

Ambientes de Desenvolvimento Avançados

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

Aula 8 Engenharia Informática

2006/2007

José António Tavares
jrt@isep.ipp.pt



Microsoft®

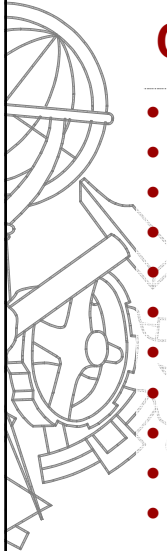
Designing Data Tier Components and Passing Data Through Tiers

PARTE 2

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



patterns & practices
proven practices for predictable results



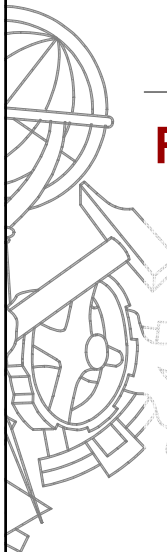
Conteúdo

- 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)

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

3



Resumo aula 6

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

4

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.

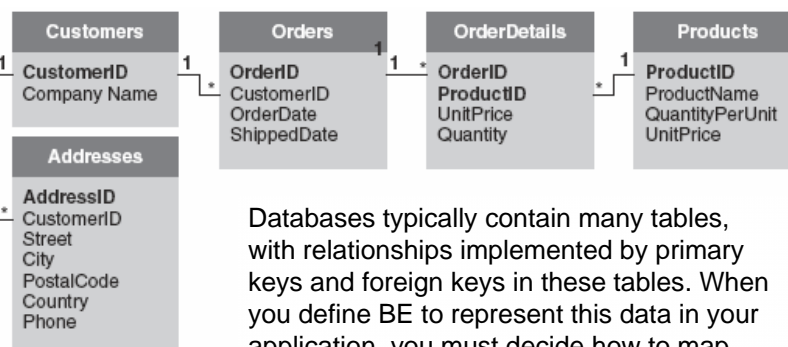
2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

5

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.

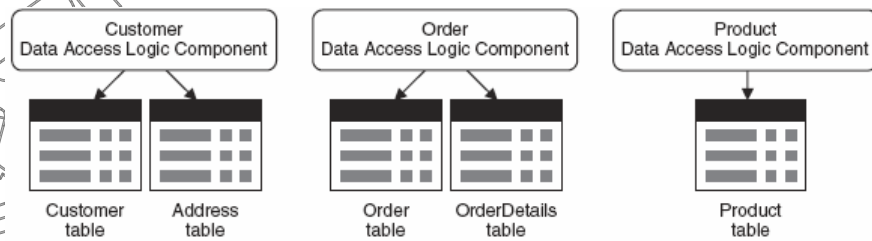
2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

6

Mapeamento de Dados Relacionais a Entidades de Negócio

The relationships between the DALC and the tables that they represent in the database.

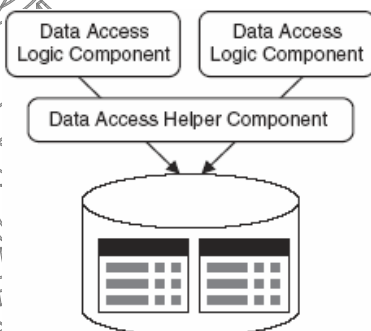


2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

7

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.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

8

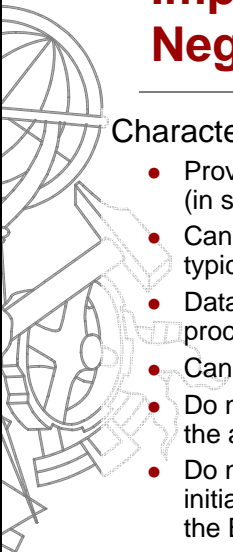


Implementação das Entidades de Negócio

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

9



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.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

10



Implementação das Entidades de Negócio

- There are various ways to represent business entities in the applications, 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

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

11



Implementação das Entidades de Negócio

- To help the in decision of the most appropriate representation for BE in a particular circumstance, the following tasks for each BE format have to be taken into account:
 - Organize collections of BE
 - Data bind BE to user interface controls
 - Serialize business entity data
 - Pass BE data between tiers

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

12



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 XML – Guidelines 1/2

- Decide whether the XML document should contain a single BE or a collection of BE.
- Use a namespace to uniquely identify the XML document, to avoid name clashes with content in other XML documents.
- Choose appropriate names for elements and attributes. Choose names that make sense for your application.



Representing BE as XML – Guidelines 2/2

- Use one of the following approaches to retrieve your BE in XML format:
 - If you are using SQL Server 2000, you can use the FOR XML clause in your queries or stored procedures.
 - Retrieve a DataSet and transform it or write it out as an XML stream.
 - Build an XML document from output parameters or by using a data reader.



Representing BE as XML – Advantages

- **Standards support.**
XML is a World Wide Web Consortium (W3C) standard data representation format. (see <http://www.w3.org/xml/>)
- **Flexibility.**
XML can represent hierarchies and collections of information.
- **Interoperability.**
XML is an ideal choice for exchanging information with external parties and trading partners, regardless of platform. If the XML data will be consumed by ASP.NET or WinForms applications, you can load the XML data into a DataSet to take advantage of the data binding support provided by DataSets.



Representing BE as XML – Disadvantages 1/3

- **Preserving type fidelity.** Type fidelity is not preserved in XML. However, you can use XSD schemas for simple data typing.
- **Validating XML.** To validate XML, you can parse the code manually or use an XSD schema. Both approaches are relatively slow.
- **Displaying XML.** You cannot automatically display XML data in the user interface. You can write an XSLT style sheet to transform the data into a DataSet; however, style sheets are not easy to write. Alternatively, the style sheet can transform the XML into a displayable format such as HTML.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

17



Representing BE as XML – Disadvantages 2/3

- **Parsing XML.** To parse XML, you can use the Document Object Model (DOM) or the XmlReader class provided in the .NET Framework class library. XmlReader provides fast-forward only, read-only access to XML data, but DOM is more flexible because it provides random read/write access. However, parsing an XML document by using DOM is slower;
- **Using private fields.** You do not have the option of hiding information.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

18

Representing BE as XML – Disadvantages 3/3

Sorting XML.

- You cannot automatically sort XML data.
- Instead, use one of the following techniques:
 - Deliver the data in presorted order. This option does not support dynamic resorting of data in the calling application.
 - Apply an XSLT style sheet to sort the data dynamically. If necessary, you can alter the sort criteria in the XSLT style sheet at run time, by using DOM.
 - Transform the XML data into a DataSet, and use a DataView object to sort and search the data elements.

2006/2007

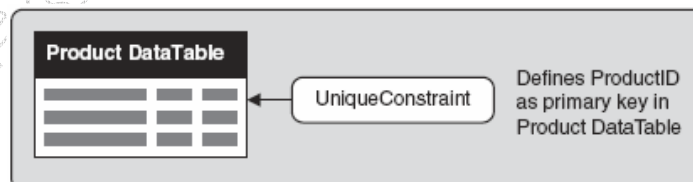
ADAV
Ambientes de Desenvolvimento Avançados

19

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.



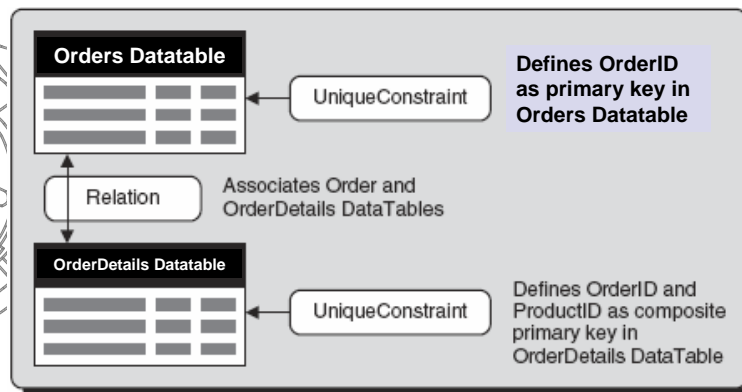
2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

20

Representing BE As a Generic DataSet

A generic DataSet object for the Order BE.



2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

21

Representing BE As a Generic DataSet

```
// Create a ProductDALC object
ProductDALC dal cProduct = new ProductDALC();

// Call a method on ProductDALC to get a DataSet
// containing information for all products
DataSet dsProducts = dal cProduct.GetProducts();

// Use DataSet in the client. For example, bind the
// DataSet to user interface controls
dataGrid1.DataSource = dsProducts.Tables[0].DefaultView;
dataGrid1.DataBind();

// When you are ready, pass the updated DataSet to
// the ProductDALC to save the changes back to the DB
dal cProduct.UpdateProducts(dsProducts);
```

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

22



Representing BE as a Generic DataSet – Advantages 1/2

- **Flexibility.** DataSets can contain collections of data, and can represent complex data relationships.
- **Serialization.** DataSets natively support serialization when passing across tiers.
- **Data binding.** DataSets can be bound to any user interface controls in ASP.NET and Windows Forms applications.
- **Sorting and filtering.** DataSets can be sorted and filtered by using DataView objects. An application can create several DataView objects for the same DataSet.
- **Interchangeability with XML.** DataSets can be read/written in XML format. This is a useful technique in remote and disconnected applications. Applications can also persist DataSets to XML.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

23



Representing BE as a Generic DataSet – Advantages 2/2

- **Availability of metadata.** Full metadata can be provided for a DataSet, in the form of an XSD schema. You can also programmatically obtain metadata for the DataSet by using methods in the DataSet, DataTable, DataColumn, Constraint, and Relation classes.
- **Optimistic concurrency.** When you are updating data, you can use DataSets, in conjunction with data adapters, to perform optimistic concurrency checks easily.
- **Extensibility.** If the database schema is modified, the methods in the DALC can create DataSets that contain modified DataTable and DataRelation objects as appropriate. The DALC method signatures do not change. The calling application can be modified to use these new elements in the DataSet.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

24



Representing BE as a Generic DataSet – Disadvantages 1/2

- Client code must access data through collections in the DataSet.
- To access a table in a DataSet, client code must index into the DataTable collections by using an integer indexer or a string indexer.
- To access a particular column, you must index into the DataColumn collection by using a column number or a column name.

```
...  
// Get the product name for the product in the first row of a  
// DataSet called dsProducts. Note the collections are zero-based.  
String str =  
    (String)dsProducts.Tables["Products"].Rows[0]["ProductName"];  
...
```

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

25



Representing BE as a Generic DataSet – Disadvantages 2/2

- **High instantiation and marshalling costs.**
 - DataSets result in the creation of several sub-objects (DataTable, DataRow, and DataColumn), which means that DataSets can take longer to instantiate and marshal than XML strings or custom entity components.
 - The relative performance of DataSets improves as the amount of data increases, because the overhead of creating the internal structure of the DataSet is less significant than the time it takes to populate the DataSet with data.
- **Private fields.**
You do not have the option of hiding information.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

26



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.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

27



Representing BE as a Typed DataSet - Advantages

- **Code readability.**

To access tables and columns in a typed DataSet, you can use typed methods and properties, as shown in the following code:

```
// Get the product name for the product in the  
// first row of a typed DataSet called  
// dsProducts. Note the collections are  
// zero-based.  
String str = dsProducts.Products[0].ProductName;
```

- **Compile type checking.**

Invalid table names and column names are detected at compile time rather than at run time.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

28



Representing BE as a Typed DataSet – Disadvantages 1/2

- **Deployment.**
The assembly containing the typed DataSet class must be deployed to all tiers that use the BE.
- **Support of Enterprise Services (COM+) callers.**
If a typed DataSet will be used by COM+ clients, the assembly containing the typed DataSet class must be given a strong name and must be registered on client computers. Typically, the assembly is installed in the GAC (*Global Assembly Cache*).

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

29



Representing BE as a Typed DataSet – Disadvantages 2/2

- **Extensibility issues.**
If the DB schema is modified, the typed DataSet class might need to be regenerated. The regeneration process will not preserve any custom code that was implemented in the typed DataSet class.
- **Instantiation.**
You cannot instantiate the type by using the new operator.
- **Inheritance.**
Your typed dataset must inherit from DataSet, which precludes the use of any other base classes.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

30



Defining Custom BE Components 1/2

- Custom classes that represent BE typically contain the following:
 - Private fields to cache the BE data locally. These fields hold a snapshot of the data in the DB at the time the data was retrieved from the DB by the DALC.
 - Public properties to access the state of the entity, and to access sub-collections and hierarchies of data inside the entity.
 - The properties can have the same names as the database column names, but this is not an absolute requirement. *Choose property names according to the needs of your application, rather than the names in the database.*

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

31



Defining Custom BE Components 2/2

- Custom classes that represent BE typically contain the following - Continuation:
 - 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.

2006/2007

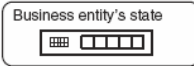
ADAV
Ambientes de Desenvolvimento Avançados

32

Defining Custom BE Components

Business Entity Component
- Custom business entity classes expose state accessor functions

```
Public string Name {
    set{myData.Name = value} }
```



Caller

Business entity data passed to
business process components or
Data Access Logic Component

Data Access Logic Component
- Exposes CRUD Operations that receive Order
business entity data in some format
OrderDALC.CreateOrder(OrderEntity,OrderData)

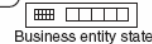


OrdersBusiness.PlaceOrder
(OrderEntity,OrderData)

FORMAT:
- User - defined XML
- Dataset
- Business Entity Component



ADAV



2006/2007

Ambientes de Desenvolvimento Avançados

33

Defining Custom BE Components – Recommendations 1/6

- **Choosing between structs and classes.**
For simple BE that do not contain hierarchical data or collections, consider defining a struct to represent the BE. For complex BE, or for BE that require inheritance, define the entity as a class instead.
- **Representing the BE's state.**
For simple values such as numbers and strings, define fields by using the equivalent .NET data type.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

34



Defining Custom BE Components – Recommendations 2/6

- **Representing sub-collections and hierarchies in a custom BE Component.**

There are two ways to represent sub-collections and hierarchies of data in a custom entity:

- **A .NET collection such as ArrayList.** The .NET collection classes offer a convenient programming model for resizable collections, and also provide built-in support for data binding to user interface controls.
- **A DataSet.** DataSets are well suited for storing collections and hierarchies of data from a relational database or from an XML document. Additionally, DataSets are preferred if you need to be able to filter, sort, or data bind your sub-collections.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

35



Defining Custom BE Components – Recommendations 3/6

- **Supporting data binding for user interface clients.**

If the custom entity will be consumed by user interfaces and you want to take advantage of automatic data binding, you may need to implement data binding in your custom entity. Consider the following scenarios:

- **Data binding in Windows Forms.** You can data bind an entity instance to controls without implementing data binding interfaces in your custom entity. You can also data bind an array or a .NET collection of entities.
- **Data binding in Web Forms.** You cannot data bind an entity instance to controls in a Web Form without implementing the IBindingList interface. However, if you want to data bind only sets, you can use an array or a .NET collection without needing to implement the IBindingList interface in your custom entity.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

36



Defining Custom BE Components – Recommendations 4/6

- Exposing events for internal data changes.
 - Exposing events is useful for rich client user interface design because it enables data to be refreshed wherever it is being displayed.
 - The events should be for internal state only, not for data changes on a server.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

37



Defining Custom BE Components – Recommendations 5/6

- Making your business entities serializable.

Making business entities serializable enables the business entity's state to be persisted in interim states without database interactions. The result can be to ease offline application development and design of complex user interface processes that do not affect business data until they are complete. There are two types of serialization:
- XML serialization by using the XmlSerializer class.
- Formatted serialization by using the BinaryFormatter or SoapFormatter class.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

38

Defining Custom BE Components – Recommendations 6/6

- XML serialization by using the XmlSerializer class.
 - Use XML serialization when you need to serialize only public fields and public read/write properties to XML. Note that if you return BE data from a Web service, the object is automatically serialized to XML through XML serialization.
- Formatted serialization by using the BinaryFormatter or SoapFormatter class.
 - Use formatted serialization when you need to serialize all the public and private fields and object graphs of an object, or if you will pass an entity component to or from a remoting server.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

39

Defining Custom BE Components – Advantages 1/3

- **Code readability.**
To access data in a custom entity class, you can use typed methods and properties;

```
// Create a ProductDALC object
ProductDALC dal cProduct = new ProductDALC();

// Use the ProductDALC object to create and populate a
// ProductEntity object. This code assumes the ProductDALC class
// has a method named GetProduct, which takes a Product ID as a
// parameter (21 in this example) and returns a ProductEntity
// object containing all the data for this product.
ProductEntity aProduct = dal cProduct.GetProduct(21);

// Change the product name for this product
aProduct.ProductName = "Roasted Coffee Beans";
```

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

40

Defining Custom BE Components – Advantages 2/3

- **Encapsulation.**

Custom entities can contain methods to encapsulate simple business rules. These methods operate on the business entity data cached in the entity component, rather than accessing the live data in the database.

Consider the following example:

```
// Call a method defined in the ProductEntity  
// class.  
aProduct.IncreaseUnitPriceBy(1.50);
```

Defining Custom BE Components – Advantages 3/3

- **Modeling of complex systems.**

If you are modeling a complex domain problem that has many interactions between different BE, it may be beneficial to define custom entity classes to absorb the complexity behind well-defined class interfaces.

- **Localized validation.**

Custom entity classes can perform simple validation tests in their property accessors to detect invalid BE data.

- **Private fields.**

You can hide information that you do not want to expose to the caller.



Defining Custom BE Components – Disadvantages 1/3

- **Collections of business entities.**
A custom entity represents a single BE, not a collection of BE. The calling application must create an array or a collection to hold multiple BE.
- **Serialization.**
You must implement your own serialization mechanism in a custom entity. You can use attributes to control how entity components are serialized, or you can implement the `ISerializable` interface to control your own serialization.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

43



Defining Custom BE Components – Disadvantages 2/3

- **Representation of complex relationships and hierarchies in a BE.**
You must implement your own mechanism for representing relationships and hierarchies of data in a BE Component. As described previously, `DataSets` are often the easiest way to achieve this effect.
- **Searching and sorting of data.**
You must define your own mechanism to support searching and sorting of entities.
- **Deployment.**
You must deploy, on all physical tiers, the assembly containing the custom entity.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

44



Defining Custom BE Components – Disadvantages 3/3

- **Support for Enterprise Services (COM+) clients.**
If a custom entity will be used by COM+ clients, the assembly containing the entity must be given a strong name and must be registered on client computers. Typically, the assembly is installed in the GAC.
- **Extensibility issues.**
If the database schema is modified, you might need to modify the custom entity class and redeploy the assembly.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

45



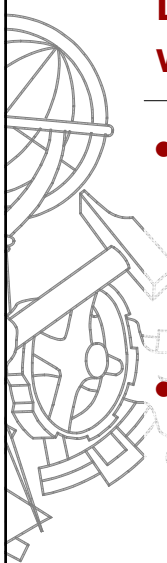
Defining Custom BE Components with CRUD Behaviors

- 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.

2006/2007

ADAV
Ambientes de Desenvolvimento Avançados

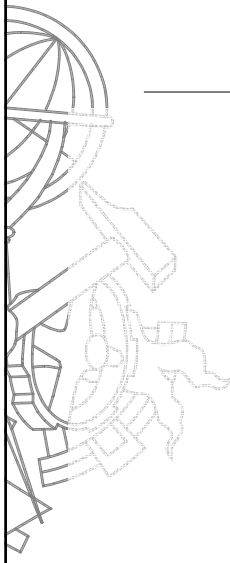
46



Defining Custom BE Components with CRUD Behaviors

- Advantages
 - Encapsulation
 - Interface to caller
 - Private fields
- Disadvantages
 - Dealing with sets of BE
 - Increased development time

2006/2007 ADAV 47
Ambientes de Desenvolvimento Avançados



Questões

?

2006/2007 ADAV 48
Ambientes de Desenvolvimento Avançados