Web 应用系统跨多级防火墙访问企业数据资源的解决方案①

刘艳艳[1]　李忠延[2]　(1.中国海洋大学 信息科学与工程学院 山东 青岛 266110;
2.潍坊交运公司 潍坊汽车总站 山东 潍坊 261041)

摘　要： 分析了企业网的结构，针对防火墙在保障网络安全的同时严格限制外部访问的情况，提出了 Web 应用系统跨越多级防火墙访问企业数据资源的软件架构。以汽车站客运班次查询为例，基于轻量级 JavaEE 平台开发了采用该架构的应用系统，阐述了该解决方案的具体实现方法和步骤。并进一步分析了应用服务器，给出了提高应用服务器性能的若干方法。最后，对方案进行了评价，指出了该方案易于扩展、便于维护、灵活性高的特点。实践表明，该方案具有较好的推广应用价值。

关键词： 跨防火墙 多层架构 Hessian JavaEE 车站查询

A Solution That Allows Web Applications to Transverse Multi-Level Firewalls to Access Enterprise Data Resources

LIU Yan-Yan[1], LI Zhong-Yan[2]
(1. College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China)
(2. Weifang Bus Terminal Station, Weifang Transport Company, Weifang 261041, China)

Abstract: This paper analyses the enterprise network structures. According to the situation that firewalls protect enterprise networks with strict restrictions on external accesses, this paper proposes a software framework that allows web applications to traverse multi-level firewall to access enterprise data resources. Taking the bus station passenger transport schedule query as the example, this paper describes the application system developed with this software framework based on the lightweight JavaEE platform, and presents the detailed implementation methods and procedures. This paper also analyses the application server and describes methods to improve the application server performance. The solution system is easy to extend and maintain with high flexibility. Practical applications prove that the solution has good popularization and application values.

Keywords: firewall traversal; multi-layer framework, Hessian; JavaEE; Bus Station Query

1　引言

企业构建的内部网络，按照业务的不同而分成许多不同的子网。子网通过三层交换机、路由器、防火墙等网络设备连接起来。子网间的相互访问通常由防火墙控制，防火墙将企业网划分成多个相互隔离的子网，对不同的子网实施不同的安全策略。从而使网络在拓朴结构上产生分层效果。

在这里，我们将企业网分为三个层次：核心子网，应用子网，外网。核心子网用于企业日常的生产运营，存有企业关键的数据资源，因而受到防火墙的严格保

① 收稿时间:2009-05-11

护；应用子网将依据核心子网提供的数据，开展管理、办公、服务等活动，该类子网内也保存有相关的数据资源，比如办公文件，人事、工资数据等，防火墙将限制外部用户对应用子网内部的访问；外网则面向互联网，对公众提供企业服务，受到防火墙限制最小，相应地它访问内部网络的权限也最小，通常防火墙会禁止它直接访问任何内部网络的数据资源。

这种由防火墙基于“深度防御”的原则[1]划分的分层网络，我们称之为安全网络环境。这种网络的拓朴结构如图 1 所示。

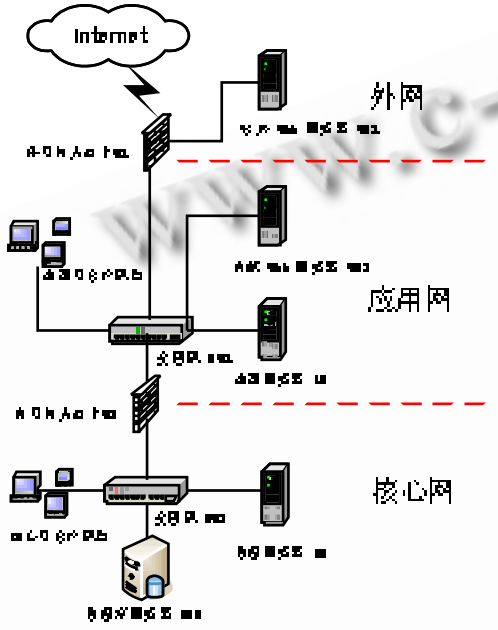

图 1　由防火墙分层的网络结构图

2　Web 应用程序跨越多级防火墙

在安全网络环境下，企业要开展互联网业务，就要允许 Web 应用系统访问各层子网的数据资源乃至核心子网的生产数据。防火墙禁止外部系统直接访问内部主机，但它可以被配置为允许外部系统跨越子网边界访问目标子网的本地代理。因此，Web 应用系统需要通过代理来间接访问各个子网。禁止外部访问的各个子网要对外提供服务时，均可设置本地代理，由子网内的本地代理负责接收外部请求，访问子网内数据资源，然后发送给外部用户。在这里，我们根据所在子网的不同而指定不同的代理名称。如，核心子网的代理称为数据服务器(DS)，应用子网的代理称为应用服务器(AS)。对于穿越多个防火墙访问核心子网的情况，实际上就是建立一个代理链，形如：Web ->AS->…AS->DS。

本文基于 Spring 开源框架实现安全网络环境下 Web 应用程序对企业数据资源的分层访问。Spring 的远程通讯协议主要有以下几种：RMI，Web Service，Spring HTTP，Hessian，Burlap。通过对比分析，决定使用 Hessian 协议。Hessian 基于 HTTP/二进制协议，能实现类似 Web 服务的功能，具有跨语言集成能力，支持多种客户端语言，传输大数据块时性能很高，非常适合企业内部通讯。

3　实现案例

这里以汽车站客运班次查询业务为例，来介绍 Web 应用系统跨越多层防火墙访问核心子网数据资源的方法。汽车站业务系统中一个常见功能是班次查询，即客户通过输入一个站点名来查询经过该站点的所有班次的信息。下面，我们建立了 4 个应用程序：

stationpubweb：互联网服务程序，为旅客提供在线查询、订票、购票等服务。位于图 1 中 WS1 位置。

stationbizweb：在线业务管理程序，为车站办公子网、上级主管部门、下属车站以及合作单位提供服务。位于图 1 中 WS2 位置。

stationappserver：应用服务器，用于接收 Web 系统请求，连接 stationdataserver，获取数据，并以服务形式提供给请求者。位于图 1 中 AS 位置。

stationdataserver：数据服务器，用于接收应用服务器/Web 系统的请求，从数据库服务器(DBS)获取数据，提供给请求者。位于图 1 中 DS 位置。

4 个程序基于 Spring 框架开发，其中 Web 应用程序还使用了 Struts2 框架。程序间以 Hessian 通讯[2]。stationpubweb,stationbizweb 被称为前端 web 系统；stationappserver,stationdataserver 被称为后端服务系统。

下面的顺序图通过来自 stationpubweb 的一次查询请求被服务的过程，展示了 4 个程序的运作流程。

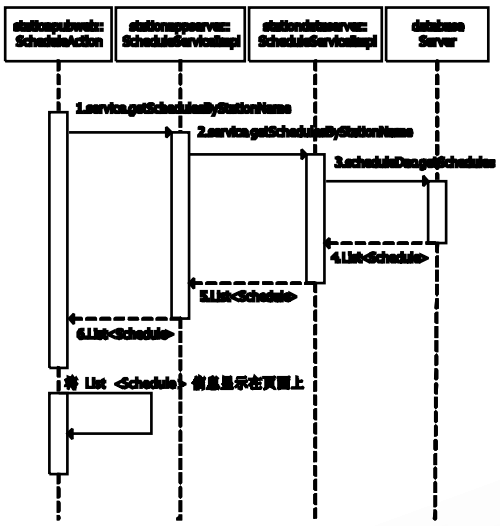

图 2　Web 系统访问数据资源顺序图

基于 Spring 面向接口的开发原则，设计程序接口和类，减少层间耦合[3]，程序中用到的主要接口及类如图 3 所示：

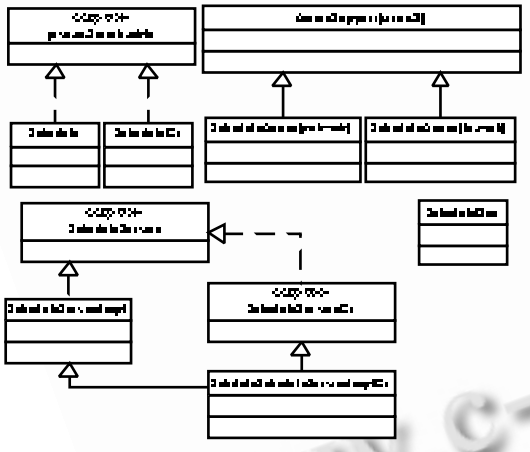

图 3　系统主要接口及类图

3.1 系统通讯规范

系统各部分通讯规范定义为两个接口和两个值对象。

3.1.1 向外网提供的服务接口

```
public interface ScheduleService {
    public List<Schedule>
    getSchedulesByStationName(String name);
}
```

该接口中的方法接收“站点”名称参数，执行后返回经过该站点的所有班次的基本信息列表。

3.1.2 向内网提供的服务接口

```
public interface ScheduleServiceEx extends
ScheduleService {
    public List<ScheduleEx>
getSchedulesByStationNameEx(String name);
}
```

它除了继承班次服务接口的基本信息列表，还可以由本身的方法提供详细的班次信息列表。

3.1.3 值对象

上述接口的方法返回 Schedule / ScheduleEx 类型的值对象列表。

Schedule 类包含班次名，终点站，总可售票数，剩票数同，查询站点，查询票价等旅客查询必须提供的班次信息。

ScheduleEx 类继承了 Schedule 类，它还包含终点票价，里程，车号，运行计划，运行线路等更详细的班次信息。

按照以上定义的系统通讯规范，后端服务系统就可以向前端 web 系统提供不同的服务，使得服务可以分层。服务接口和值对象规定了各层之间的通讯规约，各层必须遵守服务接口的方法调用规则，调用后返回规定的值对象。下面分别介绍 4 个程序的具体实现。

3.2 stationdataserver 数据服务器

stationdataserver 实际上是一个 Hessian 代理，它位于核心子网内，外部请求只有通过它才能获取核心子网内数据库服务器中的数据，它提供了两种服务。

3.2.1 /ScheduleService 服务提供基本班次信息列表

程序的配置文件 services-servlet.xml 规定了 /ScheduleService 服务由 ScheduleServiceImpl 类提供，该实现类需要实现 ScheduleService 服务接口。ScheduleServiceImpl 类的主要代码如下所示：

```
public class ScheduleServiceImpl implements
ScheduleService {
 public List<Schedule>
 getSchedulesByStationName(String name){
        ScheduleDao scheduleDao = new
ScheduleDao();
        List<Schedule> schedules = new
ArrayList<Schedule>();
        schedules=scheduleDao.getSchedules
```

```
(name);
    return schedules;
    }
}
```

本类的 getSchedulesByStationName 方法通过 ScheduleDao 类查询数据库中的班次时刻表、线路表、站点表、票价表等数据表，将查询得到的每一个班次的数据组装起来，形成班次值对象 Schedule，将所有 Schedule 值对象加入 List 列表中，把列表返回给 ScheduleServiceImpl 类，由该类向请求 /ScheduleService 服务的客户提供基本班次信息列表。

3.2.2 /ScheduleServiceEx 服务提供详细的班次信息列表

/ScheduleServiceEx 服务用于内部查询。它由 ScheduleServiceImplEx 类提供，如图 3 所示，该类继承 ScheduleServiceImpl 类并实现 ScheduleServiceEx 接口。它的 getSchedulesByStationNameEx 方法的实现过程与第 3.2.1 节类似。它既能向客户端提供基本班次信息列表，也能提供详细班次信息列表。系统只需使用该服务就能实现两种类型的班次查询。可供内部用户按授权情况分别使用。

3.3 stationappserver 应用服务器

应用服务器接收客户端班次查询请求后，将调用数据服务器提供的服务接口方法，获取班次列表信息，并将列表返回给客户端。程序的配置文件 services-servlet.xml 定义了两种服务，与数据服务器提供的服务相对应。

3.3.1 /ScheduleService 服务

客户端向应用服务器请求 /ScheduleService 服务时，应用服务器接受请求并由 ScheduleServiceImpl 类处理，该类的 getSchedulesByStationName 方法关键代码如下：

```
return service. getSchedulesByStation Name (name);
```

service 是 ScheduleServiceImpl 类的属性，Spring 将 HessianProxyFactoryBean 对象注入 service 属性，使之成为 Hessian 客户端代理[4]，该代理将访问stationdataserver 的/Schedule Service 服务，通过配置该代理的 serviceUrl 属性，形如 http://www.../stationdataserver/services/ScheduleService，使得 service 可以访问指定的 stationdataserver，获取 List<Schedule> 基本班次信息列表。

3.3.2 /ScheduleServiceEx 服务

/ScheduleServiceEx 服务由 ScheduleServiceImplEx 类提供，如图 3 所示，该类继承了 ScheduleServiceImpl 类实现了 ScheduleServiceEx 服务接口。既可查询基本班次信息，也可查询详细班次信息。

3.4 Web 前端系统

3.4.1 stationpubweb

本程序接受用户输入的站点，然后调用应用服务器的/Schedule 服务，接收返回的基本班次信息列表，并将列表显示在页面上。

前台页面输入查询的站点提交后，控制转移给 ScheduleAction 类，它负责提供班次列表信息。该类拥有的 service 属性由 Spring 根据配置注入 ScheduleService 接口类型的远程服务代理类。在 ScheduleAction.execute 方法中通过 service 属性实现信息获取，如下代码所示：

```
public String execute() throws Exception {
List<Schedule> schedules = new ArrayList <Schedule>();
schedules=service.getSchedulesByStationName( name);
 if (schedules != null){
        ActionContext.getContext().
getSession().put("schedules" , schedules);
        return SUCCESS;
}else{ return ERROR; }
}
```

存入 session 的 schedules 列表将被显示在查询结果页面的表格中。

3.4.2 stationbizweb

业务管理 Web 应用系统，实现流程与 stationpubweb 非常类似，只是它查询返回的列表类型为 List<ScheduleEx>，比公布给外网旅客的列表 List<Schedule>提供更详细、更全面的班次信息。

4 应用服务器分析

防火墙通常对企业数据资源严格保护，只允许特定 IP 地址访问数据库，这些特定 IP 地址一般是应用服务器(代理)。应用服务器是数据库与外界联系的通

道，这就既保证了网络的安全，又对外提供了企业的核心数据。

在这里，应用服务器具有路由、转发的功能，通过修改应用服务器服务实现类中 service 属性所代表的 Hessian 代理的 serviceUrl 属性，就能路由到不同的应用服务器/数据服务器请求不同的服务，并最终将请求的结果转发给客户。

应用服务器具有以下重要的特性：

(1) 应用服务器级联：对于多级防火墙，每个防火墙后面都有一台应用服务器。跨越多级防火墙的多个应用服务器之间的连接就形成了级联。通过级联转发，将请求传递到最深层的子网中，并将核心数据库产生的响应反馈给前端 Web 应用系统。

(2) 应用服务器缓存：当应用服务器需要服务的客户大量增加时，建立应用服务器缓存，能够显著提高它的性能。在本文案例中，stationappserver 可以建立站名、班次列表缓缓存，当应用服务器接收到请求时，将在缓存中查找传入的站名参数，如果与缓存中已查询过的站名列表项相匹配，就可以直接从应用服务器获取对应的班次信息列表，而不必向深层网络发送请求，从而大大加速查询。即便是考虑到数据库不断的更新，将该类查询的缓存生命周期设定为 1 分钟，也会大大提高应用服务器并发响应能力。

(3) 应用服务器集成：从各个子网的数据库中提取企业各种运营数据信息，加工整理，作为服务提供给更高层的前端系统，从而无缝集成整个企业网的各种业务，创建企业网应用系统平台。

应用服务器不仅是一个管道，除了数据资源收集、整合之外，它还要对访问进行控制。应用服务器在封闭的防火墙上打开了企业数据资源通向外界的通道，这使得保证应用服务器安全显得非常重要。应用服务器不但应根据 Web 前端系统的不同而提供不同的接口和规约以屏蔽前端 Web 系统获知企业内部系统的详细信息，而且应通过各种权限机制对访问加以限制。在轻量级 Java EE 应用系统中建议使用 Spring Acegi 对访问进行控制[5]，限于篇幅不再讨论。

5 结语

本文讨论了轻量级 Java EE 应用系统的物理分层架构。采用这种架构，不但在保证企业网络高安全性的前提下，实现了企业数据资源的分层访问，而且也为实现系统的高并发访问奠定了基础。

轻量级 JavaEE 应用系统中的每层应用程序部署到独立的 Tomcat 服务器，当某层应用负载较重时，可以使用 Apache 和 Tomcat 集群，对系统进行水平扩展，能够使系统承担更大的访问量。而软件系统不需要或需要很少的改动，具有很高的灵活性和适应能力。

轻量级 Java EE 多层体系结构，不需要部署 EJB 服务器，只需要部署 Web 服务器，使得系统管理更简单，可维护性更好。

读者可以从 http://download.csdn.net/source/1221746 下载本文的源代码，以便深入了解本文所讨论的方案。

参考文献

1 Zou CC, Towsley D, Gong WB. A Firewall Network System for Worm Defense in Enterprise Networks. Technical Report: TR-04-CSE-01. Amherst. Univ. Massachusetts. 2004.

2 Mak G. Spring Recipes: A Problem-Solution Approach. Berkeley, 2008.588－591.

3 Harrop R, Machacek J. Spring 专业开发指南. Redsaga 小组.北京:电子工业出版社出版, 2006.125－126.

4 Walls C, Breidenbach R. Spring in Action. 2nd ed. New York: Manning Publications, 2007. 316－322.

5 陈雄华.精通Spring 2.x—企业应用开发详解.北京:电子工业出版社, 2007.553.