Introduction to Purchase Flow

Presenter's Name
Presenter's Title

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Training Materials – Usage Agreement

Use of this Site ("Site") or Materials constitutes agreement with the following terms and conditions:

1. Oracle Corporation ("Oracle") is pleased to allow its business partner ("Partner") to download and copy the information, documents, and the online training courses (collectively, "Materials") found on this Site. The use of the Materials is restricted to the non-commercial, internal training of the Partner's employees only. The Materials may not be used for training, promotion, or sales to customers or other partners or third parties.

2. All the Materials are trademarks of Oracle and are proprietary information of Oracle. Partner or other third party at no time has any right to resell, redistribute or create derivative works from the Materials.

3. Oracle disclaims any warranties or representations as to the accuracy or completeness of any Materials. Materials are provided "as is" without warranty of any kind, either express or implied, including without limitation warranties of merchantability, fitness for a particular purpose, and non-infringement.

4. Under no circumstances shall Oracle or the Oracle Authorized Delivery Partner be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this Site of Materials. As a condition of use of the Materials, Partner agrees to indemnify Oracle from and against any and all actions, claims, losses, damages, liabilities and expenses (including reasonable attorneys' fees) arising out of Partner's use of the Materials.

5. Reference materials including but not limited to those identified in the Boot Camp manifest can not be redistributed in any format without Oracle written consent.

Agenda

- Commerce Business Layer
- Pipelines and Repository

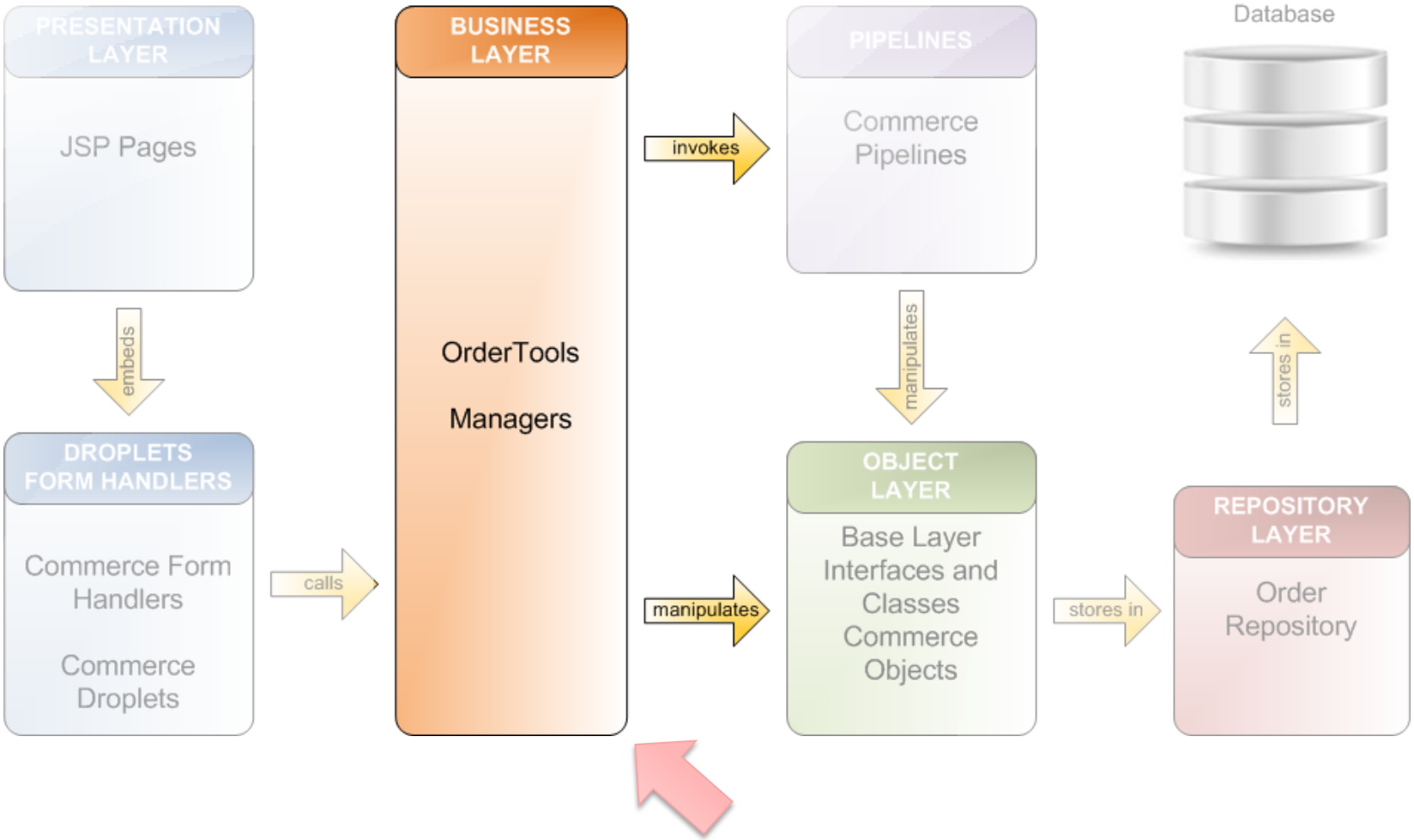
Learning Objectives

At the end of this lesson you should be able to:

- Learn about the Business Layer Classes in ATG Commerce
- Use Order Tools and Manager Classes to manipulate commerce objects
- Use and extend the ATG Commerce pipeline to meet application requirements.
- Understand the various states of order object
- Learn about the order repository

Section 1:
Commerce Business Layer

BUSINESS LAYER



Business Layer Classes (1)

- The business layer classes contain the implementation of business rules for purchase process.
- These methods contain logic that alters the order's data structure and maintains its accuracy.
- All calls to alter an order should be made through these classes.
- They are globally scoped components.
- In general, the main classes consists of OrderTools and managers for order, CommerceItem, Shipping, Payment, and HandlingInstructions.
- These classes are typically called by the commerce form handlers.

Business Layer Classes (2)

- The main components of business layer are:
  - **OrderTools** is a low level interface for editing the order data structure.
  - **OrderManager** contains most of the functionality for working with the order.
  - **SimpleOrderManager** extends OrderManager and provides a very high-level interface for altering an order.
  - **Managers** for CommerceItem, Shipping and Payment, and HandlingInstructions which provide functionality for working with the corresponding commerce objects.
  - **OrderQueries** contains methods for looking up the order.

OrderTools

- OrderTools is a low level interface containing the logic for editing an order data structure.
- It is not meant to contain business logic or to be used directly.
- Developers must use:
  - OrderManager, or
  - SimpleOrderManager.
- These managers simplify access to the OrderTools and provide higher level methods than the OrderTools.
- OrderTools contains a set of properties that you can use to customize the purchase process.
- The default property settings should be suitable for most sites, but can be changed.

OrderManager Component

- OrderManager component is the business layer object for managing and manipulating orders.
- It holds all the business logic for manipulating an order.
- The various types of methods in this class are used for:
  - Creating objects,
  - Adding objects to an order,
  - Adding objects to other objects,
  - Removing objects from other objects.

SimpleOrderManager Component

- SimpleOrderManager component is a simple business layer object for order manipulation.
- Use of this class eliminates the need to know about relationships and only requires basic knowledge of the commerce class hierarchy.
- Logically, this class is one level above the OrderManager class and attempts to do as much as possible for the user.
- The methods in the class allow CommerceItems to be added to or removed from an order and ShippingGroup with a single call.
- It also allows movement of CommerceItems between ShippingGroups.

Example: Using OrderManager and SimpleOrderManager

- Using OrderManager to create an order:

```
// Get a reference to the OrderManager
OrderManager orderManager =
       (OrderManager) request.resolveName
                     ("/atg/commerce/order/OrderManager");
// Create the Order
Order order = orderManager.createOrder(profileId);
```

- Using a SimpleOrderManager to add items to shipping group:

```
// Get a reference to the SimpleOrderManager

SimpleOrderManager simpleOrderManager =
       (SimpleOrderManager) request.resolveName
              ("/atg/commerce/order/SimpleOrderManager");

simpleOrderManager.addItemToShippingGroup (
        order, skuId, productId, quantity, shippingGroup);
```

Business Layer Manager

- Other manager classes used are:
  - **CommerceItemManager** is used for creating and modifying CommerceItems, adding them to orders, etc.
  - **ShippingGroupManager** is used for creating and modifying ShippingGroups, creating relationships, etc.
  - **PaymentGroupManager** is used for creating and modifying PaymentGroups, creating relationships, etc.
  - **HandlingInstructionManager** contains functionality for working with handling instructions including methods to create and remove them.
- OrderQueries contain lookup methods for the order.

Example using CommerceItemManager

- Follow these steps to create a new CommerceItem and associate it with an order:
  - Call CommerceItemManager.createCommerceItem().
  - Make any changes to the CommerceItem.
  - Call CommerceItemManager.addItemToOrder(pOrder, pCommerceItem) to add the CommerceItem to the order.

```
// Get a reference to the OrderManager
OrderManager orderManager = (OrderManager)
    request.resolveName("/atg/commerce/order/OrderManager");

// Create the CommerceItem
CommerceItem commerceItem =
     commerceItemManager.createCommerceItem(pCatalogRefId);
commerceItem.setQuantity(3);

// Add the CommerceItem to the Order
commereceItemManager.addItemToOrder(pOrder, commerceItem);
```

Example of ShippingGroupManager

- Follow these steps to create a new ShippingGroup and add it to an order:
  - Call ShippingGroupManager.createShippingGroup().
  - Make any changes to the ShippingGroup.
  - Call addShippingGroupToOrder to add the ShippingGroup to the order.

```
// Get a reference to the OrderManager
OrderManager orderManager = (OrderManager)
    request.resolveName("/atg/commerce/order/OrderManager");

// Create the ShippingGroup
ShippingGroup shippingGroup =
           shippingGroupManager.createShippingGroup();

// Add the ShippingGroup to the Order
shippingGroup.addShippingGroupToOrder (
                      pOrder, shippingGroup);
```

Example of PaymentGroupManager

- Follow these steps to create a new PaymentGroup and add it to an order:
  - Call PaymentGroupManager.createPaymentGroup().
  - Make any changes to the PaymentGroup.
  - Call addPaymentGroupToOrder to add the payment group to the order.

```
// Get a reference to the OrderManager
OrderManager orderManager = (OrderManager)
   request.resolveName("/atg/commerce/order/OrderManager");

// Create the PaymentGroup
PaymentGroup paymentGroup =
         paymentGroupManager.createPaymentGroup();

// Add the PaymentGroup to the Order
paymentGroupManager.addPaymentGroupToOrder
                              (pOrder, paymentGroup);
```

Section 1 ✔
Check Your Understanding

What does OrderQueries business layer class do?

Answer:

OrderQueries contains lookup methods for the order.

Section 1 
Check Your Understanding

What layer should be used to manage commerce objects?

Answer:

The business layer should be used exclusively to manipulate commerce objects.

Section 1 ✔
Check Your Understanding

What is OrderTools?

Answer:

OrderTools is a low level interface containing the logic for editing an order data structure. It is not meant to contain business logic or to be used directly.

Section 1
Check Your Understanding

What are the various types of methods in the OrderManager?

Answer:

Methods for creating objects, adding objects to an order, adding objects to other objects, removing objects from other objects.

Section 1 ✔
Check Your Understanding

What is SimpleOrderManager component used for?

Answer:

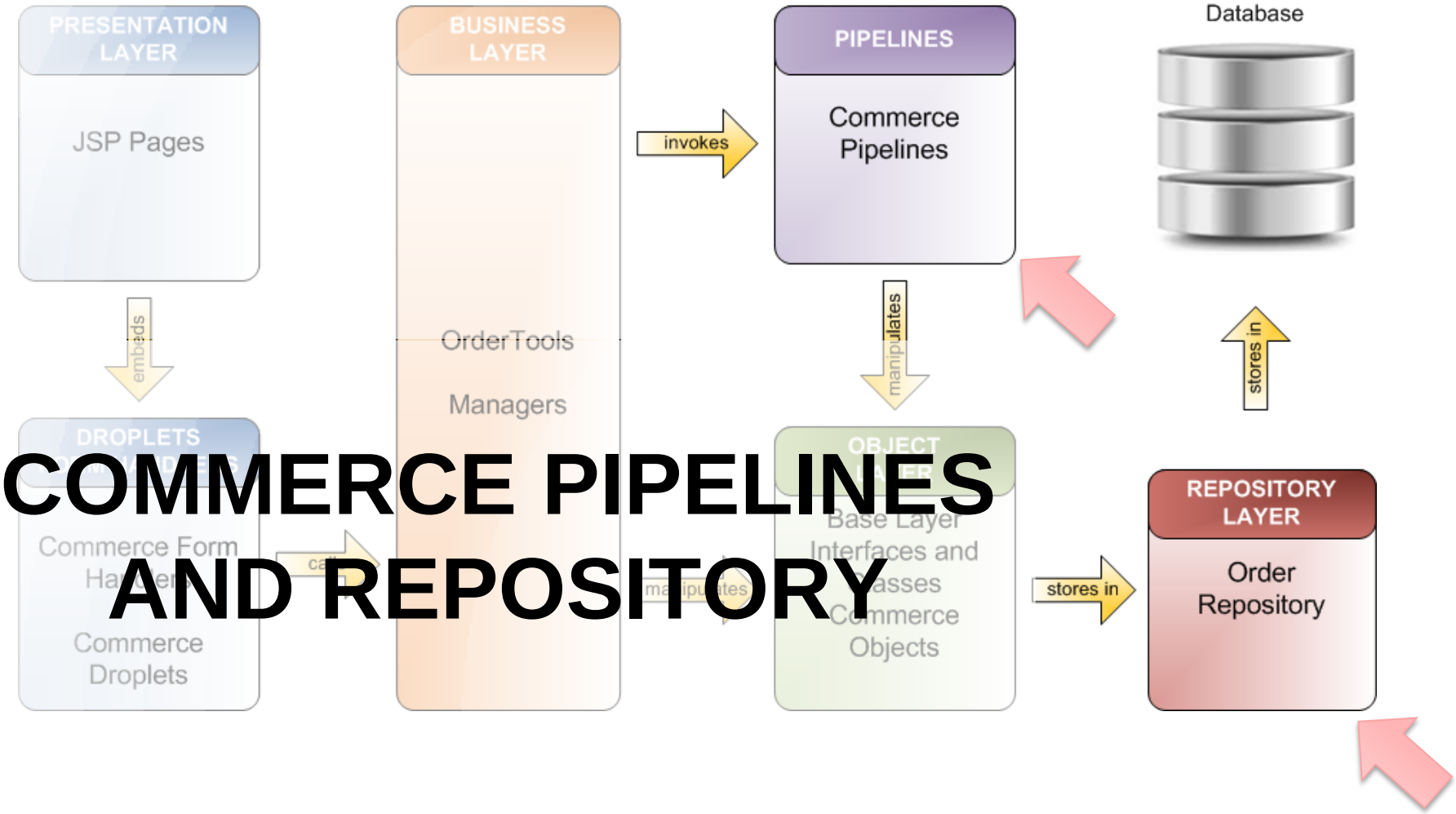
Use of this class eliminates the need to know about relationships and only requires basic knowledge of the commerce class hierarchy.

Summary

- The business layer classes contain the implementation of business rules for the purchase process.
- In general, the main classes consists of OrderTools and managers for Order, CommerceItem, Shipping, Payment, and HandlingInstructions.
- OrderTools is a low level interface for editing the Order data structure.
- OrderManager and SimpleOrderManager provide a high level interface for altering an order.
- CommerceItemManager, ShippingGroupManager, PaymentGroupManager, and HandlingInstructionManager manage the corresponding commerce objects.
- OrderQueries contains lookup methods for the order.

Section 2:
Commerce Pipelines and Repository

COMMERCE PIPELINES AND REPOSITORY



ATG Commerce Pipelines

- A pipeline is an execution mechanism that allows for modular code execution.
- ATG Commerce uses pipelines to execute tasks such as:
  - Loading an order,
  - Saving orders,
  - Checkout orders.
- PipelineManager implements the pipeline execution mechanism.
- It is called by a business layer such as OrderManager, etc.
- It executes a series of operations (processors).

Fundamentals of Pipelines

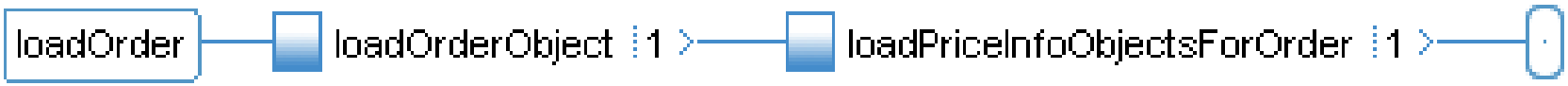
- A pipeline chain is a series of actions to be executed in a sequence.
- Each chain consists of processors, each performing a specific function.
- ATG Commerce has multiple pipeline chains which are executed depending on where the customer is in the purchase process.
- This mechanism allows developers to easily customize the chain by inserting their own processors.
- Pipeline chains are configured in an XML file.
- The PipelineManager component controls processor chains and can be used for customization.

Examples of Core Commerce Pipelines

- Some of the examples of core commerce pipelines are:
  - **loadOrder** pipeline chain loads the order from the repository.
  - **updateOrder** pipeline saves the order.
  - **refreshOrder** pipeline chain reloads an order from the repository.
  - **processOrder** pipeline chain submits the given order for checkout.
  - **repriceOrder** pipeline chain prices the order.
  - **moveToConfirmation** pipeline chain validates the order for order confirmation stage.

Load Order Pipeline Chain

- **LoadOrder** pipeline creates and populates the order object from the repository.
- This pipeline chain is called from the OrderManager's loadOrder() method.
- It contains the following individual processors:
  - **loadOrderObject** : Given an order ID supplied by the PipelineManager, this processor creates an order object and loads its properties from the order repository.
  - **loadPriceInfoObjectsForOrder:** Creates OrderPriceInfo and TaxPriceInfo objects for the given order and loads their properties from the order repository.



ProcessOrder Pipeline Chain

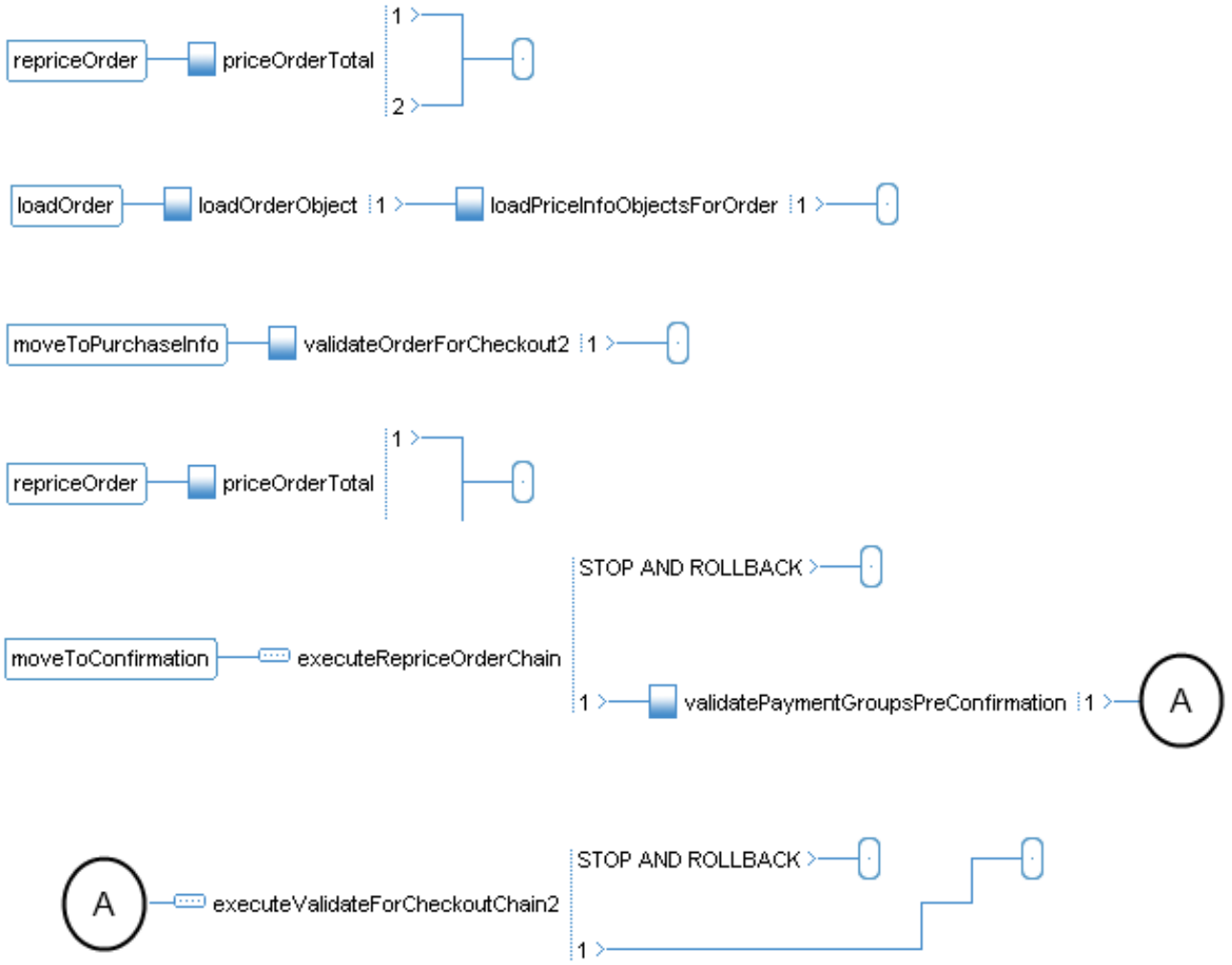
- The process order pipeline chain submits a given order for checkout.
- It is executed by the processOrder() method of the OrderManager.
- This pipeline chain is called when the customer has provided all the methods necessary for checkout and checkouts the order.

Steps in the Process Order Pipeline

- The steps are:
  - **executeValidateForCheckoutChain,**
  - **checkForExpiredPromotions,**
  - removeEmptyShippingGroups,
  - removeEmptyPaymentGroups,
  - createImplicitRelationships,
  - setPaymentGroupAmount,
  - **moveUsedPromotions,**
  - **authorizePayment,**
  - updateGiftRepository,
  - sendGiftPurchasedMessage,
  - setSubmittedSiteId,
  - **addOrderToRepository,**
  - sendPromotionUsedMessage,
  - **sendFulfillmentMessage.**

Process Order Chain

Other Pipeline Examples

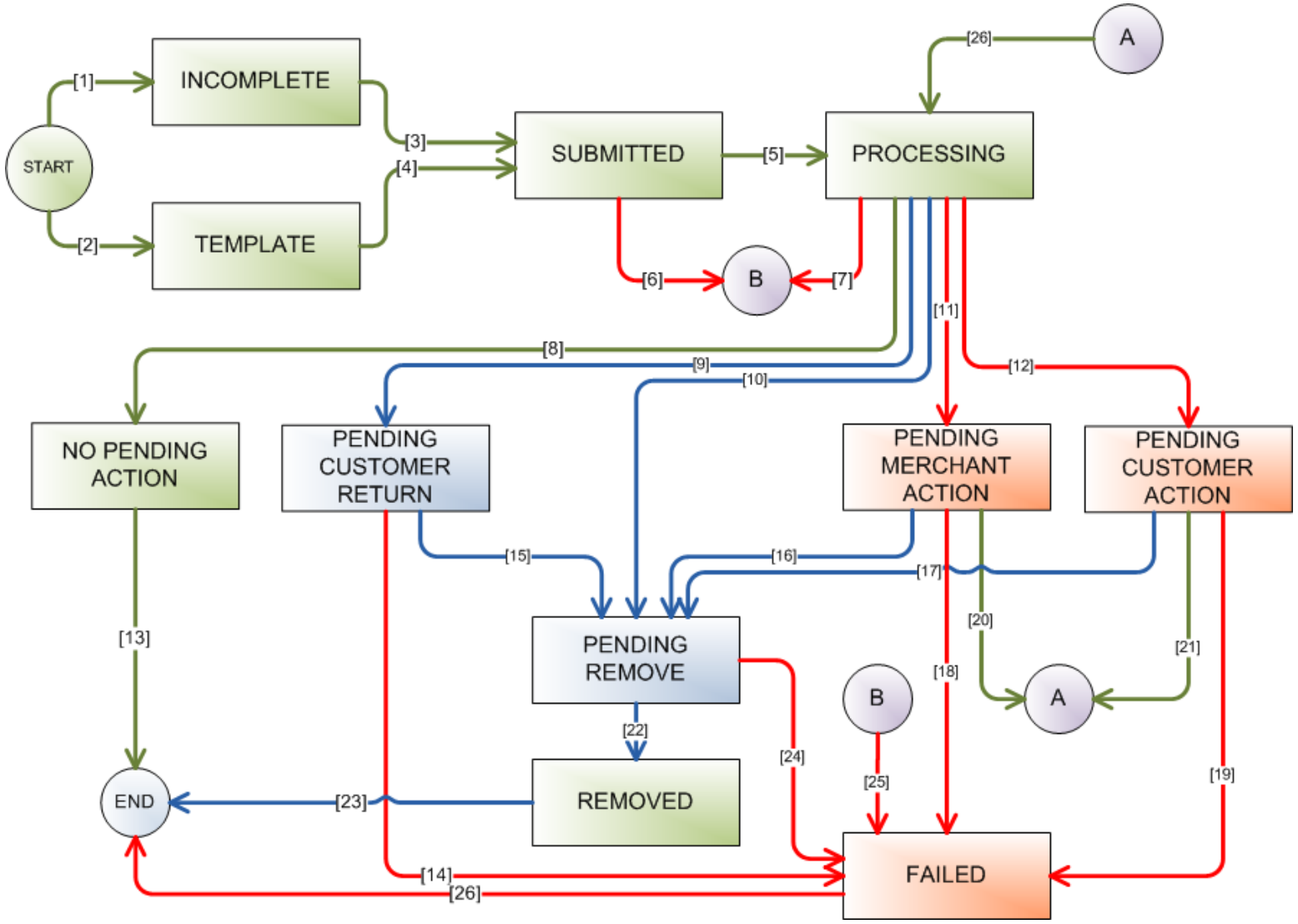


COMMERCE OBJECT STATES

Commerce Object States

- Each commerce object has a set of states that represent the state they are in the processing and fulfillment of an order.
- The objects that have the state are:
  - Order,
  - CommerceItem,
  - ShippingGroup,
  - PaymentGroup,
  - ShippingGroupCommerceItemRelationship.

Order State Transition



Order States Explained (1)

- An order is in INCOMPLETE state when it is created until it is submitted by the user (1).
- When submitted, it is in SUBMITTED state (3).
- A TEMPLATE order (2 and 4) is a scheduled order that can also be SUBMITTED.
- From SUBMITTED state, the order can move to PROCESSING (5) when it is picked up by fulfillment engine.
- From SUBMITTED AND PROCESSING the order could go to FAILED state due to some failures (6 and 7).
- From SUBMITTED the order could go to NO PENDING ACTION (8) and finally end (13). This is the happy path of a successful order.

Order States Explained (2)

- When a customer requests a return and business processes allow it, the order can go to PENDING CUSTOMER RETURN and/or PENDING REMOVE states (9,15 and 10).
- From there an order could go to FAILED (14) or be REMOVED (22).
- When an order cannot be completed without customer interaction such as incomplete address, it goes to PENDING CUSTOMER ACTION state (12).
- Similarly, when an order requires merchant action such as no inventory, it goes to PENDING MERCHANT ACTION state (11).

Order States Explained (3)

- From PENDING MERCHANT ACTION and PENDING CUSTOMER ACTION, the order could go to FAILED (18 and 19) or PENDING REMOVE (16 and 17).
- More commonly, the order would go back to PROCESSING (20 and 21) to resume its processing.
- Typically, not all states are supported or implemented in an eCommerce site.
- Developers customize these states to represent business processes of the merchant.
- Each commerce object has their own states.
- New states can be added as needed.

ORDER REPOSITORY

Order Repository

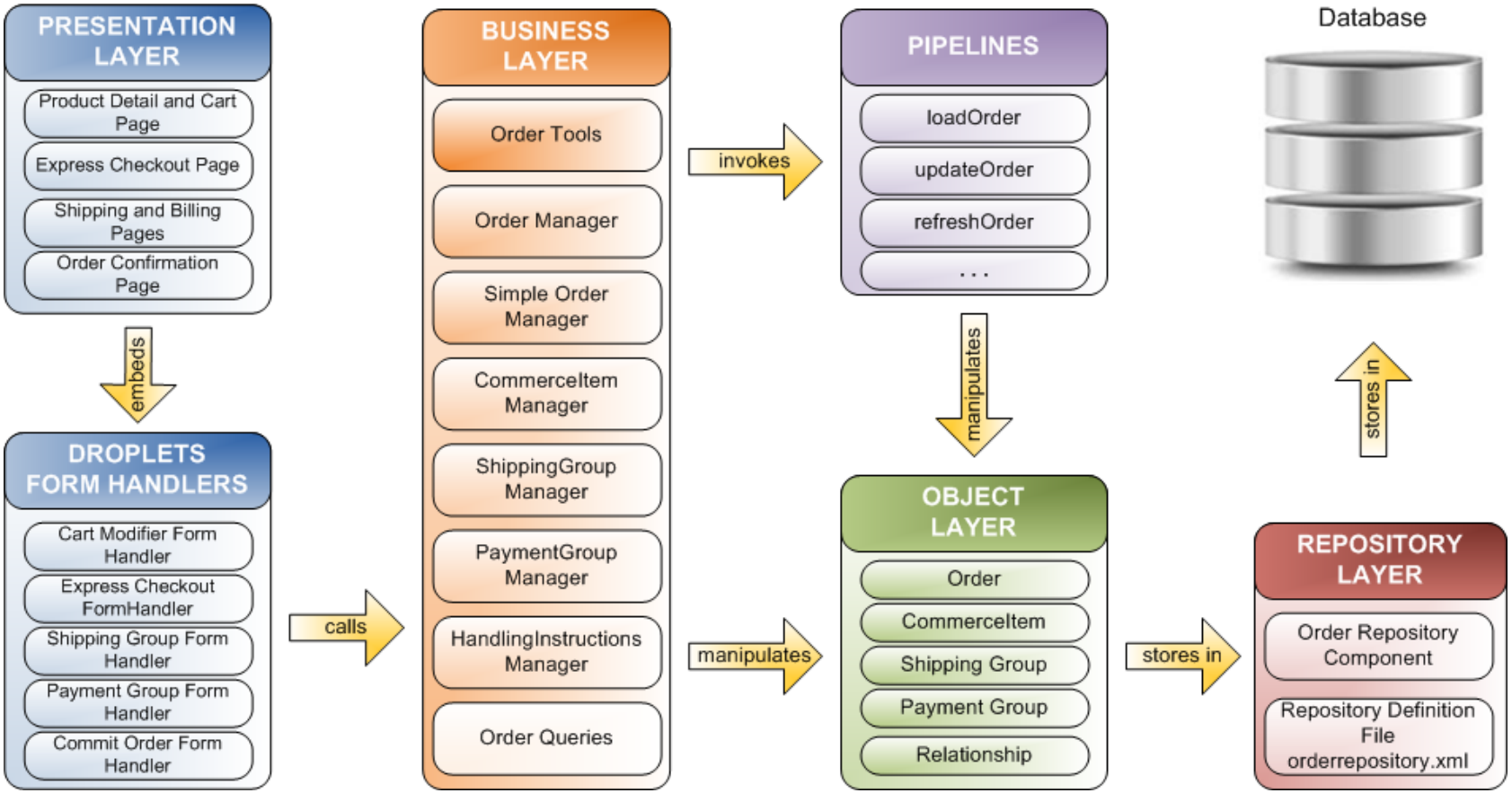
- The order repository is the layer between ATG Commerce and the database server.
- It is a repository where orders in various states are stored. It is implemented as a SQL repository.
- An order object is not a repository item and does not implement the RepositoryItem interface.
- The beanNameToItemDescriptorMap property of the OrderTools component maps the order repository item descriptor to bean names.
- The processors that save and load the order look for this mapping and mapping of corresponding commerce object classes to do their work.
- Order repository is defined in config/atg/commerce/order/orderrepository.xml.

PUTTING IT ALL TOGETHER

The Purchase Process Subsystems

- As you will recall, the purchase process is made of the following subsystems:
  - Base commerce classes and interfaces,
  - Commerce form handlers,
  - Business layer classes,
  - Pipelines,
  - Order repository.
- Each part can be used as is or extended to meet specific business requirements.

Purchase Process Subsystems



Extending the Order Processing System

- Developers may require extensions to the order processing system to handle business requirements unique to their customers.
- Any extensions may change any one of the order processing subsystems:
  - Base layer interfaces,
  - Commerce form handlers,
  - Business layer manager classes,
  - Pipelines,
  - Order repository.
- Extension to all of the above are fairly typical in a commerce implementation.

Extending the Order Repository

- To extend the order repository:
  - Using XML combine, add, modify, or remove properties from the order repository items.
  - Add or change tables and columns in the underlying database definition.
  - Extend the requisite processor that needs the new attributes.
  - Configure the mapping between repository and commerce object bean properties in the OrderTools component.
- Extensions to order repository are common and are no more difficult than extending any other repository in ATG.

Customizing a Pipeline (1)

- The commerce pipeline is defined in the XML definition files **`commercepipeline.xml`**.
- The following steps are required to customize a pipeline chain:
  - Create a processor.
  - Create a nucleus component for the processor.
  - Create or edit the pipeline chain XML file.
  - Register the pipeline chain XML file with the PipelineManager.
- Creating the processor:
  - Implement the interface atg.service.pipeline.PipelineProcessor.
  - The runProcess() method should execute and return a status code.
- Create a nucleus component for your processor element:
  - Register the processor created as a nucleus component.

Customizing a Pipeline (2)

- Name the pipeline processor chain and refer to component in a new or existing XML file.

```
<processor
      jndi="/atg/commerce/order/processor/ABCProcessor"/>
<pipelinelink name="ABCChain">
```

- If you are creating a new chain via an XML file above, register the file with the PipelineManager.

```
definitionFile=/atg/commerce/mycommercepipeline.xml
```

Section 1 
Check Your Understanding

Which relationship commerce object can have a state?

Answer: ShippingGroupCommerceItemRelationship.

Section 1 ✔
Check Your Understanding

What does the process order pipeline do?

Answer:

Validates, authorizes payment, sends to fulfillment, etc.

Section 1 ✔
Check Your Understanding

What is the state of a new order that has not been placed by the customer?

Answer:

INCOMPLETE.

Section 1 ✔
Check Your Understanding

What are the states that require either customer or merchant action?

Answer:

PENDING MERCHANT ACTION and PENDING CUSTOMER ACTION

Section 1 
Check Your Understanding

Where is the commerce pipeline defined?

Answer:

The XML definition files commercepipeline.xml.

Summary

- ATG Commerce uses pipelines to execute tasks such as loading an order, saving orders, check out orders, etc.
- A pipeline chain is a series of actions to be executed in a sequence. Each chain consists of processors, each performing a specific function.
- The PipelineManager component controls processor chains and can be used for customization.
- Each commerce object such as order, commerceitem, shippinggroup, paymentgroup have a state.
- The order goes from INCOMPLETE to SUBMITTED, PROCESSING, and then NO_PENDING_ACTION.

Q&A

ORACLE IS THE INFORMATION COMPANY


ORACLE®