Lesson:

Java Application Development

PW SKILLS

List of Concepts Involved:

- Hibernate Introduction
- Hibernate Architecture
- Usage of API to perform persistence operation
- Usage of Caching and Connection pooling in Hibernate
- Spring Introduction
- Spring core - Dependency Injection in annotation approach and IoC
- Introduction to Spring MVC using SpringBoot
- Spring MVC Architecture

When we already have JDBC technology, why do we need to go for Hibernate?

Limitations of JDBC

1. If we use JDBC to develop persistence logic,we need to write sql queries by following the syntax of "Database".
2. DBQueries are specific to Database,this makes JDBC not portable across multiple databases.
3. JDBC technology if we use and write a code, there would be a boilerplate code in our application.
4. JDBC technology throws only one Exception called "SQLException',but it is a CheckedException which means you should have handling logic otherwise code would not compile.

   ```
   try{}catch(SQLException e){}
   public static void main(String... args) throws SQLException{}
   ```

5. JDBC technology has only an Exception class called "SQLException",so we don't have a detailed hierarchy of Exceptions related to different problems.
6. The JDBC ResultSet object is not serializable, so we can't send it over the network, we need to use Bean/ POJO to send the data over the network by writing our own logic.
7. While closing the JDBC connection object, we need to analyse the code a lot otherwise it would result in "NullPointerException".

   eg: Connection con = DriverManager.getConnection(url,user,password)

   ```
   if(con!=null){.....}
   ```

   closing the connection object should take place in "finally" block only.

   To make the usage of AutoCloseable, we need to know the syntax of **"try with Resource".**
8. Java (OOP's based language)

   Assume we need to send a Student object to the database, can we write a logic of Database query at Object level if we use JDBC?

   No, Not possible because DBqueries always expects the value,but not the object directly.

9. JDBC does not have good support of **Transaction Management**
   a. local
   b. global(no support in JDBC)

10. JDBC supports only positional parameters,it is difficult for the user to inject the values
    It does not support named parameters.

11.JDBC does not supports versioning ,timestamp as inbuilt features
    versioning:: keeping track of how many times the record got modified.
    timestamp:: keep track of when record was inserted and when lastly it was modified.

12. To use JDBC, Strong knowledge of SQL is required.
13. While developing persistence logic using JDBC, we can't enjoy oops features like
    a. inheritance
    b. polymorphism
    c. composition
    Because JDBC does not allow objects as input values in sql queries.

Note:
Boilerplate code
A code which would repeat in multiple parts of the project with no change/small change is called boilerplate code.

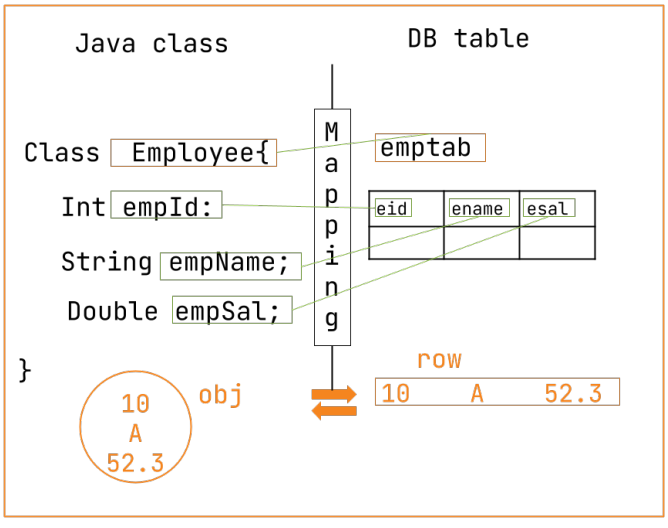
CRUD
=====

1. Load and register the driver(automatic from JDBC4.X)
2. Establish the connection
3. Create PrepareStatement
4. Execute the Query
5. Process the ResultSet
6. Handle the Exception
7. Closing the Resource

Step1,2,3,6,7 boiler plate code becoz it is a common logic.

Solution: To all the problems mentioned above we develop persistence logic using "ORM".
ORM -> Object Relational Mapping

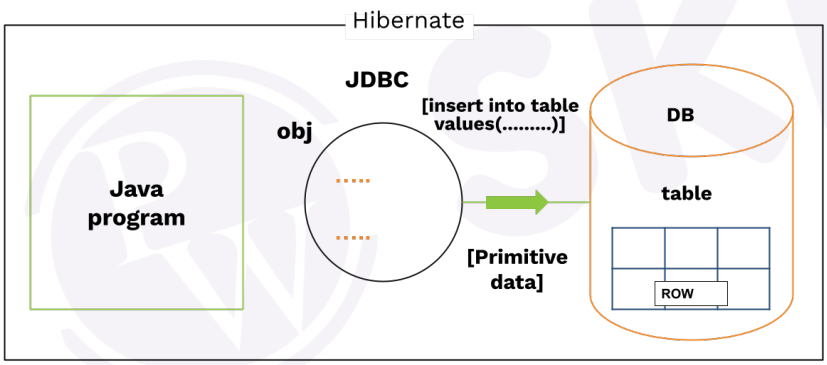
ORM:- (ORM stands for): Object Relational Mapping. It is a theory concept used at database programming to perform operations like insert. Update, delete and select in object format only ie.
JDBC converts object to primitive data and SQL Query should be written by programmer using primitives, which is not following OOPs.
ORM says “Do not convert object data, do operations in OOPs. Format only”.

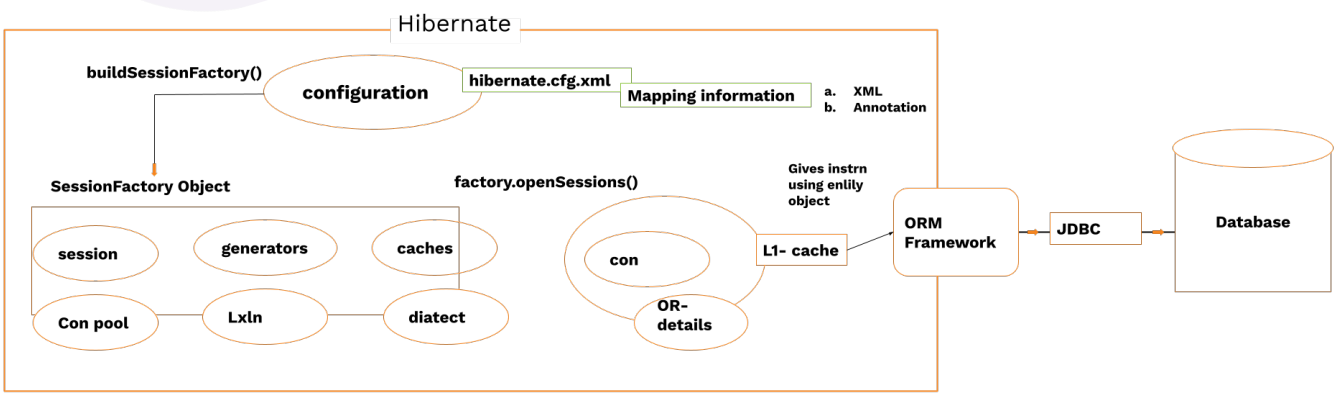


Hibernate Introduction

1. Hibernate is Database independent, it can be used for any type of Database.
2. Hibernate is applicable for both standalone applications and Enterprise Applications.
3. Hibernate is providing very good support for Associations and Joins.
4. Hibernate is having very good Annotations in order to reduce XML tech dependency in enterprise applications.
5. Hibernate is having very good implementations for Primary key generation algorithms in order to generate and insert a unique primary key value for each and every insertion operation.
6. Hibernate is providing very good Collections support to manage data.
7. Hibernate is having its own query language, which is database independent, Object Oriented, that is, HQL in order to perform database operations.
8. Hibernate has provided very good Cache mechanisms in order to reuse the results.
9. Hibernate is having very good support for Connection Pooling mechanisms in order to improve Connection re usability.
10. Hibernate is supported by almost all IDEs and Application Servers.
11. Hibernate is against for SQL Queries in enterprise applications directly, but, if we want to write database dependent sql queries then it is possible to provide database dependent sql queries by using "Native SQL".
12. Hibernate has provided very good transaction support.

HibernateDesign





Behind the scenes of Hibernate

- Hibernate Configuration File will provide all the configuration details like driver class name, driver URL, Database User name , Database password,.... which we require to establish connection with database and to setup JDBC environment.

- The Hibernate Mapping file will provide all the mapping details like Bean Class name and Database table name, ID property and Primary key column , Normal Properties and Normal Columns,....

Client Application will perform the following three actions in Hibernate applications mainly.

1. Activating Hibernate Software, in this case, Hibernate Software will take all configuration details from the hibernate configuration file and Hibernate Software will set up the required
2. JDBC Environment to perform database operations.
3. Prepare a Persistence Object with the persistence data.
4. Perform Persistence Operation.

When Client Application performs a persistence operation, Hibernate Software will perform the following actions.

1) Hibernate Software will take persistence method call and identify persistence Object.
2) Hibernate Software will take all mapping details from a hibernate mapping file like database table name and all column names on the basis of Persistence object.
3) Hibernate Software will prepare database dependent sql query on the basis of table names and column names and with the persistence object provided data.
4) Hibernate Software will execute the generated database dependent sql query and perform the required persistence operation.

Steps to prepare Hibernate Application:

1) Prepare Persistence Class or Object.

The main intention of the Persistence class or object in Hibernate applications is to manage Persistence data which we want to store in Database or by using this we want to perform the database operations like select, update, delete.

In Hibernate applications, to prepare Persistence classes we have to use the following Guidelines

1) In Hibernate applications Persistence classes must be POJO classes [Plain Old Java Object], they must not extend or implement a predefined Library.
2) In hibernate Applications Persistence classes must be public, non-abstract and non-final.
   - Where the main intention to declare persistence classes as public is to bring persistence classes scope to Hibernate software in order to create objects.
   - Where the main intention to declare persistence classes as Non abstract is to allow to create Objects for Persistence classes
   - Where the main intention to declare persistence classes as Non final is to allow to extend one persistence class to another persistence class as per the requirement.

3) In Persistence classes all Properties must be declared as per database table provided columns, where names are not required to be matched, but data types must be compatible.

4) In Persistence classes, all properties must be declared as private in order to improve Encapsulation.

5) In Persistence classes , we must define a separate set of setXXX () and getXXX () methods for each and every property.

6) In persistence classes, we must declare all methods are public.

7) In Persistence classes, if we want to provide any constructor then it must be public and 0-arg constructor, because, while creating object for persistence class Hibernate software will search and execute only public and 0-arg constructor.

8) In Hibernate applications , if we want to provide our own comparison mechanisms while comparing Persistence objects then it is suggestible to Override equals(--) method.

9) In Hibernate applications, if we want to provide our own hashcode values then it is suggestible to Override hashCode () method.

10) In Hibernate applications, we will use POJO classes, which are not extending and implementing predefined library, but, it is suggestible to implement java.io.Serializable marker interface in order to make eligible Persistence object for Serialization and Deserialization.

11) In Persistence classes, we have to declare a property as an ID property, it must represent the primary key column in the respective table.

2) Prepare Mapping File.

The main intention of mapping file in Hibernate applications is to provide mapping between a class,id property and normal properties from Object Oriented Data Model and a table, primary key column and normal columns from Relational data model.

In Hibernate applications, mapping file is able to provide the mapping details like Basic OR mapping, Component mapping, inheritance mapping, Collections mapping, Associations mapping,

To prepare mapping files in Hibernate applications we have to provide mapping file name with the following format.

"POJO_Class_Name.hbm.xml".

The above format is not mandatory, we can use any name but we must provide that intimation to the hibernate software.

In Hibernate applications, we can provide any no of POJO classes, w.r.t each and every POJO class we can define a separate mapping file.

It is also possible to provide mapping information through **annotations** also.

3) Prepare Hibernate Configuration File

The main purpose of the hibernate configuration file is to provide all configuration details of the hibernate application which includes Jdbc parameters to prepare connection , Transactions configurations,Cache mechanisms configurations, Connection pooling configurations,.....

In Hibernate applications, the standard name of hibernate configuration file is "hibernate.cfg.xml", it is not fixed, We can provide any name but that name must be given to Hibernate software.

In Hibernate applications, we are able to provide more than one configuration file , but, for each and every database, that is, in hibernate applications if we use multiple databases then we are able to prepare multiple configuration files.

To prepare a hibernate configuration file with basic configuration details we have to use the following XML tags.

4) Prepare Hibernate Client Application

The main intention of Hibernate Client application is to activate Hibernate Software, creating persistence objects and performing Persistence operations.

To prepare Client Application in hibernate applications we have to use the following steps.

1. **Create Configuration class object**
2. **Create Session Factory object**
3. **Create Session Object**
4. **Create a Transaction object if it is required.**
5. **Perform Persistence operations**
6. **Close Session Factory and Session objects.**

1. Create Configuration class object

In Hibernate, the main intention of the Configuration object is to store all the configuration details which we provided in the hibernate configuration file.

To represent Configuration object Hibernate has provided a predefined class in the form of "org.hibernate.cfg.Configuration".

To create a Configuration class object we have to use the following constructor from Configuration class.

public Configuration()

EX: Configuration cfg = new Configuration();

If we use the above instruction in Hibernate applications then we are able to get an empty Configuration object in heap memory, it will not include any Configuration details.

If we want to store Configuration details from the Configuration file we have to use either of the following methods.

1. public Configuration configure()

This method will get configuration details from the configuration file with the name hibernate.cfg.xml.

2. public Configuration configure(String config_file_Name)

This method can be used to get configuration details from a hibernate configuration file with a name, it will be used when we change configuration file name from hibernate.cfg.xml file to some other name .

3. public Configuration configure(File file)

This method can be used to get configuration details from a file which is represented in the form of java.io.File class object.

public Configuration configure(URL url)

This method can be used to get Configuration details from a file which is available in the network represented in the form of java.net.URL .

EX: Configuration cfg = new Configuration();

cfg.configure();

When we use configure() method then Hibernate Software will search for hibernate.cfg.xml file, if it is available then Hibernate software will load the content of hibernate.cfg.xml file, parse it and read content from configuration file to Configuration object.

2. Create Session Factory object:

In Hibernate, the main intention of Session Factory object is to manage Connections, Statements, Cache levels, ... and it is able to provide no of Hibernate Session objects.

To represent Session Factory object Hibernate has provided a predefined interface in the form of "org.hibernate.Session Factory".

To get a SessionFactory object we have to use the following method from the Configuration class.

public SessionFactory buildSessionFactory()

EX: SessionFactory sf = cfg.buildSessionFactory();

In Hibernate applications, if we use multiple Databases then we have to prepare multiple Configuration files, multiple Configuration Object, w.r.t this, we have to prepare multiple Session Factory objects.
Session Factory object is heavy weight and it is thread safe upto a particular Database, because it is able to allow more than one thread at a time.

3. Create Session Object:

In Hibernate, for each and every database interaction a separate Session will be created.
In Hibernate, Session is able to provide no persistence methods in order to perform persistence operations.
To represent Session object, Hibernate has provided a predefined interface in the form of "org.hibernate.Session".
To get a Session object, we have to use the following method from Session Factory.

public Session openSession()

EX: Session s =sf.openSession();
In Hibernate, Session object is light weight and it is not thread safe, because, for each and every thread a separate Session object will be created.

4. Create Transaction Object:

Transaction is a unit of work performed by Front End applications on Back end systems.
To represent Transactions, Hibernate has provided a predefined interface in the form of "org.hibernate.Transaction".

To get a Transaction object we will use either of the following methods.
1. public Transaction getTransaction()
It will return a Transaction object without begin, where to begin Transaction we have to use the following method.

public void begin()

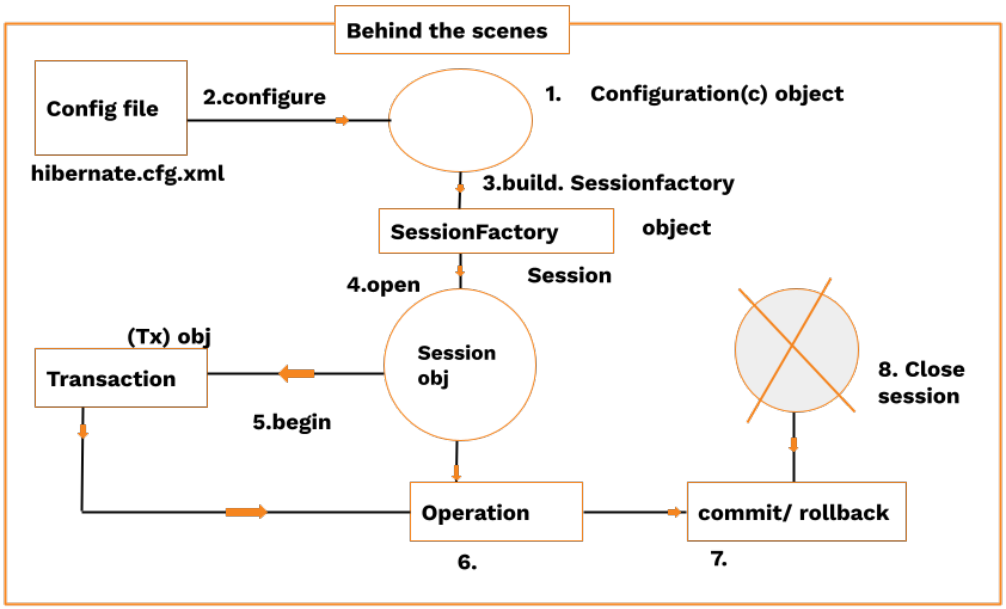
2. public Transaction beginTransaction()
It will return Transaction and begin Transaction.
In Hibernate applications, after performing persistence operations we must perform either commit or rollback operations in order to complete Transactions, for this, we have to use the following methods from Transaction.

5. Perform Persistence Operations:

In Hibernate applications, to perform persistence operations Session has provided the following methods.

To perform persistence operations we use the following methods

1. save()/persist()
2. get()/load()
3. update(),saveOrupdate(),merge()
4. delete()



Link Refer: hibernatecrudcode.zip

Usage of Caching and Connection pooling in Hibernate

- It is a temporary memory that holds the data for a temporary period of time.
- Cache at client side will hold server data and use it across the multiple same requests to reduce the network trip b/w client and server.

Hibernate supports 2 levels of Cache

a. First Level Cache(L-1 cache/session cache/default cache)
b. Second Level Cache(L2- cache/configurable cache)
eg: Stock Market trading,live game score,weather report,.....

Note:

- session.save(obj),session.saveOrUpdate(obj),session.delete(obj) methods keep the object in L1 Cache until tx.commit() is called.
- session.get() will get the object and keep it in L-1Cache and the same object will be used across multiple session.get() method calls with the same entity object id.

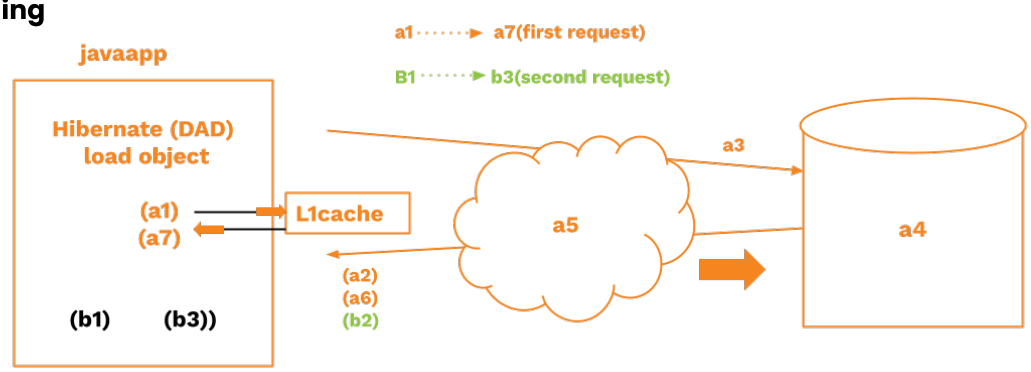
Methods used in Caching

a. evict(Object obj)
it will remove a particular object from L1-cache.

b. clear()
it will remove all objects present in L1-cache.

c. In L1-cache, duplicate objects won't be available.

caching



2nd level cache

- This caching is associated with "SessionFactory", so we call it "Global Cache".
- Application will start to search for entity object in the following order
  - L1 cache of current session(if not there)
  - L2 cache of SessionFactory object(if not there)
  - Collect from db and keep in L2 cache and L1 cache then give it to the application.
  - It is a configurable cache and we can enable or disable it.
  - hibernate supports L2 cache through "EHCache"

To configure EHCache in our hibernate projects we use

1. Add EHCache jars to the project
2. configure ehcache.xml as shown below

```
<ehcache>
    <diskStore path="java.io.tmpdir"/>
    <defaultCache
       maxElementsInMemory="100"
       eternal="false"
       timeToIdleSeconds="10"
       timeToLiveSeconds="30"
       overflowToDisk="true"
    />
</ehcache>
```

Also make changes in hibernate.cfg.xml file as shown below

```
        <!-- Configuring EH cache... -->
<property  name="hibernate.cache.use_second_level_cache">true</property>
<property  name="hibernate.cache.region.factory_class">
      org.hibernate.cache.ehcache.EhCacheRegionFactory
</property>
<property name="net.sf.ehcache.configurationResourceName">ehcache.xml</property>
```

3. In the model class inform hibernate to use Caching strategy for Read purpose.

```
@Entity
@Cache(usage = CacheConcurrencyStrategy.READ_ONLY)
public class InsurancePolicy implements Serializable{}
```

Locking in hibernate

- If multiple apps or threads simultaneously accessing and manipulating the records there is a possibility of getting concurrency problems.
- To Avoid this problem we need to use "Locking " of a record in hibernate.

Hibernate supports 2 levels of Locking

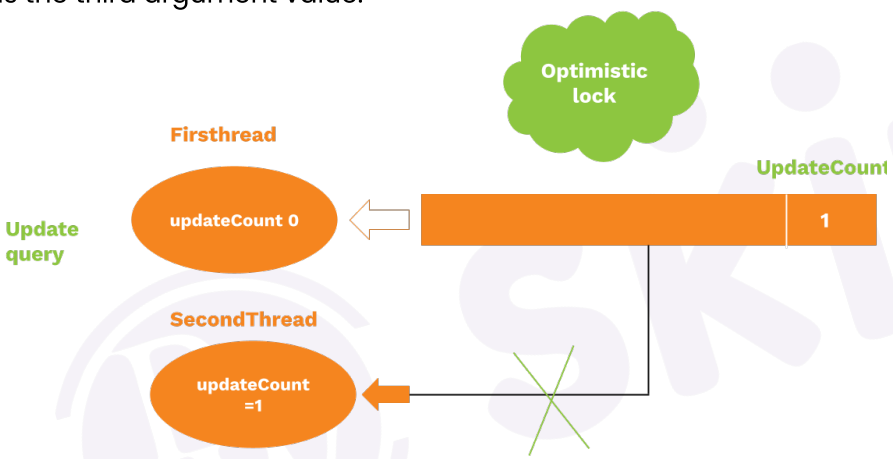
a. Optimistic Locking

It allows the second thread simultaneously to access and modify the record, then the first thread notices the modification and throws "Exception".
To enable this locking we need to use "@Version" in our application.
if we use **@Version** then automatically optimistic locking will be achieved.

b. Pessimistic Locking

- It will allow FirstThread to Lock the record itself,so if the second thread tries to access and modify the record then it should throw an "Exception".
- To enable this locking we need to use

session.get(Class c,Serializable s**,LOCKMODE.UPGRADE_NOWAIT)**
as the third argument value.



Spring Core Introduction

Java Learning is all about 3 things
a. Java Language(Core java course)
b. Java Technology(JDBC,Servlet,JSP,JSTL,EJB's,JMS,....)
EJB-> Enterprise Java Bean
JMS-> Java messaging Service
c. Framework(Hibernate,Spring,SpringBoot,MicroServices,RestApi's,....)

Framework is not a new technology, rather it is an abstraction provided on top of technology.
Third Party team would give apis in the form of jars which would generate boilerplate code based on the inputs we give to the internal containers of the framework.
eg: hibernate => based on configuration details supplied, it will create JDBC environment
Spring => based on configuration details supplied, it will create an object and maintains the object and performs dependency injection.

Different types of Framework to build application

a. Web application based framework
b. ORM Framework
c. Application Framework
d. BigData Framework
e. Distributed Application Development framework
etc....

WebApplication Framework

These frameworks provide abstraction on top of Servlet,JSP and simplify MVC architecture based development.

eg: Struts(Apache foundation)
SpringMVC(part of Spring)----> interface21(pivotal team)
JSF(Java Server Faces) -----> fromSUNMS/OracleCorporation
WebWork -------> symphony

ORM Framework

These frameworks provide abstraction on top of JDBC and simplify to develop object based. DBS/w independent persistence logic

eg: Hibernate ------> redhat
TopLink ------> oracle
Ibatis ------> apache

Application Framework

It is an allrounder framework that provides abstraction on top of multiple jee technologies and even on some frameworks to develop all kinds of logic and different type of app's.

eg: Distributed application
eg: myntra application
flipkart application
amazon application....
facebook application(web applications)

SpringFramework is not good in developing Distributed applications, so we prefer using "WebServices".

Distributed App development Framework

It simplifies the process of developing Distributed App's/Remoting Apps.
SOAP(outdated),Rest/RestfulServices/Restful WebServices(latest) :: jersy,RestEasy,....

Based on the mode of development we do, we have 2 types of framework

a. Invasive Framework
b. Non-Invasive Framework

Invasive Framework

=> Developer class will extend or implement an interface given by framework api.
=> Because of extends and implements the developer code would be tightly coupled with framework api.
=> It won't promote portability(moving the classes to a new framework would not execute).
eg: Servlet,Struts(1.X)

Note: working for a company without a bond.

How Spring evolved?

1995 --->Applet(Good for gaming)
1996 --->JavaBean[Technology used earlier]
(Started developing by using java classes + java beans)

Limitations

a. Doesn't allow remote clients,it works only with local clients.
b. Not suitable for large scale application
c. Programmers should handle Middleware web services along with the primary logic of the application.

1998 ---> EJB(Enterprise java Bean) for building Distributed application

Advantage

a. It can handle both remote and local clients.
b. Gives built in middleware services

Disadvantage

a. It runs only in server mode(heavy weight containers)
b. It is very complex to learn and use.

2002/2003---> Spring Framework --------> Rod Johnson(interface21)
|===> Provides abstraction over technology/frameworks

Advantage

a. All kinds of development is possible
b. Application development is non-invasive programming.
c. Built in middle-ware services[Transactionservice,connection pool service,....)
d. Lightweight application development
e. Easy to learn and easy to use.

Is Spring alternative to EJB,Struts,Hibernate,JEE technology?

Spring vs EJB

Answer: No , Spring framework is used to develop all kinds of apps.
WebServices are alternatives to EJB's.

Spring Vs Struts

Answer: No, Struts will be used to build only web based application
Spring can be used to build any type of application.
SpringMVC is an alternative to Struts.

Spring vs Hibernate

Answer: NO, hibernate is ORM framework to build persistence logic
Spring has its own ORMmodule through which it promotes abstraction
SpringORM,SpringDataJPA is an alternative to hibernate.

Spring vs JEE(Java Enterprise Edition)

Answer. No, JEE is a technology which gives api for persistence logic and building webapps.
Where as Spring provides an abstraction on top of JEE api's
SpringJDBC-> JDBC,SpringMVC-> Servlet,JSP

SpringCore

=> It is base module for other modules
=> This module is given to supply Springcontainers to perform Dependency management.
=> This module gives 2 spring containers/IOC[Inversion Of Control] containers called
a. BeanFactory
b. ApplicationContext(Latest one)
=> These 2 containers perform the following operations
a. It manages the SpringBean life cycle
b. It performs Dependency Management
a. Dependency LookUp
b. Dependency Injection[commonly used]

SpringApp can be developed in 4 approaches

a. XML approach(only used in maintenance projects).
b. using Annotation driven configuration.
c. using java code configuration.
d. using Spring boot auto driven configuration.

Different modes of DependencyInjection

1. Setter Injection.
2. Constructor injection.
3. Field injection.
4. MethodInjection/Method replacer.
5. LookUp Method Injection.
6. Dependency LookUp Injection.

Dependency Management

=> It is the process of assigning dependant object to Target object by loading both the classes and by creating the objects for both the classes.
=> The classes/objects which use the other class services are called "Target class".
=> The class that acts like a helper class to the main/target class is called "Dependent class".
Ex: Target class => Flipkart, Vehicle, Student , Mobile
Dependant class => DTDC , Engine , CourseMaterial, SIM

refer: IOCProj1-SetterInjection (LINK)