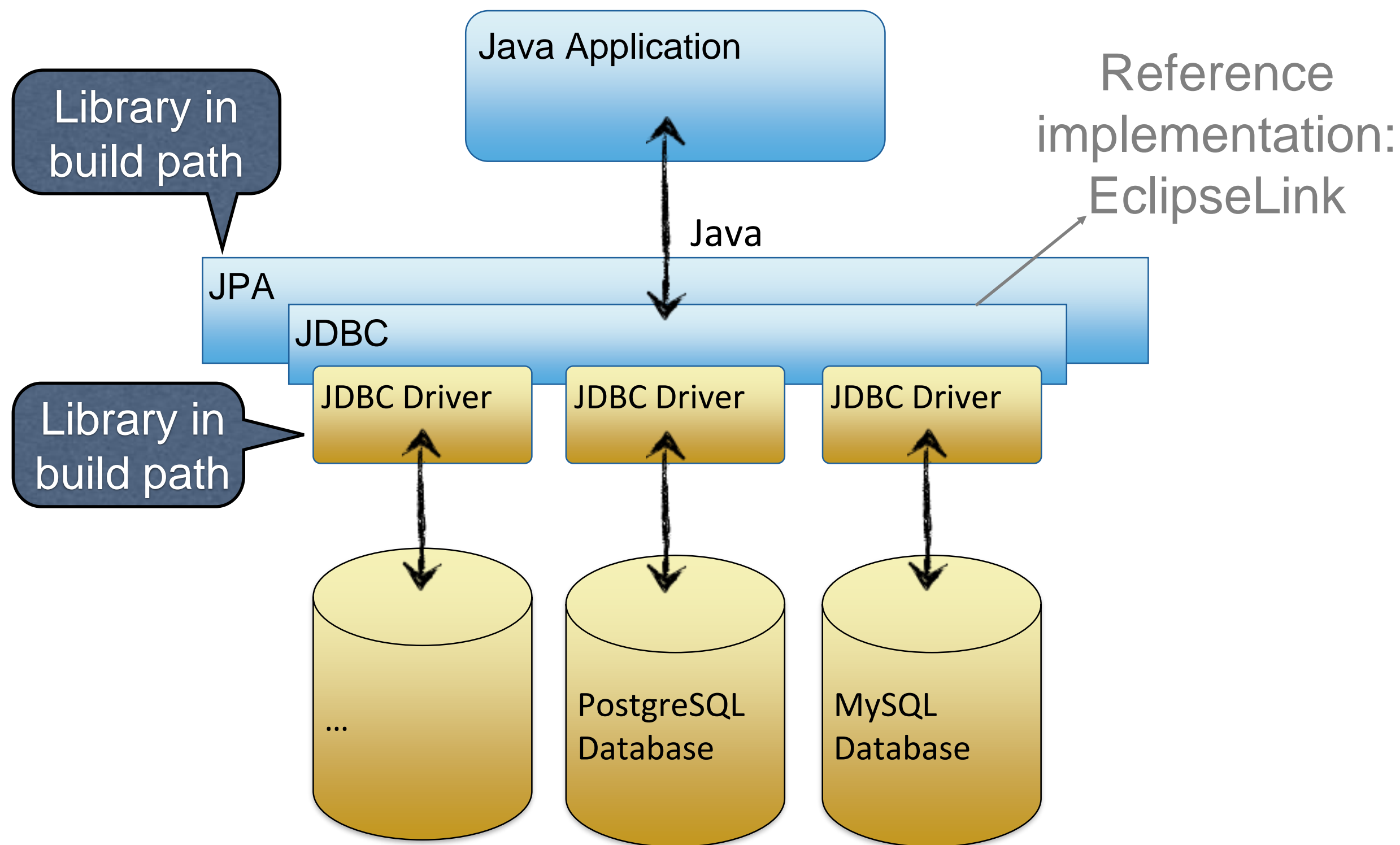
- ☒ Write database class
- ☒ Modify entity class
- ☒ Write persistence.xml
- ☐ Add libraries



Add libraries

JPA

JAVA PERSISTENCE API





Model:

- ☒ Write database class
- ☒ Modify entity class
- ☒ Write persistence.xml
- ☒ Add libraries



JAVABEAN

```
public class Country {  
    private String name;
```

```
    public Country() {  
    }
```

public no-arg constructor

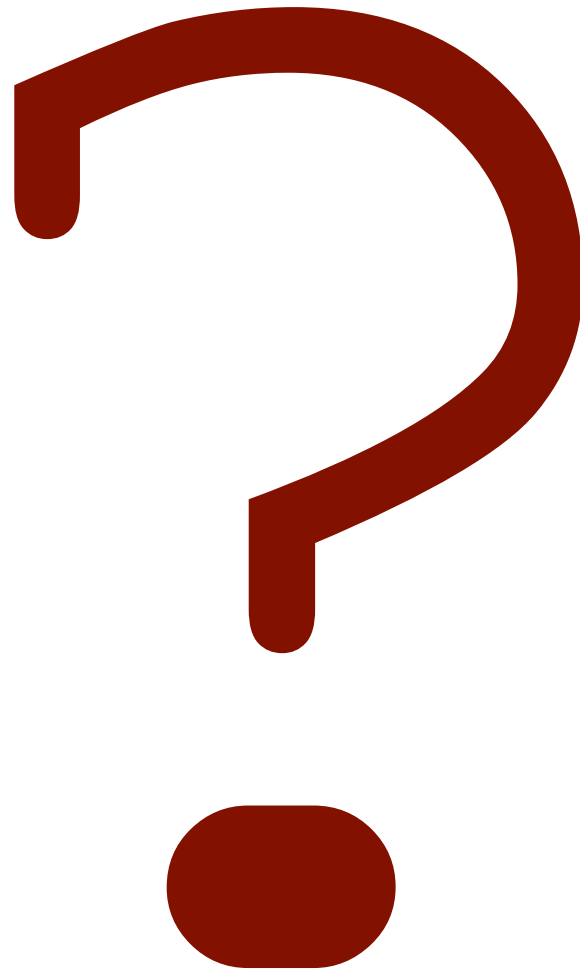
```
    public String getName() {  
        return name;  
    }
```

getter and setter
define a property

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

```
}
```





REFERENCES

- <http://hibernate.org/orm/what-is-an-orm/>
- <http://docs.oracle.com/javaee/7/api/index.html?javax/persistence/package-summary.html>
- <https://docs.oracle.com/javaee/7/tutorial/partpersist.htm>