Bs3学习笔记

## Main方法,框架的启动入口

整个框架的学习，从main函数开始

com.umpay.front3.core.Main

|  |
| --- |
| public class Main {  public static void main(String[] args) throws Exception {  if (args == null || args.length == 0) {  System.out.println("请输入应用上下文配置文件路径!");  System.exit(-1);  }  addSecurityProvider();  String[] context = args[0].split(",", -1);  Context4Spring SprintContext = Context4Spring.getInstance(context);  Service4Inf r = (Service4Inf) SprintContext.getBean("ServiceMain");  r.start();  }  /\*\*  \* 增加SecurityProvider  \*/  private static void addSecurityProvider() {  try {  Security.addProvider(new com.sun.crypto.provider.SunJCE());  Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());  } catch (Exception e) {  }  }  } |

Front3\_common.xml

|  |
| --- |
| <!--主服务-->  <bean id="ServiceMain" class="com.bs2.core.ServiceMain" singleton="true">  <description>主服务：自动启动所有子服务</description>  <property name="\_context"><ref bean="ServiceContext" /></property>  <property name="\_manager"><ref bean="**ServiceGroup**" /></property>  <property name="\_exithook"><ref bean="ServiceHooking" /></property>  </bean>  <bean id="ServiceContext" class="com.bs2.core.Context4Spring" factory-method="getInstance" singleton="true" />  <bean id="ServiceHooking" class="com.umpay.front3.core.Hooking"></bean> |

app\_my.xml

|  |
| --- |
| <bean id="**ServiceGroup**" class="com.bs2.core.ServiceGroup">  <property name="\_context">  <ref bean="ServiceContext" />  </property>  <property name="\_namelist">  <list>  <!--1.服务队列-->  <value>recvUMQ</value>  <value>send2UMQ</value>  <value>recvRemoteQ</value>  <value>send2RemoteQ</value>  <!--2.C/S FOR UM-->  <value>Client\_UM\_HTTP</value>  <!--3.C/S FOR RM-->  <value>Server\_RM\_HTTP</value>  <!--6.定时器-->  <value>TickletGroup</value>  <!--7.本地模拟测试-->  <!--<value>BusiTrigger</value>-->  </list>  </property>  </bean> |

看下com.bs2.core.ServiceMain.start() 做了什么事情

|  |
| --- |
| public void start() {  int DL = Util.getDynamicLoading();// 动态加载级别，Ticklet/QObj;  // 显示运行状态  Runtime rt = Runtime.getRuntime();  StringBuffer sbInfo = new StringBuffer();  sbInfo.append("============= ServiceMain v3.4.110526 ============= ")  .append(Util.strDateTime()).append(CRLF);  sbInfo.append(Util.pn("Start Time", PLEN)).append(": ")  .append(Util.strDateTime()).append(CRLF);  sbInfo.append(Util.pn("CopyLeft", PLEN))  .append(": (1997-2011) Nike.Lius@GMail.Com").append("/DL:")  .append(DL).append(CRLF);  ………………….  ………………….  // 依次初始化各种服务 services  Object[] serviceIds = \_manager.getNamesAsArray();  if (\_context != null) {  for (int i = 0; i < serviceIds.length; i++) {  if (serviceIds[i] == null)  continue;  String svcId = (String) serviceIds[i];  Service4Inf svcOne = (Service4Inf) \_context.getBean(svcId,  Service4Inf.class);  \_log.debug("### START 服务(ID:" + svcId + ")/CLASS:"  + svcOne.getClass().getName());  **svcOne.start()**; // 启动线程运行！启动顺序由配置先后决定！  String svcName = Util.strBlank(svcId, 22);// 18,20  String svcInfo = svcOne.getInfo(); // 确保先start后getInfo！  if (i < 10)  sbInfo.append(' ');  sbInfo.append(i).append('.').append(svcName).append(": ");  sbInfo.append(svcInfo).append(CRLF);  }  }  } |

下面以其中一个队列recvUMQ看下此线程类如何工作?

首先看下该队列的配置

|  |
| --- |
| <bean id="recvUMQ" class="com.bs2.core.ext.Service4QObj">  <description>前置对业务平台的接收队列</description>  <property name="\_dependences">  <map>  <entry key="m\_beancreator"><ref bean="beancreator" /></entry><!-- ClassLoader -->  <entry key="m\_threadgroup"><ref bean="threadgroup4service" /></entry><!-- 线程组 -->  <entry key="m\_threadpool"><ref bean="threadpool4recvUMQ" /></entry><!-- 线程池 -->  <entry key="m\_classname"><value>com.umpay.core.RecvUMQHandler</value></entry>  <!-- 主动队列处理类 -->  <entry key="send2RemoteQ"><ref bean="send2RemoteQ" /></entry>  <entry key="send2UMQ"><ref bean="send2UMQ" /></entry>  <entry key="m\_busimonitor"><ref bean="m\_busimonitor" /></entry><!-- 交易监控器 -->  <entry key="m\_fifo\_remote"><ref bean="m\_fifo\_remote" /></entry><!-- 发送到对端系统的缓存 -->  <entry key="m\_fifo\_um"><ref bean="m\_fifo\_um" /></entry><!-- 发送到平台的缓存 -->  <entry key="m\_logFilter"><ref bean="m\_logFilter" /></entry><!-- 日志过滤器 -->  <entry key="m\_remoteExceptionHandler"><ref bean="m\_remoteExceptionHandler" /></entry>  <!-- 对端发起的报文异常处理 -->  </map>  </property>  <property name="\_properties">  <props>  <prop key="NAME">recvUMQ</prop>  </props>  </property>  </bean> |

在类com.bs2.core.ServiceMain讲解中，看到其调用每个队列的.start() 方法Service4QObj 继承了Service4QRun，start()方法定义在此类中

Service4QRun.start()

|  |
| --- |
| public void start() {  this.newHandler(); // 先创建处理类！（允许动态加载）  if (\_thread != null) return;  \_thread = this.newThread();  \_thread.start();  if (\_log.isInfoEnabled()) \_log.info("start()...hashcode="+this.hashCode());  } |

Service4QObj.newHandler()

|  |
| --- |
| protected void newHandler() {  try {  \_log.debug("# newHandler()..."+this.m\_classname);  Datalet2Inf h = (Datalet2Inf)m\_beancreator.newInstance(m\_classname);  if (h instanceof DependencyInjectionInf) {  DependencyInjectionInf h\_di = (DependencyInjectionInf)h;  if (\_properties!=null) h\_di.set\_properties(\_properties);  if (\_dependences!=null) h\_di.set\_dependences(\_dependences);  if (\_log.isDebugEnabled()) \_log.debug("# newHandler()..."+h);  }else if (\_log.isInfoEnabled()) \_log.info("# newHandler()...Not DependencyInjectionInf");  if (h!=null) this.m\_handler = h;//成功后才修改m\_executor  } catch(ClassNotFoundException e) {  if(\_log.isWarnEnabled()) \_log.warn("E newHandler() "+e);  } catch (InstantiationException e) {  if(\_log.isWarnEnabled()) \_log.warn("E newHandler() "+e);  } catch (IllegalAccessException e) {  if(\_log.isWarnEnabled()) \_log.warn("E newHandler() "+e);  }  } |

在start方法中首先调用newHandler, 通过m\_beancreator.newInstance(m\_classname);创建一个请求处理类，看下配置

|  |
| --- |
| <entry key="m\_beancreator"><ref bean="beancreator" /></entry><!-- ClassLoader -->  <entry key="m\_classname"><value>com.umpay.core.RecvUMQHandler</value></entry> |

其中 beancreator 是一个类加载器，用它可以创建一个类的实例，此处创建了com.umpay.core.RecvUMQHandler类的一个实例，该类是一个队平台请求数据的处理类。

|  |
| --- |
| public void **set\_dependences**(Map deps) {  m\_fifo\_remote = (FifoMapInf) Util.getMapValue(deps, "m\_fifo\_remote", FifoMapInf.class);  m\_fifo\_um = (FifoMapInf) Util.getMapValue(deps, "m\_fifo\_um", FifoMapInf.class);  m\_monitor = (BusiMonitor) Util.getMapValue(deps, "m\_busimonitor", BusiMonitor.class);  m\_logFilter = (LogFilterInf) Util.getMapValue(deps, "m\_logFilter", LogFilterInf.class);  if (deps.containsKey("m\_remoteExceptionHandler")) m\_remoteExceptionHandler = (ExceptionHandleInf) Util.getMapValue(deps, "m\_remoteExceptionHandler", ExceptionHandleInf.class);  if (deps.containsKey("send2RemoteQ")) send2RemoteQ = (Service4QObj) Util.getMapValue(deps, "send2RemoteQ", Service4QObj.class);  if (deps.containsKey("send2UMQ")) send2UMQ = (Service4QObj) Util.getMapValue(deps, "send2UMQ", Service4QObj.class);  if (deps.containsKey("recvRemoteQ")) recvRemoteQ = (Service4QObj) Util.getMapValue(deps, "recvRemoteQ", Service4QObj.class);  if (deps.containsKey("recvUMQ")) recvUMQ = (Service4QObj) Util.getMapValue(deps, "recvUMQ", Service4QObj.class);  } |

|  |
| --- |
| public void **set\_properties**(Properties props) {  m\_diyBusinessHandlerName = props.getProperty("DiyBusinessHandler");  if (!Tools.isEmpty(m\_diyBusinessHandlerName)) getDiyBusinessHandler();  } |