**终极秘典之JAVA框架**

1. **Hibernate**
2. **什么是ORM？**

ORM即对象关系映射，实体与表之间的映射，操作该实体就相当于操作实体对应的表，就是将数据从一种形式转换到另外一种形式；

1. **Hibernate注解**

[@Entity](mailto:1.@Entity) 标识这个类为一个实体

[@Table](mailto:2.@Table) 跟表映射，name标识表名

@Id 标识此为主键，

@GeneratedValue 标识主键策略strategy=GenerationType.IDENTITY ,表示主键自增

@ManyToOne 标识多对一的关系，需要添加外键

@JoinColumn 创建外键，name标识外键的名字

@OneToMany 标识一对多的关系，需要指定mappedBy,指定由来维护关联关系，不添加，会创建关联表

@ManyToMany 标识多对多的关系

@JoinTable(name="stu\_subject",joinColumns=@JoinColumn(name="s\_no"),inverseJoinColumns=@JoinColumn(name="sb\_no"))

创建关联表

1. **Hibernate级联关系详解：**
   * 1. 级联，就是对一个对象进行操作的时候，会把他相关联的对象也一并进行相应的操作；
     2. CascadeType.MERGE级联更新：修改了一个对象，保存时同时修改他关联的对象；
     3. CascadeType.REFRESH级联刷新：查询时即会重新查询数据库里的最新数据；
     4. CascadeType.PERSIST级联保存：对一个对象保存时也对关联对象保存
     5. CascadeType.REMOVE级联删除：对一个对象删除也对相关对象进行删除操作。
2. **如何理解Hibernate的延迟加载机制？**

1、在多对多或多对一的情况下，并不是在读取的时候就把数据加载进来，而是等到使用时再加载

2、@ManyToMany(fetch = FetchType.Lazy)；

1. **Hibernate中Session的load和get方法的区别是什么？**
   * 1. 如果没有找到符合条件的记录，get方法返回null，load方法抛出异常；
     2. get方法直接返回实体类对象，load方法返回实体类对象的代理(延迟查询)；
     3. 原本get方法对二级缓存是只写不读，而load则直接可以访问二级缓存，Hibernate3之后，get方法也可以访问二级缓存了；
2. **Hibernate中DetachedCriteria类跟Criteria比较**
   * + 1. 用法基本一致；
       2. Criteria依赖于session的createCriteria()方法；
       3. DetachedCriteria则不依赖于session，也被称为离线的criteria，只有在使用时才会关联相关session；
3. **Hibernate如何实现分页查询？**

1、调用Session的createCriteria()方法

2、设置查询起始行数（调用Query或Criteria接口的setFirstResult()方法）

3、最大查询行数（调用Query或Criteria接口的setMaxResults()方法）

4、调用Query或Criteria接口的list()方法

1. **Hibernate的一级缓存、二级缓存和查询缓存**

1、一级缓存：一级缓存就是Session级别的缓存，一个Session做了一个查询操作，它会把这个操作的结果放在一级缓存中；

2、二级缓存：由sessionfactory提供，其全局性决定了所有session都可以共享该缓存，默认关闭，需要用到外置缓存，比如ehcache；

3、ehcache：可以把查询出来的数据存储在内存或者磁盘，节省下次同样查询语句再次查询[数据库](http://lib.csdn.net/base/mysql" \t "_blank" \o "MySQL知识库)，大幅减轻数据库压力

4、查询缓存：对普通属性进行缓存，将查询语句和结果以键值对缓存，对于同样的查询可直接从缓存中获取数据，默认关闭，需要显示开启，开启方法：criteria.setCacheable(true)；

1. **@OneToMany注解的mappedBy属性**

1、只有OneToOne，OneToMany，ManyToMany上才有mappedBy属性，ManyToOne不存在该属性；

2、mappedBy标签一定是定义在被拥有方的；

3、拥有方能够自动维护跟被拥有方的关系；

4、mappedBy跟joinColumn/JoinTable总是处于互斥的一方，mappedBy这方定义JoinColumn/JoinTable总是失效的，不会建立对应的字段或者表；

1. **数据库连接池**

高效的管理数据库连接，在系统启动时先初始化几条数据库连接放到池子里，需要用时去池子里取，用完后再放回池子。因为每次创建数据库连接开销较大。

Hibernate推荐使用C3P0连接池

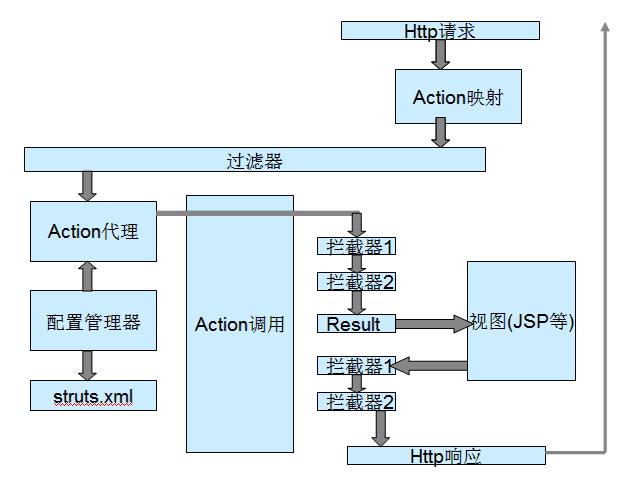
Spring 推荐使用DBCP连接池

1. **C3P0连接池与DBCP连接池的区别**

C3P0：安全稳定，它能自动回收空闲连接，提供最大空闲时间，超过最大空闲时间，当前连接会被断掉

DBCP：速度快，提供最大连接数，超过最大连接数，所有连接会被断开

1. **Struts 2**

****

1. **Struts 2请求流程**

1.一个Action请求先到Struts2核心过滤器

2.经过各类拦截器，过滤请求

3.去Struts.xml里面匹配相应的Action

4.去相应的Action里面执行execute方法

5. execute方法返回相应的跳转页面

1. **拦截器**
2. 框架级的内部拦截器
3. 自定义的拦截器。需继承AbstractInterceptor，重写intercept方法,还需struts.xml里配置拦截器
4. **值栈**

Struts2优势之一，前后台的值传递特别的方便。但是也是Struts2的弊端之一，特别影响性能，成本大。

1.对象栈

2.Map栈

1. **OGNL表达式**
   * 1. Struts 2默认的表达式语言是OGNL，EL的加强版
     2. OGNL是通常要结合Struts 2的标志一起使用。主要是#、%和$这三个符号的使用
2. **如何在web.xml中配置Struts 2？**

  <filter>

        <filter-name>struts2</filter-name>

        <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>

    </filter>

    <filter-mapping>

        <filter-name>struts2</filter-name>

        <url-pattern>/\*</url-pattern>

    </filter-mapping>

1. **SpringMvc**
2. **SpringMvc简介**
3. MVC模型时下最流行的实现，注解是它的特点
4. DispatcherServlet 是SpringMvc的核心Servlet
5. 将请求映射到相应的Controller
6. 视图解析器，把返回的值，拼接成 全径的页面
7. **SpringMvc的访问流程**

1.请求先提交给DispatcherServlet，由他来分发请求。  
2.通过@RequestMapping找到处理请求的Controller.  
3.Controller进行业务逻辑处理后，会返回一个ModelAndView  
5.通过视图解析器,拼接返回的页面路径

6.视图对象负责渲染返回给客户端

1. **SpringMvc常用注解**
2. @Controller 注入Controller组件
3. @RequestMapping 匹配请求
   1. value="login" 匹配的值
   2. method=RequestMethod.POST 请求的方式
4. @ResponseBody 以Json的格式返回数据
5. **Map, Model, ModelMap, MondelAndView的区别**

1. 效果都一样，都相当于request作用域

    @RequestMapping(value="login",method=RequestMethod.POST)

    public ModelAndView login(Map<String, Object> map,Model model,ModelMap modelMap) {

        map.put("map", "map");

        model.addAttribute("model", "model");

        modelMap.addAttribute("modelMap", "modelMap");

        ModelAndView mv = new ModelAndView("login").addObject("modelAndView", "modelAndView");

        return mv;

    }

1. **如何在web.xml中配置SpringMvc？**

<servlet>

<servlet-name>appServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:servlet-context.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

1. **SpringMvc获取Session,request ,直接在方法的参数里传入**

    @RequestMapping("login")

    public String login(HttpSession session, HttpServletRequest request)

    {

        session.setAttribute("session", "session");

        request.setAttribute("request", "request");

        // 返回跳转的页面view

        return "login";

    }

1. **SpringMVC与Struts2的主要区别？**
2. springmvc的入口是一个servlet即前端控制器，而struts2入口是一个filter过虑器。
3. springmvc是基于方法开发，传递参数是通过方法形参，可以设计为单例或多例(建议单例)，struts2是基于类开发，传递参数是通过类的属性，只能设计为多例。
4. Struts采用值栈存储请求和响应的数据，通过OGNL存取数据， springmvc通过参数解析器是将request对象内容进行解析成方法形参，将响应数据和页面封装成ModelAndView对象，最后又将模型数据通过request对象传输到页面。 Jsp视图解析器默认使用jstl。
5. **Spring**
6. **Spring简介**

Spring是一个开源的轻量级后台框架

1. **Spring组件扫描注解**
   * 1. @Component: 基本注解, 标识了一个受 Spring 管理的组件
     2. @Respository: 标识持久层组件
     3. @Service: 标识服务层(业务层)组件
     4. @Controller: 标识表现层组件
2. **Spring两大特性**

1.IOC：控制反转 DI：依赖注入

IOC:将由程序员操纵的javabean交由IOC容器统一管理；

DI:javabean之间的关联也由IOC容器来动态注入；

2.AOP：面向切面

把一些公共模块(如日志、事务、校验、安全)切入到业务逻辑中，让业务逻辑单纯处理业务，不关注公共模块的业务处理，以此达到模块单一化的目的，便于维护；

横切关注点：所要插入的公共模块，例如事务、日志等；

切面：公共模块的具体实现；

切入点：@Pointcut 相当于查询条件，可以匹配多个连接点

连接点：Joinpoint 相当于数据的记录

1. **Spring依赖注入的两种方式**
2. 属性注入

<bean id="car" class="com.igeek.entity.Car">

<property name="brand" value="阿斯顿马丁"></property>

</bean>

也可以用p标签进行属性注入:

<bean id="fouckCar" class="com.igeek.entity.Car"

p:brand="宝马"></bean>

1. 构造器注入

<bean id="huoCar" class="com.igeek.entity.Car">

<constructor-arg value="布加迪"></constructor-arg>

</bean>

1. **Spring 自动注入的两种方式**

可以用autowire来实现bean之间依赖关系的自动注入；

* 1. byName：根据bean的name来自动注入
  2. byType：根据bean的类型来自动注入

1. **Bean的作用域scope**

1.singleton:单例，IOC中只会存在一个对象

2.prototype：原型，每getBean,都会得到一个新的对象

3.request:每次请求都会创建一个对象

4.session：每个Session都会创建一个对象

1. **Spring创建Bean的3种方式**

1、缺省方式：使用构造器创建

2、工厂方法创建对象

静态工厂方法创建对象

实例工厂方法创建对象

3、使用FactoryBean创建对象

1. **Spring配置文件中加载配置**

<!--  加载配置文件 -->

<context:property-placeholder location="user.properties"/>

1. **Spring创建 Bean 后置处理器**

实现BeanPostProcessor接口

1. Bean前置处理方法：postProcessAfterInitialization
2. Bean后置处理方法：postProcessBeforeInitialization
3. **SPEL详解**

Spring的EL表达式，栗子：

<bean id=*"dailyWage"* class=*"DailyWage"*>

<property name=*"WorkTime"* value=*"3"*></property>

<property name=*"Money"* value=*"#{3\*100}"*></property>

</bean>

<bean id=*"Employee"* class=*"Employee"* init-method = *"init"*>

<property name=*"Name"* value=*"#{'acc'}"*></property>

<property name=*"Age"* value=*"#{dailyWage.money}"*></property>

<property name=*"DailyWage"* value = *"#{dailyWage}"*></property>

</bean>

1. **在Web项目中配置Spring的IoC容器**

如果需要在Web项目中使用Spring的IoC容器，可以在Web项目配置文件web.xml中做出如下配置：

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext.xml</param-value>

</context-param>

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

1. **@AutoWired和@Resource的区别**

1、@Resource:属于JDK1.6的注解，必须根据name或者type进行注入

@Resource(type=EmployeeDao.class)

@Resource(name="employeeDao")

2、@AutoWired：属于Spring注解，和@Resource作用一样，相当于@Resource的根据type注入

@Autowired

EmployeeDao employeeDao;

1. **Spring中@Qualifier注解有什么用？**

1、当Autowired注入的依赖关系能匹配到多个实现时，用@Qualifier指定具体注入的实现；

2、@Qualifier()在不指定实现别名的情况下，注入的实现类首字母小写；

@Autowired

@Qualifier("employeeDao")

BaseDao baseDao;

1. **B/S和C/S**

B/S即Browser/Server web端对服务器段，即网页编程；

1、便于升级维护，只需要升级服务端即可；

2、开发成本较低；

3、支持跨平台；

C/S即Client/Server 客户端对服务器端，即客户端编程；

1、运行较为稳定，安全系数更高；

2、动画渲染效果更高；

3、开发成本高；

4、升级维护麻烦，须安装才能使用；

1. **AOP的几种通知**

@Pointcut("execution(\* com.Acc.\*(..))")

public void pointCut(){}

1、前置通知：

@Before("pointCut()")

2、后置通知：

@After("pointCut()")

3、返回通知：

@AfterReturning(value = "pointCut()",returning = "result")

4、异常通知：

@AfterThrowing(value = "pointCut()",throwing = "exc")

5、环绕通知：方法由环绕通知自己执行

@Around(value = "pointCut()")

1. **Spring事务**

1、编程式事务，用代码的方式来控制事务，不常用

2、声明式事务，Spring推荐的非侵入式编程

a) 注解方式

b) aop方式

1. **Spring中Propagation类的事务属性详解**  
   REQUIRED：支持当前事务，如果没有事务，就新建一个事务。SUPPORTS：支持当前事务，如果没有事务，以非事务方式执行。   
   MANDATORY：支持当前事务，如果当前没有事务，就抛出异常。   
   REQUIRES\_NEW：新建事务，如果当前存在事务，把当前事务挂起。   
   NOT\_SUPPORTED：以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。   
   NEVER：以非事务方式执行，如果当前存在事务，则抛出异常。   
   NESTED：支持当前事务，如果当前事务存在，则执行一个嵌套事务，如果当前没有事务，就新建一个事务。
2. **Mybatis**

**Mybatis是一个半Orm框架，基于SQL语句的映射**

1. **为什么说Mybatis是半自动ORM映射工具？它与全自动的区别在哪里？**

Hibernate属于全自动ORM映射工具，使用Hibernate查询关联对象或者关联集合对象时，可以根据对象关系模型直接获取，所以它是全自动的。而Mybatis在查询关联对象或关联集合对象时，需要手动编写sql来完成，所以，称之为半自动ORM映射工具。

1. **Mybatis中#{}和${}的区别是什么？**

1、${}是Properties文件中的变量占位符，它可以用于标签属性值和sql内部，属于静态文本替换。

2、#{}是sql的参数占位符，Mybatis会将sql中的#{}替换为?号，在sql执行前会使用PreparedStatement的参数设置方法，按序给sql的?号占位符设置参数值。

1. **Xml映射文件中，除了常见的select|insert|updae|delete标签之外，还有哪些标签？**

<resultMap>、<parameterMap>、<sql>、<include>、<select

Key>，加上动态sql的9个标签，trim|where|set|foreach|if|

choose|when|otherwise|bind等，其中<sql>为sql片段标签，通过<include>标签引入sql片段，<selectKey>为不支持自增的主键生成策略标签。

1. **Mybatis是如何进行分页的？分页插件的原理是什么？**

Mybatis使用RowBounds对象进行分页，它是针对ResultSet结果集执行的内存分页，而非物理分页，可以在sql内直接书写带有物理分页的参数来完成物理分页功能，也可以使用分页插件来完成物理分页。