Web开发-JavaEE应用&ORM框架&SQL预编译&JDBC&MyBatis&Hibernate&Maven

****

#知识点：  
1、安全开发-原生PHP-超级全局变量  
2、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
0、安全开发-原生PHP-数据库通讯  
1、安全开发-原生PHP-身份验证技术  
2、安全开发-Cookie&Session&Token  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-弱类型脆弱  
2、安全开发-原生PHP-函数&数据类型  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-开发组件集合  
2、安全开发-原生PHP-模版引擎渲染  
3、安全开发-原生PHP-第三方编辑器  
---------------------------------------------  
1、安全开发-框架技术-ThinkPHP  
2、安全开发-框架安全-版本&写法  
3、安全开发-ThinkPHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-文件安全操作  
2、安全开发-原生PHP-上传读取删除包含等  
3、安全开发-原生PHP-代码审计文件安全---------------------------------------------  
1、安全开发-原生PHP-RCE安全  
2、安全开发-原生PHP-代码执行&命令注入  
3、安全开发-原生PHP-代码审计案例  
---------------------------------------------  
1、安全开发-原生PHP-PHP.INI安全  
2、安全开发-原生PHP-全局文件&单函数  
3、安全开发-原生PHP-流量检测&AI算法  
---------------------------------------------  
1、安全开发-原生JS-语法模型概念  
2、安全开发-Ajax技术-JQuery&Axios  
3、安全开发-Ajax应用-文件上传&用户登录---------------------------------------------  
1、安全开发-原生JS-DOM树&BOM对象  
2、安全开发-原生JS-DOM安全&安全案例  
---------------------------------------------  
1、安全开发-原生JS-数据加密&代码混淆  
2、安全开发-原生JS-数据解密安全案例  
---------------------------------------------  
1、安全开发-NodeJS-开发环境&功能实现  
2、安全开发-NodeJS-安全漏洞&案例分析  
3、安全开发-NodeJS-特有漏洞&代码审计  
---------------------------------------------  
1、安全开发-WebPack-构建打包器  
2、安全开发-WebPack-源码泄漏还原  
---------------------------------------------  
1、安全开发-微信小程序-搭建&开发&架构&安全  
2、安全开发-微信小程序-编译调试&反编译&泄露  
---------------------------------------------  
1、安全开发-JavaEE-Servlet技术  
2、安全开发-JavaEE-监听器&过滤器  
---------------------------------------------  
0、安全开发-JavaEE-构建工具-Maven  
1、安全开发-JavaEE-ORM框架-JDBC  
2、安全开发-JavaEE-ORM框架-Mybatis  
3、安全开发-JavaEE-ORM框架-Hibernate  
4、安全开发-JavaEE-ORM框架-SQL注入&预编译  
  
#章节点：(待补充)  
#具体点：(待补充)

****

演示案例：

* WEB开发-JavaEE-Servlet&路由&生命周期
* WEB开发-JavaEE-过滤器Filter&监听器Listen
* WEB开发-JavaEE-Maven&JDBC&Mybatis&Hibernate
* WEB开发-JavaEE-ORM框架-SQL注入&预编译

#Maven配置  
参考：https://blog.csdn.net/cxy2002cxy/article/details/144809310  
  
#JDBC  
参考：https://www.jianshu.com/p/ed1a59750127  
1、引用依赖（pom.xml）  
https://mvnrepository.com/  
2、注册数据库驱动  
Class.forName("com.mysql.jdbc.Driver");  
3、建立数据库连接  
String url ="jdbc:mysql://localhost:3306/phpstudy";  
Connection connection=DriverManager.getConnection(url,"root","123456");  
4、创建Statement执行SQL  
Statement statement= connection.createStatement();  
ResultSet resultSet = statement.executeQuery(sql);  
5、结果ResultSet进行提取  
while (resultSet.next()){  
 int id = resultSet.getInt("id");  
 String page\_title = resultSet.getString("page\_title");  
 .......  
}  
  
安全注入例子：  
预编译：PreparedStatement  
安全写法(预编译)： "select \* from admin where id=?"  
不安全写法(拼接)： "select \* from admin where id="+id  
  
#Hibernate  
1、引用依赖（pom.xml）  
https://mvnrepository.com/  
hibernate-core，mysql-connector-java  
2、Hibernate配置文件  
src/main/resources/hibernate.cfg.xml  
<?xml version='1.0' encoding='utf-8'?>  
<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
 <session-factory>  
 <!-- 数据库连接配置 -->  
 <property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
 <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/phpstudy?useUnicode=true&characterEncoding=UTF-8&serverTimezone=UTC</property>  
 <property name="hibernate.connection.username">root</property>  
 <property name="hibernate.connection.password">123456</property>  
  
 <!-- 数据库方言 -->  
 <property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>  
  
 <!-- 显示 SQL 语句 -->  
 <property name="hibernate.show\_sql">true</property>  
  
 <!-- 自动更新数据库表结构 -->  
 <property name="hibernate.hbm2ddl.auto">update</property>  
  
 <!-- 映射实体类 -->  
 <mapping class="com.example.entity.User"/>  
 </session-factory>  
</hibernate-configuration>  
3、映射实体类开发  
用来存储获取数据：  
src/main/java/com/example/entityUser.java  
4、Hibernate工具类  
用来Hibernate使用：  
src/main/java/com/example/util/HibernateUtil.java  
5、Servlet开发接受：  
src/main/java/com/example/servlet/UserQueryServlet.java  
  
安全注入例子：  
安全写法：String hql = "FROM User WHERE username=:username";  
不安全写法：String hql = "FROM User WHERE username='"+username+"'";  
  
#MyBatis  
1、引用依赖（pom.xml）  
mybatis，mysql-connector-java  
2、MyBatis配置文件  
src/main/resources/mybatis-config.xml  
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE configuration  
 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-config.dtd">  
<configuration>  
 <environments default="development">  
 <environment id="development">  
 <transactionManager type="JDBC"/>  
 <dataSource type="POOLED">  
 <property name="driver" value="com.mysql.cj.jdbc.Driver"/>  
 <property name="url" value="jdbc:mysql://localhost:3306/phpstudy?serverTimezone=UTC"/>  
 <property name="username" value="root"/>  
 <property name="password" value="123456"/>  
 </dataSource>  
 </environment>  
 </environments>  
 <mappers>  
 <mapper resource="AdminMapper.xml"/>  
 </mappers>  
</configuration>  
3、AdminMapper.xml创建  
src/main/resources/mybatis-config.xml  
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE mapper  
 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.example.jdbcdemo43.mapper.AdminMapper">  
 <select id="selectAdminById" resultType="com.example.jdbcdemo43.model.Admin">  
 SELECT \* FROM admin WHERE id = #{id}  
 </select>  
</mapper>  
4、创建数据实体类  
com/example/mybatisdemo43/model/User.java  
5、创建mapper实体类  
com/example/mybatisdemo43/mapper/AdminMapper.java  
6、创建servlet接受类  
com/example/mybatisdemo43/servlet/SelectServlet.java  
// 加载 MyBatis 配置文件  
String resource = "mybatis-config.xml";  
InputStream inputStream = Resources.getResourceAsStream(resource);  
SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);  
// 获取 SqlSession  
try (SqlSession session = sqlSessionFactory.openSession()) {  
 // 获取 Mapper 接口  
 AdminMapper mapper = session.getMapper(AdminMapper.class);  
 // 执行查询  
 Admin admin = mapper.selectAdminById(Integer.parseInt(id));  
 // 输出结果  
  
安全注入例子：  
1、安全写法： select \* from admin where id = #{id}  
2、不安全写法：select \* from admin where id = ${id}  
  
#Spring JPA  
由于涉及到开发框架，后续讲到，安全基本和Hibernate相似

涉及资源：[资源下载地址](https://docs.qq.com/doc/DQ3Z6RkNpaUtMcEFr)