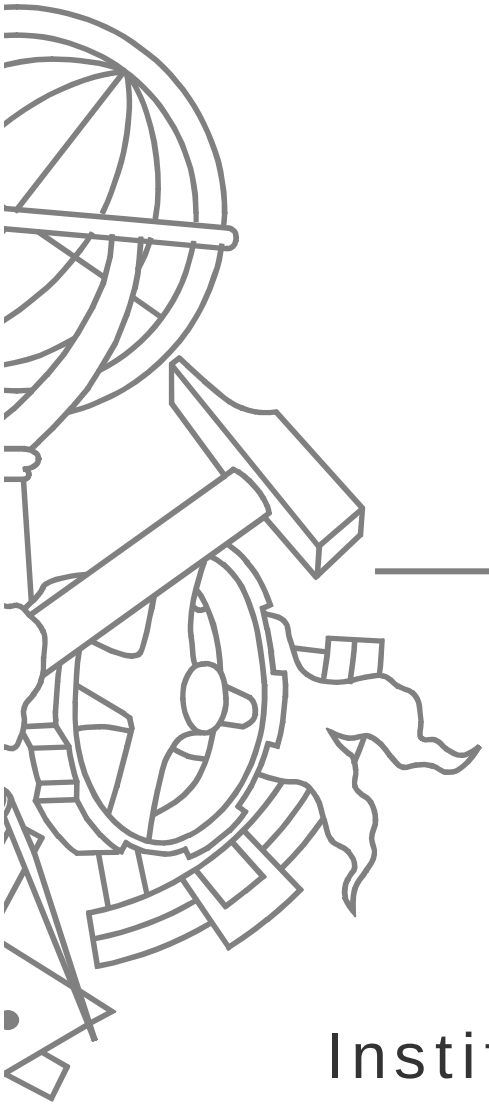


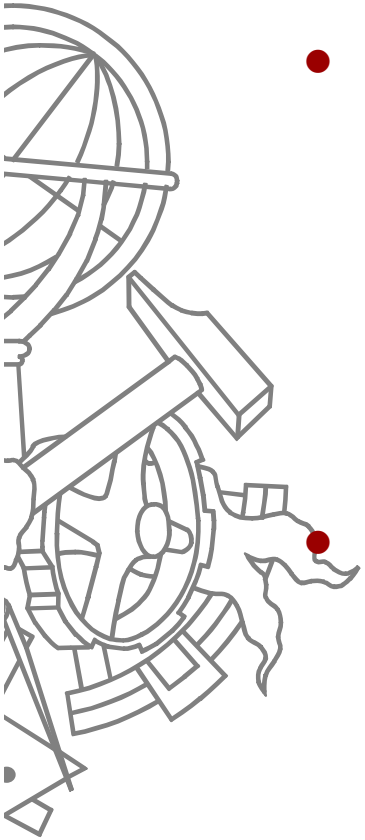
PetShop

Paulo Sousa

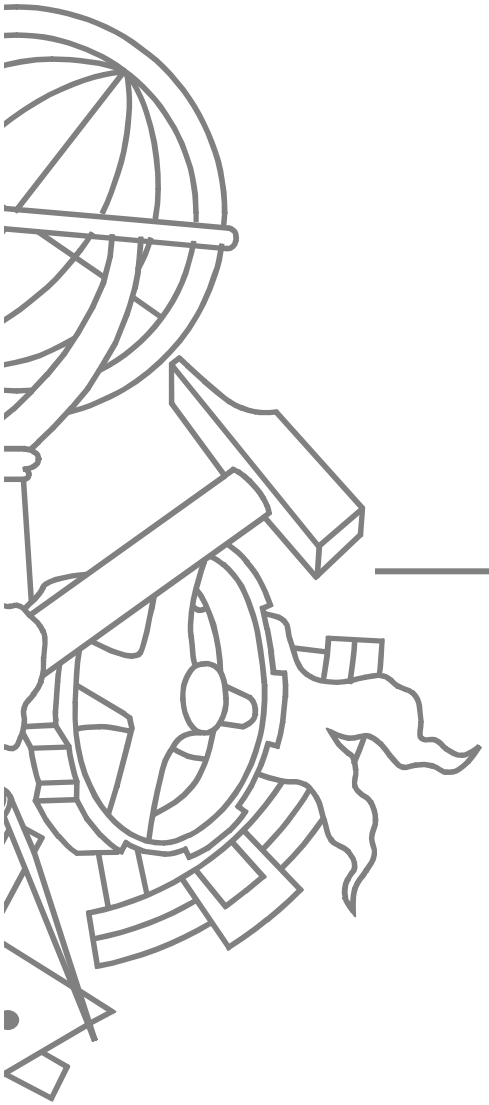
Engenharia da Informação

Instituto Superior de Engenharia do Porto

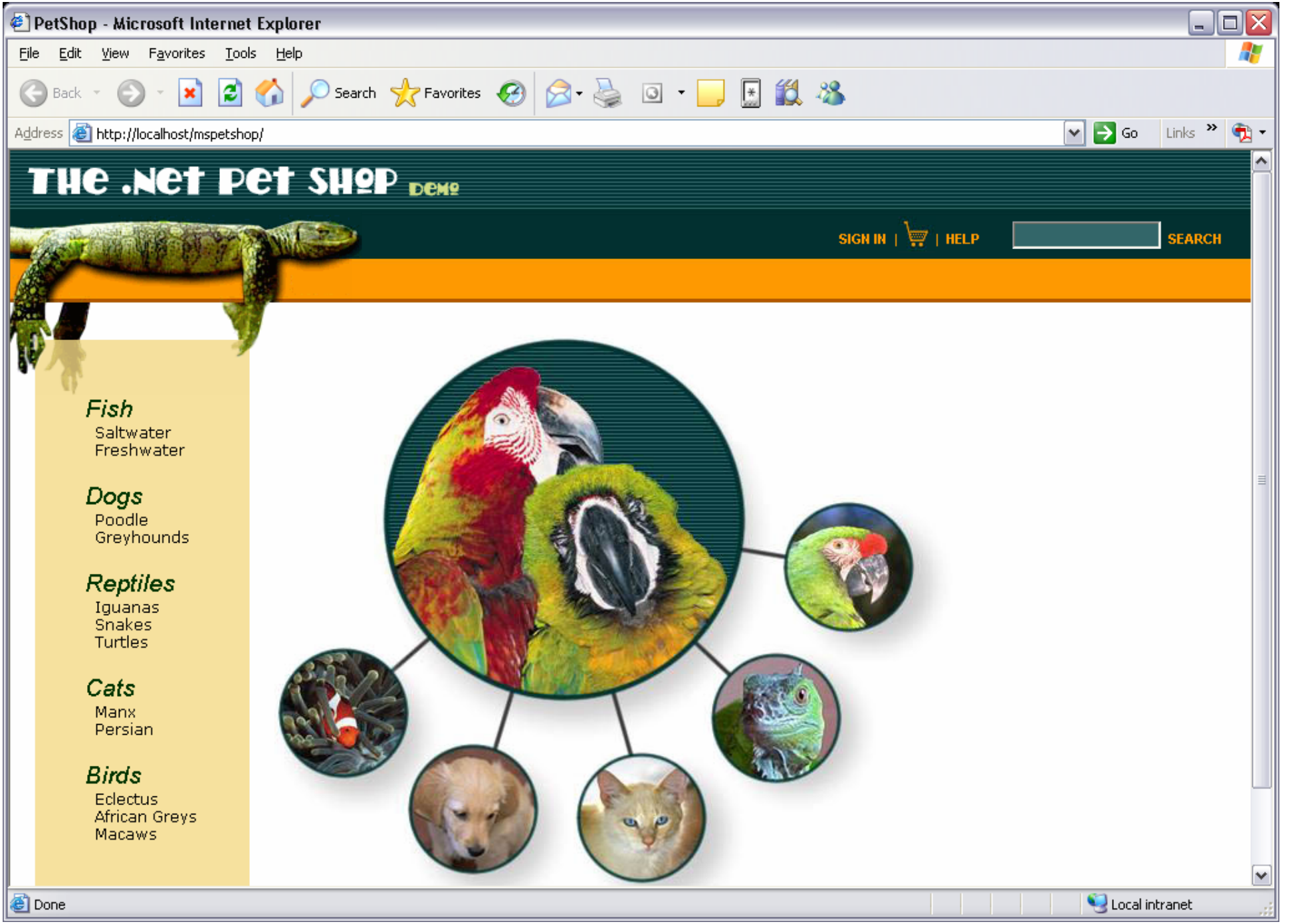
Onde obter

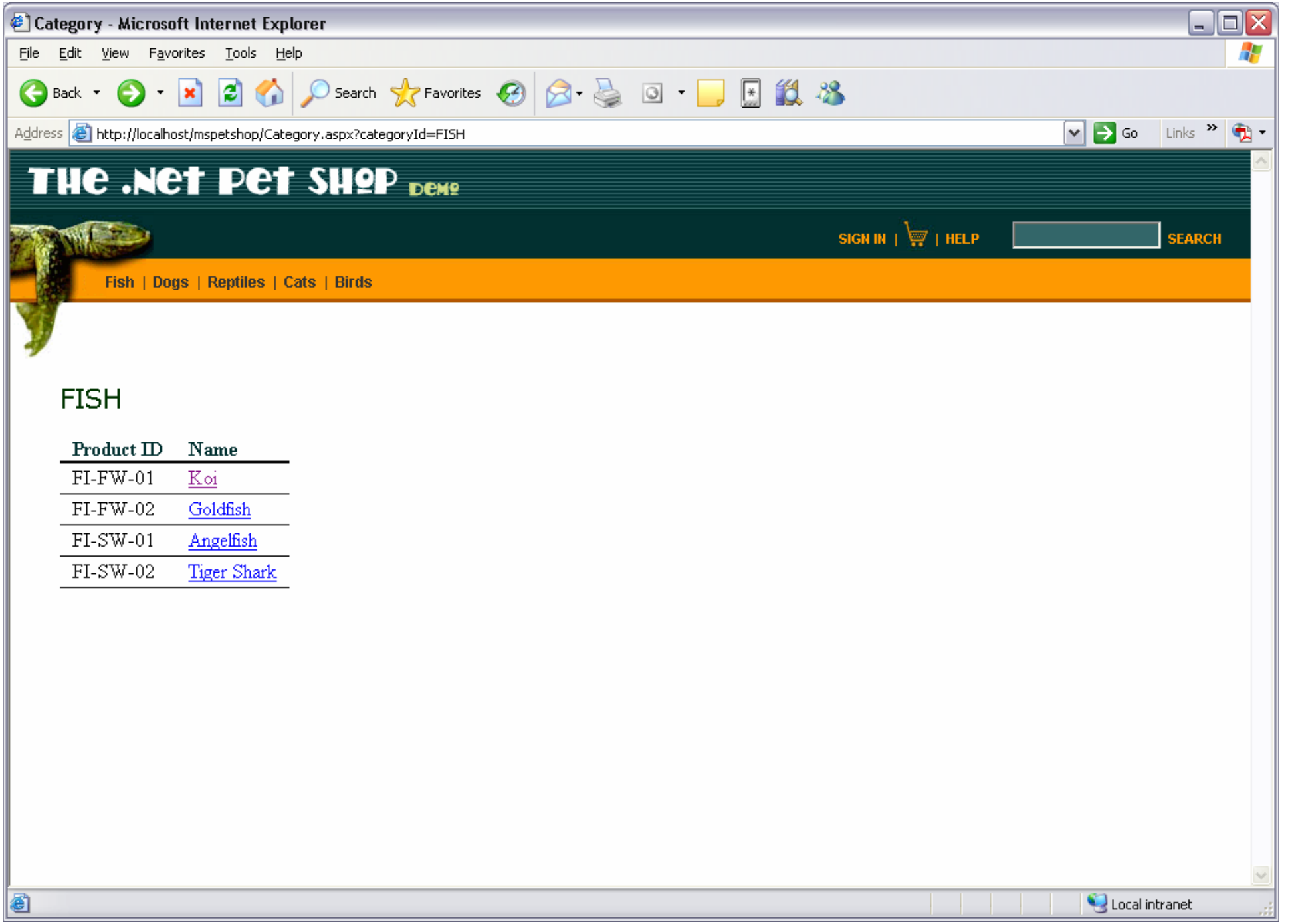


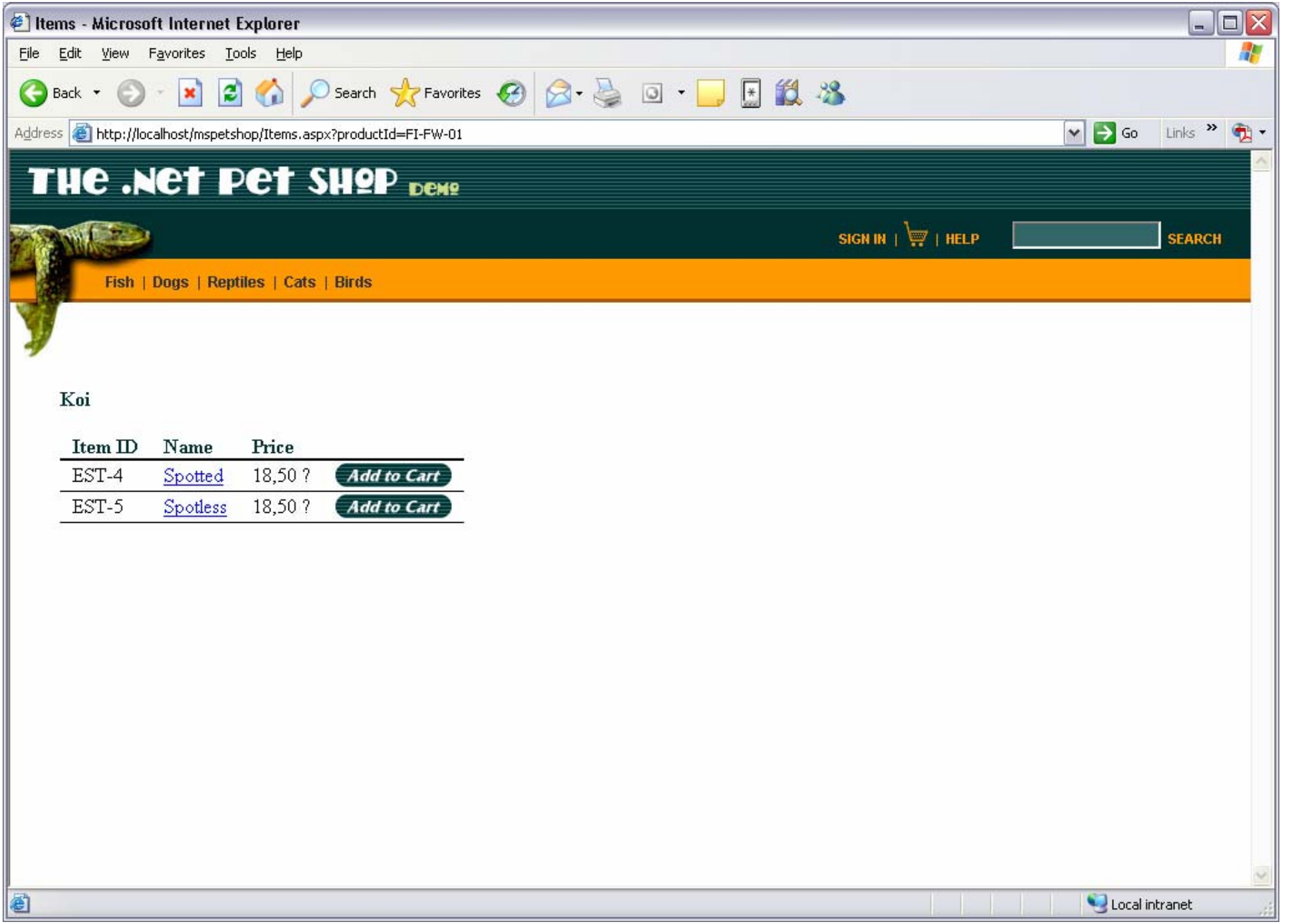
- Versão .net
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/PetShop3x.asp
  - http://www.microsoft.com/downloads/details.aspx?FamilyId=E2930625-3C7A-49DC-8655-A8205813D6DB&displaylang=en
- Versão Java
  - http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/app-arch/app-arch.html
  - http://java.sun.com/blueprints/code/index.html#java_pet_store_demo



PetShop .net


PetShop - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/
Go Links
THE .NET PET SHOP DEMO
SIGN IN | | HELP
SEARCH
Fish
Saltwater
Freshwater
Dogs
Poodle
Greyhounds
Reptiles
Iguanas
Snakes
Turtles
Cats
Manx
Persian
Birds
Eclectus
African Greys
Macaws
Done
Local intranet


Category - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/Category.aspx?categoryId=FISH
Go Links
THE .NET PET SHOP DEMO
SIGN IN | | HELP
SEARCH
Fish | Dogs | Reptiles | Cats | Birds
FISH
Product ID Name
FI-FW-01 Koi
FI-FW-02 Goldfish
FI-SW-01 Angelfish
FI-SW-02 Tiger Shark
Local intranet


Items - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/Items.aspx?productId=FI-FW-01
Go Links
THE .NET PET SHOP DEMO
SIGN IN | | HELP
SEARCH
Fish | Dogs | Reptiles | Cats | Birds
Koi
Item ID Name Price
EST-4 Spotted 18,50 ? Add to Cart
EST-5 Spotless 18,50 ? Add to Cart
Local intranet

Item Details - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites

Address http://localhost/mspetshop/ItemDetails.aspx?itemId=EST-4 Go Links

THE .NET PET SHOP DEMO

SIGN IN | | HELP SEARCH

Fish | Dogs | Reptiles | Cats | Birds

fish from Japan

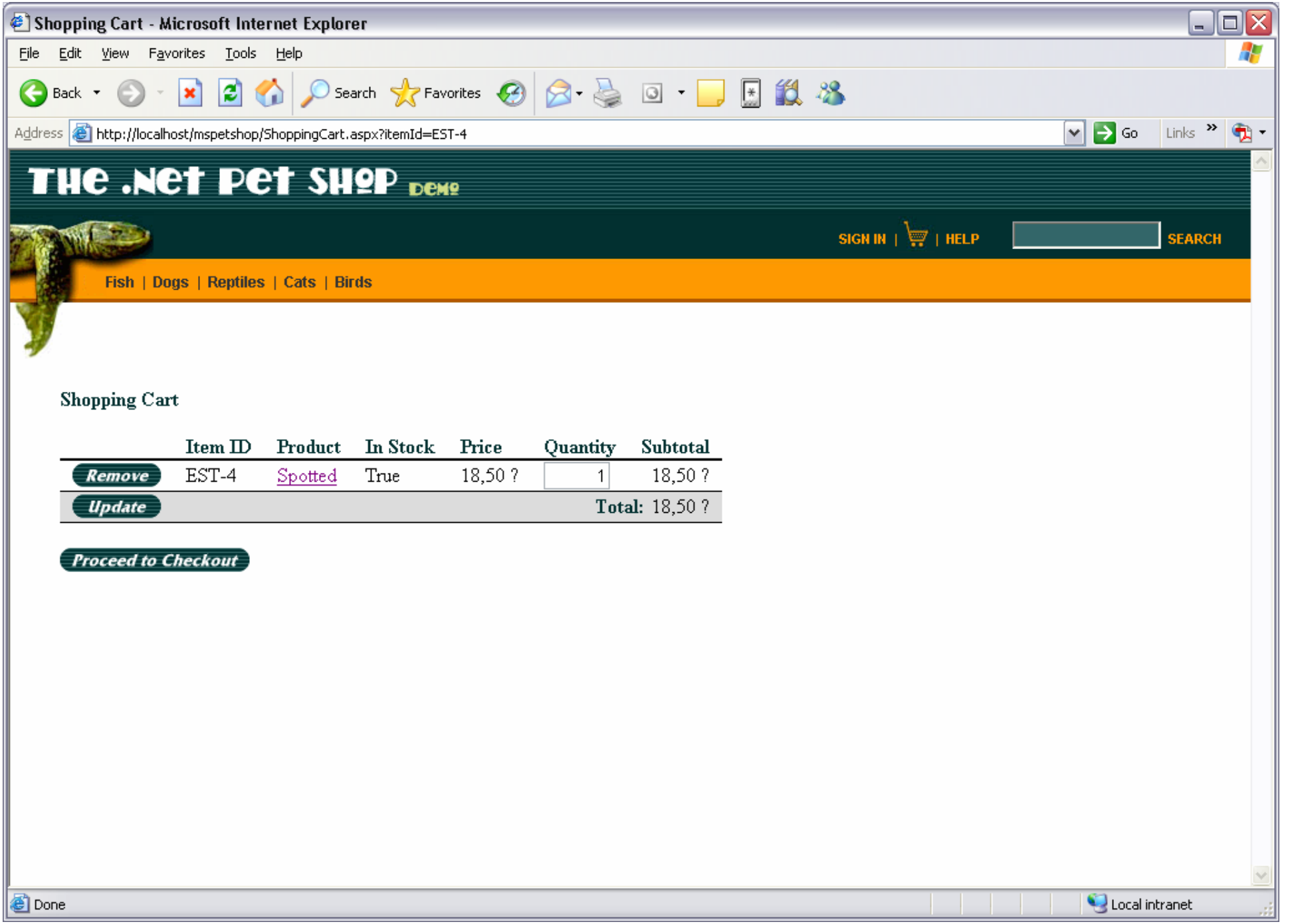
Spotted Koi

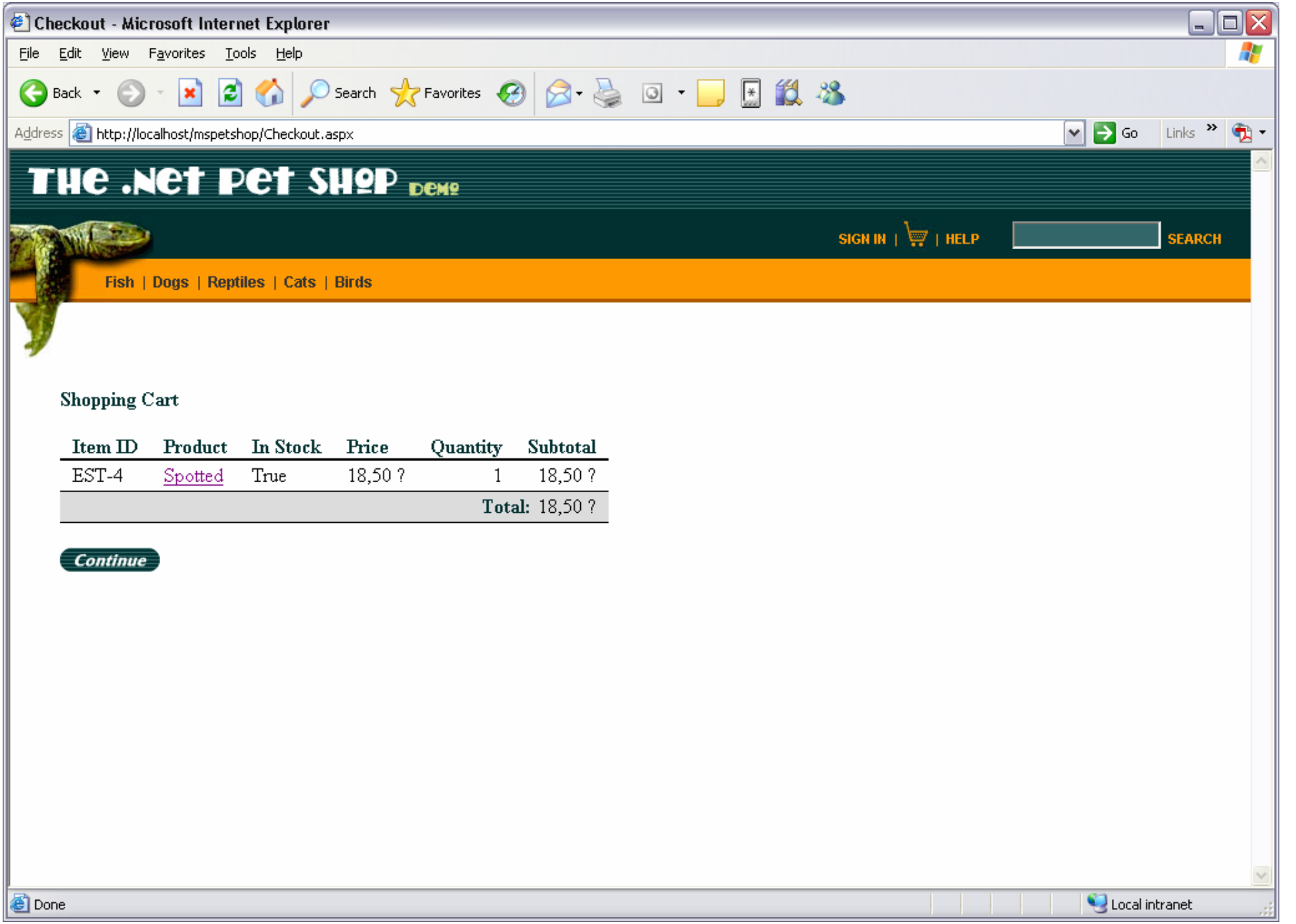
Freshwater

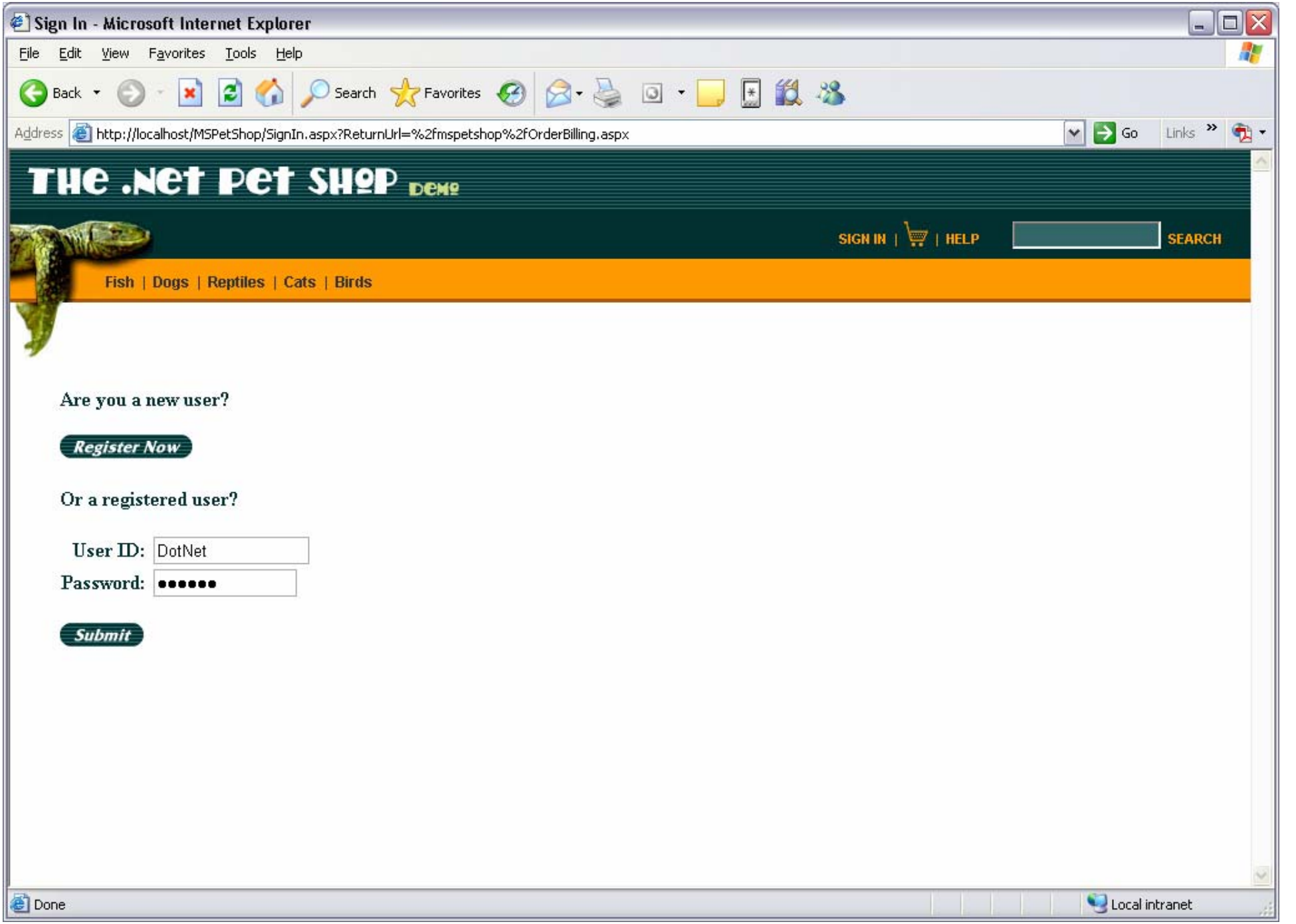
Price: 18,50 ?
Quantity: 10000

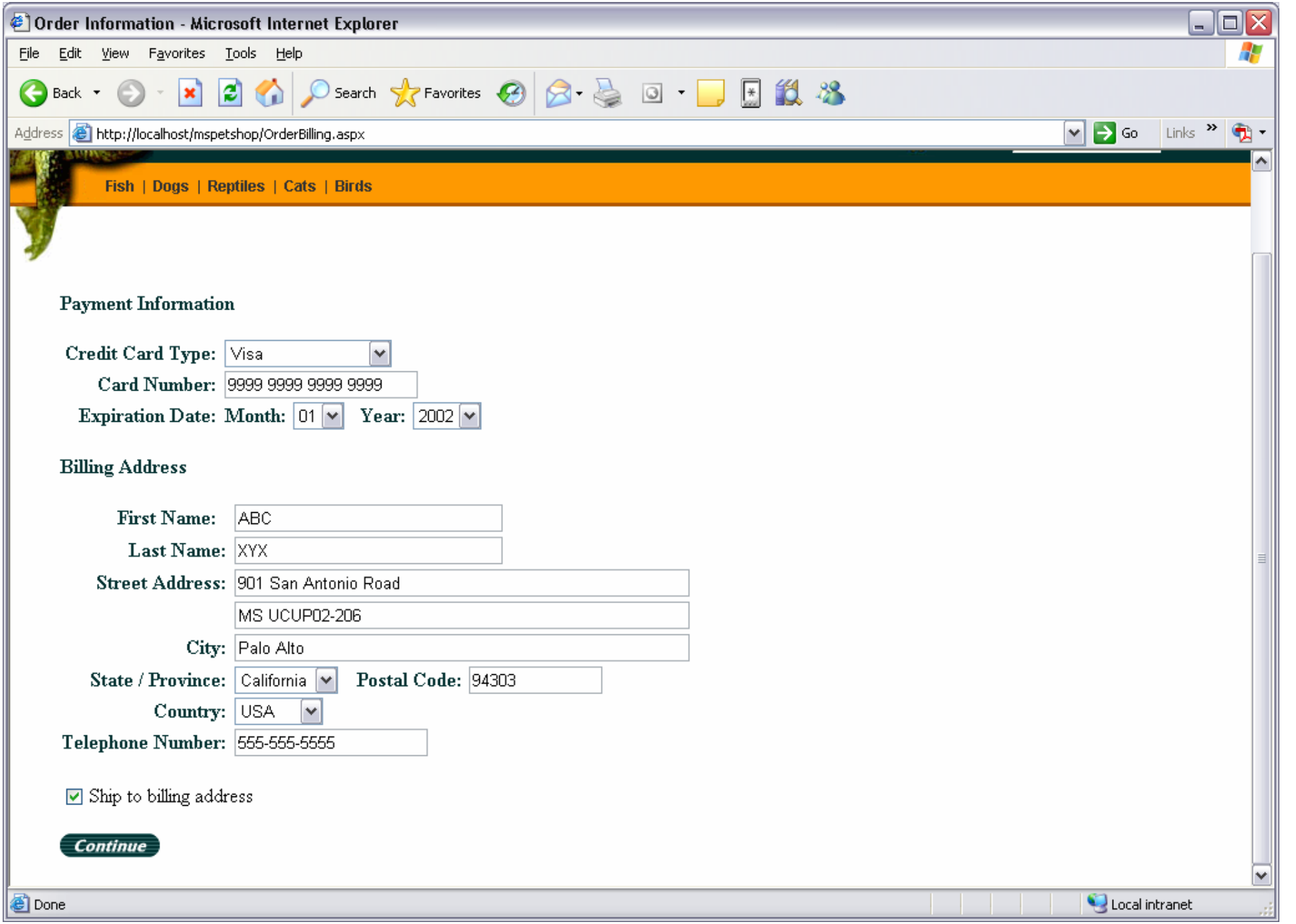
Add to Cart

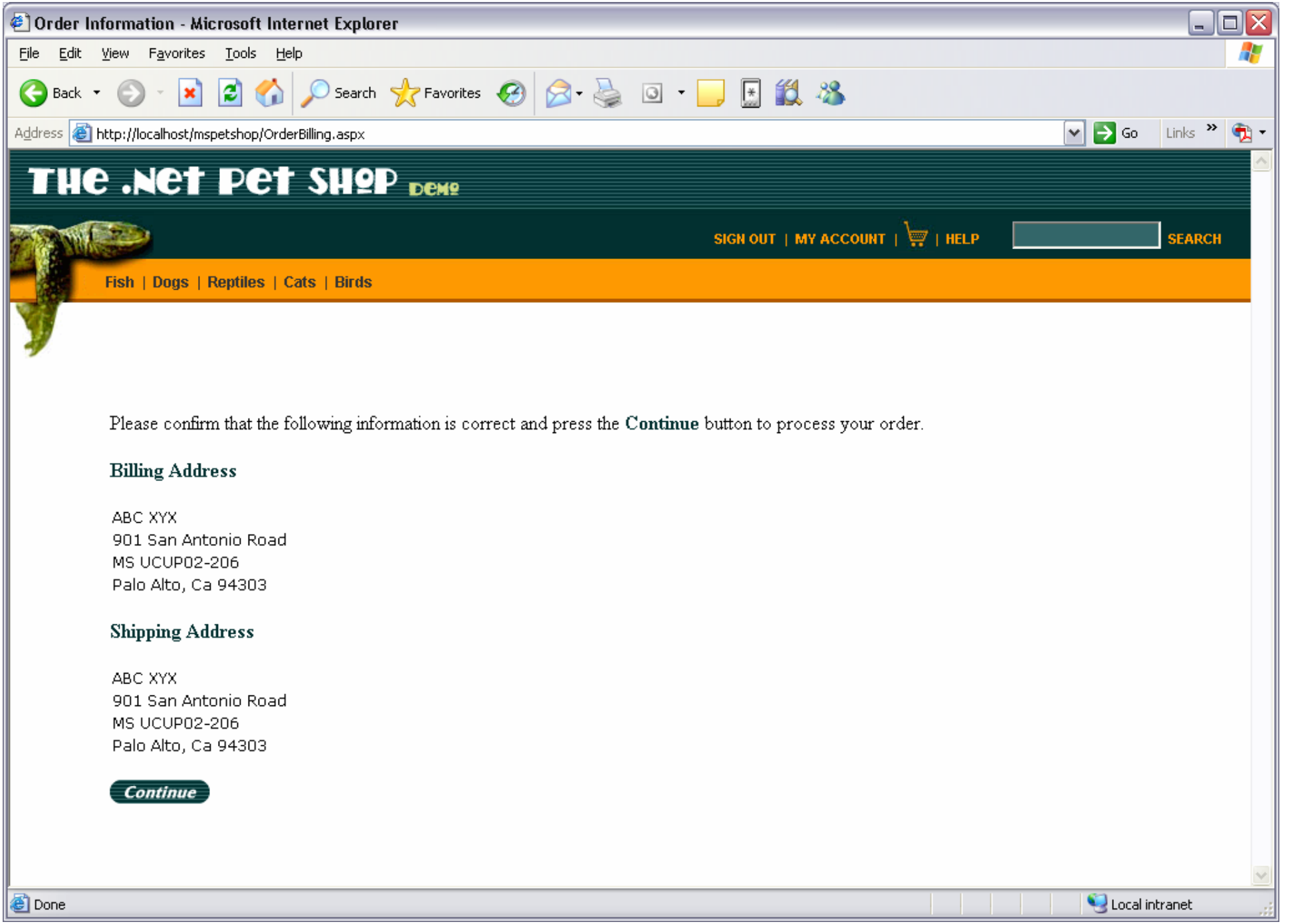
Done Local intranet

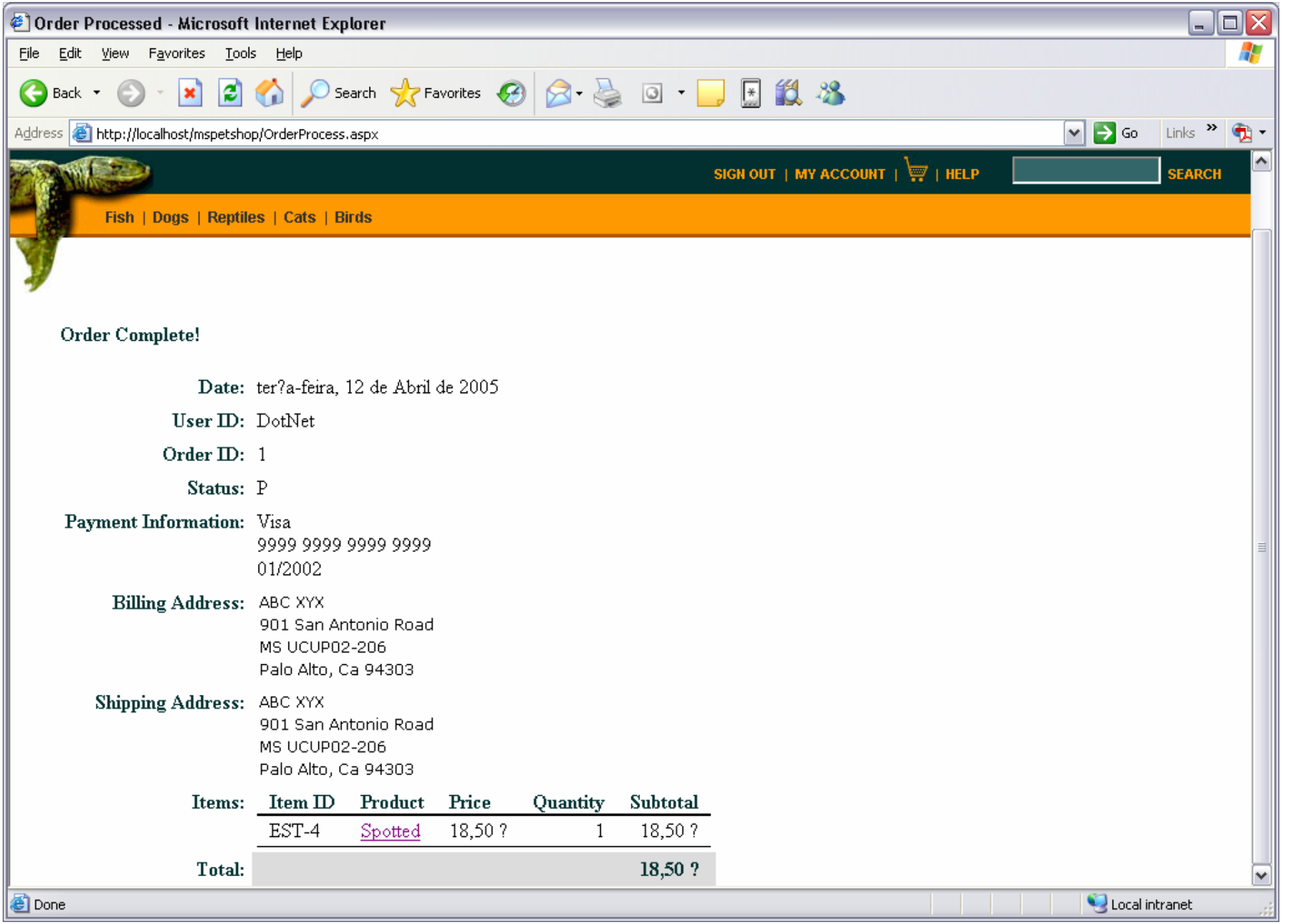

Shopping Cart - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/ShoppingCart.aspx?itemId=EST-4
Go Links
THE .NET PET SHOP DEMO
SIGN IN | | HELP
SEARCH
Fish | Dogs | Reptiles | Cats | Birds
Shopping Cart
Item ID Product In Stock Price Quantity Subtotal
Remove EST-4 Spotted True 18,50 ? 1 18,50 ?
Update Total: 18,50 ?
Proceed to Checkout
Done
Local intranet

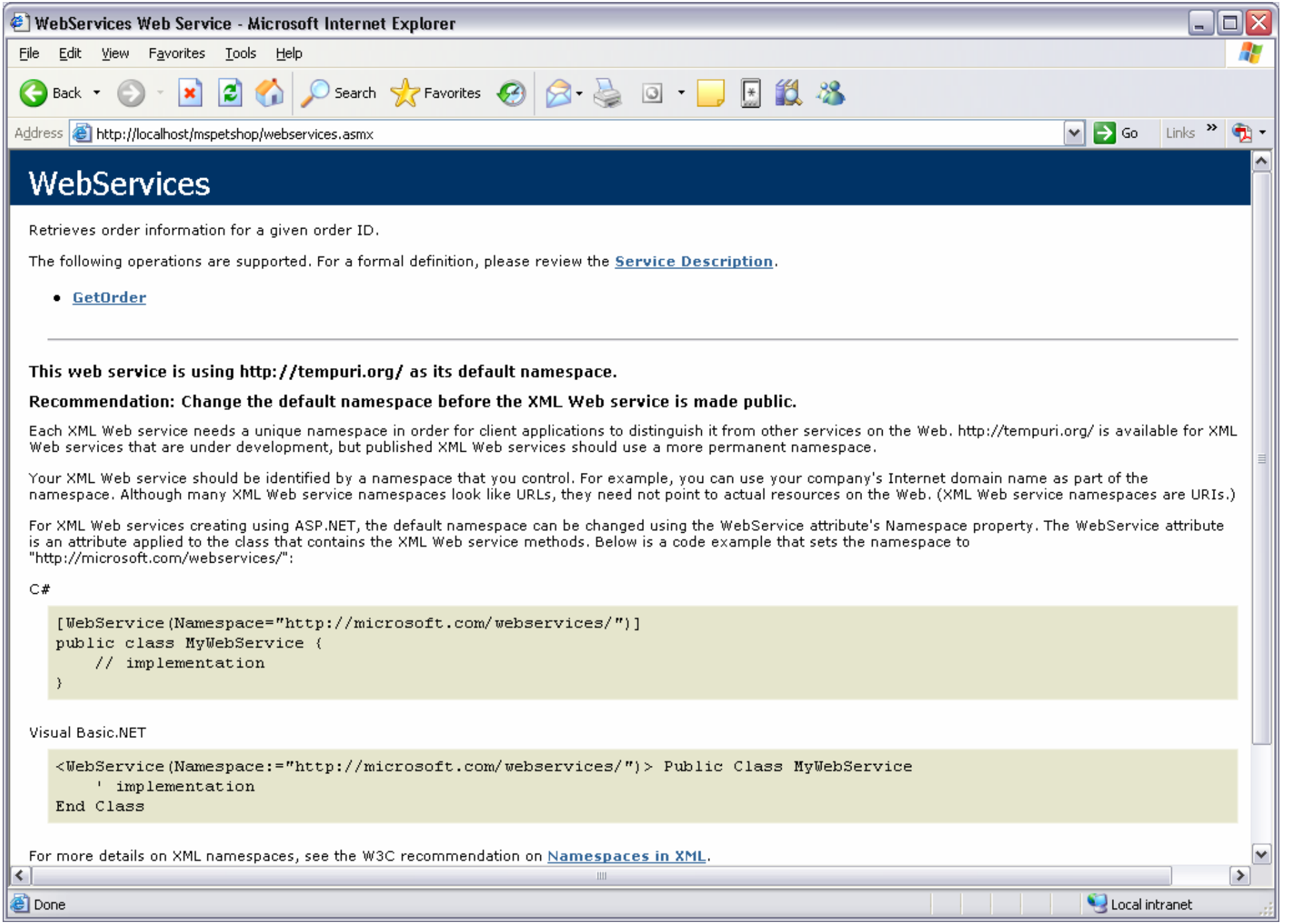

Checkout - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/Checkout.aspx
Go Links
THE .NET PET SHOP DEMO
SIGN IN | | HELP
SEARCH
Fish | Dogs | Reptiles | Cats | Birds
Shopping Cart
Item ID Product In Stock Price Quantity Subtotal
EST-4 Spotted True 18,50 ? 1 18,50 ?
Total: 18,50 ?
Continue
Done
Local intranet

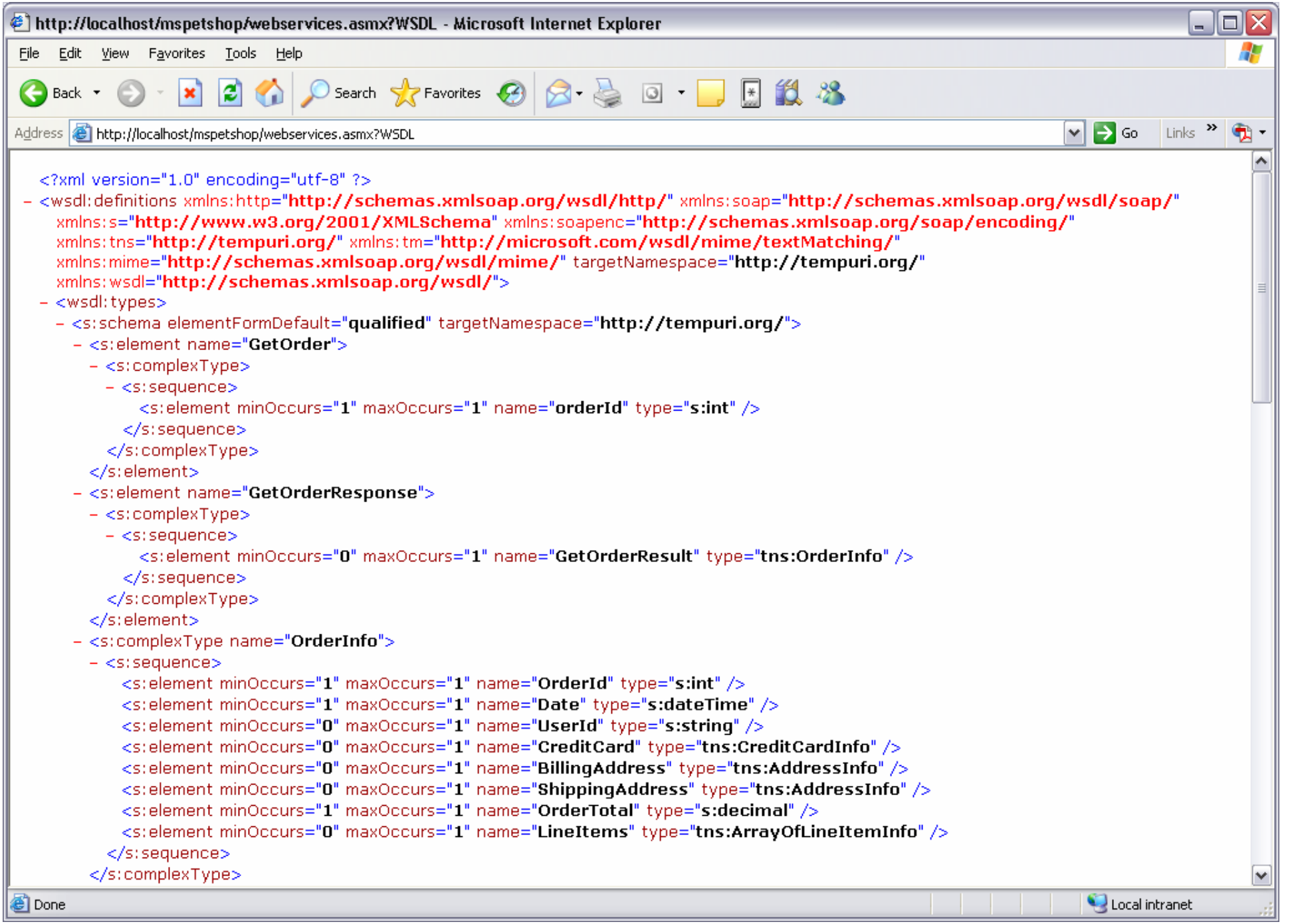

Sign In - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/MSPetShop/SignIn.aspx?ReturnUrl=%2fmspetshop%2fOrderBilling.aspx
Go Links
THE .NET PET SHOP DEMO
SIGN IN | | HELP
SEARCH
Fish | Dogs | Reptiles | Cats | Birds
Are you a new user?
Register Now
Or a registered user?
User ID: DotNet
Password: ••••••
Submit
Done
Local intranet

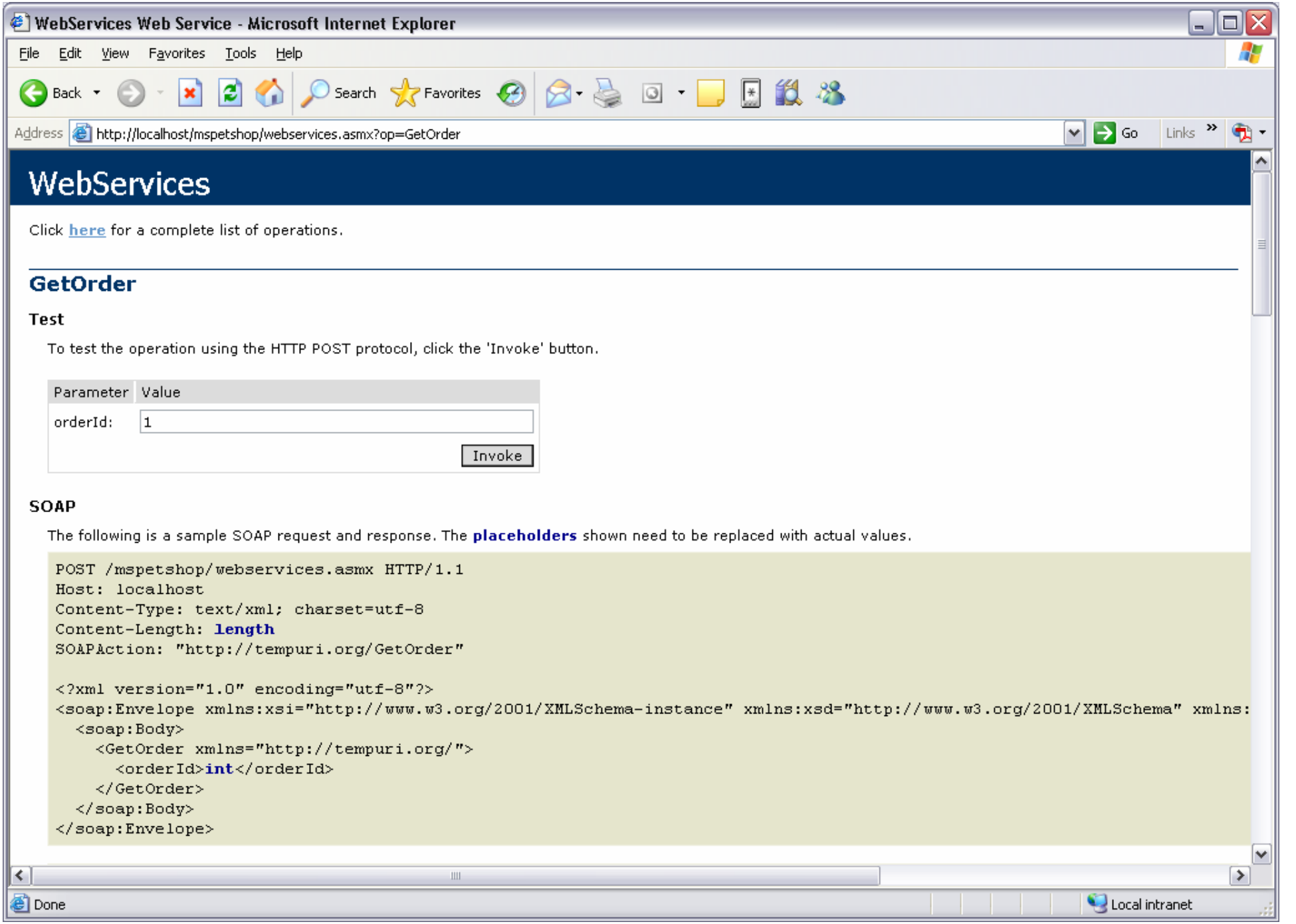

Order Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/OrderBilling.aspx
Go Links
Fish | Dogs | Reptiles | Cats | Birds
Payment Information
Credit Card Type: Visa
Card Number: 9999 9999 9999 9999
Expiration Date: Month: 01 Year: 2002
Billing Address
First Name: ABC
Last Name: XYX
Street Address: 901 San Antonio Road
MS UCUP02-206
City: Palo Alto
State / Province: California Postal Code: 94303
Country: USA
Telephone Number: 555-555-5555
Ship to billing address
Continue
Done
Local intranet

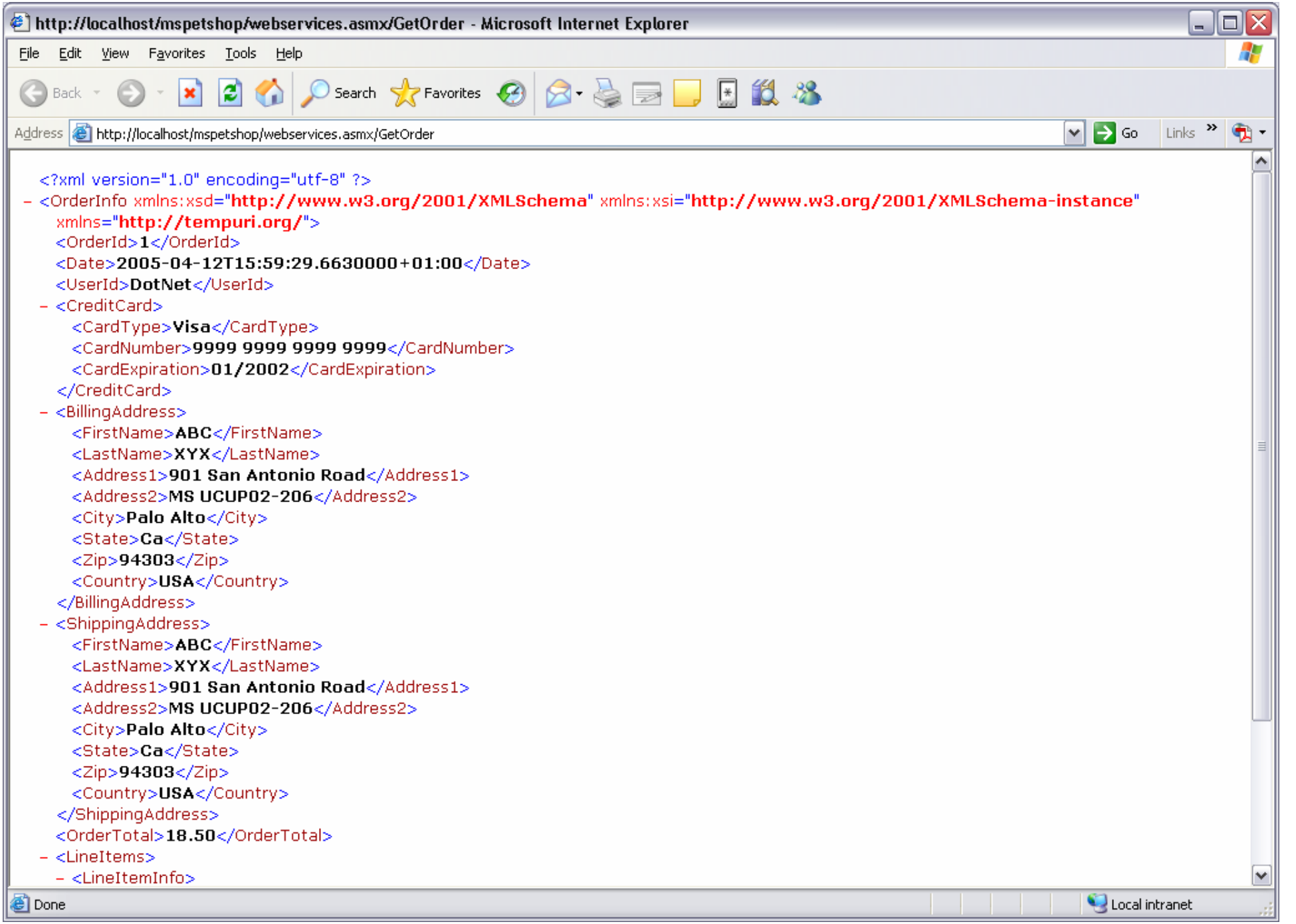

Order Information - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/OrderBilling.aspx
Go Links
THE .NET PET SHOP DEMO
SIGN OUT | MY ACCOUNT | | HELP
SEARCH
Fish | Dogs | Reptiles | Cats | Birds
Please confirm that the following information is correct and press the Continue button to process your order.
Billing Address
ABC XYX
901 San Antonio Road
MS UCUP02-206
Palo Alto, Ca 94303
Shipping Address
ABC XYX
901 San Antonio Road
MS UCUP02-206
Palo Alto, Ca 94303
Continue
Done
Local intranet

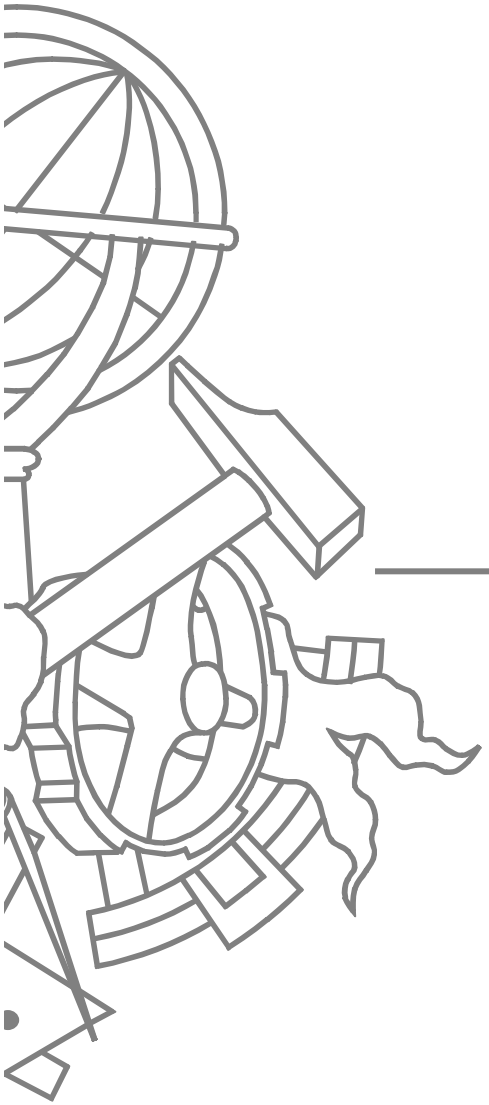

Order Processed - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/OrderProcess.aspx
Go Links
SIGN OUT | MY ACCOUNT | | HELP
SEARCH
Fish | Dogs | Reptiles | Cats | Birds
Order Complete!
Date: ter?a-feira, 12 de Abril de 2005
User ID: DotNet
Order ID: 1
Status: P
Payment Information: Visa
9999 9999 9999 9999
01/2002
Billing Address: ABC XYX
901 San Antonio Road
MS UCUP02-206
Palo Alto, Ca 94303
Shipping Address: ABC XYX
901 San Antonio Road
MS UCUP02-206
Palo Alto, Ca 94303
Items:
Item ID Product Price Quantity Subtotal
EST-4 Spotted 18,50 ? 1 18,50 ?
Total: 18,50 ?
Done
Local intranet


WebServices Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/webservices.asmx
Go Links
WebServices
Retrieves order information for a given order ID.
The following operations are supported. For a formal definition, please review the Service Description.
GetOrder
This web service is using http://tempuri.org/ as its default namespace.
Recommendation: Change the default namespace before the XML Web service is made public.
Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.
Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)
For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":
C#
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
// implementation
}
Visual Basic.NET
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
' implementation
End Class
For more details on XML namespaces, see the W3C recommendation on Namespaces in XML.
Done
Local intranet


http://localhost/mspetshop/webservices.asmx?WSDL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/webservices.asmx?WSDL
Go Links
<?xml version="1.0" encoding="utf-8" ?>
<wsdl:definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns="http://tempuri.org/" xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" targetNamespace="http://tempuri.org/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
<s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
<s:element name="GetOrder">
<s:complexType>
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="orderId" type="s:int" />
</s:sequence>
</s:complexType>
</s:element>
<s:element name="GetOrderResponse">
<s:complexType>
<s:sequence>
<s:element minOccurs="0" maxOccurs="1" name="GetOrderResult" type="tns:OrderInfo" />
</s:sequence>
</s:complexType>
</s:element>
<s:complexType name="OrderInfo">
<s:sequence>
<s:element minOccurs="1" maxOccurs="1" name="OrderId" type="s:int" />
<s:element minOccurs="1" maxOccurs="1" name="Date" type="s:dateTime" />
<s:element minOccurs="0" maxOccurs="1" name="UserId" type="s:string" />
<s:element minOccurs="0" maxOccurs="1" name="CreditCard" type="tns:CreditCardInfo" />
<s:element minOccurs="0" maxOccurs="1" name="BillingAddress" type="tns:AddressInfo" />
<s:element minOccurs="0" maxOccurs="1" name="ShippingAddress" type="tns:AddressInfo" />
<s:element minOccurs="1" maxOccurs="1" name="OrderTotal" type="s:decimal" />
<s:element minOccurs="0" maxOccurs="1" name="LineItems" type="tns:ArrayOfLineItemInfo" />
</s:sequence>
</s:complexType>
Done
Local intranet

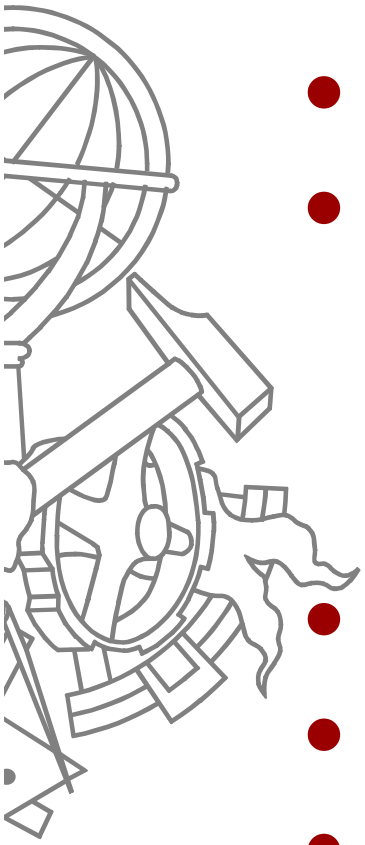

WebServices Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/webservices.asmx?op=GetOrder
Go Links
WebServices
Click here for a complete list of operations.
GetOrder
Test
To test the operation using the HTTP POST protocol, click the 'Invoke' button.
Parameter Value
orderId: 1
Invoke
SOAP
The following is a sample SOAP request and response. The placeholders shown need to be replaced with actual values.
POST /mspetshop/webservices.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/GetOrder"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:
<soap:Body>
<GetOrder xmlns="http://tempuri.org/">
<orderId>int</orderId>
</GetOrder>
</soap:Body>
</soap:Envelope>
Done
Local intranet


http://localhost/mspetshop/webservices.asmx/GetOrder - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/mspetshop/webservices.asmx/GetOrder
Go Links
<?xml version="1.0" encoding="utf-8" ?>
<OrderInfo xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://tempuri.org/">
<OrderId>1</OrderId>
<Date>2005-04-12T15:59:29.6630000+01:00</Date>
<UserId>DotNet</UserId>
<CreditCard>
<CardType>Visa</CardType>
<CardNumber>9999 9999 9999 9999</CardNumber>
<CardExpiration>01/2002</CardExpiration>
</CreditCard>
<BillingAddress>
<FirstName>ABC</FirstName>
<LastName>XYX</LastName>
<Address1>901 San Antonio Road</Address1>
<Address2>MS UCUP02-206</Address2>
<City>Palo Alto</City>
<State>Ca</State>
<Zip>94303</Zip>
<Country>USA</Country>
</BillingAddress>
<ShippingAddress>
<FirstName>ABC</FirstName>
<LastName>XYX</LastName>
<Address1>901 San Antonio Road</Address1>
<Address2>MS UCUP02-206</Address2>
<City>Palo Alto</City>
<State>Ca</State>
<Zip>94303</Zip>
<Country>USA</Country>
</ShippingAddress>
<OrderTotal>18.50</OrderTotal>
<LineItems>
<LineItemInfo>
Done
Local intranet



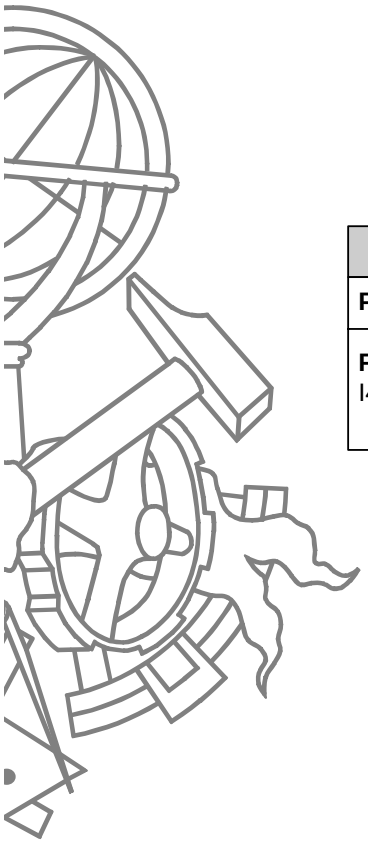
Descrição técnica

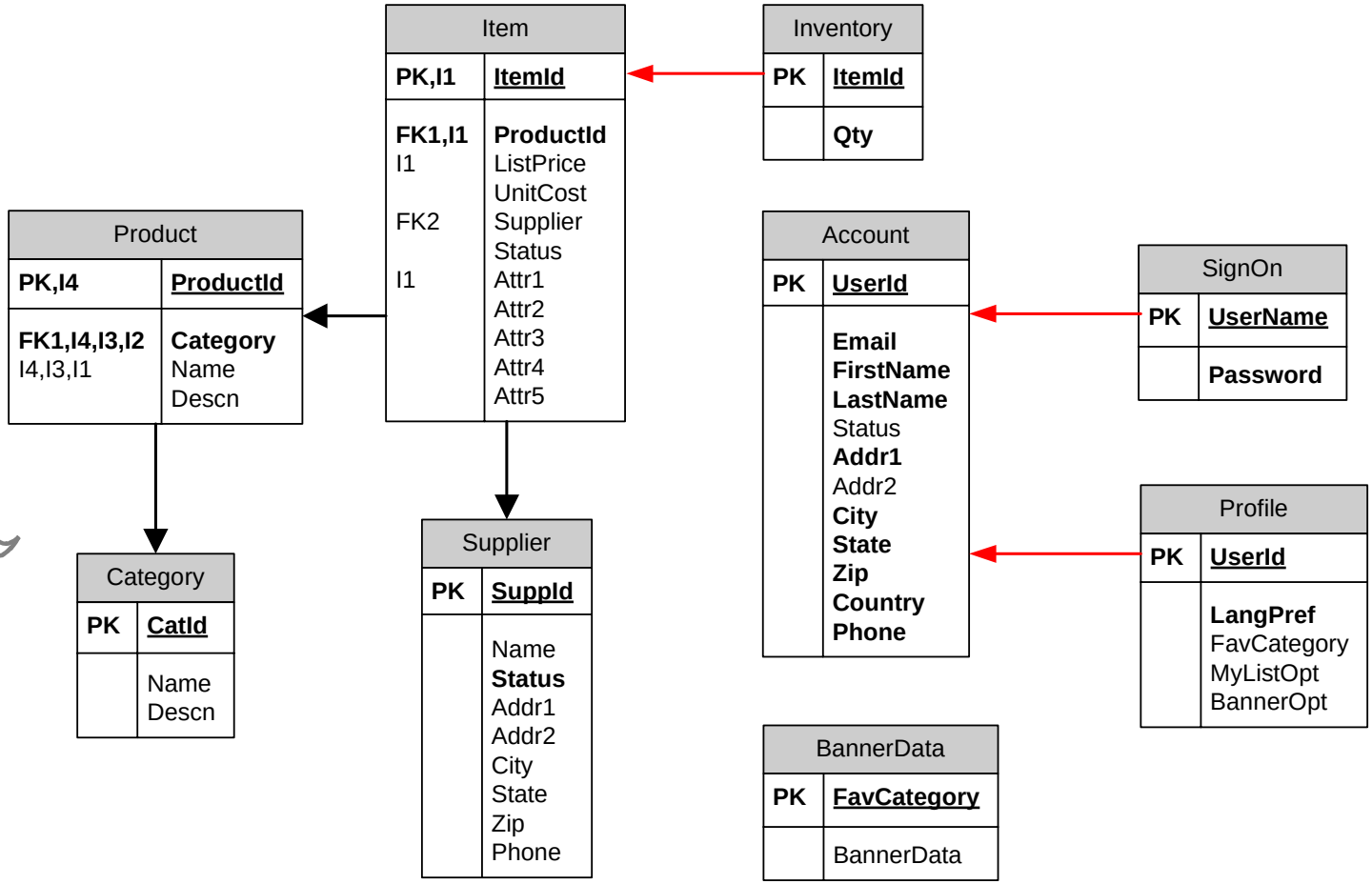
O que é?



- Site de venda online de animais
- Sistema de processamento de encomendas em separado
  - Outra base de dados
  - Implica transacção distribuída
- Aplicação em 3 camadas
- Utiliza padrões de software
- Solução multi-projecto

Modelo de Dados: sistema de venda

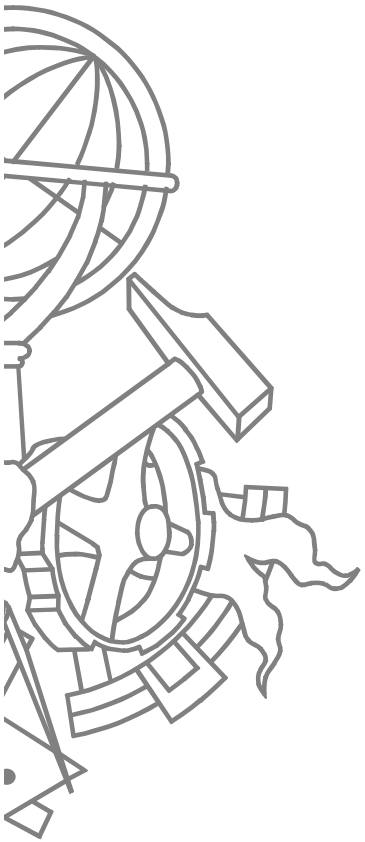


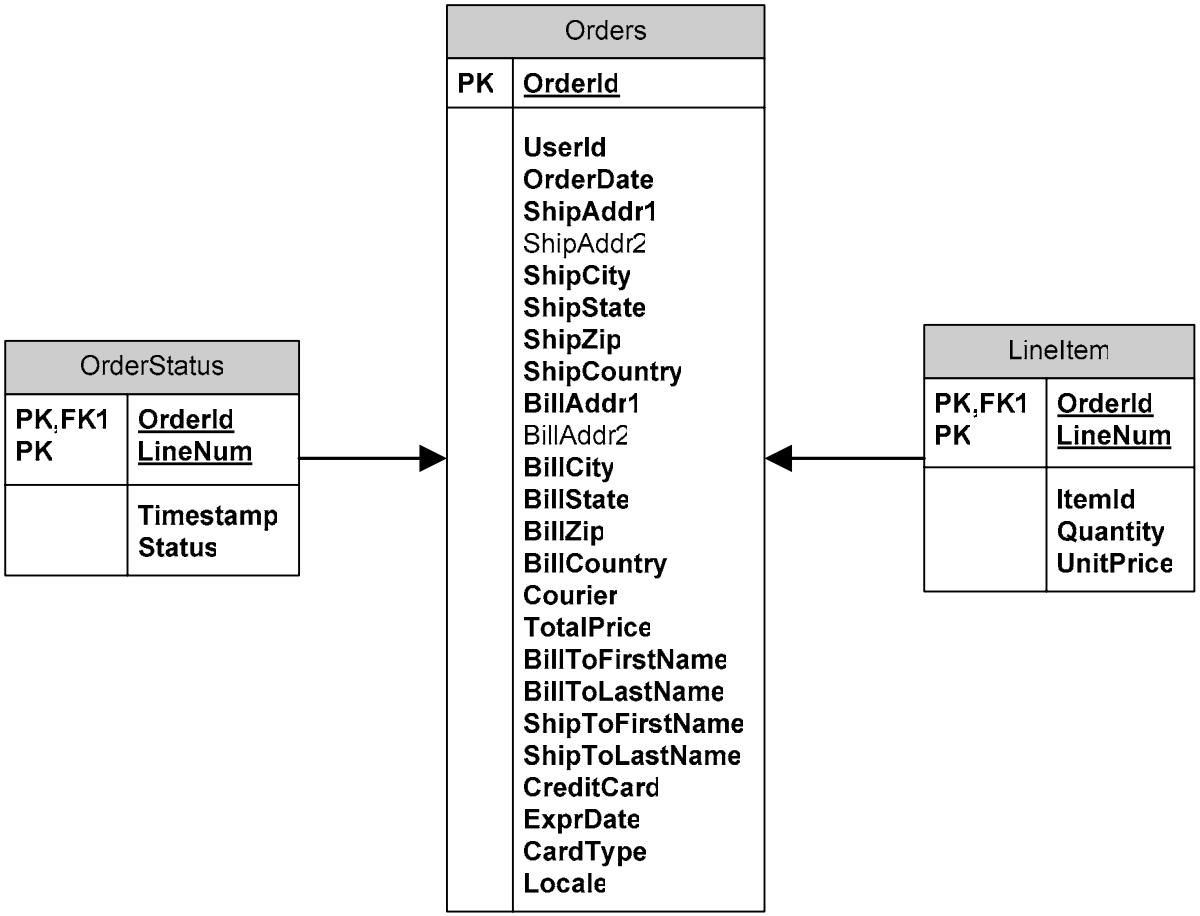


→ Relações existentes na BD

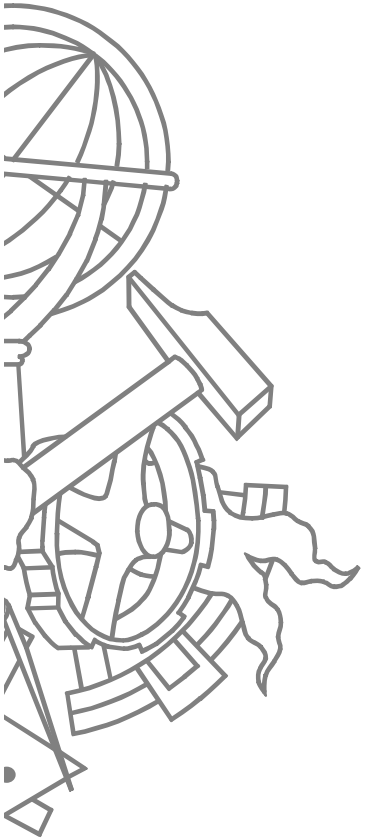
→ Relações existentes mas não declaradas na BD

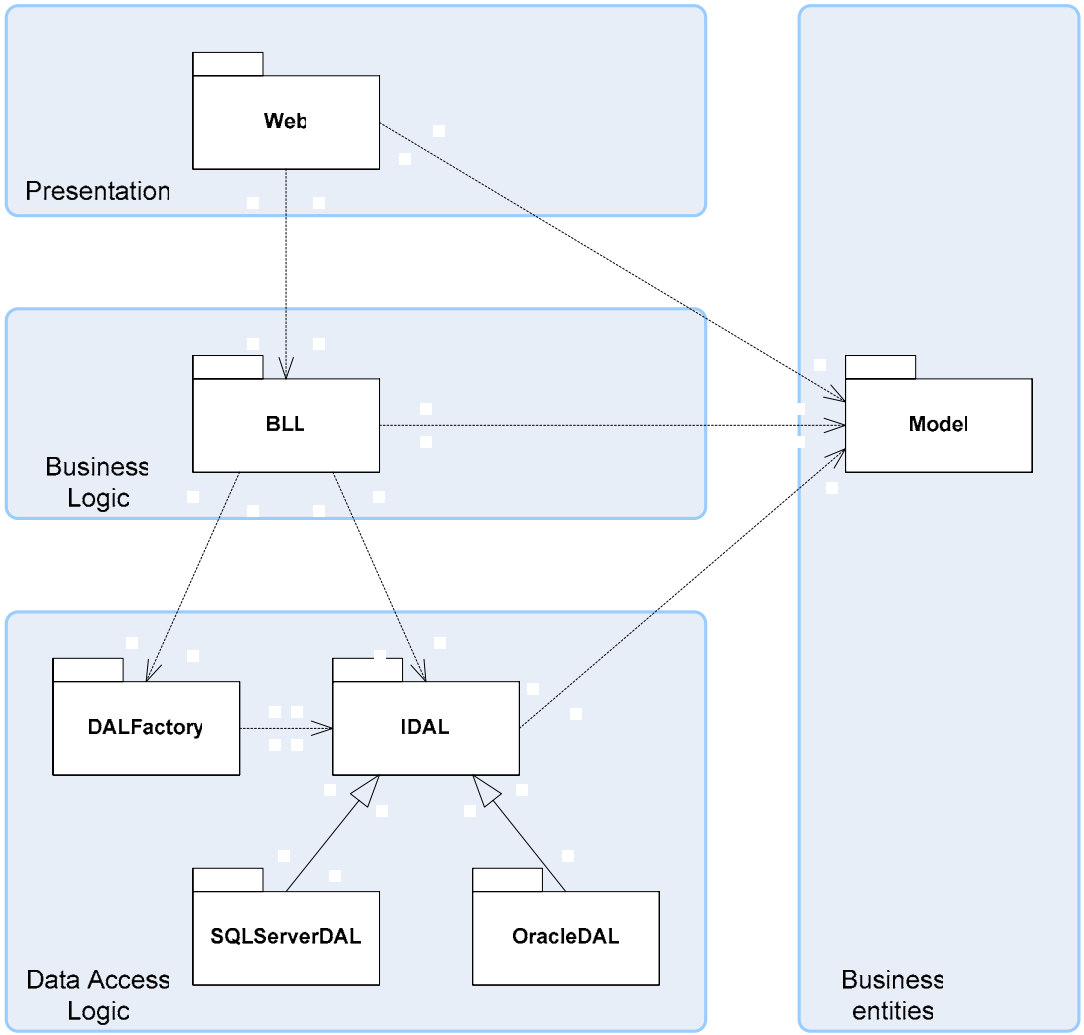
Modelo de Dados: sistema de encomendas



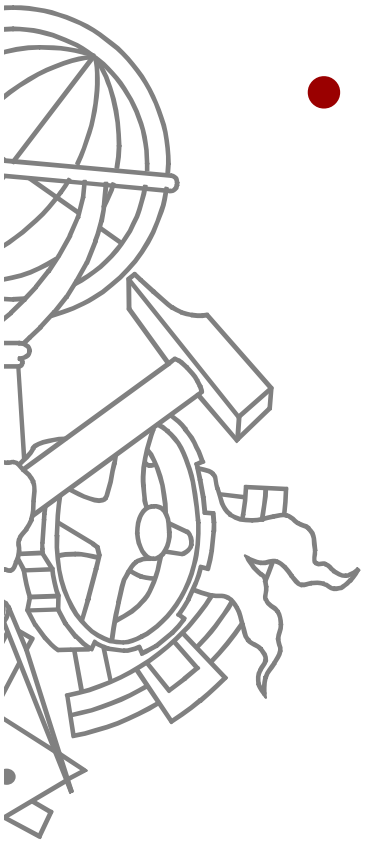


Arquitectura



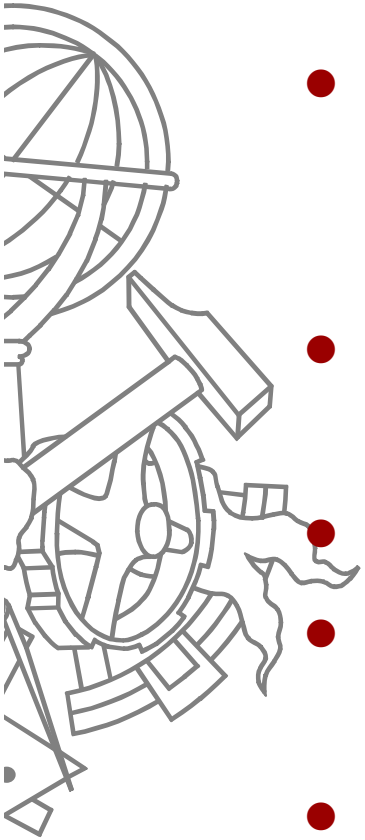


Entidades de negócio



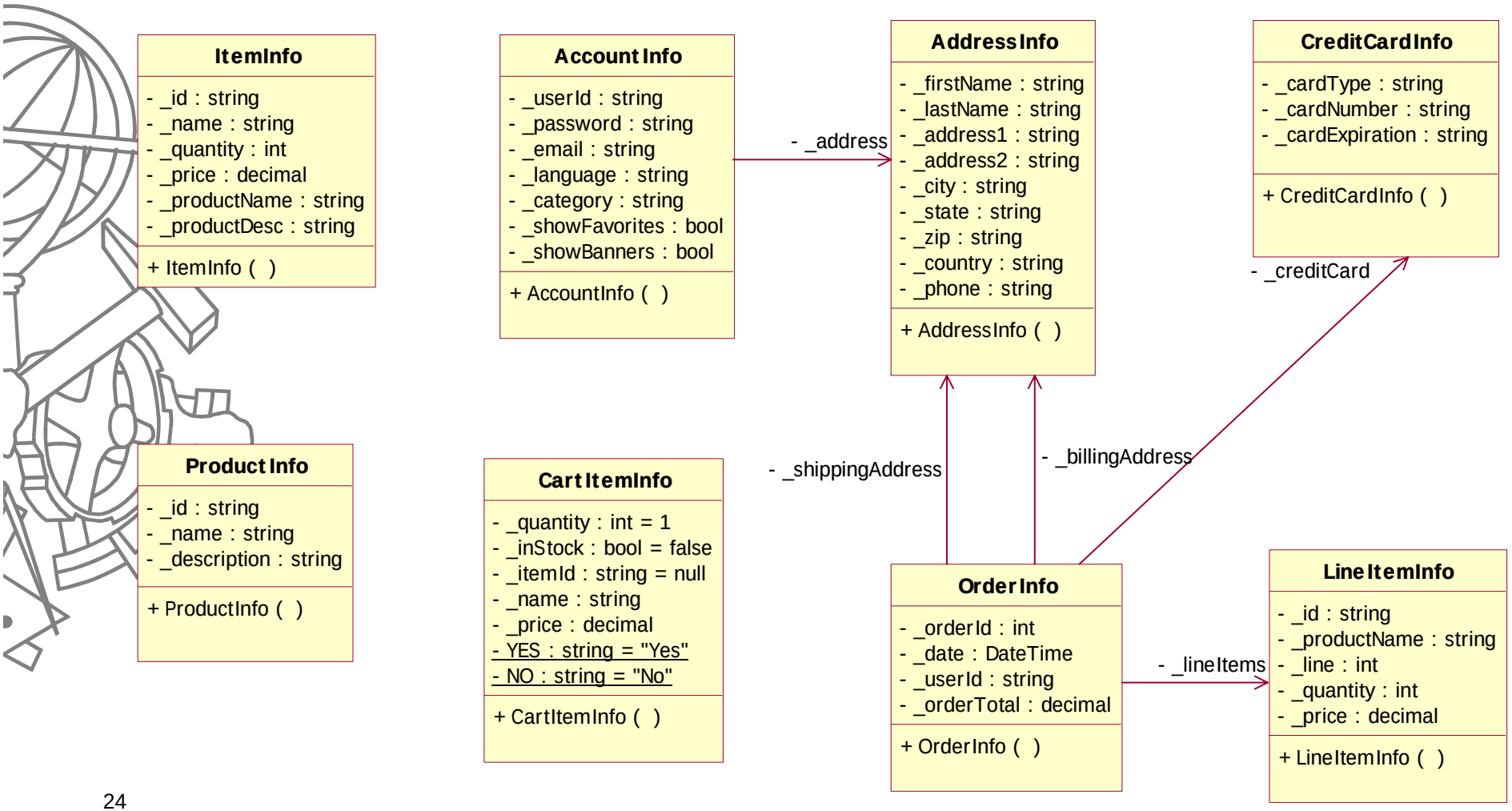
- Usa ***Custom classes*** para representar os conceitos de negócio existentes
  - Account
  - Address
  - CartItem
  - CreditCard
  - Item
  - Order + LineItem
  - Product

Entidades de negócio

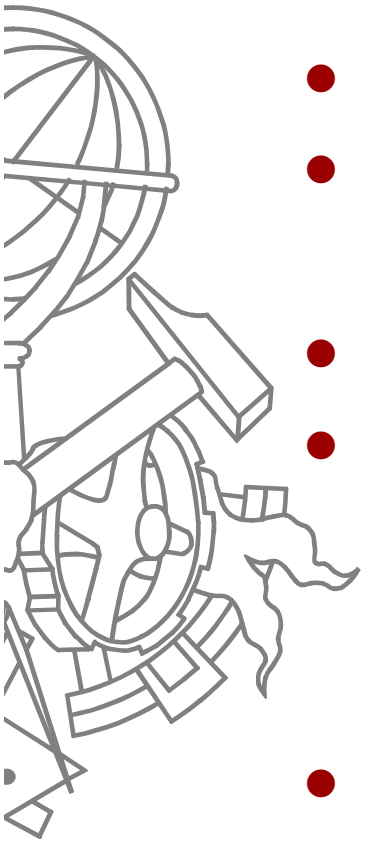


- Colecções de classes são retornadas usando um dos contentores da plataforma que implemente a interface **`IList`**
- Cada classe tem o nome do conceito concatenado de “Info”
- Todas as classes são *Serializable*
- Classes apenas possuem dados e Atributos e construtor para facilitar a utilização
- Conceitos complexos compostos usam *arrays*
  - Exemplo: Order + LineItem

Entidades de Negócio



Lógica de apresentação

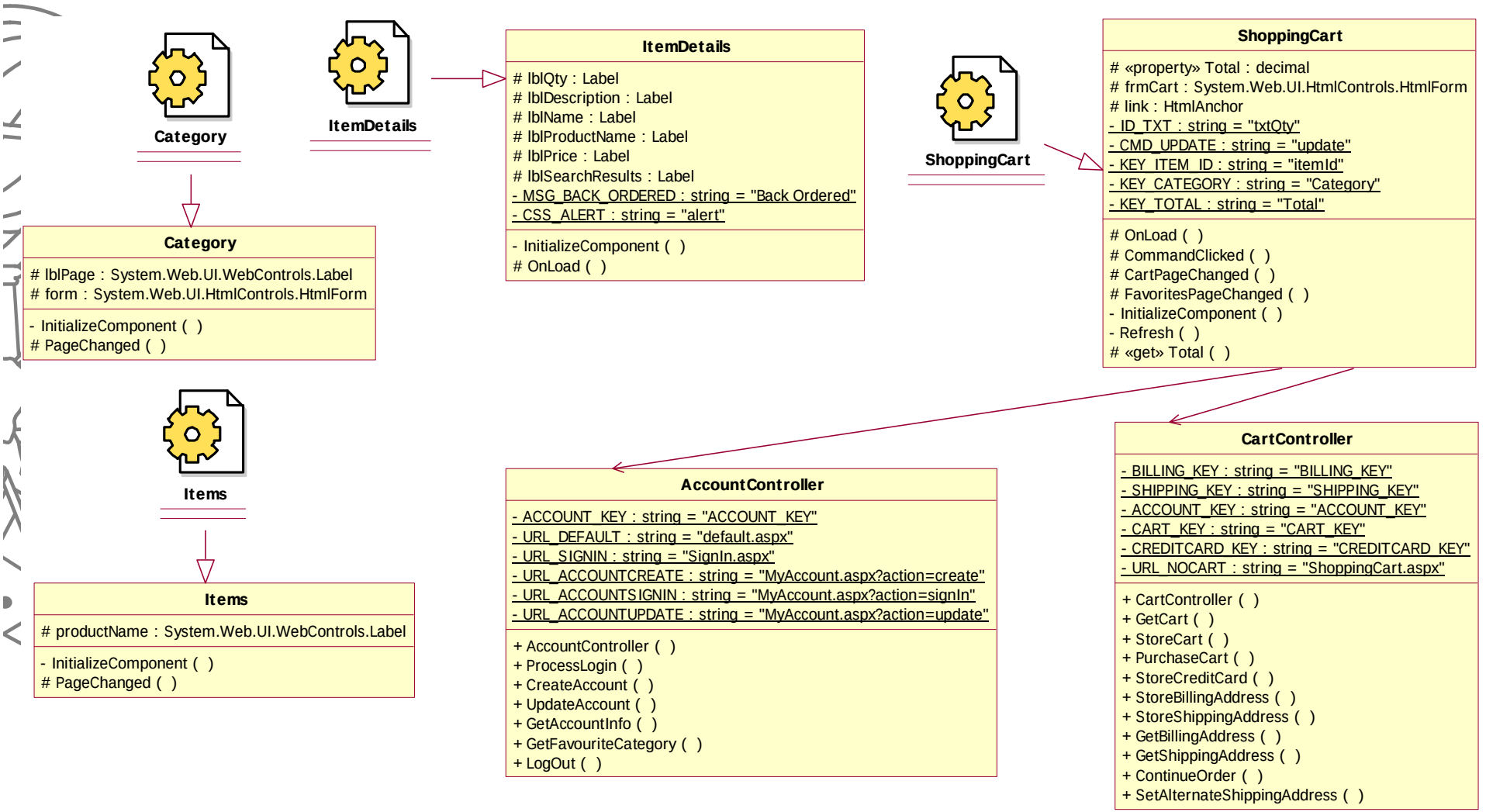


- Template view
- Page Controller

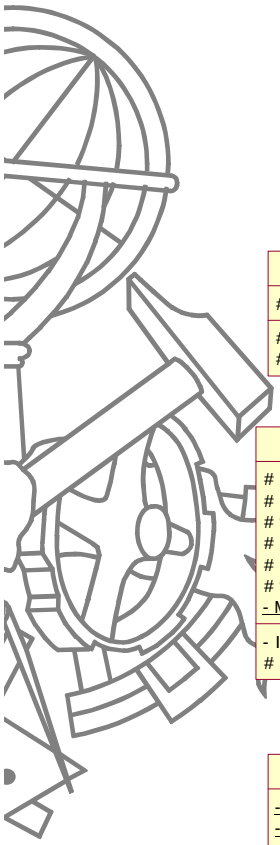
- Páginas ASP.net com *code behind*
- Page flow controller
  - Account
  - Cart

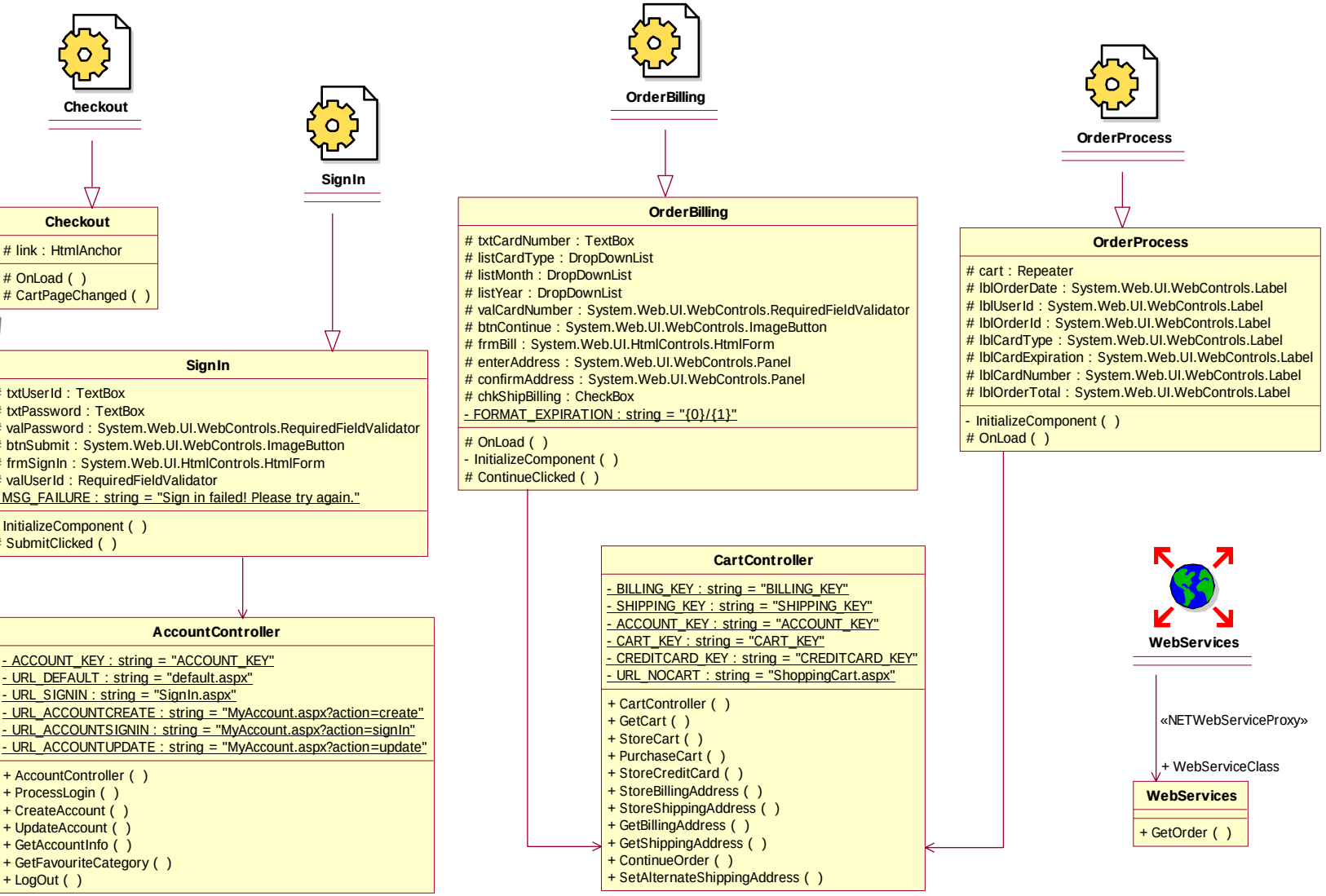
- Web service para obter dados de uma encomenda

Lógica de apresentação (1/2)

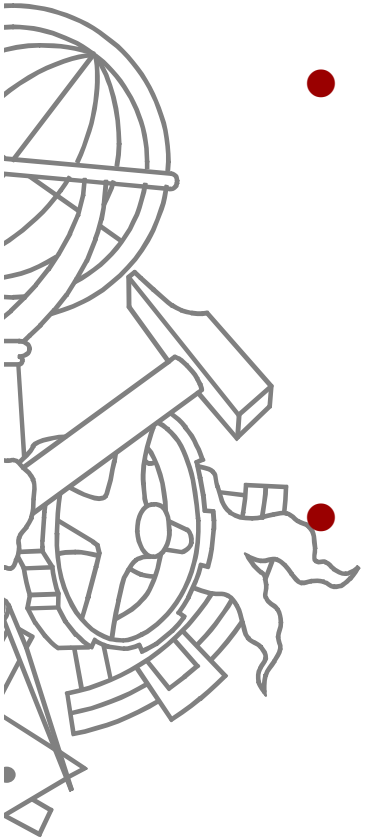


Lógica de apresentação (2/2)



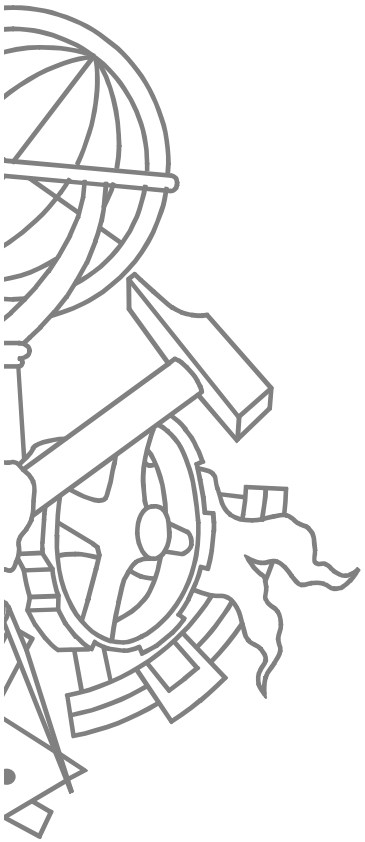


Lógica de negócio



- Table Module
  - Parâmetros e tipos de retorno dos métodos são objectos do tipo das entidades de negócio
  - Classe **`Cart`** é na realidade um Domain Model
- Classe “Order” dividida em duas: **`OrderRead`** e **`OrderInsert`**
  - Transação distribuída na inserção
  - Lidar com *overhead* de *serviced components*

Lógica de negócio (1/2)



Item

+ GetItemsByProduct ([in] productId : string) : IList
+ GetItem ([in] itemId : string) : ItemInfo

Product

+ GetProductsByCategory ([in] category : string) : IList
+ GetProductsBySearch ([in] text : string) : IList

Profile

+ GetBannerPath ([in] favCategory : string) : string

Account

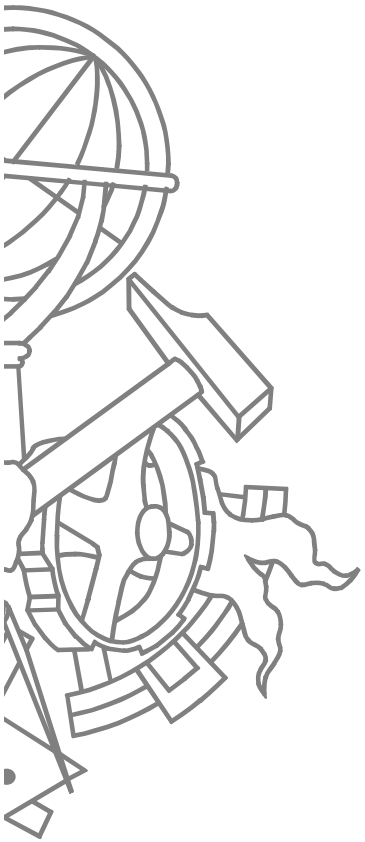
+ SignIn ([in] userId : string , [in] password : string) : AccountInfo
+ GetAddress ([in] userId : string) : AddressInfo
+ Insert ([in] account : AccountInfo)
+ Update ([in] account : AccountInfo)

Cart

+ «property» Total : decimal
+ «property» Count : int
+ «indexer» this[int] : CartItemInfo
- _total : decimal = 0

+ GetEnumerator ( ) : IEnumerator
+ Add ([in] ItemId : string)
+ Remove ([in] itemId : string)
+ RemoveAt ([in] index : int)
+ GetCartItems ( ) : ArrayList
+ GetOrderLineItems ( ) : ArrayList
- GetInStock ([in] ItemId : string) : int
+ «get» Total ( ) : decimal
+ «set» Total ([in] value : decimal)
+ «get» Count ( ) : int
+ «get» this ([in] index : int) : CartItemInfo

Lógica de negócio (2/2)



Inventory

+ CurrentQuantityInStock ([in] itemId : string) : int
+ TakeStock ([in] items : LineItemInfo[])

OrderRead

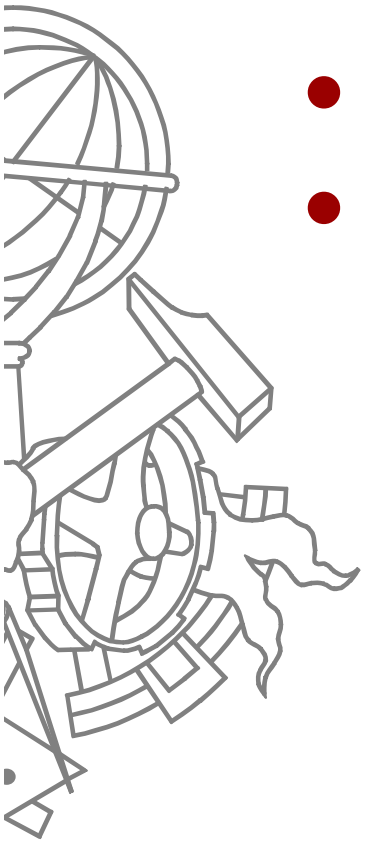
+ GetOrder ([in] orderId : int) : OrderInfo

OrderInsert

- ACID_USER_ID : string = "ACID"
- ACID_ERROR_MSG : string = "ACID test exception thrown for distributed transaction!"

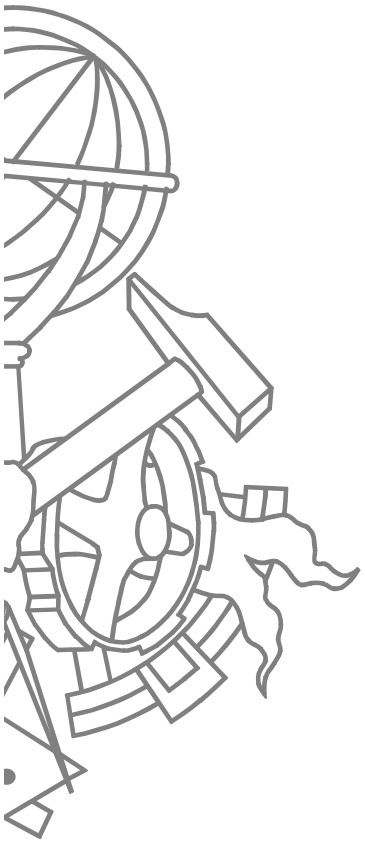
CanBePooled ( ) : bool
+ Insert ([in] order : OrderInfo) : int

Lógica de acesso a dados



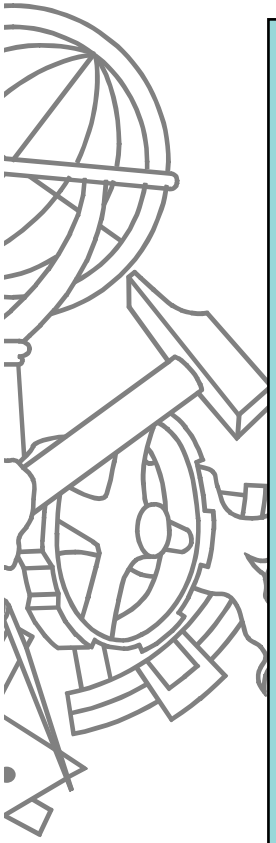
- Table Data Gateway
- Factory

Lógica de acesso a dados: IDAL



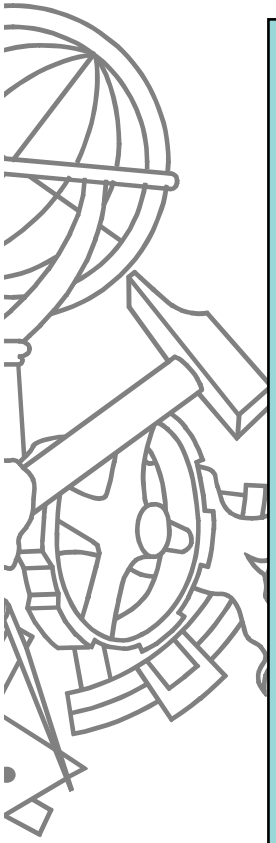
- IInventory
- IOrder
- IItem
- IProfile
- IProduct
- IAccount

Lógica de acesso a dados: IDAL



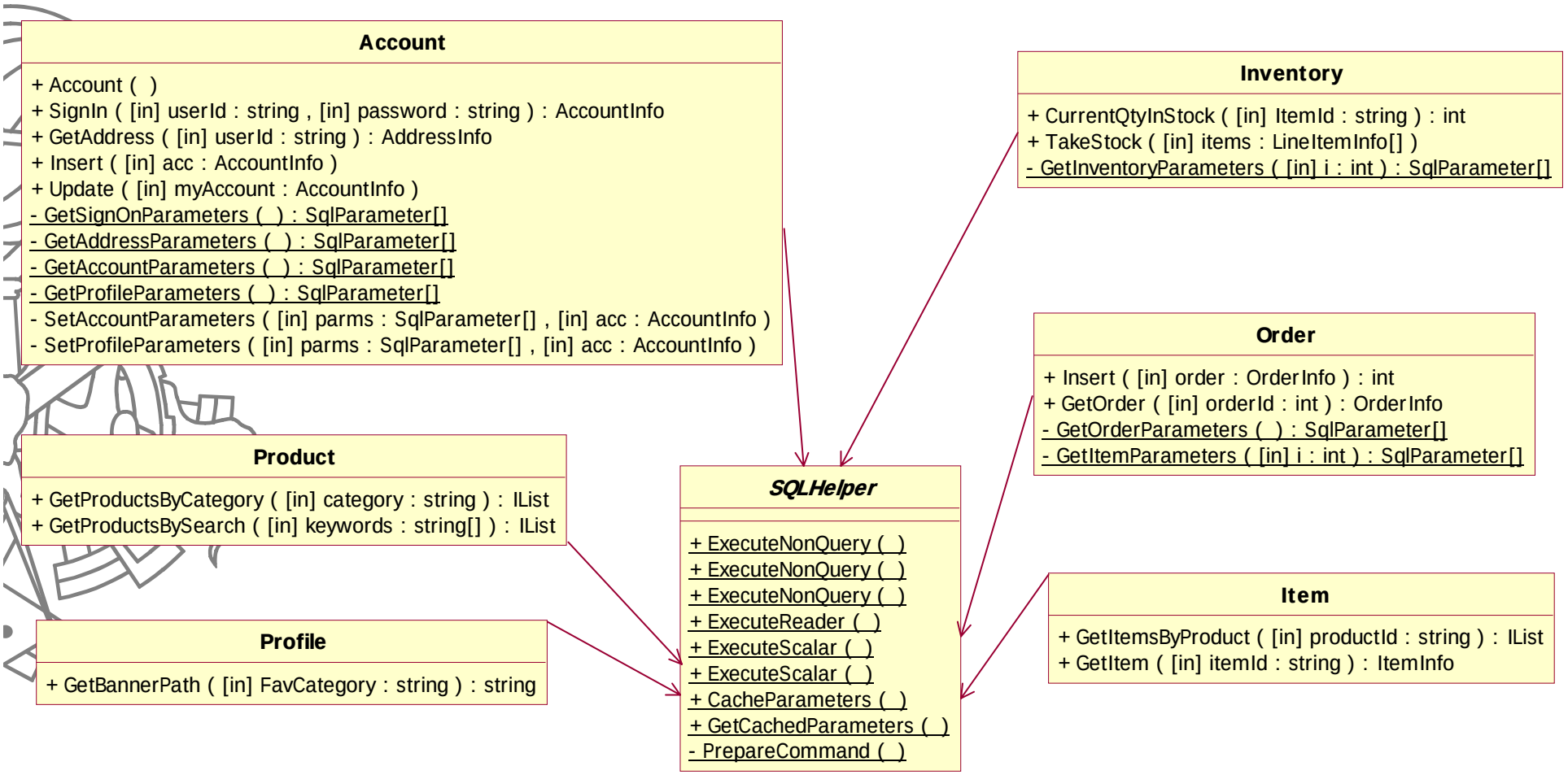
```
public interface IProfile
{
     string GetBannerPath(string FavCategory);
}

public interface IAccount
{
     AccountInfo SignIn(string userId, string password);
     AddressInfo GetAddress(string userId);
     void Insert(AccountInfo account);
     void Update(AccountInfo Account);
}

public interface IInventory
{
     int CurrentQtyInStock(string ItemId);
     void TakeStock(LineItemInfo[] items);
}
```

Lógica de acesso a dados: IDAL

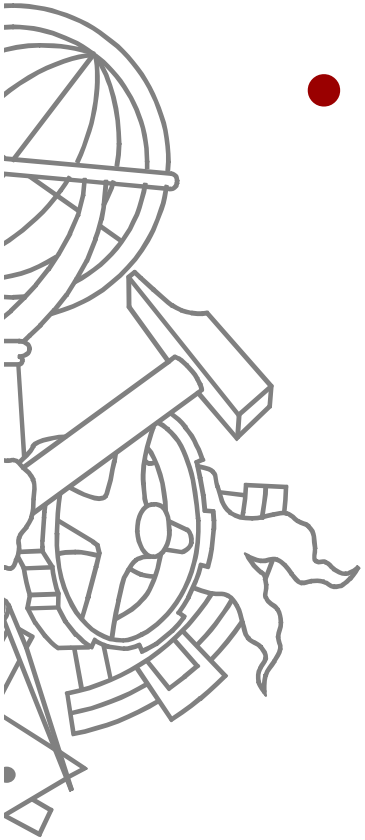


```
public interface IItem
{
     IList GetItemsByProduct(string productId);
     ItemInfo GetItem(string itemId);
}

public interface IProduct
{
     IList GetProductsByCategory(string category);
     IList GetProductsBySearch(string[] keywords);
}

public interface IOrder
{
     int Insert(OrderInfo order);
     OrderInfo GetOrder(int orderId);
}
```

Lógica de acesso a dados: SQLServerDAL



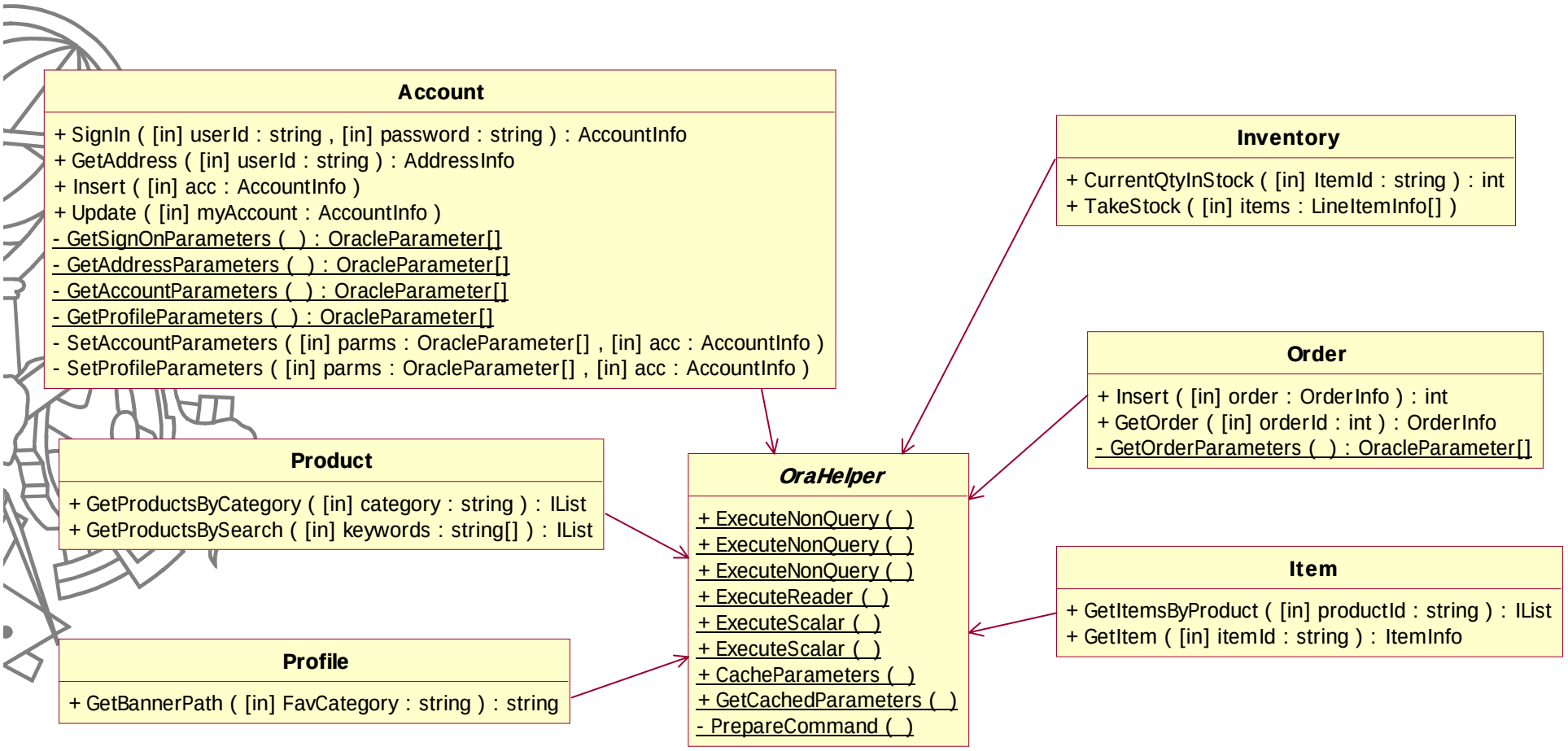
Lógica de acesso a dados: SQLServerDAL



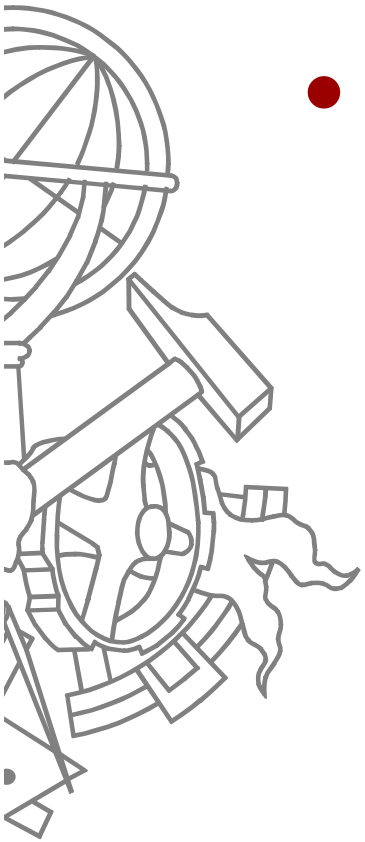
- Uma das classes

Inventory
- SQL_SELECT_INVENTORY : string = "SELECT Qty FROM Inventory WHERE ItemId = @ItemId" - SQL_TAKE_INVENTORY : string = "UPDATE Inventory SET Qty = Qty - "
+ CurrentQtyInStock ( ) + TakeStock ( ) - GetInventoryParameters ( )

Lógica de acesso a dados: OracleDAL



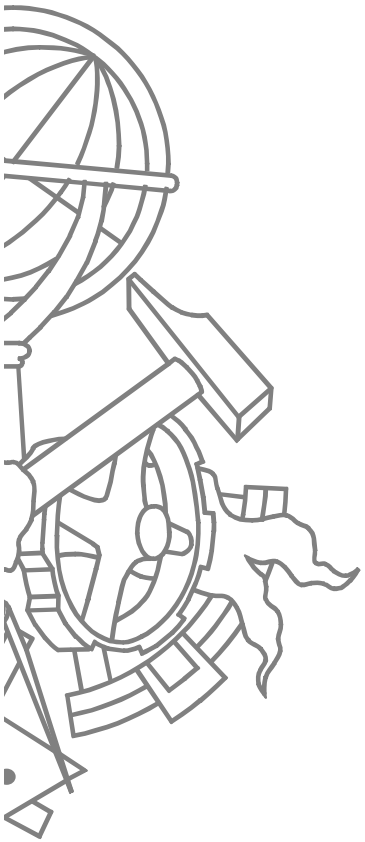
Lógica de acesso a dados: OracleDAL



- Uma das classes

Inventory
- SQL_SELECT_INVENTORY : string = "SELECT Qty FROM Inventory WHERE ItemId = :ItemId" - SQL_TAKE_INVENTORY : string = "UPDATE Inventory SET Qty = Qty - :Quantity{0} WHERE ItemId = :ItemId{0}"
+ CurrentQtyInStock ( ) + TakeStock ( )

Lógica de acesso a dados: DALFactory



Account
+ Create ( ) : PetShop.IDAL.IAccount

Profile
+ Create ( ) : PetShop.IDAL.IProfile

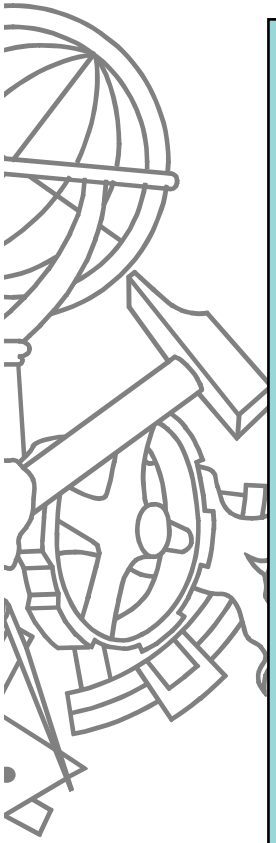
Inventory
+ Create ( ) : PetShop.IDAL.IInventory

Order
+ Create ( ) : PetShop.IDAL.IOrder

Product
+ Create ( ) : PetShop.IDAL.IProduct

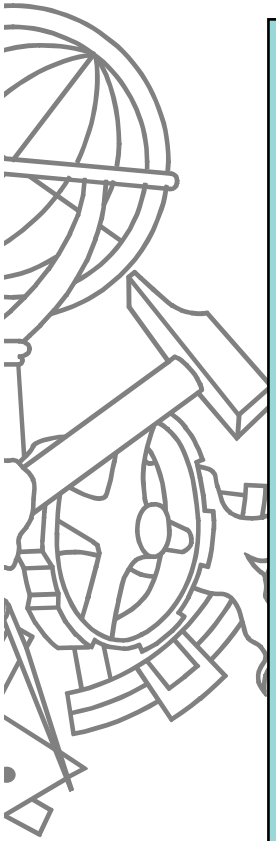
Item
+ Create ( ) : PetShop.IDAL.IItem

Lógica de acesso a dados: DALFactory

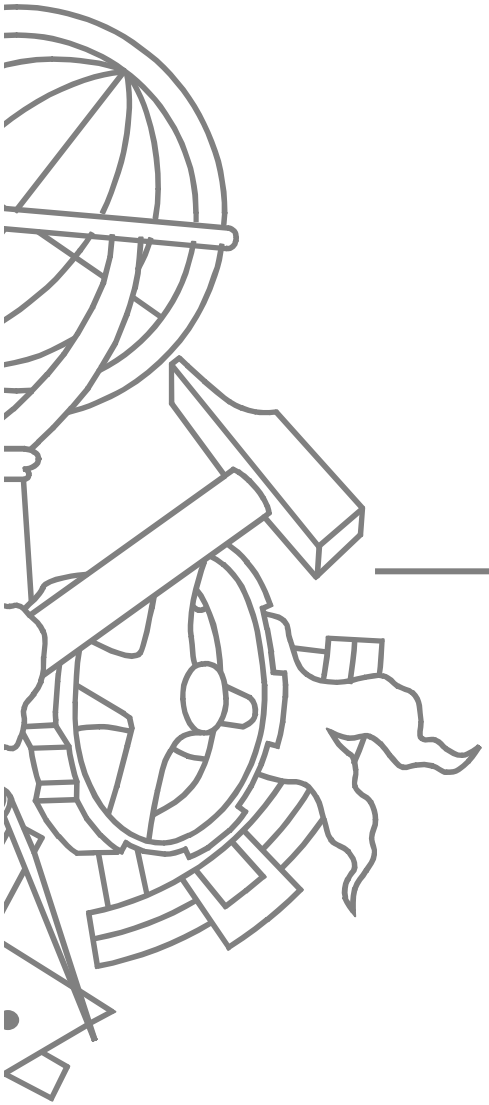


```
namespace PetShop.DALFactory
{
public class Account
{
    public static PetShop.IDAL.IAccount Create()
    {
          /// Look up the DAL implementation we should be using
          string path =
    System.Configuration.ConfigurationSettings.AppSettings["WebDAL"];
          string className = path + ".Account";

          // Using the evidence given in the config file load the
    appropriate assembly and class
          Assembly asb = Assembly.Load(path);
          return (PetShop.IDAL.IAccount)asb.CreateInstance(className);
    }
}
}
```

Ficheiro de configuração

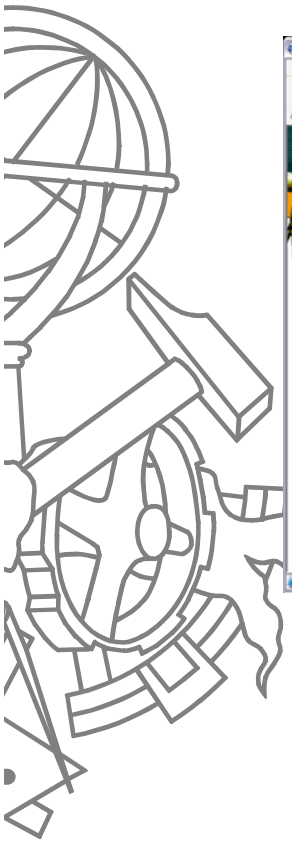


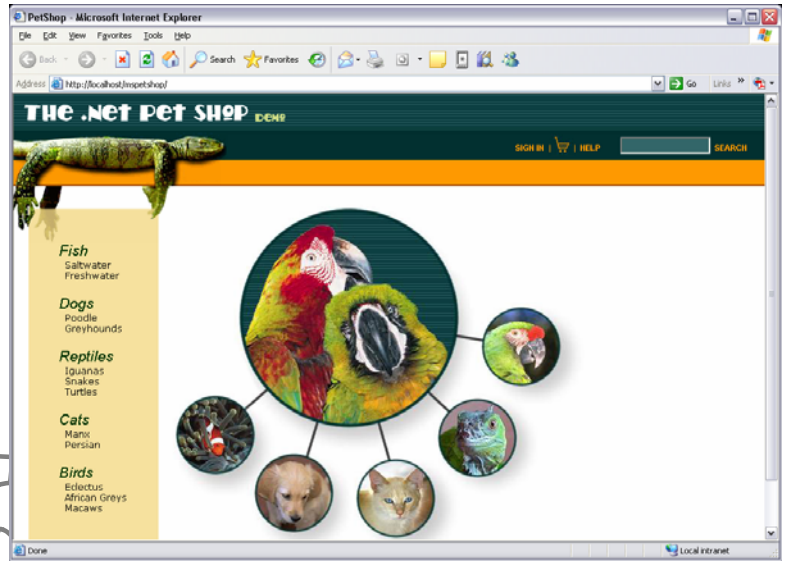
```
<configuration>
   <appSettings>
        ...

        <add key="WebDAL"
               value="PetShop.SQLServerDAL" />
        <add key="OrdersDAL"
               value= "PetShop.OracleDAL" />
        ...
   </appSettings>

   ...
</configuration>
```



Fluxo de execução

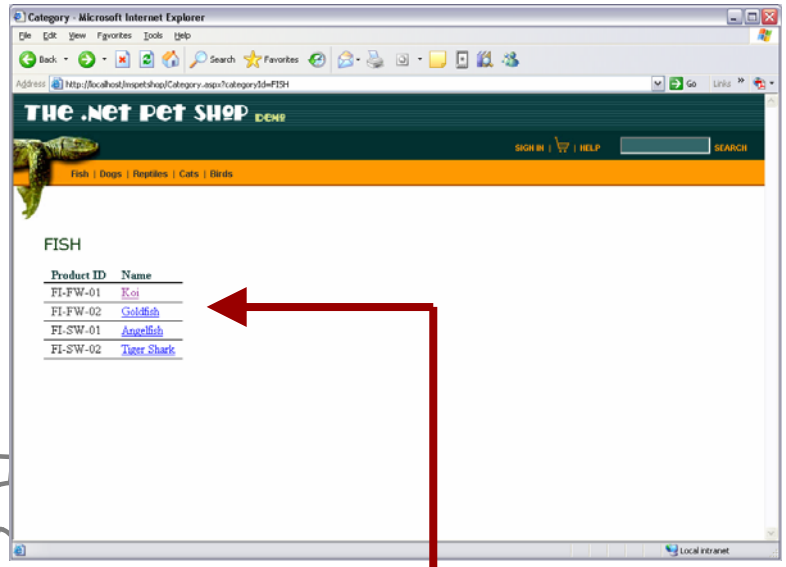
Página inicial





- Default.aspx
- Conteúdo estático
- Cada link redirecciona para página Category.aspx passando um parâmetro na *query string*
  - /Category.aspx?categoryId =FISH

Visualizar produtos de uma categoria

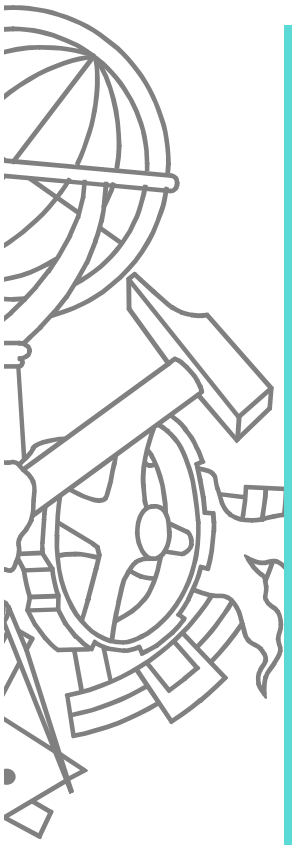


DataSource

- Category.aspx
- A lista de produtos da categoria passada por parâmetro é construída pelo ASP.net usando uma classe que estende **`Repeater`**
  - **`SimplePager`**

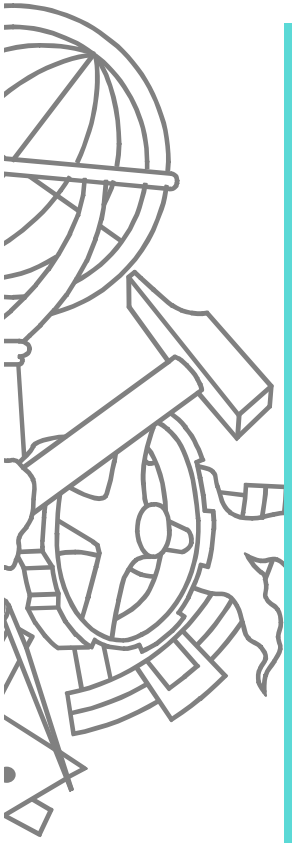
```
Product product = new Product();
IList productsByCategory =
        product.GetProductsByCategory(categoryKey);
```

BLL.Product



```
public IList GetProductsByCategory(string category)
{
   // Return null if the string is empty
   if (category.Trim() == string.Empty)
        return null;

   // Get an instance of the Product DAL using the
   DALFactory
   IProduct dal = DALFactory.Product.Create();

   // Run a search against the data store
   return dal.GetProductsByCategory(category);
}
```

DAL.Product

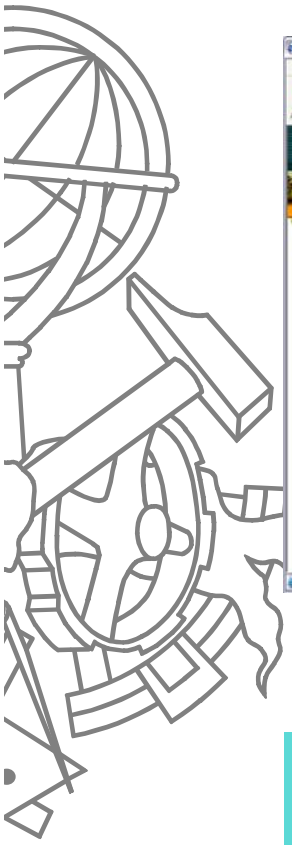


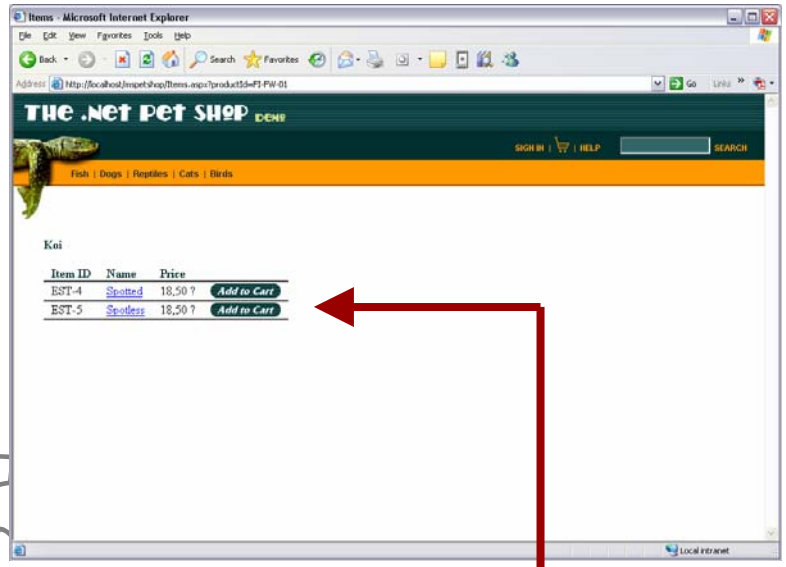
```
public IList GetProductsByCategory(string category)
{
    IList productsByCategory = new ArrayList();

    SqlParameter parm = new SqlParameter(PARM_CATEGORY,
                                         SqlDbType.Char, 10);
    parm.Value = category;

    //Execute a query to read the products
    using (SqlDataReader rdr =
           SQLHelper.ExecuteReader(SQLHelper.CONN_STRING_NON_DTC,
           CommandType.Text, SQL_SELECT_PRODUCTS_BY_CATEGORY, parm))
    {
        while (rdr.Read()){
            ProductInfo product = new ProductInfo(rdr.GetString(0),
                                                  rdr.GetString(1),
                                                  null);
            productsByCategory.Add(product);
        }
    }

    return productsByCategory;
}
```

SQL

```
SELECT ProductId, Name
FROM Product
WHERE Category = @Category
```

Visualizar itens de um produto



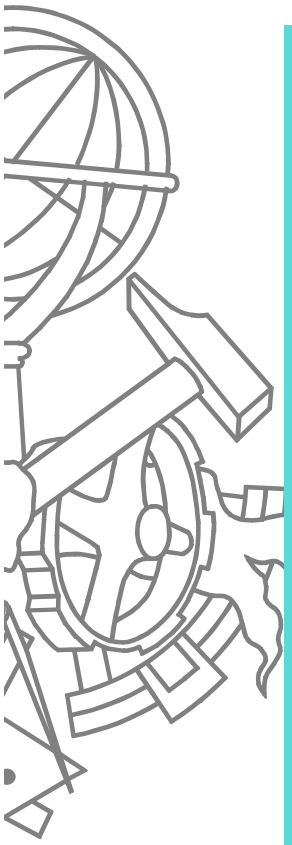


DataSource

- Itens.aspx
- Usa parâmetro para determinar que produto visualizar
  - /Items.aspx?productId =FI-FW-01

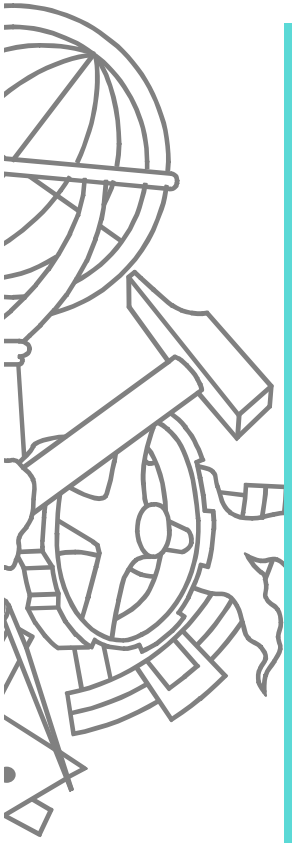
```
Item item = new Item();
IList itemsByProduct =
                item.GetItemsByProduct(productId);
```

BLL.Item



```
public IList GetItemsByProduct(string productId)
{
   // Validate input
   if (productId.Trim() == string.Empty)
        return null;

   // Get an instance of the Item DAL using the DALFactory
   IItem dal = PetShop.DALFactory.Item.Create();

   // Use the dal to search by productId
   return dal.GetItemsByProduct(productId);
}
```

DAL.Item

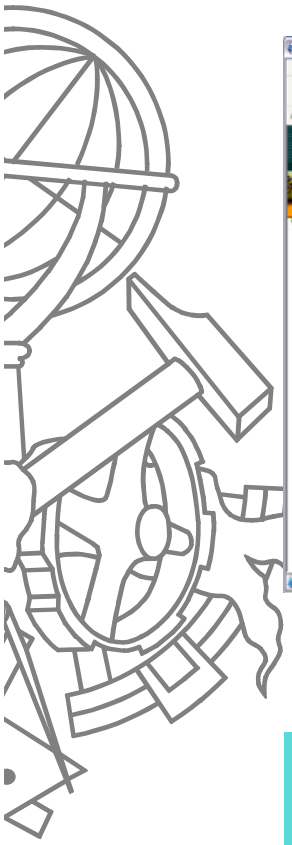


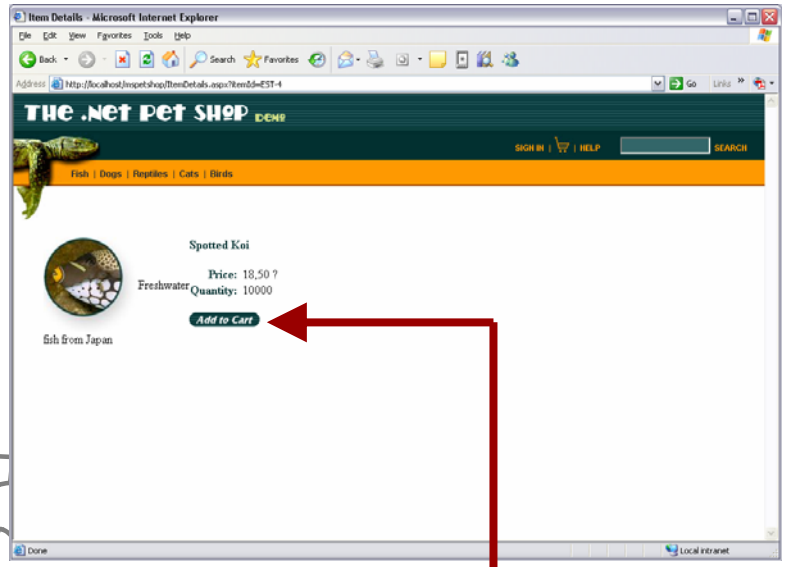
```
public IList GetItemsByProduct(string productId)
{
    IList productsByCategory = new ArrayList();
    SqlParameter parm = new SqlParameter(PARM_PRODUCT_ID,
                                         SqlDbType.Char, 10);
    parm.Value = productId;

    //Execute the query against the database
    using (SqlDataReader rdr =
           SQLHelper.ExecuteReader(SQLHelper.CONN_STRING_NON_DTC,
           CommandType.Text, SQL_SELECT_ITEMS_BY_PRODUCT, parm))
    {
        // Scroll through the results
        while (rdr.Read()){
            ItemInfo item = new ItemInfo(rdr.GetString(0).Trim(),
                            rdr.GetString(1), rdr.GetDecimal(2),
                            rdr.GetString(3), null);
            //Add each item to the arraylist
            itemsByProduct.Add(item);
        }
    }

    return productsByCategory;
}
```

SQL

```
SELECT ItemId, Attr1, ListPrice, Name
FROM Item INNER JOIN Product ON
   Item.ProductId = Product.ProductId
WHERE Item.ProductId = @ProductId
```

Visualizar item



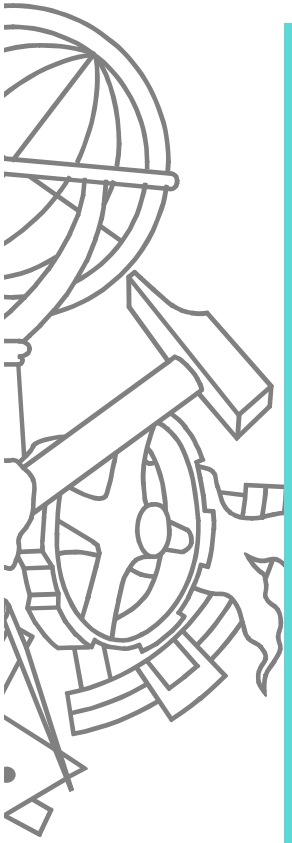


- ItemDetails.aspx
- Usa parâmetro
  - /ItemDetails.aspx?itemId =EST-4

OnClick

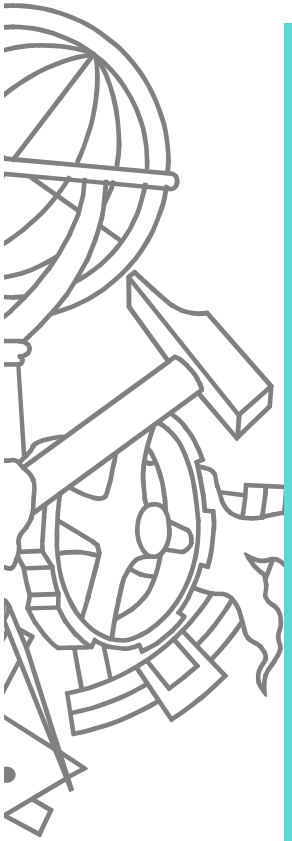
```
ShoppingCart.aspx?itemId= <% Request["itemId"] %>
```

Web.ItemDetails



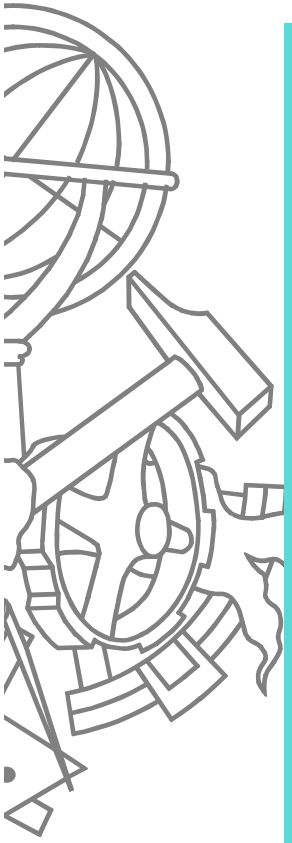
```
override protected void OnLoad(EventArgs e)
{
    // Fetch the key field from the query stirng
    string itemId = WebComponents.CleanString.InputText(Request["itemId"],
    50);
    // Create an instance of the item business components
    Item item = new Item();
    // Get the item info from the item component
    ItemInfo itemInfo = item.GetItem(itemId);
    // If an item is found then display the results
    if(itemInfo != null){
          lblDescription.Text = itemInfo.ProductDesc;
          lblName.Text = itemInfo.Name;
          lblProductName.Text = itemInfo.ProductName;
          lblPrice.Text = itemInfo.Price.ToString("c");
          // Modify the message if an item is out of stock
          if (itemInfo.Quantity > 0)
                    lblQty.Text = itemInfo.Quantity.ToString();
          else {
                    lblQty.Text = MSG_BACK_ORDERED;
                    lblQty.CssClass = CSS_ALERT;
          }
    }else{
          lblSearchResults.Text = "Item not found";
    }
}
```

BLL.Item



```
public IList GetItem(string itemId)
{
   // Validate input
   if (itemId.Trim() == string.Empty)
        return null;

   // Get an instance of the Item DAL using the DALFactory
   IItem dal = PetShop.DALFactory.Item.Create();

   // Use the dal to search by itemId
   return dal.GetItem(itemId);
}
```

DAL.Item

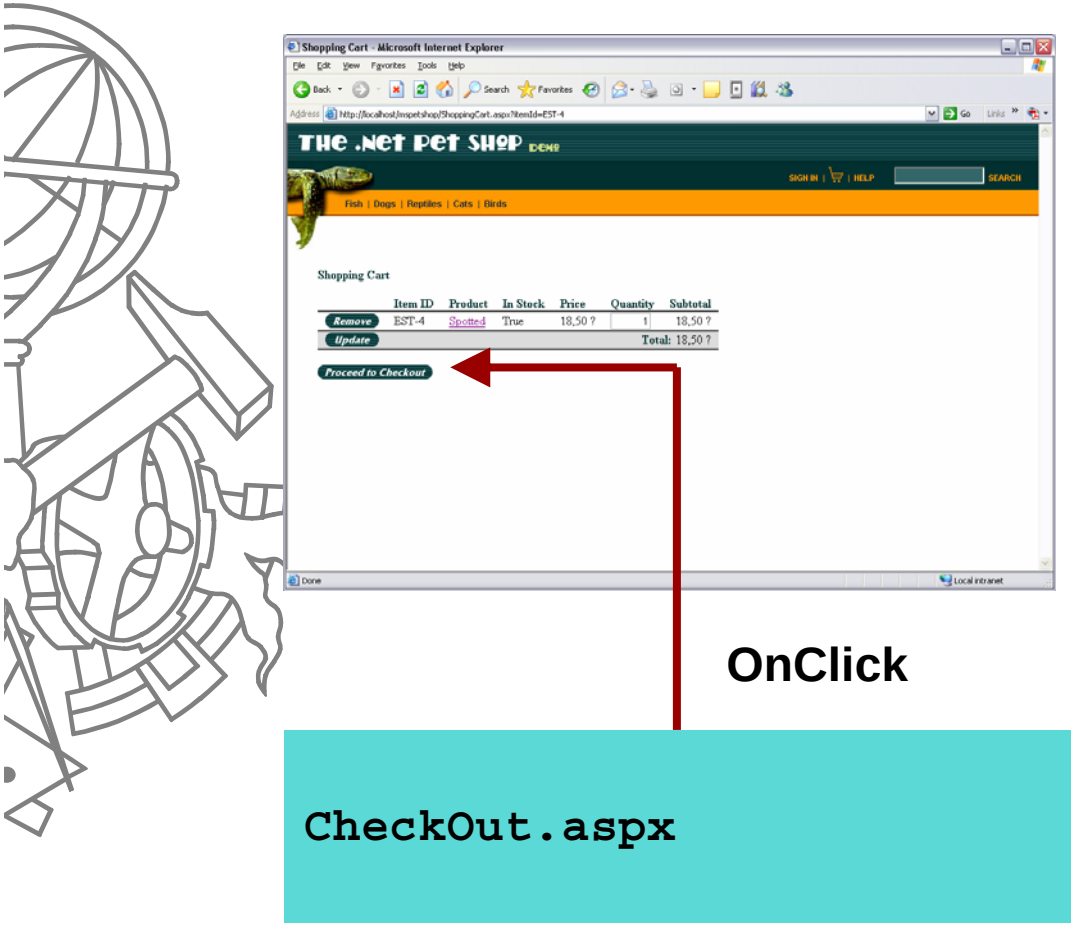


```
public ItemInfo GetItem(string itemId)
{
    //Set up a return value
    ItemInfo item = null;

    //Create a parameter
    SqlParameter parm = new SqlParameter(PARM_ITEM_ID, SqlDbType.Char,
                                                    10);
    //Bind the parameter
    parm.Value = itemId;

    //Execute the query
    using (SqlDataReader rdr =
           SQLHelper.ExecuteReader(SQLHelper.CONN_STRING_NON_DTC,
           CommandType.Text, SQL_SELECT_ITEM, parm))
    {
        rdr.Read();
        item = new ItemInfo(rdr.GetString(0).Trim(), rdr.GetString(1),
                            rdr.GetInt32(2), rdr.GetDecimal(3),
                            rdr.GetString(4), rdr.GetString(5));

    }
    return item;
}
```

SQL

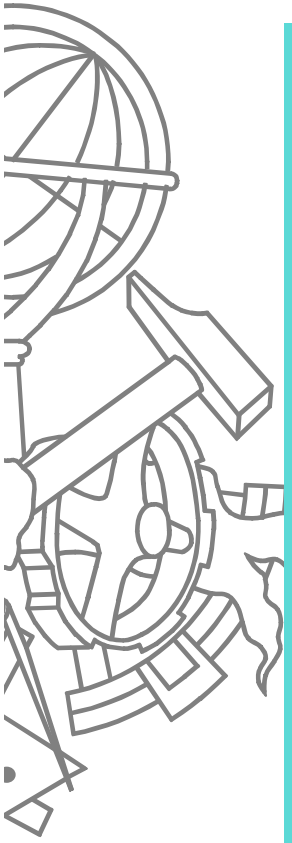
```
SELECT Item.ItemId, Item.Attr1,
   Inventory.Qty, Item.ListPrice,
   Product.Name, Product.Descn
FROM Item INNER JOIN Inventory ON
   Item.ItemId = Inventory.ItemId INNER
   JOIN Product ON Item.ProductId =
   Product.ProductId
WHERE Item.ItemId = @ItemId
```

Visualizar carrinho de compras



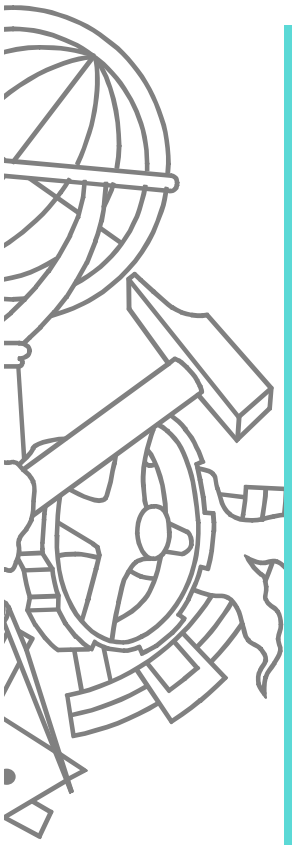
- ShoppingCart.aspx
- Se receber parâmetro adiciona um item ao carrinho
  - /ShoppingCart.aspx?itemId=EST-4
- Utiliza classes controladoras do processo `CartController` e `AccountController`

Web.ShoppingCart



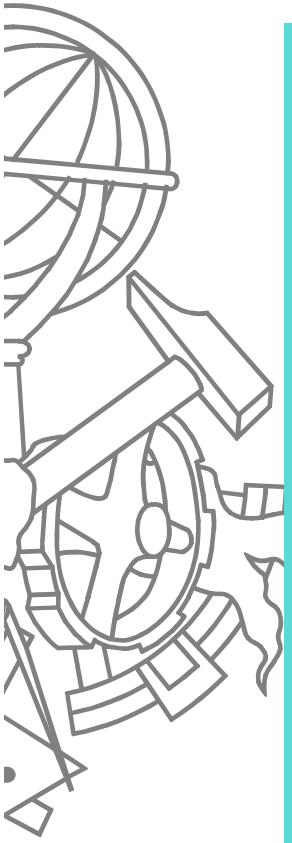
```
override protected void OnLoad(EventArgs e){
    // Create an instance of the cart controller
    ProcessFlow.CartController cartController = new
                                        ProcessFlow.CartController();
    myCart = cartController.GetCart(true);
    if (!Page.IsPostBack){
        string itemId = Request["itemId"];
        if (itemId != null){
            itemId = WebComponents.CleanString.InputText(itemId, 50);
            myCart.Add(itemId);
            cartController.StoreCart(myCart);
            }
        }
    //Get an account controller
    ProcessFlow.AccountController accountController = new
                                        ProcessFlow.AccountController();
    //Get the user's favourite category
    string favCategory = accountController.GetFavouriteCategory();
    //If we have a favourite category, render the favourites list
    if (favCategory != null){
        favorites.Visible = true;
        ViewState[KEY_CATEGORY] = favCategory;
    }
    Refresh();
}
```

BLL.Cart



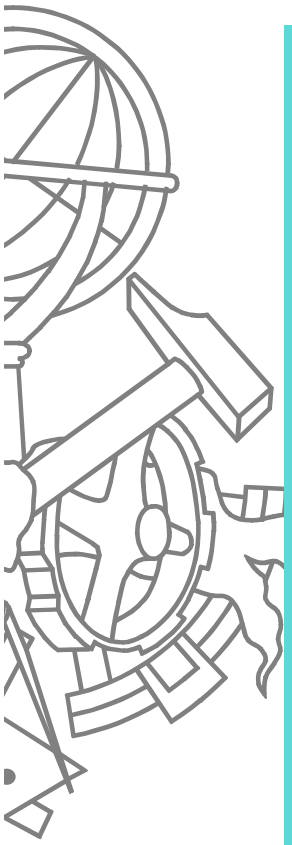
```
public void Add(string ItemId)
{
   foreach (CartItemInfo cartItem in _items)
   {
      if (ItemId == cartItem.ItemId)
      {
         cartItem.Quantity++;
         cartItem.InStock = (GetInStock(ItemId) -
               cartItem.Quantity) >= 0 ? true : false;
         _total = _total+(cartItem.Price*cartItem.Quantity);
         return;
      }
   }

   Item item = new Item();
   ItemInfo data = item.GetItem(ItemId);
   CartItemInfo newItem = new CartItemInfo(ItemId,data.Name,
            (data.Quantity >= 1), 1, (decimal)data.Price);
   _items.Add(newItem);
   _total = _total+(data.Price);
}
```

Web.ProcessFlow.CartController



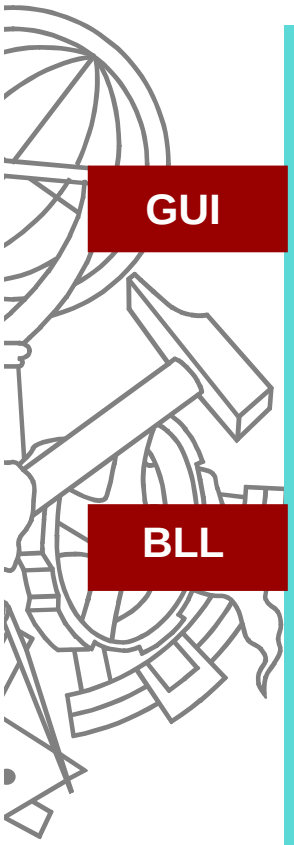
```
public Cart GetCart(bool create)
{
   // Fetch the cart object from session state
   Cart myCart =
            (Cart)HttpContext.Current.Session[CART_KEY];
   if ( myCart == null ){
      if (create){
         myCart = new Cart();
      }else{
         HttpContext.Current.Server.Transfer(URL_NOCART);
         return null;
      }
   }
   return myCart;
}

public void StoreCart(Cart cart)
{
   // Store the cart object in session state
   HttpContext.Current.Session[CART_KEY] = cart;
}
```

Web.ProcessFlow.AccountController



```
public string GetFavouriteCategory()
{
   AccountInfo myAccount =
      (AccountInfo)HttpContext.Current.Session[ACCOUNT_KEY];

   if (myAccount != null && myAccount.IsShowFavorites)
   {
      return myAccount.Category;
   }
   else
   {
      return null;
   }
}
```

Web.ShoppingCart

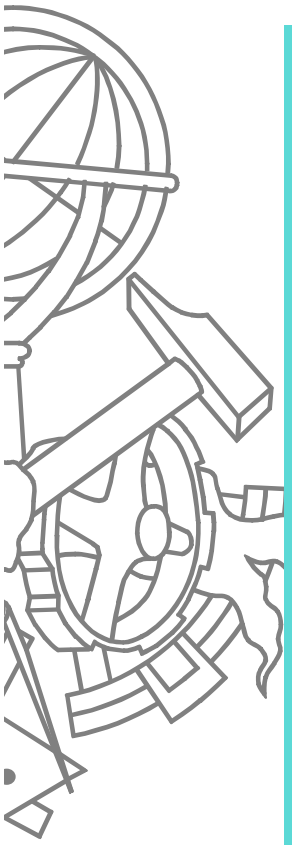


GUI

BLL

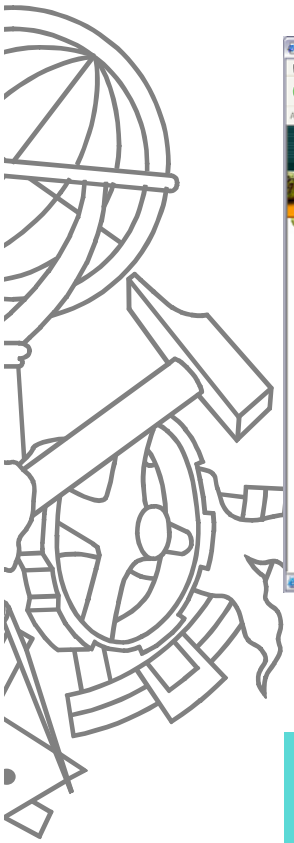
```
override protected void CommandClicked(object sender,
                                       RepeaterCommandEventArgs e) {
    if (e.CommandName == CMD_UPDATE) { // Check for update button
        // Go through each item on the page
        for (int i = 0, j = cart.Items.Count; i < j; i++) {
            // lookup the control
            textBox txt = (TextBox)cart.Items[i].FindControl(ID_TXT);
            try {
                int qty = int.Parse(txt.Text);
                int index = cart.CurrentPageIndex * cart.PageSize + i;
                if (qty <= 0) // remove the item from the cart
                    myCart.RemoveAt(index);
                else          // Update the item with the new quantity
                    myCart[index].Quantity = qty;
            }
            catch {}
        }
    }else // otherwise the command is to remove the an item
        myCart.Remove((string)e.CommandArgument);
    Refresh();
    int pageCount = (myCart.Count - 1) / cart.PageSize;
    cart.SetPage(Math.Min(cart.CurrentPageIndex, pageCount));
}
```

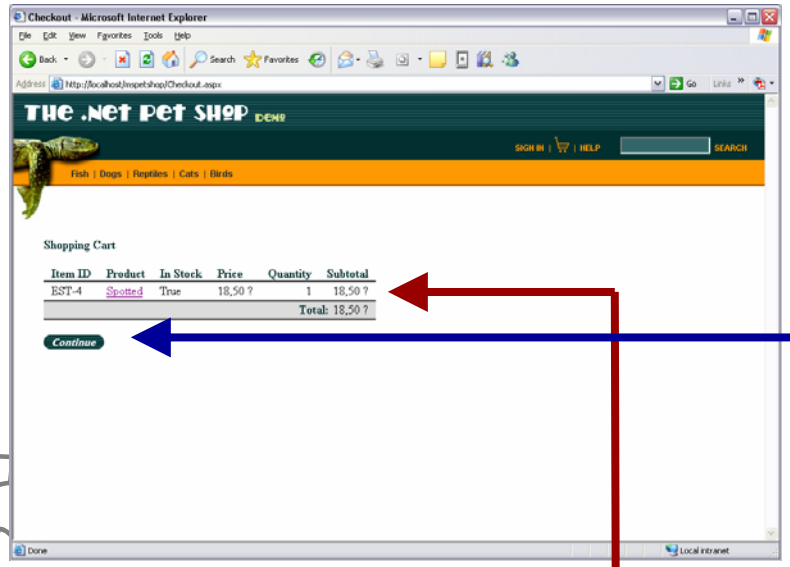
BLL.Cart



```
public void RemoveAt(int index)
{
   CartItemInfo item = (CartItemInfo)_items[index];
   _total = _total-(item.Price*item.Quantity);
   _items.RemoveAt(index);
}

public void Remove(string itemId)
{
   foreach (CartItemInfo item in _items)
   {
      if (itemId == item.ItemId)
      {
         _items.Remove(item);
         _total = _total-(item.Price*item.Quantity);
         return;
      }
   }
}
```

Conferir encomenda





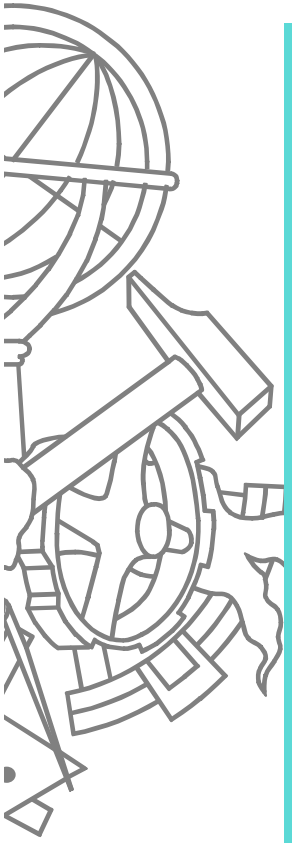
- CheckOut.aspx

OrderBilling.aspx

DataSource

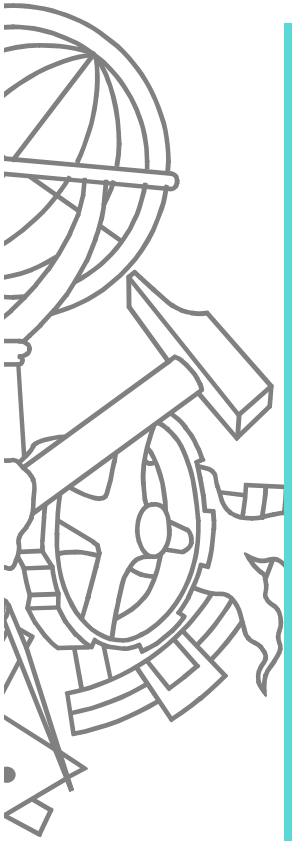
```
// Create an instance of the cart controller
ProcessFlow.CartController cartController = new
                    ProcessFlow.CartController();
// Fetch the cart state from the controller
myCart = cartController.GetCart(false);
```

Web.OrderBilling



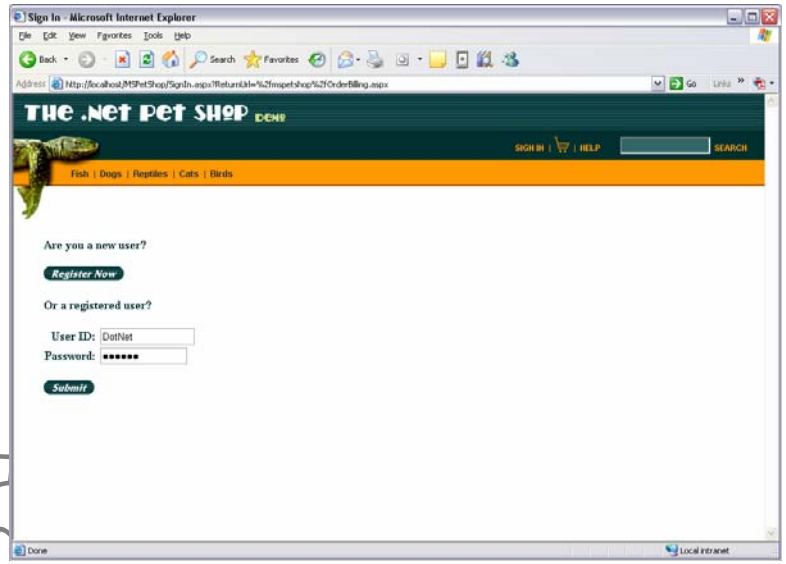
```
protected void override OnLoad(EventArgs e)
{
    if (!IsPostBack)
    {
        enterAddress.Visible = true;
        confirmAddress.Visible = false;

        ProcessFlow.AccountController accountController = new
                                    ProcessFlow.AccountController();

        AccountInfo myAccount= accountController.GetAccountInfo(true);

        if (myAccount != null)
        {
            Account account = new Account();
            billAddr.Address = account.GetAddress(myAccount.UserId);
        }
    }
}
```

Web.ProcessFlow.AccountController



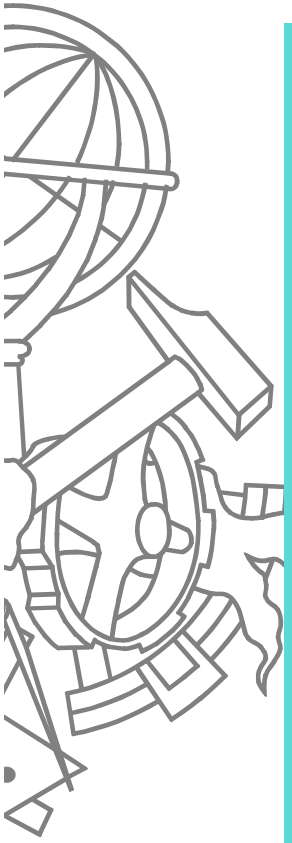
```
public AccountInfo GetAccountInfo(bool required)
{
    AccountInfo myAccount =
            (AccountInfo)HttpContext.Current.Session[ACCOUNT_KEY];

    if (myAccount == null)
    {
        if(required)
        {
            HttpContext.Current.Response.Redirect(URL_SIGNIN, true);
        }
        return null;
    }
    else
    {
        return myAccount;
    }
}
```

Efectuar login



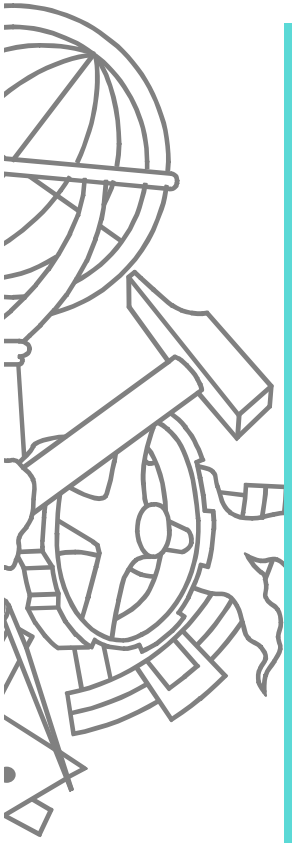
- SignIn.aspx
- Usa parametro para indicar o URL de retorno através de mecanismo de autentiação do ASP.net
  - /SignIn.aspx?ReturnUrl=%2fmspetshop%2fOrderBilling.aspx

Web.SignIn



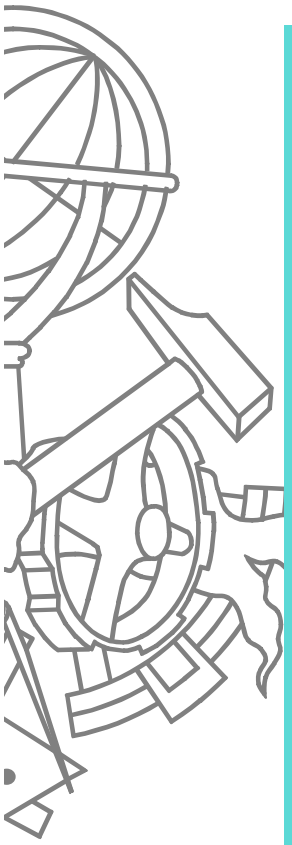
```
protected void SubmitClicked(object sender,
                             ImageClickEventArgs e)
{
    if (Page.IsValid)
    {
        // Get the user info from the text boxes
        string userId =
    WebComponents.CleanString.InputText(txtUserId.Text, 50);
        string password =
    WebComponents.CleanString.InputText(txtPassword.Text, 50);

        // Hand off to the account controller for naviagtion
        ProcessFlow.AccountController accountController = new
                  ProcessFlow.AccountController();

        if (!accountController.ProcessLogin(userId, password))
        {
            // If we fail to login let the user know
            valUserId.ErrorMessage = MSG_FAILURE;
            valUserId.IsValid = false;
        }
    }
}
```

Web.ProcessFlow.AccountController



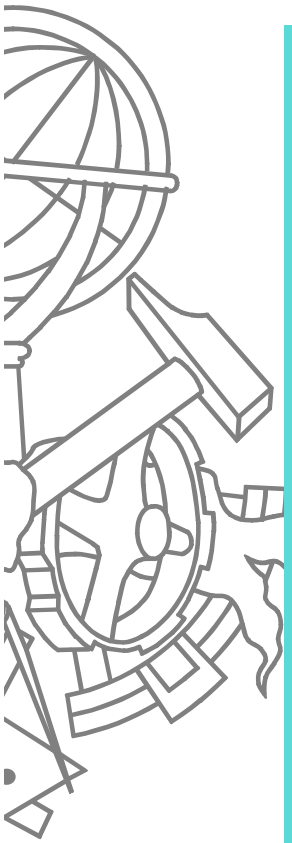
```
protected bool ProcessLogin(string userId, string password){
    // Use the account business logic layer to login
    Account account = new Account();
    AccountInfo myAccountInfo = account.SignIn(userId, password);
    //If login is successful then store the state in session and redirect
    if (myAccountInfo != null) {
        HttpContext.Current.Session[ACCOUNT_KEY] = myAccountInfo;
        // Determine where to redirect the user back too
        // If they came in from the home page, take them to a similar page
        if (FormsAuthentication.GetRedirectUrl(userId, false
                                        ).EndsWith(URL_DEFAULT)) {
            FormsAuthentication.SetAuthCookie(userId, false);
            HttpContext.Current.Response.Redirect(URL_ACCOUNTSIGNIN, true);
        } else {
            // Take the customer back to where the came from
            FormsAuthentication.SetAuthCookie(userId, false);
            HttpContext.Current.Response.Redirect(
                FormsAuthentication.GetRedirectUrl(userId, false), true);
        }
        return true;
    } else {
        // Login has failed so return false
        return false;
    }
}
```

BLL.Account



```
public AccountInfo SignIn(string userId, string password)
{
    // Validate input
    if ((userId.Trim() == string.Empty) ||
        (password.Trim() == string.Empty))
        return null;

    // Get an instance of the account DAL using the DALFactory
    IAccount dal = PetShop.DALFactory.Account.Create();

    // Try to sign in with the given credentials
    AccountInfo account = dal.SignIn(userId, password);

    // Return the account
    return account;
}
```

DAL.Account

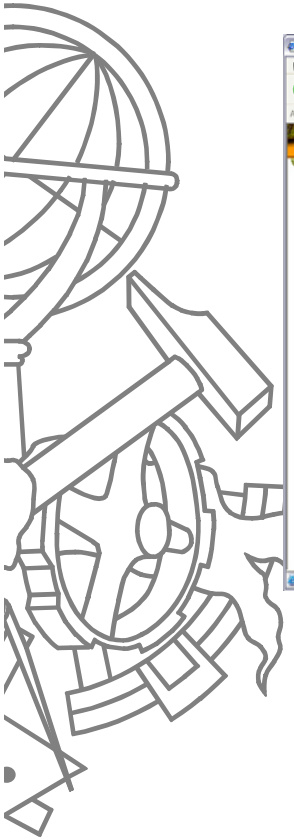


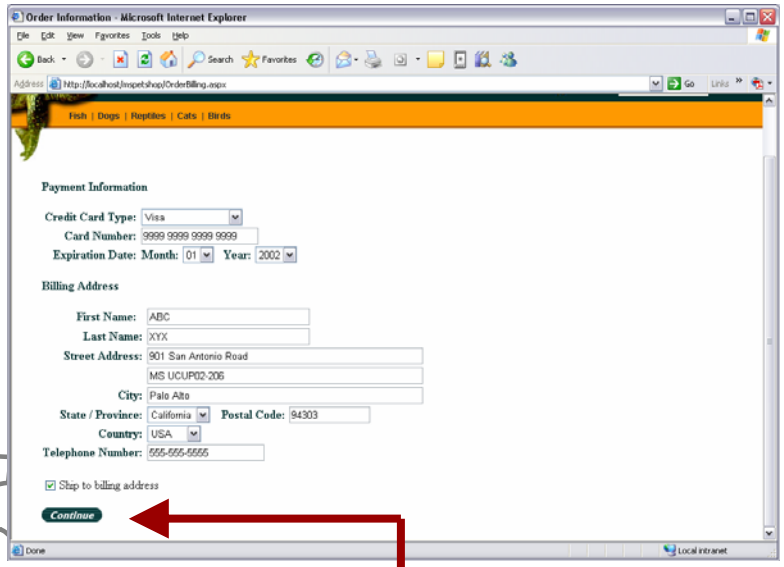
```
public AccountInfo SignIn(string userId, string password) {
    SqlParameter[] signOnParms = GetSignOnParameters();
    signOnParms[0].Value = userId;
    signOnParms[1].Value = password;

    using (SqlDataReader rdr =
           SQLHelper.ExecuteReader(SQLHelper.CONN_STRING_NON_DTC,
           CommandType.Text, SQL_SELECT_ACCOUNT, signOnParms)) {
        if (rdr.Read()) {
           AddressInfo myAddress = new AddressInfo(rdr.GetString(1),
              rdr.GetString(2), rdr.GetString(3), rdr.GetString(4),
              rdr.GetString(5), rdr.GetString(6), rdr.GetString(7),
              rdr.GetString(8), rdr.GetString(9));
           return new AccountInfo(userId, password, rdr.GetString(0),
                  myAddress, rdr.GetString(10), rdr.GetString(11),
                  Convert.ToBoolean(rdr.GetInt32(12)),
                  Convert.ToBoolean(rdr.GetInt32(13)));
        }
    return null;
    }
}
```

SQL

```
SELECT Account.Email, Account.FirstName,
    Account.LastName, Account.Addr1,
    Account.Addr2, Account.City, Account.State,
    Account.Zip, Account.Country, Account.Phone,
    Profile.LangPref, Profile.FavCategory,
    Profile.MyListOpt, Profile.BannerOpt
FROM Account INNER JOIN Profile ON
    Account.UserId = Profile.UserId INNER JOIN
    SignOn ON Account.UserId =
    SignOn.UserName
WHERE SignOn.UserName = @UserId AND
    SignOn.Password = @Password
```

Introduzir dados de pagamento



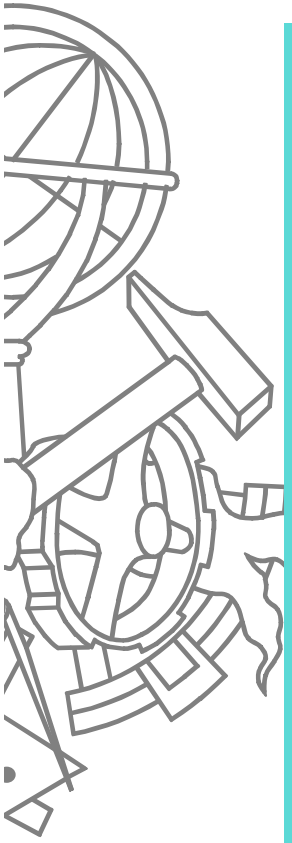


- OrderBilling.aspx

OnClick

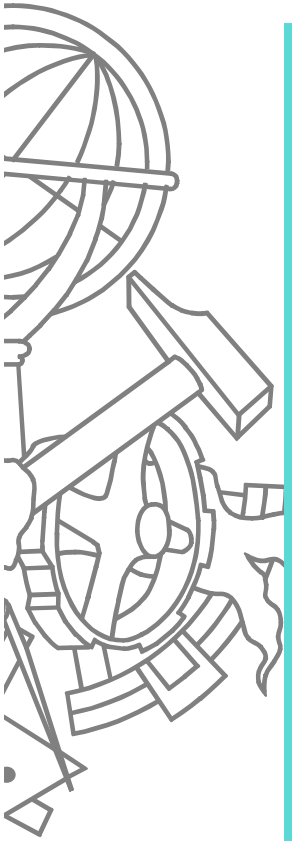
```
ContinueClicked
```

Web.OrderBilling



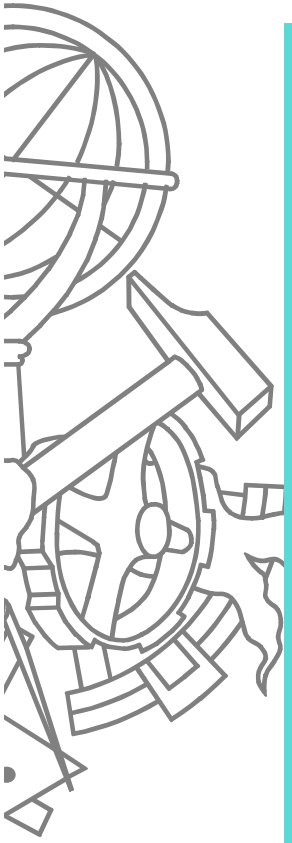
```
protected void override OnLoad(EventArgs e)
{
    if (!IsPostBack)
    {
        enterAddress.Visible = true;
        confirmAddress.Visible = false;

        ProcessFlow.AccountController accountController = new
                                    ProcessFlow.AccountController();

        AccountInfo myAccount= accountController.GetAccountInfo(true);

        if (myAccount != null)
        {
            Account account = new Account();
            billAddr.Address = account.GetAddress(myAccount.UserId);
        }
    }
}
```

Web.OrderBilling



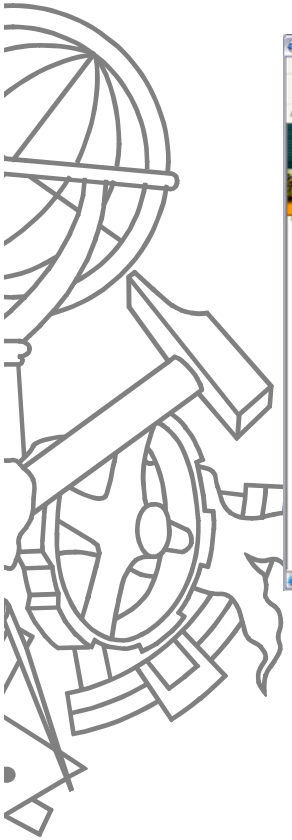
```
protected void ContinueClicked(object sender, ImageClickEventArgs e) {
    if (Page.IsValid) {
        // Create a new cartController object
        ProcessFlow.CartController cartController = new
                                                ProcessFlow.CartController();
        // Fetch the creditcard info and store it
        string cardType =
        WebComponents.CleanString.InputText(listCardType.SelectedItem.Text, 10);
        string cardNumber =
            WebComponents.CleanString.InputText(txtCardNumber.Text, 20);
        string cardYear =
            WebComponents.CleanString.InputText(listYear.SelectedItem.Text, 4);
        string cardMonth =
            WebComponents.CleanString.InputText(listMonth.SelectedItem.Text, 2);
        CreditCardInfo creditCard = new CreditCardInfo(cardType, cardNumber,
                    string.Format(FORMAT_EXPIRATION, cardMonth, cardYear));
        cartController.StoreCreditCard(creditCard);
        AddressInfo billingAddress = billAddr.Address;
        // Now store the billing information
        cartController.StoreBillingAddress(billAddr.Address);
        // Continue with the order process
        cartController.ContinueOrder(chkShipBilling.Checked);

        enterAddress.Visible = false;
        confirmAddress.Visible = true;
        staticAddressBilling.ShowAddress(billingAddress);
        staticAddressShipping.ShowAddress(billingAddress);
    }
}
```

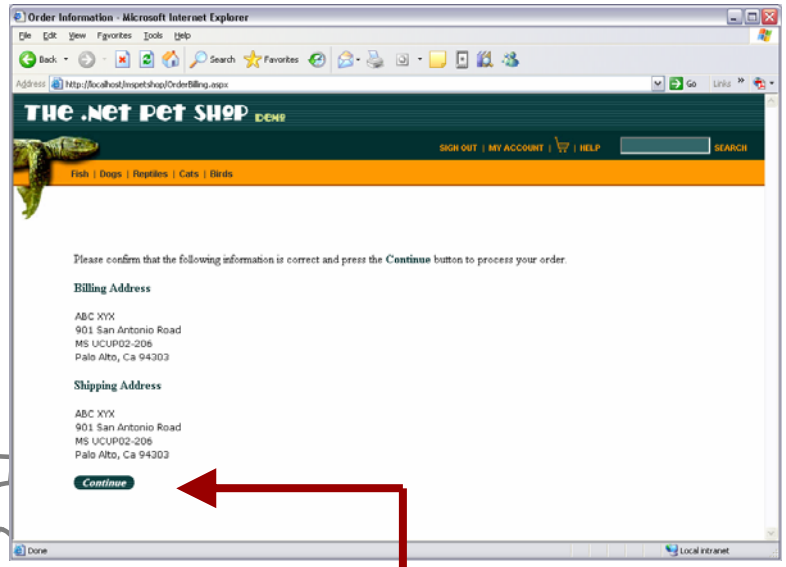
Web.ProcessFlow.CartController



```
public void StoreCreditCard(CreditCardInfo creditCard)
{
    HttpContext.Current.Session[CREDITCARD_KEY] = creditCard;
}

public void StoreBillingAddress(AddressInfo billingAddress)
{
    HttpContext.Current.Session[BILLING_KEY] = billingAddress;
}

public void ContinueOrder(bool useBillingAddress)
{
    // Set the billing address depending on what the user has selected
    if (useBillingAddress)
    {
        HttpContext.Current.Session[SHIPPING_KEY] =
                HttpContext.Current.Session[BILLING_KEY];
    }
    else
    {
        //If the user wants to use a different address then take them to the
    next page
        HttpContext.Current.Response.Redirect("OrderShipping.aspx", true);
    }
}
```

Conferir nota de débito



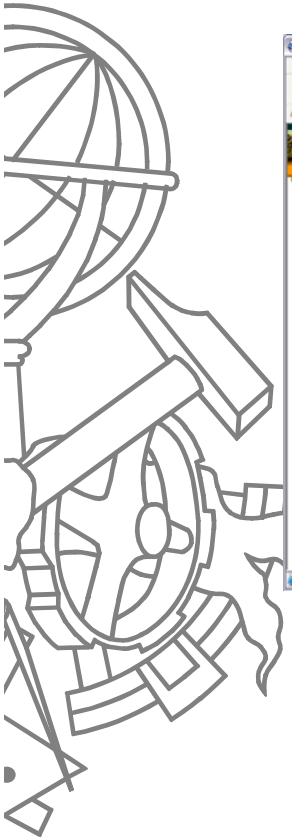


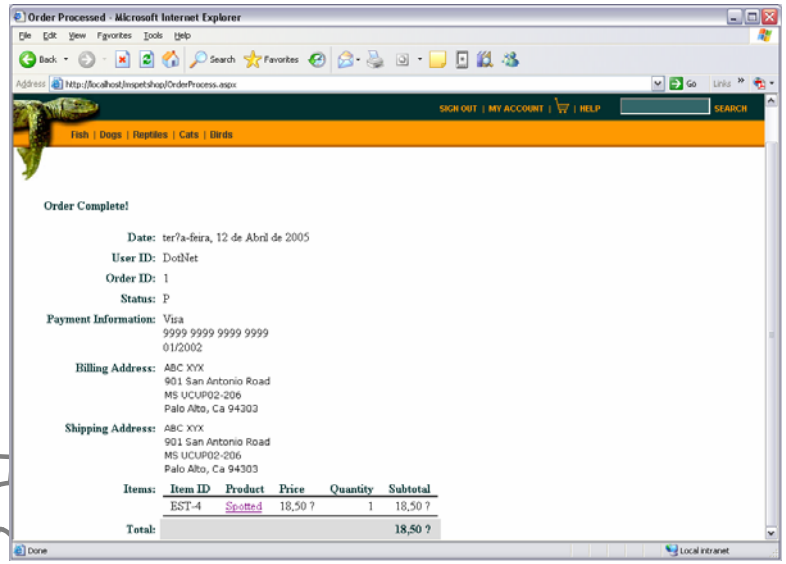
- OrderBilling.aspx

OnClick

`OrderProcess.aspx`

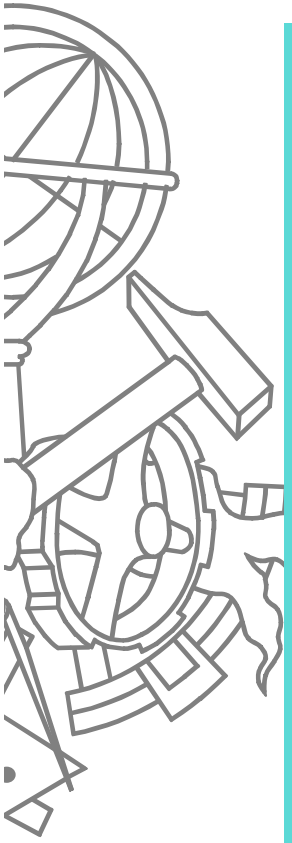
Visualizar encomenda efectuada





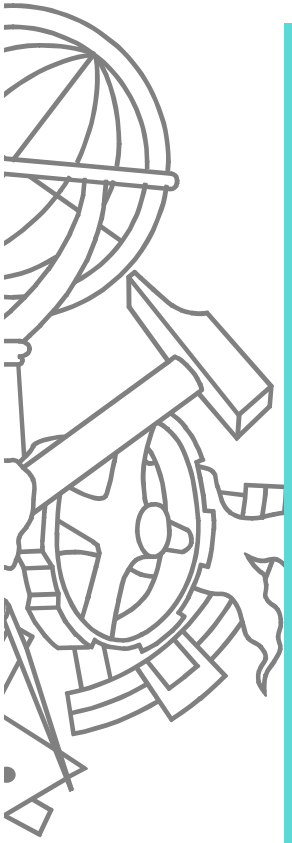
- OrderProcess.aspx

Web.OrderProcess



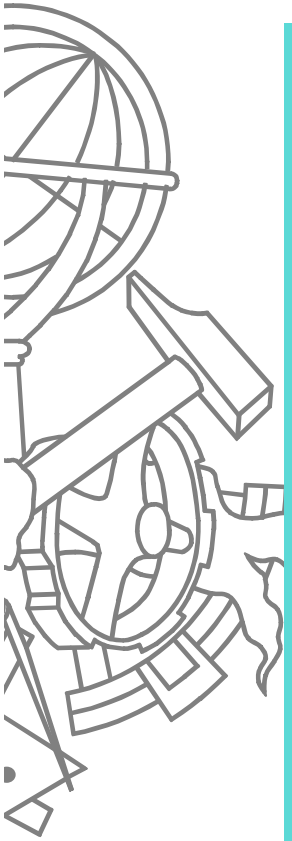
```
override protected void OnLoad(EventArgs e)
{
    ProcessFlow.CartController cartController = new
                                        ProcessFlow.CartController();
    OrderInfo newOrder = cartController.PurchaseCart();

    //Display the order info to the user
    lblOrderId.Text = newOrder.OrderId.ToString();
    lblOrderDate.Text = newOrder.Date.ToLongDateString();;
    lblUserId.Text = newOrder.UserId;
    lblCardType.Text = newOrder.CreditCard.CardType;
    lblCardNumber.Text = newOrder.CreditCard.CardNumber;
    lblCardExpiration.Text = newOrder.CreditCard.CardExpiration;
    statAddrBill.address = newOrder.BillingAddress;
    statAddrShip.address = newOrder.ShippingAddress;

    cart.DataSource = newOrder.LineItems;
    cart.DataBind();
    lblOrderTotal.Text = newOrder.OrderTotal.ToString("c");
}
```

Web.ProcessFlow.CartController (1)



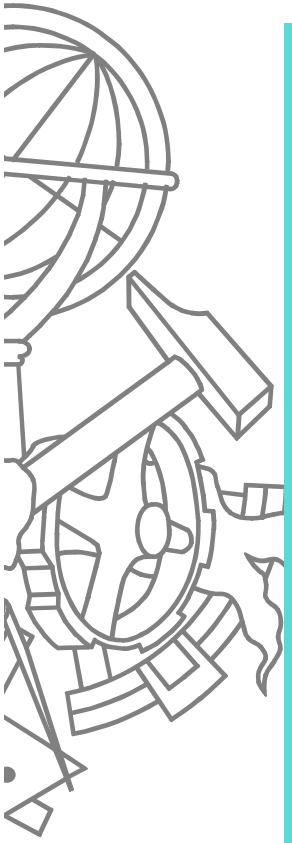
```
public OrderInfo PurchaseCart()
{
    // Fetch the cart from session
    Cart myCart = (Cart)HttpContext.Current.Session[CART_KEY];
    // Make some checks on the cart
    if ( ( myCart == null ) || ( myCart.Count==0 ) )
    {
        HttpContext.Current.Server.Transfer(URL_NOCART);
        return null;
    }
    else
    {
        // Build up the order
        OrderInfo newOrder = new OrderInfo();

        newOrder.UserId = ((AccountInfo)
                  HttpContext.Current.Session[ACCOUNT_KEY]).UserId;
        newOrder.CreditCard = (CreditCardInfo)
                  HttpContext.Current.Session[CREDITCARD_KEY];
        newOrder.BillingAddress = (AddressInfo)
                  HttpContext.Current.Session[BILLING_KEY];
```

Web.ProcessFlow.CartController (2)



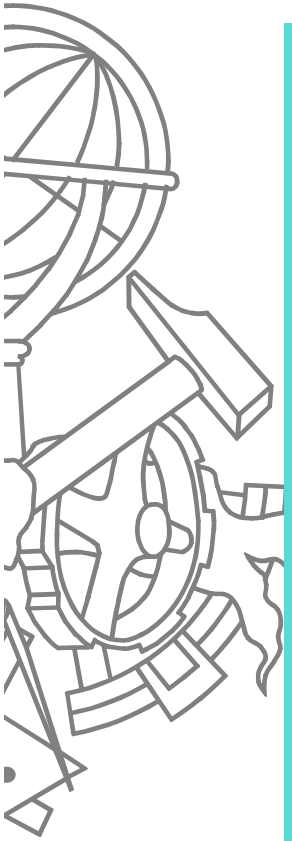
```
        newOrder.ShippingAddress = (AddressInfo)
                            HttpContext.Current.Session[SHIPPING_KEY];

        newOrder.LineItems = (LineItemInfo[])
            myCart.GetOrderLineItems().ToArray(typeof(LineItemInfo));

        newOrder.OrderTotal = myCart.Total;
        newOrder.Date = DateTime.Now;

        // Send the order to the middle tier
        OrderInsert order = new OrderInsert();
        newOrder.OrderId = order.Insert(newOrder);

        // clear the session objects used
        HttpContext.Current.Session[CART_KEY] = null;
        HttpContext.Current.Session[CREDITCARD_KEY] = null;
        HttpContext.Current.Session[BILLING_KEY] = null;
        HttpContext.Current.Session[SHIPPING_KEY] = null;

        return newOrder;
    }
}
```

BLL.OrderInsert



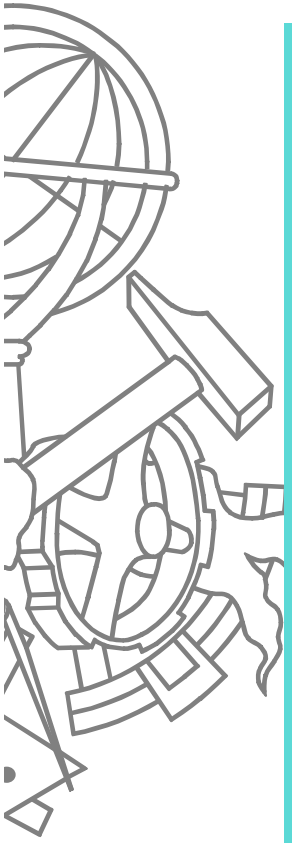
```
[Transaction(System.EnterpriseServices.TransactionOption.Required)]
[ClassInterface(ClassInterfaceType.AutoDispatch)]
[ObjectPooling(MinPoolSize=4, MaxPoolSize=4)]
[Guid("14E3573D-78C8-4220-9649-BA490DB7B78D")]
public class OrderInsert : ServicedComponent
{
    [AutoComplete]
    public int Insert(OrderInfo order) {
        // Get an instance of the Order DAL using the DALFactory
        IOrder dal = PetShop.DALFactory.Order.Create();
        // Call the insert method in the DAL to insert the header
        int orderId = dal.Insert(order);
        // Get an instance of the Inventory business component
        Inventory inventory = new Inventory();
        inventory.TakeStock( order.LineItems);
        // As part of the sample application we have created a user
        // you can tested distributed transactions with
        // If the order has been created with the user 'Acid',
        // then throw an exception which will rollback the entire
    transaction
        if (order.UserId == ACID_USER_ID)
            throw new ApplicationException(ACID_ERROR_MSG);
        // Set the orderId so that it can be returned to the caller
        return orderId;
    }
}
```

BLL.Inventory



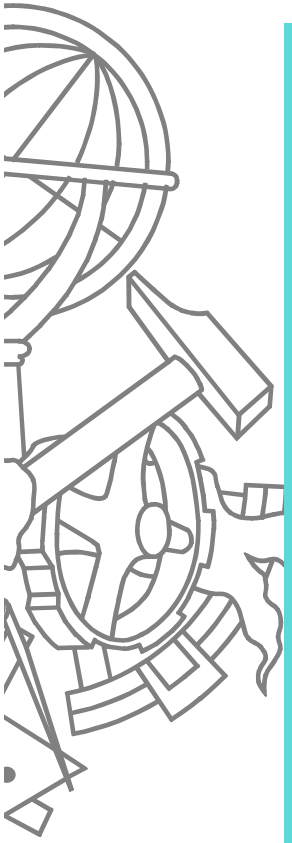
```
public void TakeStock(LineItemInfo[] items)
{
   // Get an instance of the Inventory DAL using the
   DALFactory
   IInventory dalc = PetShop.DALFactory.Inventory.Create();

   // Reduce the stock level in the data store
   dalc.TakeStock(items);
}
```

DAL.Order (1)



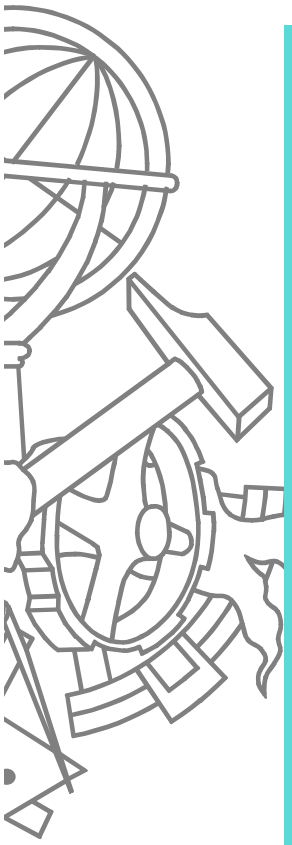
```
public int Insert(OrderInfo order)
{
    int orderId = 0;
    string strSQL = null;
    try {
        // Get each commands parameter arrays
        SqlParameter[] orderParms = GetOrderParameters();
        SqlParameter statusParm = new SqlParameter(PARM_ORDER_ID,
                                                   SqlDbType.Int);
        SqlCommand cmd = new SqlCommand();

        // Set up the parameters
        orderParms[0].Value = order.UserId;
        orderParms[1].Value = order.Date;
        orderParms[2].Value = order.ShippingAddress.Address1;
        orderParms[3].Value = order.ShippingAddress.Address2;
        orderParms[4].Value = order.ShippingAddress.City;
        orderParms[5].Value = order.ShippingAddress.State;
        orderParms[6].Value = order.ShippingAddress.Zip;
        orderParms[7].Value = order.ShippingAddress.Country;
        orderParms[8].Value = order.BillingAddress.Address1;
        orderParms[9].Value = order.BillingAddress.Address2;
        orderParms[10].Value = order.BillingAddress.City;
        orderParms[11].Value = order.BillingAddress.State;
        orderParms[12].Value = order.BillingAddress.Zip;
        orderParms[13].Value = order.BillingAddress.Country;
```

DAL.Order (2)



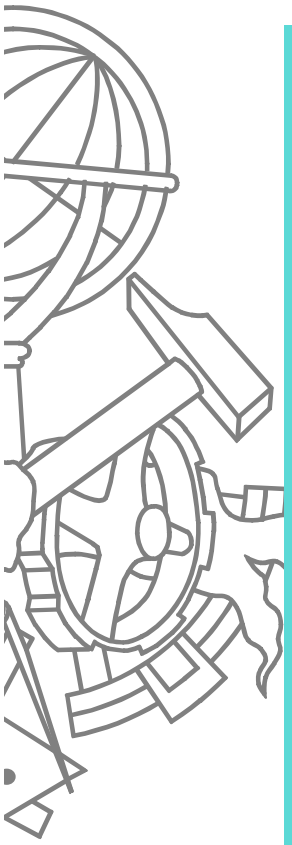
```
orderParms[14].Value = order.OrderTotal;
orderParms[15].Value = order.BillingAddress.FirstName;
orderParms[16].Value = order.BillingAddress.LastName;
orderParms[17].Value = order.ShippingAddress.FirstName;
orderParms[18].Value = order.ShippingAddress.LastName;
orderParms[19].Value = order.CreditCard.CardNumber;
orderParms[20].Value = order.CreditCard.CardExpiration;
orderParms[21].Value = order.CreditCard.CardType;

foreach (SqlParameter parm in orderParms)
    cmd.Parameters.Add(parm);

// Create the connection to the database
using (SqlConnection conn = new
            SqlConnection(SQLHelper.CONN_STRING_DTC_ORDERS))
{
    // Insert the order status
    strSQL = SQL_INSERT_ORDER;
    SqlParameter[] itemParms ;
```

DAL.Order (3)



```
        // For each line item, insert an orderline record
        int i = 0;
        foreach (LineItemInfo item in order.LineItems)
        {
            strSQL = strSQL + SQL_INSERT_ITEM + " @ID" + ",
@LineNumber"+i + ", @ItemId" + i+ ", @Quantity" + i + ", @Price" + i +
"); SELECT @ERR=@ERR+@@ERROR;";

            //Get the cached parameters
            itemParms = GetItemParameters(i);

            itemParms[0].Value = item.Line;
            itemParms[1].Value = item.ItemId;
            itemParms[2].Value = item.Quantity;
            itemParms[3].Value = item.Price;

            //Bind each parameter
            foreach (SqlParameter parm in itemParms)
                cmd.Parameters.Add(parm);
            i++;
        }
```

DAL.Order (4)

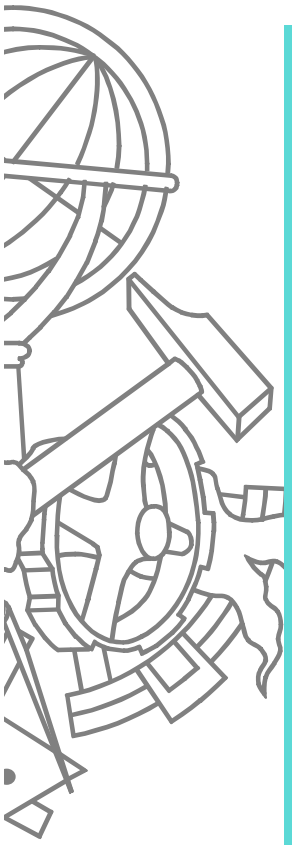


```
            conn.Open();
            cmd.Connection = conn;
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = strSQL + "SELECT @ID, @ERR";
            // Read the output of the query, should return orderid and
    error count
            using (SqlDataReader rdr =
                cmd.ExecuteReader(CommandBehavior.CloseConnection)) {
                //Read the result
                rdr.Read();
                // If the error count is not zero throw an exception
                if (rdr.GetInt32(1) != 0)
                    throw new Exception("DATA INTEGRITY ERROR ON ORDER
    INSERT - ROLLBACK ISSUED");
                //Fetch the orderId
                orderId = rdr.GetInt32(0);
            }
            //Clear the parameters
            cmd.Parameters.Clear();
        }
    } catch(Exception e) {
        throw e;
    } finally {
    }
    return orderId;
}
```

SQL

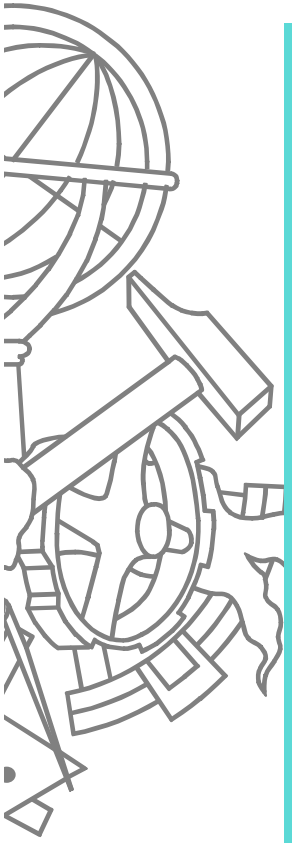
```
Declare @ID int;
Declare @ERR int;
INSERT INTO Orders
VALUES(@UserId, @Date, @ShipAddress1,
    @ShipAddress2, @ShipCity, @ShipState, @ShipZip,
    @ShipCountry, @BillAddress1, @BillAddress2, @BillCity,
    @BillState, @BillZip, @BillCountry, 'UPS', @Total,
    @BillFirstName, @BillLastName, @ShipFirstName,
    @ShipLastName, @CardNumber, @CardExpiration,
    @CardType, 'US_en');
SELECT @ID=@@IDENTITY;
INSERT INTO OrderStatus
VALUES(@ID, @ID, GetDate(), 'P');
SELECT @ERR=@@ERROR;

INSERT INTO LineItem VALUES(
```

DAL.Inventory (1)



```
Public void TakeStock(LineItemInfo[] items) {
    SqlParameter[] inventoryParms;
    SqlCommand cmd = new SqlCommand();
    //Open a connection
    using (SqlConnection conn = new
                  SqlConnection(SQLHelper.CONN_STRING_DTC_INV))
    {
        string strSQL = null;
        int i = 0;
        //Append a statement to the batch for each item in the array
        foreach (LineItemInfo item in items)
        {
            strSQL = strSQL + SQL_TAKE_INVENTORY;
            inventoryParms = GetInventoryParameters(i);
            strSQL = strSQL + "@Quantity" +i + " WHERE ItemId =
    @ItemId" + i+ ";";
            //Bind parameters
            inventoryParms[0].Value = item.Quantity;
            inventoryParms[1].Value = item.ItemId;

            foreach (SqlParameter parm in inventoryParms)
                cmd.Parameters.Add(parm);
            i++;
        }
```

DAL.Inventory (2)



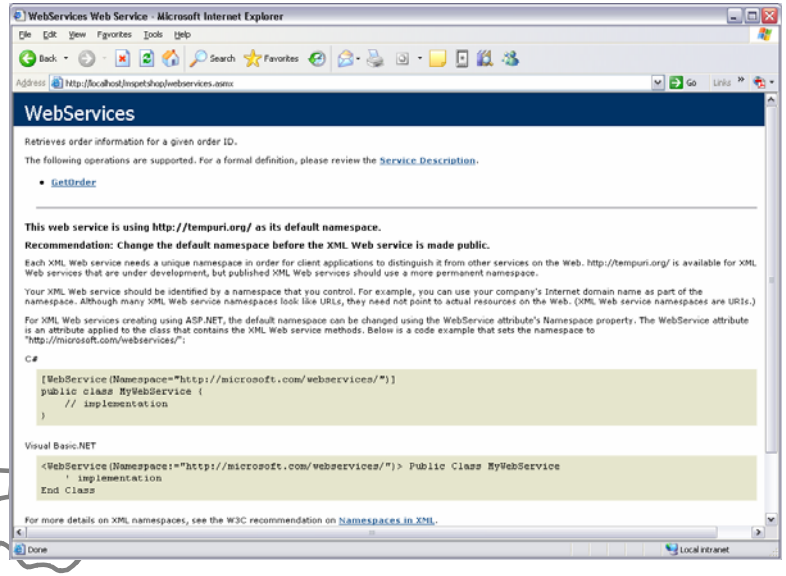
```
        // Open the connection
        conn.Open();

        //Set up the command
        cmd.Connection = conn;
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = strSQL;

        //Execute the query
        cmd.ExecuteNonQuery();
        cmd.Parameters.Clear();
    }
}
```

SQL

UPDATE Inventory

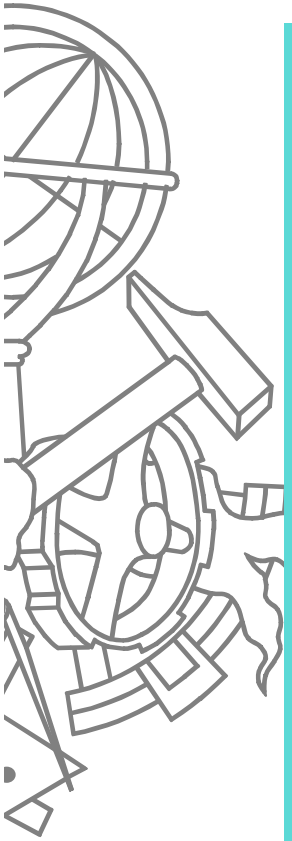
SET Qty = Qty -

Web Service



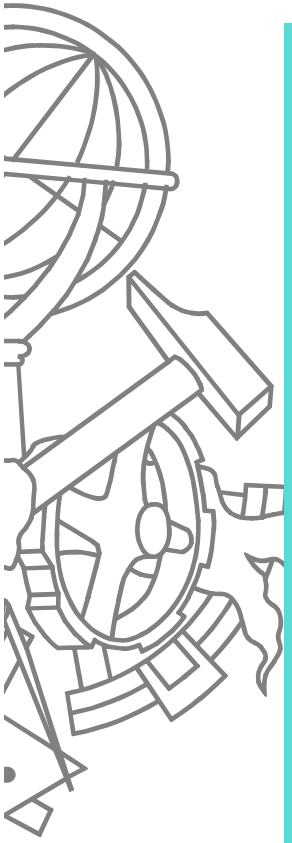
- Webservice.asmx
- ASP.net gera automaticamente página de teste e WSDL

Web.WebServices



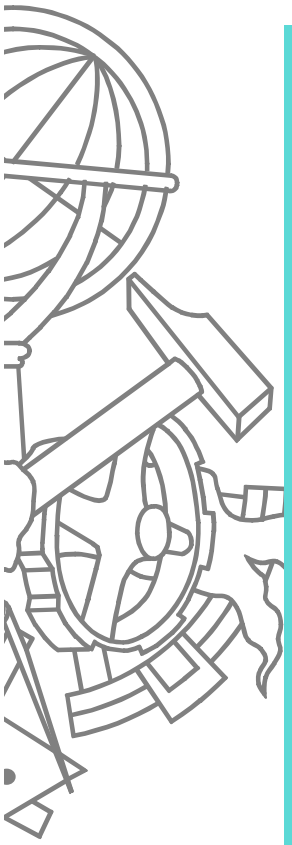
```
[WebService(Description="Retrieves order information for a
   given order ID.")]
public class WebServices : WebService
{
    [WebMethod]
    public OrderInfo GetOrder(int orderId)
    {
        // Use the order component optimized for reads
        OrderRead orderWS = new OrderRead();

        return orderWS.GetOrder(orderId);
    }
}
```

BLL.OrderRead



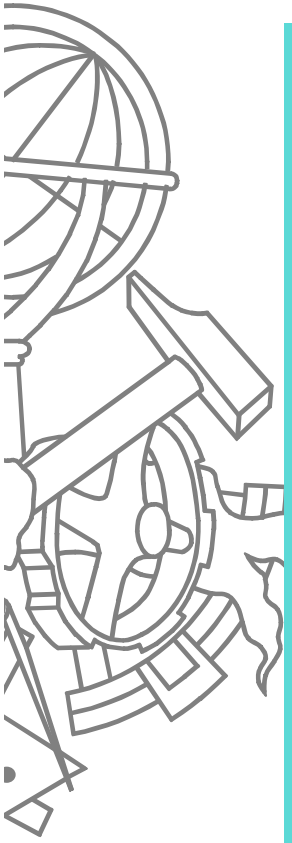
```
public OrderInfo GetOrder(int orderId)
{
   // Validate input
   if (orderId < 1)
      return null;

   // Get an instance of the Order DAL using the DALFactory
   IOrder dal = PetShop.DALFactory.Order.Create();

   // Return the order from the DAL
   return dal.GetOrder(orderId);
}
```

DAL.Order (1)



```
public OrderInfo GetOrder(int orderId)
{
    //Create a parameter
    SqlParameter parm = new SqlParameter(PARM_ORDER_ID,
                                         SqlDbType.Int);
    parm.Value = orderId;
    //Execute a query to read the order
    using (SqlDataReader rdr =
        SQLHelper.ExecuteReader(SQLHelper.CONN_STRING_DTC_ORDERS,
                  CommandType.Text, SQL_SELECT_ORDER, parm)) {
        if (rdr.Read()) {
            //Generate an order header from the first row
            CreditCardInfo creditCard = new
                  CreditCardInfo(rdr.GetString(2), rdr.GetString(3),
                                 rdr.GetString(4));
            AddressInfo billingAddress = new
                  AddressInfo(rdr.GetString(5), rdr.GetString(6),
                  rdr.GetString(7), rdr.GetString(8),
                  rdr.GetString(9), rdr.GetString(10),
                  rdr.GetString(11), rdr.GetString(12), null);
            AddressInfo shippingAddress = new
                  AddressInfo(rdr.GetString(13), rdr.GetString(14),
                  rdr.GetString(15), rdr.GetString(16),
                  rdr.GetString(17), rdr.GetString(18),
                  rdr.GetString(19), rdr.GetString(20), null);
```

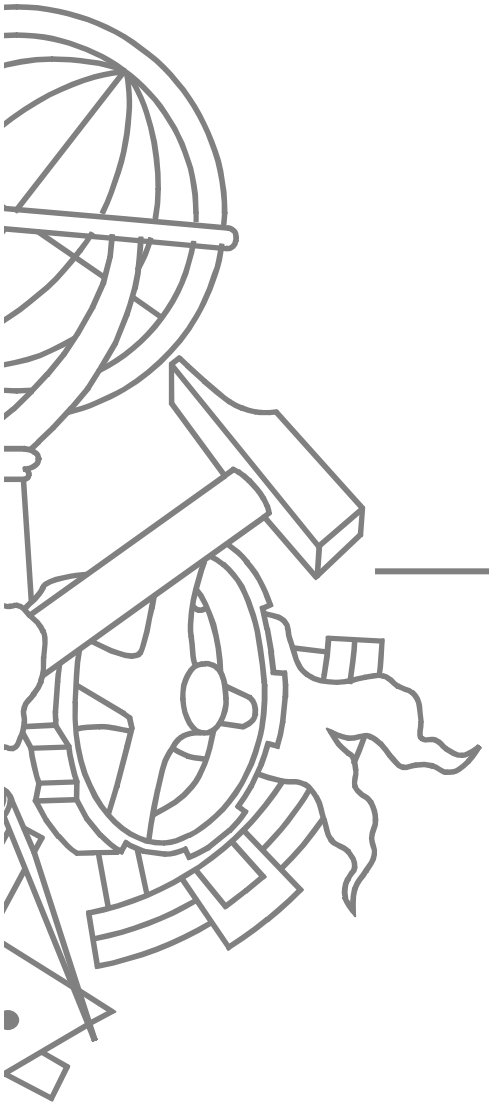
DAL.Order (2)



```
            OrderInfo order = new OrderInfo(orderId,
                    rdr.GetDateTime(0), rdr.GetString(1), creditCard,
                    billingAddress, shippingAddress,
                    rdr.GetDecimal(21));

            ArrayList lineItems = new ArrayList();
            LineItemInfo item = null;
            //Create the lineitems from the first row and subsequent
    rows
            do{
                item = new LineItemInfo(rdr.GetString(22),
                            string.Empty, rdr.GetInt32(23),
                            rdr.GetInt32(24), rdr.GetDecimal(25));
                lineItems.Add(item);
            } while(rdr.Read());

            order.LineItems =
            (LineItemInfo[])lineItems.ToArray(typeof(LineItemInfo));

            return order;
        }
    }
    return null;
}
```

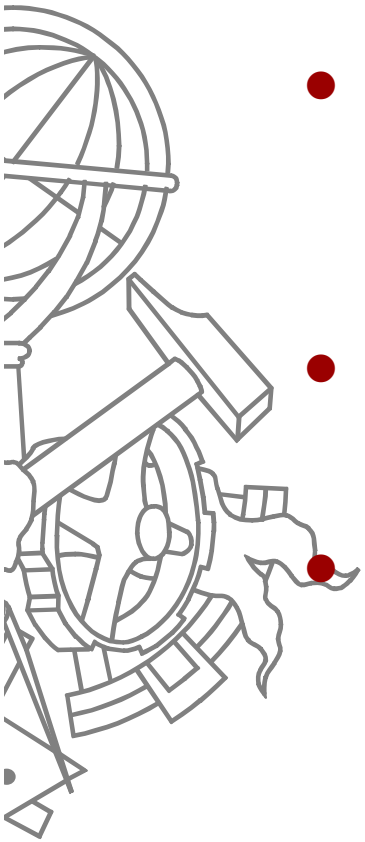
SQL

```
SELECT o.OrderDate, o.UserId, o.CardType,
    o.CreditCard, o.ExprDate, o.BillToFirstName,
    o.BillToLastName, o.BillAddr1, o.BillAddr2,
    o.BillCity, o.BillState, BillZip, o.BillCountry,
    o.ShipToFirstName, o.ShipToLastName,
    o.ShipAddr1, o.ShipAddr2, o.ShipCity,
    o.ShipState, o.ShipZip, o.ShipCountry,
    o.TotalPrice, l.ItemId, l.LineNum, l.Quantity,
    l.UnitPrice
FROM Orders as o, lineitem as l
WHERE o.OrderId = @OrderId AND o.orderid =
    l.orderid
```



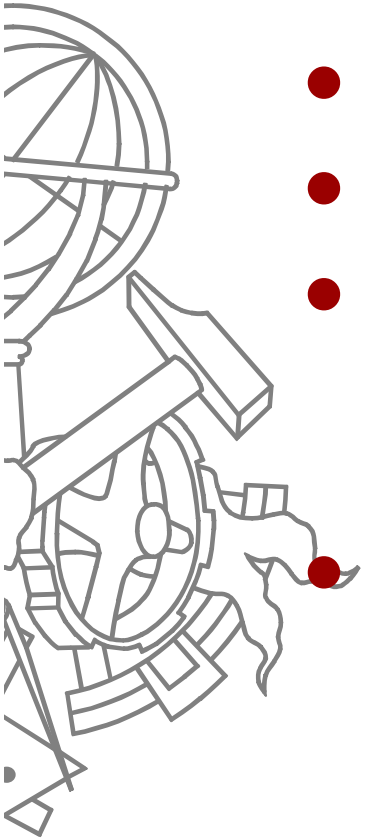
Alternativas

Alternativas



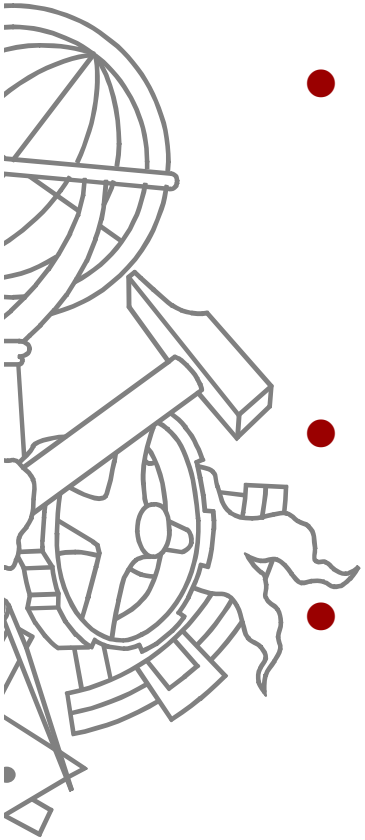
- Usar DataSets / Typed Datasets para representar as entidades de negócio
- Usar Domain Model para a lógica de negócio
- Implementar sistema de processamento de encomendas numa aplicação separada via serviço web
  - Usar service gateway + service layer

Usar DataSets



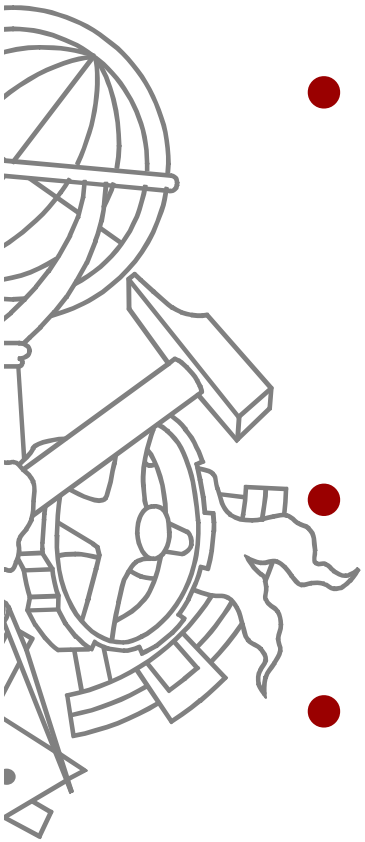
- Solução mais natural no .net
- Adequada ao padrão Table Module
- Usando Typed DataSets tem-se uma aproximação mista entre padrão RecordSet e *Custom classes*
- Evitava “confusão” de carregamento e leitura de valores de atributos das classes para os parâmetros dos comandos SQL

Usar Domain Model



- Uma vez que já usamos *custom classes* seria simples passar a lógica dos métodos no *package* BLL para as classes no *package* Model e criar um verdadeiro Domain Model
- Para evitar *overhead* de *serviced components* poderia usar-se um Decorator
- Numa primeira aproximação poderíamos usar a actual camada de acesso a dados que poderia depois ser convertida para Row Data Gateway

Implementar sistema de encomendas em aplicação separada



- Encapsular a funcionalidade de processamento de encomendas num serviço
  - Aplicação separada
- Usar Service Gateway para aceder ao serviço
- Como controlar transação distribuída?