## 产品定位

展现中间件，价值：

* 更好用的前端界面
* 更高的开发效率

适合基于B/S架构的管理类应用，不适合面向互联网的开放式站点。原因：

* 界面元素风格-信息管理类的维护界面与面向互联网的界面元素风格存在较大的差异性。
* 对搜索引擎的友好度-对于这种注重AJAX操作的前端开发技术本身就与搜索引擎存在不适应的情况。

Dorado7最适合采用ORM技术实现数据持久层的开发模式，它在设计之初就是为ORM考虑的，但是考虑到之前Dorado5那种JDBC类型的快速开发方式依然能满足不同类型客户的需求，在Dorado7中依然提供了基于JDBC的开发模式。

## 更好用的前端界面

* 丰富的界面控件（70多个控件）
* 支持各种主流浏览器（IE、Chrome、Firefox、Safari、Opera等）
* 相比Dorado5更加Ajax化，更多的采用异步处理机制，可以更好的支持异步服务端通讯，从而进一步提高界面操作的友好性
* JS、CSS的按需装载和Server端自动合并

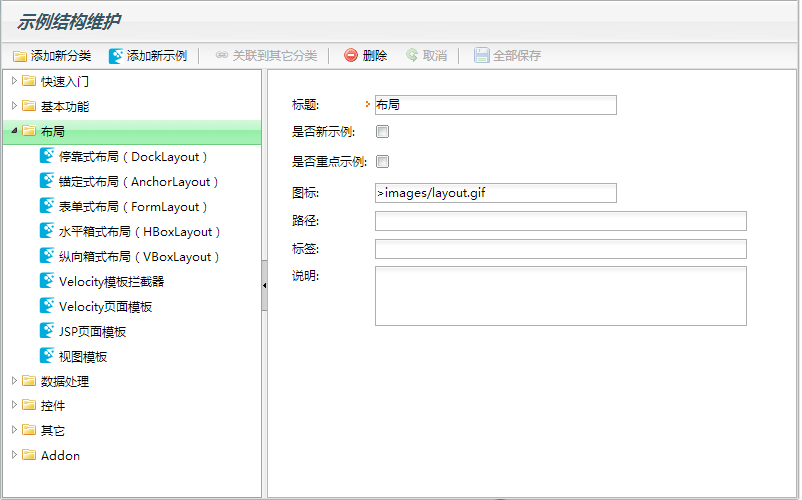
## 更高的开发效率

* 全新的基于Eclipse的IDE
* Dorado产品一贯的特色——以XML配置简化前端界面的定义，并彻底以XML替代传统JSP
* 以数据模型为核心的开发模式。如：数据敏感控件，立体数据模型
* 遵循“约定优于配置”的设计思想，例如自动方式适配

## 以XML替代传统JSP的原因

1）管理类应用界面的特点，采用控件+布局管理器的界面定义模式最为高效。

这种类型界面的特点：信息量大，对控件利用率的要求较高；需很好的适应各种分辨率；界面元素相对程式化。



2）有利于简化界面开发过程和降低后期维护成本。

* 基于HTML的开发模式页面代码冗长且不易维护。
* 基于HTML的开发模式对开发者的要求较高，需时刻考虑HTML代码对各种浏览器的兼容性

## 以数据模型为核心的开发模式

UI控件发展的两个阶段：

* 离散控件集型控件集——Dojo、Ext3
* 数据模型驱动型控件集——Dorado、Ext4
* Ext的发展历程映证了数据模型驱动型控件集是更加高效的

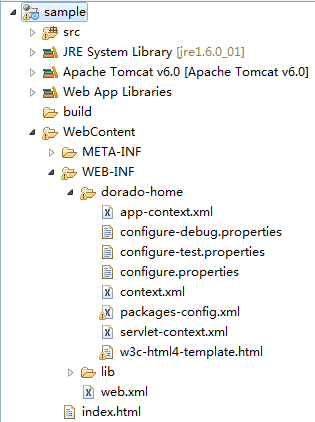
Dorado一贯主张将数据与表现形式分离，实现这种分离，需要几个概念：

* 数据载体——DataSet
* 数据感知控件——DataGrid、TextEditor、...

数据模型驱动型控件集的优点：

* 开发者日常接触的API的数量大大减少
* 数据模型中提供了很多附加功能，例如：状态管理、翻页、校验、提交等等。可以在很大程度上简化开发
* 培养开发人员以更加贴近业务逻辑的角度思考问题。始终把经历放在核心的业务逻辑的实现上，而不必因为各种零散的界面特效而致思维发散
* 页面代码更加集中，所以更加容易阅读和维护。大部分的页面代码会围绕数据模型展开
* 当界面设计改变时，重构变得更加简单。因为数据模型几乎不需要改变，开发者需要的只是改用一些其他的数据感知控件

## 项目基本结构和配置



### 1）web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.4"

xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd">

<listener>

<listener-class>com.bstek.dorado.web.servlet.SpringContextLoaderListener</listener-class>

</listener>

<servlet>

<servlet-name>doradoServlet</servlet-name>

<servlet-class>com.bstek.dorado.web.servlet.DoradoServlet</servlet-class>

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

</servlet>

<servlet-mapping>

<servlet-name>doradoServlet</servlet-name>

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

</servlet-mapping>

<servlet-mapping>

<servlet-name>doradoServlet</servlet-name>

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

</servlet-mapping>

<servlet-mapping>

<servlet-name>doradoServlet</servlet-name>

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

</servlet-mapping>

</web-app>

SpringContextLoaderListener

Dorado7将Spring作为自己比较底层的依赖包，因此在web.xml中可以看到由dorado提供却以Spring命名的listener：SpringContextLoaderListener，它是继承Spring中标准的ContextLoaderListener的一个类，用于装载dorado的一些基础配置。

doradoServlet

doradoServlet用于提供dorado引擎服务的Servlet，也可以用于Spring的MVC服务。

|  |  |
| --- | --- |
| 拦截字符串 | 说明 |
| \*.d | dorado最基本的服务，如视图请求，AJAX调用请求等。 |
| \*.dpkg | dorado的package资源请求，根据dorado资源打包方式，可包含的资源有js，css等。dorado中资源的按需装载和自动合并功能都是通过这个后缀实现的 |
| /dorado/\* | dorado自身的服务：控制台服务，对外界提供的特殊服务；如为IDE提供的服务 |

### 2）doradohome

dorado工程在WEB-INF目录下会包含一个doradohome的文件夹，用于定义dorado的一些系统全局配置和关键的Spring配置。

可以通过以下方式指定dorado-home文件夹的位置：

* 默认位于WEB-INF下
* 通过web.xml的context-param定义，参数名为doradoHome

<context-param>

<param-name>doradoHome</param-name>

<param-value>xxxxx</param-value>

</context-param>

* 通过系统环境变量DORADO\_HOME（如果一台服务器有多个dorado项目，则这种方式就不适用了）

doradohome文件夹中包含以下文件：

|  |  |
| --- | --- |
| 文件名 | 说明 |
| context.xml | dorado核心服务配置文件 |
| config.properties | dorado属性配置文件 |
| packages-config.xml | JS文件和相关客户端资源文件的打包规则设定文件 |
| servlet-context.xml | 用于存放SpringMVC相关配置文件 |
| w3c-html4-template.html | dorado视图的默认模板 |

### 3）运行模式

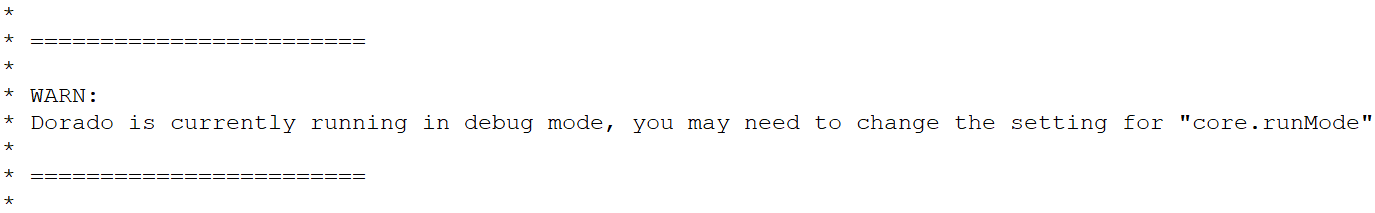
* 设计说明：

在项目开发中经常会为了配合系统的调试做很多临时性的设定，当我们向正式环境中部署时，系统部署人员往往不是很了解这些设定，没有将这些属性设定调整为生产环境所需，从而导致很多问题，包括安全性方面的。根据这种情况dorado提供了运行模式处理机制，它允许你根据不同的运行环境定义名称不同的配置文件，而在最终发布时，只要调整一个系统全局变量就可以进行切换。这样就可以从一种运行环境切换到另一种环境时，系统发布人员只要修改一个变量的值。很显然这很受系统部署人员的欢迎。

dorado7通过config.properties文件中runMode的值做运行模式的切换：



系统运行时，会自动检测运行模式，并会给出运行模式的警告：



dorado中默认提供了三种运行机制：production（生产环境），debug（开发和调试），test（测试）。默认就是production运行模式

* 配置装载规则

configure.properties是首先被装载的属性文件，如果发现core.runMode做了设定，则系统会自动查找configure-xxx.properties文件，并加载。

产品正式发布的时候建议将core.Mode的值清空，默认情况下会按production模式运行。

* dorado默认提供的常用配置属性

|  |  |  |
| --- | --- | --- |
| 属性名 | 默认值 | 说明 |
| core.runMode | production | 生产环境 |
| view.debugEnabled | false | 是否允许启用Dorado提供的Debugger工具，如果启用，则在浏览器中的视图中按F2快捷键打开Debugger工具 |
| view.useMinifiedJavaScript | true | 是否使用压缩后的JavaScript，一般产品调试时采用非压缩状态，有利于调试，而正是发布后采用压缩状态，有利于降低网络的数据流量，提供性能 |
| view.useMinifiedStyleSheet | true | 是否使用压缩后的Style，原理同上 |
| view.outputPrettyJson | false | 是否格式化输出JSON数据，原理同上 |
| view.skin | default | 默认皮肤 |
| view.mergeJavaScript | true | 是否合并JavaScript，正是发布后建议采用合并，原理同上 |
| view.mergeStyleSheet | true | 是否合并CSS文件，正式发布后建议采用合并，原理同行 |
| view.useRandomStringAlias | true | 为LoadRunner这类压力测试工具专门设计的参数  是否启用随机别名机制，例如作为服务的view的名字 |
| core.classPathResourceReloadable | false | 主要针对资源类文件的加载，如js,css等，一般用在deubg模式，避免重新发布和重启 |
| data.config.autoReloadEnabled | false | 针对doradohome文件的重新加载，也适用于debug模式 |
| data.config.autoRecalculatePaths | false | autoRecalculatePaths与autoReloadEnabled类似，处理pathPrefix |
| view.showExceptionStackTrace | false | 是否显示ExceptionStackTrace |

### 4）Spring上下文

configure.properties只能用于配置dorado7的一些最基本的参数，更多的dorado配置信息还是位于像context.xml这种类型的Spring文件中，由于dorado7将Spring作为底层最基础的依赖，dorado7自身的服务也是通过Spring搭建起来的，当我们需要扩展dorado7的时候，大多数情况下我们都需要修改这些Spring配置文件。

* .properties与context.xml的关系

举例说明：如context.xml中有如下代码：

<bean parent="dorado.dataConfigLoader">

<property name="configLocation" value="${model.root}/\*.model.xml" />

</bean>

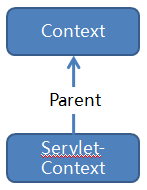
注意这个bean-dorado.dataConfigLoader的属性设定中

”${model.root}/.model.xml”，其中${model.root}是Spring本身就支持的EL表达式的语法。通常情况下它都是用于从.properties配置文件中读取一些参数并在运行时进行替换。

.properties与Spring配置文件之间的关系：.properties作为Spring配置文件中EL表达式的属性来源，另外.properties文件也作为dorado自身属性配置的存放处。对于专门为Spring配置文件提供服务的属性，例如刚才的“model.root”，它的变量名我们也可以修改的，只要在修改时保证.properties文件的属性名与Spring配置文件的EL表达式保持一致就可以。

* 运行时Spring的上下文

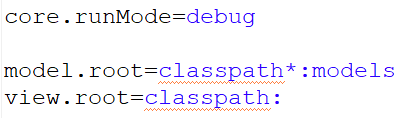
根据Spring的运行机制，一旦配置了一个Spring的MVC，系统产生的上下文对象除了默认的Context之外还会产生一个Servlet-context，其中Servlet-Context会包含上面的Context对象，并作为parent关系存在。



如果定义了多个Spring的MVC，就会有多个Servlet-Context，但是Context只有一个。根据运行机制知道Servlet-Context可以访问Context中的bean，但是反过来却不行，这也是之前我们要求的在doradohome文件夹中，如果用户定义Spring的MVC最好将它定义为servlet-context.xml中的主要原因。

### 5）资源定位表达式

.properties中



model.root的值，有以classpath\*作为前缀的，也有以file作为前缀的，因此下面要简单介绍一下dorado中的资源定位表达式（资源定位表达式通常用来描述一个或一组文件的位置，它的基本格式为：[协议:]路径）。

目前支持的协议有：

* file：表示操作系统的文件系统路径；
* home：表示doradoHome所在位置的起始路径；
* classpath：Java资源路径
* classpath\*：Java资源路径，是特别为了从多个jar包中根据通配符批量的提取资源而提供的；

（协议是可以省略的，当不指定具体的协议时Dorado7会自行判断应如何理解用户给出的路径。

如果路径以“/”开头，Dorado会认为该路径描述是项目web根下的文件。

如果不以“/”开头，Dorado按照classpath协议来处理路径。）



### 6）context.xml文件基本配置简介

context.xml内容：



darado.dataConfigLoader

用于设定dorado数据模型文件的存放位置，默认该位置为src下的models这个package下，当我们将数据模型文件放在models这个package下，dorado7就会自动装载这些模型文件。

dorado.viewConfigFactoryRegister

用于设定dorado视图文件与请求URL的匹配规则的配置文件。

dorado.packagesConfigLoader

用于设定dorado中js和css资源打包规则的配置文件的存放位置。