Grails项目有多种部署方式，每一种部署方式都有其优势和劣势。

# grails run-app

grails run-app命令是在开发活动中最常用的一个命令。该命令会启动一个grails内嵌的Tomcat 服务器并部署上web应用。开发的源代码有任何改动都会被抓取重新装载到tomcate上。

这种方式因为需要实时检查代码状态并自动热部署任何有变动的源代码，导致应用的性能不是很好，所以不建议在生产环境使用。

run-app命令默认使用dev环境配置运行应用，也可以在命令中指定应用要运行的环境：

|  |
| --- |
| grails prod run-app//针对生产环节配置运行应用 |

# grails run-war

run-war命令和run-app命令类似，不同之处是run-war部署一个war包到tomcat上，所以没有代码的热部署功能。

# war

war命令就是打包命令，打出的war文件用于部署到生产环境：

|  |
| --- |
| grails war |

war命令默认使用prod环境的配置信息。

war命令也可以自定义输出的war文件，例如输出目录和文件名：

|  |
| --- |
| grails war /opt/java/tomcat-5.5.24/foobar.war |

war命令还可以控制哪些jar包需要被打到war的lib目录下面。默认情况下Grails会将项目依赖到的所有jar包都打到war包的lib目录下，这有时候会导致jar包和服务器本身的jar包冲突。可以在BuildConfig.groovy文件中设置属性grails.war.dependencies来明确说明哪些jar包需要打到war包内:

|  |
| --- |
| def deps = [  "hibernate3.jar",  "groovy-all-\*.jar",  "standard-${servletVersion}.jar",  "jstl-${servletVersion}.jar",  "oscache-\*.jar",  "commons-logging-\*.jar",  "sitemesh-\*.jar",  "spring-\*.jar",  "log4j-\*.jar",  "ognl-\*.jar",  "commons-\*.jar",  "xstream-1.2.1.jar",  "xpp3\_min-1.1.3.4.O.jar" ]  grails.war.dependencies = {  fileset(dir: "libs") {  for (pattern in deps) {  include(name: pattern)  }  }  } |

两台另个配置项是grails.war.copyToWebApp和grails.war.resources。

grails.war.copyToWebApp用来定义web-app目录下的哪些内容会被打包到WAR文件中。

grails.war.resources可以定义一些在WAR包最终生成前的一些额外的工作。

|  |
| --- |
| // This closure is passed the command line arguments used to start the  // war process.  grails.war.copyToWebApp = { args ->  fileset(dir:"web-app") {  include(name: "js/\*\*")  include(name: "css/\*\*")  include(name: "WEB-INF/\*\*")  }  }  // This closure is passed the location of the staging directory that  // is zipped up to make the WAR file, and the command line arguments.  // Here we override the standard web.xml with our own.  grails.war.resources = { stagingDir, args ->  copy(file: "grails-app/conf/custom-web.xml",  tofile: "${stagingDir}/WEB-INF/web.xml")  } |