**JAVAMAIL**

**(JavaMail ejemplo gmail)**

El API JavaMail proporciona un marco independiente de la plataforma y de manera independiente del protocolo para construir aplicaciones de correo y mensajería.

**JavaMail** no viene por defecto con Java y necesita, además, otra librería adicional, **JAF**  (o **JavaBeans Activation Framework**).

**Introducción al API JavaMail**

El API JavaMail es un paquete opcional (extensión estándar) para leer, componer, y enviar mensajes electrónicos.

Usamos este paquete para crear programas del tipo MUA (**Mail User Agent**), similares a Eudora, Pine, y Microsoft Outlook. Su propósito principal no es transportar, enviar, o re-enviar mensajes como sendmail u otros programas del tipo MTA (**Mail Transfer Agent**). En otras palabras, los usuarios interactúan con los programas para leer y escribir e-mails. Los programas MUA tratan con los programas MTA para el envío real.

El API JavaMail está diseñado para proporcionar acceso independiente del protocolo para enviar y recibir mensajes dividiéndose en dos partes:

* La primera parte del API es el foco de este tutor. Básicamente, cómo enviar y recibir mensajes independientemente del proveedor/protocolo.
* La segunda parte habla de lenguajes específicos del protocolo como SMTP, POP, IMAP, y NNTP. Con el API, JavaMail para poder comunicar con un servidor, necesitamos un **proveedor** para un protocolo.

**Revisión de los Protocolos Relacionados**

Antes de mirar dentro de las especificaciones del API JavaMail, echemos un vistazo a los protocolos usados con el API. Básicamente son cuatro:

* SMTP
* POP
* IMAP
* MIME

También ejecutaremos sobre NNTP y algunos otros. Entender lo básico de cada protocolo nos ayudará a entender mejor cómo usar el API JavaMail. Aunque el API sea independiente del protocolo, no podemos evitar las limitaciones de los protocolos subyacentes. Si una capacidad no está soportada por el protocolo elegido, el API JavaMail no hará que esa capacidad aparezca por arte de magia.

(Como veremos pronto, este es un problema normal cuando trabajamos con POP).

**SMTP**

El protocolo **Simple Mail Transfer Protocol** (SMTP) está definido por la [**RFC 821**](http://www.cis.ohio-state.edu/htbin/rfc/rfc821.html). Define el mecanismo para enviar e-mail. En el contexto del API JavaMail, nuestro programa basado en JavaMail comunicará con el servidor SMTP de nuestro proveedor de servicios (ISP). Este servidor SMTP dejará el mensaje en el servidor SMTP del recipiente(s) para que sea recogido por los usuarios a través de POP o IMAP. Esto no requiere que nuestro servidor SMTP sea un role abierto, pues se utiliza la autentificación, pero es nuestra responsabilidad asegurarnos de que el servidor SMTP se configure correctamente. No hay nada en el API JavaMail sobre tareas como el configuración de un servidor para retransmitir mensajes o para agregar y para quitar cuentas del e-mail.

**POP**

POP viene de **Post Office Protocol.** Actualmente en la versión 3, también conocido como POP3, la [**RFC 1939**](http://www.imc.org/rfc1939) define este protocolo.

POP es el mecanismo que la mayoría de la gente usa en Internet para conseguir su correo. Define el soporte de un sólo mailbox por cada usuario. Ésto es todo lo que lo hace, y ésta también es la fuente de la mayoría de la confusión. Muchas de las cosas con que gente se familiariza cuando usa POP, como la capacidad de ver cuántos mensajes de correo nuevos tienen, no lo soporta POP en absoluto. Estas capacidades se construyen en programas como Eudora o Microsoft Outlook, que recuerdan cosas como los últimos correos recibidos y calculan cuántos tenemos nuevos. Así pues, al usar el API JavaMail, si queremos este tipo de información tendremos que calcularla nosotros mismos.

**IMAP**

IMAP es un protocolo más avanzado para recibir mensajes. Definido en la [**RFC 2060**](http://www.imc.org/rfc2060), IMAP viene de **Internet Message Access Protocol**, y está actualmente en la versión 4, también conocida como IMAP4. Para usar el IMAP, nuestro servidor de correo debe soportar este protocolo. No podemos simplemente cambiar nuestro programa para usar IMAP en vez de POP y que se soporte todo IMAP. Si asumimos que nuestro servidor de correo soporta IAMP, nuestro programa basado en JavaMail puede aprovecharse de los usuarios que tienen carpetas múltiples en el servidor y estas carpetas se pueden compartir por varios usuarios.

Debido a las capacidades más avanzadas, podríamos pensar que IMAP sería utilizado por todos. Pero no es así. Sobrecarga mucho el servidor de correo, requiriendo que el servidor reciba los nuevos mensajes, los entregué a los usuarios cuando sean solicitados, y los mantiene en las distintas carpetas de cada usuario. Aunque que esto centraliza las copias de seguridad, también hace que las carpetas de correo a largo plazo de los usuarios se hagan cada vez más grandes, y todo el mundo sufre cuando se agota el espacio en el disco. Con POP, los mensajes recuperados son eliminados del servidor de correo.

**MIME**

MIME viene de **Multipurpose Internet Mail Extensions**. No es un protocolo de transferencia de e-mail. En su lugar, define el contenido de lo que se está transfiriendo: el formato de los mensajes, los attachments, etc. Hay muchos documentos que tienen efecto sobre esto: las [**RFC 822**](http://www.imc.org/rfc822), [**RFC 2045**](http://www.imc.org/rfc2045), [**RFC 2046**](http://www.imc.org/rfc2046), y [**RFC 2047**](http://www.imc.org/rfc2047). Como usuario del API JavaMail, normalmente no tendremos que preocuparnos sobre estos formatos. Sin embargo, estos formatos existen y son utilizados por nuestros programas.

**NNTP y Otros**

A causa de la división del API JavaMail entre proveedor y cualquier otra cosa, podemos fácilmente soportar protocolos adicionales. Sun Mantiene una lista de [**proveedores de terceras partes**](http://java.sun.com/products/javamail/Third_Party.html) que se aprovechan de los protocolos para los que Sun no proporciona soporte. Allí encontraremos soporte para NNTP **Network News Transport Protocol** [newsgroups], S/MIME  **Secure Multipurpose Internet Mail Extensions**, y más.