Appia

Uma introdução prática

Liliana Rosa
lrosa@lasige.di.fc.ul.pt

Appia

- Plataforma de composição e execução de protocolos.
- Linguagem *Java*
- Comunicação em grupo

Conceitos

- *Camadas* – descrição de um protocolo (st).
- *QdSs* – pilhas de camadas (st).
- *Sessões* – instâncias de protocolos com estado (dyn).
- *Canais* – pilhas de sessões (dyn).
- *Eventos* – estrutura de dados responsável pela troca de informação entre camadas.

Camadas

```
public class DelayLayer extends Layer{
   public DelayLayer(){
     evProvide=new Class[1];
     evProvide[0]=appia.protocols.delay.DelayTimer.class;

     evRequire= new Class[2];
     evRequire[0]=appia.protocols.group.intra.View.class;
     evRequire[1]= appia.protocols.group.events.GroupSendableEvent.class;

     evAccept=new Class[5];
     evAccept[0]= appia.protocols.group.events.GroupSendableEvent.class;
     evAccept[1]= appia.events.channel.ChannelInit.class;
     evAccept[2]= appia.protocols.delay.DelayTimer.class;
     evAccept[3]= appia.protocols.group.intra.View.class;
     evAccept[4]= appia.protocols.group.sync.BlockOk.class;}

   public Session createSession(){
     return new DelaySession(this);}}
```

Sessões

```
public class DelaySession extends Session{
    public void init(DelayConfig c){
      config = c; }
    public void handle(Event e){
      if(e instanceof ChannelInit){
          handleChannelInit((ChannelInit)e);
      }
      else if(e instanceof BlockOk){
          handleBlockOk((BlockOk)e);
      }
      else … }
    private void handleBlockOk(BlockOk e){
      blocked = true;
      try{
          e.go();
      }catch(AppiaEventException ex){
          ex.printStackTrace();
      }} … }
```

Qualidade de Serviço

```
private static Layer[] qos={
    new appia.protocols.tcpcomplete.TcpCompleteLayer(),
    new appia.protocols.group.bottom.GroupBottomLayer(),
    new appia.protocols.group.heal.GossipOutLayer(),
    new appia.protocols.group.inter.MergeOutLayer(),
    new appia.protocols.group.suspect.SuspectLayer(),
    new appia.protocols.group.intra.IntraLayer(),
    new appia.protocols.group.inter.InterLayer(),
    new appia.protocols.group.heal.HealLayer(),
    new appia.protocols.group.stable.StableLayer(),
    new appia.protocols.group.leave.LeaveLayer(),
    new appia.protocols.group.sync.VSyncLayer(),
    new ApplLayer()
  };
```

Canais

```
QoS myQoS = new QoS("Appl QoS", qos);

Channel myChannel =
    myQoS.createUnboundChannel("ApplChannel");

…

 myChannel.start();
```

Inicialização de Sessões em Canais

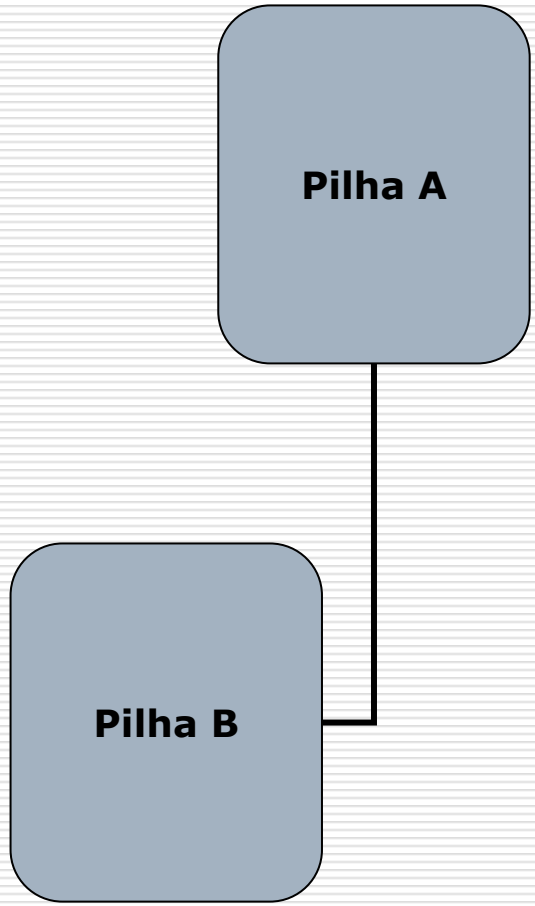
```
ApplSession as = (ApplSession)qos[qos.length-1].createSession();

as.init(port,multicast,gossips,group,viewAddrs);

ChannelCursor cc = myChannel.getCursor();

cc.top();
cc.setSession(as);

myChannel.start();
```

Eventos

```
public class DrawEvent extends GroupSendableEvent {

        private Point point;

        public DrawEvent() {
                super();}

        public DrawEvent(Point point) {
                super();
                this.point = point;}

        public Point getPoint() {
                return point;}

        public void setPoint(Point p) {
                point = p;}
}
```

Eventos

- Comunicação entre camadas da mesma pilha: acesso directo à informação.
- Comunicação entre camadas de pilhas diferentes: a informação necessita de ser serializada.



Eventos

- A

```
PointEvent pev = new PointEvent(point);
ObjectsMessage message = pev.getObjectsMessage();
message.pushObject(pev.getPoint());
```

… o evento é enviado para B

- B

```
message = pev.getObjectsMessage();
pev.setPoint((Point)message.popObject());
```

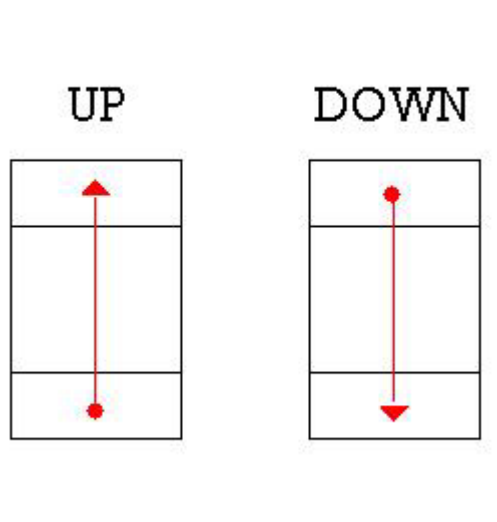
Eventos

- event.setSource(session);
- event.setChannel(channel);
- event.setDir(Direction.DOWN);

- event.dest = destinatario;

- event.init();

- event.go();
- event.asyncGo(channel, Direction.UP);



Camadas para Com. em Grupo

- *GroupBottom* – interface entre com ponto a ponto e de grupo.
- *Suspect* – detector de falhas
- *VSync* – sincronia virtual
- *Stable* – estabiliza as mensagens da vista
- *Leave* – retira membros do grupo
- *Intra* – mudança de vista
- *Inter* – junta vistas concorrentes
- *Heal* – detecta vistas concorrentes

Vistas no Appia

- Uma vista é um grupo de membros
- No Appia a gestão das vistas é feita com base na fusão de vistas concorrentes.
- Um novo membro cria uma vista em que ele é o único membro. Essa vista é depois unida à vista antiga, criando-se uma nova vista que incluiu o novo membro.
- A comunicação entre as vistas que se fundem é assegurada pela camada *MergeOutLayer*.

Sincronia Virtual

- O Appia assegura o sincronismo virtual através dos protocolos oferecidos.
- Todos os participantes sabem quem pertence à vista.
- As mudanças de vista são efectuadas na mesma sequência para todos os participantes. Ordenação total das vistas.
- Todas as mensagens enviadas numa vista são entregues antes da mudança de vista.

Sincronia Virtual no Appia

- Pedido: ViewChange
- Evento NewView
- Evento PreView
- Evento InstallView
- Evento BlockOK
- Evento Sync

Ordem

- Ordem Fifo:

  appia.protocols.fifo.FifoLayer
- Ordem Total:

  appia.protocols.totalAbcastLayer

GossipServer

- Necessário à comunicação em grupo no Appia.
- Servidor externo para detecção de vistas concorrentes.
- O seu funcionamento implica a camada *GossipOutLayer*, responsável pela comunicação entre o gossip e a pilha de com., utilizando um canal dedicado.
- A comunicação é conseguida através dos eventos *GossipOutEvent*.

Erros Frequentes

- Nenhuma camada da pilha fornece (Provide) um evento necessário a um determinada camada (Require).
- Os eventos necessários a um protocolo não são aceites na Layer correspondente.
- Não é feito o *go* de um evento logo as camadas não o recebem.

Referências

http://appia.di.fc.ul.pt

- Tutorial do Appia
- Appia Group Communication Manual
- API do Appia
- Javadoc
- Demos presentes na distribuição do Appia

Appia

AppiaXML

Conceito

- Configuração de um ou mais canais de comunicação utilizando uma descrição em XML.
- Interpretação, criação e inicialização de canais de comunicação do Appia.
- Facilita a modificação da configuração dos canais.

AppiaXML DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT appia (template*,channel*)>
<!ELEMENT template (session+)>
<!ATTLIST template
    name CDATA #REQUIRED>
<!ELEMENT session (protocol)>
<!ATTLIST session
    name CDATA #REQUIRED
    sharing (private|label|global) #REQUIRED>
<!ELEMENT protocol (#PCDATA)>
```

AppiaXML DTD

```
<!ELEMENT channel (chsession*)>
<!ATTLIST channel
    name CDATA #REQUIRED
    template CDATA #REQUIRED
    initialized (yes|no) #REQUIRED
    label CDATA #IMPLIED>
<!ELEMENT chsession (parameter*)>
<!ATTLIST chsession
    name CDATA #REQUIRED>
<!ELEMENT parameter (#PCDATA)>
<!ATTLIST parameter
    name CDATA #REQUIRED>
```

Template

- *Name*: nome do template
  - *Session* – uma sessão
    - *Name*: nome da sessão
    - *Sharing* [private, label, global]: partilha da sessão pelos canais.
      - *Protocol* – identificador do protocolo que constitui a sessão.

Channel

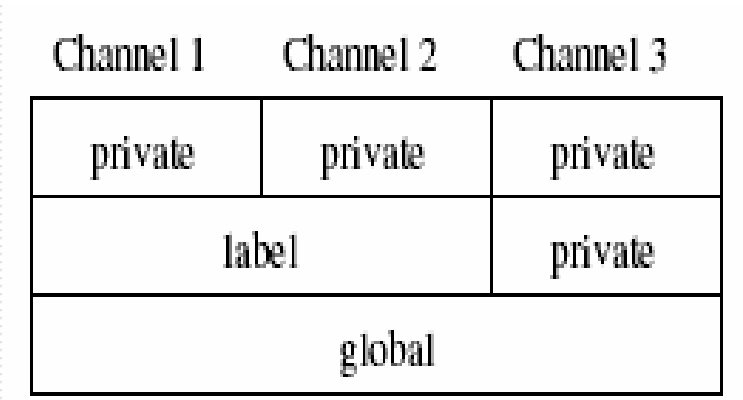
- *Name*: nome do canal a criar
- *Template*: a partir de que template deve ser criado
- *Initialized*: inicializar ou não o canal
- *Label*: canais com a mesma label partilham sessões com partilha *label*
  - *Chsession* – sessão a ser inicializada com parâmetros específicos.
    - *Name*: o nome da sessão em questão.
      - *Parameter* – parâmetro a passar
        - *Name*: nome correspondente

Exemplo de uma Configuração

```
<appia>
    <template name="ecco_t">
        <session name="tcp_s" sharing="global">

        <protocol>appia.protocols.tcpcomplete.TcpCompleteLayer</protocol>
        </session>
        <session name="ecco_s" sharing="private">
                <protocol>appia.test.xml.ecco.EccoLayer</protocol>
        </session>
    </template>
    <channel name="ecco_c" template="ecco_t" initialized="yes">
        <chsession name="ecco_s">
                <parameter name="localport">4000</parameter>
                <parameter name="remotehost">localhost</parameter>
                <parameter name="remoteport">4001</parameter>
        </chsession>
    </channel>
</appia>
```

Partilha de Sessões

- *Global*: as sessões são partilhadas todos os canais que as utilizam
- *Label*: as sessões são partilhadas pelos canais que têm a mesma sessão e label.
- *Private*: nunca existe partilha da sessão.



Passagem de Parâmetros (runtime)

- ChannelProperties
- SessionProperties
- Interface InitializableSession

```
ChannelProperties cp = new ChannelProperties();
SessionProperties sp = new SessionProperties();
sp.setProperty("hostname",ipfixo.getText());
sp.setProperty("port",portofixo.getText());
cp.putParams("wsl",sp);
```

Load das Configurações em XML

- AppiaXML.load(xmlfile);

  Apenas carrega a configuração.

- AppiaXML.loadAndRun(xmlfile);

  Carrega e executa a configuração;

  Para aplicações que não necessitam de efectuar inicializações das sessões em tempo de execução.

Criar Canais em Tempo de Execução

```
Channel ch =
    AppiaXML.createChannel(nome_canal,
    nome_template, label, channel_props,
    inicializado);
```

Referências

- Appia XML: A brief tutorial.
- Demo: Ecco