普元 PAAS 平台服务扩展开发手册

第 201507 版

普元软件

二〇一五年七月五日

版本控制信息

版本	日期	拟稿和修改	说明
1.0.0	2015-07-04	李中文	

文档安全控制信息

机密等级	☐国密☐商密一级☐商密二级☐商密三级☒内部使用☐公开		
机密性期限	永久		
授权访问/分发范围	普元信息技术股份有限公司		
备注			
文档操作方式			
复制	☒禁止	**打印**	☐禁止
复印	☐禁止		

目录

1　文档说明

1.1 编写目的

给第三方人员扩展开发 PAAS 平台做参考。扩展一个服务需要定义服务模型（服务和集群模型），服务管理和集群管理实现扩展；服务页面开发；服务脚本开发；服务打包；配置菜单和权限等。本文举例如何扩展实现 Redis 服务。

1.2 参考文档

无。

2　服务模型扩展

2.1 扩展接口说明

2.1.1 服务接口

com.primeton.paas.manage.api.model.IService

抽象实现类为：（一般扩展的服务，继承这个抽象实现即可）

com.primeton.paas.manage.api.model.AbstractService

2.1.2 集群接口

com.primeton.paas.manage.api.model.ICluster

抽象实现类为：（一般扩展的服务，继承这个抽象实现即可）

com.primeton.paas.manage.api.model.AbstractCluster

2.2 Redis 服务扩展示例

2.2.1 RedisService

```
/**
 *
 */
package com.primeton.paas.manage.api.service;
```

```
import com.primeton.paas.manage.api.model.AbstractService;
import com.primeton.paas.manage.api.model.IService;

/**
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public class RedisService extends AbstractService {

    /**
     * serialVersionUID
     */
    private static final long serialVersionUID = -
6767512771943979795L;

    public static final String TYPE = "Redis";

    public static final String RUN_AS_MASTER = "master";
    public static final String RUN_AS_SLAVE = "slave";

    private static final String RUN_MODE = "runMode";

    /**
     * Default. <br>
     */
    public RedisService() {
        super();
        setType(TYPE);
        setMode(IService.MODE_PHYSICAL);
    }

    /**
     *
     * @return
     */
    public String getRunMode() {
        return getValue(RUN_MODE, RUN_AS_MASTER);
    }

    /**
     *
     * @param runMode
     */
    public void setRunMode(String runMode) {
        if (RUN_AS_MASTER.equalsIgnoreCase(runMode)) {
            setValue(RUN_MODE, RUN_AS_MASTER);
        } else if (RUN_AS_SLAVE.equalsIgnoreCase(runMode)) {
            setValue(RUN_MODE, RUN_AS_SLAVE);
        }
    }

}
```

说明：PAAS 中的服务分为物理服务和逻辑服务，举例 MySQL 进程可以看作物理服务，MySQL 服务的一个 Schema 可以看作是逻辑服务，区分物理服务和逻辑服务：物理服务一般有进程存在，开服务端口；而逻辑服务一般没有。要为扩展的服务定义一个类型（不能与系统的冲突），在构造方法中调用父类 setType 方法来设置。根据不同服务的需要定义扩展属性，遵照 JavaBean 规范，但与 JavaBean 不同，参考上面示例，定义一些属性的 KEY 值，getter 和 setter 中调用父类的 getValue 和 setValue 来实现。在 META-INF/services 下配置 ServiceLoader，新建或编辑 com.primeton.paas.manage.api.model.IService 文件，添加扩展的 RedisService 实现类。

2.2.2 RedisCluster

```
/**
 *
 */
package com.primeton.paas.manage.api.cluster;

import com.primeton.paas.manage.api.model.AbstractCluster;
import com.primeton.paas.manage.api.service.RedisService;
```

```
/**
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public class RedisCluster extends AbstractCluster {

    /**
     * serialVersionUID
     */
    private static final long serialVersionUID =
5498253348014201718L;

    public static final String TYPE = RedisService.TYPE;

    /**
     * Default. <br>
     */
    public RedisCluster() {
        super();
        setType(TYPE);
        setName(TYPE + "-Cluster"); //$NON-NLS-1$
    }

}
```

说明：集群模型定义与服务模型定义类似，参考上述代码。在 META-INF/services 下配置 ServiceLoader,新建或编辑 com.primeton.paas.manage.api.model.ICluster 文件，添加扩展的 RedisCluster 实现类。

3 服务管理器扩展

3.1 扩展接口说明

3.1.1 服务管理器接口

com.primeton.paas.manage.api.manager.IServiceManager

默认实现类：（扩展的实现类一般继承默认实现，覆盖某些方法）

com.primeton.paas.manage.api.impl.manager.DefaultServiceManager

获取实现类方式：

IRedisServiceManager manager = com.primeton.paas.manage.api.factory.ServiceManagerFactory.getManager(RedisService.TYPE);

3.1.2 集群管理器接口

com.primeton.paas.manage.api.manager.IClusterManager

默认实现类：（扩展的实现类一般继承默认实现，覆盖某些方法）

com.primeton.paas.manage.api.impl.manager.DefaultClusterManager

获取实现类方式：

IClusterManager manager = com.primeton.paas.manage.api.factory.ClusterManagerFactory.getManager(RedisService.TYPE);

3.2 Redis 服务管理器扩展示例

3.2.1 IRedisServiceManager

如果扩展的服务有定义特殊接口的需要（IServiceManager 不能满足需求）可以自定接口，但要求自定义的接口继承 IServiceManager，参考下面示例代码：

```
/**
 *
 */
```

```java
package com.primeton.paas.manage.api.manager;

import java.util.List;

import com.primeton.paas.manage.api.exception.ServiceException;
import com.primeton.paas.manage.api.service.RedisService;

/**
 * Redis 服务管理器. <br>
 *
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public interface IRedisServiceManager extends IServiceManager {

    /**
     * New service instance. <br>
     *
     * @param service instance
     * @param clusterId cluster
     * @param number 1 (master) + N (slave)
     * @return redis instances
     * @throws ServiceException
     */
    List<RedisService> add(RedisService service, String clusterId,
int number)
            throws ServiceException;

}
```

3.2.2 RedisServiceManager

```
package com.primeton.paas.manage.api.impl.manager;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.gocom.cloud.cesium.mqclient.api.Command;
import org.gocom.cloud.cesium.mqclient.api.ServiceCommandMessage;
import
org.gocom.cloud.cesium.mqclient.api.exception.MessageException;
import org.gocom.cloud.common.logger.api.ILogger;

import com.primeton.paas.manage.api.exception.HostException;
import com.primeton.paas.manage.api.exception.ServiceException;
import com.primeton.paas.manage.api.factory.ClusterManagerFactory;
import com.primeton.paas.manage.api.factory.ManageLoggerFactory;
import com.primeton.paas.manage.api.impl.util.PathUtil;
import com.primeton.paas.manage.api.impl.util.SendMessageUtil;
import com.primeton.paas.manage.api.impl.util.ServiceUtil;
import com.primeton.paas.manage.api.impl.util.SystemVariables;
import com.primeton.paas.manage.api.manager.IClusterManager;
import com.primeton.paas.manage.api.manager.IRedisServiceManager;
import com.primeton.paas.manage.api.model.IService;
import com.primeton.paas.manage.api.service.RedisService;
import com.primeton.paas.manage.api.util.StringUtil;

/**
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public class RedisServiceManager extends DefaultServiceManager
        implements IRedisServiceManager {

    public static final String TYPE = RedisService.TYPE;
    private static ILogger logger =
ManageLoggerFactory.getLogger(RedisServiceManager.class);

    /* (non-Javadoc)
     * @see
com.primeton.paas.manage.api.impl.manager.DefaultServiceManager#ge
tType()
     */
    public String getType() {
        return TYPE;
    }

    /*
     * (non-Javadoc)
     * @see
com.primeton.paas.manage.api.impl.manager.DefaultServiceManager#st
art(java.lang.String)
     */
    public void start(String id) throws ServiceException {
        if (StringUtil.isEmpty(id)) {
            return;
        }
        RedisService service = getServiceQuery().get(id,
RedisService.class);
        if (service == null) {
            String message ="Redis instance [" + id + "] not exists,
can not execute start action.";
            if (logger.isDebugEnabled()) {
                logger.debug(message);
            }
            return;
        }
        if
(RedisService.RUN_AS_MASTER.equalsIgnoreCase(service.getRunMode())
) {
            super.start(id);
            return;
        }
        IClusterManager manager =
ClusterManagerFactory.getManager(IClusterManager.DEFAULT_TYPE);
        String clusterId = manager.getClusterId(id);
        if (StringUtil.isEmpty(clusterId)) {
            super.start(id);
            return;
        }

        List<RedisService> services =
getServiceQuery().getByCluster(clusterId, RedisService.class);
        if (null == services || services.isEmpty() ||
services.size() == 1) {
            super.start(id);
            return;
        }
        // Find master
        RedisService master = null;
        for (RedisService instance : services) {
            if
(RedisService.RUN_AS_MASTER.equalsIgnoreCase(instance.getRunMode()
)) {
                master = instance;
                break;
            }
        }
        if (null == master) {
            super.start(id);
            return;
        }
        // start
        ServiceCommandMessage message = new
ServiceCommandMessage();
        message.setAction(ServiceCommandMessage.ACTION_START);
        message.setClusterName(clusterId);
        message.setIp(service.getIp());
        message.setPort(service.getPort());
        message.setSrvDefName(RedisService.TYPE);
        message.setSrvInstId(service.getId());
        message.setNeedResponse(true);

        Map<String, String> args = copy(service.getAttributes());
        args.put("masterIp", master.getIp()); //$NON-NLS-1$
        args.put("masterPort", String.valueOf(master.getPort()));
//$NON-NLS-1$

        Command body = new Command(args,
PathUtil.getScriptPath(RedisService.TYPE, SH_START));
        message.setBody(body);

        try {
            SendMessageUtil.sendMessage(message,
SystemVariables.getMaxWaitMessageTime());
        } catch (MessageException e) {
            throw new ServiceException(e);
        }
    }

    /**
     *
     * @param map
     * @return
     */
    private static Map<String, String> copy(Map<String, String>
map) {
        if (null == map) {
            return null;
        }
        Map<String, String> newMap = new HashMap<String, String>();
        for (String key : map.keySet()) {
            newMap.put(key, map.get(key));
        }
        return newMap;
    }

    /* (non-Javadoc)
     * @see
com.primeton.paas.manage.api.impl.manager.DefaultServiceManager#ad
d(com.primeton.paas.manage.api.model.IService, java.lang.String)
     */
    public <T extends IService> T add(T service, String clusterId)
            throws ServiceException {
        return create(service, clusterId);
    }

    /* (non-Javadoc)
     * @see
com.primeton.paas.manage.api.manager.IRedisServiceManager#add(com.
primeton.paas.manage.api.service.RedisService, java.lang.String,
int)
     */
    public List<RedisService> add(RedisService service, String
clusterId,
            int number) throws ServiceException {
        if (null == service || StringUtil.isEmpty(clusterId) ||
number < 1) {
            return new ArrayList<RedisService>();
        }
        service.setHaMode(IService.HA_MODE_CLUSTER);
        // 申请机器
        String[] ips = null;
        try {
            ips =
getHostAssignManager().apply(service.getPackageId(),
service.getType(), service.isStandalone(),
                    number,
getHostAssignManager().getInstallTimeout(getType()));
```

```
        } catch (Throwable t) {
            logger.error("Assign host resource error.", t);
        }
        if (ips == null || ips.length == 0) {
            logger.error("Assign host resource error, not enough 
host resource.");
            throw new ServiceException("Not enough host resource for 
apply. require [" + number + "]");
        }
        if (number == ips.length) {
            logger.info("Create redis service required machine [" + 
number + "] has applied for.");
        } else {
            try {
                getHostManager().release(ips);
            } catch (HostException e) {
                logger.error(e);
            }
            throw new ServiceException("Not enough host resource for 
apply, require [" + number + "], left [" + ips.length + "].");
        }

        List<RedisService> services = new 
ArrayList<RedisService>();
        for (int i=0; i<number; i++) {
            RedisService instance = ServiceUtil.copy(service);
            instance.setRunMode(i == 0? RedisService.RUN_AS_MASTER : 
RedisService.RUN_AS_SLAVE);
            instance.setIp(ips[i]); // 设置IP
            services.add(super.add(instance, clusterId)); // 保存至数
据库
        }
        return services;
    }


}
```

说明：必须覆盖 getType()方法，必须提供默认的无参数构造方法，返回值是 RedisService.TYPE；根据实际需要可以覆盖某个默认实现；在 META-INF/services 下配置 ServiceLoader,新建或编辑 com.primeton.paas.manage.api.manager.IServiceManager 文件，添加扩展的 RedisServiceManager 实现类。

3.2.3 RedisClusterManager （非必需）

```java
/**
 * Copyright © 2009 - 2015 Primeton. All Rights Reserved.
 */
package com.primeton.paas.manage.api.impl.manager;

import java.util.List;

import org.gocom.cloud.common.logger.api.ILogger;

import com.primeton.paas.manage.api.exception.ServiceException;
import com.primeton.paas.manage.api.factory.ManageLoggerFactory;
import com.primeton.paas.manage.api.factory.ServiceManagerFactory;
import com.primeton.paas.manage.api.manager.IRedisServiceManager;
import com.primeton.paas.manage.api.service.RedisService;
import com.primeton.paas.manage.api.util.StringUtil;

/**
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public class RedisClusterManager extends DefaultClusterManager {

   public static final String TYPE = RedisService.TYPE;

   private ILogger logger =
ManageLoggerFactory.getLogger(RedisClusterManager.class);
```

```
    private static IRedisServiceManager redisServiceManager = 
ServiceManagerFactory.getManager(RedisService.TYPE);

    /**
     * Default. <br>
     */
    public RedisClusterManager() {
        super();
    }

    /* (non-Javadoc)
     * @see 
com.primeton.paas.manage.api.impl.manager.DefaultClusterManager#ge
tType()
     */
    public String getType() {
        return TYPE;
    }

    /*
     * (non-Javadoc)
     * @see 
com.primeton.paas.manage.api.impl.manager.DefaultClusterManager#st
art(java.lang.String)
     */
    public void start(String id) throws ServiceException {
        if (StringUtil.isEmpty(id)) {
            return;
        }
        List<RedisService> services = 
getServiceQuery().getByCluster(id, RedisService.class);
        if (null == services || services.isEmpty()) {
            return;
        }
        StringBuffer errorMessage = new StringBuffer();
        boolean isError = false;
        for (RedisService service : services) {
            try {
                getRedisServiceManager().start(service.getId());
            } catch (Throwable t) {
                logger.error(t);
                isError = true;
                errorMessage.append("\nStart redis
[").append(service.getId()).append("]
error.\n").append(t.getMessage()).append("\n");
            }
        }
        if (isError) {
            throw new ServiceException(errorMessage.toString());
        }
    }

    /**
     *
     * @return
     */
    protected static IRedisServiceManager getRedisServiceManager()
{
        return redisServiceManager = (null == redisServiceManager)
                ? (IRedisServiceManager)
ServiceManagerFactory.getManager(RedisService.TYPE)
                : redisServiceManager;
    }

}
```

说明：必须覆盖 getType()方法，必须提供默认的无参数构造方法，返回值是 RedisService.TYPE；根据实际需要可以覆盖某个默认实现；在 META-INF/services 下配置 ServiceLoader,新建或编辑 com.primeton.paas.manage.api.manager.IClusterManager 文件，添加扩展的 RedisClusterManager 实现类。

4　服务脚本扩展

4.1　扩展脚本

需要扩展四个脚本:start.sh / stop.sh / install.sh / uninstall.sh。

4.2　代码结构

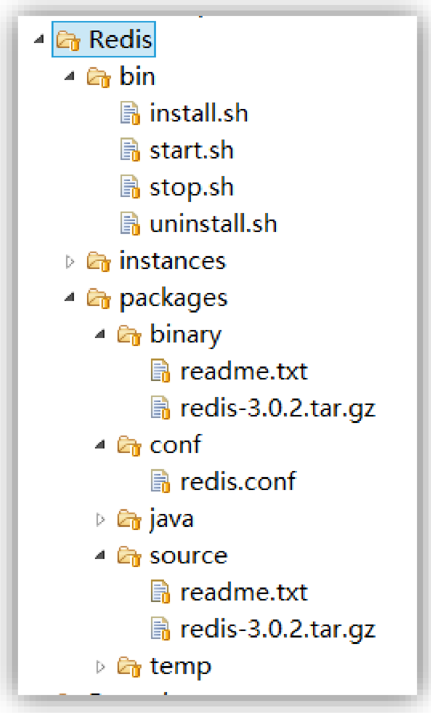
必须在头部引入 paas-env.sh，尾部引入 templates.sh;实现_dohelp(打印帮助信息)，_doparse(参数解析),_doexecute(main 方法，主要业务代码执行入口)三个方法;参考下面示例代码：

```
#!/bin/bash


#
# -----------------------------------------------------------------
# Copyright (c) 2009 - 2015 Primeton. All Rights Reserved.
#-----------------------------------------------------------------
#
# author ZhongWen.Li (mailto:lizw@primeton.com)
#


# import paas-env.sh
source $(cd $(dirname ${0})/../..; pwd)/Common/bin/paas-env.sh

# my variables
# Root directory for programs
program_home=${PROGRAME_HOME_PATH}/Redis
bin_home=${BIN_HOME_PATH}/Redis

# Help, print help information to terminal.
function _dohelp() {
    echo "Usage: ./uninstall.sh -reqId uuid"
    echo "-h) Print help"
    echo "-reqId) Request id"
}
```

```
# Parse execute arguments
function _doparse() {
   while [ -n "$1" ]; do
      case $1 in
      -arg0) arg0=$2;shift 2;;
      *) break;;
      esac
   done
}

# Write your core/main code
function _doexecute() {
   # print variables
   echo "program_home = ${program_home}"
   echo "bin_home = ${bin_home}"

   _return "[`date`] Begin uninstall Redis."

   # remove program
   _return "[`date`] Remove ${program_home}."
   if [ -d ${program_home} ]; then
      rm -rf ${program_home}
   fi

   # remove bin
   _return "[`date`] Remove ${bin_home}."
   if [ -d ${bin_home} ]; then
      rm -rf ${bin_home}
   fi

   _return "[`date`] End uninstall Redis."

   _success
   exit 0
}
```

```
# import template.sh
source ${BIN_HOME_PATH}/Common/bin/template.sh
```

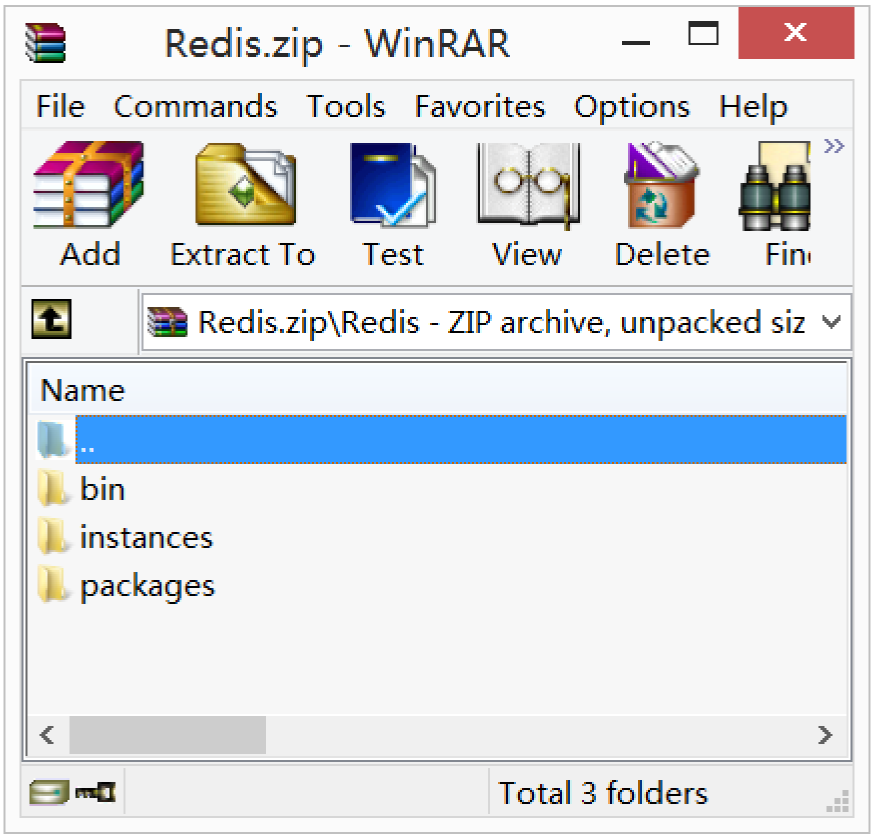
4.3 目录结构



```
Redis
  bin
    install.sh
    start.sh
    stop.sh
    uninstall.sh
  instances
  packages
    binary
      readme.txt
      redis-3.0.2.tar.gz
    conf
      redis.conf
    java
    source
      readme.txt
      redis-3.0.2.tar.gz
    temp
```

packages/source 下存放的是未进行编译的 redis 源码安装包；packages/binary 下存放的是已经编译后的 redis 二进制安装包。

该目录结构即为服务包的目录结构，最终开发完成的 Redis 服务需要打包成 Redis.zip，如下图所示：



4.4 上线部署

上传至 HTTP 软件仓库下，如下图所示：

```
suse:/primeton/paas/programs/Repository/jetty/webapps/default/services # l
total 444896
drwxrwxrwx  2 root root      4096 Jul  4 04:17 ./
drwxr-xr-x 10 root root      4096 Jun  7 21:36 ../
----------  1 root root   9018207 Jun  7 15:26 CEPEngine.zip
----------  1 root root   2983392 Jun  7 15:26 Collector.zip
----------  1 root root      4548 Jun  7 15:26 Common.zip
----------  1 root root   3433341 Jun  7 15:26 HaProxy.zip
----------  1 root root  15905250 Jun  7 15:26 Jetty.zip
----------  1 root root  10827767 Jun  7 15:26 Keepalived.zip
----------  1 root root   8119552 Jun  7 15:26 Mail.zip
----------  1 root root   6958515 Jun  7 15:26 Memcached.zip
----------  1 root root 202861324 Jun  7 15:26 MySQL.zip
----------  1 root root  27613698 Jun  7 15:26 Nginx.zip
----------  1 root root   9117618 Jun  7 15:26 OpenAPI.zip
-rw-r--r--  1 root root         1 Jul  4 04:16 Redis.zip
----------  1 root root 158183334 Jun  7 15:27 SVN.zip
----------  1 root root      6634 Jun  7 15:27 SVNRepository.zip
-rw-r--r--  1 root root         1 Jul  4 04:17 Tomcat.zip
```

5 服务初始化 SQL

5.1 服务定义

```
INSERT INTO `cld_service` (`SERVICE_NAME`, `SERVICE_DISPLAY_NAME`, `SERVICE_DESC`, `SERVICE_MODE`) VALUES ('Redis', 'Redis', 'Redis', 'PHYSICAL');
```

5.2 端口定义

```
INSERT INTO `cld_variable` (`VARIABLE_TYPE`, `VARIABLE_NAME`, `VARIABLE_VALUE`, `VARIABLE_DESC`) VALUES ('HOST', 'PORT_RANGE_Redis', '8200-8210', 'Redis 服务端口段');
```

6 订单处理器扩展

6.1 接口说明

订单管理器接口：（查询，新增，修改，删除，处理等方法，通过 OrderManagerFactory 获取实例）

com.primeton.paas.manage.api.manager.IOrderManager

订单处理器接口：（通过扩展该接口实现不同订单类型的服务开通/销毁/变更/配置等处理；一般实现类集成他的抽象实现类即可）

com.primeton.paas.manage.api.manager.IOrderProcessor

抽象实现类：

com.primeton.paas.manage.api.impl.manager.AbstractOrderProcessor

```
/**
 * Copyright © 2009 - 2015 Primeton. All Rights Reserved.
 */
package com.primeton.paas.manage.api.manager;
```

```
import com.primeton.paas.manage.api.exception.OrderException;
import com.primeton.paas.manage.api.model.Order;

/**
 * 订单处理器. <br>
 *
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public interface IOrderProcessor {

    /**
     * 处理器类型, 与订单类型映射. <br>
     *
     * @see com.primeton.paas.manage.api.model.Order#getOrderType()
     *
     * @return 处理器类型
     */
    String getType();

    /**
     * 处理订单. <br>
     *
     * @param order 订单
     * @throws OrderException
     */
    void process(Order order) throws OrderException;

    /**
     * 仅仅处理某个订单项. <br>
     *
     * @param order 订单
     * @param itemId 订单项标识
     * @throws OrderException
     */
    void process(Order order, String itemId) throws OrderException;
}
```

6.2 Redis 服务订单处理器示例

举例：处理创建 Redis 服务订单实现，参考下面代码：

```
/**
 * Copyright © 2009 - 2015 Primeton. All Rights Reserved.
 */
package com.primeton.paas.manage.api.impl.order;

import java.util.Date;
import java.util.List;

import org.gocom.cloud.common.logger.api.ILogger;

import com.primeton.paas.manage.api.cluster.RedisCluster;
import com.primeton.paas.manage.api.exception.ClusterException;
import com.primeton.paas.manage.api.exception.OrderException;
import com.primeton.paas.manage.api.exception.ServiceException;
import com.primeton.paas.manage.api.factory.ClusterManagerFactory;
import com.primeton.paas.manage.api.factory.ManageLoggerFactory;
import com.primeton.paas.manage.api.factory.ServiceManagerFactory;
import
com.primeton.paas.manage.api.impl.manager.AbstractOrderProcessor;
import com.primeton.paas.manage.api.impl.util.ServiceRemoveUtil;
import com.primeton.paas.manage.api.impl.util.SystemVariables;
import com.primeton.paas.manage.api.manager.IClusterManager;
import com.primeton.paas.manage.api.manager.IRedisServiceManager;
import com.primeton.paas.manage.api.model.ICluster;
import com.primeton.paas.manage.api.model.IService;
import com.primeton.paas.manage.api.model.Order;
import com.primeton.paas.manage.api.model.OrderItem;
import com.primeton.paas.manage.api.model.OrderItemAttr;
import com.primeton.paas.manage.api.service.RedisService;
import com.primeton.paas.manage.api.util.StringUtil;

/**
 * 订单处理器 : 创建简单的Redis服务集群及其服务实例. <br>
 *
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public class SingleCreateRedisOrderProcessor extends
AbstractOrderProcessor {

    private static ILogger logger =
ManageLoggerFactory.getLogger(SingleCreateRedisOrderProcessor.clas
s);

    /**
     * Type of Order. <br>
     */
    public static final String TYPE = "SingleCreateRedis";

    private static IClusterManager clusterManager =
ClusterManagerFactory.getManager(RedisService.TYPE);
    private static IRedisServiceManager serviceManager =
ServiceManagerFactory.getManager(RedisService.TYPE);

    /* (non-Javadoc)
     * @see
com.primeton.paas.manage.api.manager.IOrderProcessor#getType()
     */
    public String getType() {
        return TYPE;
    }

    /* (non-Javadoc)
```

```
     * @see
com.primeton.paas.manage.api.manager.IOrderProcessor#process(com.p
rimeton.paas.manage.api.model.Order)
     */
    public void process(Order order) throws OrderException {
        if (null == order) {
            return;
        }
        OrderItem item = getItemByType(order, RedisService.TYPE);
        if (null == item) {
            List<OrderItem> items = order.getItemList();
            if (null == items || items.isEmpty()) {
                order =
getOrderManager().getOrder(order.getOrderId());
                items = order.getItemList();
            }
            item = (null == items || items.isEmpty()) ? null :
items.get(0);
        }
        if (null == item) {
            logger.warn("Order [" + order.getOrderType() + ", " +
order.getOrderId() + "] none redis item.");
            performOk(order);
            return;
        }
        item.setItemStatus(OrderItem.ITEM_STATUS_PROCESSING);
        item.setHandleTime(new Date());
        getOrderManager().updateItem(item);

        String displayName = getItemValue(item,
OrderItemAttr.ATTR_DISPLAY_NAME, "redis service"); //$NON-NLS-1$
        String packageId = getItemValue(item,
OrderItemAttr.ATTR_HOSTPKG_ID); //$NON-NLS-1$
        int clusterSize = getItemValue(item,
OrderItemAttr.ATTR_CLUSTER_SIZE, 1); //$NON-NLS-1$
        clusterSize = clusterSize < 1 ? 1 : clusterSize;
        String isStandalone = getItemValue(item,
OrderItemAttr.ATTR_IS_STANDALONE, "N"); //$NON-NLS-1$

        if (StringUtil.isEmpty(packageId)) {
            throw new OrderException("Create redis error, packageId
is empty. Please check you order information.");
        }

        RedisCluster cluster = new RedisCluster();
        cluster.setOwner(order.getOwner());
        cluster.setName(displayName);
        cluster.setMinSize(clusterSize);
        cluster.setMaxSize(clusterSize);
        cluster.setScope(ICluster.SCOPE_INNER);

        ICluster redisCluster = null;
        logger.info("Begin create redis cluster [" + displayName +
"].");
        try {
            redisCluster = getClusterManager().create(cluster);
        } catch (ClusterException e) {
            logger.error("Create redis cluster [" + displayName + "]
error", e);
            throw new OrderException(e);
        }

        String redisClusterId = redisCluster.getId();
        logger.info("End create redis cluster [" + redisClusterId +
"].");

        RedisService service = new RedisService();
        service.setCreatedDate(new Date());
        service.setCreatedBy(order.getHandler());
        service.setOwner(order.getOwner());
        service.setName(displayName);
        service.setPackageId(packageId);
        service.setScope(IService.SCOPE_INNER);
        service.setStandalone("Y".equalsIgnoreCase(isStandalone));
//$NON-NLS-1$
        service.setState(IService.STATE_NOT_RUNNING);

        List<RedisService> services = null;
        try {
            services = getServiceManager().add(service,
redisClusterId, clusterSize);
        } catch (ServiceException e) {
            if (SystemVariables.isCleanAfterHandleError()) {
                logger.info("Begin clean redis cluster [" +
redisClusterId + "].");
                ServiceRemoveUtil.removeRedisCluster(redisClusterId);
            }
            throw new OrderException("Create redis service for
cluster [" + redisClusterId +"] error.", e);
        }

        if (null == services || services.isEmpty()) {
            if (SystemVariables.isCleanAfterHandleError()) {
                logger.info("Begin remove redis cluster [" +
redisClusterId + "].");
                ServiceRemoveUtil.removeRedisCluster(redisClusterId);
            }
            throw new OrderException("Create redis service error ["
+ redisClusterId + "].");
        }

        logger.info("Begin start redis cluster [" + redisClusterId
+ "].");
        try {
            getClusterManager().start(redisClusterId);
        } catch (ServiceException e) {
            logger.error(e);
        }
        // 更新订单项状态
        item.setItemStatus(OrderItem.ITEM_STATUS_SUCCEED);
        item.setFinishTime(new Date());
        getOrderManager().updateItem(item);

        // 触发检查订单状态
        performOk(order);
    }

    /* (non-Javadoc)
     * @see
com.primeton.paas.manage.api.manager.IOrderProcessor#process(com.p
rimeton.paas.manage.api.model.Order, java.lang.String)
     */
    public void process(Order order, String itemId) throws
OrderException {
        if (null == order || StringUtil.isEmpty(itemId)) {
            return;
        }
        // only one item
        process(order);
    }

    /**
     *
     * @return IClusterManager of Redis
     */
    protected static IClusterManager getClusterManager() {
        return clusterManager = (null == clusterManager)
                ? ClusterManagerFactory.getManager(RedisService.TYPE)
                : clusterManager;
    }

    /**
     *
     * @return
     */
    protected static IRedisServiceManager getServiceManager() {
        return serviceManager = (null == serviceManager)
                ?
(IRedisServiceManager)ServiceManagerFactory.getManager(RedisServic
e.TYPE)
                : serviceManager;
    }

}
```

说明：集群模型定义与服务模型定义类似，参考上述代码。在 META-INF/services 下配置 ServiceLoader,新建或编辑 com.primeton.paas.manage.api.manager.IOrderProcessor 文件，添加扩展的订单处理器实现类。

7 门户页面开发

默认采用 Jsery （REST 风格）+ JSP + NUI 来开发页面，也可以使用其他技术，不做严格限制。

7.1 CONSOLE-APP

7.1.1 菜单定义

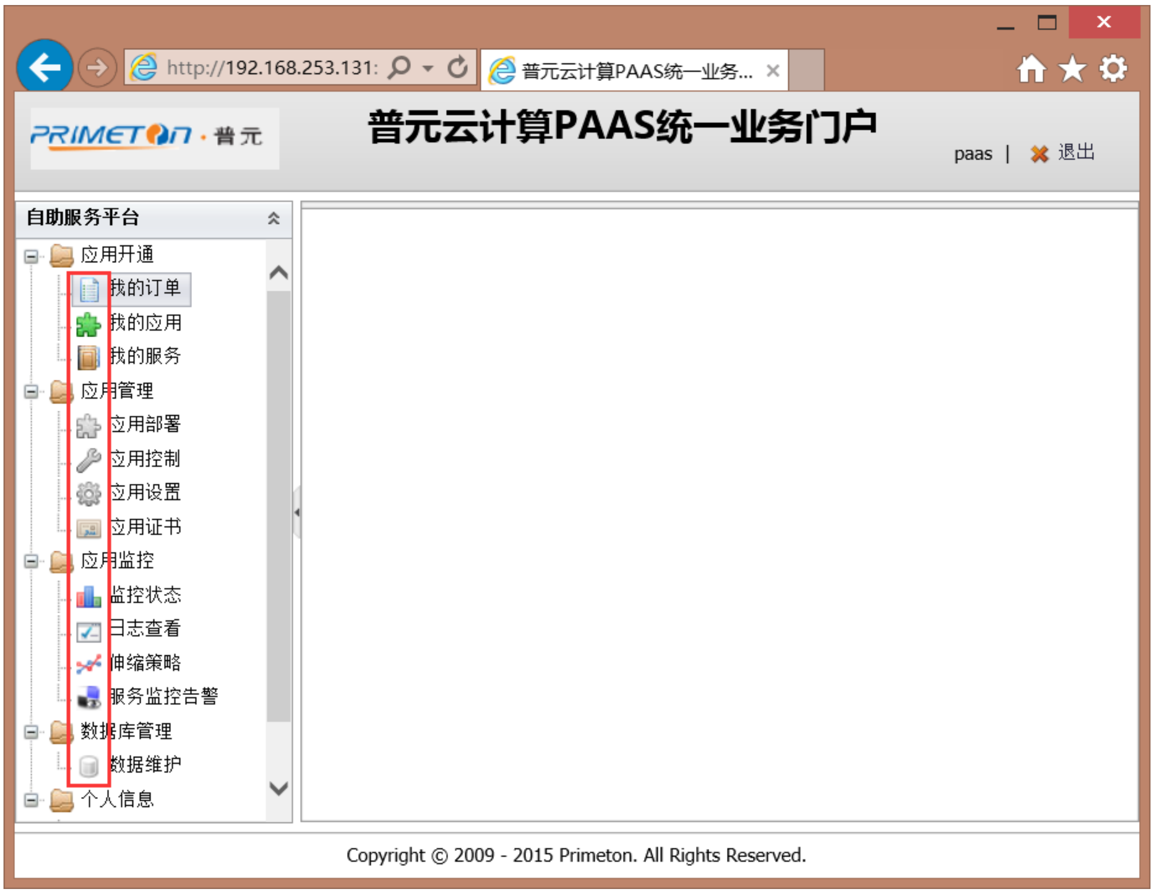
菜单配置文件${WebContentRoot}/common/data/app-menu.txt，采用 JSON 格式配置，内容如下：

```
[
    { id: "app", text: "自助服务平台"},
    { id: "appOpen", text: "应用开通",pid: "app"},
    { id: "myOrderMgr", pid: "appOpen", text: "我的订单", iconCls:
"app_myOrder", url: "../app/open/order/myOrderMgr.jsp" },
    { id: "myApplications", pid: "appOpen", text: "我的应用",
iconCls: "app_myApp", url:
"../app/open/application/myAppMgr.jsp" },
    { id: "myServices", pid: "appOpen", text: "我的服务", iconCls:
"app_myServices", url: "../app/open/service/myServicesMgr.jsp" },
    { id: "appManage", text: "应用管理",pid: "app"},
    { id: "listApplications_deploy", pid: "appManage", text: "应用部
署", iconCls: "app_deployApp", url:
"../app/mgr/deploy/listApplications.jsp" },
    { id: "listApplications_control", pid: "appManage", text: "应用
控制", iconCls: "app_controlApp", url:
"../app/mgr/control/listApplications.jsp" },
    { id: "appSetting", pid: "appManage", text: "应用设置", iconCls:
"app_settingApp", url: "../app/mgr/setting/appSetting.jsp" }   ,
    { id: "appCert", pid: "appManage", text: "应用证书", iconCls:
"app_certApp", url: "../app/mgr/cert/appCert.jsp" }   ,



    { id: "monitor", text: "应用监控",pid: "app"},
    { id: "appMonitor", pid: "monitor", text: "监控状态", iconCls:
"app_appMonitor", url: "../app/monitor/app/showAppMonitor.jsp" },
    { id: "appLog", pid: "monitor", text: "日志查看", iconCls:
"app_appLog", url: "../app/monitor/logs/appLog.jsp" },
    { id: "appStretchStrategy", pid: "monitor", text: "伸缩策略",
iconCls: "app_appStrategyConfig", url:
"../app/monitor/stretch/appStrategyConfig.jsp" },
    { id: "serviceAlarmStrategy", pid: "monitor", text: "服务监控告警
", iconCls: "platform_appServiceMonitor", url:
"../app/monitor/appservice/appServiceMonitorAndAlarm.jsp" },

    { id: "mysql", text: "数据库管理",pid: "app"},
    { id: "mySqlAdmin", pid: "mysql", text: "数据维护", iconCls:
"app_mySqlAdmin", url: "../app/data/db/mySqlAdmin.jsp" },



    { id: "person", text: "个人信息",pid: "app"},
    { id: "updatePwd", pid: "person", text: "修改密码", iconCls:
"app_updatePwd", url: "../app/account/update/updatePwd.jsp" },
    { id: "myAccount", pid: "person", text: "资料维护", iconCls:
"app_myAccount", url: "../app/account/info/myAccount.jsp" }
```

```
]
```

可以在这个配置文件中添加其他扩展的菜单。菜单的样式 iconCls（图片）可以修改${WebContentRoot}/css/icon.css；格式如下：

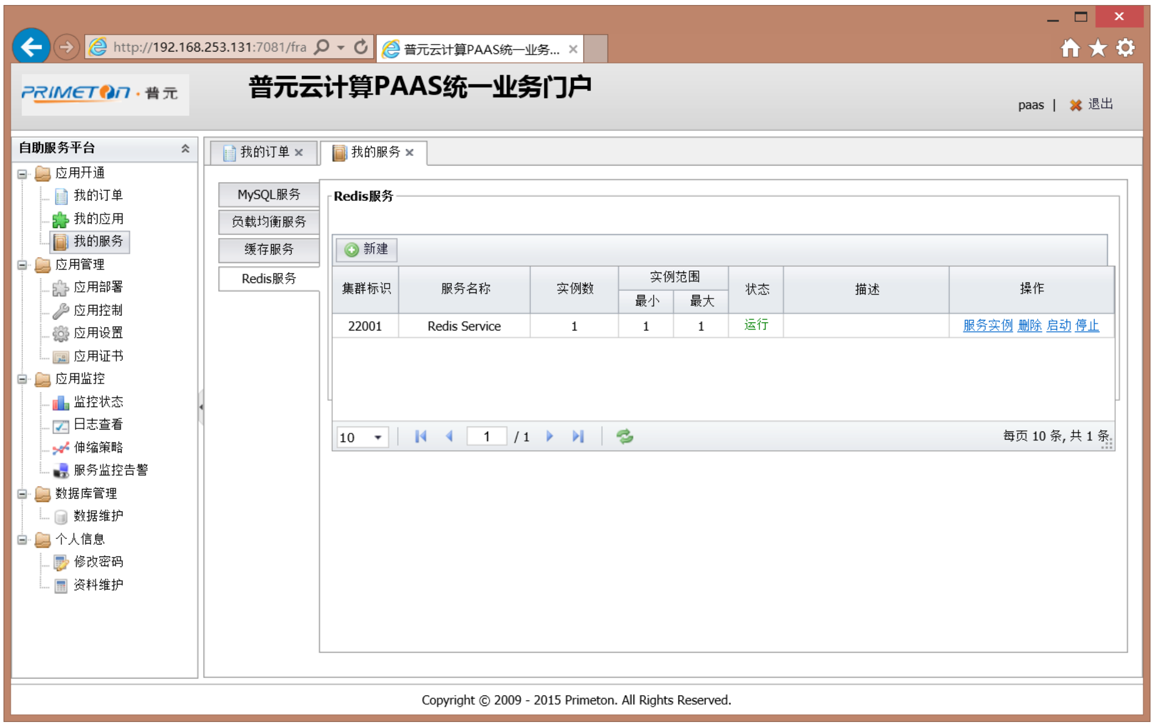
```
.default
{
    background:url(../images/icons/default.png) no-
repeat;width:16px;height:16px;
}

.app_deployApp
{
    background:url(../images/icons/app_deployApp.png) no-
repeat;width:16px;height:16px;

}
```



7.1.2 页面开发

一般展示个人的服务集群信息， 以及集群中的实例信息，提供创建服务向导；参考下面代码。

展示 Redis 集群信息：



http://192.168.253.131:7081/fra

普元云计算PAAS统一业务...

PRIMETON · 普元

普元云计算PAAS统一业务门户

paas | 退出

自助服务平台
- 应用开通
  - 我的订单
  - 我的应用
  - 我的服务
- 应用管理
  - 应用部署
  - 应用控制
  - 应用设置
  - 应用证书
- 应用监控
  - 监控状态
  - 日志查看
  - 伸缩策略
  - 服务监控告警
- 数据库管理
  - 数据维护
- 个人信息
  - 修改密码
  - 资料维护

我的订单 | 我的服务

MySQL服务
负载均衡服务
缓存服务
Redis服务

Redis服务

新建

集群标识	服务名称	实例数	实例范围		状态	描述	操作
			最小	最大			
22001	Redis Service	1	1	1	运行		服务实例 删除 启动 停止

10 | 1 /1 | 每页 10 条，共 1 条

Copyright © 2009 - 2015 Primeton. All Rights Reserved.

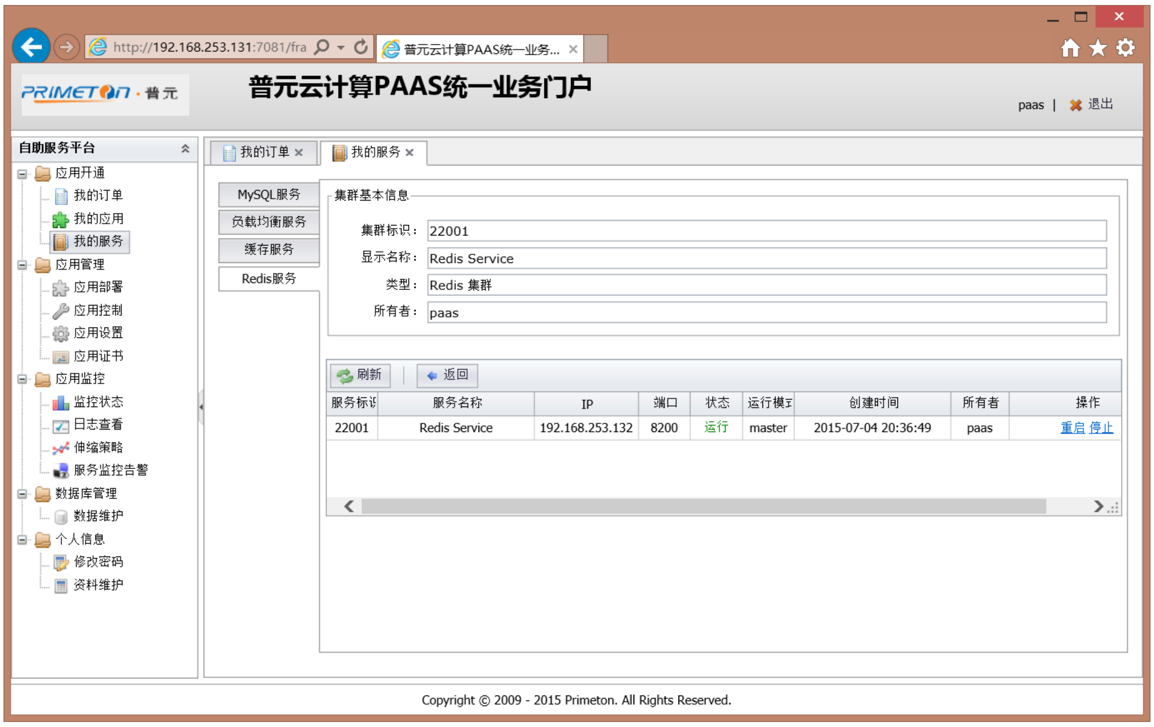
```
<%@page pageEncoding="UTF-8" contentType="text/html; charset=UTF-
8" %>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="content-type" content="text/html;
charset=UTF-8" />
        <script
src="<%=request.getContextPath() %>/common/nui/nui.js"
type="text/javascript"></script>
        <script
src="<%=request.getContextPath() %>/common/cloud/dictEntry.js"
type="text/javascript"></script>
        <script
src="<%=request.getContextPath() %>/common/cloud/common.js"
type="text/javascript"></script>
    </head>
    <body style="height:100%">
```

```
       <fieldset style="border:solid 1px 
#aaa;padding:3px;height:45%">
           <legend ><b>Redis服务</b></legend>
           <div id="datagrid" class="nui-datagrid" 
style="width:100%;height:90%;" allowResize="true" 
                 idField="id" multiSelect="true" 
url="<%=request.getContextPath() %>/srv/myService/listRedis">
                 <div property="columns">
                    <div field="id" width="60" 
headerAlign="center" align ="center" allowSort="true" >集群标识
</div>
                    <div field="name" width="120" 
headerAlign="center" align ="center" allowSort="true">服务名称
</div>
                    <div field="attributes.size" width="80" 
headerAlign="center" align ="center"  allowSort="true">实例数</div>
                   <div header="实例范围" headerAlign="center" 
align ="center" >
                       <div property="columns" 
headerAlign="center" align ="center">
                          <div field="minSize" width="50" 
headerAlign="center" align ="center">最小</div>
                          <div field="maxSize" width="50" 
headerAlign="center" align ="center">最大</div>
                       </div>
                    </div>


                    <div field="attributes.state" width="50" 
headerAlign="center" align ="center" allowSort="true" 
renderer="onSrvStatusRenderer">状态 </div>
                    <div field="desc" width="150" 
headerAlign="center" align ="center" allowSort="true" >描述 </div>
                    <div name="action" width="150" 
headerAlign="center" align="center" renderer="onDoActionRenderer" 
cellStyle="padding:0;">操作</div>
                 </div>
```

```
            </div>
        </fieldset>
        <script type="text/javascript">
        // parse DOM
        nui.parse();

        var datagrid = nui.get("datagrid");
        datagrid.load();

        function onDoActionRenderer(e) {
            var grid = e.sender;
            var record = e.record;
            var clusterId = record.id;
            var type = record.type;
            var state = record.attributes.status;
            if (state == 1) {
                return '<a href="javascript:detailsRow(\'' +
clusterId + '\')">服务实例</a> '
                    + ' <a href="javascript:removeCluster(\'' +
clusterId + '\'' + ',' + '\'' + type + '\')">删除</a> '
                    + ' <a href="javascript:restartCluster(\'' +
clusterId+'\'' +','+'\''+type+'\''+','+'\''+state+ '\')">重启</a> '
                    + ' <a href="javascript:stopCluster(\'' +
clusterId+'\'' +','+'\''+type+'\''+','+'\''+state+ '\')">停止</a>
';
            } else if (state == 0) {
                return '<a href="javascript:detailsRow(\'' +
clusterId + '\')">服务实例</a> '
                    + ' <a href="javascript:removeCluster(\'' +
clusterId + '\'' + ',' + '\''+ type + '\')">删除</a> '
                    + ' <a href="javascript:startCluster(\'' +
clusterId+'\'' +','+'\''+type+'\''+','+'\''+state+ '\')">启动</a> '
                    + ' <a href="javascript:stopCluster(\'' +
clusterId+'\'' +','+'\''+type+'\''+','+'\''+state+ '\')">停止</a>
';
            }
```

```
          return '<a href="javascript:detailsRow(\'' + clusterId +
'\')">服务实例</a> '
              + ' <a href="javascript:removeCluster(\'' + clusterId
+ '\'' + ',' + '\''+ type + '\')">删除</a> '
              + ' <a href="javascript:startCluster(\'' +
clusterId+'\'' +','+'\''+type+'\''+','+'\''+state+ '\')">启动</a> '
              + ' <a href="javascript:stopCluster(\'' +
clusterId+'\'' +','+'\''+type+'\''+','+'\''+state+ '\')">停止</a>
';;
       }

       function detailsRow(clusterId) {
          window.location="redisServices.jsp?clusterId=" +
clusterId;
       }

       function removeCluster(clusterId, type) {
          if (!confirm("确定删除该集群及集群内所有服务?")) {
              return;
          }
          $.ajax({
              url :
"<%=request.getContextPath() %>/srv/service/removeCluster",
              data : {clusterId : clusterId, type : type},
              type : "post",
              success : function (text) {
              },
              error : function (jqXHR, textStatus, errorThrown) {
                 nui.alert("系统错误！请稍后重试！<br/>" +
jqXHR.responseText);
              }
           });
          nui.alert("集群删除订单已提交，请到订单管理查看处理结果！");
       }

       function createCluster() {
          nui.open({
              url:
"<%=request.getContextPath() %>/app/open/service/createRedisServic
e.jsp",
              title: "新增Redis服务向导", width: 800, height: 250,
              onload: function () {
                 //nothing
              },
              ondestroy: function (action) {
                 if (action == 'ok') {
                     datagrid.reload();
                 }
              }
          });
       }

       function remove(gName) {
          var rows = nui.get(gName).getSelecteds();
          var count = rows.length;
          if (count != 1) {
              nui.alert('请选中一条记录!');
              return;
          }
          var row = rows[0];

          var row = rows[0];
          if (row.attributes.state == 2) {
              nui.alert('此服务已经是移除状态!');
              return;
          }

          var id = row.id;

          nui.mask({
              el: document.body,
              cls: 'mini-mask-loading',
              html: 'Loading...'
          });

          $.ajax({
              // appName , @PathParam("appType") String  appType
              url:
"<%=request.getContextPath() %>/srv/myService/removeService/" +
id,
              contentType: "application/json; charset=utf-8",
              cache: false,
              async: false,
              success: function (text) {
                 var o = nui.decode(text);
                 if (o) {
                     nui.alert('提交移除申请成功!');
                     reload();
                 } else {
                     nui.alert('提交移除申请失败!');
                 }
                 nui.unmask(document.body);
              }
          });

       }

       function reload() {
          datagrid.load();
       }

   </script>
</body>

</html>
```

Redis 服务实例信息：



集群标识：22001
显示名称：Redis Service
类型：Redis 集群
所有者：paas

服务标识	服务名称	IP	端口	状态	运行模式	创建时间	所有者	操作
22001	Redis Service	192.168.253.132	8200	运行	master	2015-07-04 20:36:49	paas	重启 停止

```
<%@page pageEncoding="UTF-8" contentType="text/html; charset=UTF-
8"%>


<%@page
import="com.primeton.paas.manage.api.service.RedisService"%>


<%
    String clusterId = request.getParameter("clusterId");
%>


<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="content-type" content="text/html;
charset=UTF-8" />
    <script src="<%=request.getContextPath() %>/common/nui/nui.js"
type="text/javascript"></script>
    <script
src="<%=request.getContextPath() %>/common/cloud/dictEntry.js"
```

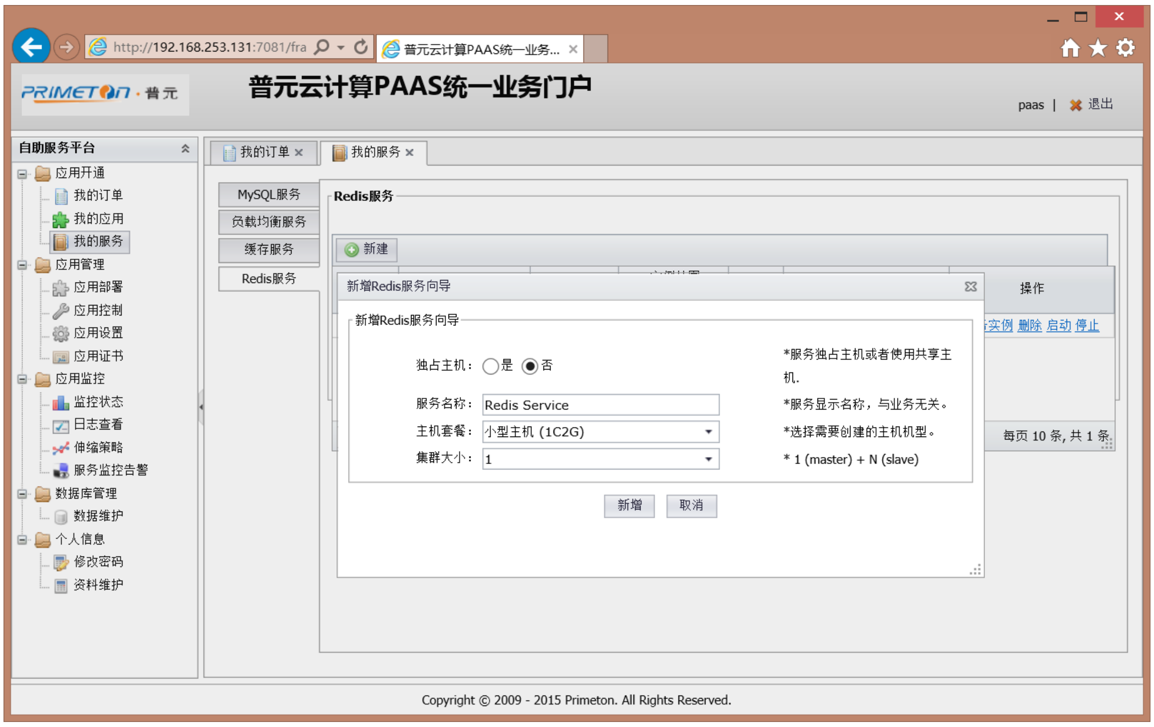
```
type="text/javascript"></script>
    <script
src="<%=request.getContextPath() %>/common/cloud/common.js"
type="text/javascript"></script>
    <style type="text/css">
    .asLabel .nui-textbox-border,
    .asLabel .nui-textbox-input,
    .asLabel .nui-buttonedit-border,
    .asLabel .nui-buttonedit-input,
    .asLabel .nui-textboxlist-border
    {
        border:0;background:none;cursor:default;
    }
    .asLabel .nui-buttonedit-button,
    .asLabel .nui-textboxlist-close
    {
        display:none;
    }
    .asLabel .nui-textboxlist-item
    {
        padding-right:8px;
    }
 </style>
 <script type="text/javascript">
        var datas = [{ id: 'true', text: '是'}
        , { id: 'false', text: '否'}
         ];
    </script>
</head>
<body>
    <form id="form1" method="post"
action="<%=request.getContextPath() %>/srv/service/getRedisCluster
Detail">
        <fieldset style="border:solid 1px #aaa;padding:3px;">
            <legend >集群基本信息</legend>
            <div style="padding:5px;">
```

```
              <table style="width:100%">
                  <tr>
                      <td style="width:10%;" align="right">集群标识：
</td>
                      <td style="width:80%;">
                          <input name="id" id="id" class="nui-
textbox asLabel" width="100%"/>
                      </td>
                  </tr>
                  <tr>
                      <td style="width:10%;"  align="right">显示名称：
</td>
                      <td style="width:80%;">
                          <input name="name" class="nui-textbox
asLabel" width="100%"/>
                      </td>
                  </tr>
                  <tr>
                      <td style="width:10%;"  align="right">类型：
</td>
                      <td style="width:80%;">
                          <input name="type" class="nui-textbox
asLabel" width="100%"/>
                      </td>
                  </tr>
                  <tr>
                      <td style="width:10%;"  align="right">所有者：
</td>
                      <td style="width:80%;">
                          <input name="owner" class="nui-textbox
asLabel" width="100%"/>
                      </td>
                  </tr>

              </table>
          </div>
```

```
        </fieldset>
    </form>
    <br/>
    <div style="width:100%;">
        <div class="nui-toolbar" style="border-
bottom:0;padding:0px;">
            <table style="width:100%;">
                <tr>
                    <!--
                    <td style="width:100%;">
                        <a class="nui-button" iconCls="icon-add"
onclick="increase()">新增实例</a><span class="separator"></span>
                        <a class="nui-button" iconCls="icon-remove"
onclick="decrease()">移除实例</a>
                    </td>
                     -->
                    <td style="white-space:nowrap;">
                        <a class="nui-button" iconCls="icon-reload"
onclick="reFresh()">刷新</a>
                        <span class="separator"></span>
                        <a class="nui-button" iconCls="icon-upgrade"
onclick="goBack()">返回</a>
                    </td>
                </tr>
            </table>
        </div>
    </div>
    <div id="servicegrid" class="nui-datagrid"
style="width:100%;height:120px" allowResize="true"
          idField="id" multiSelect="true"
url="<%=request.getContextPath() %>/srv/service/redisService/<%=cl
usterId %>" showPager="false" onloaderror="onLoadErrorRenderer">
        <div property="columns">
            <!-- <div type="checkcolumn" ></div> -->
            <div field="id" width="50" headerAlign="center" align
="center" allowSort="false" renderer="onActionRenderer">服务标识
```

```
</div>
          <div field="name" width="150" headerAlign="center" align
="center" allowSort="false">服务名称</div>
          <div field="ip" width="100" headerAlign="center" align
="center" allowSort="false">IP</div>
          <div field="port" width="50" headerAlign="center" align
="center" allowSort="false">端口</div>
          <div field="state" width="50" headerAlign="center" align
="center" renderer="onServiceStateRenderer">状态</div>
          <div field="attributes.runMode" width="50"
headerAlign="center" align ="center">运行模式</div>
          <div field="createdDate" width="150"
headerAlign="center" align ="center" allowSort="false"
renderer="onDateRenderer">创建时间</div>
          <div field="owner" width="100" headerAlign="center"
align ="center" allowSort="false">所有者</div>
          <div name="action" width="150" headerAlign="center"
align="center" renderer="onServiceActionRenderer"
cellStyle="padding:0;">操作</div>
       </div>
   </div>
   <script type="text/javascript">
       nui.parse();

       var grid = nui.get("servicegrid");
       var form = new nui.Form("form1");
       initForm();
       grid.load();

       function initForm() {
           var clusterId = '<%=clusterId%>';
           form.loading("操作中，请稍后 ...");
           $.ajax({
               url:
"<%=request.getContextPath() %>/srv/service/getRedisClusterDetail/
" + clusterId,
                success: function (text) {
                    form.unmask();
                    var o = nui.decode(text);
                    o.data.type = o.data.type + " 集群";//for display
                    o.data.attributes.storageSize =
o.data.attributes.storageSize + " G";
                    if (o.data.attributes.enableSession == ''
                            || o.data.attributes.enableSession == null)
{
                        o.data.attributes.enableSession = false;
                    }
                    setNUIForm(form, true, true); // see common.js
                    form.setData(o.data);
                    form.setChanged(false);
                },
                error: function (jqXHR, textStatus, errorThrown) {
                    form.unmask();
                    nui.alert(jqXHR.responseText);
                 }
            });
        }

        function reFresh() {
            grid.reload();
        }

        function goBack() {
            window.location = "myRedisClusters.jsp";
        }

    </script>
</body>
</html>
```

新建 Redis 服务向导：


http://192.168.253.131:7081/fra

普元云计算PAAS统一业务…

普元云计算PAAS统一业务门户

paas | 退出

自助服务平台

- 应用开通
  - 我的订单
  - 我的应用
  - 我的服务
- 应用管理
  - 应用部署
  - 应用控制
  - 应用设置
  - 应用证书
- 应用监控
  - 监控状态
  - 日志查看
  - 伸缩策略
  - 服务监控告警
- 数据库管理
  - 数据维护
- 个人信息
  - 修改密码
  - 资料维护

我的订单 × 我的服务 ×

MySQL服务
负载均衡服务
缓存服务
Redis服务

Redis服务

新建

操作

实例 删除 启动 停止

每页 10 条，共 1 条

新增Redis服务向导

新增Redis服务向导

独占主机：	○是 [✓ 否]	*服务独占主机或者使用共享主机.
服务名称：	Redis Service	*服务显示名称，与业务无关。
主机套餐：	小型主机 (1C2G)	*选择需要创建的主机机型。
集群大小：	1	* 1 (master) + N (slave)

新增 取消

Copyright © 2009 - 2015 Primeton. All Rights Reserved.

```
<%@page pageEncoding="UTF-8" contentType="text/html; charset=UTF-
8" %>


<%@page import="java.util.List"%>
<%@page import="java.util.ArrayList"%>
<%@page import="java.util.Map"%>
<%@page import="java.util.HashMap"%>
<%@page import="net.sf.json.JSONObject"%>
<%@page import="net.sf.json.JSONArray"%>
<%@page import="com.primeton.paas.console.common.SystemVariable"%>
<%@page
import="com.primeton.paas.manage.api.model.OrderItemAttr"%>
<%@page import="com.primeton.paas.manage.api.model.HostPackage"%>
<%@page import="com.primeton.paas.manage.api.model.OrderItem"%>
<%@page
import="com.primeton.paas.manage.api.service.RedisService"%>


<%
    HostPackage[] hostpkgs = SystemVariable.getAllHostPackages();
    List<Object> hostpkgsList = new ArrayList<Object>();
    for (HostPackage st : hostpkgs) {
        Map<String, Object> hostpkgsMap = new HashMap<String,
Object>();
        hostpkgsMap.put("id", st.getId());
        hostpkgsMap.put("text", st.getName() + " ("
                + st.getModel().getCoreCPU() + "C"
                + st.getModel().getMemorySize() + "G" + ")");
        hostpkgsList.add(hostpkgsMap);
    }
    Object hostpkgsData = JSONArray.fromObject(hostpkgsList);

    int maxSize = SystemVariable.getMaxRedisSize();
    maxSize = maxSize >= 1 ? maxSize : 5;
    List<Object> sizeList = new ArrayList<Object>();
    for (int i=1; i<=maxSize; i++) {
        Map<String, Object> entry = new HashMap<String, Object>();
        entry.put("id", String.valueOf(i));
        entry.put("text", String.valueOf(i));
        sizeList.add(entry);
    }
    Object sizeArrayData = JSONArray.fromObject(sizeList);
%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="content-type" content="text/html;
charset=UTF-8" />
    <script src="<%=request.getContextPath() %>/common/nui/nui.js"
type="text/javascript"></script>
    <script
src="<%=request.getContextPath() %>/common/cloud/dictEntry.js"
type="text/javascript"></script>
</head>
<body>
```

```
    <form id="form1" method="post"
action="<%=request.getContextPath() %>/srv/service/createRedis">
        <fieldset style="border:solid 1px #aaa;padding:3px;">
            <legend >新增Redis服务向导</legend>
            <div style="padding:5px;">
                <input name="order.itemList[0].itemType"
value="<%=RedisService.TYPE %>" class="nui-hidden" />
                <input name="order.orderType"
value="SingleCreateRedis" class="nui-hidden" /> <!-- 创建简单的Redis
服务集群及其服务实例 -->
                <table border="0">
                    <tr>
                        <td style="width:20%;" align="right">独占主机:
</td>
                        <td style="width:50%;">
                            <input
name="order.itemList[0].attrList[0].attrName"
value="<%=OrderItemAttr.ATTR_IS_STANDALONE%>" class="nui-hidden"
/>
                            <input
name="order.itemList[0].attrList[0].attrValue"  class="nui-
radiobuttonlist"  textField="text" valueField="id" value="N"
data=" [{ id: 'Y', text: '是'}, { id: 'N', text: '否'}]"
onvalidation="onNullValidation" width="80%"/>
                        </td>
                        <td align="left">*服务独占主机或者使用共享主
机.</td>
                    </tr>
                    <tr>
                        <td style="width:15%;" align="right">服务名称:
</td>
                        <td style="width:50%;">
                            <input
name="order.itemList[0].attrList[1].attrName"
value="<%=OrderItemAttr.ATTR_DISPLAY_NAME%>" class="nui-hidden" />
                            <input
```

```
name="order.itemList[0].attrList[1].attrValue" class="nui-textbox"
onvalidation="onNullValidation" value="<%=RedisService.TYPE %>
Service" width="80%"/>
                    </td>
                    <td align="left">*服务显示名称，与业务无关。</td>
                </tr>
                <tr>
                    <td style="width:15%;" align="right">主机套餐：
</td>
                    <td style="width:50%;">
                        <input
name="order.itemList[0].attrList[2].attrName"
value="<%=OrderItemAttr.ATTR_HOSTPKG_ID%>" class="nui-hidden" />
                        <input
name="order.itemList[0].attrList[2].attrValue"  class="nui-
combobox" data='<%=hostpkgsData %>' value="20130517J001"
width="80%"/>
                    </td>
                    <td align="left">*选择需要创建的主机机型。</td>
                </tr>
                <tr>
                    <td style="width:15%;" align="right">集群大小：
</td>
                    <td style="width:50%;">
                        <input
name="order.itemList[0].attrList[3].attrName"
value="<%=OrderItemAttr.ATTR_CLUSTER_SIZE%>" class="nui-hidden" />
                        <input
name="order.itemList[0].attrList[3].attrValue"  class="nui-
combobox" data='<%=sizeArrayData %>' value="1" width="80%"/>
                    </td>
                    <td align="left">* 1 (master) + N (slave)</td>
                </tr>
            </table>
        </div>
    </fieldset>
```

```
        <div style="text-align:center;padding:10px;">
            <a class="nui-button" style="width:8%; "
onclick="onAdd">新增</a>  &nbsp;
            <a class="nui-button" style="width:8%; "
onclick="onCancel">取消</a>
        </div>
    </form>

    <script type="text/javascript">
        nui.parse();

        var form = new nui.Form("form1");

        function onNullValidation(e) {
            if (!e.value) {
                e.errorText = "不能为空！";
                e.isValid = false;
            }
        }

        function closeWindow(action) {
            if (window.CloseOwnerWindow) {
                return window.CloseOwnerWindow(action);
            } else {
                window.close();
            }
        }

        function onAdd() {
            if (!confirm("你确定要提交Redis服务创建申请？")) {
                return false;
            }
            form.validate();
            if (form.isValid() == false) {
                return;
```

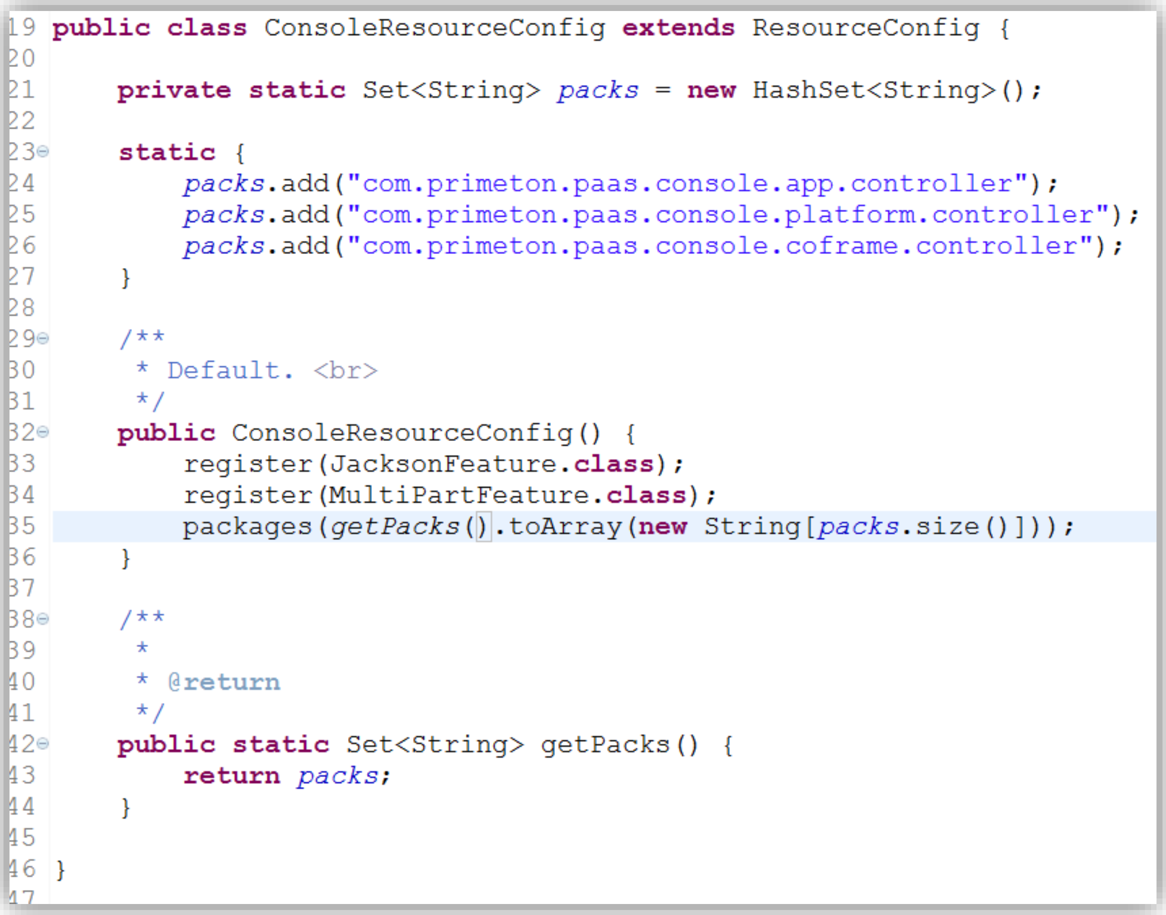
```
          }

          var data = form.getData().order;
          var json = nui.encode(data);

          $.ajax({
             url:
"<%=request.getContextPath() %>/srv/myService/createRedis",
             contentType: "application/json; charset=utf-8",
             data: json,
             type: "PUT",
             success: function (text) {
                nui.alert("订单已提交，请到订单管理中查看处理结果！");
             }
          });
          closeWindow("ok");
       }

       function onCancel(){
          closeWindow("cancel");
       }

    </script>
</body>

</html>
```

7.1.3 MVC : Controller 开发

Jersey Rest 风格开发，示例如下：

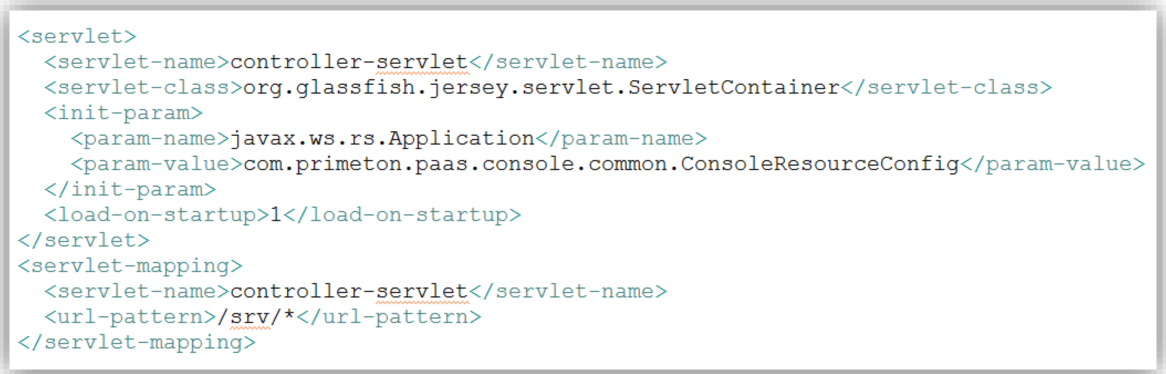
```
    /**
     *
     * @param order
     * @return
     */
    @Path("createRedis")
    @PUT
    @Produces(MediaType.APPLICATION_JSON)
    @Consumes(MediaType.APPLICATION_JSON)
    public Response createRedis(Order order) {
        if (order != null) {
            if (StringUtil.isEmpty(order.getOrderType())) {

    order.setOrderType(SingleCreateRedisOrderProcessor.TYPE);
            }
            // 创建订单提交,但不自动审批,返回订单id
            order = SrvApplyUtil.submitOrder(order, false);
        }
        Map<String, Object> result = new HashMap<String, Object>();
        result.put("orderId", null == order ? null :
order.getOrderId()); //$NON-NLS-1$
        ResponseBuilder builder = Response.ok(result);
        return builder.build();

    }
```

默认使用 ConsoleResourceConfig 注册了下面几个包，



```java
public class ConsoleResourceConfig extends ResourceConfig {

    private static Set<String> packs = new HashSet<String>();

    static {
        packs.add("com.primeton.paas.console.app.controller");
        packs.add("com.primeton.paas.console.platform.controller");
        packs.add("com.primeton.paas.console.coframe.controller");
    }

    /**
     * Default. <br>
     */
    public ConsoleResourceConfig() {
        register(JacksonFeature.class);
        register(MultiPartFeature.class);
        packages(getPacks().toArray(new String[packs.size()]));
    }

    /**
     *
     * @return
     */
    public static Set<String> getPacks() {
        return packs;
    }

}
```

如果扩展的 Controller 在其他包中，则需要覆盖 ConsoleResourceConfig 或直接继承 ResourceConfig 并修改 web.xml 配置，如下图所示：



```xml
<servlet>
  <servlet-name>controller-servlet</servlet-name>
  <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
  <init-param>
    <param-name>javax.ws.rs.Application</param-name>
    <param-value>com.primeton.paas.console.common.ConsoleResourceConfig</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>controller-servlet</servlet-name>
  <url-pattern>/srv/*</url-pattern>
</servlet-mapping>
```

示例代码-1：

```java
/**

 * Copyright © 2009 - 2015 Primeton. All Rights Reserved.
 */
package com.primeton.paas.console.common;

import java.util.HashSet;
import java.util.Set;

/**
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public class NewResourceConfig extends ConsoleResourceConfig {

   private static Set<String> newPacks = new HashSet<String>();

   static {
      newPacks.add("com.xxx.paas.app.controller");
      newPacks.add("com.xxx.paas.platform.controller");
   }

   /**
    * Default. <br>
    */
   public NewResourceConfig() {
      super();
      packages(newPacks.toArray(new String[newPacks.size()]));
   }

}
```

示例代码-2:

```
/**
 * Copyright © 2009 - 2015 Primeton. All Rights Reserved.
 */
package com.primeton.paas.console.common;
```

```

import java.util.HashSet;
import java.util.Set;

import org.glassfish.jersey.server.ResourceConfig;

/**
 * @author ZhongWen.Li (mailto:lizw@primeton.com)
 *
 */
public class NewResourceConfig extends ResourceConfig {

    private static Set<String> newPacks = new HashSet<String>();

    static {
        newPacks.add("com.xxx.paas.app.controller");
        newPacks.add("com.xxx.paas.platform.controller");
    }

    /**
     * Default. <br>
     */
    public NewResourceConfig() {
        super();
        packages(ConsoleResourceConfig.getPacks().toArray(
                new
String[ConsoleResourceConfig.getPacks().size()]));
        packages(newPacks.toArray(new String[newPacks.size()]));
    }

}
```

7.2 CONSOLE-PLATFORM

7.2.1 菜单定义

```
INSERT INTO cap_function (FUNC_ID, TENANT_ID, FUNC_NAME,
FUNC_TYPE, FUNC_DESC, FUNC_ACTION, ISCHECK, ISMENU, CREATEUSER,
CREATETIME) VALUES ('PAS_RedisMgr', 'default', 'Redis服务', 'page',
'Redis集群及服务管理', '../platform/srvmgr/redis/ClusterMgr.jsp',
'1', '1', 'sysadmin', '2014-05-30 06:28:47');
```

```
INSERT INTO cap_menu (MENU_ID, TENANT_ID, MENU_CODE, MENU_NAME,
LINK_TYPE, LINK_RES, LINK_ACTION, MENU_LEVEL, MENU_SEQ, ISLEAF,
PARENT_MENU_ID, IMAGEPATH, EXPANDPATH, OPENMODE, CREATEUSER,
CREATETIME) VALUES (284, 'default', 'PAS_1023_redis', 'Redis服务',
'function', 'PAS_RedisMgr',
'../platform/srvmgr/redis/ClusterMgr.jsp', 2, '42.284.', '1', 42,
'platform_redisClusterMgr', null, '0', 'sysadmin', '2015-01-23
11:45:02');
```

```
INSERT INTO cap_resauth (TENANT_ID, PARTY_ID, PARTY_TYPE, RES_ID,
RES_TYPE, RES_STATE, PARTY_SCOPE, CREATEUSER, CREATETIME) VALUES
('default', '102', 'role', 'PAS_RedisMgr', 'function', '1', '0',
'sysadmin', '2014-05-30 06:50:28');
```

7.2.2 页面开发

参考 6.1.2 章节。参考 Console 安装介质中的 WAR。

7.2.3 MVC : Controller 开发

参考 6.1.3 章节。