



# Ambientes de Desenvolvimento Avançados

<http://www.dei.isep.ipp.pt/~jtavares/ADAV/ADAV.htm>

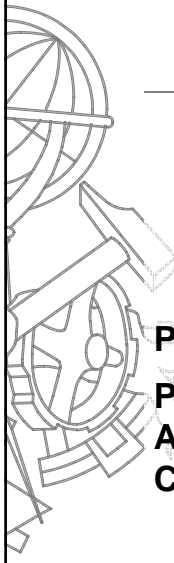
## Aula 8

### Engenharia Informática

2004/2005

José António Tavares  
jrt@isep.ipp.pt

1




# Microsoft®

---

## Designing Data Tier Components and Passing Data Through Tiers

### PARTE 3

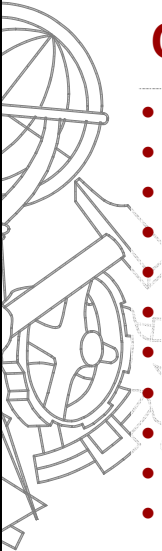
### Projecto de Componentes da Camada de Acesso a Dados e Passagem de Dados entre Camadas



2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

2




## Conteúdo

- Introdução
- Componentes Lógicos de Acesso a Dados
- Representação de Entidades de Negócio
- Mapeamento de Dados Relacionais a Entidades de Negócio
- Implementação de Componentes Lógicos de Acesso a Dados
- *Implementação de Entidades de Negócio*
- Transacções
- Validações
- Gestão de Excepções
- Autorização e Segurança
- Distribuição e Instalação (Deployment)

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

3



---

## Enquadramento: Resumo aulas 6 e 7

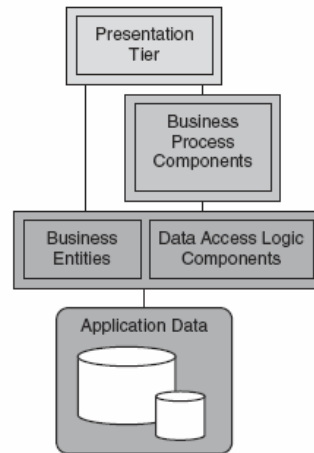
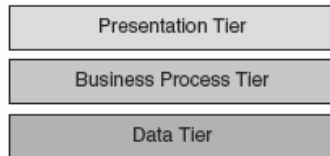
2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

4

# Introdução

## Camadas comuns numa aplicação distribuída



2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

5

## DALC vs BE

- **DALC** (**D**ata **A**ccess **L**ogic **C**omponent) has methods to implement business logic against the database.
- **BE** (**B**usiness **E**ntity) – Data is used to represent real world business entities, such as products or orders. There are numerous ways to represent these business entities in your application — for example, XML or DataSets or custom object-oriented classes — depending on the physical and logical design constraints of the application.

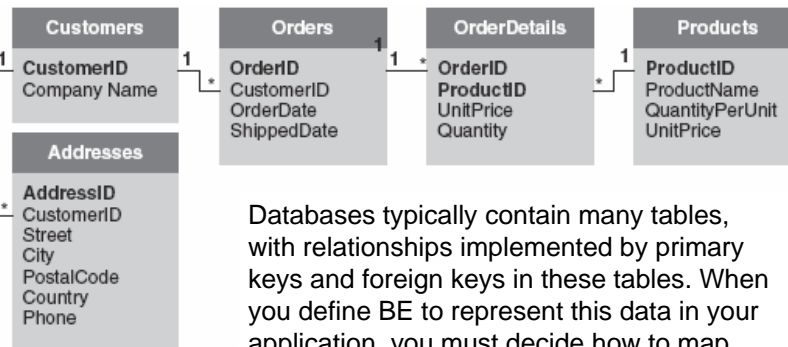
2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

6

## Mapeamento de Dados Relacionais a Entidades de Negócio

### An hypothetical retailer's database



Databases typically contain many tables, with relationships implemented by primary keys and foreign keys in these tables. When you define BE to represent this data in your application, you must decide how to map these tables to BE.

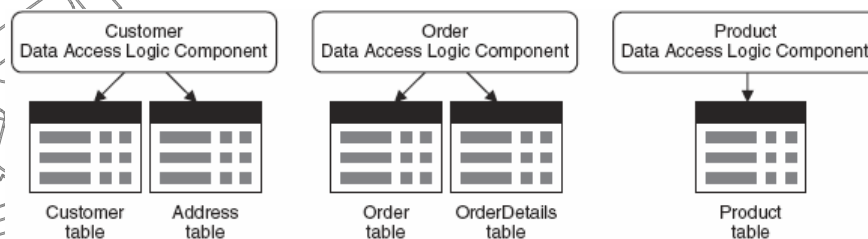
2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

7

## Mapeamento de Dados Relacionais a Entidades de Negócio

The relationships between the data access logic components and the tables that they represent in the database.

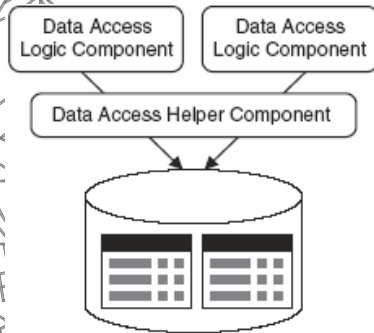


2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

8

## Implementação das classes de um DALC



- DALC use ADO.NET to execute SQL statements or call stored procedures.
- If your application contains multiple DALC, you can simplify the implementation of DALC classes by using a **data access helper component**.
- Design your DALC classes to provide a consistent interface for different types of clients.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

9

## How to Define a DALC Class

### Sample

- The code is a sample definition of a class named **CustomerDALC**, which is the DALC class for Customer BE. The **CustomerDALC** class implements the CRUD operations for the Customer BE and provides additional methods to encapsulate business logic for this object.

Pág 50-51

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

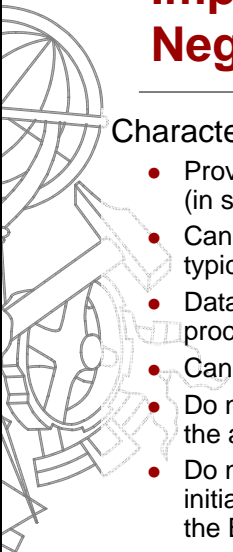
10



## Implementação das Entidades de Negócio

Continuação da aula anterior (aula 7)

2004/2005 ADAV 11  
Ambientes de Desenvolvimento Avançados



## Implementação das Entidades de Negócio

Characteristics:

- Provide **stateful** programmatic access to business data and (in some designs) related functionality.
- Can be built from data that has complex schemas. The data typically originates from multiple related tables in the DB.
- Data can be passed as part of the I/O parameters of business processes.
- Can be serializable, to persist the current state of the entities.
- Do not access the DB directly. All DB access is provided by the associated DALC.
- Do not initiate any kind of transaction. Transactions are initiated by the application or business process that is using the BE.

2004/2005 ADAV 12  
Ambientes de Desenvolvimento Avançados

## Implementação das Entidades de Negócio

- There are various ways to represent business entities in your application, ranging from a data-centric model to a more object oriented representation:
  - XML
  - Generic DataSet (.NET Framework)
  - Typed DataSet (.NET Framework)
  - Custom BE components
  - Custom BE components with CRUD behaviors

## Representing BE as XML

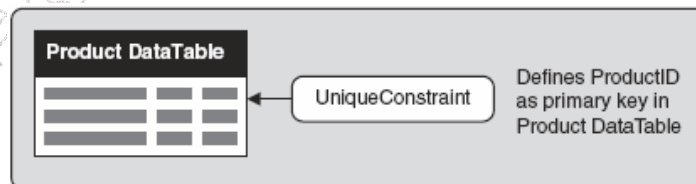
**Example:** The BE consists of a single product

```
<?xml version="1.0"?>
<Product xmlns="urn:aUniqueNamespace">
  <ProductID> 1 </ProductID>
  <ProductName>Chai </ProductName>
  <QuantityPerUnit>10 boxes x 20 bags</QuantityPerUnit>
  <UnitPrice> 18.00 </UnitPrice>
  <UnitsInStock> 39 </UnitsInStock>
  <UnitsOnOrder> 0 </UnitsOnOrder>
  <ReorderLevel> 10 </ReorderLevel>
</Product>
```

## Representing BE As a Generic DataSet

- A generic DataSet is an instance of the DataSet class, which is defined in the System.Data namespace in ADO.NET.
- A DataSet object contains one or more DataTable objects to represent information that the DALC retrieves from the DB.

A generic DataSet object for the Product BE.



2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

15

## Representing BE as a Typed DataSet

- A typed DataSet is a class that contains strongly typed methods, properties, and type definitions to expose the data and metadata in a DataSet.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

16





## Defining Custom BE Components

---

- Custom classes that represent BE typically contain the following:
  - **Private fields** to cache the BE data locally.
  - **Public properties** to access the state of the entity, and to access sub-collections and hierarchies of data inside the entity.
  - **Methods and properties to perform localized processing** by using the data in the entity component.
  - **Events** to signal changes to the internal state of the entity component.



## Defining Custom BE Components – Advantages

---

- Code readability
- Encapsulation
- Modeling of complex systems
- Localized validation
- Private fields



## Defining Custom BE Components – Disadvantages

---

- Collections of BE
- Serialization
- Representation of complex relationships and hierarchies in a BE
- Searching and sorting of data
- Deployment
- Support for Enterprise Services (COM+) clients
- Extensibility issues

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

19



## Defining Custom BE Components with CRUD Behaviors

---

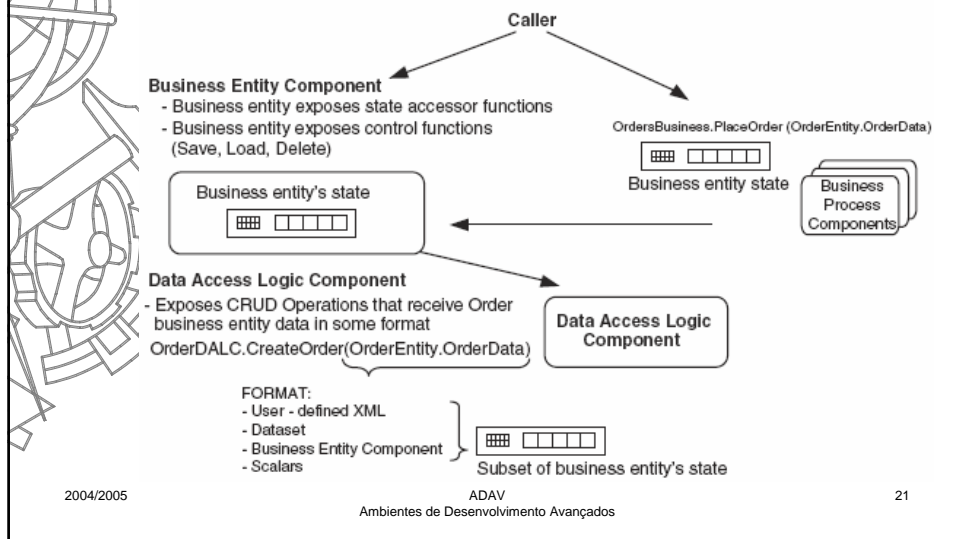
- When you define a custom entity, you can provide methods to completely encapsulate the CRUD operations on the underlying DALC.
- This is the more traditional object-oriented approach, and may be appropriate for complex object domains. The client application no longer accesses the DALC class directly.
- Instead, the client application creates an entity component and calls CRUD methods on the entity component. These methods forward to the underlying DALC.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

20

## Defining Custom BE Components with CRUD Behaviors



## Defining Custom BE Components with CRUD Behaviors - Advantages

- **Encapsulation.**  
The custom entity encapsulates the operations defined by the underlying DALC.
- **Interface to caller.**  
The caller must deal with only one interface to persist BE data. There is no need to access the DALC directly.
- **Private fields.**  
You can hide information that you do not want to expose to the caller.



## Defining Custom BE Components with CRUD Behaviors - Disadvantages

---

- **Dealing with sets of BE.**  
The methods in the custom entity pertain to a single BE instance. To support sets of BE, you can define static methods that take or return an array or a collection of entity components.
- **Increased development time.**  
The traditional object-oriented approach typically requires more design and development

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

23



## Recommendations for Representing Data and Passing Data Through Tiers 1/3

---

- The way in which you represent **data throughout your application**, and the way in which you pass that **data through the tiers**, do not necessarily need to be the same. However, **having a consistent and limited set of formats** yields performance and maintenance benefits that reduce your need for additional translation layers.
- The data format that you use should depend on your specific application requirements and how you want to work with the data. There is no universal way to represent your data, especially because many of today's applications are required to support multiple callers.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

24



## Recommendations for Representing Data and Passing Data Through Tiers 2/3

---

- However, it is recommended that you follow these general guidelines to represent your data:
  - If your application mainly works with sets and needs functionality such as sorting, searching, and data binding, **Datasets** are recommended.
  - However, if your application works with instance data, **scalar values** will perform better.
  - If your application mainly works with instance data, **custom BE components** may be the best choice because they prevent the overhead caused when a DataSet represents one row.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

25



## Recommendations for Representing Data and Passing Data Through Tiers 3/3

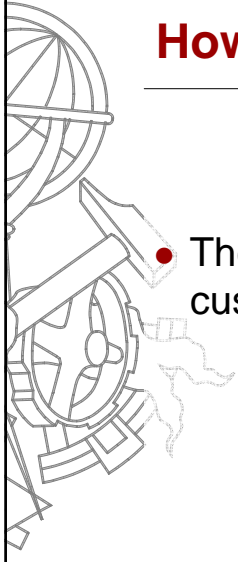
---

- More recommendations:
  - In most cases, design your application to use a **data-centric format**, such as XML documents or DataSets:
    - you can use the flexibility and native functionality provided by the DataSets to support multiple clients more easily,
    - reduce the amount of custom code,
    - and use a programming API that is familiar to most developers.
  - **Although** working with the data in an object-oriented fashion provides some benefits, custom coding complex BE **increases development and maintenance** costs in proportion to the amount of features you want to provide.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

26



## How to Define a BE Component

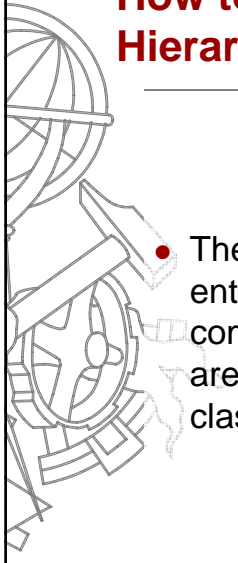
---

### Sample

- The example shows how to define a custom entity class for the Product BE.

Pág 54-55

2004/2005 ADAV 27  
Ambientes de Desenvolvimento Avançados



## How to Represent Collections and Hierarchies of Data in a BE Component

---

### Sample

- The example shows how to define a custom entity class for the Order BE. Each order comprises many order items; these order items are stored in a DataSet in the **OrderEntity** class.

Pág 55-56

2004/2005 ADAV 28  
Ambientes de Desenvolvimento Avançados



## How to Expose Events in a BE Component

---

### Sample

- Custom entities can raise events when the BE state is modified. These events are useful for rich client, user-interface design because data can be refreshed wherever it is being displayed. The sample shows how to raise BE related events in the `OrderEntity` class.

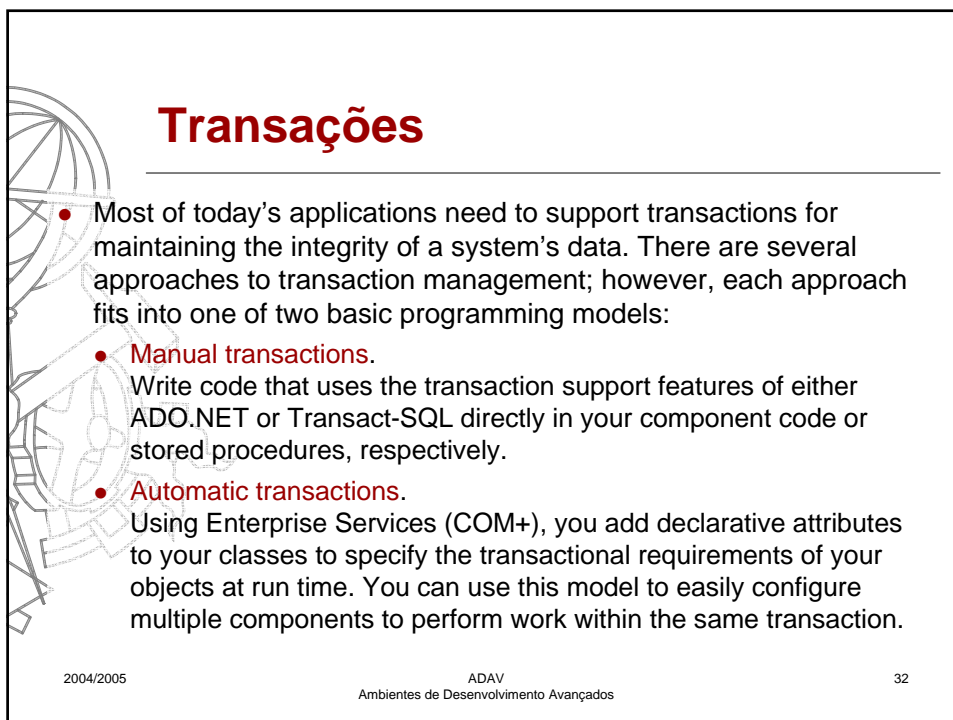
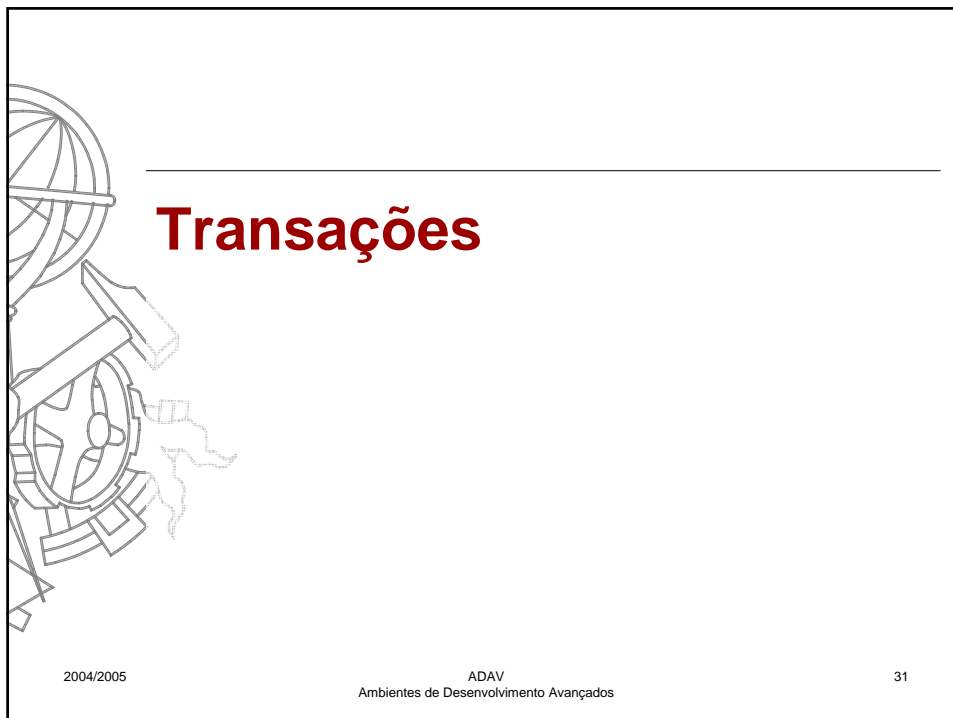
Pág 58-59



## Others “How to...”

---

- How to Use XML to Represent Collections and Hierarchies of Data
- How to Apply a Style Sheet Programmatically in a .NET Application
- How to Create a Typed DataSet
- How to Bind BE Components to User Interface Controls
- How to Serialize BE Components to XML Format
- How to Serialize BE Components to SOAP Format
- How to Serialize BE Components to Binary Format



- Most of today's applications need to support transactions for maintaining the integrity of a system's data. There are several approaches to transaction management; however, each approach fits into one of two basic programming models:
  - **Manual transactions.**  
Write code that uses the transaction support features of either ADO.NET or Transact-SQL directly in your component code or stored procedures, respectively.
  - **Automatic transactions.**  
Using Enterprise Services (COM+), you add declarative attributes to your classes to specify the transactional requirements of your objects at run time. You can use this model to easily configure multiple components to perform work within the same transaction.

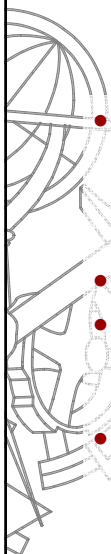




## Implementação de Transações

---

- In most circumstances, the root of the transaction is the business process rather than a DALC or a BE Component. The reason is that business processes typically require transactions that span multiple BE, not just a single BE.
- However, situations may arise where you need to perform transactional operations on a single BE without the assistance of a higher-level business process.

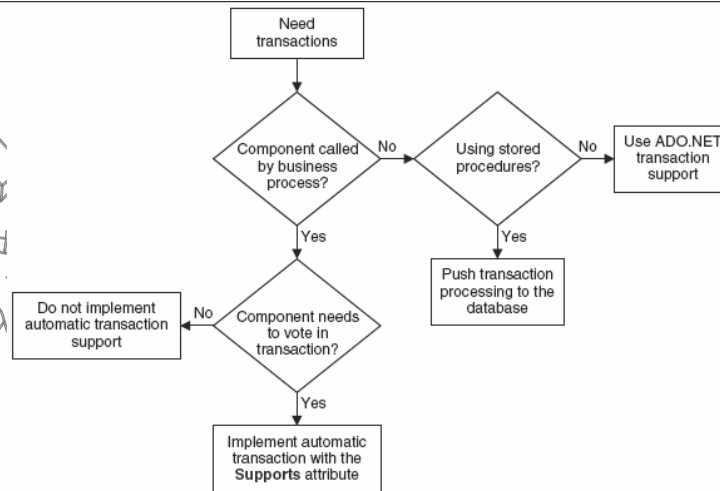


## Implementação de Transações

---

- For example, to add a new customer to the DB discussed earlier, you must perform the following operations:
  - Insert a new row in the Customer table.
  - Insert a new row or rows in the Address table.
- Both of these operations must succeed.
- If the Customer BE will never be a part of a larger business process that will initiate the transaction, use manual transactions within the Customer BE.
- **Manual transactions** are significantly **faster** than **automatic transactions** because they do not require any inter-process communication with the Microsoft Distributed Transaction Coordinator (DTC).

## Implementação de Transações



2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

35

## Recomendações para Utilização de Transações Manuais nos DALC

- Where possible, perform your processing in stored procedures. Use the Transact-SQL statements BEGIN TRANSACTION, END TRANSACTION, and ROLLBACK TRANSACTION to control transactions.
- If you are not using stored procedures, and the DALC will not be called from a business process, you can control transactions programmatically by using ADO.NET.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

36



## Recomendações para Utilização de Transações Automáticas nos DALC 1/4

---

- Despite the overhead associated with COM+ transactions, automatic transactions provide a simpler programming model
- They are necessary when your transactions span multiple distributed data sources as they work in conjunction with the DTC.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

37



## Recomendações para Utilização de Transações Automáticas nos DALC 2/4

---

- If you implement automatic transactions in DALC, consider:
  - The DALC must inherit from the `ServiceComponent` class in the `System.EnterpriseServices` namespace. Note that any assembly registered with COM+ services must have a strong name.
  - Annotate the DALC with the `Transaction(TransactionOption.Supported)` attribute so that you can perform read and write operations in the same component. This option avoids the overhead of transactions where they are not required — unlike `Transaction(TransactionOption.Required)`, which always requires a transaction.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

38



## Recomendações para Utilização de Transações Automáticas nos DALC 3/4

---

- The following code sample shows how to support automatic transactions in a DALC class:

```
using System. EnterpriseServices;

[Transaction(TransactionOption. Supported)]
public class CustomerDALC : ServicedComponent
{
    ...
}
```

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

39



## Recomendações para Utilização de Transações Automáticas nos DALC 4/4

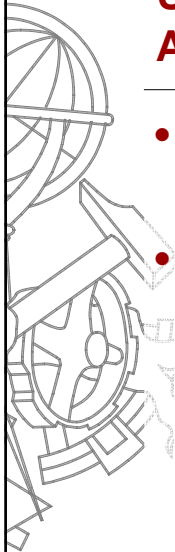
---

- If you use automatic transactions, your DALC should vote in transactions to indicate whether the operation succeeded or failed.
- To vote implicitly, annotate your methods by using the **AutoComplete** attribute and throw an exception if the operation fails.
- To vote explicitly, call the **SetComplete** or **SetAbort** method on the **ContextUtil** class.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

40



## Utilização de Transações Automáticas nos Componentes BE

---

- If you implement custom BE components that have behaviors, you can use automatic transactions to specify the transactional behavior of these objects.
- The recommendations for using automatic transactions to specify the transactional behavior of BE components are the same as the previously listed recommendations for implementing automatic transactions in DALC.



---

## Validações



## Validações

---

- Data validation can be performed at many tiers in an application.
- Different types of validation are appropriate in each tier:
  - The client application can validate BE data locally, before the data is submitted.
  - Business processes can validate business documents as the documents are received by using an XSD schema.
  - DALC and stored procedures can validate data to ensure referential integrity and to enforce constraints and nontrivial business rules.



## Tipos de Validações

---

- **Point-in-time validation.** This is a validation that is performed at a specific point in time. For example, a business process validates an XML document when the document is received.
- **Continuous validation.** This is validation that is performed on an ongoing basis at many different levels in your application.



## Tipos de Validações

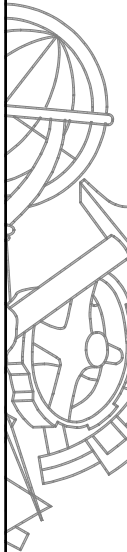
---

- Examples of **Continuous validation** include the following:
  - User interfaces can specify maximum field lengths to prevent the user from entering strings that are too long.
  - DataSets can specify the maximum length of data columns.
  - Custom BE components can perform range checks, length checks, non-null checks, and other simple tests on entity data.
  - DALC, stored procedures, and the DB itself can perform similar tests to ensure that data is valid before it is saved in the DB.



## Gestão de Excepções

---



## Gestão de Excepções

---

- When errors occur in .NET applications, the general advice is to throw exceptions rather than return error values from your methods.
- This advice has implications for the way you write DALC and BE components.
- Isto é o que Microsoft recomenda ...
- Nem sempre se deve seguir esta recomendação ...



## Gestão de Excepções

---

- There are two general kinds of exceptions that will occur:
  - Technical exceptions, which include:
    - ADO.NET
    - Connection to database
    - Resources (such as database, network share, and Message Queuing) are unavailable
  - Business logic exceptions, which include:
    - Validation errors
    - Errors in stored procedures that implement business logic



## Gestão de Exceções nos DALC

```
public class CustomerDALC
{
    public void UpdateCustomer(Dataset aCustomer)
    {
        try
        {
            // Update the customer in the database...
        }
        catch (SqlException se)
        {
            // Catch the exception and wrap, and rethrow
            throw new DataAccessException(
                "Database is unavailable", se);
        }
        finally
        {
            // Cleanup code
        }
    }
}
```

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

49

## Gestão de Exceções nos Componentes BE

```
public class CustomerEntity
{
    public void Update()
    {
        // Check that the user has provided the required
        // data. In this case a first name for the customer
        if (FirstName == "" )
        {
            // Throw a new application exception that
            // you have defined
            throw new MyArgumentOutOfRangeException(
                "You must provide a First Name.");
        }
        ...
    }
}
```

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

50

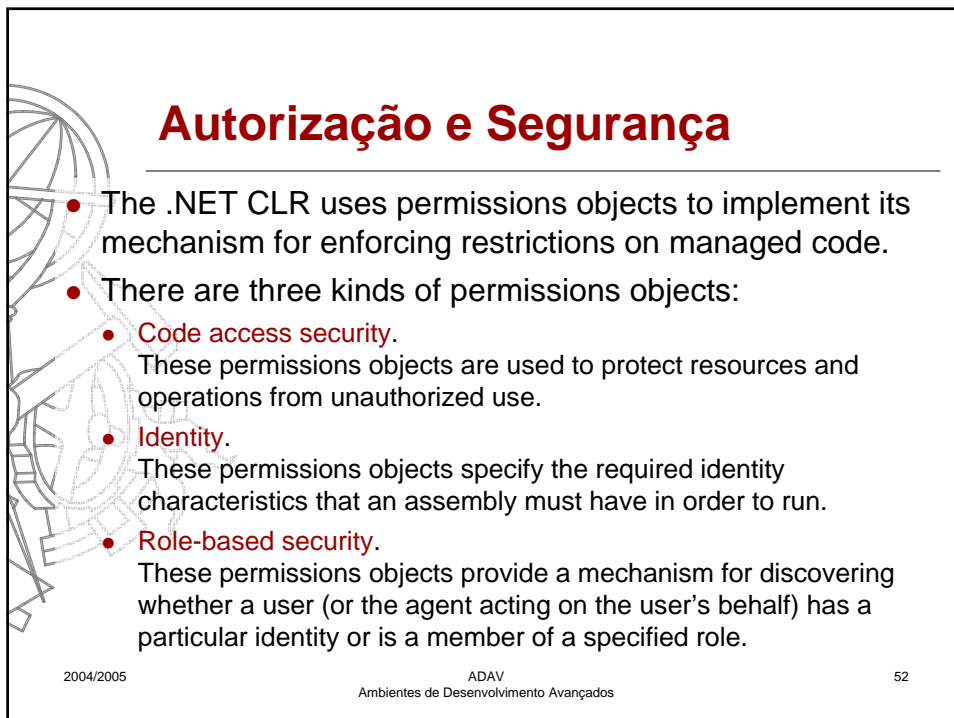


## Autorização e Segurança

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

51



## Autorização e Segurança

- The .NET CLR uses permissions objects to implement its mechanism for enforcing restrictions on managed code.
- There are three kinds of permissions objects:
  - **Code access security.**  
These permissions objects are used to protect resources and operations from unauthorized use.
  - **Identity.**  
These permissions objects specify the required identity characteristics that an assembly must have in order to run.
  - **Role-based security.**  
These permissions objects provide a mechanism for discovering whether a user (or the agent acting on the user's behalf) has a particular identity or is a member of a specified role.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

52



## Segurança nos DALC - Recomendações

---

- DALC are designed to be used by other application components, and are the last place in your application code where you can implement security before the caller has access to your data.
- Often, DALC can rely on the security context set by the caller.
- However, there are some situations where the DALC should perform its own authorization checks to determine whether a principal is allowed to perform a requested action.
- Authorization occurs after authentication and uses information about the principal's identity and roles to determine what resources the principal can access.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

53



## Segurança nos DALC - Recomendações

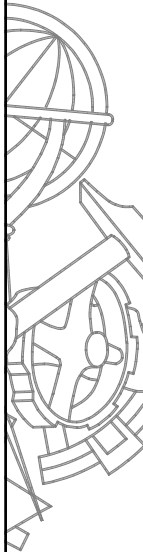
---

- Perform authorization checks at the DALC level if you need to:
  - Share DALC with developers of business processes that you do not fully trust.
  - Protect access to powerful functions exposed by the data stores.
- After you define identity and principal objects, there are three different ways to perform role-based security checks:
  - Use the `PrincipalPermission` object to perform imperative security checks.
  - Use the `PrincipalPermissionAttribute` attribute to perform declarative security checks.
  - Use the properties and the `IsInRole` method in the `Principal` object to perform explicit security checks.

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

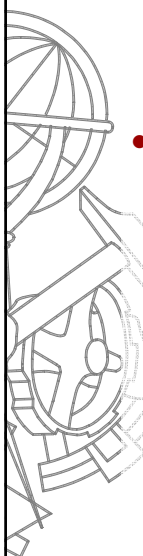
54



## Windows Authentication

---

- Ideally, you should use Windows Authentication, rather than using SQL Server Authentication, when you connect to the database.
- However, you should use service accounts and avoid impersonating through to the database, because this will impede connection pooling.
- Connection pooling requires identical connection strings; if you try to open the database by using different connection strings, you will create separate connection pools, which will limit scalability.



## Secure Communication Recommendations

---


- To achieve secure communication between calling applications and DALC, consider the following recommendations:
  - If your DALC are called over the wire from diverse tiers, and the exchange involves sensitive information that needs to be protected, use Distributed Component Object Model (DCOM), Secure Sockets Layer (SSL), or Secure Internet Protocol (IPSec) secure communication technologies.
  - If data is stored encrypted in a DB, DALC are usually responsible for encrypting and decrypting the data. If the risk of exposing the information is high, strongly consider securing the communication channel to and from the DALC.



## Recommendations for Security in BE Components

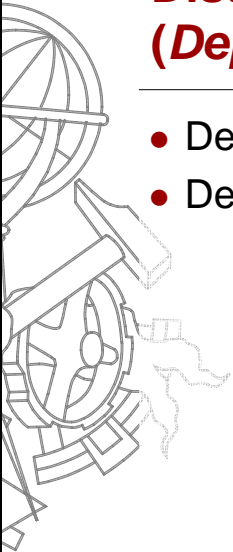
---

- If you implement your BE as data structures (such as XML or DataSets), you do not need to implement security checks.
- However, if you implement your BE as custom BE components with CRUD operations, consider the following recommendations:
  - If the entities are exposed to business processes that you do not fully trust, implement authorization checks in the BE components and in the DALC. If you do implement checks at both levels, however, you may encounter the maintenance issue of keeping the security policies synchronized.
  - BE components should not deal with communication security or data encryption. Leave these tasks to the corresponding DALC.



## Distribuição e Instalação (Deployment)

---



## Distribuição e Instalação (Deployment)

---

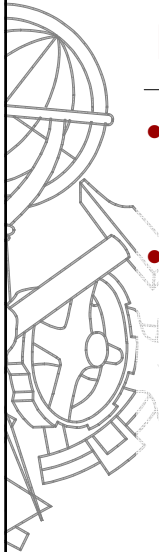
- Deploying DALC
- Deploying BE



## Deploying DALC 1/3

---

- Deploy DALC together with the business process objects. This deployment method provides optimum performance for data transfers, and has several additional technical benefits:
  - Transactions can flow seamlessly between the business process objects and the DALC. However, transactions do not flow seamlessly across remoting channels. In the remoting scenario, you need to implement transactions by using DCOM. Furthermore, if the business process and the DALC were separated by a firewall, you would require firewall ports open between both physical tiers to enable DTC communication.
  - Deploying business process objects and DALC together reduces the number of transaction failure nodes.
  - The security context flows automatically between the business process objects and the DALC. There is no need to set principal objects.



## Deploying DALC 2/3

---

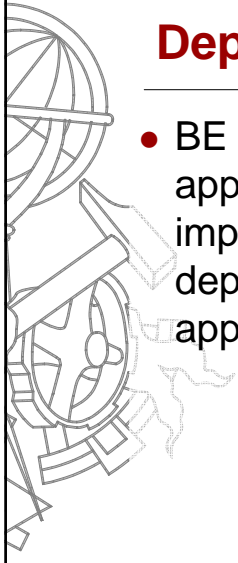
- Deploy DALC together with the user-interface code. DALC are sometimes used directly from UI components and UI process components.
- To increase performance in Web scenarios, you can deploy DALC together with the UI code; this deployment method enables the UI layer to take advantage of data reader streaming for optimum performance.



## Deploying DALC 3/3

---

- However, if you do consider this deployment method, bear in mind the following:
  - A common reason for not deploying DALC together with UI code is to prevent direct network access to your data sources from your Web farms.
  - If your Web farm is deployed in a DMZ scenario, firewall ports must be opened to access SQL Server. If you are using COM+ transactions, additional firewall ports must be opened for DTC communication.



## Deploying BE

---

- BE are used at many different tiers in your application. Depending on how you implement your BE, you may need to deploy them to multiple locations if your application spans physical tiers.



## Deploying BE - Implementation scenarios

---

- **Deploying BE implemented as typed DataSets.** The typed DataSet class must be accessed by the DALC and by the calling application. Therefore, the recommendation is to define typed DataSet classes in a common assembly to be deployed on multiple tiers.
- **Deploying BE implemented as custom business entity components.** The custom entity class may need to be accessed by the DALC, depending on how you defined the method signatures in the DALC. Follow the same recommendation as for typed DataSets by defining custom entity classes in a common assembly to be deployed on multiple tiers.
- **Deploying BE implemented as generic DataSets or XML strings.** Generic DataSets and XML strings do not represent a separate data type. There are no deployment issues for BE implemented in these formats.





# Pet Shop Application

2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

65



# PetShop - Microsoft

**Microsoft .NET Pet Shop**

<http://msdn.microsoft.com/library/en-us/dnbda/html/bdasamppet.asp>

<http://java.sun.com/developer/releases/petstore/>



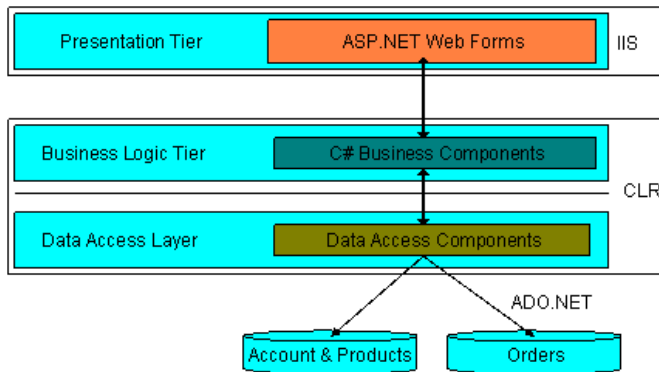
2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

66

# PetShop - Microsoft

## .NET Pet Shop high-level logical architecture



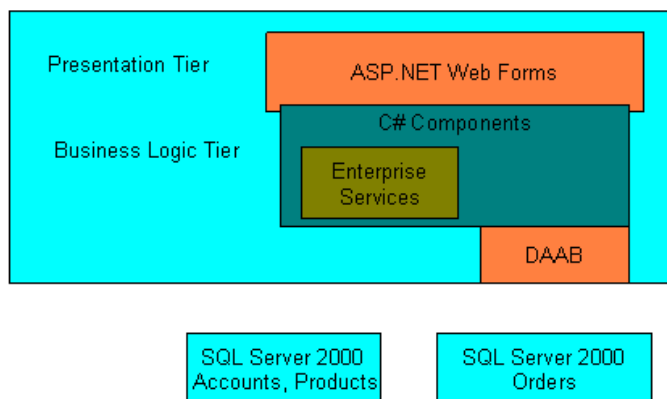
2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

67

# PetShop - Microsoft

## .NET Pet Shop 2.0 Architecture



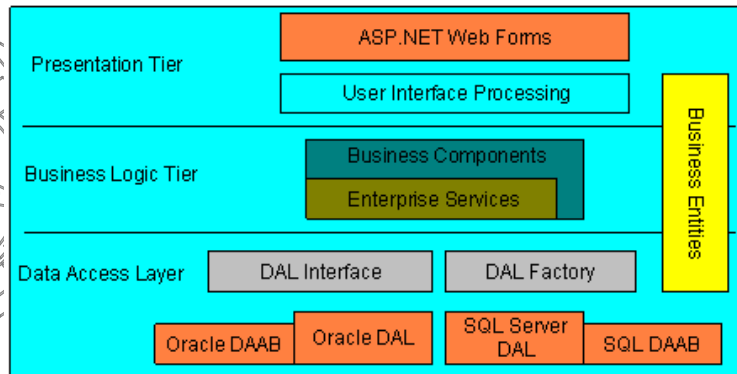
2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

68

# PetShop - Microsoft

## .NET Pet Shop 3.0 Architecture



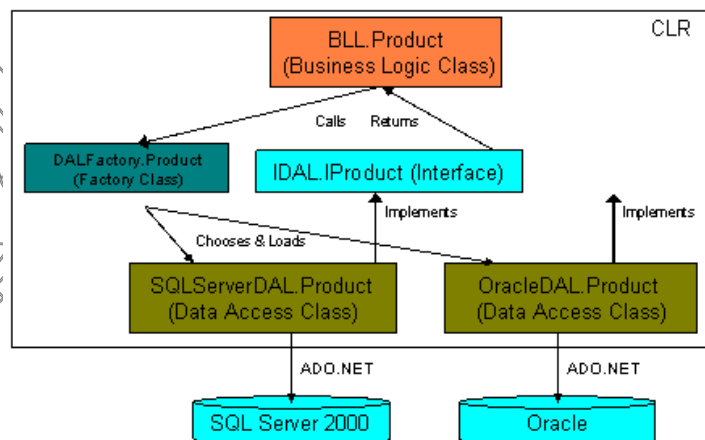
2004/2005

ADAV  
Ambientes de Desenvolvimento Avançados

69

# PetShop - Microsoft

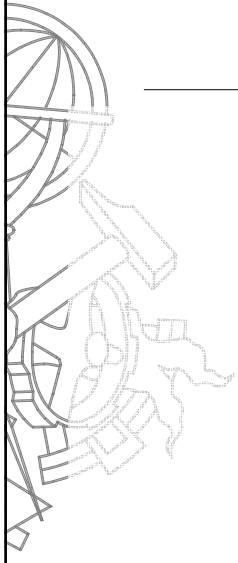
## DAL factory implementation in .NET Pet Shop



2004/2005

Ambientes de Desenvolvimento Avançados

70



---

## Questões

