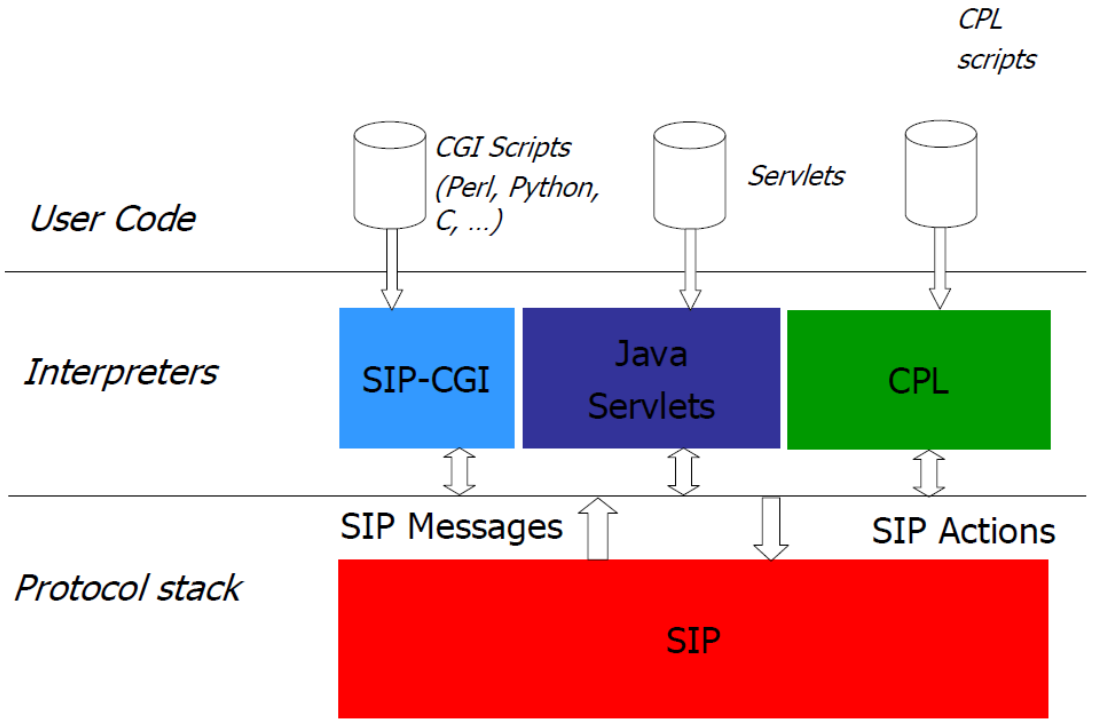
Servicios Multimedia Avanzados
Servicios sobre SIP

Máster I. de Telecomunicación

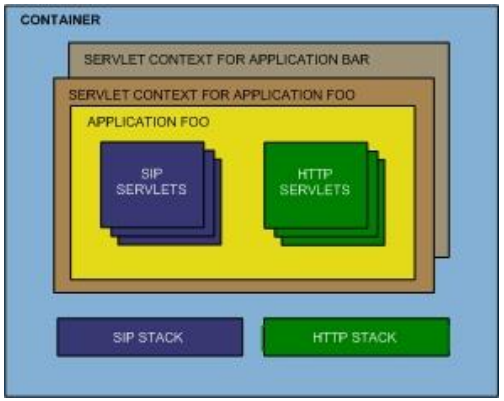
Múltiples posibilidades para crear servicios en un SIP AS

- SIP-CPL
- SIP-CGI
- SIP Servlet
- JAIN SIP
- JAIN SIP Lite

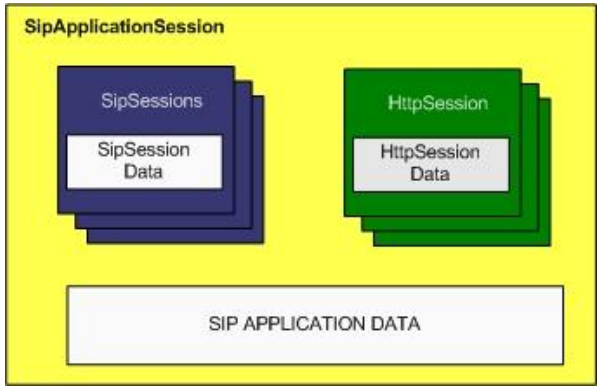


- Todas ellas se basan en la definición programática de cómo responder ante eventos generados por el protocolo de señalización (SIP)

Creando servicios SIP - Servlets SIP



- Contenedor de Servlets
  - javax.servlet.http
  - javax.servlet.sip
- ServletContext
  - Una app tiene sus descriptores de despliegue:
    - ✓ web.xml
    - ✓ sip.xml
- SipServlet
- SipServletRequest, SipServletResponse y SipServletMessage
- SipSession
- SipApplicationSession
- SipFactory
- Proxy
- Listeners…

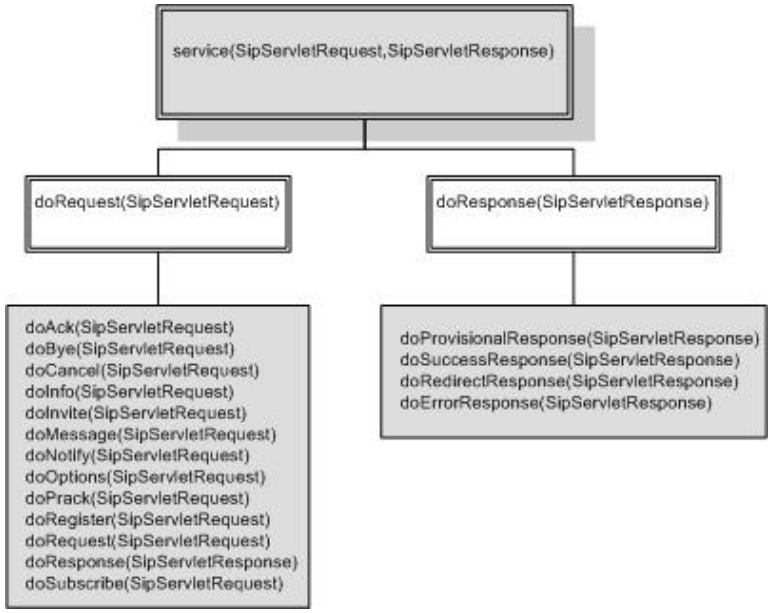


Servlets SIP

- **Se extiende la clase abstracta javax.servlet.sip.SipServlet**
  - **Métodos tipo doRequest (como doRegister) y doResponse (como doSuccessResponse)**
  - **Objetos a usar en estos métodos:**
    - ✓ **SipServletRequest**
    - ✓ **SipServletResponse**
    - ✓ **SipSession**
    - ✓ **SipApplicationSession**
- **Mirar**
  - **http://www.wesip.com/mediawiki/API/index.html**
  - **http://docs.oracle.com/cd/E19502-01/821-0203/gflsc/index.html**
  - **https://cipango.atlassian.net/wiki/display/DOC/Cipango+Documentation**
  - **http://www.cipango.org/javadoc/jsr289/**

Métodos a extender de SipServlet

- doInvite - para peticiones SIP INVITE
- doAck - para peticiones SIP ACK
- doOptions - para peticiones SIP OPTIONS
- doBye - para peticiones SIP BYE
- doCancel - para peticiones SIP CANCEL
- doRegister - para peticiones SIP REGISTER
- doSubscribe - para peticiones SIP SUBSCRIBE
- doNotify - para peticiones SIP NOTIFY
- doMessage - para peticiones SIP MESSAGE
- doInfo - para peticiones SIP INFO
- doPrack - para peticiones SIP PRACK
- doProvisionalResponse - para respuestas SIP 1xx
- doSuccessResponse - para respuestas SIP 2xx
- doRedirectResponse - para respuestas SIP 3xx
- doErrorResponse - para respuestas SIP 4xx, 5xx, y 6xx



Nos centraremos en doInvite

- protected void **doInvite**(SipServletRequest req) throws javax.servlet.ServletException, java.io.IOException
- Usaremos el objeto SipServletRequest para componer una respuesta o para reenviar el mensaje hacia el destinatario:
  - public SipServletResponse **createResponse**(int statuscode)
  - public Proxy **getProxy**(boolean create) throws TooManyHopsException

Un ejemplo de SipServlet

```
@SipListener
@SipServlet
public class SimpleProxyServlet
            extends SipServlet
            implements SipErrorListener,Servlet {

    /** Creates a new instance of SimpleProxyServlet */
    public SimpleProxyServlet() {
    }


    protected void doInvite(SipServletRequest request)
                throws ServletException, IOException {

                        if (request.isInitial()) {
                                Proxy proxy = request.getProxy();
                                proxy.setRecordRoute(true);
                                proxy.setSupervised(true);
                                proxy.proxyTo(request.getRequestURI()); // bobs uri
                        }
                        System.out.println("SimpleProxyServlet: Got request:\n" + request);
                }

    protected void doBye(SipServletRequest request) throws
                                ServletException, IOException {

        System.out.println("SimpleProxyServlet: Got BYE request:\n" + request);
        super.doBye(request);
    }


    protected void doResponse(SipServletResponse response)
        throws ServletException, IOException {

        System.out.println("SimpleProxyServlet: Got response:\n" + response);
                        super.doResponse(response);
    }

    // SipErrorListener

    public void noAckReceived(SipErrorEvent ee) {
        System.out.println("SimpleProxyServlet: Error: noAckReceived.");
    }

    public void noPrackReceived(SipErrorEvent ee) {
                        System.out.println("SimpleProxyServlet: Error: noPrackReceived.");
    }

}
```

Otro ejemplo sencillo de Servlet SIP

- **Un fragmento (no compilaría tal cual)**

http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/rsip_servletsample1.html?cp=SSAW57_8.5.5%2F1-18-11-29&lang=es

```
public class SimpleProxy extends SipServlet implements Servlet {

        public SimpleProxy() {
                super();
        }

        protected void doInvite(SipServletRequest request) throws ServletException,
IOException {

                try {
                        if (request.isInitial() == true)
                        {
                        Integer state = new Integer(INVITE_RECEIVED);
                        SipSession session = request.getSession();
                        session.setAttribute(STATE_KEY, state);
                        Proxy proxy = request.getProxy();
                        SipFactory sipFactory = (SipFactory)
getServletContext().getAttribute(SIP_FACTORY);
                         String callingNumber = request.getTo().toString();
                        String destStr = format_lookup(callingNumber);
                        URI dest = sipFactory.createURI(destStr);
                         proxy.setRecordRoute(true);
                        proxy.proxyTo(dest);
                }
}
```

Simplificando los Servlet SIP

Varios interfaces

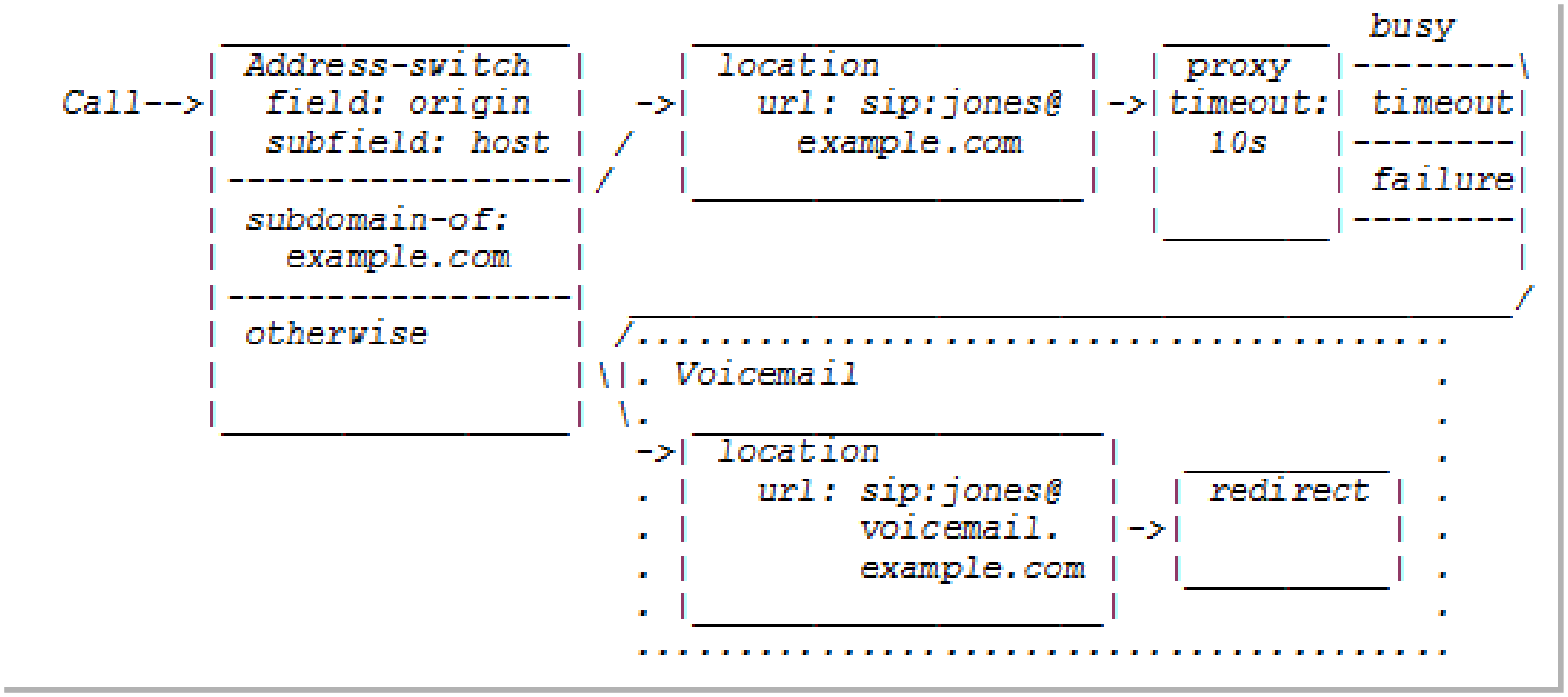
```
public interface SIPServletInterface {
        public void doInvite(SipServletRequest request);
}

public interface SipServletRequest {
        public String getCallerURI();
        public String getCalleeURI();
        public SipServletResponse createResponse(int statuscode);
        public Proxy getProxy();}

public interface SipServletResponse{
        public void send();
}

public interface Proxy {
        public void proxyTo(String uri);
}
```

Creando servicios SIP: CPL

- **Gramáticas XML para especificar el manejo de la llamada.**
- **Ejemplo CPL:**

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl xmlns="urn:ietf:params:xml:ns:cpl"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:ietf:params:xml:ns:cpl cpl.xsd ">
  <subaction id="voicemail">
    <location url="sip:jones@voicemail.example.com">
      <redirect />
    </location>
  </subaction>
  <incoming>
    <address-switch field="origin" subfield="host">
      <address subdomain-of="example.com">
        <location url="sip:jones@example.com">
          <proxy timeout="10">
            <busy> <sub ref="voicemail" /> </busy>
            <noanswer> <sub ref="voicemail" /> </noanswer>
            <failure> <sub ref="voicemail" /> </failure>
          </proxy>
        </location>
      </address>
      <otherwise>
        <sub ref="voicemail" />
      </otherwise>
    </address-switch>
  </incoming>
</cpl>
```

Creando servicios SIP: CPL



Creando servicios SIP: Nuestra gramática

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://sma.org/service.xsd"
xmlns=" http://sma.org/service.xsd" elementFormDefault="qualified">

 <xs:element name="service" type="serviceType"/>

 <xs:complexType name="serviceType">
  <xs:sequence>
   <xs:element name="inbound" type="inboundType"/>
   <xs:element name="outgoing" type="outgoingType"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="inboundType">
  <xs:sequence>
   <xs:element name="availableFrom" type="xs:string"/>
   <xs:element name="availableUntil" type=" xs:string"/>
   <xs:choice>
      <xs:element name="bannedUser" type="xs:string" minOccurs=”0” maxOccurs=”unbounded”>
      <xs:element name="allowedUser" type="xs:string" minOccurs=”0” maxOccurs=”unbounded”>
   </xs:choice>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="outgoingType">
  <xs:sequence>
   <xs:element name="availableFrom" type="xs:string"/>
   <xs:element name="availableUntil" type=" xs:string"/>
   <xs:choice>
      <xs:element name="bannedUser" type="xs:string" minOccurs=”0” maxOccurs=”unbounded”>
      <xs:element name="allowedUser" type="xs:string" minOccurs=”0” maxOccurs=”unbounded”>
   </xs:choice>
  </xs:sequence>
</xs:complexType>
```

Nuestro objetivo en la asignatura

- Desarrollar un proxy capaz de ejecutar una versión simplificada del API de Servlets SIP.
- Usar un Servlet especial que procese y actúe en función de lo especificado en un fichero XML

Ejercicios

- Codificar el método doInvite de un SIP Servlet que rechace todas las llamadas entre las 10 y las 22 horas.

- Utilizando la gramática XML simplificada, crear un servicio que permita llamar entre las 10 y las 22 horas exclusivamente a jones@domain.org y que pueda recibir en la misma franja horaria de cualquiera menos de jones y james.

Solución (I)

```
import java.util.Calendar;


protected void doInvite(SipServletRequest req) throws javax.servlet.ServletException, java.io.IOException
      {

  int hora, codigo;
  Calendar calendario = Calendar.getInstance();

  hora = calendario.get(Calendar.HOUR_OF_DAY);

  if ((hora < 10) || (hora > 22)) codigo=200  // OK
  else codigo = 500;   // error


 SipServletResponse resp = req.createResponse(codigo);
 resp.send();

}
```

Solución (II)

```
<?xml version="1.0"?><service>
<inbound>
    <availableFrom>10</availableFrom>
    <availableUntil>22</availableUntil>
    <bannedUser>sip:jones@domain.org</bannedUser>
    <bannedUser>sip:james@domain.org</bannedUser>
</inbound>
<outgoing>
    <availableFrom>10</availableFrom>
    <availableUntil>22</availableUntil>
    <allowedUser>sip:jones@domain.org</allowedUser>
</outgoing>
</service>
```