部署工具使用手册-2018.12

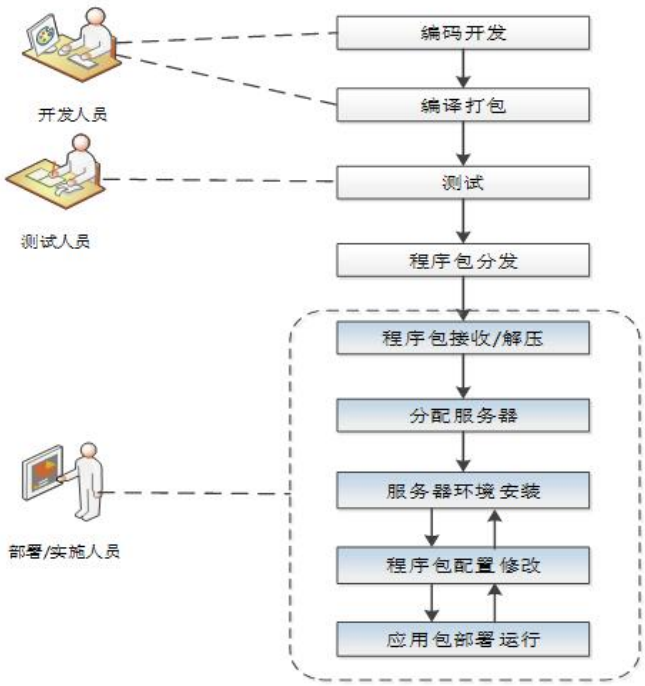
java tool

---

1. 使用场景

1.1 传统部署方式痛点

软件应用（项目型）一般流程是由开发人员进行编码开发，调试，提交测试，由测试人员测试，然后应用包发布，最后由项目实施人员进行项目应用部署。具体工作及流程见下图：



如上图可见，从程序包分发出去开始，即由项目实施人员对程序包进行安装、部署。其中部署环境包括apache/nginx/tomcat/jdk/mysql/https/mq/solr等等。产品的正常运行，还依赖各种配置文件的正确设置（如xml文件/properties文件/其它文件/db/等，需配置相应的ip/端口/名称/地址等），一旦有地方配置有误，运行报错，实施人员则需要反复检查相应的配置文件/环境配置是否正确。若无法检查出问题，则还会需要找相应的开发人员寻找原因。这个过程对于实施人员是反复且痛苦的。

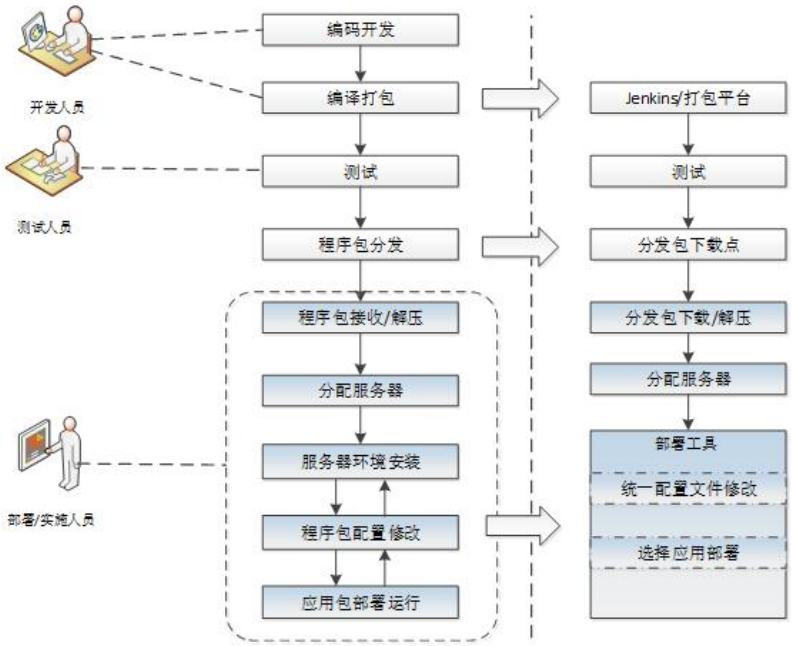
以上流程，若是通过手工执行，痛点有以下几点：

- 产品分发管理不规范：给了多少项目组使用，各项目使用的是什么版本
- 部署包传送麻烦：QQ/FTP
- 部署过程复杂
- 配置文件繁琐
- 配置容易出错
- 出错难定位问题
- 实施人员-开发人员耦合度高

1.2 自动部署方式

> 对于产品型（无需项目组参与）的应用，可使用jenkins，结合git/svn/maven编译打包，通过shell/bat脚本进行部署即可，此处不详细讨论。

对于项目型应用，基于上述的痛点，本部署工具就是用于解决项目实施人员在部署时的遇到的问题，让实施人员部署应用时更简单，快捷，少配置，少错，易查。使用本工具，结合jenkins（或自行构建打包平台）编译构建的分发包，可较好地解决打包/配置/部署的问题。流程如下所示：



简单说明一下

> - 通过jenkins(或打包平台)可对程序进行自动构建、自动生成相应的程序包。（若在打包平台，同时把相应的程序需要修改的配置在打包时一起修改）。
> - 自动构建的程序包可统一按版本放置以提供下载地方（如FTP/共享文件夹/云盘均可）。（若在打包平台，直接在打包平台下载即可）。
> - 获取程序包后，实施人员解压放置在部署工具约定的目录、并按需求分配服务器，修改统一配置的文件（若在打包平台配置好，此步骤可跳过），通过简单的选择项进行自动部署即可。

2. 功能概览

2.1 部署工具面向的人员

使用部署工具，主要是固化原来手工部署的流程，并提供简化、统一的配置项，分产品、环境、模块进行自动部署，面向的对象主要包括：

- 产品发布人员：管理产品版本，管理产品分发，根据情况修改部署工具环境、流程、配置项。
- 产品部署人员：内部产品部署、测试。

- 项目实施人员：项目实施部署。

2.2 部署工具功能

部署工具运行时（交互式界面运行），以linux下安装mysql和redis为例，如下：

项目实施人员只需根据实际情况选择需要安装的模块即可。

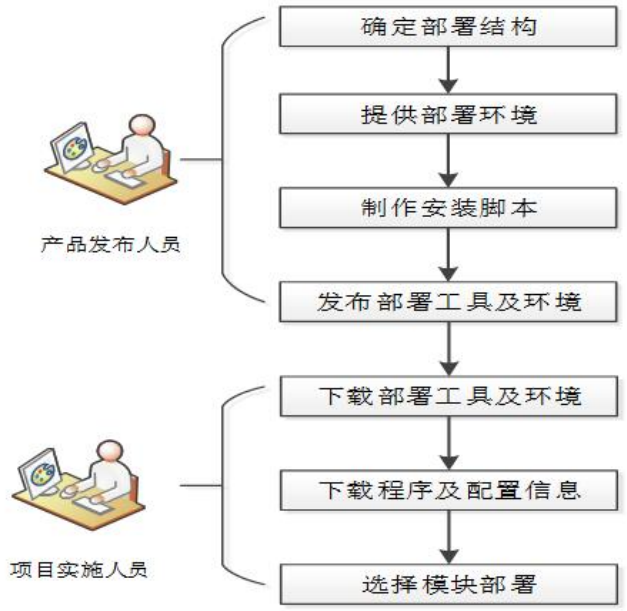
部署工具主要有以下功能点：

- 固化部署流程：各个产品部署流程都不一样，部署什么环境、产品包含什么模块，各模块安装顺序如何，均在在部署工具中进行设置、固化，以便部署实施人员使用。
- 提供简化、统一配置项：在一套产品中，需要配置的项可以集中在统一、简化的配置文件，产品中各模块需要修改的配置均可在配置此文件中进行读取，替换即可。
- 分产品、环境、模块进行操作：部署需要安装的环境、产品模块均可自定义，并在部署的命令行界面中显示及运行。
- 支持windows及linux下运行部署。
- 提供shell/bat执行功能：可自定义部署脚本，并在部署工具中运行。
- 提供数据库脚本执行功能：包括数据CURD操作。
- 提供配置文件替换功能：可按模板替换/按xml局部查找替换/properties文件key匹配替换。
- 提供生成二维码功能：可根据参数生成二维码。
- 提供动态配置项功能：某些配置需要根据已的配置计算或变换得出。
- 查看统一配置文件内容。

3. 部署工具运行流程

3.1 部署工具从制作到使用

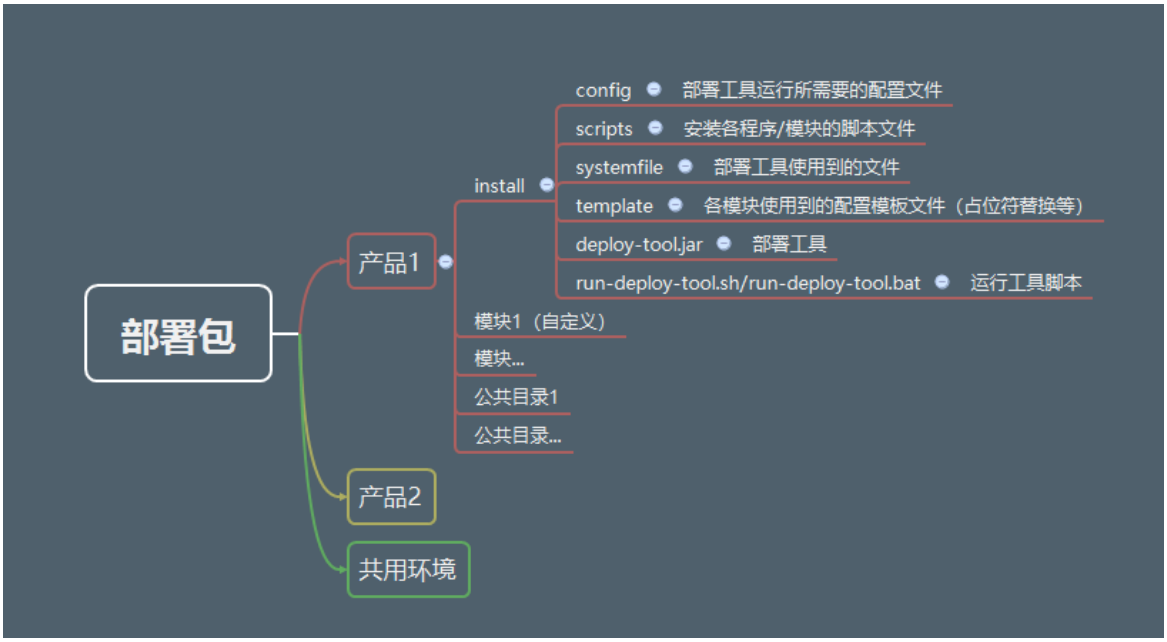
先由产品发布人员对部署工具及环境进行制作，然后发布，由项目实施人员进行部署。具体如下图所示：



- 确定部署结构：包括安装此产品应包含哪些模块及安装环境，确定它们存放的位置。
- 提供部署环境：为安装此产品所需要的环境包，如jdk，tomcat等。
- 制作安装脚本：使用shell(linux)或bat(windows)进行安装部署脚本编写。全部模块及环境的安装均依赖此安装脚本，因此，产品发布人员主要工作是保证这些安装脚本的正常运行。
- 发布部署工具及环境：产品发布人员先自行测试，完成后发布此部署工具及环境，以提供项目实施人员使用。
- 下载部署工具及环境：项目实施人员在项目实施时，先规划好产品安装的服务器划分（如web服务器，后端服务器等）。
- 下载程序及配置信息：下载程序包，根据实际服务器信息填写配置信息。
- 选择模块部署：根据实际情况选择需要安装的模块进行部署。

3.2 部署工具目录结构

部署工具运行有其相对固定的目录结构（可自定义但不建议），建议使用时目录结构按默认结构即可。结构如下：



说明：

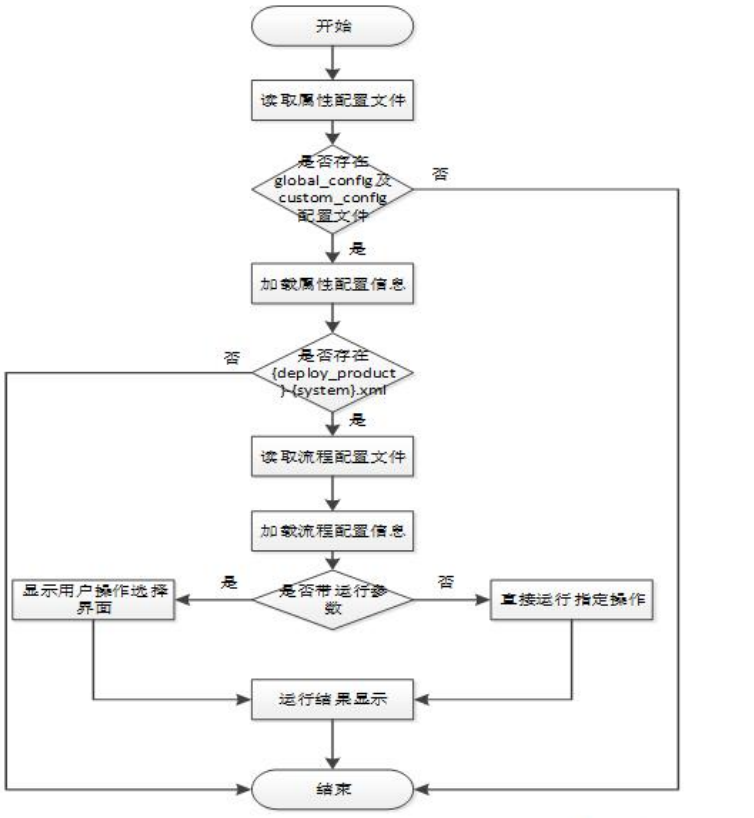
> - **部署包**是完整一套产品部署的环境+程序，它可能会包含多个产品，多个产品会共用某些环境（如数据库）。如图上所示，可以是产品1+产品2+DB。当然，若只有一个产品，则此产品作为独立部署包即可。
> - 产品可能包含多个**模块**，建议这些模块可以按服务器划分，然后再根据依赖软件或环境（software）及程序(program)划分。如前后端分离部署实例，分为前台服务器frontend及后台服务器backend，然后分别设置所需软件及程序目录。当然用户也可以自定义存放结构，只要部署工具及相应的部署脚本可找到这些目录即可。
> - `install`目录，此目录就是部署工具的完整目录，此目录名称用户不可修改，按已有的结构进行文件存放即可。
> - `config`目录，目录名称为"config"，不能修改。此目录存放部署工具运行所需要的流程配置文件、统一配置文件、共用配置文件等，部署工具的启动会先读取此目录的配置文件，再根据配置文件定义的结构运行。
> - `scripts`目录，存放各程序的安装脚本，建议按程序模块划分目录。
> - `systemfile`目录，存放部署工具使用的第三方工具。
> - `template`目录，存放各程序或环境的配置文件模板，动态配置部分使用占位符（$${}）替换。
> - `deploy-tool.jar`，部署工具程序包，需运行它来启动，它依赖JRE8运行环境。
> - `run-deploy-tool.sh`/`run-deploy-tool.bat`，运行部署工具脚本，注意:**运行部署工具依赖JRE8运行环境，因此建议在`共用环境`中存放绿色版本jre8或jdk8目录，并在此脚本中指定路径。**

3.3 运行流程

部署工具的运行方式有两种，一种是以交互式界面运行，一种是使用命令自动运行。在执行`deploy-tool.jar`时，通过是否添加参数决定使用哪一种运行方式。前者不加参数，后者添加执行操作的ID。如：

- 直接执行`run-deploy-tool.sh`或`run-deploy-tool.bat`，出现用户选择操作界面。它的执行命令是`java -jar deploy-tool.jar`。
- 通过添加参数直接执行某操作，如`java -jar /opt/bingodrive/install/deploy-tool.jar installRedis`表示执行安装Redis服务。

部署工具启动需要加载相应的配置文件，启动流程如下所示：



有以下几点需要注意

> - 读取属性配置文件：部署工具会读取`install/config`下全部文件，其中必须包含两个文件：`global_config.properties`及`custom_config.properties`。详细介绍见章节`配置文件概述`
> - 加载属性信息：部署工具会把读取到的全部properties文件的内容加载到内存，因此，各文件中不能有重复的key，若重复，则会覆盖。
> - 读取流程配置文件：流程配置文件为`{deploy_product}-{system}.xml`，如在linux系统部署db，则xml文件名为`db-linux.xml`。其中`deploy_product`在`global_config.properties`中定义。

3.4 配置文件概述

配置文件存放目录为`install/config`，其中需要运行前加载的属性文件在此目录下，流程配置文件xml及xsd则存放在`install/config/deploy-config`目录。在`install/config`下的properties文件会全部加载，因此，此目录下只能存放properties文件。其中`global_config.properties`及`custom_config.properties`必须存在。

3.4.1 全局属性配置文件global_config

全局属性配置文件`global_config.properties`主要设置部署前本产品需要指定的内容，当前只设置产品的类型（`deploy_product`），此类型主要用于部署工具寻找相应的流程配置文件（xml），寻找规则为：在`install/config/deploy-config`目录下寻找`{deploy_product}-{system}.xml`。

如在linux系统部署db，则在`global_config.properties`中设置`deploy_product=db`，则部署工具寻找流程配置文件xml文件路径为`install/config/deploy-config/db-linux.xml`。

产品类型可自定义，只要deploy_product与相应的xml名称对应上即可。

3.4.2 用户属性配置文件custom_config

用户属性配置文件`custom_config.properties`是产品配置的主要内容，此文件应由产品发布人员经过整理，合并，简化所有产品模块的配置项而形成的。后面所有需要替换的配置内容，都从此文件中读取并替换。

如假设产品包含模块1和模块2，它们都需要配置数据库连接信息，而且连接信息是一样的，则只需要在此文件中配置一次，然后在部署工具的流程配置文件中对模块1和模块2的配置文件进行替换即可。如下示例：

```
#数据库IP
server_mysql_ip = 127.0.0.1
#数据库端口
server_mysql_port = 3306
#数据库名
server_mysql_dbname = mydb
#数据库用户名
server_mysql_db_username = root
#数据库密码
server_mysql_db_password = 111111
```

3.4.3 其它属性配置文件

若需要对产品添加其它特殊配置，可在`install/config`目录下新的properties文件，部署工具也会把它一并读到内存，然后在配置替换的时候进行使用。

3.4.4 流程配置文件

流程配置文件是部署工具的核心，存放在`install/config/deploy-config`目录，它包含xml及xsd，前者是部署工具的流程配置，xsd是XML Schema文件，用于描述及规范xml的文件结构，**一般不修改此文件**。

产品发布人员需要编写及设计的是xml文件。它是部署工具运行的依赖，产品发布人员通过此文件定义部署的结构，模块，操作，显示。文件名为`{deploy_product}-{system}.xml`。其中deploy_product在`global_config.properties`，system由部署工具根据实际运行的环境自动识别。

流程配置文件的详细说明见章节`部署工具使用详解`

3.4.5 占位符

部署工具从属性配置文件加载配置内容后，在配置文件替换操作、流程配置文件中均可使用占位符。占位符格式为`$${key}`，其中`key`是属性配置文件中的key或在流程配置文件中property元素中加载的key。如属性`server_mysql_ip = 127.0.0.1`，在使用占位符`$${server_mysql_ip}`的地方会补替换为`127.0.0.1`。

> - 部署工具启动后，会自动添加一个`key`为`deployment_home`，值为`deploy-tool.jar`所在目录(`install/config`)的上一级目录。如在C盘部署test，test目录下是install目录，则`$${deployment_home}`值为`c:\test`。用户可使用此key定位部署目录。
> - 占位符若在流程配置文件（xml）中使用，在元素的值中建议使用`<![CDATA[]]`进行转义。
>
> 如`<![CDATA[jdbc:mysql://$${server_mysql_ip}:$${server_mysql_port}/$${server_mysql_dbname}?autoReconnect=`

4. 部署工具使用详解

要使用部署工具，需要两个重要步骤：编写流程配置文件，编写部署脚本。以下对配置文件的编写及部署脚本需要注意的事项进行说明。

4.1 流程配置文件简单示例

流程配置文件`{deploy_product}-{system}.xml`允许用户对需要部署的模块进行设计、划分，对安装流程，脚本执行，功能设置进行设定。用户需按照配置文件约定的规则进行编写即可。先看以下示例，此示例功能是对tomcat进行安装与卸载。

```
<?xml version="1.0" encoding="UTF-8"?>
<deployment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="deploy-config-schema.xsd">
    <executions>
        <group name="web服务安装与卸载">
            <execution name="安装Tomcat" id="installWebTomcat" display="true" class-
name="deploy.OperRunCommand">
                <configuration>
                    <commands>
                        <command>
                            <exec><![CDATA[scripts/windows/disk/webapp/install_tomcat.bat]]></
exec>
                            <args>
                                <arg><![CDATA[${web_extranet_port}]]></arg>
                            </args>
                        </command>
                    </commands>
                </configuration>
            </execution>
            <execution name="卸载Tomcat" id="uninstallWebTomcat" display="true" class-name="de
ploy.OperRunCommand">
                <configuration>
                    <commands>
                        <command>
                            <exec><![CDATA[scripts/windows/disk/webapp/uninstall_tomcat.bat]]>
</exec>
                        </command>
                    </commands>
                </configuration>
            </execution>
        </group>
    </executions>
</deployment>
```

由上述配置文件内容片断可以看到xml配置文件的结构与元素。此文件通过设置操作名称/ID等信息，指定安装tomcat需要运行的脚本(`install_tomcat.bat`,`uninstall_tomcat.bat`)。这样，在运行部署工具时，即可显示组名称为*web服务安装与卸载*，其下有两个操作，分别是*安装tomcat*及*卸载tomcat*,如下所示：

下面详细说明xml配置文件的结构规则。

4.2 流程配置文件结构

流程配置文件用于配置部署流程，使用xml文件对部署需要做的事情进行描写，各元素功能及使用说明如下：

4.2.1 首行及根元素

第一行设置它的版本及编码

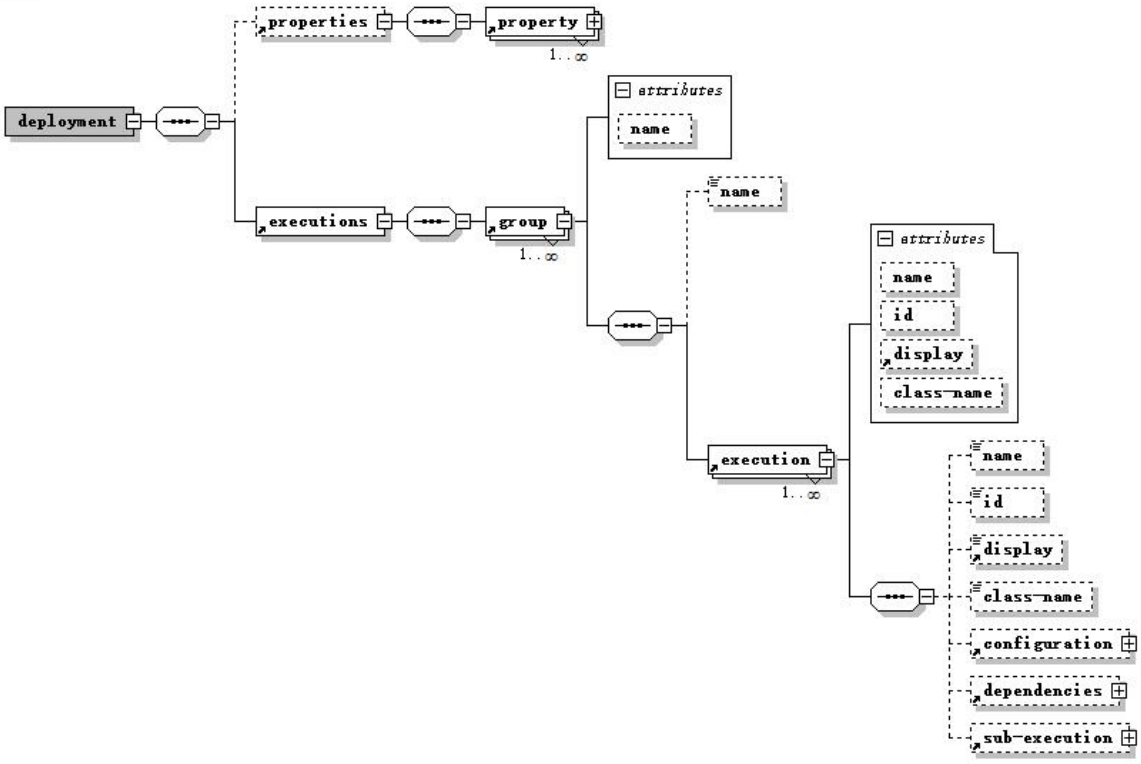
```
<?xml version="1.0" encoding="UTF-8"?>
```

它需以deployment为根元素，并使用xml schema文件，以此规范xml的格式正确。

```
<deployment xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="deploy-config-schema.xsd">
</deployment>
```

4.2.2 xml文件结构

配置文件结构如下图所示：



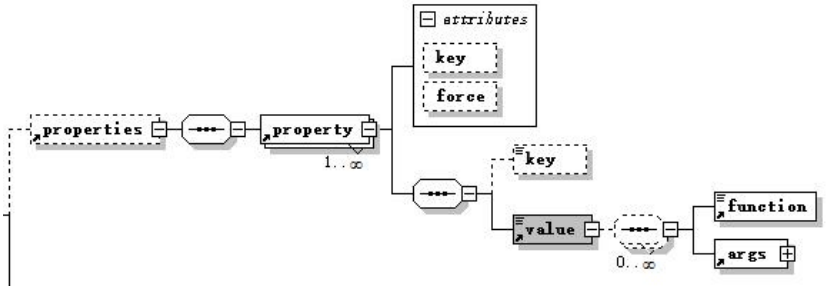
上图中，虚线表示可选，实线为必需，attributes为属性,方框为元素

- 由上图可见，xml主要包括两大元素，分别是`properties`及`executions`元素。如全部配置项已在custom_config.properties文件中设置，`properties`元素可不设置。`executions`是执行器，所有执行流程在此元素下设置。
- `properties`，可选，下有1或多个`property`。
- `executions`下有1个或多个`goup`元素，`group`元素下可有1个或多个`execution`。每一个`group`表示一组相关操作，每一个execution表示一个执行操作。

- `execution`下可以有`name/id/display/class-name/configuration/dependencies/sub-execution`等元素对执行流程进行设置。

下面详细说明各元素的设置及作用。

4.2.3 properties/property元素



property元素主要用于动态添加需要计算或变换得出的属性值，可使用占位符组合成新的property属性，也可通过`function`对参数值进行处理。以key-value值保存到properties文件对应的属性内存中，以便后续使用。

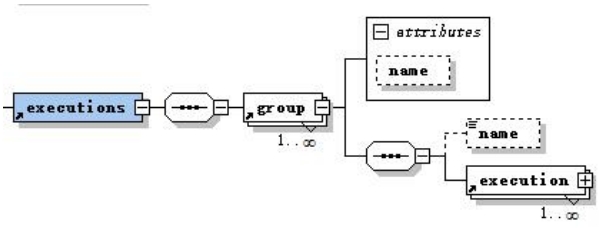
如下是添加tomcat_webapps_home属性(使用占位符`$${deployment_home}`与其它目录路径组合得出）：

```
<properties>
    <property key="tomcat_webapps_home">
        <value><![CDATA[$${deployment_home}/bingodrive_web/program/webapps]]>
        </value>
    </property>
</properties>
```

说明：

- `properties`元素是可选的，若没有需要可不配置。
- `properties`元素下须有>=1个`property`元素。
- `property`元素是一个key-value值，均需要配置。其中key可为属性，也可为元素，写其一即可。value为元素。
- 若是使用function，则需要设置`function`元素及相应的参数`args`及`arg`元素。
- 此处配置的property元素的force属性，若force为false，则若在前面读取的配置文件（properties文件）中没有此配置，则添加，已有则忽略此配置。若force为true，则若在前面读取的配置文件（properties文件）中没有此配置，则添加，已有强制覆盖替换成此处配置。

4.2.4 executions/group元素

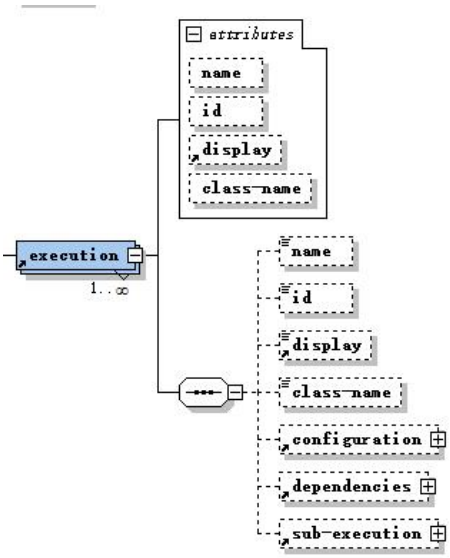


`executions`元素及`group`元素主要用于包含需要配置的操作，对操作进行分类。如产品部署服务器划分、产品模块划分，均可使用group进行分类。

说明：

> - `executions`元素是必需的，此元素下有>=1个`group`元素。
> - `group`元素主要是对产品服务器或产品模块划分，`name`可以是属性，也是可以是元素，写其一即可，以便显示。如上面安装tomcat示例中，*web服务安装与卸载*即为组名，显示时，部署工具会自动根据`group`的数量及位置添加序号。如第1个`group`，显示为**"1-web服务安装与卸载"**
> - `group`元素下有>=1个`execution`元素，`execution`元素是对操作的配置，部署工具就是根据execution的配置进行操作的。

4.2.5 execution元素



`execution`元素是需要执行的每一个操作，它通过设置对应的操作类，结合操作配置信息(`configuration`)，依赖(`dependencies`)，子操作(`sub-execution`)，组合成为一个完整的操作，以实现安装部署功能。如上面安装tomcat的示例中，使用`OperRunCommand`操作，并指定要执行的脚本路径或参数，即可完成tomcat安装。

```
<execution name="安装Tomcat" id="installWebTomcat" display="true" class-
name="deploy.OperRunCommand">
    <configuration>
        <commands>
            <command>
                <exec><![CDATA[scripts/windows/disk/webapp/install_tomcat.bat]]></exec>
                <args>
                    <arg><![CDATA[${web_extranet_port}]]></arg>
                </args>
            </command>
        </commands>
    </configuration>
</execution>
```

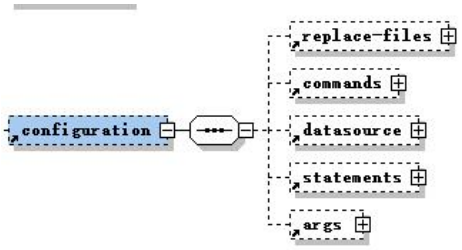
说明：

> - `execution`元素在显示时，会根据所在的`group`序号及当前`execution`在`group`中的位置自动添加序号。

如`group`的序号是1，当前`execution`的位置是1，则它显示的序号是"1.1"，后续的`execution`依次递增(1.2/1.3/1.4/...)。

- `name`,必填，可为属性或元素，写其一即可，根据实际操作命名。
- `id`，必填，可为属性或元素，写其一即可，此ID全局唯一，即其它execution的ID不能与之重复。此ID命名规则与java变量命名规则一致($ 、字母、下划线开头都行，后面的可以是数字、字母、下划线)。此ID在运行部署工具时，可作为运行参数输入，这样就不会出现交互界面，相当于直接执行此操作。
- `display`，必填，可为属性或元素，写其一即可，可选值为"true"或"false"。若为"true"，则在交互界面中会显示出来，供用户选择。若为"false"，则不在交互界面显示，用户也无法选择。
- `class-name`，必填，可为属性或元素，写其一即可。此值填写部署工具提供的操作类，每种操作类均有其相应的功能。当前共有7种操作，包括`OperRunCommand`运行命令操作,`OperGenerateLicense`生成许可证,`OperGenerateQrcode`生成二维码操作,`OperListConfig`显示当前属性配置操作,`OperRunDbStatement`运行sql语句操作,`OperRunDependency`只运行依赖的操作,`OperUpdateFile`更新配置文件操作。在填写`class-name`时，需要填写统一前缀`deploy.`，如`deploy.OperRunCommand`
- `configuration`元素，针对每种操作，有其相应的配置内容，请根据`class-name`对应的操作进行设置，详细见`configuration`元素的章节说明。
- `dependencies`元素，若当前`execution`操作是需要依赖其它操作，即在执行本`execution`时，会先执行`dependencies`中设置的依赖操作，待依赖操作完成后再执行本操作。详细见`dependencies`元素的说明。
- `sub-execution`元素，若此操作可分解为多个子操作，可在这里进行设置。同样，部署工具会自动添加序号，规则与`execution`的序号一致。如假设当前`execution`是2.1，`sub-execution`在中的有2个`execution`，则它们的序号分别是2.1.1及2.1.2。详细见`sub-execution`说明。

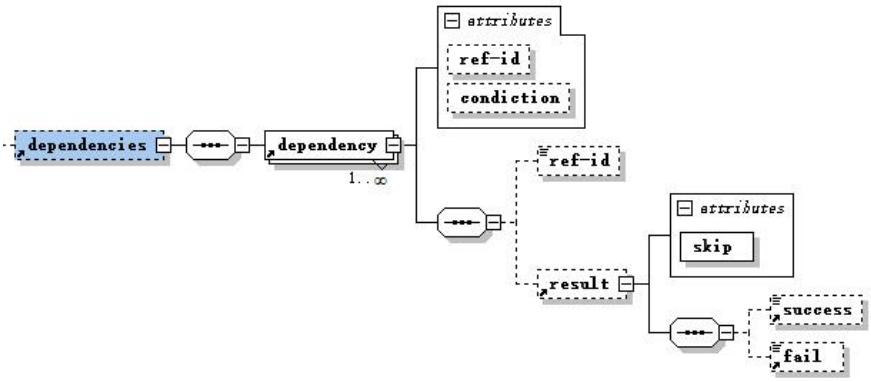
4.2.6 configuration元素



`configuration`元素是针对具体的每种操作，需要设置相应的配置信息，以完成此操作完成的功能。如上面的安装tomcat的示例中，是实现命令执行操作的配置信息。根据当前部署工具已经提供的操作类，主要配置内容如下。

- `commands`元素，适用于`OperRunCommand`运行命令操作。此操作可以运行脚本文件，执行单个shell语句，bat语句等。它使用`commands`元素的配置。其它可不配置。
- `datasource`元素及`statements`元素，适用于`OperRunDbStatement`运行sql语句操作,此操作可对数据库执行指定的sql语句，它使用`datasource`元素及`statements`元素，分别设置数据库连接信息，需要执行的sql语句。
- `OperRunDependency`只运行依赖的操作,此操作不需要使用`configuration`元素，只需要使用`dependencies`元素即可。因此可不配置。
- `replace-files`元素，适用于`OperUpdateFile`更新配置文件操作，此操作只使用`replace-files`元素。其它可不配置。
- `OperGenerateQrcode`生成二维码操作,此操作使用`args`元素，输入相应的参数（如二维码内容，图片宽度、图片高度，输出路径），即可生成二维码图片。
- `OperListConfig`显示当前属性配置操作,此操作不需要配置。

4.2.7 dependencies元素



`dependencies`元素主要设置当前`execution`元素需要依赖的操作，若设置了此依赖，在运行`execution`前会先按顺序执行依赖项，完成后再执行当前的`execution`。它主要在`OperRunDependency`操作及`OperRunCommand`操作中设置。如下所示是设置安装redis前，需要先更新配置文件、变更文件权限：

```
<execution name="安装Redis" id="installRedis" display="true" class-
name="deploy.OperRunCommand">
                <configuration>
                    <commands>
                        <command charset="utf-8">
                            <exec><![CDATA[scripts/linux/db/redis/install_redis.sh]]></exec>
                        </command>
                        <!-- 开通端口 -->
                        <command charset="utf-8">
                            <exec><![CDATA[scripts/linux/common/add_port.sh]]></exec>
                            <args>
                                <arg><![CDATA[${server.redis.port}]]></arg>
                            </args>
                        </command>
                    </commands>
                </configuration>
                <dependencies>
                    <dependency ref-id="updateConfigFiles"/>
                    <dependency ref-id="chmodFile"/>
                </dependencies>
            </execution>
```

说明：

- `dependencies`元素下有>=1个`dependency`元素。
- `dependency`元素使用`ref-id`表示依赖的`execution`，`ref-id`值为要执行的`execution`的id。`ref-id`可为属性或元素，写其一即可。
- `dependency`元素可使用`condiction`属性设置条件，对符合此条件的才执行此`ref-id`的操作，不符合条件则不操作。如使用`condiction="${server_web_extranet_protocol}==https"`，表示当用户使用https部署，设置`server_web_extranet_protocol`的值为https时符合条，若不等于https，则不符合，跳过此依赖。
- `dependency`元素下可使用`result`元素设置此依赖的操作执行后，根据它的执行结果如何进行下一步。若不设置此元素，则采用默认操作。即

  `<result skip="false">`

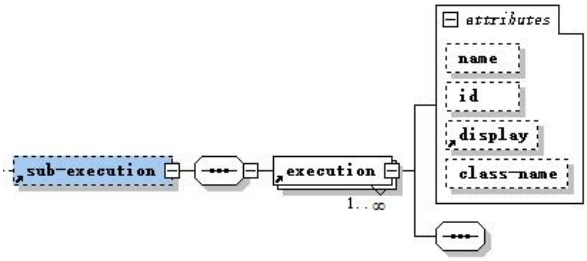
  `<success>move</success>`

  `<fail>stop</fail>`

  `</result>`

> 即当此依赖执行成功后，才进行下一个依赖操作，失败后即停止，返回到交互界面。如果设置为`skip`为true，则不管此依赖是否执行成功，都会进行下一步操作。当`skip`为true时，`success`及`fail`元素不用设置。

4.2.8 sub-execution元素



`sub-execution`元素用于对当前execution的分解,若此操作可分解为多个子操作，可在这里进行设置。`sub-execution`元素下可有>=1个`execution`元素。`execution`元素的定义与前面介绍的`execution`元素一致。

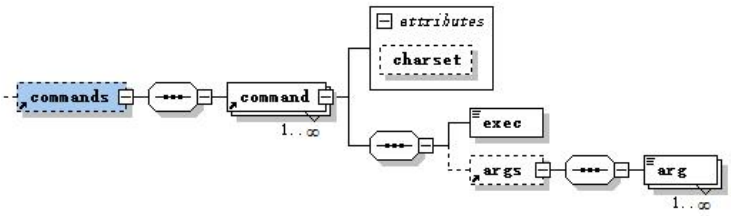
部署工具会对子操作自动添加序号，规则与`execution`的序号一致。如假设当前`execution`是2.1，`sub-execution`在中的有2个`execution`，则它们的序号分别是2.1.1及2.1.2。在交互界面中，当用户选择2.1.1时，即会执行对应的子操作。

一般有子操作时，会把当前`execution`元素设置为`OperRunDependency`操作类型，然后使用`dependencies`元素，把此操作依赖它的`sub-execution`，这样，即可在用户选择2.1时，自动执行2.1.1和2.1.2。如下面示例，安装消息服务，就是"安装Erlang与RabbitMQ"和"安装MsgService"，因此把`installMessageService`设置为`OperRunDependency`，然后依赖这两个操作：

```
<execution name="安装消息服务" id="installMessageService" display="true" class-name="deploy.OperRunDependency">
    <dependencies>
        <dependency ref-id="installErlangAndRabbitMQ">
            <result skip="true" />
        </dependency>
        <dependency ref-id="installMsgService"/>
    </dependencies>
    <sub-execution>
        <execution name="安装Erlang与RabbitMQ" id="installErlangAndRabbitMQ" display="true" class-name="deploy.OperRunCommand">
            <configuration>
                <commands>
                    <command>
                        <exec><![CDATA[scripts/windows/test/message/install_message_erlang_mq.bat]]></exec>
                    </command>
                </commands>
            </configuration>
        </execution>
        <execution name="安装MsgService" id="installMsgService" display="true" class-name="deploy.OperRunCommand">
            <configuration>
                <commands>
                    <command>
                        <exec><![CDATA[scripts/windows/test/message/install_msg_service.bat]]></exec>
                        <args>
                            <arg><![CDATA[${server_msg_tcp_port}]]></arg>
```

```
                        </args>
                    </command>
                </commands>
            </configuration>
            <dependencies>
                <dependency ref-id="updateConfigFiles"/>
            </dependencies>
        </execution>
    <sub-execution>
</execution>
```

4.2.9 commands元素



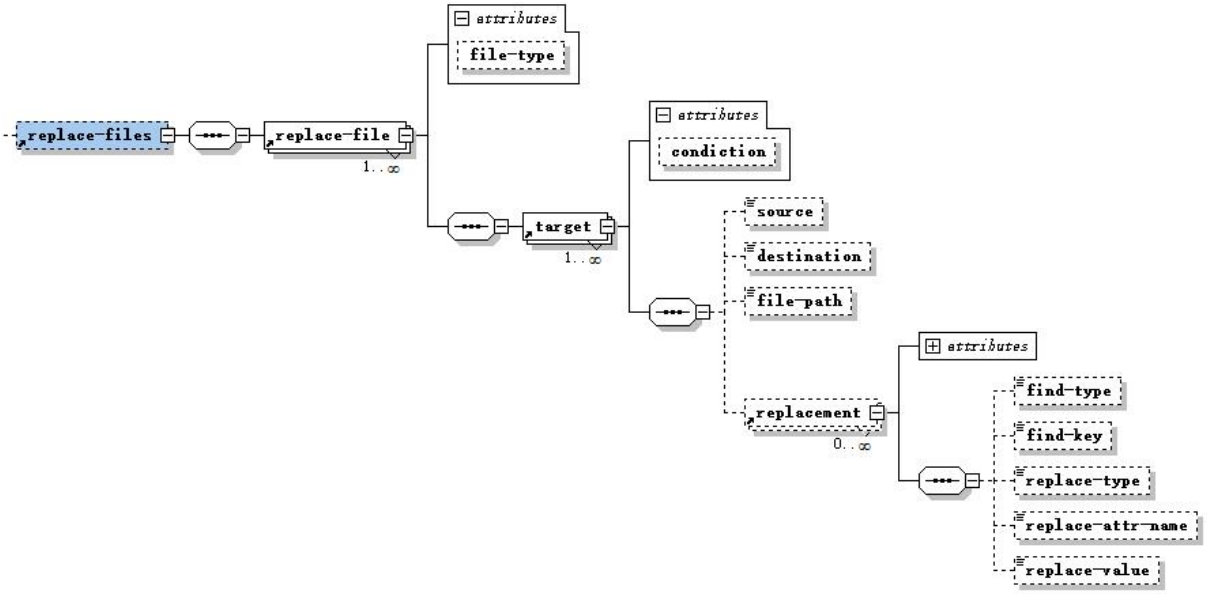
`commands`元素是在操作类型为`OperRunCommand`的时候使用，用于运行命令操作，即通过部署工具执行脚本文件、或执行shell语句，bat语句等。如当用户需要执行脚本文件，且给脚本传递参数，可按如下设置，此设置表示执行路径`scripts/windows/test/message/install_msg_service.bat`的文件，并传递占位符参数`$${server_msg_tcp_port}`，以供bat脚本使用：

```
<configuration>
    <commands>
        <command>
            <exec><![CDATA[scripts/windows/test/message/install_msg_service.bat]]></exec>
            <args>
                <arg><![CDATA[$${server_msg_tcp_port}]]></arg>
            </args>
        </command>
    </commands>
</configuration>
```

说明：

> - `commands`元素下有>=1个`command`元素，每个command对应一个运行命令，部署工具会依次执行。
> - `command`元素下包含`exec`及`args`元素，分别是要执行的命令及传递的参数。
> - `exec`元素是需要执行的命令，它可以是脚本文件，也可以是某一脚本命令。若是脚本文件，可写脚本文件的绝对路径或相对路径（相对install目录的路径）。若是脚本命令，则直接写命令代码即可。
> - `args`元素是传递给`exec`元素命令的参数。`args`下可有多个`arg`元素，一个`arg`对应一个参数。

4.2.10 replace-files元素



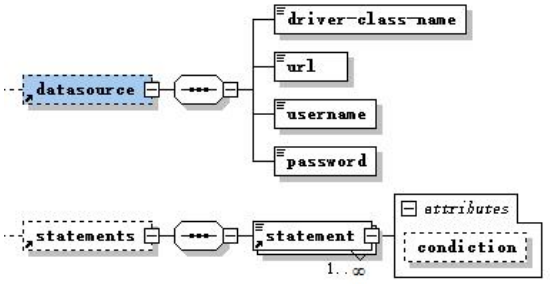
`replace-files`元素主要用于设置文件替换，它包含>=1个`replace-file`元素，`replace-file`元素表示进行文件替换操作，并通过`file-type`属性指定替换方式，部署工具当前支持的文件替换方式有三种：

(1) **使用模板文件替换**：`file-type`属性设置为`template`，此方式会对使用指定的模板文件，部署工具对模板文件进行占位符替换后，直接替换到指定的目录中。若指定目录已有同名文件，则会对同名文件添加`_backup`后缀进行备份。使用此方式，在`target`元素中需要设置模板文件位置`source`元素及替换目标文件位置`destination`。`target`元素可以设置多个，以进行多个文件的替换。
(2) **对xml文件的查找替换**：对xml文件，可设置查找匹配相应的元素或属性，然后对匹配到的元素或属性进行替换操作。与模板替换方式不同，此操作是局部替换。首次替换时，会对要替换的文件进行添加"_backup"备份，若发现备份文件已有，则不会再备份。它需要设置`target`元素下的`file-path`元素及`replacement`元素。`file-path`元素指定需要替换的xml文件位置，`replacement`指定需要查找的方式及需要替换的值。`replacement`元素包含`find-type`,`find-key`,`replace-type`,`replace-attr-name`,`replace-value`元素，分别是:
- `find-type`查找类型：支持`attribute`,`element`,`xpath`三种查找方式，即查找属性，查找元素，xpath查找
- `find-key`查找值：若查找类型是`attribute`，则可以设置为`attribute=vlue`，如需要查找属性值为"name",值为"workDir"的元素，则设置为"name=workDir"即可。
- `replace-type`替换类型：支持`attribute`,`element`两种，即替换属性值、替换元素值。
- `replace-attr-name`替换属性的名称：找到元素后，若是替换属性值，则设置此属性名。如上面查找到"name",值为"workDir"的元素，但是要替换此元素的"value"属性，此处应设置为"value"。若是替换元素值，则不需要设置此元素。
- `replace-value`替换值：找到元素后，确定要替换的属性或元素，则把属性或元素值设置为此值。
(3) **对properties文件的匹配替换**：对properties文件，可设置查找匹配相应的key，然后替换此key的值。首次替换时，会对要替换的文件进行添加"_backup"备份，若发现备份文件已有，则不会再备份。它需要设置`target`元素下的`file-path`元素及`replacement`元素。`replacement`元素需设置`find-key`,`replace-value`元素：、
- `find-key`查找值：若查找key为`server_msg_tcp_port`，直接设置此值即可。
- `replace-value`替换值：找到key后，把此key的值设置为此值。

> - `target`元素都可添加`condiction`以限制执行此替换的条件，例如只有是https部署时，才会替换某文件，则在`target`元素中添加属性`condiction="$${server_web_extranet_protocol}==https"`，若占位符`$${server_web_extranet_protocol}`值不是https，则会跳过此target，不进行操作。

- `replace-file`元素下所有元素的值建议都添加在`<![CDATA[]]>`中，以免出现特殊字符匹配失败的情况。

4.2.11 datasourse/statements元素



`datasourse`/`statements`元素是用于设置与数据库操作相关的配置，主要在`OperRunDbStatement`操作中进行设置。`datasourse`元素设置数据库连接连接信息，与jdbc连接数据库内容一致，包括`driver-class-name`,`url`,`username`,`password`。如下示例使用配置文件中的占位符设置：

```
<datasource>
    <driver-class-name><![CDATA[com.mysql.jdbc.Driver]]></driver-class-name>
    <url><!
[CDATA[jdbc:mysql://$${server_mysql_ip}:$${server_mysql_port}/$${server_mysql_dbname}?useUnicode=true&characterEncoding=utf8]]></url>
    <username><![CDATA[$${server_mysql_db_username}]]></username>
    <password><![CDATA[$${server_mysql_db_password}]]></password>
</datasource>
```

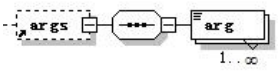
`statements`元素是需要执行sql语句。它包含>=1个`statement`元素，每个`statement`元素对应一条sql语句，一般在部署中使用较多的是`insert`,`update`,`delete`等语句。如下所示：

```
<statements>
    <statement>
        <![CDATA[UPDATE `$${server_mysql_dbname}`.`sys_config` SET `value` = '$${deployment_home}/system_file/test/source'
      WHERE `name` = 'store.server.folder.source';]]>
     </statement>
     <statement>
      <![CDATA[UPDATE `$${server_mysql_dbname}`.`sys_config` SET `value` = '$${deployment_home}/system_file/test/temp'
        WHERE `name` = 'store.server.folder.temp';]]>
    </statement>
</statements>
```

注意：

- 建议把在运行部署工具时才确定的变量，同时需要应用到数据库中的时候才放在此处执行。如上述示例中，根据当前部署路径更新到数据库表中。
- 不建议把大量的sql语句在此执行，若需要执行大量sql语句，可写在sql脚本中，然后通过`OperRunCommand`操作执行shell/bat脚本。
- 元素的值建议都添加在`<![CDATA[]]>`中，以免出现特殊字符匹配失败的情况。

4.2.12 args元素



`args`元素当前主要用在`OperGenerateQrcode`生成二维码操作中，为操作提供相应的参数，且参数的顺序是固定的。如下示例是生成二维码：

```
<execution name="生成下载移动端二维码图" id="generateQrCode" display="true" class-name="deploy.OperGenerateQrcode">
    <configuration>
        <args>
            <!-- 二维码内容 -->
            <arg><![CDATA[http://$${server_web_extranet_host}:$${server_web_intranet_port}/data]]></arg>
            <!-- 图片宽度 -->
            <arg><![CDATA[110]]></arg>
            <!-- 图片高度 -->
            <arg><![CDATA[110]]></arg>
            <!-- 生成图片存放绝对路径,包含图片名称及后缀，如scan.png -->
            <arg><![CDATA[$${deployment_home}/package/test/qrcode.png]]></arg>
        </args>
    </configuration>
</execution>
```

说明：

> - OperGenerateQrcode操作，参数顺序为（1）二维码内容，（2）图片宽度，（3）图片高度，（4）生成图片存放绝对路径

4.3 流程配置文件功能示例

了解配置文件各元素结构与作用后，下面以数据库mariadb在linux下安装卸载为示例，通过组合各元素，完成对数据库的安装。分以下三步实现：

4.3.1 分析安装及卸载mariadb需要的模块

安装及卸载mariadb，可分为（1）mariadb安装，（2）卸载。而在安装过程中需要进行：（3）配置文件替换，（4）修改文件执行权限，（5）同步调整系统时间。各模块均可独立运行，也有依赖关系。如在安装mariadb前需先完成(3)(4)(5)操作。最后，把mariadb组合为安装及卸载显示。如此分析，xml的框架就可以出来，如下图：

```
<group name="MySQL安装与卸载">
    <execution name="安装MySQL及初始化数据库" id="installMysqlAndInitDb" display="true" class-name="deploy.OperRunDependency">
    <execution name="卸载MySQL" id="uninstallMysql" display="true" class-name="deploy.OperRunCommand">
</group>
<group name="基本功能">
    <execution name="更新配置及脚本文件" id="updateConfigFiles" display="true" class-name="deploy.OperUpdateFile">
    <execution name="修改文件执行权限" id="chmodFile" display="true" class-name="deploy.OperRunCommand">
    <execution name="调整系统时间同步间隔" id="syncOsTime" display="true" class-name="deploy.OperRunCommand">
    <execution name="查看配置参数" id="listConfigData" display="true" class-name="deploy.OperListConfig"/>
</group>
```

mariadb安装、卸载，其中"安装MySQL及初始化数据库"，分为两个操作，分别是安装及初始化。而基本功能就是（3）（4）（5）功能。

4.3.2 确定用户统一配置

安装mariadb，需要配置的内容包括数据库，服务器IP端口信息等等，这些信息设置在custom_config.properties文件

中，作为占位符在部署工具的配置中使用。

```
### mysql数据库配置
#数据库IP
server.mysql.host = 127.0.0.1
#数据库端口
server.mysql.port = 3306
#数据库名
server.mysql.dbname = mytest
#数据库用户名
server.mysql.db.username = root
#数据库密码
server.mysql.db.password = 123456
```

4.3.3 编写流程配置文件

经过上面的分析，这些模块中，每个模块都可以作为一个`execution`元素，mariadb的安装与卸载可为一组(group)（3）替换配置文件（4）修改文件执行权限（5）同步系统时间可为一组，作为基本功能。因此，

- 编写mariadb安装`execution`:操作是`OperRunDependency`类，它需要执行两个依赖(`dependency`)操作。而实际的操作是两个子操作(`sub-execution`)，它们是`OperRunCommand`类，一个是安装数据库，一个是初始化数据库。分别通过执行`install_mysqldb.sh`和`import_mysql_sql.sh`实现。安装同时开通端口，安装前需先执行（3）（4）（5），因此使用`dependency`来实现。因此，xml配置如下：

```
<execution name="安装MySQL及初始化数据库" id="installMysqlAndInitDb" display="true" class-name="deploy.OperRunDependency">
    <dependencies>
        <dependency ref-id="installMysql"/>
        <dependency ref-id="createMysqlDb"/>
    </dependencies>
    <sub-execution>
        <execution name="安装MySQL" id="installMysql" display="true" class-name="deploy.OperRunCommand">
            <configuration>
                <commands>
                    <command charset="utf-8">
                        <exec><![CDATA[scripts/linux/db/mariadb/install_mysqldb.sh]]></exec>
                        <args>
                            <arg><![CDATA[${server.mysql.db.password}]]></arg>
      </args>
     </command>
     <!-- 开通端口 -->
     <command charset="utf-8">
      <exec><![CDATA[scripts/linux/common/add_port.sh]]></exec>
      <args>
       <arg><![CDATA[${server.mysql.port}]]></arg>
                        </args>
                    </command>
                </commands>
            </configuration>
            <dependencies>
                <dependency ref-id="updateConfigFiles"/>
                <dependency ref-id="chmodFile"/>
                <dependency ref-id="syncOsTime"/>
            </dependencies>
        </execution>
        <execution name="初始化MySQL数据库" id="createMysqlDb" display="true" class-name="deploy.OperRunCommand">
            <configuration>
                <commands>
```

```
                    <command charset="utf-8">
                        <exec><![CDATA[scripts/linux/db/mariadb/import_mysql_sql.sh]]></exec>
                        <args>
                            <arg><![CDATA[${server.mysql.db.username}]]></arg>
        <arg><![CDATA[${server.mysql.db.password}]]></arg>
                            <arg><![CDATA[${server.mysql.port}]]></arg>
                        </args>
                    </command>
                </commands>
            </configuration>
            <dependencies>
                <dependency ref-id="updateConfigFiles"/>
                <dependency ref-id="chmodFile"/>
                <dependency ref-id="syncOsTime"/>
            </dependencies>
         </execution>
    </sub-execution>
</execution>
```

- 由于要把mariadb安装和卸载为一组显示，添加组合(`group`)用于显示，即可。如下：

```
<group name="MySQL安装与卸载">
    安装及卸载的execution
</group>
```

此示例，使用部署工具运行后，显示结果如下：

```
****************************** 2-MySQL安装与卸载
2.1: 安装MySQL及初始化数据库
 |--2.1.1: 安装MySQL
 |--2.1.2: 初始化MySQL数据库
2.2: 卸载MySQL
```

4.4 部署脚本编写

在编写流程部署文件时，大部分部署功能都应在脚本中实现，脚本先验证部署通过后，再放到部署工具中使用。部署脚本在linux中使用shell编写，windows中使用bat脚本编写。具体shell及bat的语法不在这里介绍（读者请自己学习）。下面是linux下安装mysql的脚本示例：

```
#!/bin/bash
#安装mysql
BASE_DIR=$(dirname $(pwd))
MYSQL_HOME=${BASE_DIR}/software/mariadb
COMMON_HOME=${BASE_DIR}/install/scripts/linux/common

db_password=$1

AUTO_RUN_FILE=${MYSQL_HOME}/support-files/mysql.server
AUTO_RUN_FILE_NAME=mysqld

#安装mysql的依赖包
if [[ $MYSQLDB_HAS_INSTALL != "y" ]];then
    #1.安装依赖
    yum install -y cmake gcc gcc-c++ ncurses ncurses-devel openssl openssl-devel perl

    #2.创建mysql用户，查看是否有mysql用户，没有则创建
    grep mysql /etc/passwd &>/dev/null
    if [ "$?" -ne 0 ];then
   useradd mysql
 fi
 #授权mysql用户的mysql目录权限
 chown -R mysql:mysql ${MYSQL_HOME}
```

```
    #3.创建数据库文件
    echo -n "正在安装mysql数据库....."
    cd ${MYSQL_HOME} && ./scripts/mysql_install_db --defaults-file=${MYSQL_HOME}/my.cnf &>/dev/null &
    while true
    do
       ps -ef | grep -w mysql_install_db | grep -v "grep" &>/dev/null
       if [ "$?" -ne 0 ];then
           echo "[ok]"
           break
       else
           echo -n "....."
           sleep "0.5"
       fi
    done

    #启动mysql
    cd ${MYSQL_HOME} && ./bin/mysqld_safe --defaults-file=${MYSQL_HOME}/my.cnf &>/dev/null &
    echo -n "正在启动mysql"
    while true
    do
      if [ ! -e  ${MYSQL_HOME}/var/mysqld.pid ];then
        echo -n "........"
        sleep "0.5"
      else
        break
      fi
    done
    echo "[ok]"

    #开机自启动
    \cp -f ${MYSQL_HOME}/my.cnf /etc/my.cnf
 bash ${COMMON_HOME}/set_auto_start.sh ${AUTO_RUN_FILE} ${AUTO_RUN_FILE_NAME}

    # Update root passwd
    CMD_MYSQL="${MYSQL_HOME}/bin/mysql -uroot -S ${MYSQL_HOME}/var/mysql.sock"
    $CMD_MYSQL -e "use mysql; delete from mysql.user where user='';"
 $CMD_MYSQL -e "use mysql; update user set password=password('${db_password}') where user='root'"
 $CMD_MYSQL -e "GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY '${db_password}'";
 $CMD_MYSQL -e "flush privileges;";

    echo "安装mysql数据库完成"
    #标记mysql数据库为已安装
    export MYSQLDB_HAS_INSTALL=y
fi
```

由示例可见，一般执行脚本，都建议先定位到部署目录，然后基于部署目录设置相应的操作目录，额外的参数可通过xml流程配置中输入。其它的就是普通脚本编写了。脚本编写完成后，可统一放到scripts目录下，并按模块分目录，便于查找即可。

注意：

> - windows下bat脚本，使用`%~dp0`定位当前bat的目录。
> - linux下shell脚本，使用`$(pwd)`定位运行部署工具（deploy-tool.jar）的sh文件所在位置（即install目录）。因此获取部署目录，使用`$(dirname $(pwd))`即可。
> - linux下若使用sh文件A调用另外一个sh文件B，而sh文件B是使用`ps|grep|awk`获取进程号（以便进行kill操作），会有子Shell的（临时）进程的问题，导致获取的进程号不正确。需要对文件A与文件B进行子进程过滤。如可使用`PID=`ps aux|grep -v grep|grep -v A|grep -v B|awk '{print $2}'`。其中A及B是文件名。

5. 完整db(mariadb及redis)部署示例

5.1 mariadb及redis部署结构分析

以linux部署为例，db分为软件安装包和资源，软件安装包如mariadb及redis的二进制绿色安装包，资源则是数据库的初始化脚本。

5.1.1 模块划分

db分为mariadb安装卸载及redis安装卸载，可一起部署，也可分别部署在不同的服务器。两个安装包均是绿色免安装版。安装数据库时需要初始化sql脚本，可把这些额外文件作为资源文件，独立一个文件夹。

另外部署工具及程序运行需要使用jre8，可以把这些共用的环境作为软件统一存放到固定位置。

基于上述分析，db部署结构划分三部分,install(部署工具相关),resource(资源相关)，software（软件包），如下所示：

5.1.2 部署环境包制作

由产品发布人员对部署环境包进行制作，按前面的规划，制作流程是：
(1). 创建好相应的目录结构（如`resource/software/install`）
(2). 存放公共使用的环境到`software`目录。如存放`jre8`。
(3). 存放各模块软件包到`software`，如存放mariadb及redis软件包。
(4). 存放部署工具(`deploy-tool.jar`)到`install`目录，并提供运行部署工具的脚本，如`run-deploy-tool.sh`。
(5). 在`install/config`目录下添加`global_config.properties`配置，指定产品名称,如`db`。同时需要确定各程序需要修改的配置统一到`custom_config.properties`中，并存放在此目录。若有其它默认配置，也可在此目录下进行添加。
(6). 创建`install/config/deploy-config`，添加流程配置文件`{deploy_product}-{system}.xml`，根据产品名称及部署系统填写。如当前示例为`db-linux.xml`。
(7). 按实际部署流程设计及编写流程配置文件。
(8). 在`install/scripts`目录下，添加部署需要使用的各种脚本，由于支持多种系统，建议在`install/scripts`目录下按系统类型建立文件夹，如当前是linux的，应新建`install/scripts/linux`文件夹，并把相应的sh脚本按模块存放在此目录下。

通过上面的处理，部署环境包就已完成。环境包完成后，产品发布人员需要先自己经过验证。因此，下面是验证过程：

- 下载对应版本的程序或软件包（可从开发/运维人员、jenkins或打包平台中获取）。把程序放到相应的目录中，如mariadb的软件存放在`software`目录下。
- 编辑`custom_config.properties`文件，按实际情况修改相应的配置信息，以便在占位符替换时使用实际的配置。
- 至此，部署工具+程序包已完整，可以验证部署，在linux下，修改`run-deploy-tool.sh`执行权限(chmod)，然后运行，在交互界面中选择进行安装即可。

验证通过后，部署环境即可发布，以供项目实施人员使用。在发布前，建议先程序去除，这样规范项目实施人员在部署时，需先下载程序包，根据实际情况修改`custom_config.properties`文件，然后才能正常启动部署。

5.1.3 项目实施人员使用流程

(1). 下载部署环境包
(2). 下载对应版本程序包
(3). 根据实际情况修改`custom_config.properties`文件。（若是从打包平台下载，此步骤可省略）。
(4). 运行`run-deploy-tool.sh`或`run-deploy-tool.bat`，在交互界面中选择进行安装即可。

5.2 db部署包示例及脚本

5.2.1 linux部署

- mariadb及redis安装linux部署示例，链接地址：linux部署db示例
  提取码：`l8qu`
- 压缩包中有使用帮助文档`help.txt`，按里面说明操作即可。

5.2.2 windows部署

- mariadb及redis安装windows部署示例，链接地址：windows部署db示例
  提取码：`kohq`
- 压缩包中有使用帮助文档`help.txt`，按里面说明操作即可。

5.3 部署环境升级

由于已使用模块化的方式进行安装部署，部分环境的升级，产品发布人员可直接修改相应的目录及脚本，验证通过后进行发布，并说明此部署环境的更新时间，对应的程序版本。

6. 问题与反馈

- 脚本文件问题，可反馈给产品发布人员。
- 程序运行问题，可反馈给产品发布人员。
- 部署工具使用问题，可反馈到我邮箱：mianshenglee@foxmail.com

Author: mason lee
Date : 2018-12-20