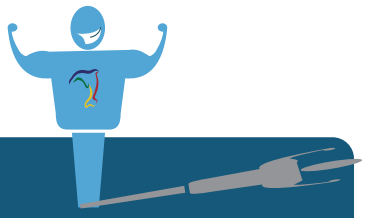
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

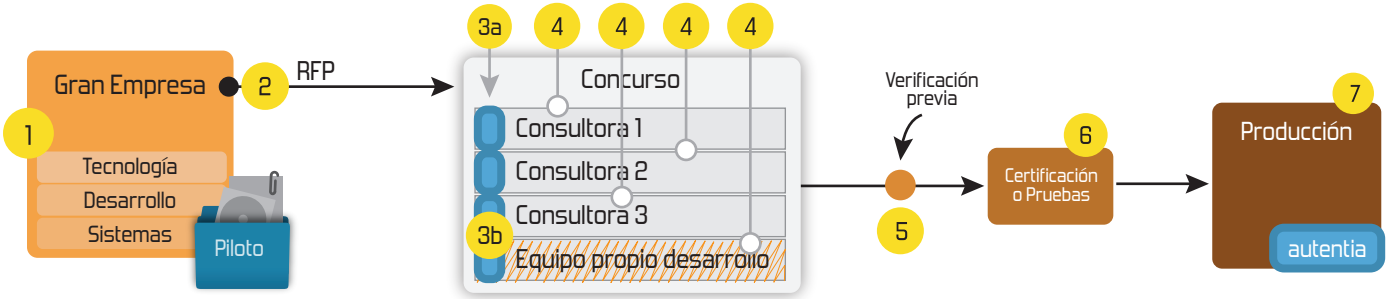
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por: Carlos García Pérez

Puedes encontrarme en Autentia
Somos expertos en Java/J2EE
Contacta en carlos.garcia@autentia.com

Jakarta Commons HttpClient

Introducción

Cuando necesitamos realizar tareas de comunicaciones bajo J2SE, normalmente hacemos uso del paquete java.net y sus extensiones.

java.net.HttpURLConnection y java.net.URL son, prácticamente las únicas clases que nos proporciona J2SE para abstraernos un poco de las comunicaciones bajo HTTP. Aunque cualquier tarea se puede hacer usando simplemente las clases que vienen de serie con la plataforma, estas son poco elegantes (en términos de Orientación a Objetos) para conseguir funcionalidad cliente HTTP.

HTTPClient, nos proporciona un amplio y elegante Framework para realizar las tareas de comunicación HTTP en el lado cliente.

Para descargar este excelente subproyecto de Jakarta u obtener una información más extensa, puede dirigirse a http://jakarta.apache.org/commons/httpclient

Características Principales

Desde mi punto de vista las principales características son:

1. Administración automática de Cookies.
2. Seguimiento automático de redirecciones.
3. Cancelación de peticiones en curso.
4. Soporte de gzip para ahorrar ancho de banda en nuestra comunicaciones.
5. Administrador de conexiones multihilo para poder tener varias tareas simultaneas en curso. (Descargarnos dos aplicaciones al mismo tiempo, por ejemplo)
6. Manejo automático de SSL
7. Logs de la información que se envia o recibe: Cabeceras enviadas/recibidas, Cookies, etc. (Muy útil para determinadas aplicaciones o para depuraciones).
8. Pool de conexiones para poder reutilizar conexiones sin tener que abrirlas o cerrarlas constantemente.
   Lo hace el API, según los parámetros de configuración que se le especifiquen. Por ejemplo, que se cierren automáticamente las conexiones que lleven 2 minutos sin ser usadas.
9. Poder subir ficheros vía POST sin tener que cargar en memoria en memoria la información que se desea subir.

Vamos a ver un ejemplo

En el siguiente ejemplo se tratan varios de los aspectos más comunes en cualquier comunicación HTTP. En concreto trateremos:

- Haremos una petición GET y otra POST añadiendo cabeceras personalizadas y parámetros.
- Configuraremos el cliente para salir a través de un Proxy
- Nos autentificaremos en caso de que el servidor lo requiera (Autentificación Digest o Basic).
- Seguiremos la redirecciones en caso de que se produzcan.
- Nos conectaremos a un servidor seguro (SSL)

He de resaltar que este API tiene dependencias con otros APIs de Jakarta Commons, por lo que debemos distribuirlos junto a nuestra aplicación para que funcione.

En concreto, estas dependencias son:

1. Jakarta Commons Codec. (Este para la gran mayoría de los casos no es necesario)
2. Jakarta Commons Logging.

HttpClientTutorial.java (El ejemplo es autocomentado)

```
package autentia.tutoriales.httpClient;

import org.apache.commons.httpclient.*;
import org.apache.commons.httpclient.auth.AuthScope;
import org.apache.commons.httpclient.methods.*;
import org.apache.commons.httpclient.params.HttpMethodParams;

import java.io.*;


/**
 * Ejemplo del uso de Jakarta Commons HttpClient
 * @author Carlos García. Autentia Real Business Solutions
 */
public class HttpClientTutorial {

          /**
           * En caso de salir a Internet a tavés de un Proxy, se deberán modificar los siguientes parámetros
           */
          public static final boolean byProxy      = false;
          public static final String  ProxyAddress = "TuProxyAddress";
          public static final int     ProxyPort    = 1234;

          /**
           * Direcciones que usamos para enviar peticiones
           */
          public static final String  TargetURLSSL = "https://www.verisign.com";
          public static final String  TargetURL   = "http://www.autentia.es";


          /**
           * Por defecto haremos una petición HTTP GET. Cambiar a false para peticiones HTTP POST
           */
          public static final boolean  ByGet = true;


          /**
           * Si queremos hacer conectarnos a un servidor seguro via SSL poner a true la siguiente variable
           */
          public static final boolean  BySSL = false;

          /**
           * En caso de necesitar autentificacion con el servidor modificar los siguientes parámetros
           */
          public static final boolean Authenticate = false;
          public static final String  UsernameCredentials = "TuUsuario";
          public static final String  PasswordCredentials = "TuPassword";




          /**
          * Punto de inicio de ejecución del ejemplo.
          */
          public static void main(String[] args) {
                    HttpClient      httpClient = null;  // Objeto a través del cual realizamos las peticiones
                    HttpMethodBase  request = null;     // Objeto para realizar las peticiines HTTP GET o POST
                    int             status = 0;         // Código de la respuesta HTTP
                    BufferedReader  reader = null;      // Se usa para leer la respuesta a la petición
                    String          line   = null;      // Se usa para leer cada una de las lineas de texto de la respuesta


                    // Instanciamos el objeto
                    httpClient = new HttpClient();


                    // Especificamos que salimos a través de un Proxy.
                    if (byProxy){
                              httpClient.getHostConfiguration().setProxy(HttpClientTutorial.ProxyAddress,
                  HttpClientTutorial.ProxyPort);
                    }

                    if (ByGet){
                              // Invocamos por GET

                              // ¿ Nos conectamos a un servidor seguro ?
                              // Observe que si están bien instalado JSSE (Java Secure Socket Extension),
                             //el código es exactamente igual
                              if (BySSL) {
                                        request = new GetMethod(HttpClientTutorial.TargetURLSSL);
                              } else {
                                        request = new GetMethod(HttpClientTutorial.TargetURL);
                              }


                              // Le indicamos que realize automáticamente el seguimiento de las redirecciones
                             // en caso de que existan.
                              request.setFollowRedirects(true);


                              // Añadimos los parámetros que deseemos a la petición
```

```
                        // GET: http://dominio/pagina?nombre=Carlos+Garc%C3%ADa
                        NameValuePair params[] = {new NameValuePair("nombre", "Carlos García")};
                        request.setQueryString(params);
                } else {
                        // Invocamos por POST
                        request = new PostMethod(HttpClientTutorial.TargetURL);

                        // Añadimos los parámetros que deseemos a la petición
                        ((PostMethod) request).addParameter("nombre", "Carlos García");
                }


                try {
                        // Si su servidor requiere autentificación necesitará una línea como la siguiente
                        if (Authenticate){
                           UsernamePasswordCredentials credentials = new
                           UsernamePasswordCredentials(UsernameCredentials, PasswordCredentials);
                           httpClient.getState().setCredentials(
                           new AuthScope(request.getURI().getHost(), 80, AuthScope.ANY_REALM), credentials);

                           // Indicamos que se autentifique si fuese requerido
                           request.setDoAuthentication(true);
                        }


                        // Indicamos reintente 3 veces en caso de que haya errores.
                        request.getParams().setParameter(HttpMethodParams.RETRY_HANDLER,
                           new DefaultHttpMethodRetryHandler(3, true));

                        // Añadimos las cabeceras personalizadas que se requieran, de la siguiente forma:
                        request.addRequestHeader("HeadKey", "HeadValue");

                        // Enviamos una cookie personalizada a las que el CookieManager ya tenga
                        httpClient.getState().addCookie(
                           new Cookie(request.getURI().getHost(), "nombreCookie", "valorCookie", "/", 3600, false));

                        // Leemos el código de la respuesta HTTP que nos devuelve el servidor
                        status = httpClient.executeMethod(request);

                        // Vemos si la petición se ha realizado satisfactoriamente
                        if (status != HttpStatus.SC_OK) {
                                System.err.println("Error\t" + request.getStatusCode() + "\t" +
                                request.getStatusText() + "\t" + request.getStatusLine());
                        } else {
                                // Leemos el contenido de la respuesta y realizamos el tratamiento de la misma.
                                // En nuestro caso, simplemente mostramos el resultado por la salida estándar
                                reader = new BufferedReader(
                              new InputStreamReader(request.getResponseBodyAsStream(), request.getResponseCharSet()));
                                line   = reader.readLine();
                                while (line != null) {
                                        System.out.println(line);
                                        line = reader.readLine();
                                }
                        }
                } catch (Exception ex){
                        System.err.println("Error\t: " + ex.getMessage());

                        ex.printStackTrace();
                } finally {
                        // Liberamos la conexión. (También libera los stream asociados)
                        request.releaseConnection();
                }
        }
}
```

Bueno, espero que os haya sido de utilidad este tutorial.

En Autentia Real Business Solutions, nos gusta compartir el conocimiento. Aquí teneis un poquito más de nuestra aportación.

Si algún día necesitais ayuda con vuestros proyectos o necesitais formación, podéis encontrarnos en Autentia

Puedes opinar sobre este tutorial aquí

Recuerda

que el personal de Autentia te regala la mayoría del conocimiento aquí compartido (Ver todos los tutoriales)

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?

¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

info@autentia.com

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Activar soporte SSL en Tomcat	Os mostramos como activar el acceso SSL en Tomcat, utilizando certificados generados por Keygen (java)
Establecimiento de conexiones HTTP desde un MIDLET	Este tutorial nos va a ayudar para resolver los problemas de conexión HTTP desde un MIDLET.
Activar SSL en IIS	Os mostramos comos activar el soporte de https en IIS, creando vuestros propios certificados autofirmados, usando OpenSSL
Certificados en IIS para activación SSL	En este tutorial vamos a habilitar el soporte SSL (Secure Socket Layer, comunicación segura por https) en un servidor IIS (Internet Information Server de Microsoft).
Leer un documento de Microsoft Word usando la librería POI de jakarta	En este docuemento aprenderemos a leer un documento de Microsoft Word usando la librería POI de jakarta
Struts Jakarta	Cuando se ha trabajado creando aplicaciones Java poco a poco se va viendo la necesidad de normalizar los desarrollo. Uno de los Framework (entornos) más extendidos es Struts
Creación e invocación de Webservices por SSL	En este tutorial se pretende enseñar a desplegar un webservice usando SSL y a invocarlo correctamente
Activar el soporte SSL en Struts	Os mostramos las particularidades de uso y configuración de Struts para trabajar con SSL