

Guia

Loja Exemplo

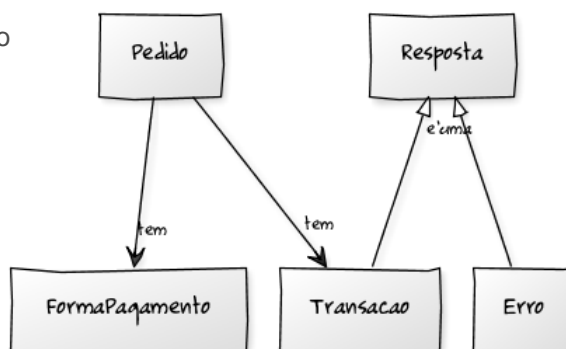
A loja exemplo tem o objetivo de ser simples e didática ao mostrar a integração entre a loja virtual e a Cielo. Ela está implementada na plataforma Java com o emprego apenas de objetos e páginas JSP.

Essas páginas servem **apenas para simular** as principais telas de uma loja virtual: o carrinho de compras, a página de espera e a de finalização do pedido. Adicionalmente, há outras telas para facilitar o entendimento das funções abaixo:

- Autorização
- Captura
- Cancelamento
- Consulta
- Geração de Token

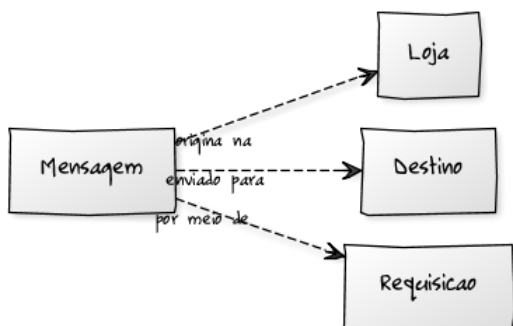
Estrutura

Os principais conceitos e sua implementação estão representados nas classes Java. Veja: um pedido possui uma forma de pagamento (crédito/débito e parcelas) e uma transação.

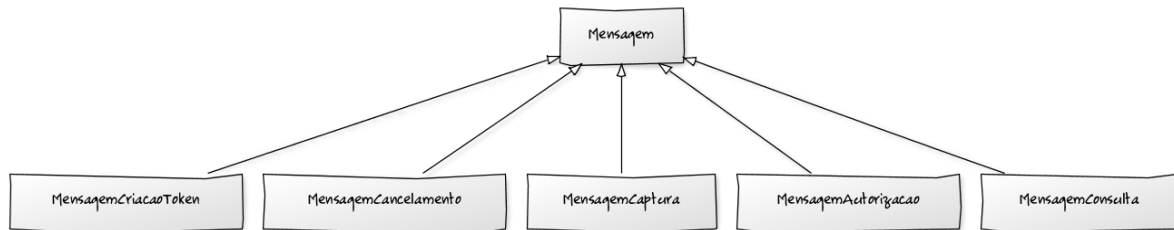


A integração entre loja virtual e Cielo é feita através de Web Services, utilizando-se mensagens de diferentes tipos que são transportadas via requisições HTTP. As mensagens são originadas no ambiente da loja e enviadas para um destino na Cielo.

Como resposta, é possível obter uma transação (caso esperado), um mensagem de retorno de Token ou um erro.



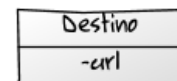
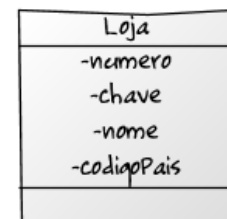
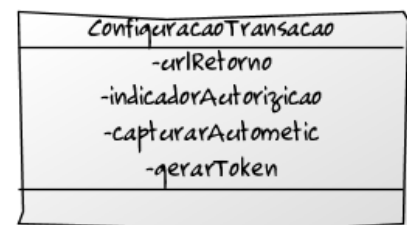
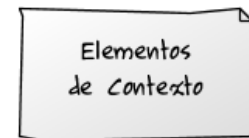
Cinco tipos de mensagens podem ser trafegados da loja virtual para a Cielo: um tipo para cada funcionalidade.



Essa integração é pautada por algumas configurações. Parâmetros que devem ser ajustados conforme a necessidade do lojista. Eles formam o contexto.

São classificados em

- Configuração da transação
- Dados da loja e
- Destino (ambiente de teste ou de produção)



Envio da mensagem

Todo envio segue o modelo abaixo:

```
public Transacao capturar(Transacao transacao, long valor) throws FalhaComunicacaoException {
    Mensagem mensagem = new MensagemCaptura(loja, transacao, valor);
    Requisicao requisicao = new Requisicao(mensagem);
    return requisicao.enviarPara(destino);
}
```

Foi empregado o componente Commons HttpClient (<http://hc.apache.org/httpclient-3.x/>) para o transporte da mensagem via HTTP. Para o tratamento do XML de retorno, o xStream (<http://xstream.codehaus.org/>). Não está no escopo deste guia qualquer explicação quanto a maneira de se realizar requisição HTTP ou interpretação de XML.

```
public Transacao enviarPara(Destino destino) throws FalhaComunicacaoException {
    String mensagemXml = mensagem.toXml();

    PostMethod httpMethod = new PostMethod(destino.getUrl());
    httpMethod.addParameter("mensagem", mensagemXml);

    if (logger.isDebugEnabled()) {
        logger.debug("Destino: " + destino.getUrl() + "\nMensagem: \n" + mensagemXml);
    }
    try {
        Stopwatch stopwatch = new Stopwatch();
        stopwatch.start();

        httpClient.executeMethod(httpMethod);

        String respostaXml = httpMethod.getResponseBodyAsString();

        stopwatch.stop();

        RegistroTempoProcessamento.registrar(stopwatch);

        if (logger.isDebugEnabled()) {
            logger.debug("Retorno [em " + stopwatch + ", id='" + mensagem.getId() + "': \n" + respostaXml);
        }

        Resposta resposta = RespostaFactory.getInstance().criar(respostaXml);

        if (resposta.getId() != null && !mensagem.getId().equals(resposta.getId())) {
            throw new IllegalArgumentException("Resposta inválida: idRecebido=" +
                resposta.getId() + ", idEnviado=" + mensagem.getId() + ".");
        }

        if (resposta instanceof Erro) {
            Erro erro = (Erro) resposta;
            throw new RequisicaoInvalidaException(erro);
        }

        Transacao transacao = (Transacao) resposta;

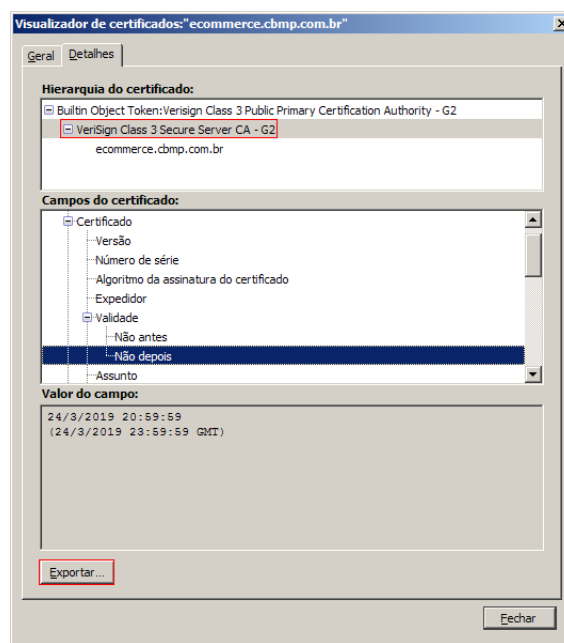
        return transacao;
    }
    catch (HttpException e) {
        logger.error(e, e);
        throw new FalhaComunicacaoException(e.getMessage());
    }
    catch (IOException e) {
        logger.error(e, e);
        throw new FalhaComunicacaoException(e.getMessage());
    }
    finally {
        httpMethod.releaseConnection();
    }
}
```

Nota: Caso esteja ocorrendo uma exceção como a abaixo (no envio da requisição HTTP)

```
javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX  
path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable  
to find valid certification path to requested target
```

é necessário registrar o certificado Intermediário na *Trust Store* utilizada. Os seguintes passos devem ser executados:

- 1) Exportar o Certificado Intermediário (pode ser feito através de um browser qualquer) e,



- 2) Importá-lo na Trust Store (usar a ferramenta Keytool). Exemplo:

```
C:\Arquivos de programas\Java\jdk1.6.0_02\jre\lib\security>keytool -keystore  
cacerts -import -alias verisignclass3g2caAfterMay19-2009 -file  
verisignclass3g2caAfterMay19-2009.der -trustcacerts  
Enter keystore password: changeit  
Certificate was added to keystore
```

Site

Para a parte navegacional fez-se uso de algumas liberdades de implementação, como armazenamento de informações na sessão web e utilização de scriptlets. **Não recomendamos essas abordagens. Apenas foram empregadas com o objetivo de simplificação.**

Duas partes merecem maior atenção:

- O passo inicial para criação da transação e redirecionamento, e;
- O fechamento do pedido após o retorno do fluxo à loja.

Na primeira, os dados da página do carrinho de compras são submetidos para a `novoPedidoAguarde.jsp` que apenas cria um novo pedido, armazena-o na sessão e retorna a mensagem “Redirecionando...”.

Por um código Javascript o fluxo é encaminhado para `novoPedido.jsp` que de fato requisita uma transação à Cielo. A resposta é tratada e o browser redirecionado para o ambiente da Cielo.

```
<%@page import="br.com.cbmp.ecommerce.util.web.WebUtils"%>
<%@page import="br.com.cbmp.ecommerce.pedido.Pedido"%>
<%@page import="br.com.cbmp.ecommerce.resposta.Transacao"%>
<%@ page errorPage="novoPedidoErro.jsp" %>

<%
    // solicita criação da transação
    Pedido pedido = new WebUtils(request).recuperarUltimoPedido();
    Transacao transacao = pedido.criarTransacao();

    // obtém URL de redirecionamento
    String urlRedirecionamento;
    urlRedirecionamento = transacao.getUrlAutenticacao();

    if(urlRedirecionamento == null){
        urlRedirecionamento = new WebUtils(request).getUrlRetorno();
    }

    // redireciona o fluxo para a Cielo
    System.out.print("urlRedirecionamento: " + urlRedirecionamento);

    response.sendRedirect(urlRedirecionamento);
%>
```

Transação com Cartão


Cielo E-commerce

Página • Segurança • Ferramentas


cielo

LoGo

Prossiga sua compra com tranquilidade, você está em um ambiente seguro Cielo.



Resumo da compra



Loja: Buy Page Cielo
Pedido: 1595680324
Data: 05/10/2012 18:36:44
Valor: R\$ 10,00
Pago com: Crédito A Visa

Dados do seu cartão

Complete as informações abaixo e clique em Confirmar compra.

Número do seu cartão:
 ?

Data de Validade:
Mês: Ano: ?

Código verificador (três dígitos no verso do seu cartão):
 ?

Confirmar compraCancelar

Transação com Celular

Após ter preenchido as informações de cartão, o fluxo é redirecionado ao banco emissor. A autenticação é realizada e a transação finalizada. Com o retorno do fluxo à loja virtual, na página especificada na criação (em nosso caso `retorno.jsp`), consulta-se a transação a fim de saber qual seu estado:

```
<%@page import="br.com.cbmp.ecommerce.pedido.StatusTransacao"%>
<%@page import="br.com.cbmp.ecommerce.resposta.Transacao"%>
<%@page import="br.com.cbmp.ecommerce.pedido.Pedido"%>
<%@page import="br.com.cbmp.ecommerce.util.web.WebUtils"%>
<%@page import="java.util.Date"%>
<%@ page errorPage="novoPedidoErro.jsp" %>
<%
    Pedido pedido = new WebUtils(request).recuperarUltimoPedido();
    boolean pedidoFinalizado = pedido.finalizar();
    Transacao transacao = pedido.getTransacao();%>
```

Lembre-se! A loja virtual tem papel ativo em dois passos muito importantes. No início para criação da transação e no fim, para consultar o resultado final.

Cielo - Loja Exemplo x

172.16.34.66:7001/lojaexemplo-2.1/carrinhoCartaoPagamento.jsp

Fechamento (Mon Feb 18 15:36:29 BRT 2013)

Número pedido	Finalizado com sucesso?	Transação	Status transação
240065029	sim	100699306908642B1001	Capturada

XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<transacao versao="1.3.0" id="733cd54e-b936-48f5-982f-914cf2271094"
xmlns="http://ecommerce.cbmp.com.br">
  <tid>100699306908642B1001</tid>
  <pan>uv9yI5tkhX9jpuCt+dftrtoSVM4U3gIjvrcwMBfZcadE=</pan>
  <dados-pedido>
    <numero>240065029</numero>
    <valor>1000</valor>
    <moeda>986</moeda>
    <data-hora>2013-02-18T15:36:24.238-03:00</data-hora>
    <descricao>[origem:172.16.34.66]</descricao>
    <idioma>PT</idioma>
    <taxa-embarque>0</taxa-embarque>
  </dados-pedido>
  <forma-pagamento>
    <bandeira>visa</bandeira>
    <produto>1</produto>
    <parcelas>1</parcelas>
  </forma-pagamento>
  <status>6</status>
  <autenticacao>
    <codigo>6</codigo>
    <mensagem>Transacao sem autenticacao</mensagem>
    <data-hora>2013-02-18T15:36:24.560-03:00</data-hora>
    <valor>1000</valor>
  </autenticacao>
</transacao>
```

[Menu](#)

[Pedidos](#)

Como executar?

No kit de integração é disponibilizado o código fonte da loja exemplo e também a aplicação pronta para instalação (`lojaexemplo.war`). Dessa forma há duas maneiras para rodar a aplicação:

- A primeira, se você tiver o Maven 2 e uma JDK que seja pelos menos a 1.5, é só executar no diretório web, o comando `mvn jetty:run` e acessar <http://localhost:8080/lojaexemplo-2.5.1>
- A outra opção é instalar a loja no seu servidor web (deve suportar a versão Servlet 2.5)