# Hibernate

# Аннотации в Java (java annotation types)

- Аннотации представляют собой некие метаданные, которые могут добавлятся в исходный код программы и семантически не влияют на нее, но могут использоваться в процессе анализа кода, компиляции и даже во время выполнения.

# Вот основные варианты использования аннтоаций:

- предоставлять необходимую информацию для компилятора;
- предоставлять метаданные различным инструментам для генерации кода, конфигураций и т.д.;
- использоваться в коде во время выполнения програмного кода (reflection).

Аннотации могут быть применены, например, к декларациям классов, полей, методов, ну и конечно же аннотаций :).

Для описания новой аннотации используется ключевое слово **@interface**. Вот банальный пример аннотации:

```
public @interface Description {
String title();
int version() default 1;
String text();
}
```

# пример ее использования:

```
@Description(title="title", version=2, text="text")
public class Clazz { /* */ }
```

# Maven

- это инструмент для сборки Java проекта: компиляции, создания jar, создания дистрибутива программы, генерации документации

# Преимущества **Maven**:

- **Независимость от OS**. Сборка проекта происходит в любой операционной системе. Файл проекта один и тот же.
- **Управление зависимостями**. Редко какие проекты пишутся без использования сторонних библиотек (зависимостей). Эти сторонние библиотеки зачастую тоже в свою очередь используют библиотеки разных версий. Мавен позволяет управлять такими сложными зависимостями. Что позволяет разрешать конфликты версий и в случае необходимости легко переходить на новые версии библиотек.
- **Возможна сборка из командной строки.** Такое часто необходимо для автоматической сборки проекта на сервере (Continuous Integration).
- **Хорошая интеграция с средами разработки.** Основные среды разработки на java легко открывают проекты которые собираются с помощью **maven**. При этом зачастую проект настраивать не нужно - он сразу готов к дальнейшей разработке. Как следствие - если с проектом работают в разных средах разработки, то **maven** удобный способ хранения настроек. Настроечный файл среды разработки и для сборки один и тот же - меньше дублирования данных и соответственно ошибок.

- Вся структура проекта описывается в файле: **pom.xml** (POM – Project Object Model), который должен находиться в корневой папке проекта.
- Ключевым понятием Maven является **артефакт** — это, по сути, любая библиотека, хранящаяся в репозитории (месте хранения). Это может быть какая-то зависимость или плагин.
- **Зависимости** — это те библиотеки, которые непосредственно используются в вашем проекте для компиляции кода или его тестирования.
- **Плагины** используются самим Maven'ом при сборке проекта или для каких-то других целей (деплоймент, создание файлов проекта для Eclipse и др.).
- **Архетип** — это некая стандартная компоновка файлов и каталогов в проектах различного рода (веб, swing-проекты и прочие). Другими словами, Maven знает, как обычно строятся проекты и в соответствии с архетипом создает структуру каталогов.

# Жизненный цикл

- **validate** — проверяет корректность метаинформации о проекте
- **compile** — компилирует исходники
- **test** — прогоняет тесты классов из предыдущего шага
- **package** — упаковывает скомпилированые классы в удобноперемещаемый формат (**jar** или **war**, к примеру)
- **integration-test** — отправляет упаковынные классы в среду интеграционного тестирования и прогоняет тесты
- **verify** — проверяет корректность пакета и удовлетворение требованиям качества
- **install** — добавляет пакет в локальный репозиторий, откуда он будет доступен для использования как зависимость в других проектах
- **deploy** — отправляет пакет на удаленный *production* сервер, откуда другие

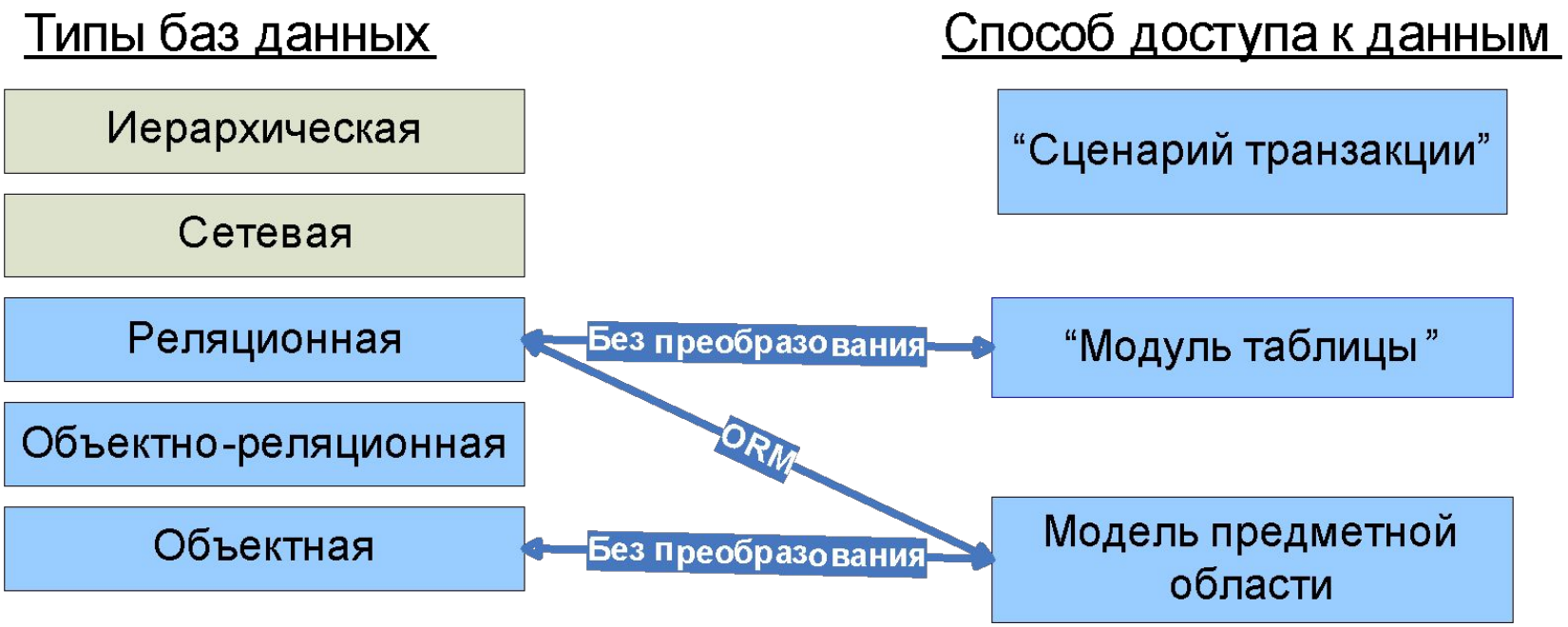
чтобы подключить *Spring Framework* необходимо определить следующую зависимость:

```
<dependencies>
  <dependency>
  <groupId>org.springframework</groupId>
    <artifactId>spring</artifactId>
    <version>2.5.5</version>
 </dependency>
</dependencies>
```

# Object Relational Mapping (ORM)

# Объектно-реляционное отображение

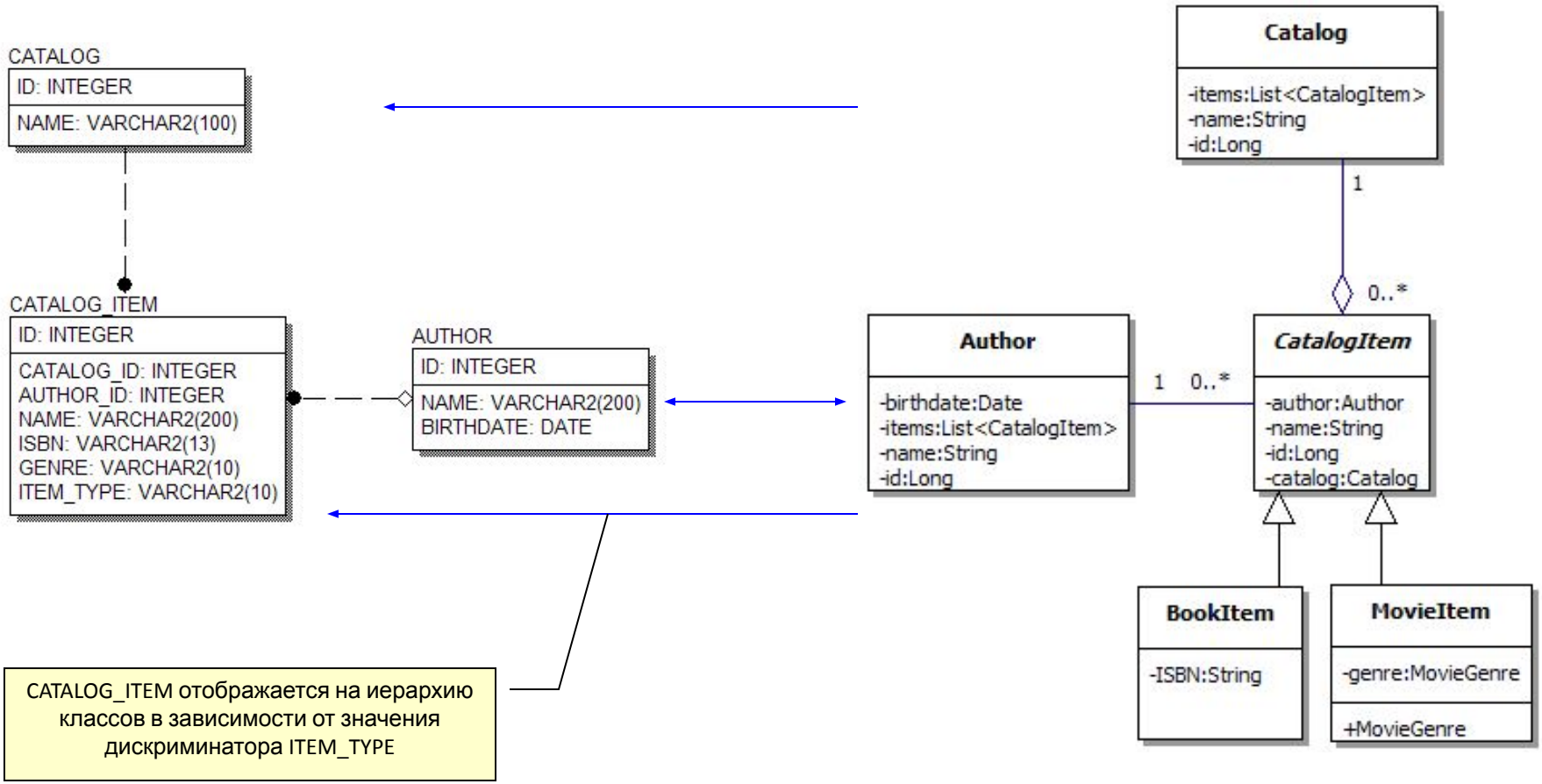
# Взаимодействие между типами баз данных и подходами доступа к данным



# Примеры реляционной и объектной модели

Реляционная модель

Объектная модель предметной области



# Отображение таблицы CATALOG и класса Catalog.java

```java
package org.mai806.catalog.model;

import java.util.List;

import javax.persistence.*;

@javax.persistence.SequenceGenerator(
    name="SEQ_ID",
    sequenceName="SEQ_ID"
)
@Entity @Table(name="CATALOG")
public class Catalog {

    private Long id;
    private String name;
    private List<CatalogItem> items;

    @Id @GeneratedValue(generator="SEQ_ID")
    @Column(name="ID")
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }

    @Column(name="NAME")
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }

    @OneToMany(cascade=CascadeType.ALL, fetch=FetchType.LAZY)
    @JoinColumn(name="CATALOG_ID", nullable = false)
    public List<CatalogItem> getItems() {
        return items;
    }
    public void setItems(List<CatalogItem> items) {
        this.items = items;
    }

}
```

- Класс Catalog связан с таблицей CATALOG
- Первичный ключ – **ID**, связан со свойством id (функции **getId()/setId()**)
- Для генерации значений первичного ключа **ID** используется sequence **SEQ_ID**
- Атрибут **NAME** связан со свойством **name** (функции **getName()/setName()**)
- Объекты **Catalog** содержат список **CatalogItem**
  - связь “один-ко-многим” (**@OneToMany**),
  - обязательная (**nullable=false**)
  - Внешний ключ, определяющий связь – **CATALOG_ID**
- Список CatalogItem загружаются по-требованию (**FetchType.LAZY**)
- При сохранении объекта **Catalog** автоматически сохраняются все его items (**CascadeType.ALL**)

# Работа с Hibernate API

```
17    // Получим сессию Hibernate
18    Session session = HibernateUtil.getSession();
19
20    // Начало транзакции
21    session.beginTransaction();
22
23    // Создание нового каталога и заполнение данными
24    Catalog catalog = new Catalog();
25    catalog.setName("Мой каталог №1");
26
27    List<CatalogItem> items =
28            new LinkedList<CatalogItem>();
29
30    Author fauler = new Author();
31    fauler.setName("Мартин Фаулер");
32    fauler.setBirthdate(new Date());
33
34    Author burton = new Author();
35    burton.setName("Тим Бартон");
36    burton.setBirthdate(new Date());
37
38    BookItem book = new BookItem();
39    book.setAuthor(fauler);
40    book.setName("UML Основы");
41    book.setISBN("1-232-12345-2");
42    book.setCatalog(catalog);
43
44    MovieItem movie = new MovieItem();
45    movie.setAuthor(burton);
46    movie.setName("Кошмар перед рождеством");
47    movie.setGenre(MovieItem.MovieGenre.MUSIC);
48    movie.setCatalog(catalog);
49
50    items.add(book);
51    items.add(movie);
52
53    catalog.setItems(items);
54
55    // Сохранение каталога в базу данных
56    session.save(catalog);
57    // Завершение транзакции
58    session.getTransaction().commit();
59
60    System.out.println("Id="+catalog.getId());
61
62    // Вывод списка каталогов
63    List<Catalog> list =
64      (List<Catalog>)session.createQuery("from Catalog").list();
65
66    for (Catalog cat : list) {
67        System.out.println(cat.getId() + " " + cat.getName());
68    }
```

- Работая с данными в объектно-ориентированном языке, мы работаем с объектами, заполняя и считывая значения полей, создавая новые или изменяя существующие объекты, определяя зависимости между объектами
- При операции save() мы передаем объект типа **Catalog**, который сохраняется в базу данных по описанным правилам отображения. В том числе сохраняются и все зависимые объекты (**CatalogItem**)
- Составляя запросы к базе данных, мы уже указываем не столбцы таблицы, а свойства объектов

# Протокол команд SQL

## Oracle

```
Hibernate: select SEQ_ID.nextval from dual
Hibernate: select SEQ_ID.nextval from dual
Hibernate: select SEQ_ID.nextval from dual
Hibernate: select SEQ_ID.nextval from dual
Hibernate: select SEQ_ID.nextval from dual
Hibernate: insert into CATALOG (NAME, ID) values (?, ?)
Hibernate: insert into AUTHOR (NAME, BIRTHDATE, id) values (?, ?, ?)
Hibernate: insert into CATALOG_ITEM (NAME, AUTHOR_ID, CATALOG_ID, ISBN, ITEM_TYPE, id) values (?, ?, ?, ?, 'BOOK', ?)
Hibernate: insert into AUTHOR (NAME, BIRTHDATE, id) values (?, ?, ?)
Hibernate: insert into CATALOG_ITEM (NAME, AUTHOR_ID, CATALOG_ID, GENRE, ITEM_TYPE, id) values (?, ?, ?, ?, 'MOVIE', ?)
Hibernate: update CATALOG_ITEM set CATALOG_ID=? where id=?
Hibernate: update CATALOG_ITEM set CATALOG_ID=? where id=?
Hibernate: select catalog0_.ID as ID0_, catalog0_.NAME as NAME0_ from CATALOG catalog0_
```

## SQL Server

```
Hibernate: insert into CATALOG (NAME) values (?)
Hibernate: insert into AUTHOR (NAME, BIRTHDATE) values (?, ?)
Hibernate: insert into CATALOG_ITEM (NAME, AUTHOR_ID, CATALOG_ID, ISBN, ITEM_TYPE) values (?, ?, ?, ?, 'BOOK')
Hibernate: insert into AUTHOR (NAME, BIRTHDATE) values (?, ?)
Hibernate: insert into CATALOG_ITEM (NAME, AUTHOR_ID, CATALOG_ID, GENRE, ITEM_TYPE) values (?, ?, ?, ?, 'MOVIE')
Hibernate: update CATALOG_ITEM set CATALOG_ID=? where id=?
Hibernate: update CATALOG_ITEM set CATALOG_ID=? where id=?
Hibernate: select catalog0_.ID as ID0_, catalog0_.NAME as NAME0_ from CATALOG catalog0_
```

# Возможности ORM

- Загрузка связанных объектов “по требованию” (lazy loading)
- Обеспечение пессимистической/оптимистической блокировок
- Кэширование загруженных объектов
- SQL-подобные запросы по объектной модели

# Преимущества ORM

- Нет необходимости писать рутинные insert/update/delete/select для CRUD операций
- Условия связи между объектами (строками таблиц) указываются декларативно в одном месте.
- Возможность использовать полиморфные запросы для иерархий классов
- Высокая степень независимости от конкретной СУБД

# Недостатки ORM

- Возможны проблемы с производительностью для сложных запросов на объектном SQL.
- Затрудняет использование специфических конструкций языка SQL конкретной СУБД.

# Реализации ORM

- Hibernate/NHibernate www.hibernate.org (Java / .NET 1.1,2.0)
- Oracle® TopLink® (Java)
- iBatis framework (Java, .NET) http://ibatis.apache.org/
- JPOX Java Data Objects (Java) http://www.jpox.org
- …

# Стандарты ORM

- EJB 1.1 Entity Beans
- Java Data Object (JDO)
  - JPOX
  - OpenAccess JDO
- EJB 3.0 Persistence API
  - Hibernate
  - Oracle TopLink

# Литература и ссылки

- Мартин Фаулер “Архитектура корпоративных программных приложений”. М., “Вильямс”, 2004
- http://en.wikipedia.org/wiki/Object-relational_mapping
- http://www.hibernate.org/hib_docs/reference/ru/html_single/