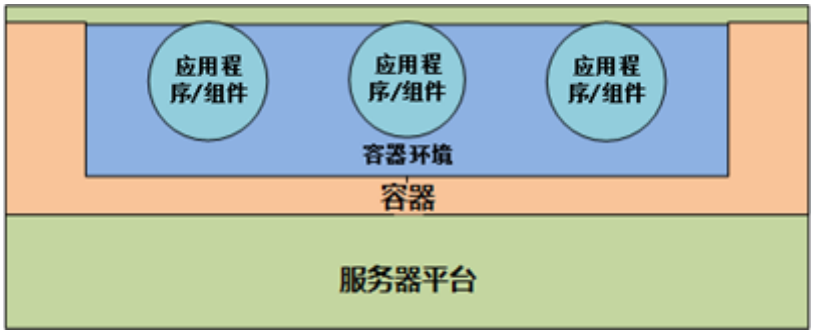
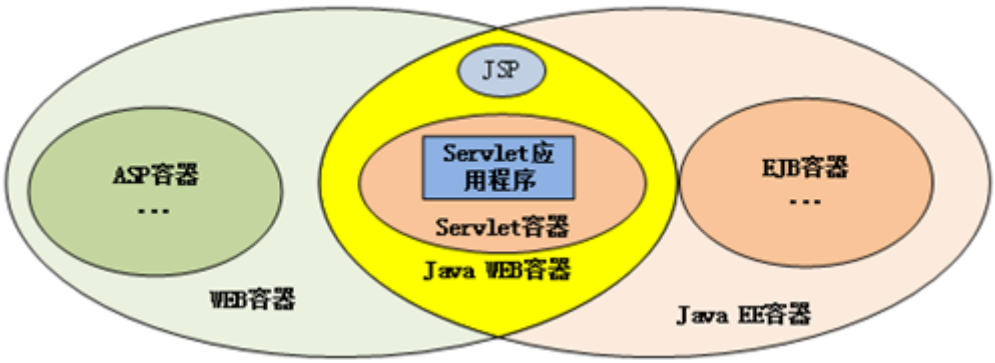
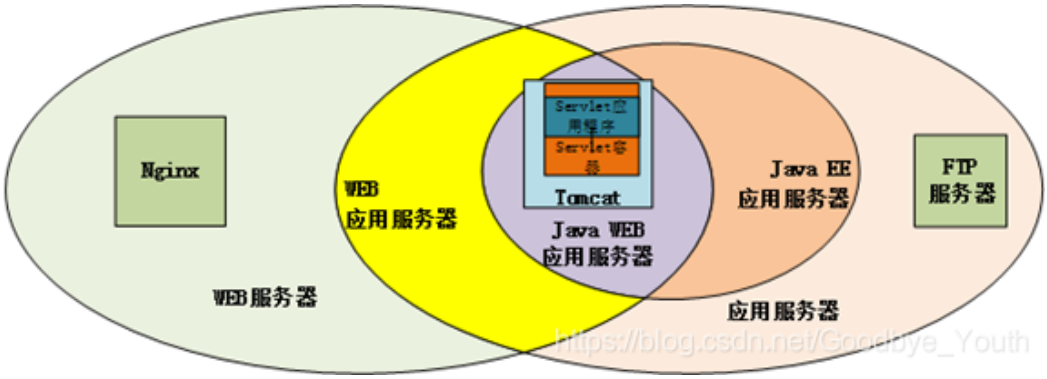
引自：https://blog.csdn.net/Goodbye_Youth/article/details/94983458

1.容器概念理解

> *容器是位于应用程序/组件和服务器平台之间的接口集合，使得应用程序/组件可以方便部署到服务器上运行*
>
> *servlet是一种服务端的 **Java** 应用程序，用于处理用户请求和响应数据，是一门用于开发动态 **web** 资源的技术*
>
> *servlet容器实现了 **Servlet** 技术规范的部署环境，它可以部署运行 **Servlet** 程序*







2.tomcat目录结构

```
bin
存放启动和关闭 Tomcat 及其他一些脚本文件
```

```
conf
存放 Tomcat 的各种配置文件

lib
存放 Tomcat 所需要的各种 jar 包

logs
存放 Tomcat 的日志文件

temp
存放 Tomcat 运行时产生的临时文件

webapps
Web 应用所在目录，即供外界访问的 Web 资源的存放目录

work
存放 Tomcat 运行时生成的文件，Tomcat 会将 JSP 页面生成的 Servlet 源文件和字节码文件放在该目录下
```

3.server.xml配置文件中各个端口的作用

```
8005
负责监听关闭 Tomcat 的请求，当监听到关闭请求时，会向该端口发送关闭服务器的命令字符串（即“SHUTDOWN”）

8080
HTTP 服务连接器端口，负责建立 HTTP 连接，即我们平常通过浏览器访问部署在 Tomcat 上 Web 应用的端口

8009
AJP 服务连接器端口，负责与其他 HTTP 服务器进行连接（主要用于和 Apache 进行连接）

8443
HTTPS 服务连接器端口，需要配置 SSL 证书
```

4.日志问题（logging.properties配置文件详解）

一、日志级别

- tomcat日志级别：SEVERE (highest value) > WARNING > INFO > CONFIG > FINE > FINER > FINEST (lowest value)
- log4j日志级别：

级别	名称	说明
	ALL	最低等级的，用于打开所有日志记录。
	TRACE	designates finer-grained informational events than the DEBUG.Since:1.2.12，很低的日志级别，一般不会使用。
	DEBUG	指出细粒度信息事件对调试应用程序是非常有帮助的，主要用于开发过程中打印一些运行信息。
	INFO	消息在粗粒度级别上突出强调应用程序的运行过程。打印一些你感兴趣的或者重要的信息，这个可以用于生产环境中输出程序运行的一些重要信息，但是不能滥用，避免打印过多的日志。
	WARN	表明会出现潜在错误的情形，有些信息不是错误信息，但是也要给程序员的一些提示。
	ERROR	指出虽然发生错误事件，但仍然不影响系统的继续运行。打印错误和异常信息，如果不想输出太多的日志，可以使用这个级别。
	FATAL	指出每个严重的错误事件将会导致应用程序的退出。这个级别比较高了。重大错误，这种级别你可以直接停止程序了。
	OFF	最高等级的，用于关闭所有日志记录。

二、日志分类

- **catalina.out**
    - 记录了 Tomcat 运行时自身输出的日志以及程序中向控制台 (console) 输出的日志
    - 可以在 Tomcat 的启动脚本 (`catalina.sh`) 中进行配置

```
if [ -z "$CATALINA_OUT" ] ; then
  CATALINA_OUT="$CATALINA_BASE"/logs/catalina.out
fi
```

- **catalina.YYYY-MM-DD.log**

  记录了 Tomcat 运行时自身输出的日志，这些日志还会输出到 catalina.out 中

  程序中向控制台 (console) 输出的日志不会输出到 catalina.YYYY-MM-DD.log 中
- **localhost.YYYY-MM-DD.log**

  记录了程序初始化 (listener, filter, servlet) 时，未处理的异常最后被 Tomcat 捕获而输出的日志

  这些未处理异常最终会导致程序无法启动
- **localhost_access_log.YYYY-MM-DD.txt**

  记录了 Tomcat 的访问日志，记录了访问地址、请求时间、请求路径、状态码等信息
- **manager.YYYY-MM-DD.log**

  记录了 Tomcat 自身 manager 项目 (用于查看 Tomcat Web 应用管理器) 输出的日志
- **host-manager.YYYY-MM-DD.log**

  记录了 Tomcat 自身 host-manager 项目 (用于查看 Tomcat 虚拟机管理器) 输出的日志

三、配置文件详解

```
############################################################
# Handler specific properties.
# Describes specific configuration info for Handlers.
############################################################

// 设置 catalina 日志输出级别为 FINE，默认为 INFO
1catalina.org.apache.juli.AsyncFileHandler.level = FINE
```

```
// 设置 catalina 日志存放路径，默认为 ${catalina.base}/logs
1catalina.org.apache.juli.AsyncFileHandler.directory =
${catalina.base}/logs
// 设置 catalina 日志名称前缀，默认为 catalina.
1catalina.org.apache.juli.AsyncFileHandler.prefix = catalina.
// 设置 catalina 日志最大保存天数，默认为 90 天
1catalina.org.apache.juli.AsyncFileHandler.maxDays = 90
// 设置 catalina 日志编码，默认为 UTF-8
1catalina.org.apache.juli.AsyncFileHandler.encoding = UTF-8

// 设置 localhost 日志输出级别为 FINE，默认为 INFO
2localhost.org.apache.juli.AsyncFileHandler.level = FINE
// 设置 localhost 日志存放路径，默认为 ${catalina.base}/logs
2localhost.org.apache.juli.AsyncFileHandler.directory =
${catalina.base}/logs
// 设置 localhost 日志名称前缀，默认为 localhost.
2localhost.org.apache.juli.AsyncFileHandler.prefix = localhost.
// 设置 localhost 日志最大保存天数，默认为 90 天
2localhost.org.apache.juli.AsyncFileHandler.maxDays = 90
// 设置 localhost 日志编码，默认为 UTF-8
2localhost.org.apache.juli.AsyncFileHandler.encoding = UTF-8

// 设置 manager 日志输出级别为 FINE，默认为 INFO
3manager.org.apache.juli.AsyncFileHandler.level = FINE
// 设置 manager 日志存放路径，默认为 ${catalina.base}/logs
3manager.org.apache.juli.AsyncFileHandler.directory =
${catalina.base}/logs
// 设置 manager 日志名称前缀，默认为 manager.
3manager.org.apache.juli.AsyncFileHandler.prefix = manager.
// 设置 manager 日志最大保存天数，默认为 90 天
3manager.org.apache.juli.AsyncFileHandler.maxDays = 90
// 设置 manager 日志编码，默认为 UTF-8
3manager.org.apache.juli.AsyncFileHandler.encoding = UTF-8

// 设置 host-manager 日志输出级别为 FINE，默认为 INFO
4host-manager.org.apache.juli.AsyncFileHandler.level = FINE
// 设置 host-manager 日志存放路径，默认为 ${catalina.base}/logs
4host-manager.org.apache.juli.AsyncFileHandler.directory =
${catalina.base}/logs
// 设置 host-manager 日志名称前缀，默认为 host-manager.
4host-manager.org.apache.juli.AsyncFileHandler.prefix = host-
manager.
// 设置 host-manager 日志最大保存天数，默认为 90 天
4host-manager.org.apache.juli.AsyncFileHandler.maxDays = 90
// 设置 host-manager 日志编码，默认为 UTF-8
4host-manager.org.apache.juli.AsyncFileHandler.encoding = UTF-8

// 设置控制台输出日志级别，默认为 FINE
java.util.logging.ConsoleHandler.level = FINE
// 设置控制台输出日志格式，默认为 org.apache.juli.OneLineFormatter
java.util.logging.ConsoleHandler.formatter =
org.apache.juli.OneLineFormatter
```

```
// 设置控制台输出编码，默认为 UTF-8（对应IDEA的server）
java.util.logging.ConsoleHandler.encoding = UTF-8

############################################################
# Facility specific properties.
# Provides extra control for each logger.
############################################################

// 设置 localhost 日志输出级别，默认为 INFO
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].level
= INFO
// 设置 localhost 日志输出处理类，默认为
2localhost.org.apache.juli.AsyncFileHandler
org.apache.catalina.core.ContainerBase.[Catalina].
[localhost].handlers = 2localhost.org.apache.juli.AsyncFileHandler

// 设置 manager 日志输出级别，默认为 INFO
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].
[/manager].level = INFO
// 设置 manager 日志输出处理类，默认为
3manager.org.apache.juli.AsyncFileHandler
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].
[/manager].handlers = 3manager.org.apache.juli.AsyncFileHandler

// 设置 host-manager 日志输出级别，默认为 INFO
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].
[/host-manager].level = INFO
// 设置 host-manager 日志输出处理类，默认为 4host-
manager.org.apache.juli.AsyncFileHandler
org.apache.catalina.core.ContainerBase.[Catalina].[localhost].
[/host-manager].handlers = 4host-
manager.org.apache.juli.AsyncFileHandler

// 设置 org.apache.catalina.util.LifecycleBase 的级别，默认为 FINE，用
于记录扩展 LifecycleBase 并更改状态的每个组件
#org.apache.catalina.util.LifecycleBase.level = FINE

// 设置 TldLocationsCache 的级别，默认为 FINE，用于查看调试消息
#org.apache.jasper.compiler.TldLocationsCache.level = FINE

// 设置 http2 的级别，默认为 FINE，用于查看 HTTP/2 处理的调试消息
#org.apache.coyote.http2.level = FINE

// 设置 websocket 的级别，默认为 FINE，用于查看 WebSocket 处理的调试消息
#org.apache.tomcat.websocket.level = FINE
```

5.tomcat上单实例多应用部署

> *特别注意：（总结第一次没成功的原因）*
>
> 1. *run.sh脚本文件中，$CATALINA_BASE1、$CATALINA_BASE2要改成正确的！*
> 2. *server.xml文件中，多应用之间下面三个端口不能重复：*
>
> ```
> <Server port="7023" shutdown="SHUTDOWN">
>
> <Connector port="7024" protocol="HTTP/1.1"
>            connectionTimeout="20000"
>            redirectPort="8443" />
>
> <Connector port="7025" protocol="AJP/1.3"
> redirectPort="8443" />
> ```

当我们在一台服务器上想要部署多个 Tomcat 实例时，在不了解 Tomcat 的相关参数时，我们通常会下载多份 Tomcat 来安装使用，而实际情况下，多个 Tomcat 实例可以共用同一个安装目录，现在就让我们来看看 Tomcat 多实例部署的流程

一、CATALINA_HOME 和 CATALINA_BASE 的区别

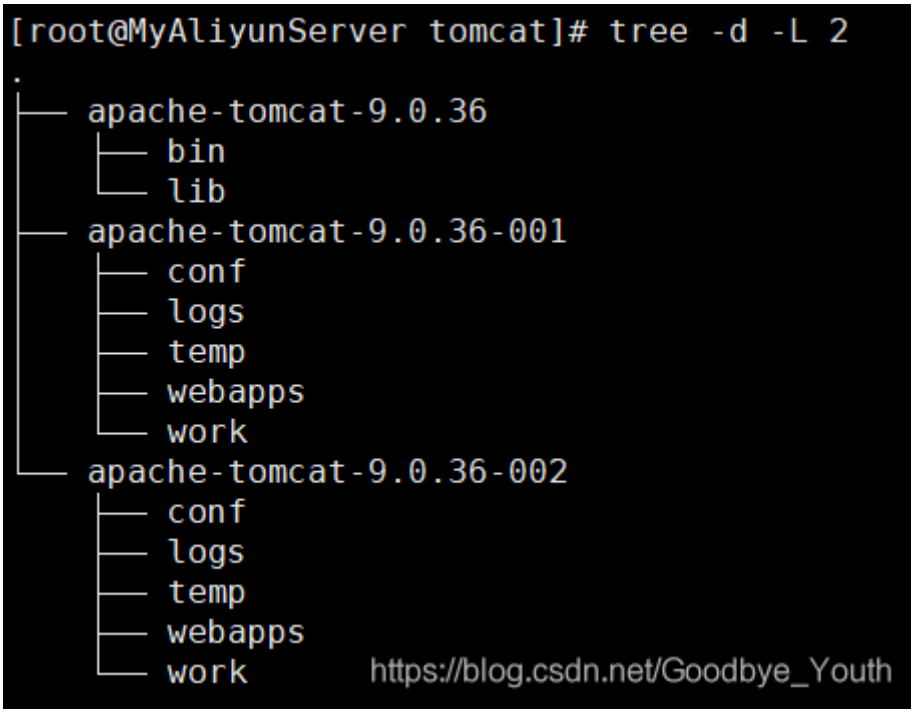
```
在了解流程之前，我们先来了解下 CATALINA_HOME 和 CATALINA_BASE 的区别

CATALINA_HOME
表示 Tomcat 安装的根目录
例如: /home/tomcat/apache-tomcat-9.0.10 或 C:\Program Files\apache-tomcat-9.0.10

CATALINA_BASE
表示特定 Tomcat 实例运行时配置的根目录（工作目录）
如果要在一台服务器上部署多个 Tomcat 实例，应该使用该属性
也就是说，如果我们想要运行多个 Tomcat 实例，但是又不想安装多个 Tomcat，那么我们就可以配置多个工作目录，每个运行实例独占一个工作目录，但是共享同一个安装目录
```

二、使用 CATALINA_BASE 的好处

- 便于 Tomcat 的升级管理，这是因为所有 Tomcat 实例共享安装目录中的二进制文件和相关 jar 包，升级的时候只需要更新安装目录 (CATALINA_HOME)，即可对所有 Tomcat 实例生效

三、多实例部署流程

创建应用工作空间目录：



vim /etc/profile：

```
#tomcat
export CATALINA_HOME=/usr/local/tomcat/apache-tomcat-9.0.36
export CATALINA_BASE1=/usr/local/tomcat/apache-tomcat-9.0.36-001
export CATALINA_BASE2=/usr/local/tomcat/apache-tomcat-9.0.36-002
```

编写控制脚本

- cd /usr/local/tomcat/apache-tomcat-9.0.36-001
- vim run.sh
- cd /usr/local/tomcat/apache-tomcat-9.0.36-002
- vim run.sh

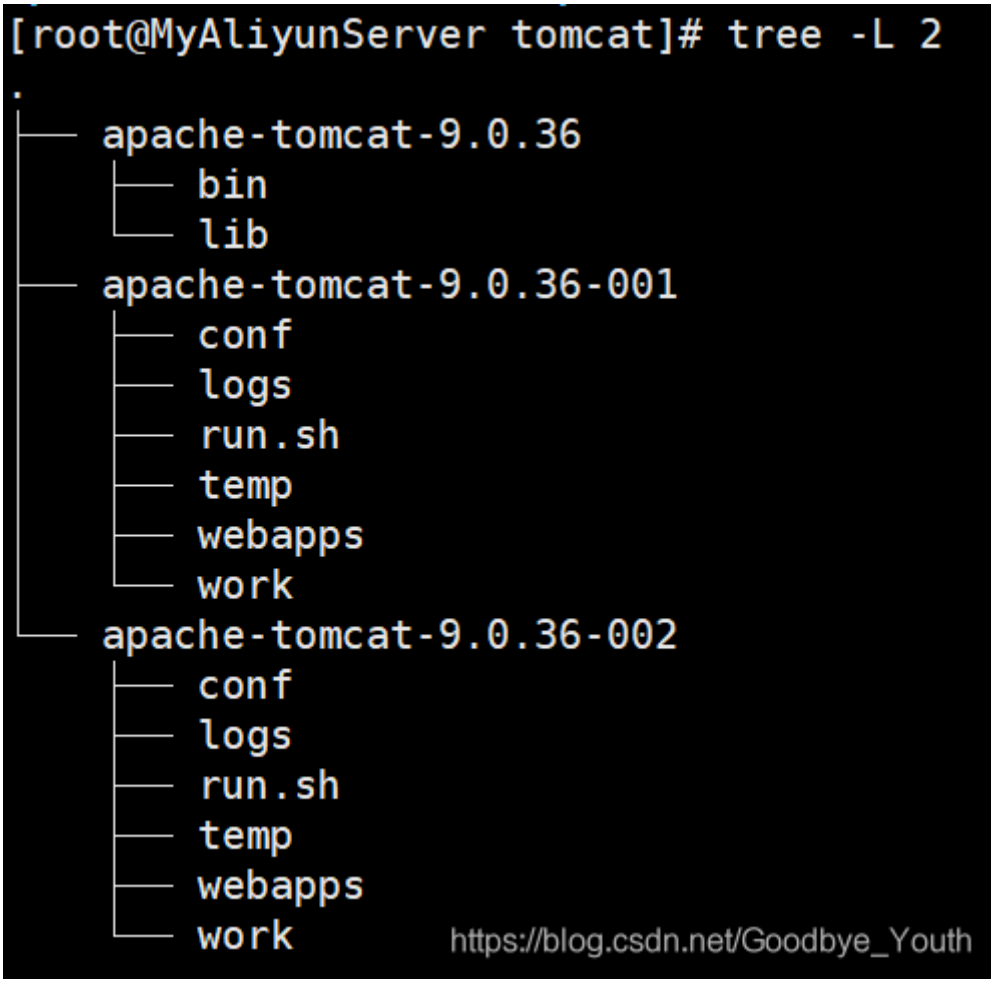
```
#!/bin/bash

export CATALINA_BASE=$CATALINA_BASE1
echo $CATALINA_BASE

#使用说明，用来提示输入参数
usage() {
    echo "Usage: startup.sh {start|stop|restart|status}"
    exit 1
}

#检查程序是否正在运行
is_running() {
    pid=`ps -ef|grep Dcatalina.base=${CATALINA_BASE}|grep -v
grep|awk '{print $2}'`
    #如果不存在则返回1，存在则返回0
    if [ -z "${pid}" ]
```

```
    then
        return 1
    else
        return 0
    fi
}

#启动
start() {
    is_running
    if [ $? -eq 0 ]
    then
        echo "${CATALINA_BASE} is already running, pid=${pid}"
    else
        exec $CATALINA_HOME/bin/catalina.sh start
    fi
}

#停止
stop() {
    is_running
    if [ $? -eq 0 ]
    then
        kill -9 $pid
    else
        echo "${CATALINA_BASE} is not running"
    fi
}

#查看运行状态
status() {
    is_running
    if [ $? -eq 0 ]
    then
        echo "${CATALINA_BASE} is running, pid is ${pid}"
    else
        echo "${CATALINA_BASE} is not running"
    fi
}

#重启
restart() {
    stop
    sleep 5
    start
}

#根据输入的参数，选择对应的执行方法，不输入则执行使用说明
case "$1" in
    start)
        start
        ;;
```

```
    stop)
        stop
        ;;
    status)
        status
        ;;
    restart)
        restart
        ;;
    *)
        usage
        ;;
esac
```

- :wq
- chmod +x run.sh

此时tomcat目录结构：



6.tomcat远程debug

修改配置

修改 **startup.sh** 文件

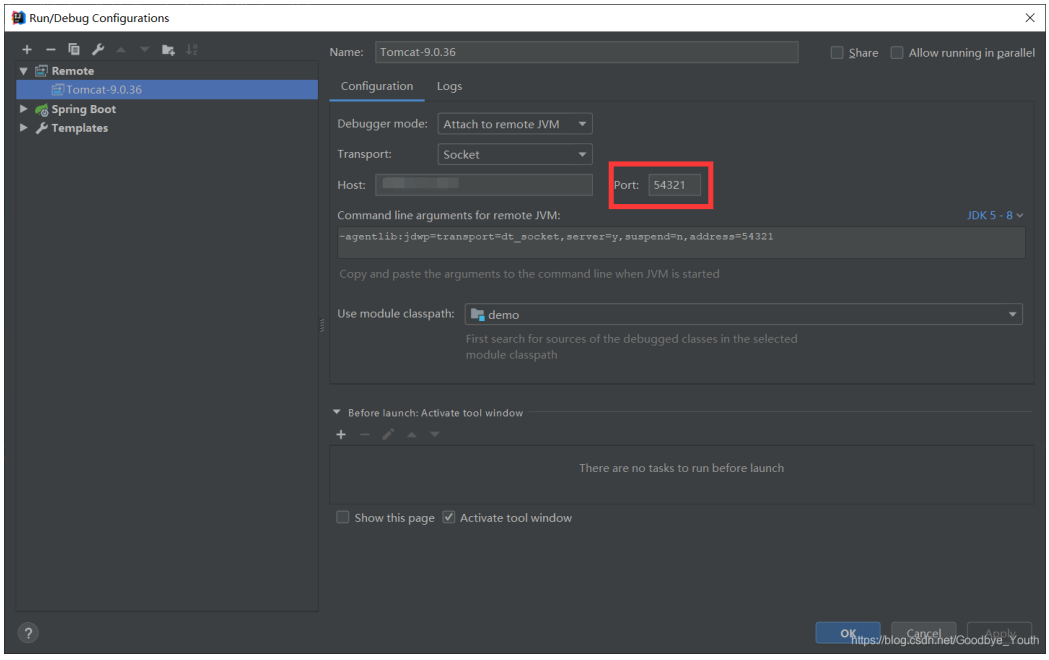
```
# Check that target executable exists
if $os400; then
  # -x will Only work on the os400 if the files are:
  # 1. owned by the user
  # 2. owned by the PRIMARY group of the user
  # this will not work if the user belongs in secondary groups
  eval
else
  if [ ! -x "$PRGDIR"/"$EXECUTABLE" ]; then
    echo "Cannot find $PRGDIR/$EXECUTABLE"
    echo "The file is absent or does not have execute permission"
    echo "This file is needed to run this program"
    exit 1
  fi
fi

exec "$PRGDIR"/"$EXECUTABLE" jpda start "$@"
```


修改 **catalina.sh** 文件(端口自定义)

```
if [ "$1" = "jpda" ] ; then
  if [ -z "$JPDA_TRANSPORT" ]; then
    JPDA_TRANSPORT="dt_socket"
  fi
  if [ -z "$JPDA_ADDRESS" ]; then
    JPDA_ADDRESS="54321"
  fi
  if [ -z "$JPDA_SUSPEND" ]; then
    JPDA_SUSPEND="n"
  fi
  if [ -z "$JPDA_OPTS" ]; then
    JPDA_OPTS="-agentlib:jdwp=transport=$JPDA_TRANSPORT,address=$JPDA_ADDRESS,server=y,suspend=$JPDA_SUSPEND"
  fi
  CATALINA_OPTS="$JPDA_OPTS $CATALINA_OPTS"
  shift
fi
```

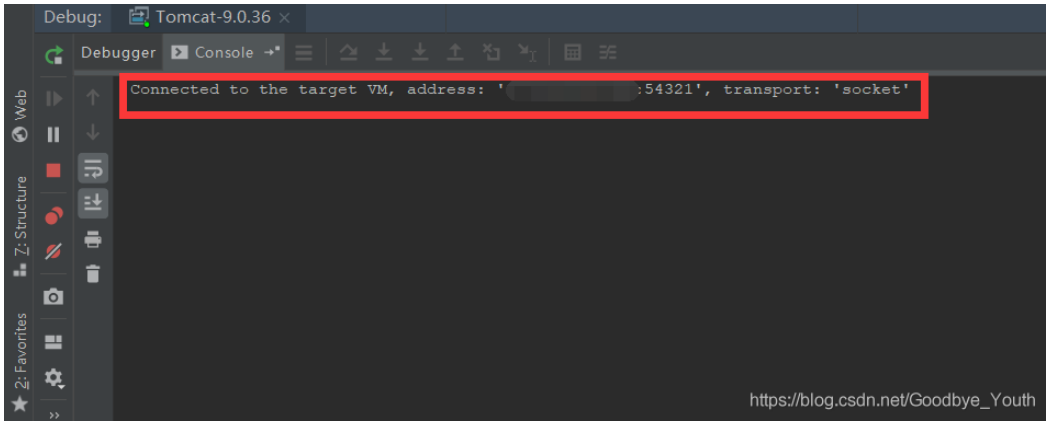

项目构建及部署

IDEA 配置远程调试



远程 Debug 测试

配置好之后，我们启动远程连接



在本地项目 debug() 方法中打上断点，通过浏览器调用部署在服务器上项目的 debug() 方法，即可进行远程调试