

**CanToolApp 2.0系统架构文档**

**现代软件工程开发小组第19组**

**2017年11月**

目录

[1. 系统框架设计 2](#_Toc30691)

[1.1 Controller (控制器)层 2](#_Toc10429)

[1.2 Model(业务逻辑)层 2](#_Toc5565)

[1.3 View(视图)层 2](#_Toc12741)

[1.4 DWR(AJAX)框架 3](#_Toc20353)

[2. 系统开发环境 4](#_Toc4103)

[3. 系统文件结构说明 5](#_Toc29624)

[3.1 整体结构 5](#_Toc16781)

[3.2 Java源代码结构 5](#_Toc29508)

[3.2.1 模块功能 7](#_Toc15021)

[3.2.2 主要类功能 8](#_Toc1078)

[3.3 Web源代码结构 9](#_Toc26622)

[4. 系统框架实现详细设计 10](#_Toc23661)

[4.1 请求响应流程 10](#_Toc15613)

[4.2 Action(控制器)层 11](#_Toc18976)

[4.3 Service(业务逻辑)层 11](#_Toc7411)

[4.4 Dao(数据访问)层 13](#_Toc25019)

[4.5 Bean(实体类) 15](#_Toc6094)

[4.6 主要配置文件说明 16](#_Toc29634)

[4.6.1 web.xml配置说明 16](#_Toc785)

[4.6.2 applicationContext.xml配置说明 19](#_Toc21132)

[4.6.3 dwr.xml配置说明 21](#_Toc30480)

1. **系统框架设计**

本系统实现采用Spring +JDBCTemplate框架整合开发，其中使用JDBCTemplate为数据持久层，Spring框架为MVC整合框架，负责整个系统的MVC的实现。系统还使用了DWR框架作为AJAX技术的实现 ，MVC架构图如下图1所示:

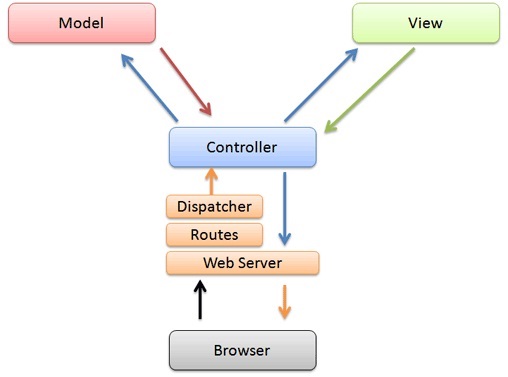


图1 MVC架构图

**1.1 Controller (控制器)层**

Controller层由Spring框架实现，所有请求交由Spring提供的DispatcherServlet前端控制器统一转发，前端控制器根据请求路径，再通过注解映射，会调用具体的业务逻辑控制器(用开发人员实现)实现前后台交互和页面转发的功能。

**1.2 Model(业务逻辑)层**

Model层包括Service(业务逻辑)层和Dao(数据访问)层，其中Service层主要包括所有的业务逻辑处理，各部分资源的调用，而Dao层主要包括所有的数据访问操作，JDBCTemplate作为Dao层数据库的持久层实现。

**1.3 View(视图)层**

View层主要包括Jsp和XML，JSON，Excel两种实现方式。

**1.4 DWR(AJAX)框架**

DWR作为AJAX实现的一个优秀框架，主要用于实时功能的实现，由Spring整合到系统当中。

1. **系统开发环境**

操作系统：Windows 8（64位），

浏览器：IE8，IE9以上版本

JDK：JDK1.8

开发工具：myEclipse 10.0

服务器：Tomcat 7.0

数据库：MySQL server 5.5

1. **系统文件结构说明**

**3.1 整体结构**

在项目整体结构中，主要包含3部分内容，src目录为java源代码的存放目录，该目录下的内容在编译后成为.class格式的字节码文件，运行时由服务器解释执行；JRE System Library、Java EE 5 Libraries和Web App Libraries为项目开发所需的基本类库，包括所需的基本数据结构类和运行时类库等；WebRoot为web根目录，主要包括jsp文件、web配置文件以及网页中的css样式文件、js文件、图片等；项目编译后的目录结构与WebRoot下目录结构相同，src目录下的java文件编译后全都存放在WebRoot下的WEB-INF/classes文件夹下。整体结构如下图2所示：

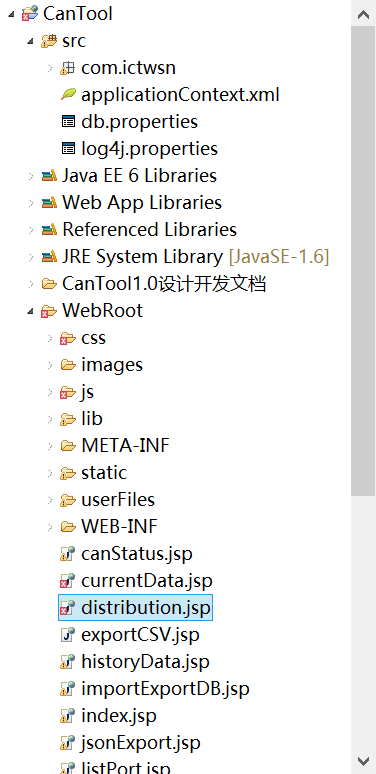


图2 项目整体结构图

**3.2 Java源代码结构**

Java源代码接口是指src目录下的java源码的目录层次结构，在src目录下主要根据不同java文件实现的功能，结合MVC框架的结构，使用文件夹将java文件进行分类，主要分为6类。如下图3.1,图3.2所示。

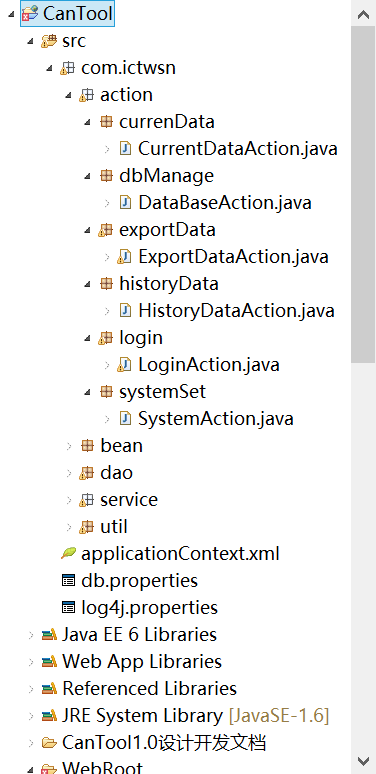


图3.1 Java源代码结构图

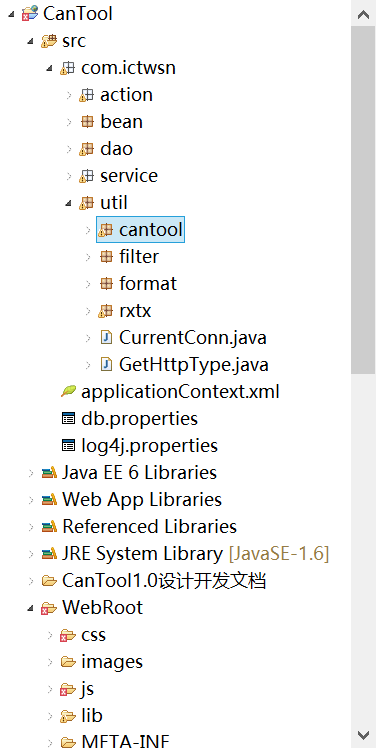


图3.2 Java源代码结构图

**3.2.1 模块功能**

DBManage模块：数据库管理模块，负责数据库树状图的显示，数据库的XML/JSON格式的导入导出等功能。

CurrentData模块：实时数据模块，负责实时数据库的图表或者仪表盘显示。

HistoryData模块：历史数据模块，负责数据的组成展示、历史数据的条件查询。

Login模块：端口连接模块，负责端口的检测和接入/断开功能。

ExportData模块：数据导出模块，负责实时数据的xls格式导出进行。

SystemSet模块：负责系统参数配置，查询和修改。

附加模块：

Util模块：工具模块，主要用来存放一些工具类，如cantool信息的解码与编码模块，拦截器模块，数据格式化处理模块，以及串口检测与接收模块。

**3.2.2 主要类功能**

(1)业务逻辑控制器

业务逻辑控制器位于com.ictwsn.action.\* 文件夹下面，action表示控制器，\*表示具体模块的名称，不同模块的控制器存放在不同的文件夹下面。

(2)实体类

实体类位于com.ictwsn.bean目录下面，是对数据的抽象。

(3)数据访问层

数据访问层位于com.web.dao文件夹下，其中dao下放的是接口，dao.impl下放的是接口的实现类，该层封装的都是对数据进行存取操作的方法，几乎不包含业务逻辑的处理，调用Hibernate或JDBC提供的接口访问数据库。该层以数据库表或者试图为划分依据，通常每个表或试图对应一个Dao类，所以在dao下不再继续划分。

(4)公共接口

公共接口位于com.web.dao.publicDao下面，包含一些公用的接口方法。

(5)业务逻辑层

数据访问层位于com.web.service.\*文件夹下，service代表的是业务逻辑层，\*代表的是具体的模块，其中service.\*下放的是接口，service.\*.impl下放的是接口的实现类。该层封装的是对用户请求的业务逻辑处理，以模块为划分依据，由于模块间的关联性，每个模块下可能存在多个业务逻辑处理的类，所以在service和service.impl下需进一步细分。

(6)工具类

工具类位于com.ictwsn.util文件夹下，该文件夹主要包括一些对数据处理的通用方法，这些方法不是某个模块特有的方法，所以不能包含在某个模块中，如对“2013-11-10”数据在字符串和时间日期类型间相互转换的函数等等。

**3.3 Web源代码结构**

Web源代码存在于WebRoot目录下面，存放不会被编译的文件，本系统中的目录结构如下图4所示。

css：存放网页布局、美化用的css样式表文件；

images：存放网页中使用到的图片；

js：存放验证和其他js代码；

userFile：存放用户的文件，包含导入和导出文件存放的路径；

static：存放用到的插件代码等；

WEB-INF：存放项目的xml配置文件等；

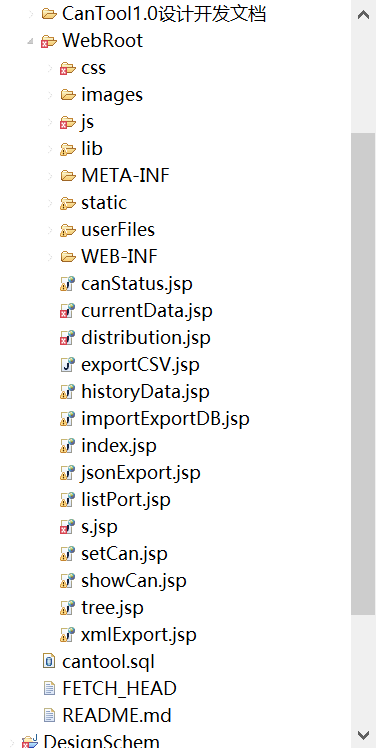


图4 Web源代码结构

1. **系统框架实现详细设计**

**4.1 请求响应流程**

用户通过浏览器发出一个请求时，请求首先到达Spring的核心控制器，核心控制器根据请求路径判别：

如果用户请求的是某一个jsp页面，则响应回一个jsp页面给浏览器；

如果用户请求的是一个动态资源，则根据请求路径分发给具体的业务逻辑控制器，业务逻辑控制器调用业务逻辑层进行业务逻辑处理；其中，业务逻辑层需调用由Hibernate实现的数据访问层进行数据存取操作，业务逻辑层本身对数据进行操作完成业务逻辑的处理；最终处理完成后返回动态数据，并由业务逻辑控制器返回视图。如下图5所示：

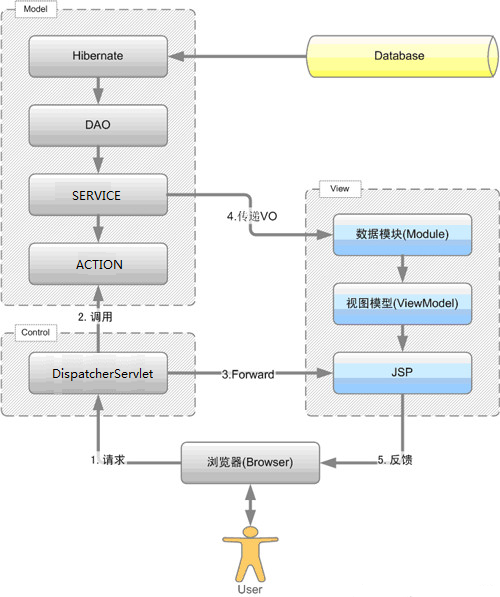


图5 请求响应流程图

**4.2 Action(控制器)层**

Action层主要指开发人员编写的业务逻辑控制器，在请求到达Spring的核心控制器后，会根据请求的路径调用相应的业务逻辑控制器完成对请求的处理和发出响应。

对一个控制器类来说，其命名根据模块或者功能来命名，结尾附加Action用以说明该类为控制器，并使用@Controller来标注，类的方法作为对请求处理的程序，在方法前用@RequestMapping(“\*”)作为请求的映射字符串标注，\*代表请求的路径。当此路径与请求路径一致，则调用该标注后的方法处理请求。在控制器内，可以引用Service层的资源，通过@Resource标注来引入。控制器大多只完成接收参数、调用Service处理和返回视图的工作。具体代码结构如下图6所示：

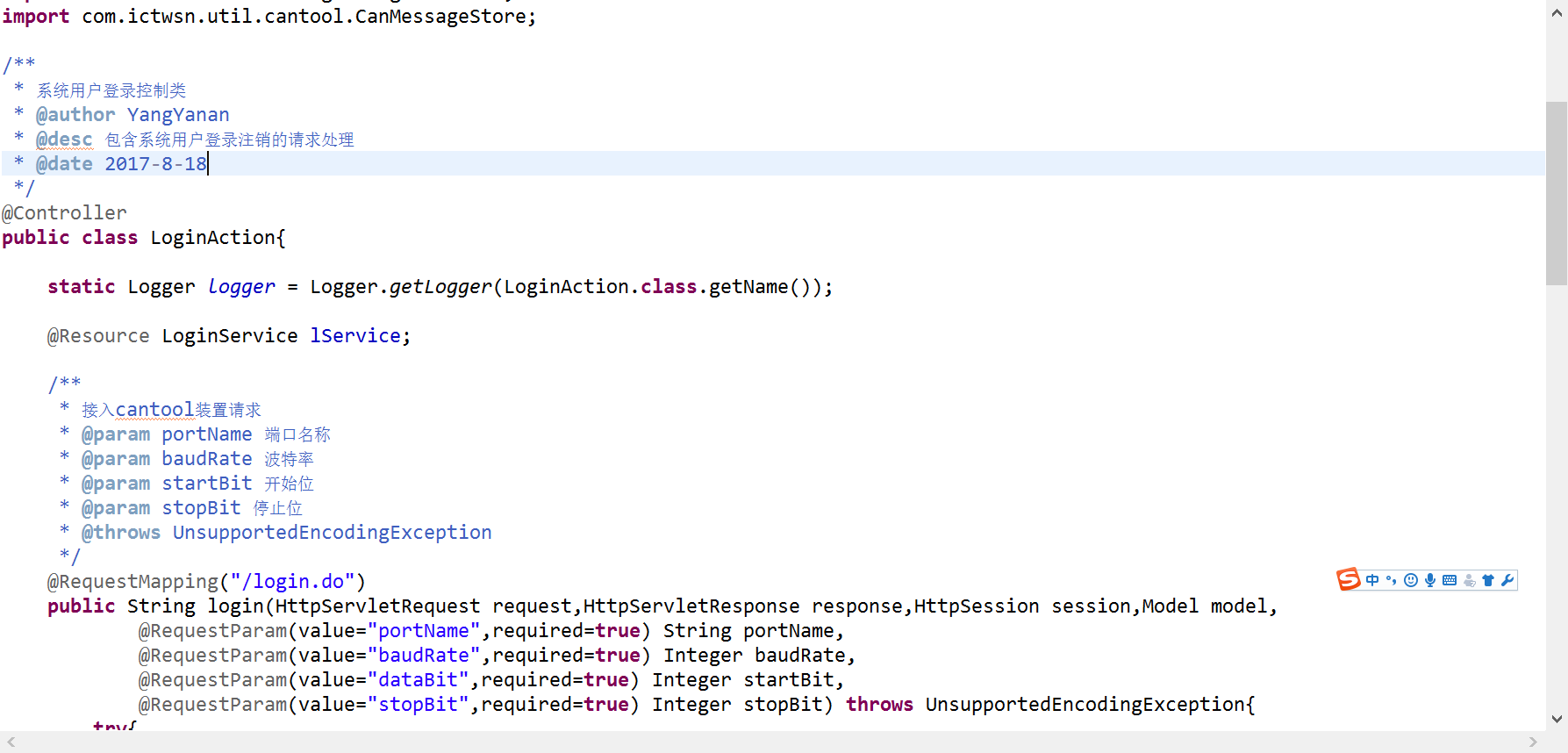


图6 Action代码结构说明

**4.3 Service(业务逻辑)层**

Service层主要指用于业务逻辑处理的层级，是Model层的一部分，该层由Action层直接调用，作为Action的引用资源。该层主要完成对数据的业务逻辑处理功能，由业务逻辑接口和接口实现类两部分组成。

1.接口说明

Service层接口是对Service完成功能的基本展示，主要包括对方法的声明，具体到返回值类型和参数个数以及类型。接口的命名也是以模块或者功能来命名，附加Service结尾，用以说明该类为业务逻辑层类。在Action层引用Service层资源时，也是以一个Service接口为单位，以一个Service接口名称来声明并实例化对象。具体代码结构如下图7所示：



图7 Service接口代码结构图

2.实现类说明

Service接口实现类是对某一个Service接口方法实现的类，其命名为实现的接口\*Service后附加Impl作为类名，并使用implements关键字实现该接口，该类中必须包含并实现所有接口中声明的方法。该类要以@Service标注在类前，用以说明该类为业务逻辑层的类，作为Spring扫描的对象，产生资源映射，作为Action中资源引用的资源实例化的对象。该类中一般只对数据做业务逻辑的处理，对于数据访问则通过@Resources注解引入Dao层资源进行处理。代码接口如下图8所示：



图8 Service接口实现类代码结构图

**4.4 Dao(数据访问)层**

Dao层是对数据库进行访问操作的层级，是Model层的第一层，由Service层直接调用，作为Service层的引用资源。该层主要完成对数据库的存取等操作，一般不包含业务逻辑处理，是直接调用Hibernate数据持久化的一层。

1.接口说明

同Service接口相似，该Dao接口主要完成的是对接口方法的声明，其命名方式主要以数据表、视图或者实体类来命名。代码结构如下图9所示：



图9 Dao接口代码结构图

2.实现类说明

Dao实现类是对某一个Dao接口方法进行实现的类，其命名为\*Dao附加Impl结尾作为类名，使用implements关键字实现该接口，Dao实现类中必须实现Dao接口中所有声明的方法。Dao实现类使用extends关键字继承对数据库进行操作的基类，该基类中用于对数据库进行操作的JDBCTemplate接口方法或JDBC接口方法，使得Dao实现类可以直接操作数据库。Dao实现类用@Repository注解作为Spring扫描的标志，用以初始化该Dao层的资源映射，来实例化Service实现类中的资源。Dao实现类的资源即为其从基类继承来的Hibernate或者JDBC数据库访问对象。代码结构如下图10所示：



图10 Dao实现类代码结构图

**4.5 Bean(实体类)**

1.Java实体类说明

Java实体类是一种数据结构，其抽象来源于数据库表、视图或者业务逻辑处理过程中用于计算的数据模型。类中一般包含表达数据模型的私有属性以及对属性进行访问的接口方法。代码结构如下图11所示：



图11 实体类代码结构图

2.JDBCTemplate映射文件说明

由于JDBCTemplate框架是实现数据持久化的框架，在使用JDBCTemplate框架时，需要在Spring的配置文件applicationContext文件中配置对应的事务处理对象，在实例和调用模块建立相应的映射，以便在使用JDBCTemplate框架时直接操作实体类对象以实现数据的持久化操作。

SpringMVC的applicationContext.xml配置文件如下图12所示：

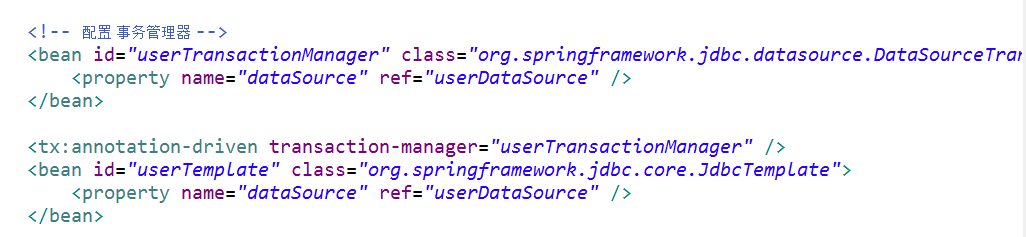


图12 JDBCTemplate配置结构图

**4.6 主要配置文件说明**

**4.6.1 web.xml配置说明**

web.xml是web应用程序最根本的配置文件，所有框架的配置都需要在web.xml中进行配置说明才能发挥效用。

(1)应用上下文配置

应用上下文配置是指在Web应用中上下文环境的配置，主要表现方式为xml配置文件。如下图13所示，<param-name>指明了配置的名称为contextConfigLocation，即对Spring的配置，<param-value>指明了Spring配置文件的位置，即在java源码的根目录。

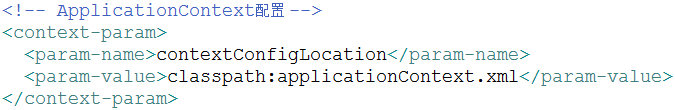


图13 应用上下文配置结构图

(2)Spring框架配置

Spring框架配置中主要是对Spring核心控制器的配置，以Servlet的形式将核心控制器配置进来。<servlet>中指明对请求处理的servlet为Spring框架中的DispatcherServlet，<servlet-mapping>中指明使用该servlet进行处理的请求的格式。/\*代表所有的请求，也就是浏览器提交过来的所有请求，都会交由Spring框架提供的DispatcherServlet进行处理，从而实现了其核心控制器的功能，将请求根据路径分发为具体业务逻辑控制器处理。配置如下图14所示：

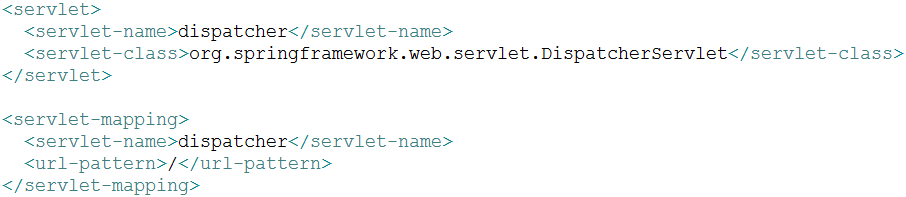


图14 Spring框架配置

(3)dwr框架配置

dwr作为ajax实现的技术框架，必须要在web.xml中配置后方能发挥效用。如下图15所示，dwr的配置也是通过servlet的配置实现的。同Spring的配置，dwr的配置<servlet-mapping>说明了进行ajax操作的请求满足的路径格式为/dwr/\*，即以dwr/开头的请求路径将使用ajax技术实现。dwr的<servlet>配置说明了ajax操作是由uk.ltd.getahead.dwr.DWRServlet实现的。



图15 DWR框架配置

(4)监听器配置

web.xml中配置的监听器包括Spring的ContextLoadListener，完成启动Web应用时，自动装配Spring的配置信息；Spring的IntrospectorCleanupListener，完成web应用关闭时，清除JavaBeans、Introspector的功能；第三个为com.web.util.Listener包下自定义监听器的配置信息。如图16所示：



图16 监听器配置

(5)过滤器配置

过滤器的配置是对请求进行过滤处理组件的配置，如下图17中配置了Spring提供的编码格式过滤器。<filter-mapping>说明过滤器将对/\*，即所有的请求进行编码格式化处理；<filter>中的<init-param>说明，所有的请求都会格式化为UTF-8编码格式。

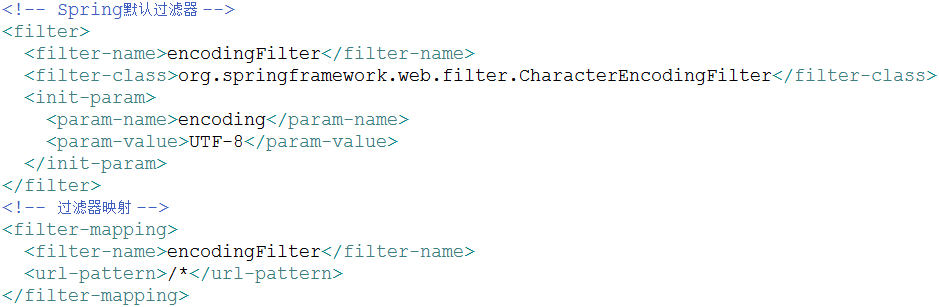


图17 过滤器配置

**4.6.2 applicationContext.xml配置说明**

applicationContext.xml是Spring的主要配置文件，里面包含了注解的扫描、数据源配置、JDBCTemplate框架整合配置、数据库访问接口配置以及dwr业务等配置。

(1)自动扫描注解配置

在Spring3.0框架中提供了注解这一功能，主要包括@Controller

(控制器)、@Service(业务逻辑)、@Repository(数据访问)、@Resource

(资源注入)等常用注解，使用注解可以避免在配置文件中对资源的大量配置，通过对指定文件夹中注解的自动扫描，会自动建立一个资源映射库，以供资源调用。配置如下图18所示：



图18 自动扫描注解配置

(2)数据源配置

Spring对MySql数据源的配置如下图19所示，其中driverClass代表加载的驱动类；jdbcUrl代表数据库的访问字符串；user代表用户名；password代表密码；minPoolSize代表最小数据源池链接数量；maxPoolSize代表最大数据源池链接数量等等。



图19 数据源配置

(4)数据访问接口配置

数据访问接口是提供给Dao数据访问层对数据库进行访问存取操作的接口，在Spring的配置文件中配置了两种访问接口。配置如下图21所示：

第一，JDBC访问接口。<bean>的id为userTemplate，其对应的接口提供类为Spring的JdbcTemplate类，dataSource指向的是上面提到的MySql数据源的id---userDataSource；访问数据库时通过userTemplate对象调用其各种方法即可。



图20 数据访问接口配置

**4.6.3 dwr.xml配置说明**

dwr.xml在WEB-INF目录下，是对使用了dwr框架进行ajax异步访问的方法的说明。

如下图22所示，<convert>中描述的是在使用dwr框架的方法中返回值的类型的说明，其中match是对应的类，converter是对应类转换后的形式。<create>中定义的是使用dwr框架的方法，其中javascript是使用ajax时调用方法的对象；<param>子标签中name属性指明了方法的来源，beanName代表通过资源名称导入方法，value是在Spring中映射的资源的名称，该资源一般为Service层资源，此时在给Service层加注解时还需再@Service后加入一个标识Service的名称，如@Service(“lService”)；<include>标签中指明的是该资源中包含进来的使用dwr方式的方法的名称，如不指定，则该资源下的全部方法均可使用dwr方式进行异步请求访问。

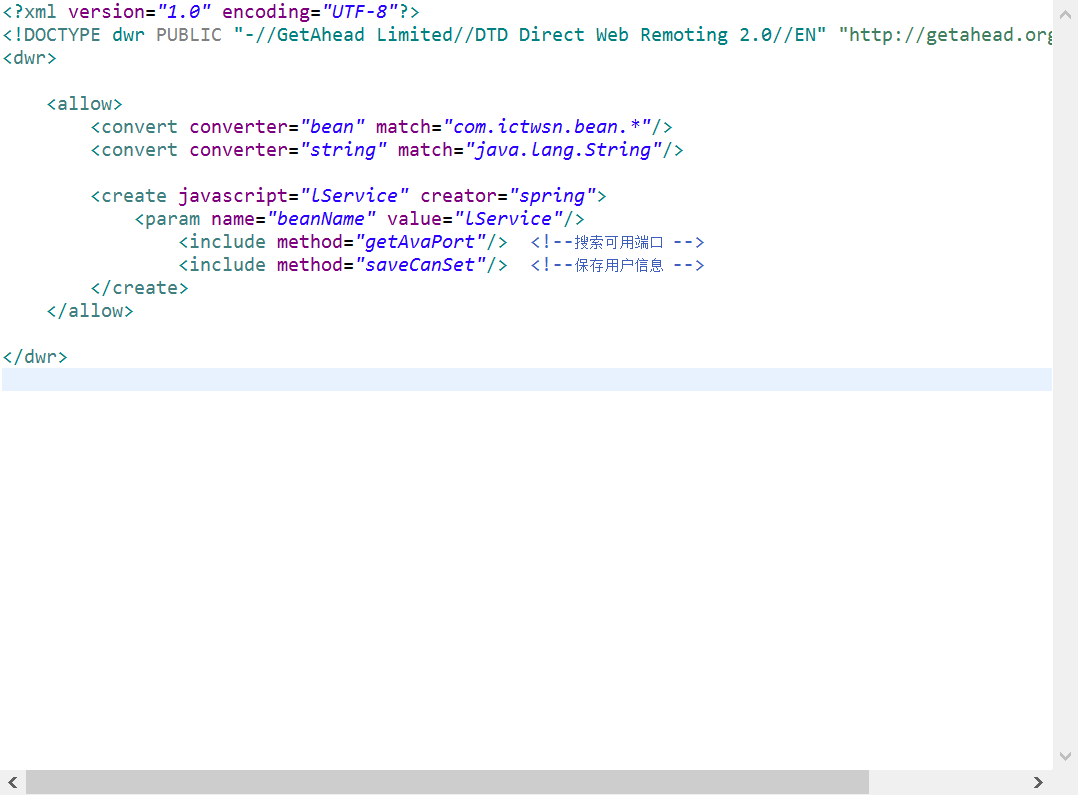


图21 dwr配置