2.安装与配置

URule2技术交流QQ群及在线体验示例
在线体验示例：http://112.124.15.63:8084/为了不同用户操作互不影响，需要注册账号登录体验操作。

URule2技术交流QQ群：423339793

URule2规则引擎产品教学视频：https://pan.baidu.com/s/1oAeJeCM，密码：evsk

安装到Maven项目

这里我们首先以Maven项目为例来介绍如何将URule2添加到项目当中。前面我们提到，URule2采用的是典型的客户端服务器运行模式，实际上，对于URule来说，它的客户端与服务器可以位于一个项目当中，那么这个项目既是客户端也是服务器，这种模式也叫“嵌入式模式”，这对于小型项目来说比较有意义，如果客户端只有一个，那么就不需要将客户器与服务器分拆到两个项目当中。首先我们将以嵌入式模式为例来说明如何安装URule2到一个现成的Maven项目当中（后面会有专门的章节说明客户端服务器隔离模式的安装与配置）。

打开pom.xml文件，在其中添加URule2的urule-console包的依赖配置，开源版依赖配置如下所示：

开源版urule-console的依赖配置

```
<dependency>
 <groupId>com.bstek.urule</groupId>
 <artifactId>urule-console</artifactId>
 <version>[version]</version>
</dependency>
```

商用PRO版依赖配置如下：

商用PRO版urule-console的依赖配置

```
<dependency>
 <groupId>com.bstek.urule</groupId>
 <artifactId>urule-console-pro</artifactId>
 <version>[version]</version>
</dependency>
```

关于URule PRO版与开源版名称问题
URule当中有两个包，开源版为urule-core和urule-console，而商用PRO版则为urule-core-pro与urule-console-pro，后面内容中统称为urule-core和urule-console；如未特别注明，安装配置方式皆相同。

说明
在URule2当中，所需要的包有两个，分别是引擎计算核心包urule-core和控制台管理用的urule-console，这里的urule-console实际上就是原URule1中的urule-repository包，只是到URule2中后更名为urule-console。对于我们的客户端来说，只需要urule-core就可以了，但对于服务器来说，urule-core与urule-console两个包都是不可少的，同时又因为urule-console依赖urule-core，所以看到我们上面的Maven依赖中只依赖了urule-console，因为依赖的传递性，会自动依赖并加载urule-core。

关于URule的具体版本号，我们可以到http://search.maven.org/上输入“urule”关键字，以查询URule相关包的具体release版的版本号，对于SNAPSHOT版我们可以到https://oss.sonatype.org/上查询。完整可以到URule位于Github上的地址https://github.com/youseries/urule上下载。

如果我们要采用https://oss.sonatype.org/中最新的SNAPSHOT版本，那么就需要在pom.xml中添加一个repository信息，告诉Maven该到这里去下载snapshot版本的包，repository信息如下所示：

库配置

```
<repository>
    <id>sonatype</id>
    <url>https://oss.sonatype.org/content/groups/public/</url>
</repository>
```

到这里，pom的配置就完成了，我们再来看看一个标准的非Maven的Web项目如何将URule添加进去。

安装到标准项目

我们知道Maven项目需要配置依赖来加载Jar包，非Maven项目则直接将Jar包复制到/WEB-INF/lib目录下即可。据此，我们可以首先点击此处下载urule-console模块与urule-core模块所需要的第三方Jar包，将它们放到我们项目中的/WEB-INF/lib目录下（这些Jar包有一些可能项目中已经存在，这时通常的做法是保留高版 本的即可，切不可放多个版本不同的相同Jar包，否则运行可能会出现错误），然后再到http://search.maven.org/上查询最新的urule-core与urule-console版本，下载下来放到/WEB-INF/lib目录中即可。这样，一个传统的的Web项目中添加URule相关Jar包的工作也就完成了，接下来我们开始进行web应用层面配置，对于Web层面的配置，无论是Maven项目还是标准项目都是一样的。

项目配置

因为urule-console模块架构在Spring之上的，所以需要加载urule-console模块中提供的Spring配置文件，这个配置文件实际上位置urule-console对应的jar的classpath根下，名为urule-console-context.xml，所以如果我们的项目也是基于Spring的，那么可以打开一个项目中的Spring配置文件，在其中通过下面的代码导入urule-console-context.xml文件：

导入urule-console的spring配置文件

```
<import resource="classpath:urule-console-context.xml"/>
```

如果你的项目不是基于spring，那么就不能采用上面的import方式加载urule-console中所需要的spring配置文件，这时我们需要打开web.xml，在其中添加下面的代码以加载urule-console的spring配置文件：

web.xml中加载urule-console的spring配置文件

```
<listener>
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<context-param>
 <param-name>contextConfigLocation</param-name>
 <param-value>classpath:urule-console-context.xml</param-value>
</context-param>
```

备注
实际上，我们当前项目中加载urule-console模块的同时也加载了urule-core模块，这个我们前面解释过，因为urule-console模块依赖于urule-core模块。所以也需要加载urule-core模块中的spring配置文件，这个文件也位于urule-core对应jar包的classpath根下，名为urule-core-context.xml，但我们这里在配置时却不需加载它，原因是这个名为urule-core-context.xml的spring配置文件，在urule-console-context.xml中已经导入了，所以在有urule-console的项目当中，就不需要再加载urule-core-context.xml，只需要加载urule-console中的urule-console-context.xml文件即可。

通常情况下，我们建议在/WEB-INF目录下创建一个名为context.xml的标准的spring配置文件，在这个context.xml中导入urule-console模块中的spring配置文件，这个context.xml的内容如下所示：

context.xml文件内容

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd
 ">
 <import resource="classpath:urule-console-context.xml"/>
</beans>
```

这样，就需要将上面在web.xml中添加到listener做些修改，contextConfigLocation的值改成/WEB-INF/context.xml，如下面代码所示：

修改后的listener配置

```
<listener>
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<context-param>
 <param-name>contextConfigLocation</param-name>
 <param-value>/WEB-INF/context.xml</param-value>
</context-param>
```

关于Properties文件加载
如果你是将URule配置到一个已存在的Spring项目中，同时项目也要Spring中加载自己的Properties文件，比如通过下面的方式加载自己的Properties文件

<context:property-placeholder location="/WEB-INF/app.properties"/>

这时启动应用，我们会发现启动过程中系统会报各种属性找不到的异常，解决办法就是在上面配置中加上 ignore-unresolvable="true" 以及 order="1" 两个属性即可，其中ignore-unresolvable属性表达忽略当前配置的Properties文件中没有配置的属性，order属性值表示加载优先级，值越小，优先级越高。

当然如果你的项目中Properties文件加载是通过配置PropertyPlaceholderConfigurer为bean的方式加载，那么同样需要为这个bean添加如下两个属性，作用也是一样：

```
<property name="ignoreUnresolvablePlaceholders" value="true"></property>
 <property name="order" value="1"></property>
```

URule中有一些默认的允许外部覆盖的属性，比如用于指定当前知识库存放目录的urule.repository.dir属性、用于指定URule Console控制台首页显示页面的urule.welcomePage属性等，对于这些属性我们可以在项目的WEB-INF目录下新建一个名为configure的properties文件，在添加设置这些属性值，然后在我们的context.xml文件中通过如下方法加载即可(如果你的项目中已有Properties文件，那么直接在这个Properties文件里配置这些属性即可)：

属性文件配置

```
<bean parent="urule.props">
 <property name="location">
  <value>/WEB-INF/configure.properties</value>
 </property>
</bean>
```

可以看到，这里配置了一个标准的spring的bean，只是这个bean是URule中提供的名为urule.props的bean的子类，这个ID为urule.props的Bean是URule中提供的一个用于加载spring外部属性文件的Bean，通过上述方式就可以将外部属性文件加载并覆盖URule中默认的属性值。当然如果我们的应用中也有属性需要加载，也可以放在这个文件中一并加载，因为这里通过urule.props加载的spring的属性文件就是标准的spring属性文件加载方式。

> 属性文件加载方式说明
> 实际上，这里的ID为urule.props的bean对应的类是org.springframework.beans.factory.config.PropertyPlaceholderConfigurer的子类，我们知道这个类就是标准spring中加载属性文件的类，所以我们也可以直接使用org.springframework.beans.factory.config.PropertyPlaceholderConfigurer来加载我们的属性文件，只不过将需要设置下它的order属性值，比如设置为100，以保证我们的属性文件会被优先加载，从而覆盖默认的URule属性。

最后我们还需要在项目的web.xml当中添加URule Console中的一个Servlet，这个Servlet负责控制台中所有页面与服务端的交互，配置信息如下：

URule Console的Servlet配置

```
<servlet>
 <servlet-name>uruleServlet</servlet-name>
 <servlet-class>com.bstek.urule.console.servlet.URuleServlet</servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>uruleServlet</servlet-name>
 <url-pattern>/urule/*</url-pattern>
</servlet-mapping>
```

在上面的servlet配置当中，需要注意的是servlet-mapping中的url-pattern的值必须是/urule/*。

到这里，我们的项目中添加URule的操作就完成了，接下来，需要打开我们放在/WEB-INF目录下的configure.properties文件，在其中添加一个名为urule.repository.dir属性，用于指定URule的资源库存放的目录，比如设置urule.repository.dir=D:/repo，表示采用D盘repo目录来作为URule的资源库目录，这时我们需要在D盘中创建一个名为repo的空目录，否则启动时会产生找不到目录的错误。属性配置好后，就可以运行项目，浏览地址：http://localhost:[port]/[contextPath]/urule/frame，就可以看到如下图所示的URule Console控制台：

看到上面的界面，就说明我们配置成功了，接下来就可以在控制台中添加项目，配置规则.....

参数配置

在前面我们添加的configure.properties中，我们已经添加了一个名为urule.repository.dir属性，用于指定URule资源库目录，除此之外，URule2还提供了下面这些属性可供配置：

属性名	值类型	描述
urule.repository.dir	String	配置当前资源库存放目录，该属性值与下面的urule.repository.xml属性值至少要有一个不为空，否则系统启动会报错。 在URule2当中，引擎支持两种存储资源方式，一种就是在这里通过配置urule.repository.dir属性指定目录，将资源存储到具体的目录当中；另一种就是通过配置urule.repository.xml属性来指定一个可存储到数据库的配置文件，这样资源就可以存储到数据库当中。 默认urule.repository.dir属性为空，所以如果我们不想用数据库来存储URule资源，那么我们就需要配置好该属性。 在配置这个目录时，我们可以给出一个绝对路径作为其资源库存储目录，也可以是一个相对于当前WEB目录的相对路径；如“D:/repo”就表示将采用D盘下repo目录为资源库目录，需要注意的是，这里repo目录必须要存在，如果不存在那么将会产生错误。 如果要采用相对于当前WEB应用的相对路径，那么可以设置成“/repo”，那就表示在当前WEB应用根下使用repo目录为资源库存储目录，如果repo目录不存在，系统将会自动创建。

urule.repository.xml	String	通过该属性在外部指定一个将资源库存储到数据库的配置文件，这样系统启动时就会按照这个文件中定义的数据库信息自动创建存储库所需要的各种表，从而实现将URule规则库存储到数据库的目的，详细描述见12.数据库中存储知识库。 如果在配置了urule.repository.xml属性后，又配置了urule.repository.dir属性指定了目录，那么这个目录就用于存储与数据库相关的缓存信息。如果没有配置urule.repository.dir属性，那么缓存信息默认将在Jvm的临时目录中存储（通过System.getProperty("java.io.tmpdir")获取到的目录）
urule.resporityServerUrl	String	客户端上配置的服务器地址，用于获取在服务器上的知识包信息，详细描述见14.客户端服务器配置。
urule.knowledgeUpdateCycle	int	一个数字，用来指定客户端多久到服务端检查当前知识包有没有更新。如果为0则每次都检查，为1则永不检查，为1以上的值，则表示每隔多少毫秒检查一次，比如10000，就表示每隔10000毫秒检查一次。默认值是0，表示每次都检查，详细描述见14.客户端服务器配置。
urule.welcomePage	String	一个URL，用于替换URule Console主界面第一次看到的工作区内容，默认是个URule说明（比如上图中显示的“URule Console Quick Start”），我们可以配置个URL，这样默认就会显示这个URL对应的内容。
urule.console.title	String	一个字符串，用来替代URule控制台页面的title，默认值是“URule Console”。
urule.authority.type	String	一个URule Pro版专用属性，用于定义PRO版本中权限配置里权限配置主体的类型，默认为“用户”，如果我们需要修改这个属性，那么需要将中文先转换为uniocde。