DWR (Direct Web Remoting ) 官方网址： <http://getahead.org/dwr>

[JavaScript的数据类型与Java数据类型的关系 1](#_Toc258026564)

[什么是DWR 2](#_Toc258026565)

[DWR安装 3](#_Toc258026566)

[常用<init-param>参数 3](#_Toc258026567)

[常用<init-param>参数－安全参数 3](#_Toc258026568)

[常用<init-param>参数－Ajax服务器加载时保护参数 4](#_Toc258026569)

[常用<init-param>参数－其他参数 4](#_Toc258026570)

[常用<init-param>参数－日志配置 4](#_Toc258026571)

[dwr.xml的配置 5](#_Toc258026572)

[一个servlet，多个dwr.xml配置文件 5](#_Toc258026573)

[Test1这个JavaBean的源代码： 6](#_Toc258026574)

[JSP文件的编写 6](#_Toc258026575)

[DWR Javascript库中常用函数的使用 7](#_Toc258026576)

[Java组件的编写和发布以及客户端引用 7](#_Toc258026577)

[DWR 实现机制 8](#_Toc258026578)

[1、DWR主要组成部分 8](#_Toc258026579)

[1. DWR前台引擎(Engine.js)的主要职责 8](#_Toc258026580)

[2、后台控制器(DWRServet)的主要职责 8](#_Toc258026581)

[3、业务组件执行器的主要职责 8](#_Toc258026582)

[4、数据类型转换器的主要职责 8](#_Toc258026583)

[5、跟踪调式DWR原码详细说明其执行步骤 9](#_Toc258026584)

[DWR配置说明 9](#_Toc258026585)

[1、DWR配置文件主要有以下四个部分组成 9](#_Toc258026586)

[2、dwr.xml的初始化配置部分Init 9](#_Toc258026587)

[3、dwr.xml的业务组件发布和调用规则配置部分Allow 9](#_Toc258026588)

[4、dwr.xml的业务组件方法签名配置部分Signatures 10](#_Toc258026589)

[6、JS开发注意事项 11](#_Toc258026590)

# JavaScript的数据类型与Java数据类型的关系

|  |  |
| --- | --- |
| JavaScript | Java |
| Array数组 | List、Collection、数组 |
| Boolean | Boolean |
| Object | Map、Java Bean |
| String | String |
| Date | Date |
| Numbers | Int、double、float等数值 |
| XML Dom | Dom |
| undefined | null |

# 什么是DWR

|  |
| --- |
| DWR是一种AJAX解决方案！  DWR包括一个java库，以及一套javascript，使得我们可以用一种非常简单的方式，在页面上使用javascript直接调用后台的java对象！  利用DWR可以在客户端利用JavaScript直接调用服务 端的Java方法并返回值给JavaScript就好像直接本地客户端调用一样(DWR根据Java类来动态生成JavaScrip代码)    DWR的javascript库包括几个部分：dwr的javascript引擎（即对AJAX调用的封装）、dwr提供的一些非常有用的辅助javascript函数库、以及dwr自动帮我们生成的专门用于调用后台JavaBean方法的javascript库！  Dwr在运行的时候，动态生成一个javascript库，这个库是对后台javabean调用的封装，我们可以直接使用这个库来实现直接调用JavaBean的目的！  **注意：是在运行的时候动态生成！** |

**DWR的组成**

|  |
| --- |
| * DWR主要由两部门组成。   + javascript与web服务器通信并更新web页；   + 运行在web服务器的Servlet处理请求并把响应发回浏览器。 |

# DWR安装

|  |
| --- |
| * 下载dwr.jar，并将其放入/WEB-INF/lib目录下 * 修改web.xml配置, 添加DWRServlet的映射   <servlet>  <servlet-name>dwr-invoker</servlet-name>  <servlet-class>org.directwebremoting.servlet.DwrServlet</servlet-class>  <init-param>  <param-name>debug</param-name>  <param-value>true</param-value>  </init-param>  </servlet>  <servlet-mapping>  <servlet-name>dwr-invoker</servlet-name>  <url-pattern>/dwr/\*</url-pattern>  </servlet-mapping>  注：  在DWR2.x里DwrServlets的类是org.directwebremoting.servlet.DwrServlet  在DWR1.x里DwrServlets的类是uk.ltd.getahead.dwr.DWRServlet |

# 常用<init-param>参数

## 常用<init-param>参数－安全参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 默认值 | 开始版本 |
| allowGetForSafariButMakeForgeryEasier | 设置成true使DWR工作在Safari 1.x , 会稍微降低安全性。 | false | 2.0 |
| crossDomainSessionSecurity | 设置成false使能够从其他域进行请求。注意，这样做会在安全性上有点冒险，参考一下这篇文章，在没有理解这个后果前不要设置成为false。 | true | 2.0 |
| debug | 设置成true使DWR能够debug和进入测试页面 | false | 1.0 |
| scriptSessionTimeout | scriptsession 的超时设置 | 1800000(30分钟) | 2.0 |
| maxCallCount | 一次批量(batch)允许最大的调用数量。(帮助保护Dos攻击) | 20 | 2.0rc2 和 1.1.4 |

## 常用<init-param>参数－Ajax服务器加载时保护参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 默认值 | 开始版本 |
| pollAndCometEnabled | 设置成true能增加服务器的加载能力，尽管DWR有保护服务器过载的机制。 | false | 2.0 |
| maxWaitingThreads | 最大等待线程数量。 | 100 | 2.0 |
| preStreamWaitTime | 对一个打开流前的反应，等待的最大时间 | 29000(单位：毫秒) | 2.0 |
| postStreamWaitTime | 对一个打开流后的反应，等待的最大时间 | 1000(单位：毫秒) | 2.0 |

## 常用<init-param>参数－其他参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 默认值 | 开始版本 |
| ignoreLastModified | 默认值支持最后修改，这样就允许服务器端对客户端请求较少资源。设置为true就能屏蔽支持。 | false | 2.0 |
| scriptCompressed | DWR能够执行简单的压缩，设置为true可以激活此功能。另外还有一个未公开的有关系的重要参数“compressionLevel”，此参数允许你配置压缩类型。查看这里得到更多详细资料 | false | 1.1 |
| sessionCookieName | DWR通过检查文档和提取当前session ID支持URL重写。一些servlet引擎使用非标准的cookie名。参数允许你改变默认值。 | JSESSIONID | 2.0 |
| welcomeFiles | 类似于web.xml的<welcome-file-list>标签 | index.html, index.htm, index.jsp | 2.0 |

## 常用<init-param>参数－日志配置

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 描述 | 默认值 | 开始版本 |
| logLevel | 值可以是：FATAL，ERROR，WARN(默认)，INFO，DEBUG | WARN |  |

# dwr.xml的配置

|  |
| --- |
| 1. 配置dwr，即在WEB-INF目录下，添加dwr.xml文件，让dwr知道在运行的时候应该给哪些JavaBean生成相应的javascript库！  * <allow> 标签中包括可以暴露给 javascript 访问的东西。 * <create> 标签中指定 javascript 中可以访问的 java 类，并定义 DWR 应当如何获得要进行远程的类的实例。 creator=“new” 属性指定 java 类实例的生成方式， new 意味着 DWR 应当调用类的默认构造函数来获得实例，其他的还有 spring 方式，通过与 IOC 容器 Spring 进行集成来获得实例等等。 javascript=“ testClass ” 属性指定 javascript代码访问对象时使用的名称。 * <param> 标签指定要公开给 javascript 的 java 类名。 * <include> 标签指定要公开给 javascript 的方法。不指定的话就公开所有方法。 * <exclude> 标签指定要防止被访问的方法。   <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE dwr PUBLIC "-//GetAhead Limited//DTD Direct Web Remoting 2.0//EN" "http://getahead.org/dwr/dwr20.dtd">  <dwr>  <allow>  <create creator="new" javascript="test1">  <param name="class" value="com.bjsxt.dwr.Test1"/>  </create>  </allow>  </dwr>  这个配置的意思是，要创建的是Test1对象的javascript库，而且这个库的名字叫test1，同时，这也是我们在JSP页面上调用这个对象的时候所使用的名称，请看下面的JavaBean代码和JSP实例： |

# 一个servlet，多个dwr.xml配置文件

|  |
| --- |
| <servlet>      <servlet-name>dwr-invoker</servlet-name>      <servlet-class>org.directwebremoting.servlet.DwrServlet</servlet-class>      <init-param>        <param-name>config-1</param-name>        <param-value>WEB-INF/dwr1.xml</param-value>      </init-param>      <init-param>        <param-name>config-2</param-name>        <param-value>WEB-INF/dwr2.xml</param-value>      </init-param>  </servlet>   * param-name的值必须以config开头。param-name可以有>=0个。如果没有param-name，那么将会读取 WEB-INF/dwr.xml。如果有大于零个param-name，那么WEB-INF/dwr.xml文件将不会被读取。 |

# Test1这个JavaBean的源代码：

|  |
| --- |
| package com.bjsxt.dwr;  public class Test1 {  public String sayHello(String name){  return "你好，"+name;  }  } |

# JSP文件的编写

|  |
| --- |
| * 首先要包含以下三个JS   <script src='dwr/engine.js'></script> <!--dwr中的核心引擎库-->  <script src=‘dwr/util.js’></script> <!--辅助工具函数库-->  <script src='dwr/interface/test1.js'></script> <!-- 实际上这个文件并不存在，这是由dwr在运行的时候动态生成的！test1这个名称，跟dwr.xml文件中配置的对应对象的javascript属性一致！而且，我们在使用的时候，直接使用test1这个名称，作为这个对象的引用。直接调用这个对象的方法：sayHello，这个方法的名称必须与JavaBean中的方法名称一致！sayHello方法有一个参数，同时返回一个值。我们可以直接传递参数，同时指定一个javascript函数来处理这个返回值  -->  其次，编写调用 java 方法的 javascript 函数  function checkRegist(){  loginName=document.f1.loginName.value;  RegistChecker.check(loginName,callback);  } |

* 如何传递参数
  + 如果要调用的方法有参数，则将参数放在前面
* 如何处理返回值
  + 定义一个函数来处理返回值，这个函数有一个参数，就是JavaBean方法的返回值

# DWR Javascript库中常用函数的使用

|  |
| --- |
| * 设置异步/同步模式   + 默认情况下，DWR以异步的方式跟服务器通信，即可以同时向服务器发送请求，如果想要改为同步的方式，可以调用：dwr.engine.setAsync(false); * 获得对某个页面元素对象的引用   + $(“元素的ID或名称”) * 对列表元素的支持   + dwr.util.removeAllOptions()   + dwr.util.addOptions()   + dwr.util.getValue()   + dwr.util.setValue()   + 级联下拉框的实现 |

# Java组件的编写和发布以及客户端引用

|  |
| --- |
| Java组件必须是一个具体类(因为接口、抽象类反射机制无法实例化)  实体类必须遵循Java Bean规范即实现Getter和Setter方法  配置文件中Allow部分的Create元素的JavaScript属性的值要唯一  如果存在Java Bean需要转换则要配置Allow部分的Convert元素  客户端引用业务组件要以dwr/interface/xxx.js的路径引用JS文件 |

# DWR 实现机制

## 1、DWR主要组成部分

## 1. DWR前台引擎(Engine.js)的主要职责

|  |
| --- |
| 截取每个后台组件的请求实现客户端的统一入口  将用于组件调用的JS参数对象重新包装组合  检测客户端浏览器的环境并选择最佳的数据提交方案  处理后台执行结果并通过执行回调函数通知客户 |

## 2、后台控制器(DWRServet)的主要职责

|  |
| --- |
| 装载Servlet时初始化DWR配置参数  实现后台业务组件调用的服务器端统一入口  分发各种不同类型的请求 JS文件请求 业务调用请求等  输出组件执行结果(JavaScript语句)到客户端浏览器 |

## 3、业务组件执行器的主要职责

|  |
| --- |
| 根据配置文件的参数确定后台组件的访问规则和生命周期  拆包前台传入的参数 包装组件执行结果  按参数类型匹配数据类型转换器  采用反射机制执行业务组件方法处理客户端请求  处理与其他系统的接口和协作 如：Spring Hibernate等 |

## 4、数据类型转换器的主要职责

|  |
| --- |
| 将一系列单个的参数转换成对应的Java对象  将Java对象转换成JavaScript对象 |

## 5、跟踪调式DWR原码详细说明其执行步骤

|  |
| --- |
| 说明后台组件对应的前台JS文件的结构  查看前台JS对象的参数包装后的内部结构  查看总控制器分发各种类型请求代码  观察业务执行器对浏览器传入参数的再次分类  以Bean转换器说明JavaScript和Java数据类型的转换机制  说明业务组件执行结果的JavaScript表现形式  前台引擎如何处理后台传入的结果并调用回调方法 |

# DWR配置说明

## 1、DWR配置文件主要有以下四个部分组成

|  |
| --- |
| 1、web.xml文件主要配置DWR的后台控制器和配置文件的载入(略)  2、[dwr.xml的初始化配置部分Init](#_2、dwr.xml的初始化配置部分Init)  3、[dwr.xml的业务组件发布和调用规则配置部分Allow](#_3、dwr.xml的业务组件发布和调用规则配置部分Allow)  4、[dwr.xml的业务组件方法签名配置部分Signatures](#_4、dwr.xml的业务组件方法签名配置部分Signatures) |

## 2、dwr.xml的初始化配置部分Init

|  |
| --- |
| Init是配置DWR主要是配置业务组件的生成规则和基本数据类型的转换器配置。DWR内置了4种业务组件生成规则new、null、spring、scripts。数据类型转换器DWR内置了Boolean等16种转换器。(一般情况下可以不用配置，如果配置重复可能会有无法预料的后果)  一、[**creator**](http://getahead.ltd.uk/dwr-demo/dtddoc/dwr10.dtd.html)**元素 主要用于前台JS业务组件对象的生成器如Spring、**  **New、Scripts等。**  **<creator id="spring“**  **class="uk.ltd.getahead.dwr.create.SpringCreator"/>**  **二、**[**converter**](http://getahead.ltd.uk/dwr-demo/dtddoc/dwr10.dtd.html)**元素用于配置各种数据类型的转换器如String、Array等。**  **<converter id="map"**  **class="uk.ltd.getahead.dwr.convert.MapConverter"/>** |

## 3、dwr.xml的业务组件发布和调用规则配置部分Allow

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Allow部分是配置业务组件的发布和访问控制规则的一些参数选项，这部分是DWR配置的重点部分。其主要有以下两个子节点组成：  一、[**create**](http://getahead.ltd.uk/dwr-demo/dtddoc/dwr10.dtd.html)**元素 这个元素是配置后台业务组件的创建规则和组件方法的**  **访问规则。**  **二、**[**convert**](http://getahead.ltd.uk/dwr-demo/dtddoc/dwr10.dtd.html)**元素 这个元素是配置Java对象和数据类型对应的转换器。DWR内置了所有java基本数据类型的转换器。基本数据类型的对象形式也采用同一的转换器转换。**  Create配置元素详细说明:  **属性：**  Creator  配置业务组件的创建类型。主要有四种:   |  |  | | --- | --- | | new | 通过它可以访问远程的任何类型的bean, 但是, 与其他创建器不同的是, 在发出远程调用之前不建立bean. | | none | 不创建对象, 适用于如下情形: (1)对象已经存在, (2)调用的是静态方法. | | spring | 通过Spring框架访问bean | | jsf | 使用JSF的对象 | | struts | 使用Struts ActionForm bean | | pageflow | 访问Beehive或者WebLogic提供的PageFlow | | ejb3 | 一个访问EJB3会还bean的实验性创建器. 这些代码没有经过完整测试, 也灭有官方维护, 稳定性有待改善. |   Spring、New、 none 、Scripts。  JavaScript  前台页面javascript引用此组件的名称。名称需要唯一。  scope  组件的作用范围或则叫做生命周期。这个和Servlet的生命周期的作用是 一样的。共Session、request、application、page四个选项。  Create配置元素详细说明:  **子节点：**  Param  配置业务组件的创建参数。Name属性设置名称,value属性设置业务组件的具体路径(类全名称)。  Include  授权方式保护业务组件方法的可见性。属性method可以访问的方法名称。如果没有配置则默认为所有方法都可以访问。  exclude  排除方式保护业务组件方法的可见性。属性method设置不可以访问的方法名称。除此之外的都可以访问。如果有需要控制业务组件方法的访问权限时建议采用这种方式。  Auth  采用EJB的角色访问控制方式。Method属性设置可以访问的方法名称，role属性设置访问角色。  Convert配置元素详细说明:  **属性：**  converter  配置数据类型的转换器。一般是设置bean转换器，因为基本类型的转换器DWR默认设置。  match  设置要使用此转换器的java实体对象。如:  **<convert converter="bean" match="demo.entity.\*"/>**  **子节点:**  Param对象属性的转换规则,name属性设置规则有include和exclude选项，作用和组件方法控制的一样。Value属性设置实体对象的属性。默认是所有属性都进行转换。 |

## 4、dwr.xml的业务组件方法签名配置部分Signatures

|  |
| --- |
| Signatures是用来配置Java Bean数据类型转换时遇到不能确定转换类型的情况。如:  Bean.setList(List a)方法，DWR设置集合a的时候就无法确定a这个集合里需要填充哪种数据类型。所以针对有输入集合对象的情况就需额外设置Signatures配置节点来设置。这个节点的配置如下:  **<signatures>**  **<![CDATA[**  **import java.util.List;**  **import demo.entity.\*;**  **import demo.busi.\*;**    **Customer.setAccounts(List<Account> accounts);**  **DemoBusinessObject.getListSet(List<Contact> inList);**  **]]>**  **</signatures>**  熟悉C++模板编程或java范型编程的开发人员对这些东西并不陌生。这里主要是采用范型机制来设置bean转换时的集合元素类型。注意DWR有套解析机制所以这里并不需要JDK1.5的支持。  Signatures主要有两个部分:  Import部分 和java文件的作用一样  另一部分对方法的签名进行范型的设定。返回结果的集合元素类型不需要设置，因为可以在运行时确定。 |

## 6、JS开发注意事项

|  |
| --- |
| **1、要注意DWR异步机制带来的数据可能不同步问题**  **2、返回结果只能是一个值所以要详细设计返回结果类型**  **3、要注意Java引用传递和JavaScript值传递的差异**  **4、连续多次后台请求尽量采用批量调用**  **5、尽量将JavaScript的逻辑处理代码抽取成一个独立的文件** |