Login Cidadão & OAuth 2

Posted on **12/11/2019** by **Carlos Souza**

O Login Cidadão utiliza o protocolo OAuth 2, o qual é muito utilizado pelos fornecedores de mídias sociais, como Google, Facebook, Twiter, entre outros, fornecendo uma forma de autenticação por outros canais.

Esse protocolo possui descritos suas caraterísticas e especificações muito bem definidas em RFCs e, além disso, muitas outras informações e dicas de como utilizar esse tipo de autenticação podem ser encontrados na internet.

Em breve, teremos um documento passo a passo de como utilizar a autenticação via Login Cidadão, dentro dos padrões da Procergs.

Por enquanto, podemos deixar aqui alguns links para minimizar a procura por informações sobre esse assunto.

Conhecendo o protocolo OAuth 2

Aqui deixo alguns links interessantes falando sobre o protocolo:

- Uma introdução ao OAuth 2
- OAuth 2 – Guia do Iniciante – aqui temos as referências às RFCs e outras mais gerais
- Como funciona o protocolo OAuth 2.0
- Uma introdução OAuth 2.0 para iniciantes (em inglês)

Além desses, temos dois artigos nossos que falam um pouco sobre OAuth 2:

- Papo Reto sobre OAuth 2 – tem um vídeo explicativo
- OAuth2 – uma abordagem para segurança de aplicações e APIs REST – que tem uma apresentação da InfoQ

E temos um documento, em inglês, que fala sobre:

- OpenID Connect Dynamic Client Registration 1.0

Aqui temos uma pequena introdução ao OAuth 2, protocolo de autenticação utilizado pelo Login Cidadão.

Qual URL do Login Cidadão utilizar?

O Login Cidadão deve ser utilizado com a URL de homologação para validação de aplicações em desenvolvimento e após liberado, deve utilizar somente a URL de produção.

As URLs são:

- **Produção:** https://logincidadao.rs.gov.br/openid/connect/authorize?
- **Homologação:** https://meu.hml.rs.gov.br/openid/connect/authorize?

Existe a URL de desenvolvimento, mas esta não deve ser utilizada, pois ela é constantemente alterada para validação da aplicação. A versão de homologação é mais estável e preferencial para utilização no período de teste e homologação das demais aplicações que vão utilizar o acesso via ***Login Cidadão***.

Temos um pequeno exemplo, de uma classe em Java, para poderem seguir como modelo. Essa classe será a nossa base para montar o passo a passo de como utilizar o ***Login Cidadão*** da melhor forma possível e sem grandes transtornos.

Exemplo de código Java: (*LogonLoginCidadao.class*)

```java
package com.procergs.meu.logincidadao;

import java.io.IOException;
import java.io.InputStream;
import java.io.Serializable;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;
import java.util.Scanner;
import javax.faces.context.FacesContext;
import javax.inject.Inject;
import javax.inject.Named;
import javax.security.sasl.AuthenticationException;
import javax.servlet.http.HttpServletRequest;
import org.apache.commons.lang3.StringUtils;
import org.omnifaces.cdi.ViewScoped;
import org.slf4j.Logger;
import com.google.i18n.phonenumbers.PhoneNumberUtil;
import com.google.i18n.phonenumbers.Phonenumber.PhoneNumber;
import com.nimbusds.oauth2.sdk.AuthorizationCode;
import com.nimbusds.oauth2.sdk.AuthorizationCodeGrant;
import com.nimbusds.oauth2.sdk.AuthorizationErrorResponse;
import com.nimbusds.oauth2.sdk.AuthorizationGrant;
import com.nimbusds.oauth2.sdk.AuthorizationResponse;
import com.nimbusds.oauth2.sdk.AuthorizationSuccessResponse;
import com.nimbusds.oauth2.sdk.ParseException;
import com.nimbusds.oauth2.sdk.ResponseType;
```

```
35 import com.nimbusds.oauth2.sdk.Scope;
36 import com.nimbusds.oauth2.sdk.TokenRequest;
37 import com.nimbusds.oauth2.sdk.TokenResponse;
38 import com.nimbusds.oauth2.sdk.auth.ClientAuthentication;
39 import com.nimbusds.oauth2.sdk.auth.ClientSecretBasic;
40 import com.nimbusds.oauth2.sdk.auth.Secret;
41 import com.nimbusds.oauth2.sdk.http.HTTPResponse;
42 import com.nimbusds.oauth2.sdk.id.ClientID;
43 import com.nimbusds.oauth2.sdk.token.BearerAccessToken;
44 import com.nimbusds.openid.connect.sdk.AuthenticationRequest;
45 import com.nimbusds.openid.connect.sdk.OIDCTokenResponse;
46 import com.nimbusds.openid.connect.sdk.OIDCTokenResponseParser;
47 import com.nimbusds.openid.connect.sdk.Prompt;
48 import com.nimbusds.openid.connect.sdk.Prompt.Type;
49 import com.nimbusds.openid.connect.sdk.UserInfoRequest;
50 import com.nimbusds.openid.connect.sdk.op.OIDCProviderMetadata;
51 import com.procergs.arqjava4.faces.FacesUtil;
52 import com.procergs.dtw.dtw_cs.dto.enumeration.TipoPessoaEN;
53 import com.procergs.dtw.dtw_cs.infra.DtwApplicationScoped;
54 import com.procergs.dtw.dtw_cs.infra.DtwSessionScoped;
55 import com.procergs.dtw.dtw_cs.log.LogRN;
56 import com.procergs.dtw.dtw_cs.log.TipoLogEN;
57 import com.procergs.dtw.dtw_cs.logincidadao.ed.LoginCidadaoED;
58 import com.procergs.dtw.dtw_cs.util.SistemaUtil;
59 import com.procergs.dtw.dtw_cs.wsclients.rpd.RpdRestClient;
60 import com.procergs.dtw.dtw_cs.wsclients.rpd.ed.EmailVO;
61 import com.procergs.dtw.dtw_cs.wsclients.rpd.ed.PessoaVO;
62 import com.procergs.dtw.dtw_cs.wsclients.rpd.ed.TelefoneVO;
63 import net.minidev.json.JSONObject;
64
65 @Named
66 @ViewScoped
67 public class LogonLoginCidadao implements Serializable {
68
69     private static final long serialVersionUID = 1L;
70     private final String URL_RESUMO = "resumo";
71     private final String URL_CONFIRMACAO = "confirmacao";
72     private final String URL_LOGIN = "loginErroAutenticacao";
73     private String url;
74
75  @Inject
76     Logger logger;
77     @Inject
78     DtwApplicationScoped dtwApplicationScoped;
79     @Inject
80     DtwSessionScoped dtwSessionScoped;
81     @Inject
82     UsuarioLCLogadoMB usuarioLCLogadoMB;
83     @Inject
84     RpdRestClient rpdRestClient;
85     @Inject
86     LogRN logRN;
87
88     /*
89   *
90   *
91   */
92  public void autenticacaoLoginCidadao() throws IOException {
93
94  HttpServletRequest request = (HttpServletRequest) FacesContext.getCurrentInstance().getExternalCor
95
96  try {
97
98             /** Return the OpenID Connect provider metadata. */
99             OIDCProviderMetadata providerMetadata = montaOIDCProviderMetadata();
100            /** Creates a new OpenID Connect authentication request builder. */
101            AuthenticationRequest requisicaoParaAutorizacaoLoginCidadao = montaAuthenticationReques
102            FacesContext.getCurrentInstance().getExternalContext().redirect(requisicaoParaAutoriza
103
```

```
        } catch (AuthenticationException e) {
            logger.error("AuthenticationException - e.getMessage(): " + e.getMessage() + ": " + e);
            FacesUtil.addErrorMessage("A central de serviços está temporariamente indisponível, ter
            FacesContext context = FacesContext.getCurrentInstance();
            context.getExternalContext().getFlash().setKeepMessages(true);

        } catch (Exception e) {
            logger.error("Exception - e.getMessage(): " + e.getMessage() + ": " + e);
            FacesContext context = FacesContext.getCurrentInstance();
            context.getExternalContext().getFlash().setKeepMessages(true);
            FacesUtil.addErrorMessage("A central de serviços está temporariamente indisponível, ter

        }
    }

/*
 *
 *
 */
private OIDCProviderMetadata montaOIDCProviderMetadata() throws AuthenticationException {

Scanner s = null;

try {

            URI issuerURI = new URI(dtwApplicationScoped.getLOGINCIDADAO_URL().get());
            URL providerConfigurationURL = issuerURI.resolve("/.well-known/openid-configuration").t
            /** Opens a connection to this {@code URL} and returns an {@code InputStream} for readi
            InputStream stream = providerConfigurationURL.openStream();
            s = new Scanner(stream);
            String providerJSON = s.useDelimiter("\\A").hasNext() ? s.next() : "";
            return OIDCProviderMetadata.parse(providerJSON);

        } catch (URISyntaxException | IOException | ParseException e) {
            throw new AuthenticationException("montaOIDCProviderMetadata(): " + e.getMessage(), e);

        } finally {
            if( s != null )
                s.close();
            // throw new AuthenticationException("validaOIDCTokenResponse(): TESTE 1...");

        }

    }

/*
 *
 *
 */
    private AuthenticationRequest montaAuthenticationRequest(HttpServletRequest request, OIDCProvic

AuthenticationRequest.Builder authRequest = null;

try {

            authRequest = new AuthenticationRequest.Builder(new ResponseType(ResponseType.Value.COD

if( StringUtils.isNotBlank(request.getParameter("prompt")) )
                authRequest.prompt(new Prompt(Type.CONSENT));

            /** Creates a new OpenID Connect authentication request. */
            AuthenticationRequest requisicaoParaAutorizacaoLoginCidadao = authRequest.build();
            // throw new AuthenticationException("validaOIDCTokenResponse(): TESTE 2...");
            return requisicaoParaAutorizacaoLoginCidadao;

        } catch (URISyntaxException e) {
            throw new AuthenticationException("montaAuthenticationRequest(): " + e.getMessage(), e)

        }
```

```
    }

/*
 *
 *
 */
   public String autenticadoLoginCidadao() {

HttpServletRequest request = (HttpServletRequest) FacesContext.getCurrentInstance().getExternalCor

try {

            AuthorizationCode code = this.obtemAutorizationCode(request);
            OIDCTokenResponse token = this.obtemToken(code, request);
            LoginCidadaoED loginCidadaoED = this.obtemDadosUsuario(token);
            logger.info("loginCidadaoED.getNroIdLC(): " + loginCidadaoED.getNroIdLC() + " loginCida
            usuarioLCLogadoMB.setIdToken(loginCidadaoED.getTokenResponse());
            usuarioLCLogadoMB.setLoginCidadaoED(loginCidadaoED);

            if( this.validaOIDCTokenResponse(token) ) {

                if( this.validaLoginCidadaoED(loginCidadaoED) ) {

PessoaVO pessoaVO = consultarPessoaRpd(loginCidadaoED);

if( pessoaVO.getCpfCnpj() != null ) {

usuarioLCLogadoMB.setPessoa(pessoaVO);
                        usuarioLCLogadoMB.atualizaVinculos();
                        this.setUrl(URL_RESUMO);

                    } else {

                        pessoaVO.setIdCentralServicos(loginCidadaoED.getNroIdLC());
                        pessoaVO.setCpfCnpj(loginCidadaoED.getTxtCpfLC());
                        pessoaVO.setNome(loginCidadaoED.getTxtNomeLC());
                        pessoaVO.setListaEmails(parseEmailLC(loginCidadaoED.getTxtEmailLC()));
                        pessoaVO.setListaTelefones(this.montaTelefoneVOs(loginCidadaoED.getTxtTelef
                        usuarioLCLogadoMB.setPessoa(pessoaVO);
                        this.setUrl(URL_CONFIRMACAO);

                    }

dtwSessionScoped.setTipoPessoa(TipoPessoaEN.FISICA);
                    dtwSessionScoped.setNroIP(request.getRemoteAddr());
                    //Salva na tabela de Logs
                    logRN.inclui(pessoaVO, TipoLogEN.CENTRAL_SERVICOS_LOGIN_CIDADAO);

                } else {

                    this.setUrl(URL_LOGIN);

                }

            } else {

                this.setUrl(URL_LOGIN);

            }

        } catch (AuthenticationException e) {
            logger.error("AuthenticationException - e.getMessage(): " + e.getMessage() + ": " + e);
            this.setUrl(URL_LOGIN);
            // FacesContext context = FacesContext.getCurrentInstance();
            // context.getExternalContext().getFlash().setKeepMessages(true);
            // FacesUtil.addErrorMessage("A central de serviços está temporariamente indisponível,

        } catch (Exception e) {
            logger.error("Exception - e.getMessage(): " + e.getMessage() + ": " + e);
```

```
242             this.setUrl(URL_LOGIN);
243             // FacesContext context = FacesContext.getCurrentInstance();
244             // context.getExternalContext().getFlash().setKeepMessages(true);
245             // FacesUtil.addErrorMessage("A central de serviços está temporariamente indisponível,
246 
247         }
248 
249         return "pretty:" + this.url;
250 
251     }
252 
253 /*
254  *
255  *
256  */
257     private PessoaVO consultarPessoaRpd(LoginCidadaoED loginCidadaoED) throws Exception {
258 
259 try {
260 
261             return rpdRestClient.consultarPessoa(loginCidadaoED.getTxtCpfLC(), loginCidadaoED.getNr
262 
263         } catch (Exception e) {
264             logger.warn("consultarPessoaRpd(): " + e.getMessage() + " loginCidadaoED.getNroIdLC():
265             throw e;
266 
267         }
268     }
269 
270 /*
271  *
272  *
273  */
274     private AuthorizationCode obtemAutorizationCode(HttpServletRequest request) throws Authenticati
275 
276 AuthorizationResponse authResponse = null;
277 
278 try {
279 
280             authResponse = AuthorizationResponse.parse(new URI(request.getRequestURL() + "?" + requ
281 
282             if( authResponse instanceof AuthorizationErrorResponse ) {
283 
284                 AuthorizationErrorResponse authorizationErrorResponse = (AuthorizationErrorResponse
285                 logger.warn("obtemAutorizationCode() authorizationErrorResponse.getErrorObject().to
286                 throw new AuthenticationException(authorizationErrorResponse.getErrorObject().getCo
287 
288             } else {
289 
290 AuthorizationSuccessResponse authorizationSuccessResponse = (AuthorizationSuccessResponse) authRes
291                 logger.info("obtemAutorizationCode() authorizationSuccessResponse.getAuthorizationC
292 
293 if( !authorizationSuccessResponse.indicatesSuccess() ) {
294 
295                     logger.warn("obtemAutorizationCode() authorizationSuccessResponse.indicatesSucc
296                     throw new AuthenticationException("obtemAutorizationCode() authorizationSuccess
297 
298                 }
299 
300             }
301 
302         } catch (ParseException | URISyntaxException e) {
303             logger.warn("obtemAutorizationCode(): " + e.getMessage(), e);
304             throw new AuthenticationException(e.getMessage(), e);
305 
306         } catch (Exception e) {
307             logger.warn("obtemAutorizationCode(): " + e.getMessage(), e);
308             throw new AuthenticationException(e.getMessage(), e);
309 
310         }
```

```
311
312         // throw new AuthenticationException("obtemAutorizationCode(): TESTE 1...");
313         return ((AuthorizationSuccessResponse) authResponse).getAuthorizationCode();
314
315     }
316
317 /*
318  *
319  *
320  */
321     private OIDCTokenResponse obtemToken(AuthorizationCode code, HttpServletRequest request) throws
322
323 TokenResponse response = null;
324
325 try {
326
327             ClientAuthentication clientAuth = new ClientSecretBasic(new ClientID(dtwApplicationScop
328             AuthorizationGrant codeGrant = new AuthorizationCodeGrant(code, new URI(SistemaUtil.get
329             TokenRequest tokenRequest = new TokenRequest(new URI(dtwApplicationScoped.getLOGINCIDAD
330             response = OIDCTokenResponseParser.parse(tokenRequest.toHTTPRequest().send());
331
332         } catch (ParseException | IOException | URISyntaxException e) {
333             logger.warn("obtemToken(): " + e.getMessage(), e);
334             throw new AuthenticationException(e.getMessage(), e);
335
336         } catch (Exception e) {
337             logger.warn("obtemToken(): " + e.getMessage(), e);
338             throw new AuthenticationException(e.getMessage(), e);
339
340         }
341
342         if( response == null ) {
343
344             logger.warn("obtemToken(): " + "obtemToken() - response == null");
345             throw new AuthenticationException("obtemToken() - response == null");
346
347         } else if( !response.indicatesSuccess() ) {
348
349             logger.warn("obtemToken(): " + "obtemToken() - response.indicatesSuccess() == false");
350             throw new AuthenticationException("obtemToken() - response.indicatesSuccess() == false'
351
352         } else if( !(response instanceof OIDCTokenResponse) ) {
353
354 logger.warn("obtemToken(): " + "obtemToken() - response instanceof OIDCTokenResponse == false");
355             throw new AuthenticationException("obtemToken() - response instanceof OIDCTokenResponse
356
357         }
358
359 // throw new AuthenticationException("obtemToken(): TESTE 2...");
360         return (OIDCTokenResponse) response;
361
362     }
363
364 /*
365  *
366  *
367  */
368     private LoginCidadaoED obtemDadosUsuario(OIDCTokenResponse successResponse) throws Authenticati
369
370 LoginCidadaoED loginCidadaoED = new LoginCidadaoED();
371
372 try {
373
374             UserInfoRequest userInfoReq = new UserInfoRequest(new URI(dtwApplicationScoped.getLOGIN
375             HTTPResponse userInfoHTTPResp = userInfoReq.toHTTPRequest().send();
376             JSONObject userInfo = userInfoHTTPResp.getContentAsJSONObject();
377             // Long id = Long.valueOf(((String)userInfo.get("id")));
378             String id = ((String) userInfo.get("id"));
379             String nome = ((String) userInfo.get("full_name"));
```

```
            String email = ((String) userInfo.get("email"));
            String cpf = ((String) userInfo.get("cpf"));
            String strDataNascimento = ((String) userInfo.get("birthdate"));
            // "birthdate":"1986-07-07T00:00:00-0300",
            DateFormat df = new SimpleDateFormat("yyyy-MM-dd");
            Date dataNascimento = null;
            Calendar cal = null;

if( strDataNascimento != null ) {

                dataNascimento = df.parse(strDataNascimento.substring(0, 10));
                cal = Calendar.getInstance();
                cal.setTime(dataNascimento);

            }

            String phoneNumber = ((String) userInfo.get("phone_number"));
            String fotoPerfil = ((String) userInfo.get("picture"));
            loginCidadaoED.setNroIdLC(id);
            loginCidadaoED.setTxtNomeLC(nome);
            loginCidadaoED.setTxtEmailLC(email);
            loginCidadaoED.setTxtCpfLC(cpf);
            loginCidadaoED.setTxtTelefoneLC(phoneNumber);
            loginCidadaoED.setDthNascimento(cal);
            loginCidadaoED.setTxtFotoPerfilLC(fotoPerfil);
            loginCidadaoED.setTokenResponse(successResponse);

        } catch (IOException | URISyntaxException | ParseException | java.text.ParseException e) {
            logger.warn("obtemDadosUsuario() : " + e.getMessage(), e);
            throw new AuthenticationException("obtemDadosUsuario() : " + e.getMessage(), e);

        } catch (Exception e) {
            logger.warn("obtemDadosUsuario() : " + e.getMessage(), e);
            throw new AuthenticationException("obtemDadosUsuario() : " + e.getMessage(), e);

        }

        // throw new AuthenticationException("obtemDadosUsuario(): TESTE 3...");
        return loginCidadaoED;

    }

/*
 *
 *
 */
    private boolean validaOIDCTokenResponse(OIDCTokenResponse token) throws AuthenticationExceptior

try {

if( !emailVerificado(token) )
                return false;

        } catch (ParseException | IOException | URISyntaxException e) {
            logger.warn("validaOIDCTokenResponse(): " + e.getMessage(), e);
            throw new AuthenticationException(e.getMessage(), e);

        } catch (Exception e) {
            logger.warn("validaOIDCTokenResponse(): " + e.getMessage(), e);
            throw new AuthenticationException(e.getMessage(), e);

        }

        // throw new AuthenticationException("validaOIDCTokenResponse(): TESTE 4...");
        return true;

    }

/*
```

```
 *
 *
 */
   private Boolean emailVerificado(OIDCTokenResponse successResponse) throws IOException, ParseExc

UserInfoRequest userInfoReq = new UserInfoRequest(new URI(dtwApplicationScoped.getLOGINCIDADAO_END
       HTTPResponse userInfoHTTPResp = userInfoReq.toHTTPRequest().send();
       JSONObject userInfo = userInfoHTTPResp.getContentAsJSONObject();
       return (Boolean) userInfo.get("email_verified");

   }

/*
 *
 *
 */
   private boolean validaLoginCidadaoED(LoginCidadaoED loginCidadaoED) throws AuthenticationExcept

try {

if( StringUtils.isBlank(loginCidadaoED.getTxtCpfLC()) ) {

               return false;

           } else if( StringUtils.isBlank(loginCidadaoED.getTxtNomeLC()) ) {

               return false;

           } else if( StringUtils.isBlank(loginCidadaoED.getTxtEmailLC()) ) {

return false;

           } else if( StringUtils.isBlank(loginCidadaoED.getTxtTelefoneLC()) ) {

return false;

           }

       } catch (Exception e) {
           logger.warn("validaLoginCidadaoED(): " + e.getMessage(), e);
           throw new AuthenticationException(e.getMessage(), e);

       }

       // throw new AuthenticationException("validaLoginCidadaoED(): TESTE 5...");
       return true;

   }

/*
 *
 */
   private List<TelefoneVO> montaTelefoneVOs(String fone) throws Exception {

Locale locale = Locale.getDefault();
       PhoneNumber phoneNumber = null;
       phoneNumber = PhoneNumberUtil.getInstance().parse(fone, locale.getCountry());

if( PhoneNumberUtil.getInstance().isValidNumberForRegion(phoneNumber, locale.getCountry()) ) {

           TelefoneVO telefoneVO = new TelefoneVO();
           telefoneVO.setDdd(getDDD(phoneNumber));
           telefoneVO.setNumero(getNumero(phoneNumber));
           telefoneVO.setTipoTelefone(Byte.valueOf("1"));
           List listaTelefone = new ArrayList();
           listaTelefone.add(telefoneVO);
           return listaTelefone;

       } else {
```

```
518
519             logger.warn("montaTelefoneVOs(): " + "PhoneNumberUtil.getInstance().isValidNumberForReg
520             throw new Exception("PhoneNumberUtil.getInstance().isValidNumberForRegion() == false");
521
522         }
523
524     }
525
526 /*
527  *
528  *
529  */
530     private String getDDD(PhoneNumber phoneNumber) {
531
532 return String.valueOf(phoneNumber.getNationalNumber()).substring(0, PhoneNumberUtil.getInstance().
533
534     }
535
536 /*
537  *
538  *
539  */
540     private String getNumero(PhoneNumber phoneNumber) {
541
542 return String.valueOf(phoneNumber.getNationalNumber()).substring(PhoneNumberUtil.getInstance().get
543
544     }
545
546 /*
547  *
548  *
549  */
550     private List<EmailVO> parseEmailLC(String email) {
551
552 EmailVO emailVO = new EmailVO();
553         emailVO.setEndereco(email);
554         List listaEmail = new ArrayList();
555         listaEmail.add(emailVO);
556         return listaEmail;
557
558     }
559
560     public String getUrl() {
561
562 return url;
563
564     }
565
566     public void setUrl(String url) {
567
568 this.url = url;
569
570     }
571
572 /*
573  *
574  *
575  */
576     // private String montaURLAuthenticationException(AuthenticationException e) {
577     // /**
578     // OAuth2Error.INVALID_REQUEST;
579     // OAuth2Error.UNAUTHORIZED_CLIENT;
580     // OAuth2Error.ACCESS_DENIED;
581     // OAuth2Error.UNSUPPORTED_RESPONSE_TYPE;
582     // OAuth2Error.INVALID_SCOPE;
583     // OAuth2Error.SERVER_ERROR;
584     // OAuth2Error.TEMPORARILY_UNAVAILABLE;
585     // */
586     // if (e.getMessage().equalsIgnoreCase("ACCESS_DENIED"))
```

```
587     // return URL_ERRO_FALHA_AUT;
588     // return URL_ERRO_FALHA_AUT;
589     // }
590     // private String encoderMsgErro(String msg) {
591     // try {
592     // return URLEncoder.encode(msg, "UTF-8");
593     // } catch (UnsupportedEncodingException e) {
594     // e.printStackTrace();
595     // }
596     // return null;
597     // }
598 }
```