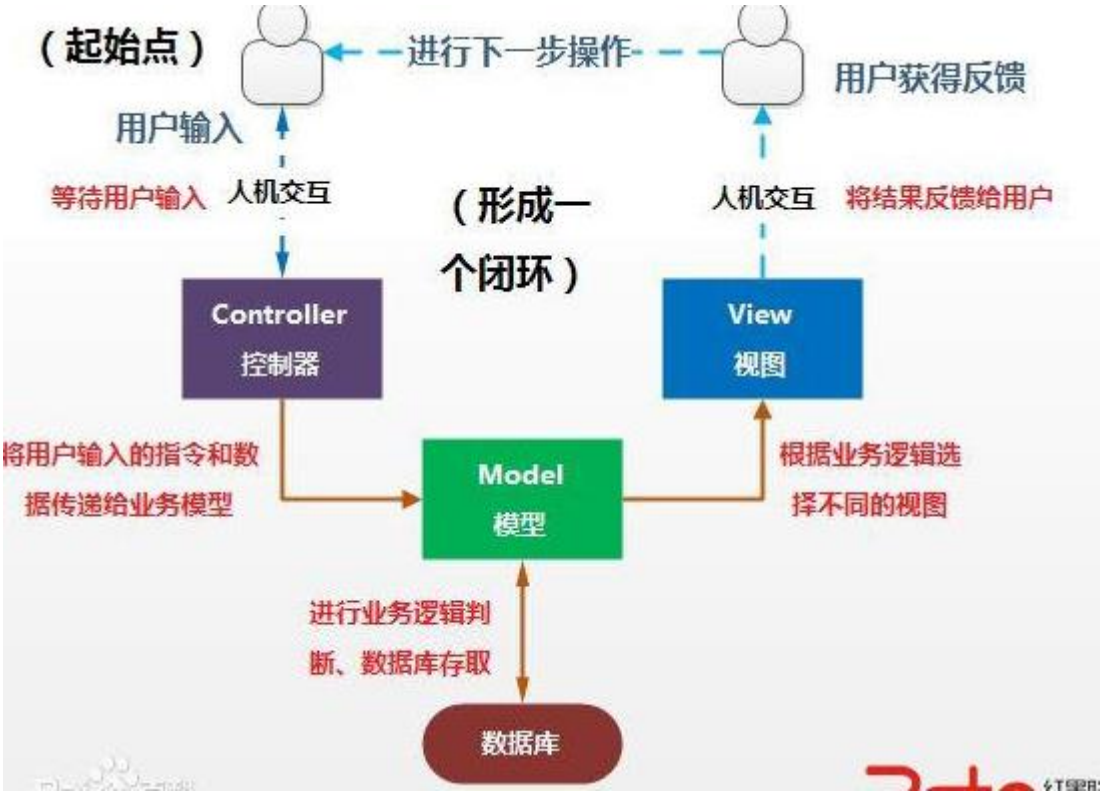
Lab11

- **以你的大作业为例，画出 web 层（或 安卓端） MVC 模型，并给出实现框架的部分关键代码或配置。**

  安卓端 MVC 模型:



  影院 view 部分代码:

```
package com.batty.ling.myapplication;



import android.app.AlertDialog;

import android.app.ListActivity;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.os.Bundle;
```

```
import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.BaseAdapter;

import java.util.HashMap;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import android.view.View.OnClickListener;

import android.view.View.OnLongClickListener;




public class MainActivity extends ListActivity {

    private List<Map<String, Object>> mData;

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        mData = getData();

        MyAdapter adapter;
```

```
        adapter= new MyAdapter(this);

        setListAdapter(adapter);

}



private List<Map<String, Object>> getData() {

        List<Map<String, Object>> list = new ArrayList<Map<String, Object>>();



        Map<String, Object> map = new HashMap<String, Object>();

        map.put("title", "Apple");

        map.put("img", R.drawable.apple);

        list.add(map);



        map = new HashMap<String, Object>();

        map.put("title", "Banana");

        map.put("img", R.drawable.banana);

        list.add(map);



        map = new HashMap<String, Object>();

        map.put("title", "Coco");

        map.put("img", R.drawable.coco);

        list.add(map);
```

```
map = new HashMap<String, Object>();

map.put("title", "Cherry");

map.put("img", R.drawable.cherry);

list.add(map);



map = new HashMap<String, Object>();

map.put("title", "Kiwi");

map.put("img", R.drawable.kiwi);

list.add(map);



map = new HashMap<String, Object>();

map.put("title", "Orange");

map.put("img", R.drawable.orange);

list.add(map);



map = new HashMap<String, Object>();

map.put("title", "Pear");

map.put("img", R.drawable.pear);

list.add(map);



map = new HashMap<String, Object>();

map.put("title", "Strawberry");
```

```
        map.put("img", R.drawable.strawberry);
        list.add(map);

        map = new HashMap<String, Object>();
        map.put("title", "Watermelon");
        map.put("img", R.drawable.watermelon);
        list.add(map);

        return list;
    }

    public void showInfo(){
        new AlertDialog.Builder(this)
                .setTitle("Sorry！")
                .setMessage("没有删除功能。。。。")
                .setPositiveButton("确定", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                    }
                })
                .show();
```

```
}

public final class ViewHolder{
    public ImageView img;
    public TextView title;
}

public class MyAdapter extends BaseAdapter{

    private LayoutInflater mInflater;

    public MyAdapter(Context context){
        this.mInflater = LayoutInflater.from(context);
    }
    @Override
    public int getCount() {
        // TODO Auto-generated method stub
        return mData.size();
    }
```

```
@Override

public Object getItem(int arg0) {

	// TODO Auto-generated method stub

	return null;

}



@Override

public long getItemId(int arg0) {

	// TODO Auto-generated method stub

	return 0;

}



@Override

public View getView(int position, View convertView, ViewGroup parent) {



	ViewHolder holder = null;

	if (convertView == null) {



		holder=new ViewHolder();



		convertView = mInflater.inflate(R.layout.activity_main, null);
```

```
        holder.img = (ImageView)convertView.findViewById(R.id.img);
        holder.title = (TextView)convertView.findViewById(R.id.title);
        convertView.setTag(holder);

    }else {

        holder = (ViewHolder)convertView.getTag();
    }


    holder.img.setBackgroundResource((Integer) mData.get(position).get("img"));
    holder.title.setText((String)mData.get(position).get("title"));

    final ViewHolder finalHolder = holder;
    final ViewHolder finalHolder1 = holder;
    holder.img.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String info;
            info = finalHolder1.title.getText().toString();
            Bundle bundle = new Bundle();
            //保存输入的信息
```

```
            bundle.putString("name", info);




            Intent intent = new Intent(MainActivity.this, NewActivity.class);

            intent.putExtras(bundle);

            finish();

            startActivity(intent);

        }

});



View.OnLongClickListener longClickListener = new OnLongClickListener() {

        @Override

        public boolean onLongClick(View v) {

            showInfo();

            return false;

        }

};

holder.img.setOnLongClickListener(longClickListener);
```

```
                return convertView;
            }
        }
}
```

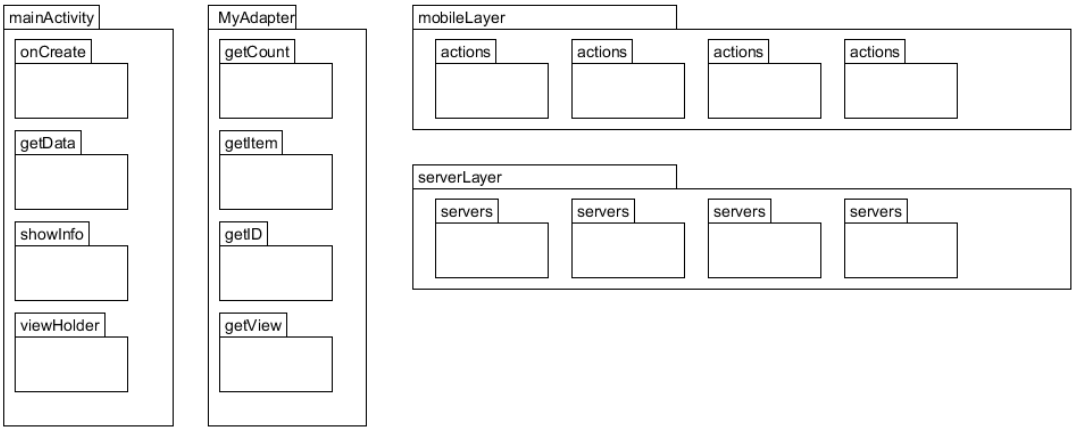
Controller 部分代码:

```
package com.batty.ling.myapplication;


import android.content.Intent;
import android.graphics.Color;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;


public class NewActivity extends AppCompatActivity {
    private Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new);
        btn = (Button) findViewById(R.id.button);
```

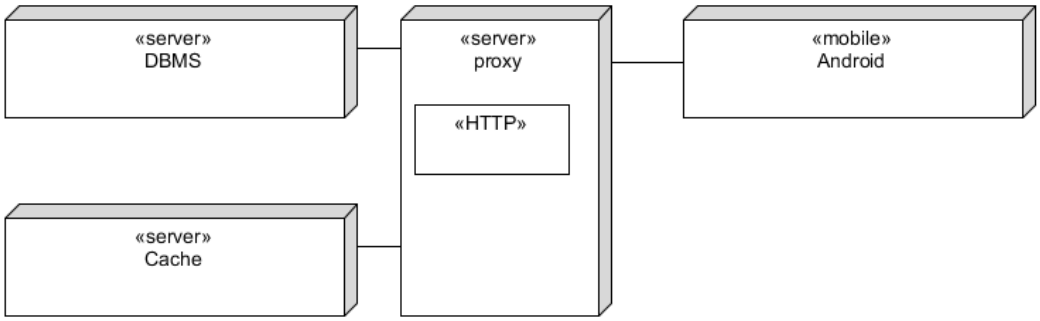
```
        View.OnClickListener listener = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(NewActivity.this,MainActivity.class);
                startActivity(intent);
            }
        };
        btn.setOnClickListener(listener);

        Bundle b=getIntent().getExtras();
        //获取 Bundle 的信息
        String info=b.getString("name");
        TextView textView;
        textView = (TextView)findViewById(R.id.textView);

        textView.setText("I love " + info + "!!!");
        textView.setTextColor(Color.rgb(255,0,0));
    }
}
```

- **使用 包图 和 部署图 描述你项目的架构。**

  包图:



部署图:



- **请使用 karaf 部署一个简单应用（见首页的教程 1）/ 或用 docker 在 ubuntu 镜像 基础上定制一个镜像（image）并部署一个简单 web 应用和服务。然后写一个学习笔记**

学习笔记:

一， 安装 karaf

安装

预安装需求

硬件：

l 20M 磁盘剩余空间。

操作系统：

l Windows：Windowsvista，Windows XP sp2，windows2000。

环境：

l Java SE 1.7 或者是更高。

l 环境变量 JAVA_HOME 必须设置为 java 运行时的安装目录，比如 C:\Program Files\Java\jdk1.7.0_51。按住 windows 和 break 键切换到高级选项，点击环境变量，把上面的路径加入到变量中。

从源码构建

如果你想从源码构建 karaf，需求会有点不同：

硬件：

l 200M 磁盘空间便于 apache karaf 源码展开或者是 SVN 的验证，以及 maven 构建和依赖 maven 组件的下载。

环境：

l JDK 1.7 或者是更高。

l Apache maven 3.2.1.

在 windows 上构建

这过程说明如何下载和安装 windows 上的源码文件。注：Karaf 需要 java7 编译，构建和运行。

1. 在浏览器输入 http://karaf.apache.org/index/community/download.html.

2. 滚动到“Apache Karaf”区域选择需要的链接。源码文件名称类似于：apache-karaf-3.0.0-src.zip.

3. 解压缩 zip 文件到你选择的目录中，请记住非法 java 路径的限制，比如!，%等。

4. 用 maven 3.2.1 或者更高的 java7 来构建 karaf。

a) Maven 搭建

i. 在浏览器输入 http://maven.apache.org/download.cgi

ii. 选择下载 apache-maven-3.2.1-bin.zip，解压缩 zip 文件到你选择的目录中

iii. 在 apache-maven-3.2.1-bin\apache-maven-3.2.1\conf 中选择，打开 settings.xml

找到<proxies>，在里面加入

```
<proxy>
      <id>optional</id>
      <active>true</active>
      <protocol>http</protocol>
       <host>proxy.fxis.co.jp</host>
      <port>8080</port>
   </proxy>
```

因为网络原因，我们选择使用 maven 的代理服务。

iv. 拷贝一份 settings.xml 到 C:\Users\Administrator\.m2 下面

构建 karaf 的方法如下：

Cd [karaf 安装路径]\src

Mvn

这两个步骤均需要 10 到 15 分钟。

5.用 zip 工具加压文件，windows 的路径是 [karaf 安装路径]\assembly\target\apache-karaf-

x.y.zip

6.转到开始 karaf 一节

Windows 安装过程

这里说明如何在 windows 系统上下载和安装二进制文件。

1.在浏览器输入 http://karaf.apache.org/index/community/download.html.

2. 滚动到“Apache Karaf”区域选择需要的链接。源码文件名称类似于：apache-karaf-3.0.0.zip.

3.解压缩 zip 文件到你选择的目录中，请记住非法 java 路径的限制，比如!，%等。

4. 转到开始 karaf 一节

5.可选：在 Windows 中启用彩色控制台输出

提示：如果你安装 karaf 到很深的路径或者是非法的 java 路径，!，%等，你可以创建一个 bat 文件来执行：subst S: "C:\your very % problematicpath!\KARAF"

这样 karaf 的路径是 s: - 这样的短类型。

安装后的步骤

在开始使用 karaf 之前强烈建议设置指向 JDK 的 JAVA_HOME 用来定位 java 可执行文件，并且应该配置为指向已安装 java se7 根目录。

二， 目录结构

Karaf 安装目录结构如下：

l /bin: 启动脚本

l /etc: 初始化文件

- /data: 工作目录
- /data /cache: OSGi 框架包缓存
- /data /generated-bundles: 部署使用的临时文件夹
- /data /log: 日志文件
- /deploy: 热部署目录
- /instances: 含有子实例的目录
- /lib: 包含引导库
- /lib/ext:JRE 扩展目录
- /lib/endorsed: 赞同库目录
- /system: OSGi 包库，作为一个 Maven2 存储库

Data 文件夹包括 karaf 所有的工作和临时文件，如果你想从一个初始状态重启，你可以清空这个目录，这和“恢复初始化设置”一样的效果。

三， 启动和停止 karaf

本章介绍如何启动和停止 Apache Karaf 和各种可用的选项。

启动 karaf

Windows 下

打开一个控制台窗口，更改到安装目录，并运行 Karaf。对于二进制文件，运行

Cd [karaf 安装目录]

然后输入：bin\karaf.bat

Linux 下

打开一个控制台窗口，更改到安装目录，并运行 Karaf。运行

Cd [karaf 安装目录]

然后输入：bin\karaf

警告：karaf 运行后不要关闭控制台，否则会终止 karaf（除非用非控制台启动 karaf）。

非控制台启动 karaf

没有控制台也可以启动 karaf，它总可以是用远程 SSH 访问，是用下面的命令：

bin\karaf.batserver

或者是 unix 下：bin/karafserver

在后台启动 karaf

采用以下命令可以轻易地在后台进程启动 karaf：

Bin\start.bat

或者在 unix 下：bin/start

重置模式启动 karaf

清空[karaf 安装目录]\data 文件夹就可以简单的在重置模式启动 karaf，为方便起见，在 karaf 启动脚本使用以下参数也可以实现重置启动：

On Windows:

bin\karaf.batclean

bin\start.batclean

停止 karaf

在 windows 你可以在 karaf 控制台采用以下命令来停止它：

karaf@root()> shutdown
Shutdown 命令会询问你是否真的想要停止，如果你确认停止并且拒绝确认信息，你可以用-f 或者—force 选项：

karaf@root()> shutdown -f
也可以用时间参数来延迟停止，时间参数有不同的形式。首先，可以是绝对时间各式 hh：mm。第二，也可以是+m，其中 m 是等待的分。现在就是+0.

下面的命令可以在 10:35am 关闭 karaf:

karaf@root()> system:shutdown 10:35
下面的命令在 10 分钟后关闭 karaf：

karaf@root()> system:shutdown +10
如果你在主控制台运行，用注销或者 Ctrl+D 退出控制台也可以终止 karaf 实例。

在控制台你可以运行如下命令：

bin\stop.bat

四， 使用控制台

查看可用的命令

按提示键 tab 可以在控制台看到可用的命令列表：

root@root><tab>Display all 182 possibilities? (y or n)

*:help addurl admin:change-opts

admin:change-rmi-registry-port admin:change-ssh-port admin:connect

admin:create admin:destroy admin:list

admin:rename admin:start admin:stop

bundle-level cancel cat

change-opts change-rmi-registry-port change-ssh-port

clear commandlist config:cancel

config:edit config:list config:propappend

config:propdel config:proplist config:propset

config:update connect create

create-dump destroy dev:create-dump

dev:dynamic-import dev:framework dev:print-stack-traces

dev:restart dev:show-tree dev:watch

display display-exception dynamic-import

each echo edit

exec exports features:addurl

features:info features:install features:list

features:listrepositories features:listurl features:listversions

features:refreshurl features:removerepository features:removeurl

features:uninstall framework get

grep head headers

help history if

imports info install

jaas:cancel jaas:commandlist jaas:list

jaas:manage jaas:roleadd jaas:roledel

jaas:update jaas:useradd jaas:userdel

jaas:userlist java list

listrepositories	listurl	listversions
log:clear	log:display	log:display-exception
log:get	log:set	log:tail
logout	ls	manage
more	new	osgi:bundle-level
osgi:headers	osgi:info	osgi:install
osgi:list	osgi:ls	osgi:refresh
osgi:resolve	osgi:restart	osgi:shutdown
osgi:start	osgi:start-level	osgi:stop
osgi:uninstall	osgi:update	packages:exports
packages:imports	print-stack-traces	printf
propappend	propdel	proplist
propset	refresh	refreshurl
removerepository	removeurl	rename
resolve	restart	roleadd
roledel	set	shell:cat
shell:clear	shell:each	shell:echo
shell:exec	shell:grep	shell:head
shell:history	shell:if	shell:info
shell:java	shell:logout	shell:more
shell:new	shell:printf	shell:sleep
shell:sort	shell:tac	shell:tail

show-tree shutdown sleep

sort ssh ssh:ssh

ssh:sshd sshd start

start-level stop tac

tail uninstall update

useradd userdel userlist

watch

root@root>

获得命令的帮助

要查看一个特定的命令的帮助，在命令后加--help 或使用 help 命令加上命令的名称：

karaf@root>features:list --help

描述

features:list

列出库中定义的所有功能。

语法

features:list [options]

选项

--help

显示此帮助信息

-i, --installed

只列出已安装的功能列表

更多…

所有可用命令列表和它们的用法，也可以在一个专门的章节。

你在开发指南会获得更多的 shell 语法的深入引导。

如开发指南解释的那样，控制台可以很容易的被新命令扩展。

五， 网络控制台

Karaf Web 控制台提供了一个运行时的图形概述。

你可以是用它来：

l •安装和卸载功能

l •启动，停止，安装捆绑

l •创建子实例

l •配置 Karaf

l •查看日志信息

l

安装 web 控制台

默认情况下 web 控制台是不安装的，安装请在 karaf 提示中运行以下命令：

karaf@root()> features:install webconsole

访问 Web 控制台

访问本地运行的一个 karaf 实例，在浏览器输入：

http://localhost:8181/system/console

使用 karaf 用户名和 karaf 密码来登录系统，如果你修改过用户名或密码，请是用修改后的。

改变 web 控制台端口号

默认情况下，控制台在 8181 端口运行，你可以通过创建属性文件 etc/org.ops4j.pax.web.cfg 并在后面添加如下属性设置（改成任意你想要的端口号）来改变端口：

org.osgi.service.http.port=8181

六， 远程控制台

使用远程实例

初始化远程实例

用它本地控制台管理 karaf 实例不总是有意义的，你可以用远程控制台远程管理 karaf。

当你启动 karaf 时，它使任何其他 Karaf 控制台或纯 SSH 客户端可以通过 SSH 访问远程控制台。远程控制台提供本地控制台的所有功能，并通过运行它里面的容器和服务给远程用户完全控制。

SSH 主机名和端口号在配置文件[karaf 安装目录]/etc/org.apache.karaf.shell.cfg 用以下默认值配置：

sshPort=8101

sshHost=0.0.0.0

sshRealm=karaf

hostKey=${karaf.base}/etc/host.key

你可以用初始化命令或者编辑上面的文件更改这个配置，但是需要重启 ssh 控制台以便它是用新的参数。

```
# define helper functions

bundle-by-sn = { bm = new java.util.HashMap ;    each (bundles) { $bm put ($it symbolicName)$it } ; $bm get $1 }

bundle-id-by-sn = { b = (bundle-by-sn $1) ; if { $b } { $b bundleId } {-1 } }

# edit config

config:edit org.apache.karaf.shell

config:propset sshPort 8102

config:update

# force a restart

osgi:restart --force (bundle-id-by-sn org.apache.karaf.shell.ssh)
```

远程连接和断开

使用 ssh：ssh 命令

你可以使用 ssh：ssh 命令来连接远程 karaf 控制台。

```
karaf@root> ssh:ssh -l karaf -P karaf -p 8101 hostname
```

注意：默认的密码是 karaf，但是我们强烈建议个更改。在安全模块查看更多信息。

为了确定你已经连接到正确的 karaf 实例，输入 ssh：info 在 karaf 提示符。返回当前连接的实例的信息，如下所示。

```
Karaf

   Karaf home                    /local/apache-karaf-2.0.0

   Karaf base                   /local/apache-karaf-2.0.0

   OSGi Framework               org.eclipse.osgi -3.5.1.R35x_v20090827

JVM
```

Java Virtual Machine Java HotSpot(TM) Server VM version14.1-b02

...

使用 karaf 客户端

Karaf 允许你安全的连接到远程 karaf 实例而不必运行本地 karaf 实例。

例如，在同一台机器上快速连接在 server 模式下运行的 karaf 实例，在 karaf 安装目录运行以下命令：

bin/client

通常情况下，你需要提供主机名，端口，用户名和密码来连接到远程实例。如果你使用的客户端在一个较大的脚本，你可以附加控制台命令如下：

bin/client -a 8101-h hostname -u karaf -p karaf features:install wrapper

显示可用的客户端选项，输入：

> bin/client--help

Apache Karaf client

-a [port] specify the port to connect to

-h [host] specify the host to connect to

-u [user] specify the user name

-p [password] specify the password

--help shows this help message

-v raise verbosity

-r [attempts] retry connection establishment(up to attempts times)

-d [delay] intra-retry delay (defaults to 2 seconds)

[commands] commands to run

如果没有指定的命令，客户端将在互动模式。

使用纯 SSH 客户端

你也可以使用你的 unix 系统中的纯 SSH 客户端或者 windows SSH 客户端像 putty 来连接。

~$ ssh -p 8101karaf@localhost

karaf@localhost'spassword:

从远程控制台断开

按 Ctrl+D，shell：logout 或者简单的在 karaf 提示符输入 logout 就可以断开远程控制台。

关闭远程实例

使用远程控制台

如果你已经用 ssh:ssh 命令或者 karaf 客户端连接到远程控制台，你可以用使用 osgi:shutdown 命令来停止远程实例。

注意：在远程控制台按 Ctrl + D 键，简单地关闭远程连接并返回到本地 shell。

使用 karaf 客户端

使用 karaf 客户端停止远程实例，在 karaf 安装目录/lib 目录运行以下命令：

bin/client -u karaf -p karaf -a 8101 hostname osgi:shutdown

七， 子实例

管理子实例

一个 Karaf 的子实例是一个副本，你可以分别启动和部署应用程序。实例不包含的完整副本 Karaf，但只有一个配置文件和数据文件夹的副本，其中包含了所有运行中的信息，日志文件和临时文件。

使用管理控制台命令

管理控制台命令允许您在同一台机器创建和管理 Karaf 实例。每一个新的运行时是运行时创建的子实例。您可以轻松地使用的名称管理子实例，而不是网络地址。有关管理命令的详细信息，请参阅管理命令。

创建子实例

你可以在 karaf 控制台输入 instance:create 创建新的运行时实例.

如下例子所示，instance:create 将使运行时在活动的运行时{实例/名称}目录创建新的运行时安装。新的实例是一个新的 karaf 实例并且分配一个 SSH 端口号基于始于 8101 的增量和一个 RMI 注册端口号基于始于 1099 的增量。

karaf@root>instance:createtest

Creating newinstance on SSH port 8102 and registry port 1100 / RMI server port

44445 at:C:\karaf\instances\test

Creating dir:C:\karaf\instances\test\bin

Creating dir:C:\karaf\instances\test\etc

Creating dir:C:\karaf\instances\test\system

Creating dir:C:\karaf\instances\test\deploy

Creating dir:C:\karaf\instances\test\data

Creating file:C:\karaf\instances\test\etc\all.policy

Creating file:C:\karaf\instances\test\etc\config.properties

Creating file:C:\karaf\instances\test\etc\custom.properties

Creating file:C:\karaf\instances\test\etc\distribution.info

Creating file:C:\karaf\instances\test\etc\equinox-debug.properties

Creating file:C:\karaf\instances\test\etc\java.util.logging.properties

Creating file:C:\karaf\instances\test\etc\jmx.acl.cfg

Creating file:C:\karaf\instances\test\etc\jmx.acl.java.lang.Memory.cfg

Creating file:C:\karaf\instances\test\etc\jmx.acl.org.apache.karaf.bundle.cfg

Creating file:C:\karaf\instances\test\etc\jmx.acl.org.apache.karaf.config.cfg

Creating file:C:\karaf\instances\test\etc\jmx.acl.org.apache.karaf.security.jmx.cfg

Creating file:C:\karaf\instances\test\etc\jmx.acl.osgi.compendium.cm.cfg

Creating file:C:\karaf\instances\test\etc\jre.properties

Creating file:C:\karaf\instances\test\etc\keys.properties

Creating file:C:\karaf\instances\test\etc\org.apache.felix.fileinstall-deploy.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.command.acl.bundle.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.command.acl.config.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.command.acl.feature.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.command.acl.jaas.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.command.acl.kar.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.command.acl.shell.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.command.acl.system.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.features.obr.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.features.repos.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.jaas.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.kar.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.log.cfg

Creating file:C:\karaf\instances\test\etc\org.ops4j.pax.logging.cfg

Creating file:C:\karaf\instances\test\etc\org.ops4j.pax.url.mvn.cfg

Creating file:C:\karaf\instances\test\etc\regions-config.xml

Creating file:C:\karaf\instances\test\etc\shell.init.script

Creating file:C:\karaf\instances\test\etc\users.properties

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.features.cfg

Creating file:C:\karaf\instances\test\etc\system.properties

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.shell.cfg

Creating file:C:\karaf\instances\test\etc\org.apache.karaf.management.cfg

Creating file:C:\karaf\instances\test\bin\karaf.bat

Creating file:C:\karaf\instances\test\bin\start.bat

Creating file:C:\karaf\instances\test\bin\stop.bat

karaf@root>

克隆一个实例

代替创建一个新的实例，你可以用: instance:clone 已经存在的实例来克隆

karaf@root()> instance:clone root test
移动实例的位置

默认情况下，新的实例会在 KARAF_HOME/instance 的下面，你可以通过命令选择新建或者克隆的实例位置。

karaf@root()> instance:create -v -l /tmp/test test
注 instance:destroy 自动移除实例，不用手动操作

改变子实例端口号

你可以使用 instance:ssh-port-change 命令来改变分配给子实例的 SSH 端口号。命令语法是：

instance:ssh-port-change 实例 端口号

karaf@root()> instance:ssh-port-change test 8104
需要注意的必须停止子实例才能运行此命令。

同样，你可以使用 instance:rmi-registry-port-change 命令改变分配给子实例的 RMI 注册端口号。命令的语法是：

karaf@root()> instance:rmi-registry-port-change test 1102
需要注意的必须停止子实例才能运行此命令。

同样，你也可以用 instance:rmi-server-port-change 改变 RMI Server 注册端口号

karaf@root()> instance:rmi-server-port-change test 44447
启动子实例

新的子实例在停止状态下被创建，用 instance:start 命令来启动子实例并使之准备主机应用。这个命令需要一个标识你想启动的子实例的 instance-name 参数。

karaf@root()> instance:start test
列出所有容器实例

要查看一个特定的安装下运行的所有 Karaf 实例的列表，使用 instance:list 命令。

karaf@root>instance:list
SSH Port | RMIRegistry | RMI Server | State |PID | Name

8101 | 1099 | 44444 | Started | 3616 | root

连接到子实例

你可以使用 instance:connect 命令连接到开始的子实例远程控制台，这需要三个参数：

karaf@root()> instance:connect test
一旦你连接到子实例，karaf 提示符显示现在实例的名字，如下：

karaf@test()>

你可以使用 karaf@test()> logout 推出实例
停止一个子实例

在实例自己内部停止一个子实例，输入 shutdown。

远程停止子实例，换句话说，从父或者兄弟实例，使用 instance:stop ：

karaf@root()> instance:stop test
注销一个子实例
你可以使用 instance:destroy 命令永久的删除一个停止的子实例：

karaf@root()> instance:destroy test
请注意只有停止的实例可以被注销。

实例重命名

你可以使用 instance:rename 命令为一个停止的子实例重新命名：

karaf@root()> instance:rename test newTest

八， 安全

管理用户名和密码

默认安全配置使用一个位于 karaf 安装目录/etc/users.properties 属性文件存储授权的用户和他们的密码。

默认的用户名是 karaf，与之相关联的密码也是 karaf。我们强烈建议在将 karaf 转移到产品前通过编辑上面的文件修改默认密码。

在 karaf 中用户现在被使用在三个地方：

l 访问 SSH 控制台

l 访问 JMX 管理层

l 访问 web 控制台

这三种方式的全部委托基于安全认证的相同的 JAAS。

users.properties 文件包含一或者多行，每行都定义了一个用户，密码和相关的角色。

USER=PASSWORD,_g_:GROUP,...

管理角色

JAAS 角色可以被各种组件使用。三个管理层（SSH,JMX 和 web 控制台）都使用用基于认证系统的全局角色。默认的角色名称在 etc/system.properties 中使用 karaf.local.roles= admin 系统属性配置，并且默认值是 admin。对管理层进行身份验证的所有用户必须有这个角色的定义。

这个值的语法如下：

[classname:]principal

其中 classname 是主要对象的类名（默认以 org.apache.karaf.jaas.modules.RolePrincipal），主要是这一类（默认为 admin）的主要的名称。

注意，可以使用以下配置 ConfigAdmin 对于一个给定的层改变角色：

Layer

PID

Value

SSH

org.apache.karaf.shell

sshRole

JMX

org.apache.karaf.management

jmxRole

Web

org.apache.karaf.webconsole

role

启用密码加密

为了避免密码是纯文本，密码可以加密存储在配置文件中。

这可以通过以下命令轻易的实现：

edit config

config:edit org.apache.karaf.jaas

config:propset encryption.enabled true

config:update

force a restart

dev:restart

用户第一次登录，密码将在 etc/users.properties 配置文件中被自动的加密。加密密码在前面加上{CRYPT}，因此很容易识别。

命令行参数执行

列举现在的用户模块 aas:realm-lis

karaf@root()> jaas:realm-list

可以通过 index 管理用户模块 jaas:realm-manage

karaf@root()> jaas:realm-manage --index 1

等等如下

jaas:user-list karaf@root()> jaas:user-list
jaas:user-add karaf@root()> jaas:user-add foo bar
jaas:user-delete karaf@root()> jaas:user-delete foo
jaas:group-add karaf@root()> jaas:group-add karaf mygroup
jaas:group-delete karaf@root()> jaas:group-delete karaf mygroup
jaas:group-role-add karaf@root()> jaas:group-role-add mygroup myrole
jaas:group-role-delete karaf@root()> jaas:group-role-delete mygroup myrole
jaas:update

jaas:cancel

管理领域

更多关于更改默认领域或者部署新领域信息将会在开发者指南中被提供。

部署安全供应商

有些应用程序需要特定的安全性提供者可用，如 BouncyCastle。JVM 在这些 jar 包的使用上施加一些限制：他们必须签署和引导类路径上可用。部署这些供应商的方法之一是把他们放在位于$ JAVA_HOME/ JRE/lib / ext 的 JRE 文件夹中并且修改安全策略配置（$JAVA_HOME/jre/lib/security/java.security）以登记等供应商。

虽然这种方法工作的很好，他将会有全局的影响并且需要你配置所有相应的服务器。

Karaf 提供了一个简单的方式来配置额外的安全性提供者：

l 把你的供应商 jar 放在 karaf-install-dir/lib/ext 中

l 修改初始化文件 karaf-install-dir/etc/config.properties ，添加如下属性：

org.apache.karaf.security.providers = xxx,yyy

这个属性的值是一个逗号分隔的提供商类名的注册名单。

例如：org.apache.karaf.security.providers = org.bouncycastle.jce.provider.BouncyCastleProvider
此外，你可能想向系统中捆绑的供应商的类提供访问，使所有束可以访问那些。它可以通过在相同的配置文件中修改 org.osgi.framework.bootdelegation 属性来实现：

org.osgi.framework.bootdelegation = ...,org.bouncycastle*

九， 日志系统

Karaf 提供先进的日志管理系统。

the OSGi Log Service
the Apache Log4j framework
the Apache Commons Logging framework
the Logback framework
the SLF4J framework
the native Java Util Logging framework
就是说应用程序使用任何的日志框架，karaf 都可以管理这些日志

在 etc/org.ops4j.pax.logging.cfg 你会发现不同的 Log4j 的元素

loggers
appenders

layouts
你也可以加入自己的构造信息。

命令行

Log:clear

清除日志

log:clear

Log:display

显示日志信息

karaf@root()> log:display
Log:exception-display

显示最近发生的异常信息

Log:get

显示日志级别

karaf@root()> log:get
Log:log

插入日志信息

karaf@root()> log:log "Hello World"
log：set

设置日志等级

karaf@root()> log:set DEBUG
log:tail

与 log:display 一样，但是是连续的显示日志

karaf@root()> log:tail