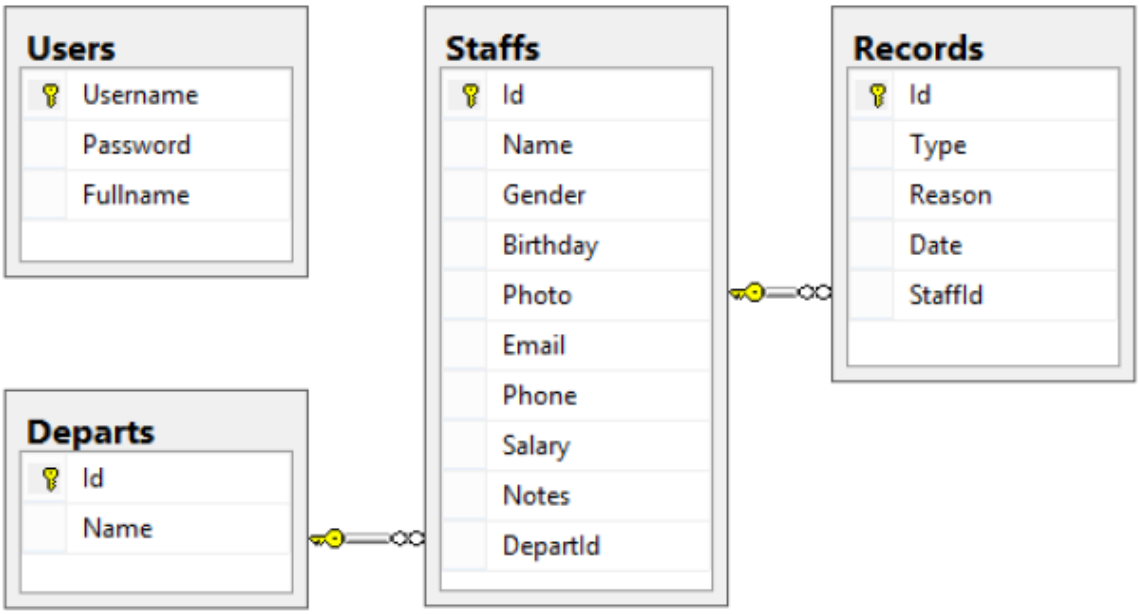
MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Biết cách tích hợp Hibernate vào Spring MVC
- ✓ Sử dụng annotation để ánh xạ thực thể
- ✓ Sử dụng Hibernate API để lập trình CSDL
- ✓ Viết câu lệnh HQL thành thạo

Cho CSDL Personel được sử dụng để quản lý thông tin nhân sự có mô hình quan hệ như sau:



Hãy tạo CSDL theo mô hình trên . Sau đây là mô tả chi tiết các bảng trong CSDL

Bảng Users

Cột	Kiểu	Ràng buộc	Mô tả
Username	NVARCHAR(50)	PK	Tên đăng nhập
Password	NVARCHAR(50)	NOT NULL	Mật khẩu
Fullname	NVARCHAR(50)	NOT NULL	Họ và tên

Bảng Departs

Cột	Kiểu	Ràng buộc	Mô tả
Id	NVARCHAR(10)	PK	Mã phòng
Name	NVARCHAR(50)	NOT NULL	Tên phòng

Bảng Staffs

Cột	Kiểu	Ràng buộc	Mô tả
Id	NVARCHAR(10)	PK	Mã nhân viên
Name	NVARCHAR(50)	NOT NULL	Họ và tên nhân viên
Gender	BIT	NOT NULL	Giới tính
Birthday	DATE	NOT NULL	Ngày sinh
Photo	NVARCHAR(50)	NOT NULL	Hình ảnh
Email	NVARCHAR(50)	NOT NULL	Địa chỉ email
Phone	NVARCHAR(25)	NOT NULL	Số điện thoại
Salary	FLOAT	NOT NULL	Lương
Notes	NVARCHAR(500)	NULL	Ghi chú
DepartId	NVARCHAR(10)	FK	Mã phòng ban

Bảng Records

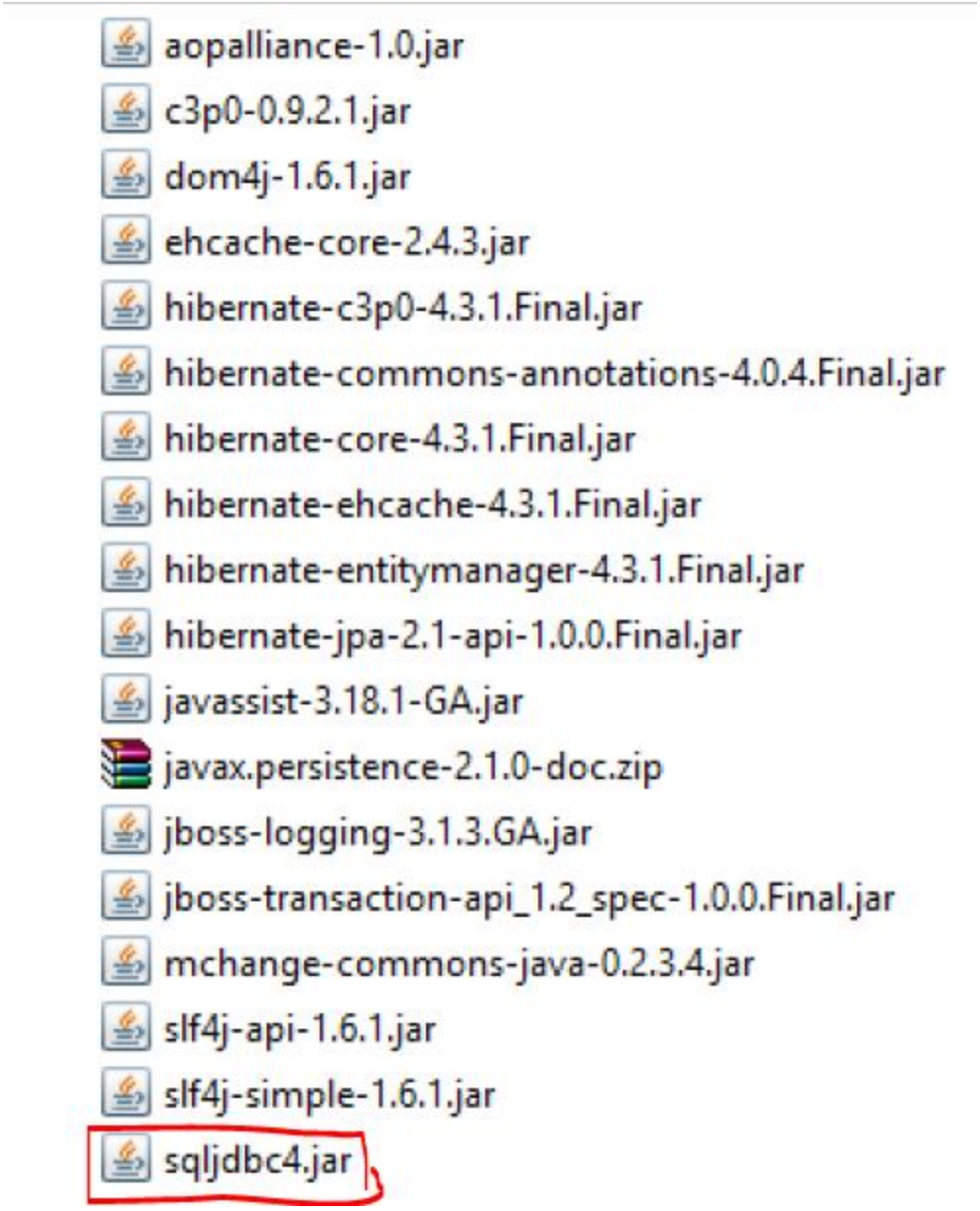
Cột	Kiểu	Ràng buộc	Mô tả
Id	BIGINT	PK, Tự tăng	Mã ghi nhận
Type	BIT	NOT NULL	Loại, 0=kỷ luật, 1=thành tích
Reason	NVARCHAR(200)	NOT NULL	Lý do
Date	DATE	NOT NULL	Ngày ghi nhận
StaffId	NVARCHAR(10)	FK	Mã nhân viên

Các bài tập sau đây sẽ thực hiện các chức năng xử lý dựa trên CSDL này

PHẦN I

Bài 1 (2 điểm): Tích hợp Hibernate

✓ Bổ sung các thư viện cần thiết sau đây vào dự án. Chú ý ngoại trừ sqljdbc4.jar là JDBC driver làm việc với SQL Server (nếu làm việc với CSDL khác, bạn chỉ cần thay đổi JDBC phù hợp với hệ quản trị CSDL đó), tất cả những thư viện còn lại là cần thiết của Spring.



✓ Xây dựng file cấu hình tích hợp Hibernate để làm việc với CSDL Personel được giới thiệu ở trên

Bài 2 (2 điểm)

Xây dựng các lớp thực thể ánh xạ vào các bảng của CSDL Personel. Cụ thể bạn phải xây dựng 4 lớp thực thể như sau

Bảng	Lớp thực thể	Chú ý Annotation
Users	User	@Entity, @Table, @Id
Departs	Depart	@Entity, @Table, @Id, @OneToMany
Staffs	Staff	@Entity, @Table, @Id, @ManyToOne, @OneToMany, @Tempral, @DateTimeFormat
Records	Record	@Entity, @Table, @Id, @Generated, @ManyToOne

Thực hiện theo các bước sau
Bước 1: Ánh xạ bảng
Tạo 4 lớp thực thể theo cấu trúc mã như sau

```
@Entity
@Table(name=" Students")
public class Student {
	@Id
	private String id;
}
```

Bước 2: Ánh xạ cột
Cứ mỗi cột trong bảng sẽ khai báo một trường với kiểu dữ liệu tương thích. Chú ý sử dụng các lớp Wrapper thay cho kiểu dữ liệu nguyên thủy vì giá trị của nó có thể null.

```
@Entity
@Table(name="Students")
public class Student {
	@Id
	private Integer id;
	private String fullname;
	private Boolean gender;
	private Date birthday;
	private Double mark;
	private String majorId;
}
```

Bước 3: Bổ sung các annotation đặc thù

- ✓ @GeneratedValue: cột khóa tự tăng
- ✓ @Tempral và @DateTimeFormat: cột kiểu Date
- ✓ @Column để ánh xạ trường với một cột nếu tên cột và tên trường khác nhau (tên cột thì không phân biệt hoa/thường còn tên trường thì có phân biệt)

```
@Entity
@Table(name="Students")
public class Student {
        @Id @GeneratedValue
        private Integer id;
        private String fullname;
        private Boolean gender;
        @Temporal(TemporalType.DATE)
        @DateTimeFormat(pattern="MM/dd/yyyy")
        private Date birthday;
        private Double mark;
        private String majorId;
}
```

Bước 4: Ánh xạ thực thể kết hợp @ManyToOne

Tìm tất cả các cột khóa ngoại và đổi thành thực thể kết hợp @ManyToOne

```
@Table(name="Students")
public class Student {
	@Id @GeneratedValue
	private Integer id;
	private String fullname;
	private Boolean gender;
	@Temporal(TemporalType.DATE)
	@DateTimeFormat(pattern="MM/dd/yyyy")
	private Date birthday;
	private Double mark;
	// private String majorId;
	@ManyToOne
	@JoinColumn(name="majorId")
	private Major major;
}
```

Bước 5: Ánh xạ thực thể kết hợp OneToMany

Ngược với ManyToOne, ở lớp thực thể kết hợp bạn khai báo thực thể kết hợp OneToMany (đối với Student thì trong Major sẽ có thực thể kết hợp OneToMany)

```
@Entity
@Table(name="Majors")
public class Major {
	@Id
	private String id;
	private String name;

	@OneToMany(mappedBy="major", fetch=FetchType.EAGER)
	private Collection<Student> students;
}
```

Lưu ý:

- ✓ Thuộc tính mappedBy chứa tên của thuộc tính kết hợp trong lớp Student

✓ Thuộc tính fetch có 2 chế độ nạp kèm thực thể kết hợp là EAGER và LAZY.

o Đối với ManyToOne thì mặc định là EAGER, nghĩa là nạp kèm thực thể kết hợp khi thực thể chính được nạp vào bộ nhớ.

o Đối với OneToMany thì mặc định là LAZY, nghĩa là không nạp kèm thực thể kết hợp khi thực thể chính được nạp vào bộ nhớ.

Bước 6: Sinh getter/setter

Bước cuối cùng cùng là bước đơn giản nhất. Bạn chỉ cần sinh mã getter/setter cho tất cả các trường (kể cả trường kết hợp)

PHẦN II

Bài 3 (2 điểm)

Xây dựng lớp UserController chứa các phương thức action thực hiện các chức năng sau đây:

✓ user/index.htm: liệt kê tất cả các User dưới dạng bảng

✓ user/insert.htm (GET): hiển thị form cho phép insert user mới

✓ user/insert.htm (POST): insert user mới vào CSDL

✓ user/update.htm (POST): update user vào CSDL

✓ user/delete.htm (POST): delete user CSDL

Thực hiện các chức năng trên theo hướng dẫn sau

Bước 1: Tạo UserController ánh xạ chung "/user/"

✓ Đính kèm @Transactional vào controller

✓ Tiêm SessionFactory vào controller

```
@Transactional
@Controller
@RequestMapping("/user/")
public class UserController {
        @Autowired
        SessionFactory factory;
}
```

Bước 2: Thêm index() ánh xạ "index" truy vấn danh sách user và đặt vào model.

Lưu ý: các thao tác truy vần nên sử dụng session được hệ thống Spring cung cấp sẵn bằng cách gọi getCurrentSession().

```
@RequestMapping("index")
public String index(ModelMap model) {
        Session session = factory.getCurrentSession();
        String hql = "FROM User";
        Query query = session.createQuery(hql);
        List<User> list = query.list();
        model.addAttribute("users", list);
        return "user/index";
}
```

Bước 3: Tạo view index.jsp hiển thị danh sách user

```
<table class="table table-hover">
<tr>
        <th>Username</th>
        <th>Password</th>
        <th>Fullname</th>
        <th></th>
</tr>
<c:forEach var="u" items="${users}">
<tr>
        <td>${u.username}</td>
        <td>${u.password}</td>
        <td>${u.fullname}</td>
        <td><a href="user/delete/${u.username}.htm">Delete</a></td>
</tr>
</c:forEach>
</table>
```

Bước 4: Chạy user/index.htm

Bước 5: Bổ sung insert() vào UserController

✓ GET: user/insert.htm để hiển thị form

✓ POST: user/insert.htm để thêm user mới. Khi làm thay đổi dữ liệu cần sử dụng cấu trúc mã để điều khiển transaction

```
@RequestMapping(value="insert", method=RequestMethod.GET)
public String insert(ModelMap model) {
        model.addAttribute("user", new User());
        return "user/insert";
}

@RequestMapping(value="insert", method=RequestMethod.POST)
public String insert(ModelMap model, @ModelAttribute("user") User user) {
        Session session = factory.openSession();
        Transaction t = session.beginTransaction();
        try {
                session.save(user);
                t.commit();
                model.addAttribute("message", "Thêm mới thành công !");
        }
        catch (Exception e) {
                t.rollback();
                model.addAttribute("message", "Thêm mới thất bại !");
        }
        finally {
                session.close();
        }
        return "user/insert";
}
```

Bước 6: Xây dựng view insert.htm. View này cần buộc bean có tên user trong model lên các điều khiển form.

```
<form:form action="user/insert.htm" modelAttribute="user">
        <div>
                <label>Username</label>
                <form:input path="username"/>
        </div>
        <div>
                <label>Password</label>
                <form:input path="password"/>
        </div>
        <div>
                <label>Fullname</label>
                <form:input path="fullname"/>
        </div>
        <div>
                <button class="btn btn-default">Insert</button>
        </div>
</form:form>
```

Bước 7: Chạy user/insert.htm sau đó nhấp vào nút *Insert+ sau đó chạy lại user/index.htm bạn sẽ thấy một user mới được tạo ra.

Tương tự xử lý thêm 2 trường hợp (update, delete).

Bài 4 (2 điểm)

Xây dựng trang web cho phép hiển thị danh sách thông tin nhân viên và tổng hợp thành tích, kỷ luật của nhân viên

✓ Cấu trúc thông tin nhân viên

Mã nhân viên	Họ và tên	Giới tính	Phòng ban
NV0001	Nguyễn Văn Tèo	Nam	Kế toán

✓ Cấu trúc tổng hợp thành tích, kỷ luật

Mã nhân viên	Tổng thành tích	Tổng kỷ luật	Tổng kết
NV0001	5	3	2
NV0002	0	1	-1

Hướng dẫn thực hiện

Các bạn tạo StaffController và thực hiện các bước tương tự như phần user/index.htm và chú ý các thay đổi sau đây

Bước 1: StaffController ánh xạ với tên "/staff/"

Bước 2: Xây dựng các phương thức action

✓ Action staff/index.htm hiển thị thông tin nhân viên

```
@RequestMapping("index")
public String index(ModelMap model) {
        Session session = factory.getCurrentSession();
        String hql = "FROM Staff";
        Query query = session.createQuery(hql);
        List<Staff> list = query.list();
        model.addAttribute("staffs", list);
        return "staff/index";
}
```

✓ Action staff/report.htm hiển thị thông tin tổng hợp thành tích và kỷ luật

```
@RequestMapping("report")
public String report(ModelMap model) {
        Session session = factory.getCurrentSession();
        String hql = "SELECT r.staff.id, "+
                     " SUM(case when r.type=1 then 1 else 0 end), "+
                     " SUM(case when r.type=0 then 1 else 0 end)"+
                     " FROM Record r "+
                     " GROUP BY r.staff.id";
        Query query = session.createQuery(hql);
        List<Object[]> list = query.list();
        model.addAttribute("arrays", list);
        return "staff/report";
}
```

Lưu ý: Thành tích và kỷ luật được lưu trong Records nên tiến hành tổng hợp từ đây

✓ Nhóm theo nhân viên: GROUP BY r.staff.id

✓ Đếm thành tích: SUM(case when r.type=1 then 1 else 0 end)

✓ Đếm kỷ luật: SUM(case when r.type=1 then 1 else 0 end)

✓ Kết quả truy vấn sẽ là một danh sách chứa các mảng, mỗi mảng gồm 3 phần tử là mã nhân viên, tổng thành tích và tổng kỷ luật

Bước 3: Tạo view index.jsp và report.jsp với những thay đổi sau đây

✓ View index.jsp

```
<table class="table table-hover">
<tr>
        <th>Mã NV</th>
        <th>Họ và tên</th>
        <th>Giới tính</th>
        <th>Phòng</th>
</tr>
<c:forEach var="s" items="${students}">
<tr>
        <td>${s.id}</td>
        <td>${s.fullname}</td>
        <td>${s.gender?'Nam':'Nữ'}</td>
        <td>${s.depart.name}</td>
</tr>
</c:forEach>
</table>
```

✓ View report.jsp

```
<table class="table table-hover">
<tr>
        <th>Mã NV</th>
        <th>Tổng thành tích</th>
        <th>Tổng kỷ luật</th>
        <th>Tổng kết</th>
</tr>
<c:forEach var="a" items="${arrays}">
<tr>
        <td>${a[0]}</td>
        <td>${a[1] }</td>
        <td>${a[2]}</td>
        <td>${a[1] * a[2]}</td>
</tr>
</c:forEach>
</table>
```

Bước 4: Chạy staff/index.htm và staff/report.htm bạn sẽ nhìn thấy kết quả

Bài 5 (2 điểm)

Xây dựng form ghi nhận thành tích và kỷ luật của nhân viên. Form này chỉ cần chọn nhân viên, chọn loại thành tích hay kỷ luật sau đó nhập lý do

Nhân viên:	ComboBox chứa danh sách nhân viên
Loại:	⊙Thành tích⊙Kỷ luật
Lý do:	Textarea
	[Ghi nhận]

Để thực hiện bài thực hành này, bạn cần tiến hành theo các bước sau

Bước 1: Tạo RecordController chứa các action như sau

```
@Transactional
@Controller
@RequestMapping("/record/")
public class StudentController {
	@Autowired
	SessionFactory factory;

	@RequestMapping(value="insert", method=RequestMethod.GET)
	public String insert(ModelMap model) {
		model.addAttribute("record", new Record());
		return "record/insert";
	}

	@RequestMapping(value="insert", method=RequestMethod.POST)
	public String insert(ModelMap model,
				@ModelAttribute("record") Record record) {
		Session session = factory.openSession();
		Transaction t = session.beginTransaction();
		try {
			record.setDate(new Date());
			session.save(record);
			t.commit();
			model.addAttribute("message", "Thêm mới thành công !");
		}
		catch (Exception e) {
```

```
                t.rollback();
                model.addAttribute("message", "Thêm mới thất bại !");
            }
            finally {
                session.close();
            }
            return "record/insert";
        }

        @ModelAttribute("staffs")
        public List<Staff> getStaffs() {
            Session session = factory.getCurrentSession();
            String hql = "FROM Staff";
            Query query = session.createQuery(hql);
            List<Staff> list = query.list();
            return list;
        }
}
```

Tổ chức gần tương tự insert() của user. Sự khác biệt ở đây là getStaffs() được viết để cung cấp dữ liệu cho ComboBox nhân viên ở giao diện form.
Bước 2: Thiết kế giao diện record.jsp

```
${message}
<form:form action="record/insert.htm" modelAttribute="record">
        <div>
                <label>Nhân viên</label>
                <form:select path="staff.id"
                                items="${staffs}" itemValue="id" itemLabel="name"/>
        </div>
        <div>
                <label>Loại</label>
                <form:radiobutton path="type" value="true" label="Thành tích"/>
                <form:radiobutton path="type" value="false" label="Kỷ luật"/>
        </div>
        <div>
                <label>Lý do</label>
                <form:textarea path="reason" rows="3"/>
        </div>
        <div>
                 <button>Insert</button>
        </div>
</form:form>
```

Chú ý: path="staff.id" sẽ chuyển mã của nhân viên chọn từ combobox vào StaffId trong bảng Staffs.

Bước 3: Chạy record/insert.htm, nhập dữ liệu và nhấp nút Insert. Kiểm tra dữ liệu trong bảng Records bạn sẽ thấy một bản ghi được thêm vào.

Tương tự như trường hợp thêm xử lý thêm 2 trường hợp (update,delete).