

Implementing Your ATG Commerce Solution Rel 10.1

Student Guide - Volume II

D79786GC10
Edition 1.0
March 2013
D81006

ORACLE®

Authors

Karin Layher

Kerwin Moy

**Technical Contributor
and Reviewer**

Rick Wilson

Editors

Rashmi Rajagopal

Malavika Jinka

Graphic Designer

Maheshwari Krishnamurthy

Publishers

Sujatha Nagendra

Jayanthy Keshavamurthy

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Introduction

- Student Introductions 1-2
- Course Goals 1-3
- Course Agenda: Days One to Three 1-4
- Course Agenda: Days Four and Five 1-5
- Objectives 1-6
- ATG Commerce 1-7
- Flexible Catalog Model 1-8
- Multisite 1-9
- Cross Selling and Upselling 1-10
- SKU Bundles 1-11
- Product Comparisons 1-12
- Gift Lists and Wish Lists 1-13
- Gift Certificates 1-14
- Flexible Pricing 1-15
- Flexible Shipping and Payment 1-16
- Abandoned Order Management 1-17
- Flexible Order Options 1-18
- Commerce-Related Personalization 1-19
- Designed for Integration 1-20
- Commerce Suite Architecture 1-21
- CRS 1-22
- Demonstration: Commerce Reference Store 1-23
- Summary 1-24

2 Commerce Reference Store

- Objectives 2-2
- Road Map 2-3
- Basic Commerce Module 2-4
- CRS Modules 2-5
- Store.EStore Module 2-6
- Store.Storefront Module 2-7
- Road Map 2-8
- CIM 2-9
- Product Selection 2-10

Commerce Addons	2-11
Interim Steps	2-12
Switching or Non-Switching Data Sources	2-13
CRS Addons	2-14
Storefront Addons	2-15
Storefront Sample Data	2-16
Additional Options	2-17
Final Steps	2-18
Quiz	2-19
Road Map	2-20
Using CRS for a New Implementation: Steps 1-3	2-21
Using CRS for a New Implementation: Steps 4-6	2-22
Module Dependencies: MyStore.Storefront	2-23
Module Dependencies: MyStore.Custom.Versioned	2-24
Configuring Store-Wide Properties	2-25
Road Map	2-26
Practice Premise	2-27
Taking It Further	2-28
Summary	2-29
For More Information	2-30
Practice 2 Overview: CRS	2-31

3 Personalization

Objectives	3-2
Road Map	3-3
User-Centric Design	3-4
Profile Object: Session Starts	3-5
Profile Object: Implicit Profiling	3-6
Profile Object: User Logs In	3-7
Profile Object Relationships (Personalization Layer Only)	3-8
Design Decision: User Registration	3-9
Quiz	3-10
Road Map	3-11
Grouping Users	3-12
User Directory	3-13
B2C Example: Organizing by Region	3-14
Linking Users to Organizations	3-15
Segments	3-16
Design Decision: Grouping Users	3-17
Road Map	3-18
Commerce Profile Extensions	3-19

B2C-Related Commerce Profile Extensions	3-20
B2B-Related Commerce Profile Extensions	3-21
Extending the Profile Repository	3-22
Adding a Simple Property	3-23
Step 1: Add an Auxiliary Table	3-24
Step 2: Create a Repository Definition Layer	3-25
Adding a Business Object to the Profile: Example	3-27
Step 1: Add Necessary Tables	3-28
Step 2: Add the Property to the User Item	3-29
Viewing Combined Layers	3-30
Combined Layers in the Component Browser	3-31
Design Decision: Profile Extensions	3-32
Quiz	3-33
Summary	3-34
For More Information	3-35
Practice 3 Overview: Extending Personalization for Commerce	3-36

4 Scenarios

Objectives	4-2
Road Map	4-3
Scenarios	4-4
Scenario Server Architecture	4-5
Scenario Events	4-6
Commonly Used Commerce Scenario Events	4-7
Commonly Used General Scenario Events	4-9
Scenario Actions	4-10
Commonly Used Scenario Actions	4-11
Scenario Conditions	4-12
Scenario: Example	4-13
Extending Scenarios	4-14
Quiz	4-15
Road Map	4-16
Custom Scenario Actions	4-17
Individual Versus Collective Scenario Actions	4-18
Creating Custom Actions	4-19
Coding Custom Actions	4-20
Sources of Information for Actions	4-21
Action Parameters	4-22
Associated Properties File	4-23
Sample Configuration Class and Properties File	4-24
Road Map	4-25

Sample Action Class: getActionName() 4-26
Sample Action Class: initialize() and configure() 4-27
Sample Action Class: executeAction() 4-28
Road Map 4-30
Configuring the Scenario Manager: Defining the Action 4-31
Configuring the Scenario Manager: Configuring Action Parameters 4-32
Configuring the Scenario Manager: Configuring an Action Parameter Value as a Repository Item 4-33
Configuring the Scenario Manager: Configuring the Action Through a Component 4-34
Design Decisions: Scenarios 4-35
Quiz 4-36
Summary 4-37
For More Information 4-38
Practice 4 Overview: Extending Scenarios for Commerce 4-39

5 Catalogs

Objectives 5-2
Road Map 5-3
Product Catalog Repository Item Types: Catalogs and Categories 5-4
Product Catalog Repository Item Types: Linking 5-5
Links Between Catalog-Related Item Types 5-6
Fixed and Dynamic Links 5-7
Catalog Item Types 5-8
Category Item Types 5-9
Selected Category Properties 5-10
Products 5-11
Selected Product Properties 5-12
SKUs 5-13
Selected SKU Properties 5-14
SKU Bundles 5-15
Configurable SKUs 5-16
Common Properties Across Catalog-Related Item Types: General Properties 5-17
Common Properties Across Catalog-Related Item Types: Media Items 5-18
Media Items 5-19
Media Item Properties 5-20
media-external 5-21
media-internal-binary 5-22
media-internal-text 5-23
Road Map 5-24
Ways to Display the Catalog 5-25

Displaying the Catalog (ATG)	5-26
Breadcrumb Trail (ATG)	5-27
Oracle Endeca Experience Manager Components	5-28
Cartridges	5-29
Templates and Cartridges	5-30
Design Decision: Catalog Structure	5-31
Road Map	5-32
Extending the Catalog	5-33
Add Properties to an Existing Item Type	5-34
Create a Subtype of an Existing Item Type	5-35
Create a New Item Type	5-36
Design Decision: Catalog Extensions	5-37
Extending the Catalog: Example	5-38
Item Subtypes	5-39
Extending Catalog Item Types	5-40
1. Create a New Table	5-41
2a. Extend the Type Property	5-42
2b. Define the Subtype Item Descriptor	5-43
3. Add Subtypes to CatalogTools	5-44
Quiz	5-45
Summary	5-46
For More Information	5-47
Practice 5 Overview: Extending the Catalog	5-48

6 Multisite Objects and Site Assignment

Objectives	6-2
Road Map	6-3
Running Multiple Sites	6-4
Multisite: Examples	6-5
Multisite Repository and Item Types	6-6
Multisite Item Types: Site Groups	6-7
Default siteConfiguration Properties: Part One	6-8
Default siteConfiguration Properties: Part Two	6-10
siteConfiguration Properties: DCS Additions	6-11
siteConfiguration Properties: CRS Additions, Part One	6-12
siteConfiguration Properties: CRS Additions, Part Two	6-13
Site Categories	6-14
Road Map	6-15
Customizing Site Repository Items	6-16
Customizing the BCC	6-17
Hiding and Disabling Fields in the BCC	6-18

Road Map	6-19
Site Assignment	6-20
ProfilePropertyPipelineServlet	6-21
Disabling Multisite	6-22
Multisite Responsibilities	6-23
Design Decisions: Multisite	6-24
Summary	6-25
For More Information	6-26
Practice 6 Overview: Customizing and Extending Multisite	6-27

7 Multisite and ATG Applications

Objectives	7-2
Road Map	7-3
New Site Templates	7-4
Site Templates	7-5
Creating a New Site Template	7-6
Creating a New Site Category Item in the BCC	7-7
Setting Default Values	7-8
Road Map	7-9
Ways to Use the Multisite Feature	7-10
Limiting Components by Site Group	7-11
Creating a Component Shareable Type	7-12
Component Shareable Type: Example	7-13
Creating an Abstract Shareable Type	7-14
Ways to Use the Multisite Feature	7-15
Setting Component Properties by Site	7-16
Steps to Configure Component Properties by Site	7-17
Site-Specific Property: Example	7-18
Ways to Use the Multisite Feature	7-19
Using Site Droplets in JSPs	7-20
Accessing the Current Site	7-21
Creating Links to Other Sites	7-22
Retrieving Sites Within a Site Group	7-23
SiteldForItem Droplet	7-24
Ways to Use the Multisite Feature	7-25
Add Listeners to Site Events	7-26
Ways to Use the Multisite Feature	7-27
Add SiteSessionManager Processors	7-28
Ways to Use the Multisite Feature	7-29
Extend Custom Repository Items	7-30
Quiz	7-31

Road Map	7-32
Multisite Extensions to Commerce	7-33
Multisite and Scenarios	7-34
Site-Aware Targeters	7-35
Sites and Catalogs	7-36
Site-Aware Promotions and Coupons	7-37
Summary	7-38
For More Information	7-39
Practice 7 Overview: Creating and Using New Site Categories	7-40

8 Orders: Overview

Objectives	8-2
Road Map	8-3
Orders	8-4
Shopping Cart	8-5
Order Object Versus Order Repository Item	8-6
Order Life Cycle	8-7
The Session Starts	8-8
SKUs Added to the Cart	8-9
(Optional) The User Logs In	8-10
The User Opens the Shipping Page	8-11
(Optional) The User Adds a New Address	8-12
The User Allocates Shipping	8-13
The User Opens the Billing Page	8-14
User Allocates Billing	8-15
(Optional) The User Explicitly Saves the Order	8-16
The User Submits the Order	8-17
(Optional) The User Creates a Scheduled Order	8-18
The Order Is Sent to Fulfillment	8-19
(Optional) The Order Is Held for Approval	8-20
Design Decision: Order Process	8-21
Quiz	8-22
Road Map	8-23
Shipping Methods	8-24
Filtering Shipping Methods	8-25
Design Decisions: Shipping Methods	8-26
Payment Methods	8-27
Design Decisions: Payment Methods	8-28
Summary	8-29
For More Information	8-30

9 Extending Orders

- Objectives 9-2
- Road Map 9-3
- Order-Related Components 9-4
- Purchase Process Form Handlers 9-5
- CartModifierFormHandler Handle Methods 9-6
- CartFormHandler Extensions (CRS) 9-7
- Extending Purchase Process Form Handlers 9-8
- Extending CRS Form Handlers 9-9
- Shopping Cart 9-10
- Ways to Access the Current Order 9-11
- Manager Components 9-12
- CommerceItemManager 9-13
- Commerce Items 9-14
- CommerceItem Object 9-15
- OrderManager 9-16
- Helper Components 9-17
- OrderTools Component 9-18
- Road Map 9-19
- Pipeline Basics 9-20
- Commonly Extended Pipelines 9-21
- Pipeline Manager Components 9-22
- Commerce Pipeline Processors 9-23
- Example: ProcessOrder Chain 9-24
- ProcessOrder Chain Processors 9-25
- Customizing a Pipeline: Steps One and Two 9-26
- Customizing a Pipeline: Step Three 9-27
- Editing Pipelines in the ACC 9-28
- Quiz 9-29
- Road Map 9-30
- Modifying the Order Programmatically 9-31
- Steps to Safely Modify an Order 9-32
- Code Examples 9-33
- Road Map 9-34
- Order Object Versus Order Repository Item 9-35
- Order-Related Java Objects 9-36
- Order Repository 9-37
- Mapping of Classes to Repository Items 9-38
- Selected OrderTools Mapping Properties: Java Class to Repository Item Mapping 9-39

Selected OrderTools Mapping Properties: Object Type Mapping	9-40
Order Property “Flow Through”	9-41
Order Persistence	9-42
Extending Order Persistence	9-43
Summary	9-44
For More Information	9-45
Practice 9 Overview: Extending Orders	9-46

10 Pricing Architecture

Objectives	10-2
Road Map	10-3
Pricing Engine	10-4
Pricing Process for Items	10-5
Pricing in Catalog Objects	10-6
Pricing in Price Lists	10-7
Road Map	10-8
Discounts	10-9
Discounts: Examples	10-10
Determining Who Gets a Potential Discount	10-11
Limiting by Distribution	10-12
Granting Individual Promotions to Users	10-13
Limiting by Conditions	10-14
Ways to Limit Promotions	10-15
Discounts Are Linked to Users	10-16
Differential Pricing Methods	10-17
Design Decision: Pricing	10-18
Summary	10-19

11 Displaying and Extending Pricing

Objectives	11-2
Road Map	11-3
Price Lists	11-4
Price List Architecture	11-5
Assigning Price Lists to Users	11-6
Road Map	11-7
Item Pricing Droplets	11-8
Static Pricing Droplets	11-9
PriceDroplet	11-10
PriceRangeDroplet	11-11
Volume Pricing	11-12
ComplexPriceDroplet	11-13

Dynamic Pricing Droplets	11-14
PricelItemDroplet	11-15
PriceEachItemDroplet	11-16
PricingCommerceltem	11-17
PricelInfo Objects	11-18
Base Class: AmountInfo	11-19
ItemPricelInfo	11-20
DetailedItemPricelInfo	11-21
Road Map	11-22
Order Pricing	11-23
OrderPricelInfo	11-24
Road Map	11-25
Objects That Can Be Priced Dynamically	11-26
How Pricing Works	11-27
Ways to Extend Pricing	11-28
Extending the Pricing Engine: Default Classes	11-29
Extending the Pricing Engine: Critical Methods	11-30
Extending Existing Item Pre-Calculators	11-31
Extending Existing Order Calculators	11-32
Adding Pre- or Post-Calculators	11-33
Extending the PricelInfo Class	11-34
Quiz	11-35
Summary	11-36
For More Information	11-37
Practice 11 Overview: Displaying and Extending Pricing	11-38

12 Promotions Architecture

Objectives	12-2
Road Map	12-3
Promotions: Overview	12-4
Promotions in the BCC	12-5
Stacking Rules	12-6
Road Map	12-7
Discount Process	12-8
Review: How Pricing Works	12-9
Gathering the Promotions List	12-10
Evaluating the Promotions	12-11
When a Price Is Requested	12-12
Sorting the Promotions	12-13
Determining the Qualifier Service to Use	12-14
Creating PricingContextObject	12-15

Calculating the Price Adjustments	12-16
Road Map	12-17
Promotion Item Descriptors	12-18
Promotion Repository Item: General Properties	12-19
Promotion Repository Item: Qualification Properties	12-20
Promotion Repository Item: Availability and Template Properties	12-21
Pricing Engine and Promotions	12-22
Pricing Engine Promotion: General Properties	12-23
Promotion-Related Pricing: Map Properties	12-24
Pricing Engine: Component Properties	12-25
Promotion-Related Pricing Engine Methods	12-26
Pricing Engine Stacking Rule Methods	12-27
PricingContext	12-28
Pricing Model Vetoers	12-29
Method Signatures	12-30
PricingModelComparator	12-31
Qualifier Service	12-32
QualifierService as an Example Qualifier Filter	12-33
PromotionQualifyingFilter	12-34
Discount Calculators	12-35
Existing Discount Calculators	12-36
Quiz	12-37
Summary	12-38
For More Information	12-39
Practice 12 Overview: Extending Order Discount Calculators	12-40

13 Extending Promotions

Objectives	13-2
Road Map	13-3
Promotion Relationships	13-4
Promotion Model Definition Language	13-5
Basic PMDL Structure	13-6
Qualifier Tag	13-7
discount-structure Tag: Percent Off Example	13-8
discount-structure Tag: Volume Discount Example	13-9
Target Tag	13-10
Road Map	13-11
Adding Custom Promotion Types	13-12
Create the Pricing Calculator Class	13-13
Understanding a Pricing Method	13-14
pExtraParameters Map	13-15

DiscountStructure Object	13-16
DiscountDetail Object	13-17
Implementing the Pricing Method	13-18
Summary	13-19
For More Information	13-20
Practice 13 Overview: Custom Promotion Calculator	13-21

14 Promotion Templates

Objectives	14-2
Road Map	14-3
Promotion Templates	14-4
Default Promotion Templates by Type	14-5
Example: BOGO Item Discount	14-6
Example: Tiered Promotion	14-7
GWP Promotion: Example	14-8
Advanced Promotions	14-9
Road Map	14-10
Promotion Template Relationships	14-11
Promotion Template Definitions	14-12
Linking UI Elements to Promotion Properties	14-13
Road Map	14-14
Promotion Template Example: Get Item Discount	14-15
Template Name and Priority	14-16
Screen Segments and Lines	14-17
Defining a Line	14-18
Defining Radio Buttons	14-20
Including and Excluding Product Set Criteria	14-21
Defining Combo Boxes	14-22
Additional Template UI Elements	14-23
Road Map	14-24
Simple Validation with Tag Attributes	14-25
Complex Validation	14-26
Multielement Translators	14-27
Multielement Translator: Example	14-28
multi-element-translators Tag	14-29
Multielement Translator: Example	14-30
PSC Multielement Translator	14-33
Default Translator Components	14-34
Writing a Translator Component Class	14-35
Road Map	14-36
Setting Promotion Properties	14-37

Setting Simple Promotion Properties	14-38
Setting the pmdlRule Property	14-39
Setting the pmdlRule Property Example: Part One	14-40
Setting the pmdlRule Property Example: Part Two	14-41
Setting the pmdlRule Property Example: Part Three	14-43
Including Closeness Qualifiers	14-44
Example: Including Closeness Qualifiers	14-45
Changing or Deleting Templates	14-46
Summary	14-47
For More Information	14-48
Practice 14 Overview: Creating a Custom Promotion Template	14-49

15 Shipping and Shipping Groups

Objectives	15-2
Road Map	15-3
Shipping Examples	15-4
Example A: The Shopper Opens the Shipping Page from the Cart Page on a New Order	15-5
ShippingGroupDroplet (SGD) Is Triggered	15-6
SGD Clears ShippingGroupContainerService (SGCS)	15-7
SGD Calls Initializer Components to Determine the Valid Shipping Groups	15-8
HardgoodShippingGroupInitializer	15-9
SGD Populates Shipping Groups	15-10
Default Initializer Components	15-11
SGD Populates CommercelItemShippingInfos (CISIs)	15-12
CommercelItemShippingInfo	15-13
SGCS Data Is Used to Populate the Shipping Page	15-14
Determining the Available Shipping Methods in the Shipping JSP	15-15
AvailableShippingMethods Droplet	15-16
The Shopper Selects Shipping Options	15-17
Shipping Options Are Changed in the Order	15-18
Example B: The Shopper Returns to the Shipping Page in an Order	15-19
Shipping Information Is Not Cleared	15-20
ShippingGroupDroplet Input Parameters	15-21
ShippingGroupDroplet Output and Open Parameters	15-22
Quiz	15-23
Road Map	15-24
Adding New Shipping Groups	15-25
Splitting Shipping	15-26
ShippingInfoFormHandler	15-27
Example C: The Shopper Ships the Same Item to Multiple Addresses	15-28

Splitting Shipping and CISIs: During the Split	15-29
Splitting Shipping and CISIs: After the Split	15-30
Road Map	15-31
Shipping Groups	15-32
Existing Shipping Group Types	15-33
ShippingGroupImpl Properties	15-34
Additional Shipping Group Properties	15-35
ShippingGroupManager	15-36
Road Map	15-37
Steps to Add a Custom Shipping Group Type	15-38
Creating a ShippingGroupInitializer	15-40
Defining the ShippingGroupInitializer Methods	15-41
Summary	15-42
For More Information	15-43
Practice 15 Overview: Creating a Custom Shipping Group	15-44

16 Pipelines and Shipping Methods

Objectives	16-2
Road Map	16-3
Review: Pipeline Basics	16-4
Invoking a Pipeline	16-5
Calling the runProcess() Method of Pipeline Links	16-6
Pipeline Arguments	16-7
Default Subclasses of ValidatePipelineArgs	16-8
Pipeline Result Object	16-9
Moving to the Next Link in a Chain	16-10
Ending the Pipeline	16-11
Pipeline Manager Components	16-12
Extending a Pipeline	16-13
PipelineProcessor Class	16-14
Example: Extending a Pipeline	16-15
Shipping Validator Example: Resource Bundle and Return Code	16-16
Shipping Validator Example: runProcess and Error Messages	16-17
Shipping Validator Example: Returning SUCCESS	16-18
Extending the validateShippingGroup Chain	16-19
Adding the Return Code and Shipping Group Type	16-20
Road Map	16-21
Shipping Methods	16-22
Shipping Calculator Class Hierarchy	16-23
ShippingCalculatorImpl Properties and Methods	16-24
Existing Shipping Calculators	16-25

Creating a New Shipping Method 16-27
Restricting Existing Shipping Methods 16-28
Quiz 16-29
Summary 16-30
For More Information 16-31
Practice 16 Overview: Customizing Shipping Methods 16-32

17 Payment Groups

Objectives 17-2
Road Map 17-3
Payment Architecture 17-4
Default Payment Collection Process 17-6
CommerceldentifierPaymentInfo 17-7
CRS Payment Collection Process 17-8
BillingInfoFormHandler Methods 17-9
BillingInfoFormHandler Pre and Post Methods 17-10
BillingFormHandler Handle Methods (CRS Only) 17-11
Road Map 17-12
Payment Groups 17-13
Payment Group Class Hierarchy 17-14
CreditCard Properties 17-15
GiftCertificate and StoreCredit Properties 17-16
Payment Manager Classes 17-17
Road Map 17-18
Claimable Payment Groups 17-19
Claimable Repository Item Descriptors 17-20
Claimable Item Descriptor 17-21
GiftCertificateClaimable and StoreCreditClaimable Item Descriptors 17-22
The PromotionClaimable Item Descriptor and the BatchPromotionClaimable Sub-Type 17-23
Road Map 17-24
Adding New Payment Groups 17-25
Changing Allocated Amounts 17-26
PaymentGroupFormHandler 17-27
PaymentGroupFormHandler Example 17-28
Road Map 17-29
Payment Process 17-30
Payment Operation Sequence 17-31
Road Map 17-32
Steps to Add a Custom Payment Group Type 17-33
Summary 17-36

For More Information 17-37

Practice 17 Overview: Creating a Custom Payment Group Type 17-38

18 Other Purchase Process Form Handlers

Objectives 18-2

Road Map 18-3

CancelOrderFormHandler 18-4

CancelOrderFormHandler: Properties and Handlers 18-5

CommitOrderFormHandler 18-6

Road Map 18-7

ExpressCheckoutFormHandler 18-8

Road Map 18-9

Saving Incomplete Orders 18-10

SaveOrderFormHandler 18-11

ShoppingCart Component 18-12

Road Map 18-13

CouponFormHandler 18-14

Summary 18-15

19 Inventory and Fulfillment

Objectives 19-2

Road Map 19-3

Inventory 19-4

Inventory Manager 19-5

Default Inventory Implementation: Architecture 19-6

Default Inventory Implementation: Inventory Item Type 19-7

Default Inventory Implementation: Updating Inventory 19-8

Customizing Inventory Management 19-9

Option 1: Modify inventory.xml 19-10

Option 2: Model a Custom Inventory Repository 19-11

Option 3: Modify the InventoryManager Class 19-12

Example: Localizing Inventory Manager 19-13

Option 4: Write a New InventoryManager Class 19-14

Design Decision: Inventory 19-15

Road Map 19-16

Default Fulfillment Process: Submitting the Order 19-17

Default Fulfillment Process: Notifying Fulfillment 19-18

Default Fulfillment Process: Processing the Order 19-19

Default Fulfillment Process: Sending FulfillOrderFragment Messages 19-22

Default Soft Good Fulfillment Process 19-23

Design Decision: Soft Goods 19-24

Default Hardgood Fulfillment Process: Picking Stock	19-25
Default Hardgood Fulfillment Process: Shipping a Shipping Group	19-26
Integrating with a Shipping System: Options for Notifying the Shipping System	19-27
Integrating with a Shipping System: Options for Communicating with ATG	19-28
Design Decisions: Hard Goods	19-29
Default Payment Processing: Notifying OrderFulfiller of Shipment	19-30
Default Payment Processing: Calling the PaymentManager.debit() Method	19-31
Options for Customizing Fulfillment	19-32
Fulfillment Options	19-33
Design Decisions: Fulfillment	19-34
Summary	19-35
For More Information	19-36
Oracle Commerce Technical Training	19-37

Appendix A: Displaying the Catalog and Breadcrumbs

Objectives	A-2
Road Map	A-3
Catalog Browsing Flow	A-4
Top-Level Categories: Non-CRS	A-5
Top-Level Categories: CRS	A-6
Drilldown Pages	A-7
Commerce Lookup Droplets	A-8
ItemLookupDroplet	A-9
Links to Categories and Products	A-10
Links to Categories and Products: Non-CRS	A-11
Links to Categories and Products: CRS	A-12
CatalogItemLink	A-13
ProductLookupItemLink	A-14
Example: CatalogItemLink	A-15
Road Map	A-16
Breadcrumb Trail	A-17
Key Breadcrumb Trail Objects	A-18
CatalogNavHistory Component	A-19
Breadcrumb Trail In Action: Step One	A-20
Breadcrumb Trail In Action: Clicking a Link	A-21
Example: Maintaining the Stack	A-22
Breadcrumb Trail In Action: Incrementing NavCount	A-23
Breadcrumb Trail In Action: Drilling Down to a Subcategory	A-24
Breadcrumb Trail In Action: Subcategory	A-25
Breadcrumb Trail In Action: Breadcrumb Display	A-26

Breadcrumb Trail In Action: Pop navAction A-27
Using the Back Button: Part One A-28
Using the Back Button: Part Two A-29
Using the Back Button: Part Three A-30
Jumping to Products or Categories A-31
Example: Jumping to Products or Categories A-32
Summary A-33
For More Information A-34

Appendix B: Additional Multisite Information

Objectives B-2
Road Map B-3
SiteContextPipelineServlet B-4
SiteContextRuleFilters B-5
Out-of-the-Box Rule Filters B-6
RequestParameterRuleFilter B-7
URLPatternMatchingRuleFilter B-8
DefaultSiteContextRuleFilter B-9
ProfilePropertyServlet B-10
ProfilePropertySetters B-11
Out-of-the-Box Property Setters B-12
SiteProfilePropertySetter B-13
CatalogProfilePropertySetter B-14
PriceListPropertySetter B-15
StoreProfilePropertySetter B-16
SiteSessionEventTrigger B-17
SiteSessionEventSender B-18
SiteVisitManager B-19
Disabling Multisite B-20
Road Map B-21
Creating a New Site Template B-22
Creating a New Item Mapping B-23
Item Mappings B-24
Item Views B-25
Property View Mappings B-26
Property Views B-27
siteConfiguration Item Mapping B-28
Item Mapping to Item View Mapping B-29
Item View Mapping to Item View B-30
Property View Mapping B-31

Property View B-32
For More Information B-33

Appendix C: Advanced Promotions Topics

Objectives C-2
Road Map C-3
Promotion Import and Export API C-4
Import and Export Templates C-5
Importing and Content Administration C-6
Configuring PublishingWorkflowAutomator Properties C-7
PromotionImportExport Component C-8
PromotionImportExportTools C-9
Import Batch Size C-10
Info Classes C-11
PromotionImportExportInfo Properties: Part One C-12
PromotionImportExportInfo Properties: Part Two C-13
ClosenessQualifierImportExportInfo Properties C-14
CouponImportExportInfo Properties C-15
Accessing the PromotionImportExport Component C-16
Starting and Ending Import and Export Sessions C-17
Import Code C-18
Exporting Promotions by ID C-19
Exporting Promotions by RQL Query C-20
Integrator Components C-21
Integrator Components and PromotionImportExport Methods C-22
Road Map C-23
Extending the PMDL C-24
Attribute Tag C-25
Generic Expression Tags C-26
Creating Custom Expression Tags C-27
Example pmdlParser Mapping C-28
Older ATG Version Expression Tags C-29
Mapping PMDL Elements to Java Objects C-30
Road Map C-31
Gift with Purchase (GWP) Promotions: Overview C-32
Default GWP Templates C-33
Example: Spend Y Get GWP Template C-34
Gift Pricing in the Shopping Cart C-35
GWPManger Component C-36
Storing Information About GWP Promotions C-37

Order Repository Extensions: gwpOrderMarker and gwpCommerceItemMarker	C-38
gwpOrderMarker Item Descriptor	C-39
giftWithPurchase Event Attribute	C-41
Pricing with the GWPManager	C-42
GWP Droplets and Form Handlers	C-43
GiftWithPurchaseSelectionsDroplet	C-44
GiftWithPurchaseSelectionChoicesDroplet	C-45
GiftWithPurchaseFormHandler	C-46
How the Droplets and Form Handler Work Together	C-48
Commerce Reference Store GWP Extensions	C-49
Road Map	C-50
Stacking Rules: Overview	C-51
Pricing Engine Maps Related to Stacking Rules and Included and Excluded Promotions	C-52
Pricing Engine Updates: setupStackingRuleCollection() Method	C-53
DID_PROMOTION_APPLY Parameter	C-54
PricingEngineService Methods Related to Stacking Rules	C-55
Pricing Engines and the applyPromotions() Method: Part One	C-56
Pricing Engines and the applyPromotions() Method: Part Two	C-57
Additional Stacking Rules Information	C-58
Road Map	C-59
Application Messaging: Stacking Rules and GWP	C-60
PricingTools Messaging Properties	C-61
MessageTools Properties	C-62
Messages	C-63
Road Map	C-64
When to Implement CalculatorInfoProvider	C-65
The CalculatorInfo Object	C-66
Constructing a CalculatorInfo Object	C-67
Extracting Values from Custom Attributes	C-68
Summary	C-69
For More Information	C-70

12

Promotions Architecture

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe
 - The properties of promotion repository items
 - The promotion-related classes and components
- Extend a discount calculator



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

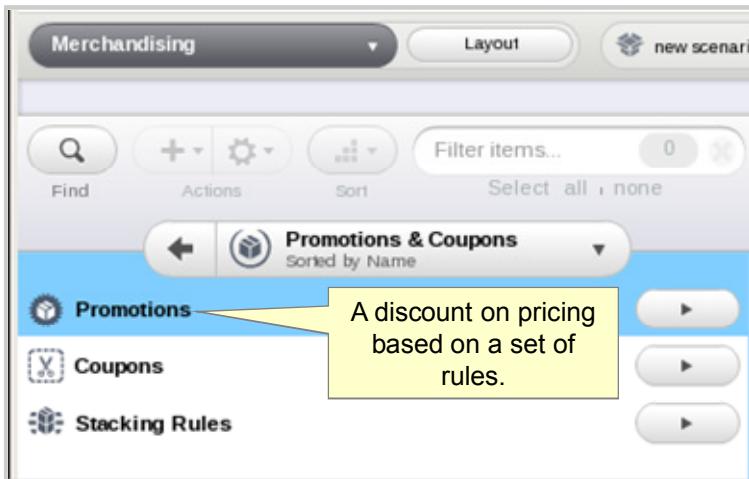
- Promotions overview
- The discount process
- Promotion-related objects and classes



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Promotions: Overview



The three types of promotions are:

- Item discount
- Order discount
- Shipping discount

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Promotions were introduced in the lessons related to pricing. They are a discount on pricing (item, order, or shipping) based on a set of rules. Promotions are defined in the BCC by your business users. You can extend promotions as needed.

The screen shot in the slide shows the Promotions & Coupons area of ATG Merchandising. You can add and edit promotions, coupons, and stacking rules.

Promotions are discounts on pricing based on a set of rules. The discounts can pertain to an item, an order, or the shipping.

Coupons are a code that customers can enter to receive one or more promotions.

Stacking rules are a way to limit multiple promotions.

Promotions in the BCC

The screenshot shows a promotion configuration screen for a template named '\$10 off Orders over \$100'. The 'Template' section is highlighted with a red box. The 'Condition and offer' area is also highlighted with a red box. The 'Condition' section contains a 'Spend' field set to '100 ex. 100'. The 'Offer' section contains a 'Discount Type' radio button selected for 'Amount Off', and a 'Discount' field set to '10 ex. \$5'. The 'Priority' section shows a value of '2'.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Promotions contain many properties, including the template, the condition and the offer defining the set of rules, and the priority. You can also control the promotion's availability, limit it to particular sites, add media for displaying the promotion, and add the promotion to a coupon. You can also specify the priority of the promotion, and include and exclude certain promotions with this promotion.

When business users create a new promotion, they select a template. The template then determines what the condition and the offer area looks like. ATG Commerce provides several out-of-the-box templates. You can also create your own custom templates, which is covered in more detail in the lesson titled "Promotion Templates."

Stacking Rules

ID	Promotion Name	Type
promo20009	Hubbard Chair Tiered Price Break	Item Discount
promo20010	Save 20% on any ATG Home item excluding...	Item Discount
se-promo230003	15% Off Home Accents	Item Discount

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Each stacking rule asset includes one or more promotions that you want to treat as a group. The rules are intended to prevent simple combination strategies (for example, to keep customers from combining “Buy one, get one” (BOGO) offers with gift with purchase offers).

Each promotion can only be associated with a single stacking rule. Each stacking rule can include any number of promotions up to the number set in the “Maximum promotions per stacking rule” property. Within a stacking rule, promotions are applied to orders according to the promotion’s priority.

Stacking rules have the following behavior:

- If an order qualifies for more than one promotion, and those promotions are linked to mutually exclusive stacking rules, the promotion with the highest priority is applied, and all other promotions are skipped.
- If you specify that Stacking Rule X is not combinable with Stacking Rule Y, Y cannot be combined with X.
- If you change the combine list on one stacking rule, the affected stacking rule is automatically updated.
- Any promotion that is linked to a stacking rule that is excluded from use with the current stacking rule cannot be applied.

Road Map

- Promotions overview
- The discount process
- Promotion-related objects and classes



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Discount Process

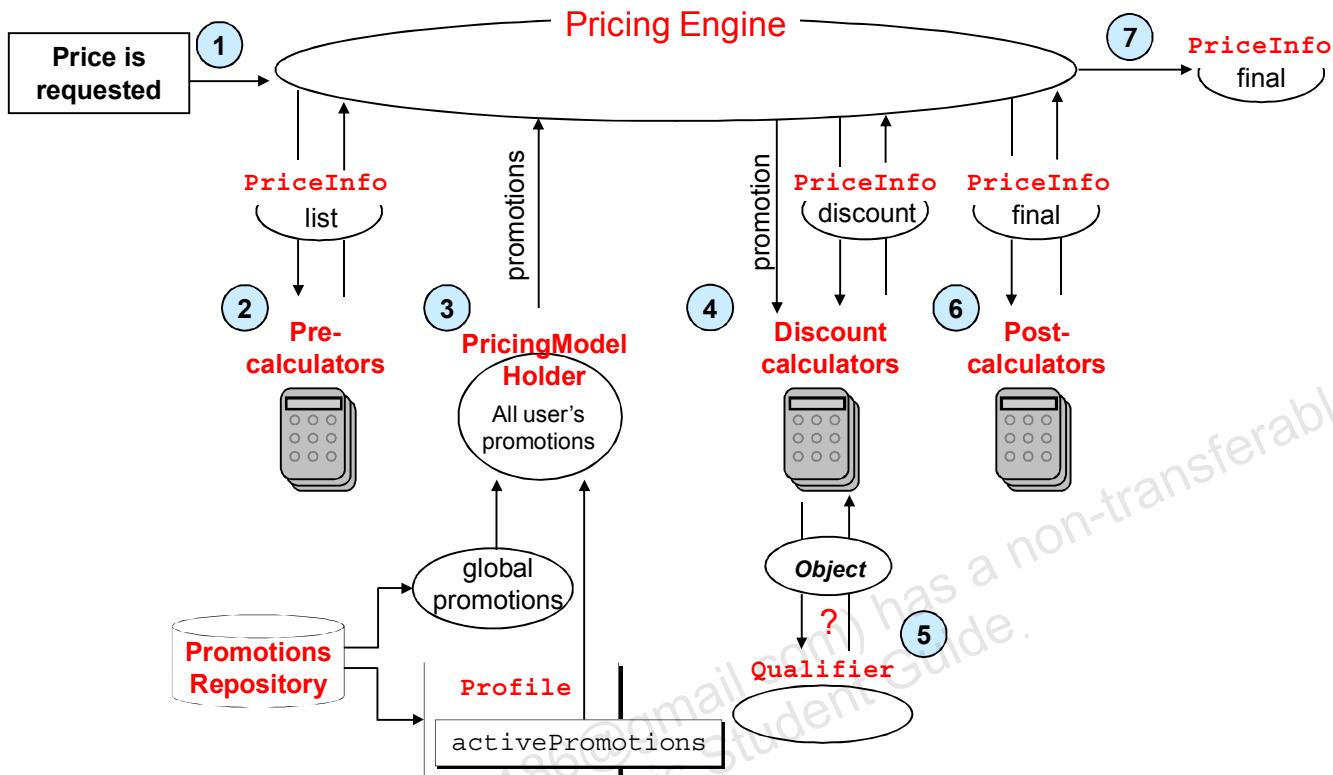
- The pricing engine goes through a series of actions to gather, evaluate, and sort promotions.
- It also needs to take into account promotions when a price is requested.
- The slides in this section looks at the components and objects involved in the actions.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Review: How Pricing Works

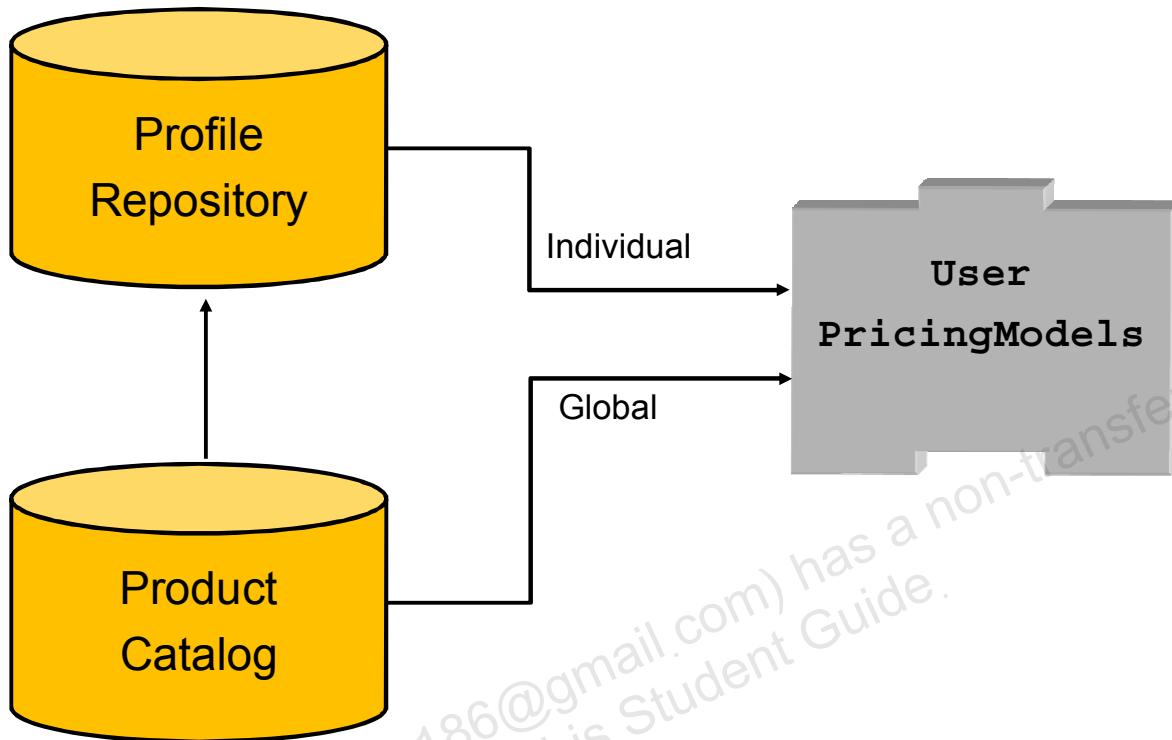


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

1. A pricing operation is requested and the **PricingEngine** is invoked.
2. **PricingEngine** constructs a new **PriceInfo** object, and passes it to pre-calculators which set an initial, undiscounted price in the **priceInfo**.
3. All discount promotions associated with the user are retrieved from the **PricingModelHolder**, which contains all promotions specific to the current user (from the **Profile**), and global promotions available to all users.
4. **PricingEngine** iterates through the user's promotions and invokes each promotion's discount calculator one at a time.
5. Each discount calculator uses the **Qualifier** service to check whether the object being priced qualifies for the promotion associated with that calculator. If so, the price is discounted, and qualified objects are marked as having been used.
6. **PricingEngine** sends **PriceInfo** to post-calculators, which may further alter the price.
7. **PricingEngine** returns the modified **PriceInfo**.

Gathering the Promotions List



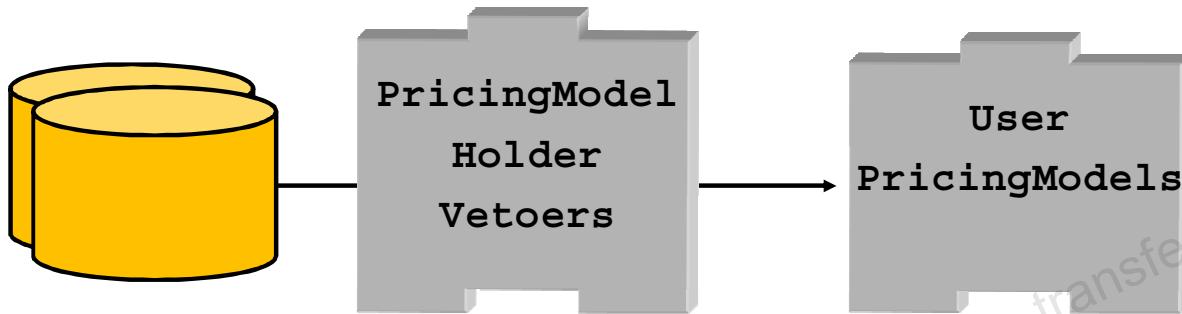
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Each pricing engine gathers the list of individual and global promotions into the `UserPricingModels` component. This list is updated on a schedule, which is set to every five minutes by default.

The full path to the `UserPricingModels` component is `/atg/commerce/pricing/UserPricingModels`. It is session-scoped, and based on the `atg.commerce.pricing.PricingModelHolder` class.

Evaluating the Promotions



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Before the promotions are loaded into `UserPricingModels`, the pricing engine runs each promotion through any of the components listed in its `pricingModelHolderVetoers` property. These components do a quick evaluation to see whether the promotion is usable by the current shopper in the current situation. Out of the box, there are no pricing model holder vetoers specified for any of the pricing engines.

When a Price Is Requested



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When a price is requested, the pricing engine retrieves the list of promotions in the shopper's UserPricingModels component. The promotions are run through another set of vetoer components, the evaluation vetoers (listed in the engine's pricingModelEvaluationVetoers property). Out of the box, each pricing engine is configured to use the /atg/commerce/pricing/SiteVetoer, based on atg.commerce.pricing.PricingModelEvaluationSiteVetoer. This component eliminates promotions that are not valid on the current site.

Sorting the Promotions

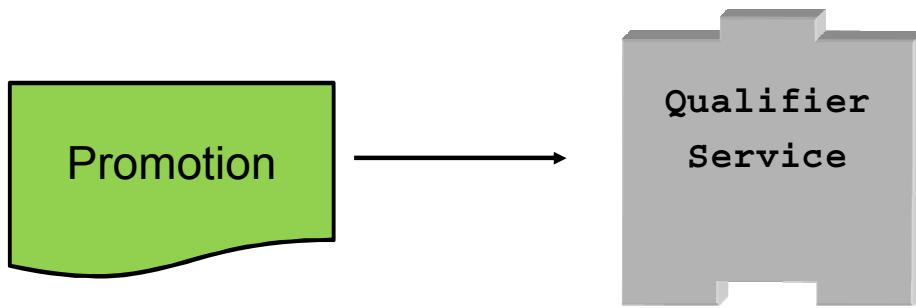


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The promotions are then sorted by the component listed in the pricing engine's `pricingModelComparator` property. By default, the promotions are sorted by the number in their `priority` property; lower numbers are higher priority.

Determining the Qualifier Service to Use

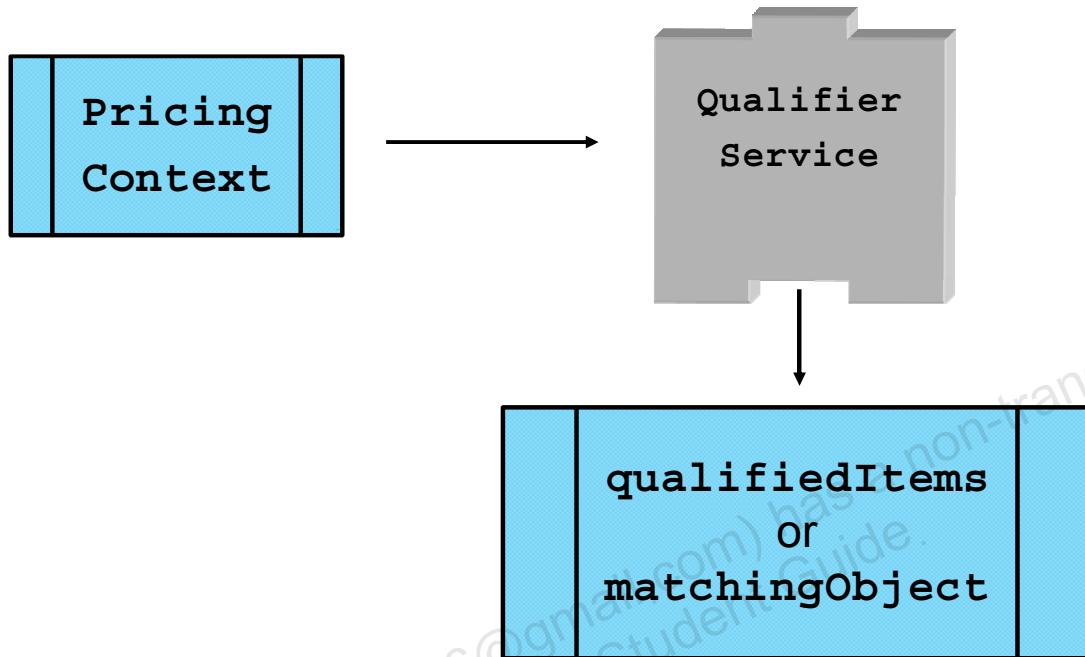


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The pricing engine determines the `QualifierService` component to use for this promotion. This is either the qualifier specified in the promotion's `qualifierService` property or the default qualifier specified in the pricing engine's `qualifierService` property.

Creating PricingContextObject

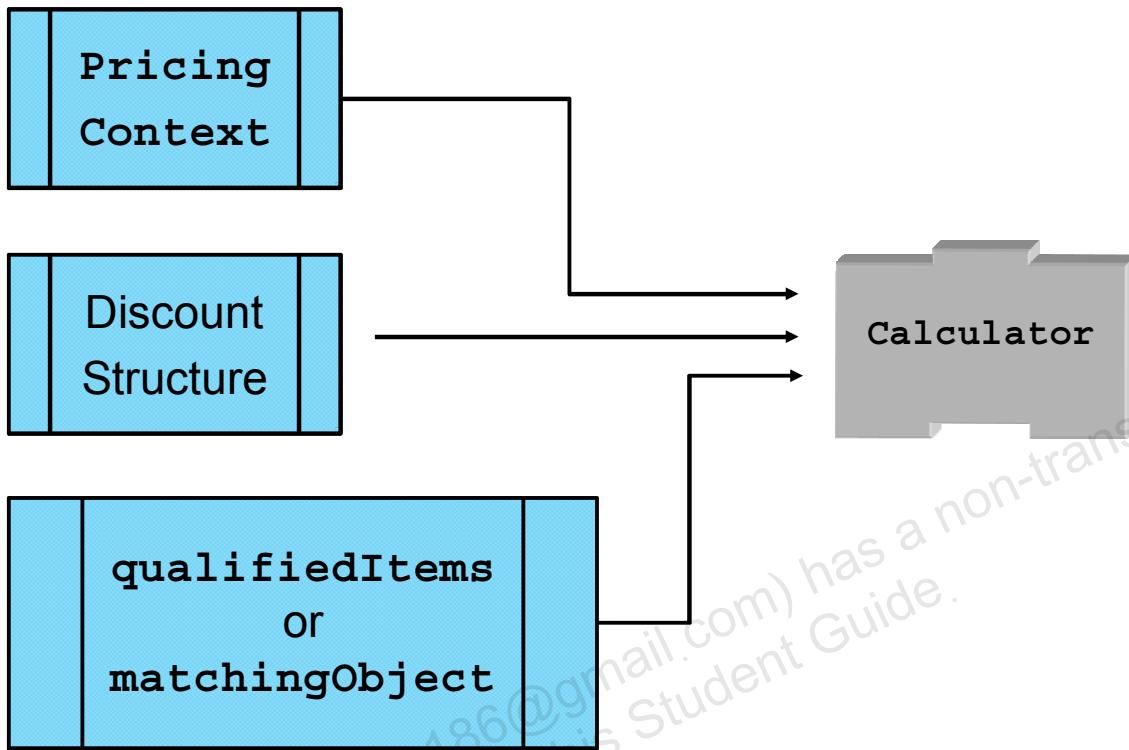


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The pricing engine creates a `PricingContext` object. This object includes a reference to the order, the commerce items, the locale, the current pricing model being evaluated, the site, and the shipping group, as well as the `priceInfo` objects for the order, shipping, and tax. The qualifier then returns either a `matchingObject` (in the case of an order or shipping discount), or a collection of `qualifiedItems` (in the case of an item discount).

Calculating the Price Adjustments



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Each `qualifiedItem` or `matchingObject` is passed to the promotion's calculator, along with the pricing context and the discount structure from the promotion's PMDL rule. The calculator performs the necessary price adjustments.

Road Map

- Promotions overview
- The discount process
- Promotion-related objects and classes

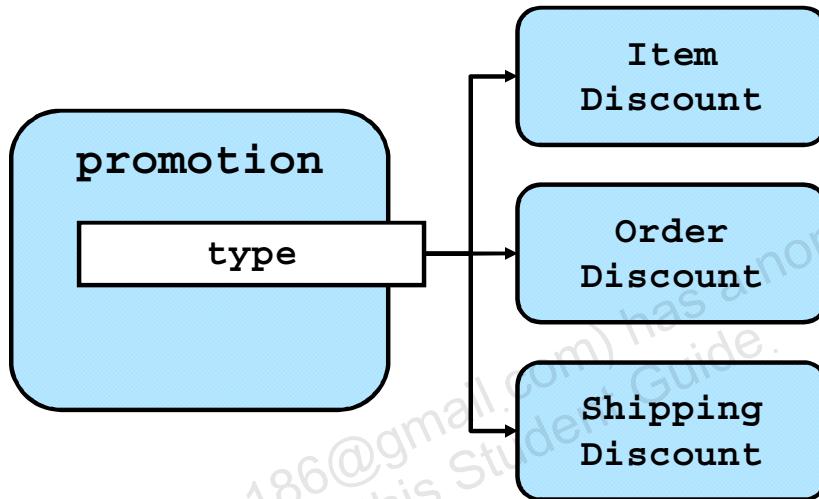


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Promotion Item Descriptors

- Promotions are stored as repository items in the Product Catalog.
- Promotion item descriptors are defined in /atg/commerce/pricing/pricingModels.xml.



ORACLE

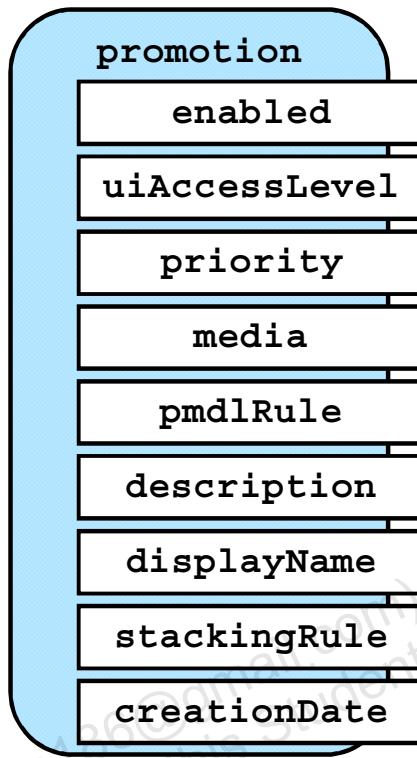
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The base item descriptor is `promotion`, which has three subtypes: `Item Discount`, `Order Discount`, and `Shipping Discount`.

Note: The `ProductCatalog` component has a `definitionFiles` property (plural, instead of singular, as you see in most repository components). That property is set as follows by default:

```
definitionFiles=\n/atg/commerce/catalog/custom/customCatalog.xml,\n/atg/commerce/pricing/pricingModels.xml
```

Promotion Repository Item: General Properties



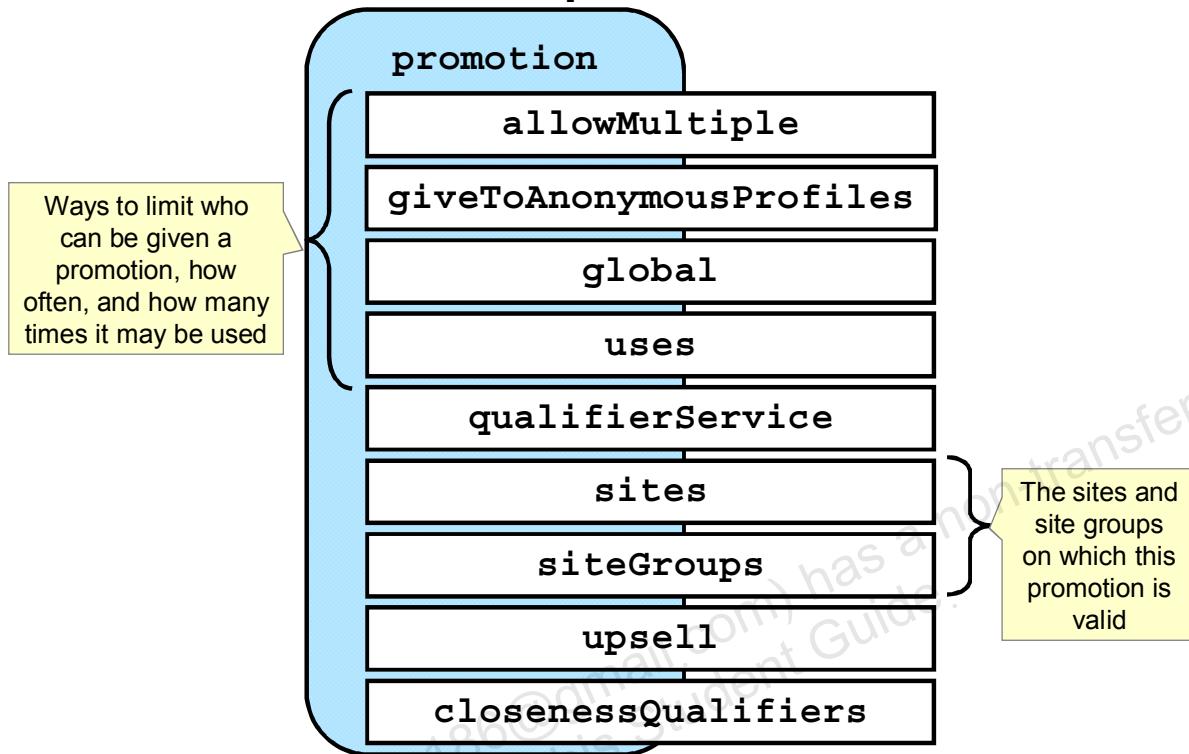
ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The promotion item descriptor has many properties that will be covered over the next few slides. Some key properties include:

- **enabled**: A Boolean that tells you if the promotion is enabled
- **uiAccessLevel**: The access allowed in the BCC: read-only or read-write. This property is especially useful if your promotions come from an outside source.
- **priority**: The order of execution when there are multiple promotions. Lower numbers are processed first.
- **media**: The associated media item
- **pmdlRule**: The rule describing when to apply the discount, and to which items. PMDL is the language in which the rule is written. You will learn more about it later in the lesson.
- **description**: A description of the promotion (optional)
- **displayName**: The name given to the promotion by the business user
- **stackingRule**: The stacking rule to which this promotion belongs, if any
- **creationDate**: The date the promotion was created

Promotion Repository Item: Qualification Properties



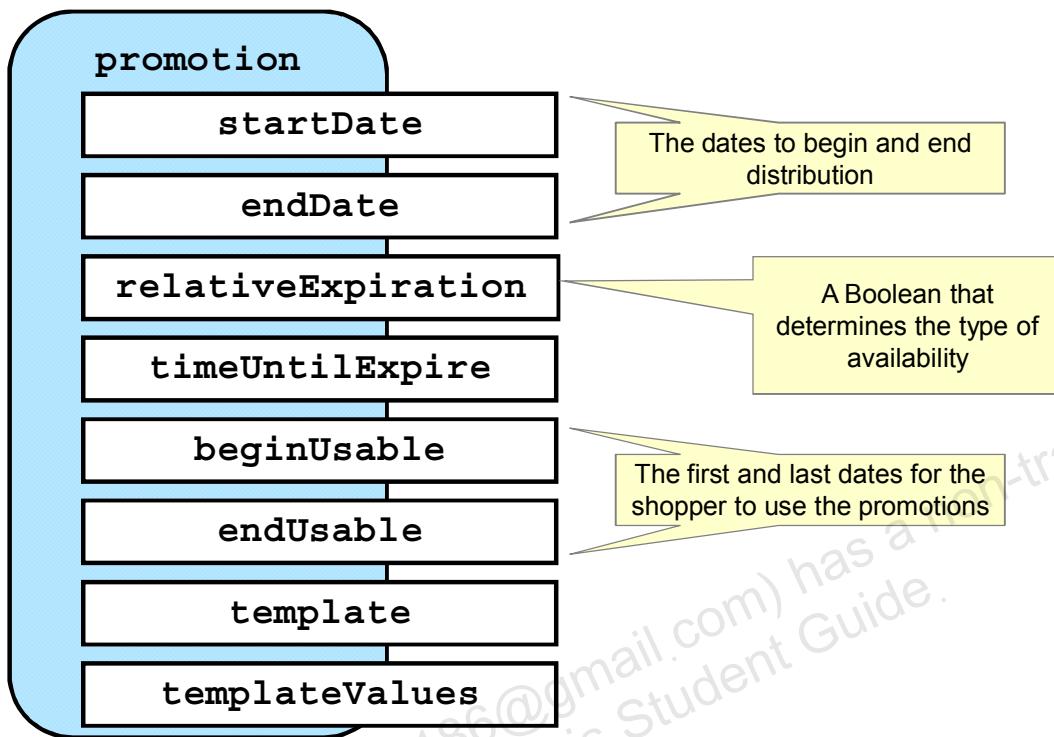
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

This slide continues listing promotion properties, with a concentration on qualifiers and limits:

- allowMultiple: A Boolean that specifies whether a promotion can be given to a shopper more than once
- giveToAnonymousProfiles: A Boolean that specifies whether anonymous shoppers qualify for the promotion
- global: A Boolean that determines whether the promotion is available for all users
- uses: The number of orders this promotion can be used to discount. When global=true, this property is ignored.
- qualifierService: The qualifier service component used by this promotion
- sites: The sites on which this promotion is valid
- siteGroups: The site groups on which this promotion is valid
- upsell: A Boolean that specifies whether closeness qualifiers are being used
- closenessQualifiers: The list of closeness qualifier objects. A closeness qualifier has rules that inform the shopper when the shopper is close to qualifying for a promotion.

Promotion Repository Item: Availability and Template Properties



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The relativeExpiration property is a Boolean. If it is set to true, the timeUntilExpire property is used, which is the number of minutes the shopper has to use the promotion. If it is false, beginUsable and endUsable are used instead.

The following properties are ignored if global is set to true: timeUntilExpire, relativeExpiration, beginUsable, and endUsable.

The template property specifies the BCC UI template. templateValues holds the values for placeholders in the template.

Pricing Engine and Promotions

The `PricingEngineService` class has promotion-related

- Properties:
 - General properties
 - Map properties
 - Component properties
- Methods
 - Promotion methods
 - Stacking rule methods

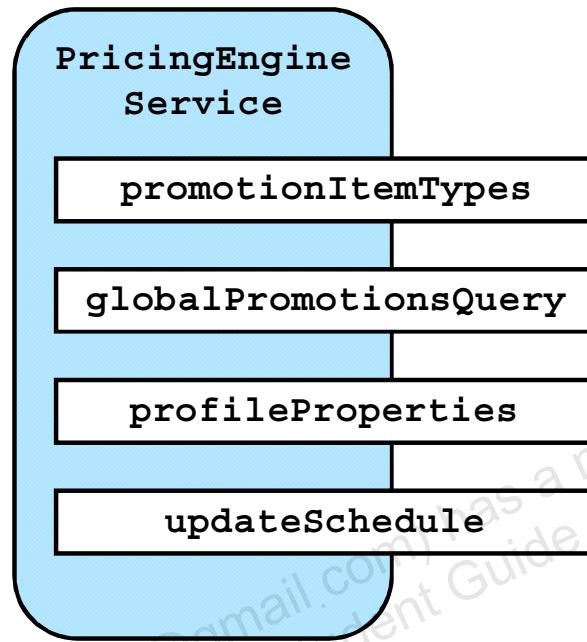


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The promotion-related properties and methods of the `PricingEngineService` are discussed in the next few slides. They have been grouped according to type or function.

For more information on how stacking rules extend the pricing engine, see the appendix titled “Advanced Promotions Topics.”

Pricing Engine Promotion: General Properties



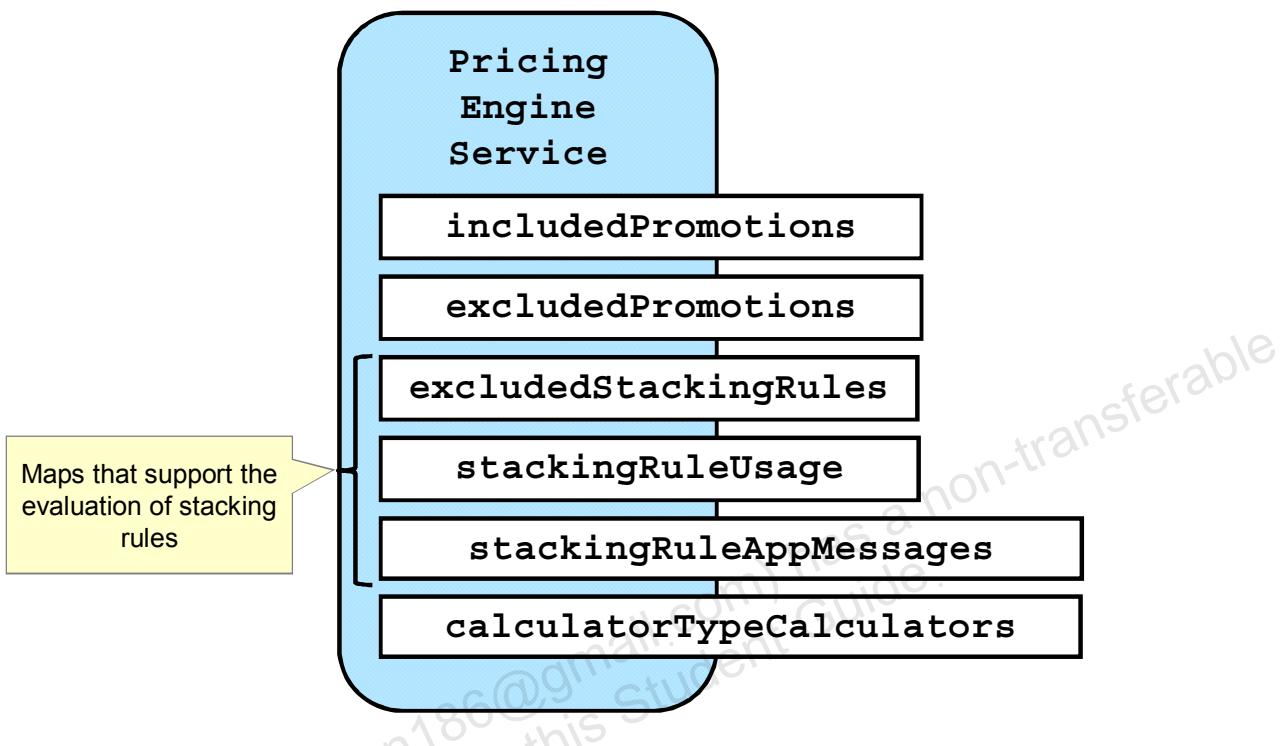
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The pricing engine has many properties and methods related to promotions. The general properties include:

- **promotionItemTypes**: The promotion types used by this engine
- **globalPromotionsQuery**: The query used to retrieve global promotions from the catalog
- **profileProperties**: Profile properties that list individual promotions
- **updateSchedule**: The schedule on which to update the cached list of promotions

Promotion-Related Pricing: Map Properties



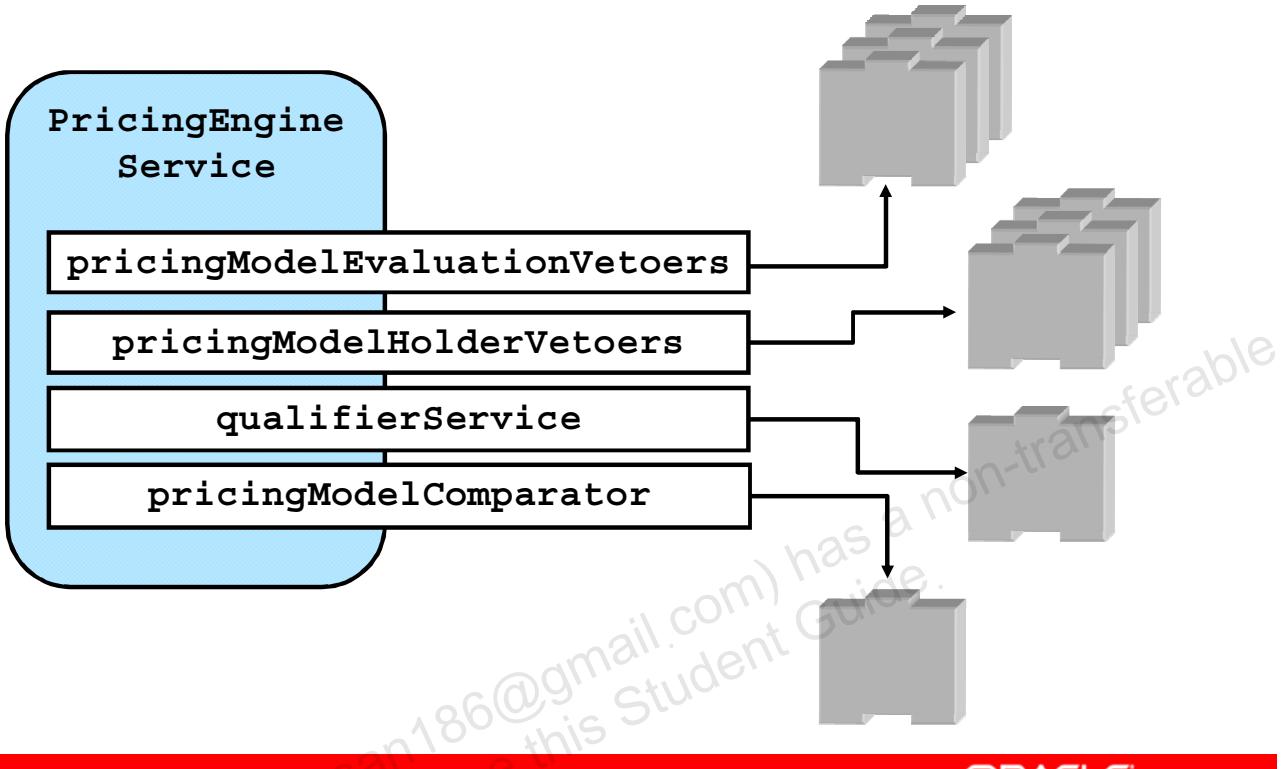
ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The first five map properties support the evaluation of stacking rules:

- **includedPromotions**: Promotions that can be applied (with the promotion currently being evaluated)
 - **excludedPromotions**: Promotions that should be prevented from being applied (with the promotion currently being evaluated)
 - **excludedStackingRules**: Any promotion that belongs to a stacking rule that is in this map should be prevented from being applied.
 - **stackingRuleUsage**: Tracks the usage of each stacking rule, which is in turn used to determine whether a promotion should be prevented from being applied
 - **stackingRuleAppMessages**: A map of application messages relating to those promotions that have been excluded and would normally have been applied
- calculatorTypeCalculators** maps discount types to calculator components.

Pricing Engine: Component Properties



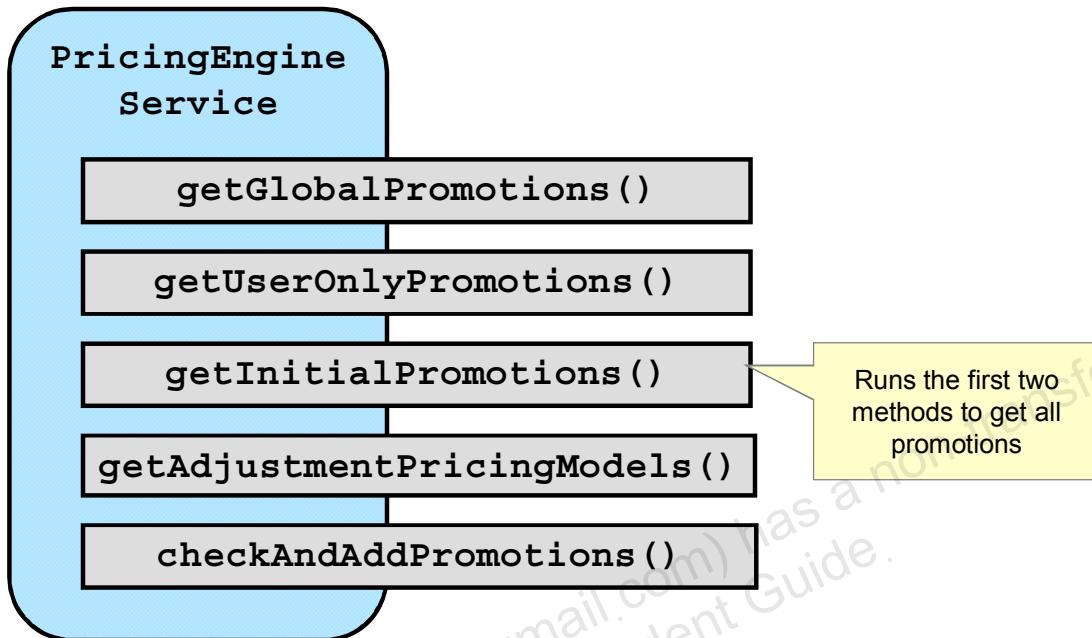
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The properties shown in the slide are properties set to one or more components:

- **pricingModelEvaluationVetoers**: The components that eliminate promotions that are not valid. By default, each pricing engine is configured to use /atg/commerce/pricing/SiteVetoer, based on atg.commerce.pricing.PricingModelEvaluationSiteVetoer. This component eliminates promotions that are not valid on the current site.
- **pricingModelHolderVetoers**: Components that do a quick evaluation to see whether the promotion is usable by the current shopper in the current situation
- **qualifierService**: The default qualifier component. The default pricing engines all set this property to the /atg/commerce/pricing/QualifierService component.
- **pricingModelComparator**: The component that sorts the promotions

Promotion-Related Pricing Engine Methods



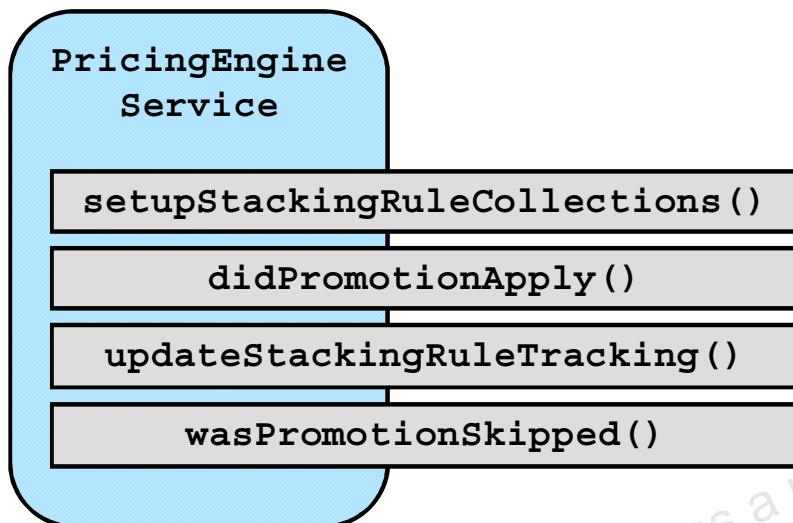
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Promotion-related pricing engine methods include:

- `getGlobalPromotions()`: Finds current global promotions
- `getUserOnlyPromotions()`: Finds current individual promotions
- `getInitialPromotions()`: Runs the `getGlobalPromotions()` and `getUserOnlyPromotions()` methods to get all promotions
- `getAdjustmentPricingModels()`: Gets promotions already applied to the object
- `checkAndAddPromotions()`: Checks that the promotions are not expired

Pricing Engine Stacking Rule Methods



ORACLE

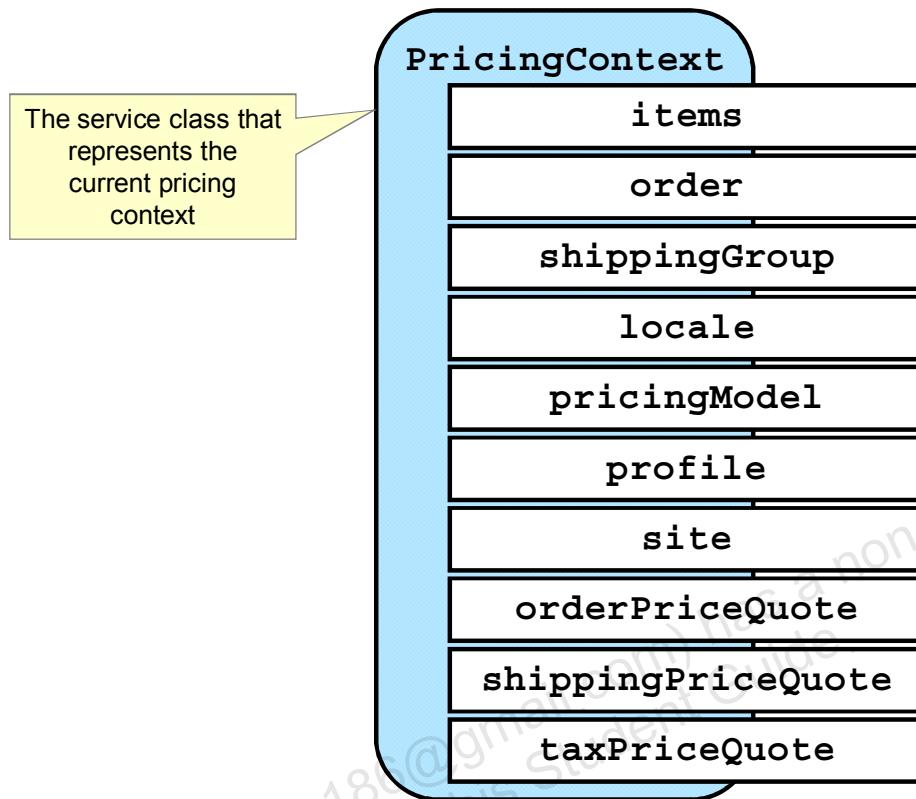
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The properties shown in the slide are properties set to one or more components:

- `setupStackingRuleCollections()` : This method sets up the stacking rule information collections:
 1. The collections are created if they do not exist.
 2. The collections are populated with information related to previously applied promotions.
 3. Messages relating to previously applied promotions are generated.Tracking collections are created in the `pExtraParameters` map in order to be made available to all pricing engines.
- `didPromotionApply()` : Determines whether a particular promotion has been applied, based on a pricing adjustment
- `updateStackingRuleTracking()` : Updates the stacking rule tracking information, such as exclusion details
- `wasPromotionSkipped()` : Determines whether a promotion should be skipped, based on the tracking details

For more information on how these methods work, see the appendix titled “Advanced Promotions Topics” or the promotions viewlet available from the knowledge article titled “Oracle ATG Web Commerce 10.1 Release Training [ID 1490709.1]” on My Oracle Support.

PricingContext



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `atg.commerce.pricing.PricingContext` class is a service class for common parameters passed within methods for a pricing operation. For some method calls, some properties may be null. It provides the items being priced, the order, the site, the promotion (`pricingModel`), the locale, and the profile used when a price was calculated. It can also include secondary information not applicable in all cases, such as the shipping group.

The price quote properties return a `priceInfo` object.

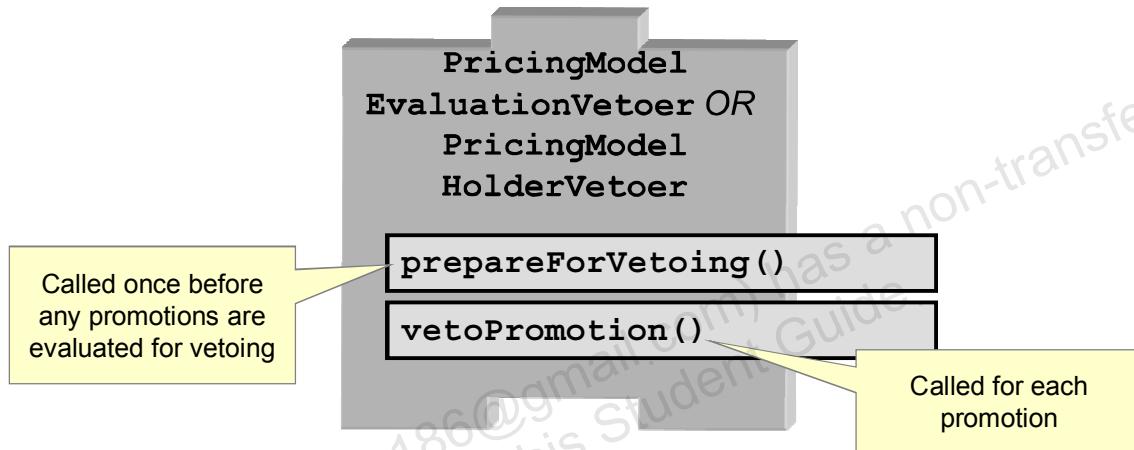
Note: To maintain a level of abstraction, an instance of this class should only be created by using `PricingContextFactory`. Combining the parameters into a single holder class and using factory abstraction provide several advantages such as:

- **Extensibility:** It is easier to extend related APIs.
- **Readability:** Method signatures are shorter and parameters specific to a method stand out.
- **Diagnostics:** Methods can be added to this class to provide diagnostics.

Pricing Model Vetoers

Pricing model vetoers are used to quickly veto promotions that are not valid in the current situation.

- Improves performance of stores with a large number of promotions
- Has two methods:



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Pricing model vetoers implement either

`atg.commerce.pricing.PricingModelEvaluationVetoer` or

`atg.commerce.pricing.PricingModelHolderVetoer` class. They have two methods:

- `prepareForVetoing()`:
 - Called once before any promotions are evaluated for vetoing
 - Can be used to gather needed information about the context, such as the current site
- `vetoPromotion()`: Called for each promotion

Method Signatures

```

Object prepareForVetoing (
    PricingEngineService pPricingEngine,
    PricingContext pPricingContext,
    Map pExtraParametersMap
)

String vetoPromotion (
    PricingEngineService pPricingEngine,
    PricingContext pPricingContext,
    Map pExtraParametersMap,
    Object pVetoParam
)

```

Returns the pVetoParam Object that will be passed into vetoPromotion() or null, if this vetoer should not be used

Returns null if the promotion should not be vetoed or a String describing why the promotion was vetoed



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

`prepareForVetoing()` prepares a vetoer for vetoing promotions. This allows a vetoer to inspect the pricing context and extract any information needed to do its vetoing. For example, a SKU vetoer might extract the SKUs from the order. This information can be returned from this method as an Object, which will subsequently be passed into the `vetoPromotion()` method for vetoing each promotion.

An implementation should only return `null` if its `vetoPromotion()` method should not be subsequently called for each promotion.

The parameters are:

- `pPricingEngine`: The `PricingEngineService` that called this vetoer
- `pPricingContext`: The `PricingContext` for this pricing operation
- `pExtraParametersMap`: The map of extra parameters, if needed

`pVetoParam` is the object returned from `prepareForVetoing()`, guaranteed not to be `null`.

The String returned by `vetoPromotion()` is a debugging message explaining why the promotion was vetoed, such as "User not a member of the loyalty club."

`vetoPromotion()` returns `null` to allow the promotion.

PricingModelComparator

- This is the class that sorts promotions before execution.
 - Instantiated as needed by the pricing engine
- The default for all pricing engines is
`atg.commerce.pricing.PricingModelComparator`
 - Sorts on priority
- Custom comparators must do one of the following:
 - Implement `java.util.Comparator`
 - Extend `PricingModelComparator`



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Qualifier Service

- The QualifierService component is used by pricing engines to determine whether an object being priced qualifies for a particular promotion.
 - Based on the promotion's rules and which objects have already been used as qualifiers or targets of other promotions
- Qualifier filters are used to limit which objects can be used to qualify for a discount (qualifiers) or be discounted (targets).
 - Target filters are only relevant for item discounts.

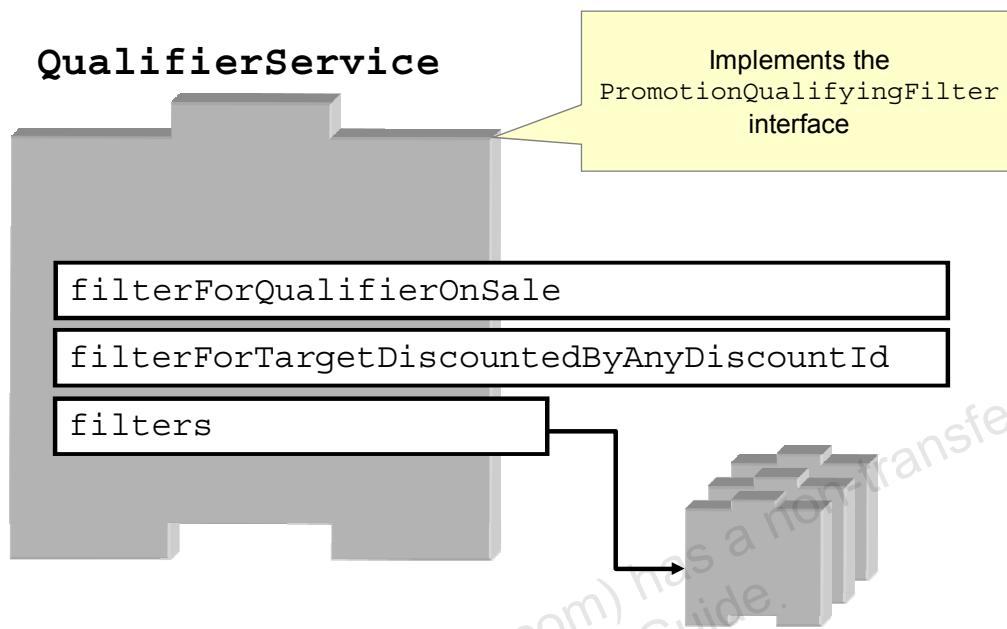


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

A “qualifier” is an object used to qualify for the promotion. A “target” is an object that is discounted. For example, if a promotion specified “buy one table, get three chairs for free,” the table would be the qualifier, and the chairs the targets.

The full name of this class is `atg.commerce.pricing.Qualifier`, which implements the `atg.commerce.pricing.PromotionQualifyingFilter` interface. The component based on it is `/atg/commerce/pricing/QualifierService`.

QualifierService as an Example Qualifier Filter



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

A qualifier filter component must implement the `atg.commerce.pricing.PromotionQualifyingFilter` interface, which the `Qualifier` class mentioned in the previous slide notes does. `QualifierService` has several Boolean properties that filter for qualifiers and targets. A value of `true` indicates that items with the specified attributes should be *excluded* from the qualification process. For a full list of properties, see the *API Platform Reference* for the class. The slide provides two examples of filter properties and an additional property:

- `filterForQualifierOnSale`: A toggle to determine whether items that were priced with a sale price should be allowed to act as qualifiers. It is `false` by default.
- `filterForTargetDiscountedByAnyDiscountId`: A toggle to determine whether items that have been discounted by the current discount should be allowed to receive the discount again (`true` by default). This prevents the same item from being discounted more than once.
- `filters`: A property that can be used to add custom filter components

PromotionQualifyingFilter

- When implementing the PromotionQualifierFilter interface, you need to implement the filterItems() method.
- The code should alter the list of filtered items to remove those that are not appropriate.
- The message signature is the following:

Indicates if a filter is being run for qualifiers or targets

```
public void filterItems(int pFilterContext,  
                      PricingContext PricingContext,  
                      Map pExtraParameters,  
                      Map pDetailsPendingActingAsQualifier,  
                      Map pDetailsRangesToReceiveDiscount,  
                      List pFilteredItems) throws PricingException;
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

```
public void filterItems(int pFilterContext,  
                      PricingContext PricingContext,  
                      Map pExtraParameters,  
                      Map pDetailsPendingActingAsQualifier,  
                      Map pDetailsRangesToReceiveDiscount,  
                      List pFilteredItems) throws PricingException;  
}
```

Discount Calculators

Discount calculators

- Are called by pricing engines once for each qualified item or matching object
- Perform the following operations
 - Adjust the price accordingly.
 - Add the promotion to the object's `priceInfo`'s `pricingAdjustments` property.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Existing Discount Calculators

- The existing discount calculators are:
 - ItemDiscountCalculator
 - OrderDiscountCalculator
 - ShippingDiscountCalculator
 - TaxDiscountCalculator
- They extend DiscountCalculatorService.
- Each calculator also has a subclass for handling volume discounts.

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Quiz

Which of the following statements describe promotions? (Select all that apply.)

- a. All the promotions in a stacking rule are applied to an order.
- b. A promotion's condition contains the rules defining how an order qualifies for the promotion.
- c. Promotions are evaluated in the following order: item, order, shipping.
- d. Promotions can exclude other promotions, but cannot include other promotions.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

Answer a: Only the promotions where the conditions are met are applied to an order. If the stacking rule has two promotions and only one of those promotions qualifies for the order, then that qualifying promotion will be applied. You can also specify in the stacking rule how many promotions from the stacking rule can be applied. For example, if you have two promotions in the stacking rule, but you limit the stacking rule so that only one promotion from the rule can be used, the promotion with the higher priority will apply.

Answer d: A promotion can both include and exclude other promotions.

Summary

In this lesson, you should have learned how to:

- Describe promotion objects
- Extend discount calculators

For More Information

- *ATG Commerce Guide to Setting Up a Store*
 - Creating and Maintaining Promotions
- *ATG Commerce Programming Guide*
 - Using and Extending Pricing Services
- *API Platform Reference*
- Appendix C: Advanced Promotions Topics

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Practice 12 Overview: Extending Order Discount Calculators

This practice covers extending OrderDiscountCalculator to recalculate points after out-of-the-box promotions are applied.



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

13

Extending Promotions

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to implement:

- A custom promotion type
- A custom promotion qualifier filter

Road Map

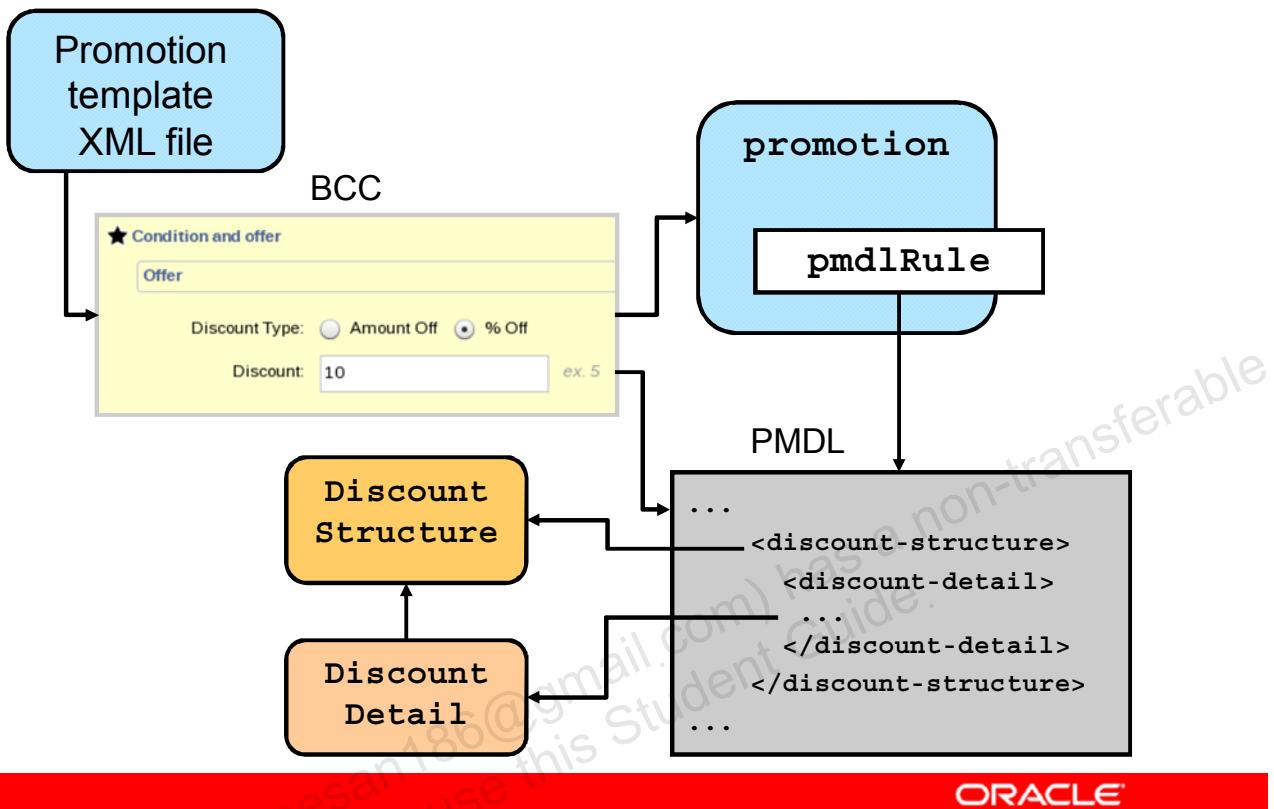
- Promotion Model Definition Language
- Custom promotion types



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Promotion Relationships



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The promotion mechanism is complex. The diagram in the slide shows some of the relationships between different elements of a promotion. At the heart of a promotion is a rule, which is written in Pricing Model Definition Language (PMDL). PMDL is the XML language used to express discounts. The promotion contains the PMDL defining the rule for what things qualify for the promotion and what things are targeted for the discount.

When you select a template in the BCC, it reads the promotion's template configuration from an XML file. The template contains the PMDL rule with placeholders. When the business user creates a promotion in the BCC, the placeholders are filled in and the new PMDL rule is saved with the promotion repository item. The PMDL defines the discount structure, which becomes the **DiscountStructure** object with a map of **DiscountDetails**. A complex promotion, such as volume pricing, would have one or more **DiscountDetail** objects.

Promotion Model Definition Language

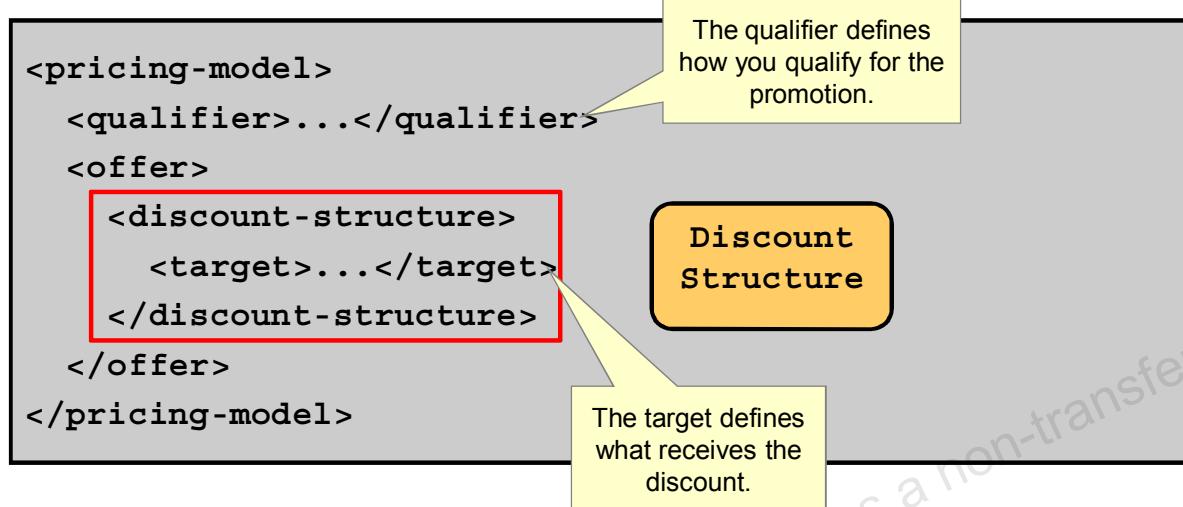
- PMDL is the XML language used to express discounts.
- A promotion's PMDL rule defines:
 - Qualifying conditions, if any
 - Target items (only for item discounts)
 - The discount amount and other details, if needed
 - The discount calculator component
- PMDL is generated by the Promotions UI in the BCC.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

PMDL is the language underlying promotions. When business users create promotions in the BCC, the PMDL is generated. When extending promotions and promotion templates, you may need to use some PMDL. The next slides provide an overview of PMDL for reference when extending promotions. For more information on PMDL, see the section titled “Understanding PMDL Discount Rules” in the “Understanding Promotions” chapter of the *ATG Commerce Programming Guide*.

Basic PMDL Structure



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When defining the PMDL rule, you need to provide the qualifier (such as an item in the cart or a certain total order amount) and the offer. The offer is defined as a discount structure that then defines the target (such as a discount on items in a particular category). **Note:** The target is only defined for item discounts.

The basic tags are:

- `pricing-model`: The root tag for the PMDL
- `qualifier`: Every `pricing-model` element requires one `qualifier` element. The element is the root tag for the condition part of the promotion.
- `offer`: Every `pricing-model` element requires one `offer` element. The offer includes one or more `discount-structure` elements.
- `discount-structure`: This tag appears under the `offer` tag and contains detailed information about the discounts and its target.
- `target`: The target specifies the rule for selecting the items to be discounted. It should be included only when the promotion is for item discounts.

Qualifier Tag

```
<qualifier>
  <comparator name="greater-than-or-equals">
    <value>order.priceInfo.amount</value>
    <constant>
      <data-type>java.lang.Double</data-type>
      <string-value>100</string-value>
    </constant>
  </comparator>
</qualifier>
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

discount-structure Tag: Percent Off Example

```
<offer>
  <discount-structure
    adjuster="25.0"
    calculator-type="standard"
    discount-type="percentOff">
    <target>...</target>
  </discount-structure>
</offer>
```

calculator-type
options are
standard, bulk, and
tiered



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Under the `offer` tag, the `discount-structure` tag holds the actual discount information. The optional `adjuster` attribute specifies the price adjustment to make for the discount, and the `discount-type` specifies how the adjustor value is applied. The options for `discount-type` include `percentOff`, `amountOff`, and `fixedPrice`. In the example in the slide, the discount will be 25 percent off.

The `discount-structure` tag may include `discount-detail` tags, which is discussed in the next slide.

discount-structure Tag: Volume Discount Example

```
<discount-structure  
calculator-type="bulk"  
discount-type="percentOff">  
    <discount-detail>  
        <attribute name="band" value="2"/>  
        <attribute name="adjuster"  
            value="10"/>  
    </discount-detail>  
    ...more discount detail tags...  
</discount-structure>
```

One discount-detail element per "band" or tier.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows a bulk discount. There would be one discount-detail element for each band of the discount. In the example, if two or more chairs were bought, each chair would be 10% off. If four or more were bought, each chair would be 15% off; six or more, 20%. Full example:

```
<discount-structure calculator-type="bulk"  
discount-type="percentOff">  
    <discount-detail>  
        <attribute name="band" value="2"/>  
        <attribute name="adjuster" value="10"/>  
    </discount-detail>  
    <discount-detail>  
        <attribute name="band" value="4"/>  
        <attribute name="adjuster" value="15"/>  
    </discount-detail>  
    <discount-detail>  
        <attribute name="band" value="6"/>  
        <attribute name="adjuster" value="20"/>  
    </discount-detail>  
</discount-structure>
```

Target Tag

```
<target>
  <every sort-by="priceInfo.listPrice"
    sort-order="ascending">
    <collection-name>items</collection-name>
    <element-name>item</element-name>
    <includes>
      <value>
        item.auxiliaryData.productRef.ancestorCategoryIds
      </value>
      <constant>
        <data-type>java.lang.String</data-type>
        <string-value>cat50001</string-value>
      </constant>
    </includes>
  </every>
</target>
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Target elements are only included in item discounts. In the example in the slide, every item that meets the criteria specified in the `includes` tag will be discounted. The `includes` tag specifies items with `cat50001` (Women's Clothing) in their ancestor categories. The items will be sorted ascending by list price, so that lower-priced items are discounted first.

The `item` is a commerce item. `item.auxiliaryData` refers to the particular sku repository item. The `value` tells the promotion to look at the ancestor category IDs for the SKU. The `constant` then supplies the particular ancestor category ID to look for.

Road Map

- Promotion Model Definition Language
- Custom promotion types



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Adding Custom Promotion Types

The steps to add custom promotion types are:

1. Create a pricing calculator class and component.
2. Add the calculator to the pricing engine's `calculatorTypeCalculators` property.
3. If needed, create a custom qualifier component.
4. (Optional) Create a custom promotion template.
 - Promotion templates will be covered in greater detail in the lesson titled “Promotion Templates.”



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Create the Pricing Calculator Class

- Implement the appropriate pricing calculator interface.
- Optionally, implement the `CalculatorInfoProvider` interface.
- Extend `DiscountCalculatorService`.
- Implement the methods called by the pricing engine, such as `priceOrder` and `priceItem`.
- Configure a new component based on the new calculator class.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You need to implement the `CalculatorInfoProvider` interface if your custom calculator requires complex input. For more details on how to implement the interface, see the appendix titled “Advanced Promotions Topics.”

Understanding a Pricing Method

```
void priceOrder(  
    OrderPriceInfo pPriceQuote,  
    Order pOrder,  
    RepositoryItem pPricingModel,  
    Locale pLocale,  
    RepositoryItem pProfile,  
    Map pExtraParameters)
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

One of the methods that you might need to extend is the `priceOrder()` method. The code in the slide shows the method signature. One of the things you need to pass in is the map of `pExtraParameters`. The next slide discusses the promotion-related keys of `pExtraParameters`.

pExtraParameters Map

- The pExtraParameters map has the following keys:
 - Constants.QUALIFIED_ITEMS (commerce items) or Constants.MATCHING_OBJECT (order or shipping group)
 - One or more objects to be discounted
 - Constants.DISCOUNT_STRUCTURE
 - DiscountStructure object
- The constants are defined in atg.commerce.pricing.Constants.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

DiscountStructure Object

- The DiscountStructure object represents a discount-structure element and values from PMDL.
- Properties include:
 - adjuster: A double that contains the price adjustment to apply
 - attributes: A Map of extra attributes that apply to this discount type, such as the bandingProperty to use
 - discountDetails: A List of custom discount details added to PMDL, modeled as DiscountDetail objects

Discount
Structure

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

DiscountDetail Object

- The DiscountDetail object represents discount-detail elements and values from PMDL.
- Properties include:
 - name: A String
 - attributes: A Map of name/value property entries describing this detail

Discount
Detail

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Implementing the Pricing Method

In general, the discount pricing method:

- Gets the matching object or qualifying items
- Gets the necessary information out of the discount structure and details
- Calculates pricing adjustment
- Sets new price in `priceInfo` object(s)
- Sets `priceInfo.discounted` to true
- Creates a new `PricingAdjustment` object and adds it to `priceInfo's` `adjustments` property



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe PMDL
- Add a new promotion types

For More Information

- *ATG Commerce Guide to Setting Up a Store*
 - Creating and Maintaining Promotions
- *ATG Commerce Programming Guide*
 - Using and Extending Pricing Services
- *API Platform Reference*



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Practice 13 Overview: Custom Promotion Calculator

This practice covers creating a custom promotion calculator that calculates total points earned based on the promotion.



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable
license to use this Student Guide.

14

Promotion Templates

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the use of promotion templates in the BCC
- Create a custom promotion template

Road Map

- Promotions templates in the BCC
- Creating custom promotion templates
- Defining the template UI
- Validating and translating user input
- Linking user input to promotion properties



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Promotion Templates

- Templates ease the creation of commonly used promotions, such as “Buy one, get one” (BOGO).
- Developers can easily add custom promotion templates.
- Promotions that do not use a template can be created by using the advanced template.
- Promotion templates are organized by type:
 - Item discount
 - Order discount
 - Shipping discount

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Default Promotion Templates by Type

<h2>Item Discount</h2> <div style="border: 1px solid #ccc; padding: 10px;"><p>Get Item Discount (by Product, Category, Brand, etc.) BOGO Buy Item X Get Item Y Spend Y Get Item Discount Spend Y In X Get Item Discount Buy Within Specified Time Frame Get Item Discount</p><hr/><p>Get Gift With Purchase (GWP) Buy Item X Get GWP Spend Y Get GWP Spend Y In X Get GWP</p><hr/><p>Tiered Price Break</p><hr/><p>Advanced Condition & Offer</p></div>	<h2>Order Discount</h2> <div style="border: 1px solid #ccc; padding: 10px;"><p>Get Order Discount Buy X Get Order Discount Spend Y Get Order Discount Spend Y In X Get Order Discount Buy Within Specified Time Period</p><hr/><p>Tiered Order Discount</p><hr/><p>Advanced Condition & Offer</p></div>
<h2>Shipping Discount</h2> <div style="border: 1px solid #ccc; padding: 10px;"><p>Get Shipping Discount Buy X Get Shipping Discount Spend Y Get Shipping Discount Spend Y In X Get Shipping Discount Buy Within Time Period Get Shipping Discount</p><hr/><p>Advanced Condition & Offer</p></div>	



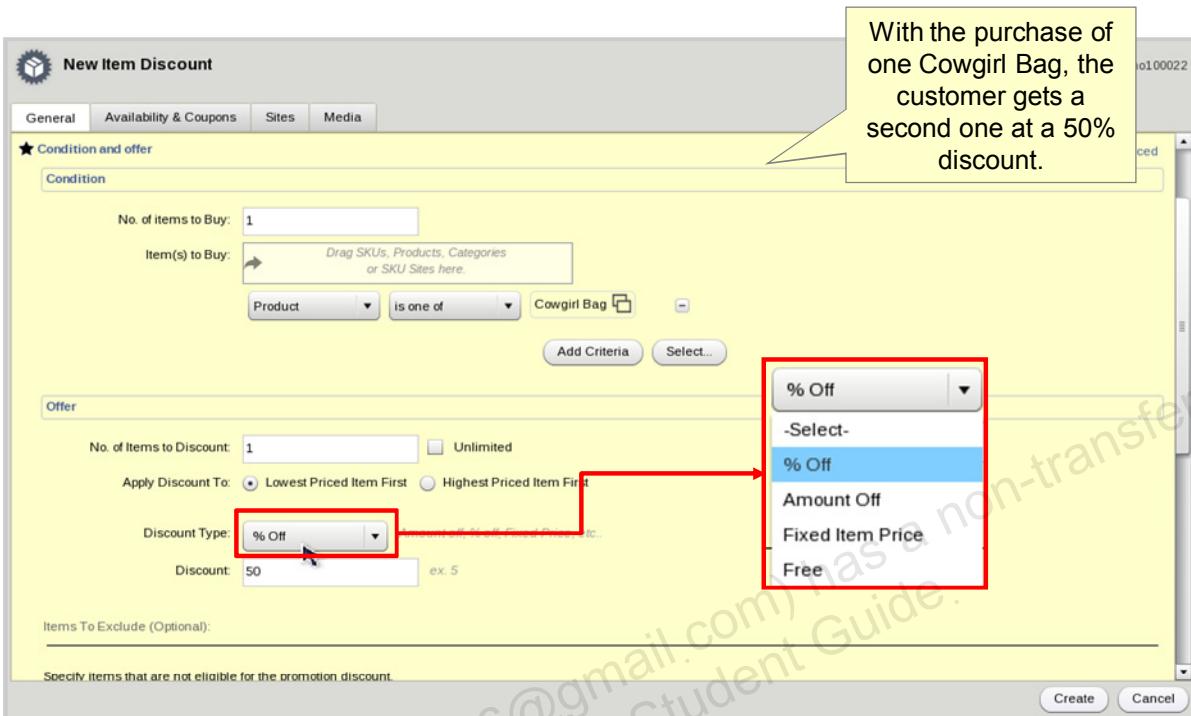
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

A number of promotion templates are available out of the box, organized by promotion type. You can create your own templates, which will be discussed later in this lesson.

The screenshots in the slide show the different types of default templates available when you create a new promotion by type. When creating an item discount, you have options such as BOGO, Buy Item X Get Item Y, Spend Y in X, Get Gift With Purchase (GWP) and variations on buying X or spending Y to get a GWP. There is also a tiered price break. Order and shipping discount templates have similar options, such as Get Order/Shipping Discount, Spend Y, and Spend Y in X. Both item and order discount options include templates for tiered pricing. All three types contain a template for buying within a specified time period. When creating a new promotion, you can always start with the advanced condition and offer template, which provides more flexibility, but is a little more complicated for business users to navigate.

Note: You can control the order and grouping of promotions in the BCC by type.

Example: BOGO Item Discount



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

The screenshot in the slide shows the condition and offer for a BOGO item discount. With the purchase of one Cowgirl Bag product, the customer gets a second one at 50% off. You also can set a template to exclude certain products, categories, or SKUs. This is not shown in the example. The example also shows the types of discounts available for the BOGO promotion, such as percent off, amount off, and free.

Example: Tiered Promotion

Offer

Offer Type: **Tiered** Standard, Tiered, Gift With Purchase

Item(s) to Discount: every (product named) Hubbard Chair)

Apply Discount To: Lowest Priced Item First Highest Priced Item First

Only applies when Item(s) to Discount begins with "Up to... (number)".

Discount Type: **% Off** Amount off, % off, Fixed Price, etc..

Discount Structure:

# to Buy	Value
2	10
4	15
6	20

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows a tiered offer. The first Hubbard Chair you buy would be at full price. The second and third chairs would have a 10% discount, the fourth and fifth chairs would have a 15% discount, and any additional chairs would be purchased with a 20% discount.

GWP Promotion: Example

The screenshot shows the Oracle ATG Commerce interface for managing promotions. At the top, it displays a promotional offer: "Free Scarf with all orders over \$200". To the right, it shows "Item Discount ID promo70015" and "Modified 5/24/2012 11:44 AM". Below this, there are tabs for "General", "Availability & Coupons", "Sites", and "Media". The "General" tab is selected. Under the "Template" section, it says "Spend Y Get GWP". The main area is titled "★ Condition and offer". It contains a "Condition" section where "Spend: 200" is specified. Below this, a note states: "Any related promotion upsells can be accessed via the "Promotion upsells" editor." The "Offer" section lists a "Gift with Purchase": "Cashmere Scarf" (SKU xsku60325). There are "Replace..." and "Clear" buttons for this offer. A note below the offer states: "Gifts are automatically added when a cart meets the specified condition. Gifts already in the cart are repriced as free. Items with a zero price cannot be used as a gift with purchase." At the bottom of the offer section, there are two radio buttons for "Auto Remove from Cart": one for "Yes, remove" (selected) and one for "No, keep in cart and reprice". A note at the very bottom says: "Only automatically added items can be auto removed."

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

GWP promotions were introduced in ATG 10.1. They allow you to determine a free gift or gifts that a customer receives when meeting the condition. GWP promotions have their own architecture, with a manager component, an item descriptor in the order repository, droplets, form handlers, and application messaging. CRS shows examples of how to implement GWP promotions. For more information, see the appendix titled “Advanced Promotions Topics” or the promotions viewlet in the My Oracle Support knowledge article titled “Oracle ATG Web Commerce 10.1 Release Training [ID 1490709.1].”

The example in slide shows one of the two GWP promotions in CRS. It uses the Spend Y Get GWP template. If you spend \$200, you receive the Cashmere Scarf (SKU xsku60325) for free. It is automatically added to the cart, because the customer does not need to select the item (with only one SKU available for free). You can choose what happens to the item in the cart if the customer removes items that mean the order no longer meets the condition (goes below 200). You can have ATG Commerce automatically remove the item from the cart or keep it in the cart and reprice the item.

Advanced Promotions

The screenshot shows the Oracle ATG Commerce Advanced Promotions configuration screen. It is divided into two main sections: 'The example condition' and 'The example offer'.

The example condition: This section contains a rule builder interface. The condition is defined as follows:

- Apply Offer: When order contains total quantity of at least 1 (product named Cowgirl Bag)
- No discounts on qualifier items

The example offer: This section contains the offer details:

- Offer Type: Standard (Standard, Tiered, Gift With Purchase)
- Item(s) to Discount: up to 1 total quantity of product in category name Home Accents
- Apply Discount To: Lowest Priced Item First
- Discount Type: % Off (Amount off, % off, Fixed Price, etc.)
- Discount Amount: 10 ex. 5

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If the templates do not offer what you need, you can create an advanced promotion. You can either start a promotion as advanced, or convert to the advanced template. Once you have converted an existing template to advanced, you cannot convert it back to the template. The conversion process, however, will convert your current condition and offer into the advanced template. The advanced template is more complicated. If you find that your business users regularly need to create a certain type of promotion that they can only create in the advanced template, you may want to consider creating your own custom promotion template.

In the example in the slide, you are seeing how an item discount looks in the advanced template. You use drop-down options to create rules for the condition and offer. The example uses the rule format to say that when the order contains a total quantity of at least one Cowgirl Bag product, the customer can get a 10% discount on one product from the Home Accents category, with the discount applying to the lowest priced item first.

Road Map

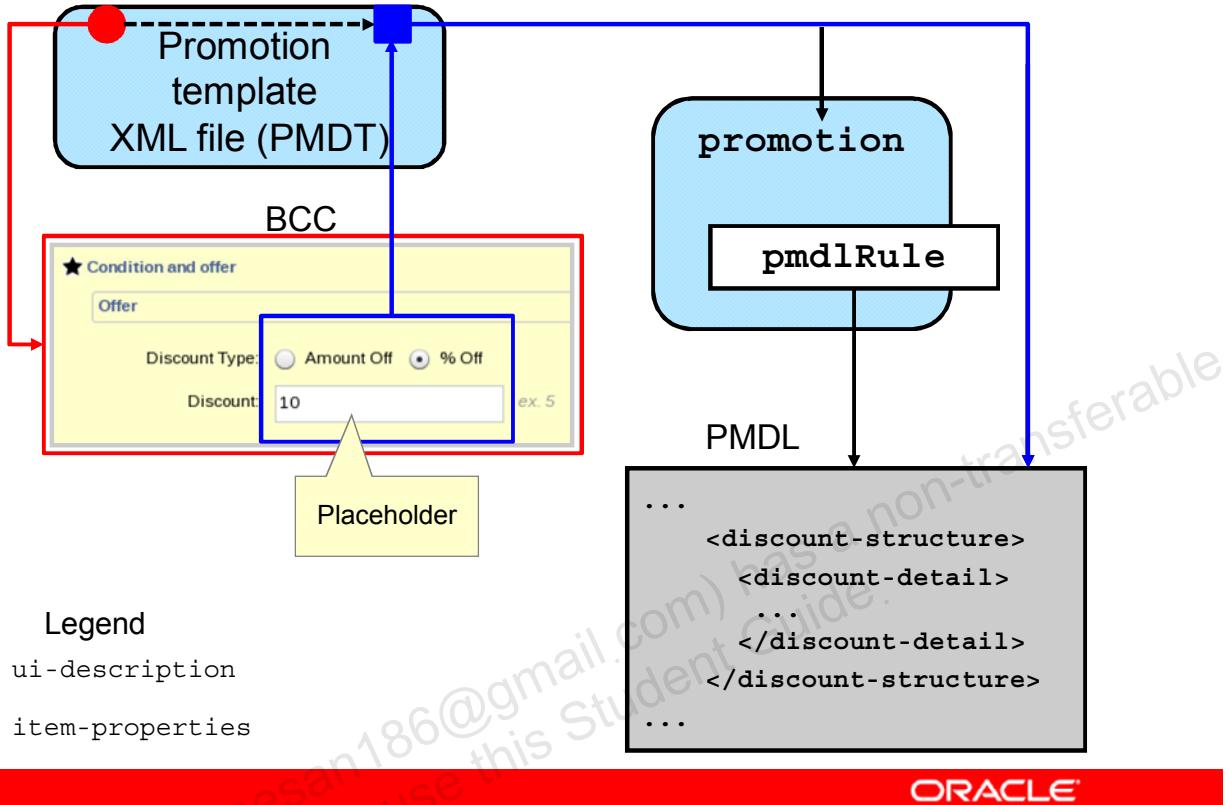
- Promotions templates in the BCC
- Creating custom promotion templates
- Defining the template UI
- Validating and translating user input
- Linking user input to promotion properties



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Promotion Template Relationships



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

The diagram in the slide is similar to the diagram in the slide “Promotion Relationships,” in the lesson titled “Extending Promotions,” but provides more details about promotion templates. When you select a template in the BCC, it reads the promotion’s template configuration from an XML file, called a PMDT file. PMDT stands for Pricing Model Description Template. The template has multiple elements in various tags. The ui-description tag creates larger field containing the description and offer in the BCC. The item-properties tag defines a property called pmdlRule that contains the actual PMDL with the defined placeholder names. When the business user creates a promotion in the BCC, the placeholders are filled in and the new PMDL rule is saved with the promotion repository item.

Promotion Template Definitions

Templates are:

- Defined in XML documents with the extension .pmdt
- Extendable through XML combination

/atg/registry/data/promotiontemplates/item/MyTemplate.pmdt

```
<template item-type="Item Discount">
  <ui-description>
    Specify UI elements to collect values
  </ui-description>
  <item-properties>
    Specify how the values collected in the UI are used in promotion properties
    (PMDL)
  </item-properties>
</template>
```

Valid item-types are
Item Discount
Order Discount
Shipping Discount

You must include a ui-description element for your template to appear in the BCC.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The default templates are found in `atgdir/config/config.jar` in the `/atg/registry/data/promotiontemplates` folder in the appropriate folder for the promotion type (advanced, item, order, or shipping).

You can use the default templates as a reference or a basis for making a new template.

The XML begins with the `<template>` tag. Valid attributes include `item-type`, `author`, `last-modified-by`, and `creation-date`. The other tags are described in more detail in subsequent slides.

Linking UI Elements to Promotion Properties

- UI elements are linked to promotion properties by the `placeholder-name` attribute.
- The `placeholder-name` may be specified in the UI element itself, or in a multielement translator.
 - Multielement translators are used to take input from UI fields and calculate values for the promotion properties.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Multielement translators are covered in more detail later in the lesson.

Road Map

- Promotions templates in the BCC
- Creating custom promotion templates
- **Defining the template UI**
- Validating and translating user input
- Linking user input to promotion properties



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Promotion Template Example: Get Item Discount

Offer

No. of Items to Discount: Unlimited

Apply Discount To: Lowest Priced Item First Highest Priced Item First

Item(s) to Discount:

Discount Type: Amount off, % off, Fixed Price, etc..

Discount: ex. 5

Items To Exclude (Optional):

Add Criteria Select...

Add Criteria Select...

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The next slides show the XML tags used to create the elements in the template titled “Get Item Discount (by Product, Category, Brand, etc.).” This template only has an offer, but no condition.

Template Name and Priority

The diagram illustrates the mapping between a user interface element and its corresponding XML configuration. A red box highlights the 'Get Item Discount (by Product, Category, Brand, etc.)' button in the UI. Another red box highlights the XML code for the same template, specifically the `<ui-description>` tag and its nested elements.

```
<ui-description>
    <display-name>
        <resource>template.item.getItemDiscount.title</resource>
    <resource>
        <bundle>atg.remote.promotion.template.Resources</bundle>
    </resource>
    <group-info groupId="1" templatePriorityId="1"/>
    <screen-segment display-name-
        <resource>template.common.segmentOfferTitle</resource>
    <line> ... </line>
    ...
</ui-description>
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `resource-bundle` attribute of the `ui-description` tag specifies the resource bundle used for all text in the promotion template's UI. If developers do not want to use a resource bundle, they can specify the display name of any element in the template by using the `display-name` attribute.

The `group-info` tag determines the grouping of the promotion and its priority within the group. In the example in the slide, the template appears in the first group as the first template.

Screen Segments and Lines

```
<ui-description ... >
  <group-info groupId="1" templatePriorityId="1" />
  <screen-segment display-name-
    resource="template.common.segmentOfferTitle">
    <line> ... </line>
  ...
</ui-description>
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The screen-segment tag defines a box of fields within the UI. The line tag defines a line of fields or text. The code example in the slide uses a resource bundle to provide the text for the screen segment. You can also use the display-name attribute to put the text directly in the PMDT file.

Defining a Line

No. of Items to Discount: Unlimited

```
<line>
  <label id="numberOfItemsToDiscountLabel" display-name-
    resource="template.common.numItemsToDiscountLabel"
    container-width="175" container-h-align="right"/>
  <textInput id="numberOfItemsToDiscount_textInput"
    disable-when=true="unlimited_checkbox" restrict="0-9"
    validator="/atg/remote/promotion/template/validators/
      PositiveNumberValidator"/>
  <checkBox id="unlimited_checkbox"
    placeholder-value-checked="-1" required="true"/>
  <label id="unlimited_label"
    display-name-resource="template.common.unlimitedLabel"/>
</line>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows how the line for number of items to discount is defined. The line consists of a label that is supplied through the resource bundle, a text input box, and a check box. The XML disables the text input amount when the check box is selected. The line concludes with the “Unlimited” text.

Many of the UI elements within a line take the following formatting attributes:

- **styleName:** The name of the CSS style
- **height:** The height of the element in pixels
- **width:** The width of the element in pixels
- **container-width:** The width of the container in pixels
- **container-height:** The height of the container in pixels
- **container-h-align:** The horizontal alignment within the container. The default value is left.
- **container-v-align:** The vertical alignment within the container, with a default value of top

There are a few attributes used in text input fields, some of which are shown in the example in the slide.

- The `disable-when-true` attribute is used to disable an input when another input is set to true. There is also an `enable-when-true` attribute available.
- The `restrict` attribute is used with text fields to restrict the characters allowed in that field. The `text` attribute can be used to suggest a default value.
- The `maxChars` attribute is used to limit the number of characters entered.
- The `validator` attribute specifies the component that will validate the field's contents.

The `placeholder-value-checked` attribute defined in the `checkbox` tag is used to specify the value provided to the corresponding promotion property when this box is selected. Developers may also specify `placeholder-value-unchecked`.

Defining Radio Buttons

Apply Discount To: Lowest Priced Item First Highest Priced Item First

```
<line>
  <label ... />
  <radioButtonGroup id="applyDiscountToRBG"
    placeholder-name="sort_order"
    id-of-selected-button="lowestPricedItemFirstRB">
    <radioButton id="lowestPricedItemFirstRB" display-name-
      resource="template.common.lowestPricedItemFirstRadioButton"
      placeholder-value="ascending"/>
    <radioButton id="highestPricedItemFirstRB" display-name-
      resource="template.common.highestPricedItemFirstRadioButton"
      placeholder-value="descending"/>
  </radioButtonGroup>
</line>
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Radio buttons are defined within the `radioButtonGroup` tag. The `id-of-selected-button` attribute specifies the default selection. This value matches the `id` value given in the `radio-button` tag.

The `placeholder-value` attribute, similar to `placeholder-value-checked` for check boxes, is used to supply a value to use in the `promotion` property when this radio button is selected.

Including and Excluding Product Set Criteria

```
<includesProductSetCriteria id="PSC"
    model-path="/atg/remote/promotion/expreditor/psc/
        PSCExpressionModel"
    event-handler-path="/atg/remote/promotion/expreditor/psc/
        PSCEventHandler"/>
```



```
<excludesProductSetCriteria id="exPSC" required="false"
    model-path="/atg/remote/promotion/expreditor/psc/
        PSCExpressionModel"
    event-handler-path="/atg/remote/promotion/expreditor/psc/
        PSCEventHandler"/>
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The tags to include and exclude product set criteria are similar. You define the `model-path` and `event-handler-path` attributes that point to components. The `model-path` attribute points to the component that provides this set of UI fields and the choices available. Both code examples in the slide point to the same `PSCExpressionModel` component. The `required` attribute can be used with any input field. The default value is `true`.

When using the `includesProductSetCriteria` or `excludesProductSetCriteria` tag, developers must also include the matching multielement translator in the template:

```
<multi-element-translator id="PSC_Translator" translator-
path="/atg/remote/promotion/expreditor/psc/PSCExpressionTranslator">

    <placeholder-info placeholder-name="PSC_value"
        translator-output-name="criteriaPMDL"/>

    <element-info element-id="PSC"
        translator-input-name="includesPSC"/>

    <element-info element-id="exPSC"
        translator-input-name="excludesPSC"/>

</multi-element-translator>
```

Defining Combo Boxes

```
<comboBox id="discount_type_combobox">
    <content-source
        path="/atg/remote/promotion/template/contentSource/
            DiscountList"/>
</comboBox>
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `content-source` tag is used to point to the component that provides the list of values for the combo box. The tag can contain any number of attribute tags that can pass values to the component, which may be hard-coded or come from other fields in the template.

The `content-source` tag can also be used to calculate default values for text inputs.

If you need to define your own custom content source component, the class used to create the component must implement

`atg.remote.promotion.template.contentSource.ContentSource`.

The component returns `atg.remote.promotion.template.model.ContentState` objects. The `data` property of the `ContentState` object contains either

- A list of Map objects for combo box selections or
- A String for the default value of the text field

The method signatures of the `ContentSource` class are

```
public ContentState getContent (String pElementID, String
pElementType, String pPromoType) throws TemplateException
```

```
public ContentState getContentWithObjectParam (String pElementID,
String pElementType, String pPromoType, Object pAdditionalParams)
throws TemplateException
```

Additional Template UI Elements

- `horizontalRule`: A horizontal line
- `spacer`: A space with a defined height
- `textArea`: A larger field than `textInput`
- `expression`: For complex sets of fields and options
 - Points to a model component
- `date`: A date selector
- `assetCollector`: An asset picker for repository items
 - Configured with `repository-path` and `item-type` attributes



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If the `assetCollector` tag is used, developers must include `AssetCollectorTranslator` in the multi-element-translators section.

Road Map

- Promotions templates in the BCC
- Creating custom promotion templates
- Defining the template UI
- Validating and translating user input
- Linking user input to promotion properties



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Simple Validation with Tag Attributes

- **restrict:** Restricts the characters allowed in the field
- **maxChars:** Limits the number of characters
- **required:** Controls if the field is required (the default is true)
- **coupled-with:** Makes a field required if another field has input



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Here is a simplified example of using `coupled-with`:

```
<textInput id="spend_close_textInput"
           coupled-with="cq_name_label_textInput"
           placeholder-name="spend_close_value" />

<textInput id="cq_name_label_textInput"
           coupled-with="spend_close_textInput"
           placeholder-name="cq_name"/>
```

In this example, if the user enters anything into either field, the other field becomes required. In other words, the user must fill in both fields or leave both fields blank. The fields are linked by ID.

Complex Validation

Validator components perform more complex validation:

```
<textInput id="numberOfItemsToDiscount_textInput"
    disable-when=true="unlimited_checkbox"
    restrict="0-9"
    validator="/atg/remote/promotion/template/
        validators/NonZeroDecimalValidator"/>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Multielement Translators

- Multielement translators are used to calculate values from multiple fields.
- Example: The number of items to discount might be a fixed number entered in text field, or “unlimited” (-1), indicated by selecting the check box.

No. of Items to Discount: Unlimited

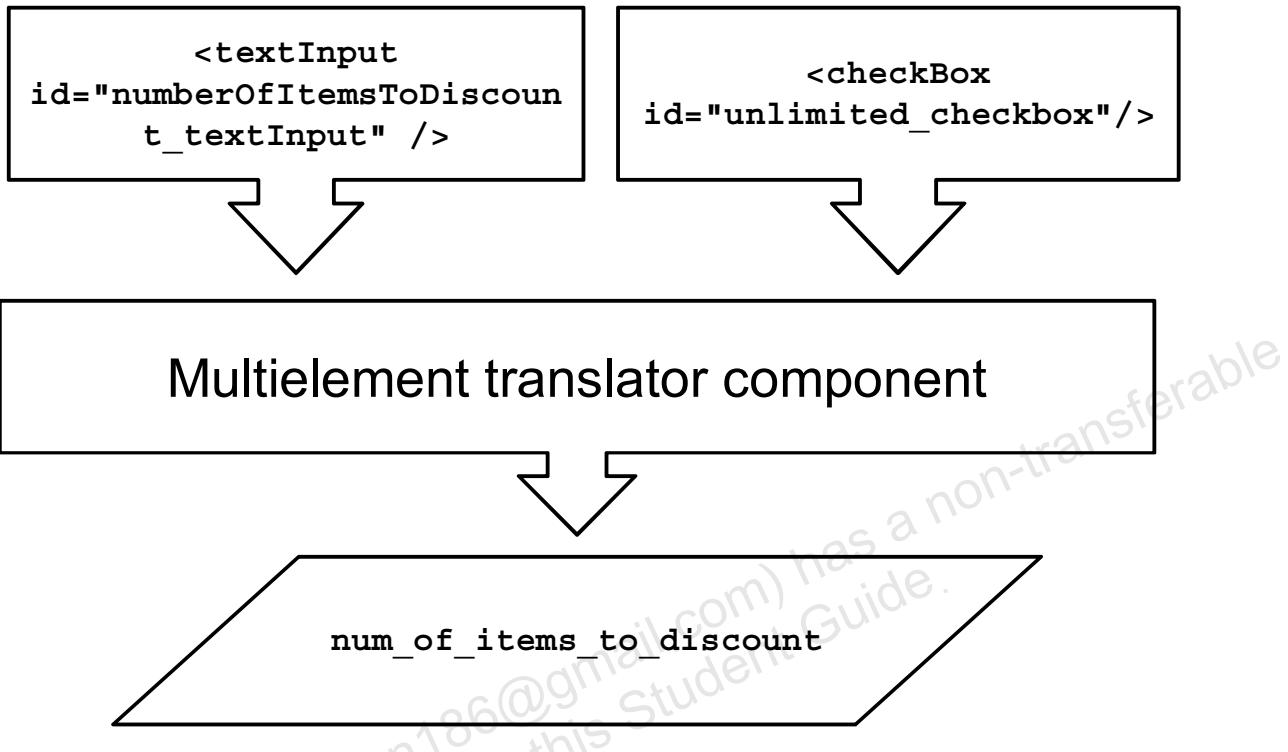
```
<textInput id="numberOfItemsToDiscount_textInput"
    disable-when=true="unlimited_checkbox" restrict="0-9"
    validator="/atg/remote/promotion/template/validators/
        PositiveNumberValidator"/>
<checkBox id="unlimited_checkbox"
    placeholder-value-checked="-1" required="true"/>
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The code example shown in the slide is a review from the slide titled “Screen Segments and Lines.” A validator validates a single field. A multielement translator calculates values from multiple fields. A subsequent slide shows how multielement translators appear in the PMDT file.

Multielement Translator: Example



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The translator component also works in reverse. When a persisted promotion is loaded in the UI, the output value is translated back to the values to display in the fields.

multi-element-translators Tag

```
<ui-description>
    <screen-segment>...</screen-segment>

    <multi-element-translators>
        ...
    </multi-element-translators>
</ui-description>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Multielement translator elements are defined within the `multi-element-translators` tag, which is within the `ui-description` tag, separate from any `screen-segment` tags.

Multielement Translator: Example

```
<multi-element-translator  
    id="unlimited_discount_translator"  
    translator-path="/atg/remote/promotion/template/  
        translators/UnlimitedDiscountTranslator">  
  
<placeholder-info  
    placeholder-name="no_of_items_to_discount"  
    translator-output-name="itemNumber"/>  
  
<element-info  
    element-id="numberOfItemsToDiscount_textInput"  
    translator-input-name="itemNumber"/>  
  
<element-info element-id="unlimited_checkbox"  
    translator-input-name="unlimited"/>  
</multi-element-translator>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The next few slides discuss different parts of the code example in the slide. The `multi-element-translator` tag includes the `translator-path` attribute, which defines the component that will do the translation.

Multielement Translator: Example

```
<multi-element-translator  
    id="unlimited_discount_translator"  
    translator-path="/atg/remote/promotion/template/  
        translators/UnlimitedDiscountTranslator">  
  
    <placeholder-info  
        placeholder-name="no_of_items_to_discount"  
        translator-output-name="itemNumber"/>  
  
    <element-info  
        element-id="numberOfItemsToDiscount_textInput"  
        translator-input-name="itemNumber"/>  
  
    <element-info element-id="unlimited_checkbox"  
        translator-input-name="unlimited"/>  
/<multi-element-translator>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `placeholder-info` tag defines `placeholder-name` that will be used to specify this value in a promotion property. `translator-output-name` matches the key that the translator component uses to refer to this output value in the map it produces.

Multielement Translator: Example

```
<multi-element-translator  
    id="unlimited_discount_translator"  
    translator-path="/atg/remote/promotion/template/  
        translators/UnlimitedDiscountTranslator">  
  
<placeholder-info  
    placeholder-name="no_of_items_to_discount"  
    translator-output-name="itemNumber"/>  
  
<element-info  
    element-id="numberOfItemsToDiscount_textInput"  
    translator-input-name="itemNumber"/>  
  
<element-info element-id="unlimited_checkbox"  
    translator-input-name="unlimited"/>  
  
</multi-element-translator>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `element-info` tags define the input for the translator component. The `element-id` attribute value specifies the ID of the input field. The `translator-input-name` attribute value is used as the key in the map of input values that gets passed to the translator component. For the example in the slide, the value for `textInput` is either null (if it is unlimited) or a number. The value for `unlimited` is either null (if unchecked) or -1.

PSC Multielement Translator

If `includesProductSetCriteria` or `excludesProductSetCriteria` are used in the template, developers must include the PSC translator.

```
<multi-element-translator id="PSC_Translator"
    translator-path="/atg/remote/promotion/expreditor/psc/
        PSCEXpressionTranslator">
    <placeholder-info placeholder-name="PSC_value"
        translator-output-name="criteriaPMDL"/>
    <element-info element-id="PSC"
        translator-input-name="includesPSC"/>
    <element-info element-id="exPSC"
        translator-input-name="excludesPSC"/>
</multi-element-translator>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Product set criteria refer to the ability to include or exclude one or more products, categories, or SKUs in the promotion.

Default Translator Components

ATG Commerce provides some default translator components, including

- AssetCollectorTranslator: Used with the assetCollector UI element
- DiscountStructureTranslator: Used with volume discounts
- DiscountTranslator: Used with the combo box
- TargetTranslator
- UnlimitedDiscountTranslator



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The components are in the /atg/remote/promotion/template/translators namespace. Their classes have the same name, and are in the atg.remote.promotion.template.translators package.

Writing a Translator Component Class

- Translator classes implement the `atg.remote.promotion.template.translators.MultiElementTranslator` interface.
- This interface requires two methods:
 - `translateInputValues()`
 - `reverseTranslateOutputValues()`



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `translateInputValues()` method takes the input values defined in the template and produces the required output value. The `reverseTranslateOutputValues()` method is used when an existing promotion is displayed in the UI for editing. The output value persisted in the relevant promotion property is translated back to the input values to populate the UI fields.

For more details on this topic, see the *API Platform Reference*.

Road Map

- Promotions templates in the BCC
- Creating custom promotion templates
- Defining the template UI
- Validating and translating user input
- Linking user input to promotion properties



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Setting Promotion Properties

The <item-properties> section of the template defines how user input is used to set the promotion repository item's properties.

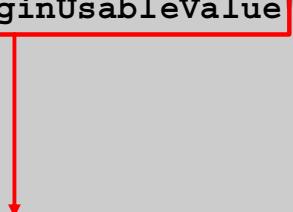
```
<item-properties>
  <property name="repository-item property name">
    hard-coded values & values from user input
  </property>
</item-properties>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Setting Simple Promotion Properties

```
<ui-description>
    <date id="start_date"
        placeholdername='beginUsableValue' />
</ui-description>
...
<item-properties>
    <property name="beginUsable">
        ${beginUsableValue}
    </property>
</item-properties>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide is from the Buy Within Specified Time Frame Get Item Discount template.

Setting the pmdlRule Property

- The most critical property of a promotion is its PMDL rule.
 - PMDL can be quite complex.
- Suggestion: Create the desired promotion type in Merchandising by using the advanced template, and then browse to the project's assets through the CA interface and view the new promotion to see the constructed PMDL.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

To view the PMDL through the BCC:

1. Go to the BCC home screen.
2. In the **To Do List** drop-down, select **Projects I Created**.
3. Find the project with the example promotion and click its **Go To Project** link.
4. On the project screen, select the **Assets** tab.
5. Click the name of your example promotion to view its properties.
6. The Text field shows the PMDL.

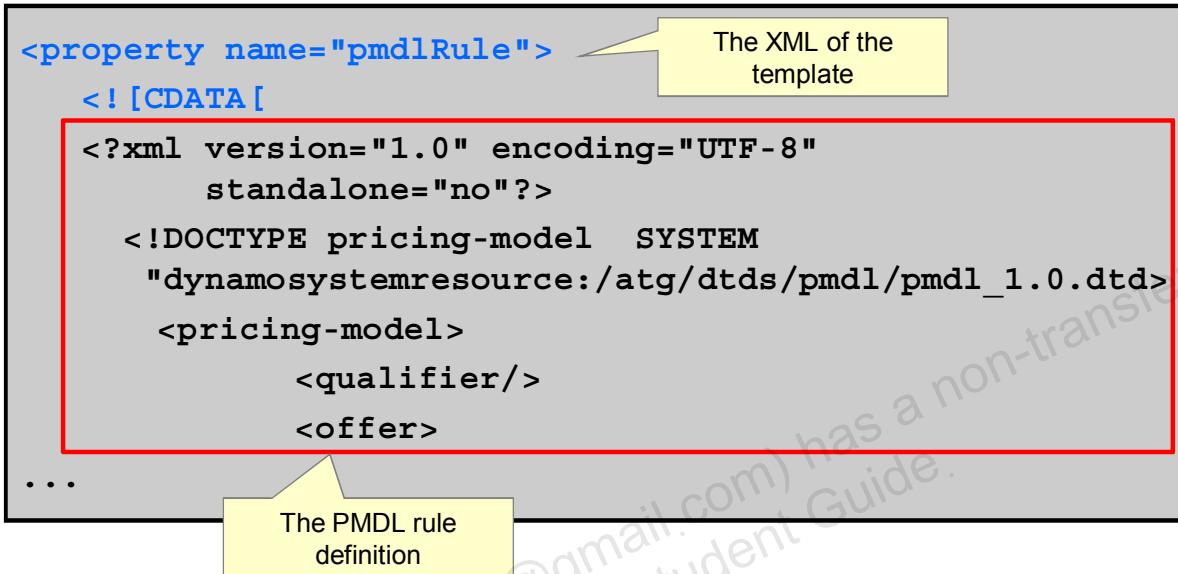
Setting the pmdlRule Property Example: Part One

```
<property name="pmdlRule">
  <![CDATA[
    <?xml version="1.0" encoding="UTF-8"
      standalone="no"?>
    <!DOCTYPE pricing-model SYSTEM
      "dynamosystemresource:/atg/dtds/pmdl/pmdl_1.0.dtd">
    <pricing-model>
      <qualifier/>
      <offer>
        ...
  ]]>

```

The XML of the template

The PMDL rule definition



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The text in blue in the code example in the slide is the XML of the template. The text in black is the PMDL. The code example continues in the next slide.

Setting the pmdlRule Property Example: Part Two

```
<discount-structure
    calculator-type="standard"
    discount-type="${discount_type_value}"
    adjuster="${discount_value}">
<target>
    <iterator name="up-to-and-including"
        number="${no_of_items_to_discount}"
        sort-by="priceInfo.listPrice"
        sort-order="${sort_order}">
        <collection-name>items</collection-name>
        <element-name>item</element-name>
        <element-quantity-property> quantity
        </element-quantity-property>
        ${PSC_value}
    </iterator>
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `discount_type_value` and `discount_value` variables are calculated by the Discount Translator discussed earlier:

```
<multi-element-translator id="discount_translator" translator-path=
"/atg/remote/promotion/template/translators/DiscountTranslator">
    <placeholder-info placeholder-name="discount_value"
        translator-output-name="adjuster"/>
    <placeholder-info
        placeholder-name="discount_type_value"
        translator-output-name="discount-type"/>
    <element-info element-id="discount_type_combobox"
        translator-input-name="discountType"/>
    <element-info element-id="discount_textInput"
        translator-input-name="adjuster"/>
</multi-element-translator>
```

The `no_of_items_to_discount` variable (in the `iterator`) tag is calculated by the Unlimited Discount Translator discussed earlier:

```
<multi-element-translator id="unlimited_discount_translator"
    translator-
    path="/atg/remote/promotion/template/translators/UnlimitedDiscountTranslator">
    <placeholder-info
        placeholder-name="no_of_items_to_discount"
        translator-output-name="itemNumber"/>
    ...
</multi-element-translator>
```

The `sort_order` variable comes from the radio button group discussed earlier:

```
<radioButtonGroup id="applyDiscountToRBG" placeholder-name="sort_order">
```

The `PSC_value` variable is calculated by the PSC Expression Translator discussed earlier:

```
<multi-element-translator id="PSC_Translator" translator-
    path="/atg/remote/promotion/expeditor/psc/PSCEXpressionTranslator">
    <placeholder-info placeholder-name="PSC_value"
        translator-output-name="criteriaPMDL"/>
    <element-info element-id="PSC" translator-input-
        name="includesPSC"/>
    <element-info element-id="exPSC" translator-input-
        name="excludesPSC"/>
</multi-element-translator>
```

Setting the pmdlRule Property Example: Part Three

```
        </target>
    </discount-structure>
</offer>
</pricing-model>
]]>
</property>
```

The closing XML of the template



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The text in blue is the XML of the template. The text in black is the PMDL XML.

Including Closeness Qualifiers

Condition

Spend:

Promotion Upsell (Optional)

Spend: ex. 100

Message: ex. Spend 100 dollars to receive an item discount

You can use promotion upsells in your site and customer communications to inform customers when they are close to qualifying for a promotion.

Offer

No. of Items to Discount: Unlimited

Templates can also include fields to create closeness qualifiers.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Example: Including Closeness Qualifiers

```
<property name="closenessQualifiers">
    <item-properties>
        <property name="pmdlRule">
            ...PMDL of qualifier item...
        </property>
        <property name="name">
            ${cq_name}
        </property>
    </item-properties>
</property>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The code example in the slide shows the section of the XML that sets the properties of the closeness qualifier repository item, which is itself a property of the promotion.

The PMDL excerpted from the slide is shown in the following:

```
<! [CDATA [
    <?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <!DOCTYPE pricing-model SYSTEM
        "dynamicsystemresource:/atg/dtds/pmdl/pmdl_1.0.dtd">
    <pricing-model>
        <qualifier>
            <comparator name="greater-than-or-equals">
                <value>order.priceInfo.amount</value>
                <constant>
                    <data-type>java.lang.Double</data-type>
                    <string-value>
                        ${spend_close_value}
                    </string-value>
                </constant>
            </comparator>
        </qualifier>
    </pricing-model>
]]>
```

Changing or Deleting Templates

- Best practice: Do *not* delete or edit template definitions once they have been used.
 - Instead, create a new template to replace the old one.
 - Once the new template is ready, set the old template's available-in-ui attribute to false.

```
<ui-description available-in-ui="false" ... />
```

- Promotions using a deleted template are displayed with the advanced promotion screen.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe default promotion templates
- Create new promotion templates

For More Information

ATG Commerce Programming Guide

- Understanding Promotions
- Adding New Promotions Templates



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Practice 14 Overview: Creating a Custom Promotion Template

This practice covers creating a promotion template for points-multiplier promotions.



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable
license to use this Student Guide.

15

Shipping and Shipping Groups

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the process for collecting shipping information
- Add custom shipping groups

Road Map

- Collecting shipping information
- Shipping form handlers
- Shipping groups
- Adding a custom shipping group type



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Shipping Examples

To help illustrate the architecture and objects involved in handling shipping choices in the ATG Commerce JSPs, the lesson uses three different examples:

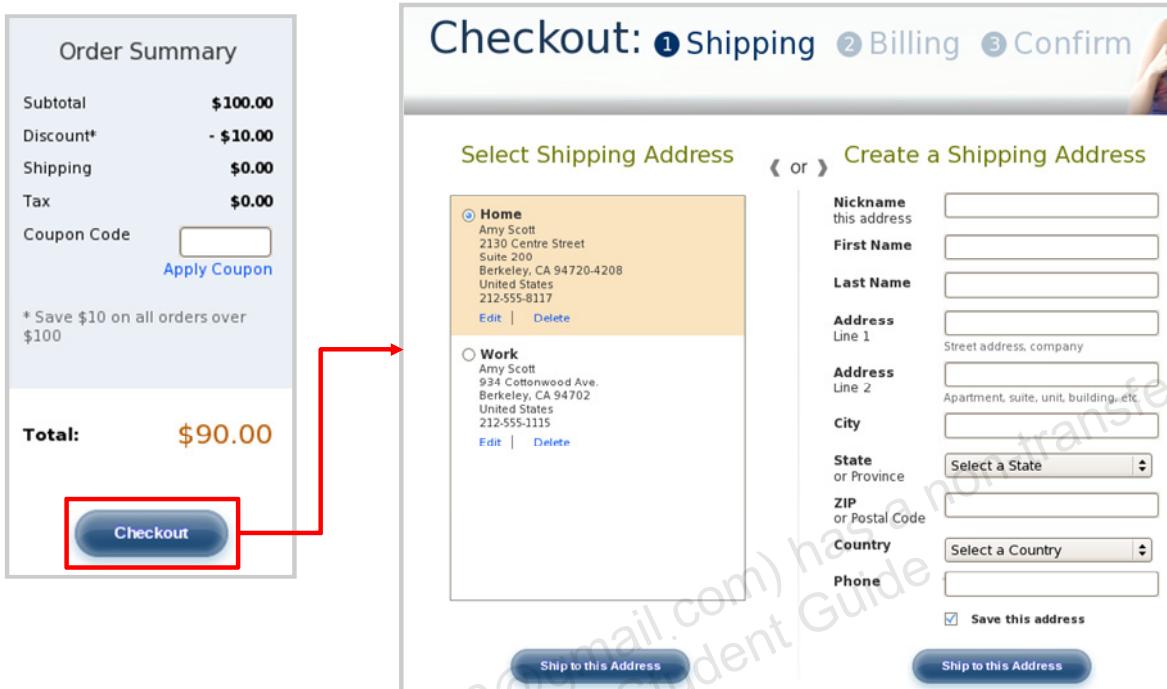
- Example A: The shopper opens the shipping page from the cart page for the first time on that order
- Example B: The shopper returns to the shipping page from later in the checkout process
- Example C: The shopper opts to ship an item to multiple addresses



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The examples are spread throughout the lesson, in conjunction with the objects that affect them.

Example A: The Shopper Opens the Shipping Page from the Cart Page on a New Order



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

The following slides follow the example of a shopper who is purchasing items and is going through the checkout process for the first time on the order. The example in the slide assumes that the shopper is logged in. The first screenshot in the slide shows the checkout button from the shopping cart page. When you click that in CRS, you go to the shipping page. The second screenshot shows the home and work addresses from the profile of a registered user named Amy Scott. You can select an existing shipping address, create a new shipping address, and optionally, split the shipping across multiple addresses. In the example, the shopper is going to the shipping page for the first time on this order, and has no address detail selected yet.

ShippingGroupDroplet (SGD) Is Triggered



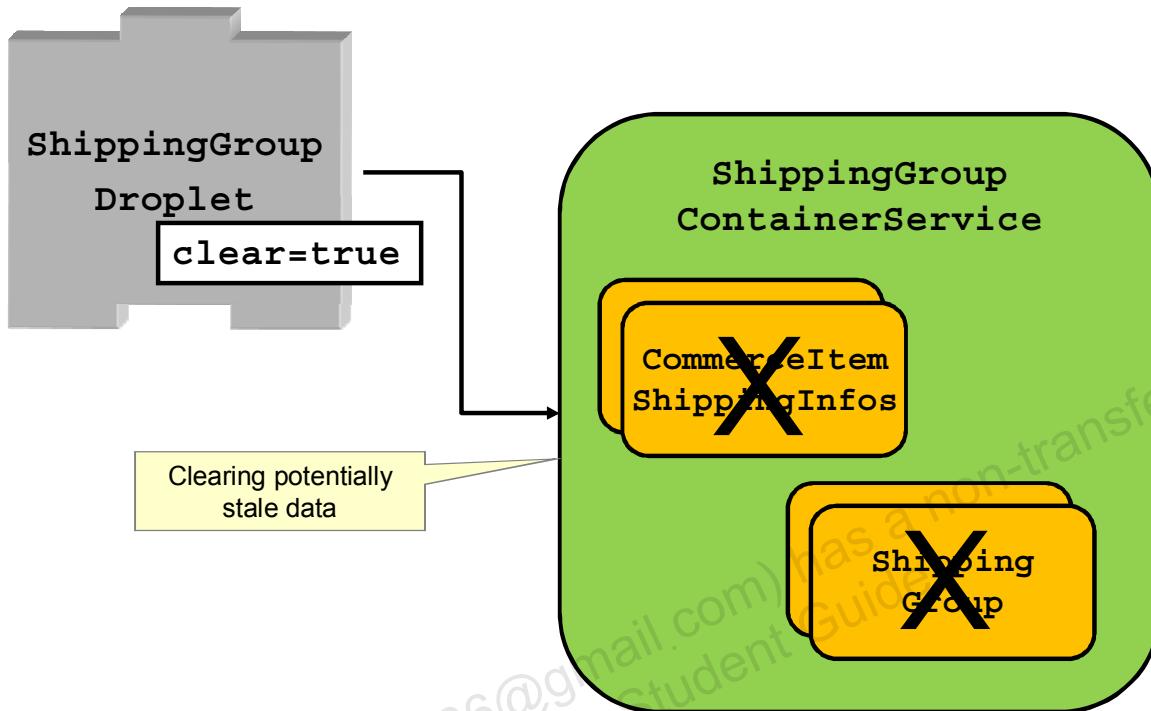
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

On the shipping page, the `/atg/commerce/order/purchase/ShippingGroupDroplet` is triggered. This droplet initializes some containers and values. When you first go to the shipping page, the droplet clears any previous information, initializes some objects (`ShippingGroups` and `ShippingInfos`), and sets the shipping group types according to the types of items in the cart. In the example in the slide, the cart contains “hard goods,” items that are physically shipped to the shopper. The shipping group type that handles that, by default, is the `HardgoodShippingGroup`.

The SGD properties with their settings as shown in the slide do the following:

- `clear=true`: Clears both the user's `CommerceItemShippingInfoContainer` and `ShippingGroupMapContainer`
- `initShippingGroups=true`: Loads the `ShippingGroupMapContainer` with the authorized shipping groups for the current user (from the profile) and the default shipping group
- `initShippingInfos=true`: Creates a `CommerceItemShippingInfo` object for each commerce item and initializes it with the:
 - Quantity of that item in the order
 - Name of the default shipping group

SGD Clears ShippingGroupContainerService (SGCS)



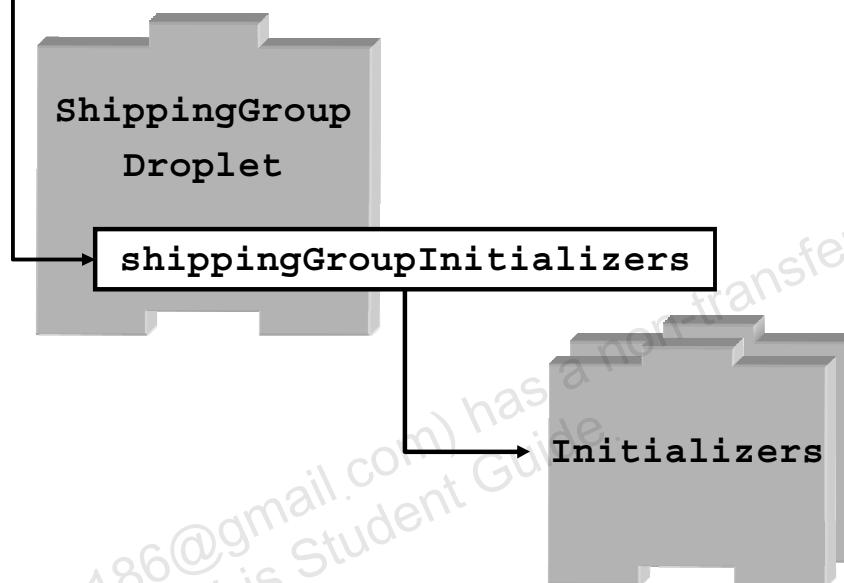
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The ShippingGroupContainerService is a session-scoped component that contains two maps: CommerceItemShippingInfoContainer and ShippingGroupMapContainer. Those maps contain the CISIs and shipping groups. If the shopper submits two orders in the same session, or if the order has been altered since the shopper first loaded the shipping page, it may contain stale data. The data should be cleared when a shopper transitions from the cart page to the shipping page.

SGD Calls Initializer Components to Determine the Valid Shipping Groups

```
shippingGroupTypes= HardgoodShippingGroup
```

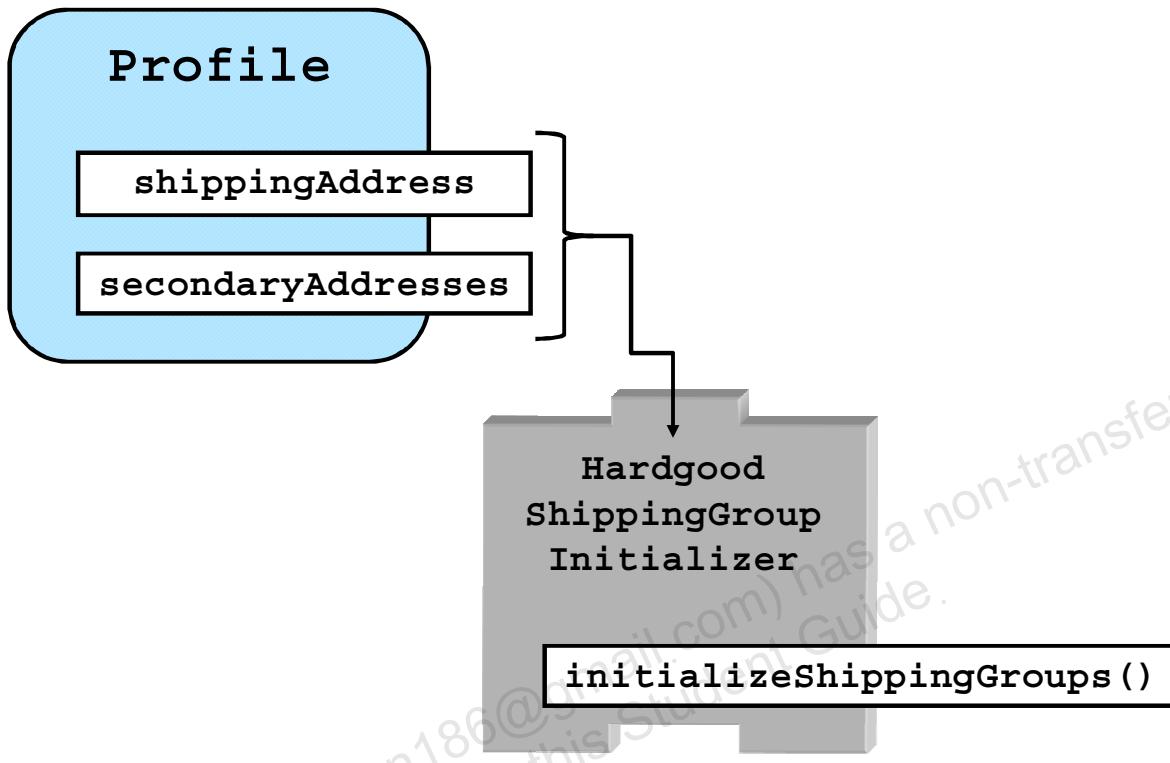


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `ShippingGroupDroplet` has a map of initializer components in its `shippingGroupInitializers` property. For each type of shipping group it has been told to initialize, it calls the `initializeShippingGroups()` method of the associated initializer component. The string given in `shippingGroupTypes` matches the key in the `shippingGroupInitializers` map.

HardgoodShippingGroupInitializer

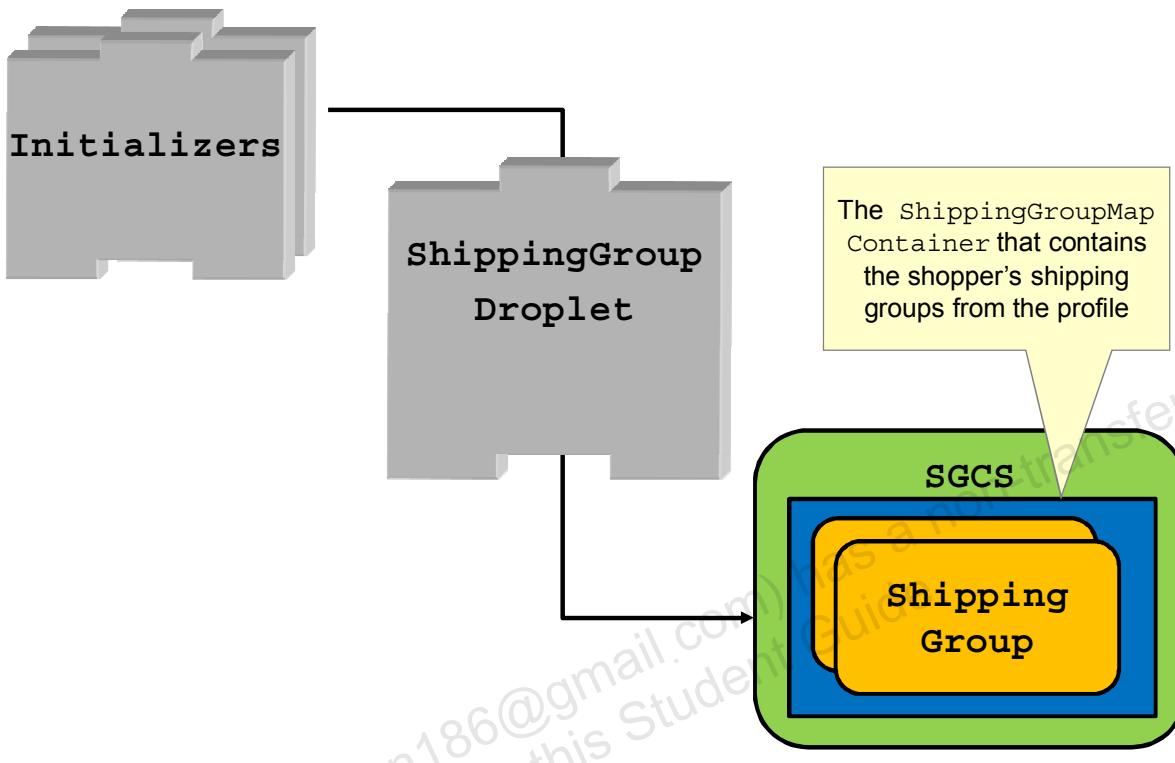


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

To initialize hardgood shipping groups, ShippingGroupDroplet calls the `initializeShippingGroups()` method of `HardgoodShippingGroupInitializer`, which creates shipping groups from physical addresses stored on the profile. The `secondaryAddresses` property is a map of `contactInfo` items.

SGD Populates Shipping Groups



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Based on the results from the initializers, the `ShippingGroupDroplet` populates the valid shipping groups within the `ShippingGroupMapContainer`. The `ShippingGroupMapContainer` holds the authorized shipping groups for the current user (from the profile) and the default shipping group.

For the sake of simplicity, the container is mentioned in the slide, but does not appear in every diagram.

Default Initializer Components

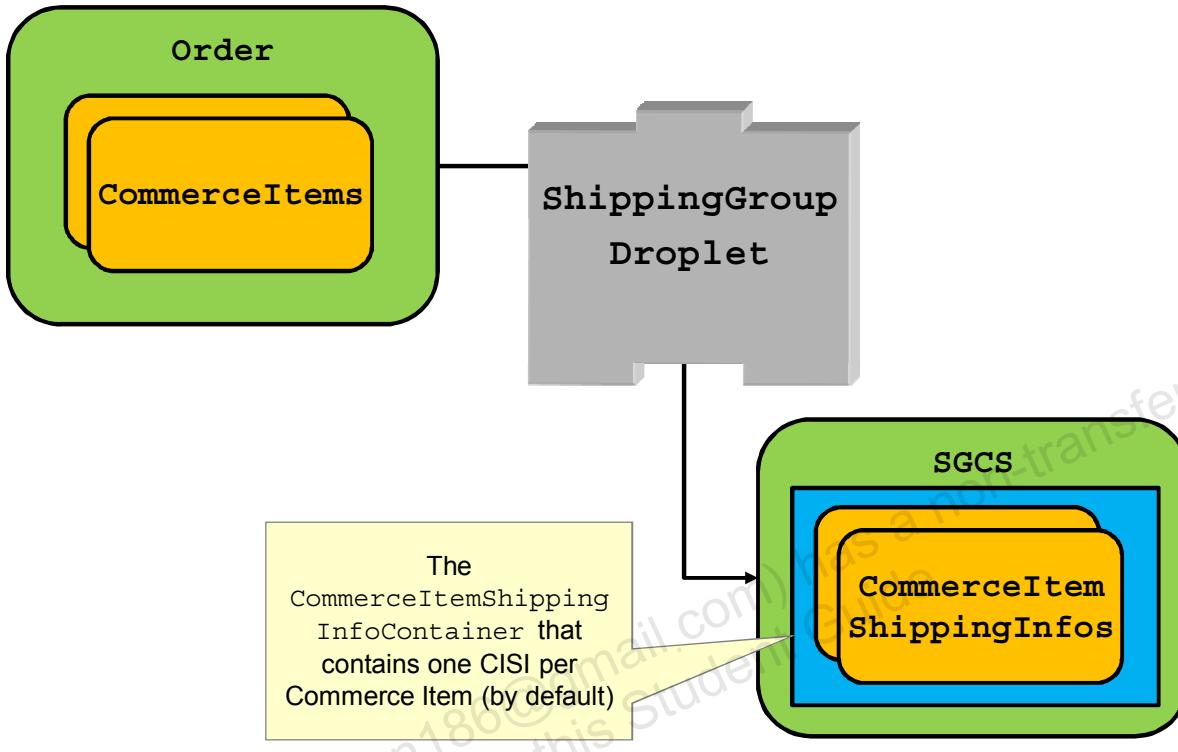
- Default initializer classes and components:
 - HardgoodShippingGroupInitializer
 - ElectronicShippingGroupInitializer
 - StoreHardgoodShippingGroupInitializer
 - A CRS extension that adds the ability to initialize shipping groups from gift list addresses
- The components are also used to compare shipping groups for equality (Are two groups the same address?).



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The default initializer components are in the /atg/commerce/order/purchase namespace. StoreHardgoodShippingGroupInitializer is the class that is configured in the HardgoodShippingGroupInitializer component in CRS.

SGD Populates CommerceItemShippingInfos (CISIs)



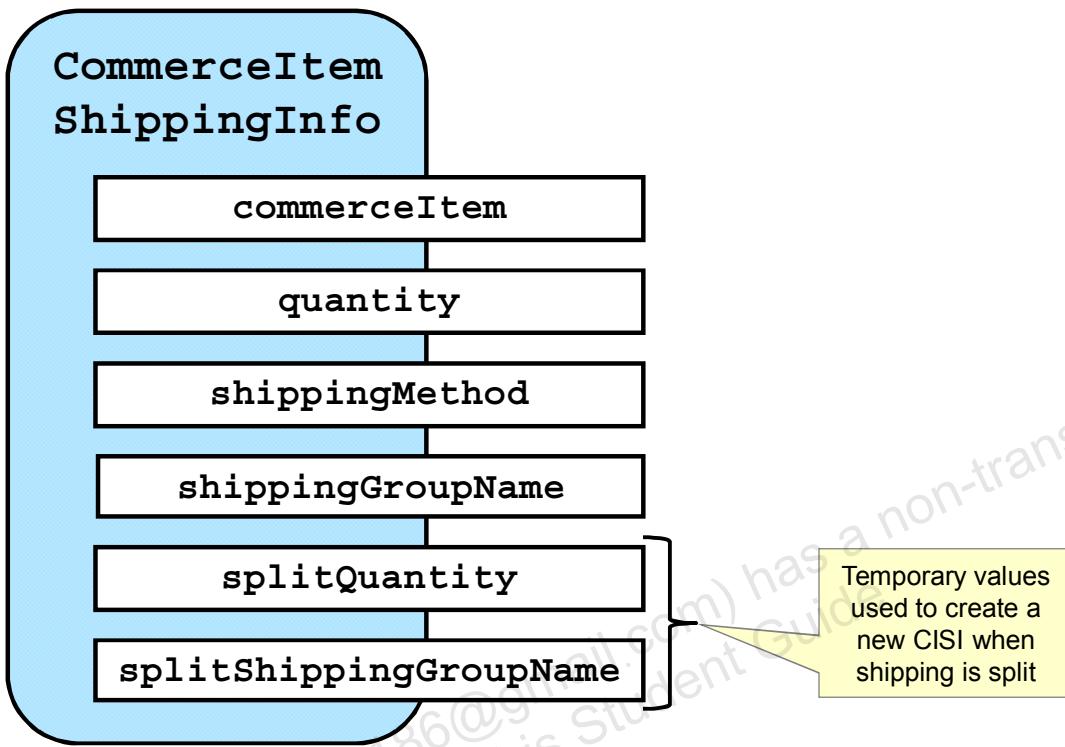
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

CommerceItemShippingInfos are temporary objects used to model the relationship between commerce items and shipping groups. When the **ShippingGroupDroplet** is first invoked on a new order, it loads two containers: **ShippingGroupMapContainer** and **CommerceItemShippingInfoContainer**. The **CommerceItemShippingInfoContainer** holds a **CommerceItemShippingInfo** object for each commerce item and initializes it with the quantity of that item in the order and the name of the default shipping group.

For the sake of simplicity, the container is mentioned in the slide, but does not appear in every diagram.

CommerceItemShippingInfo

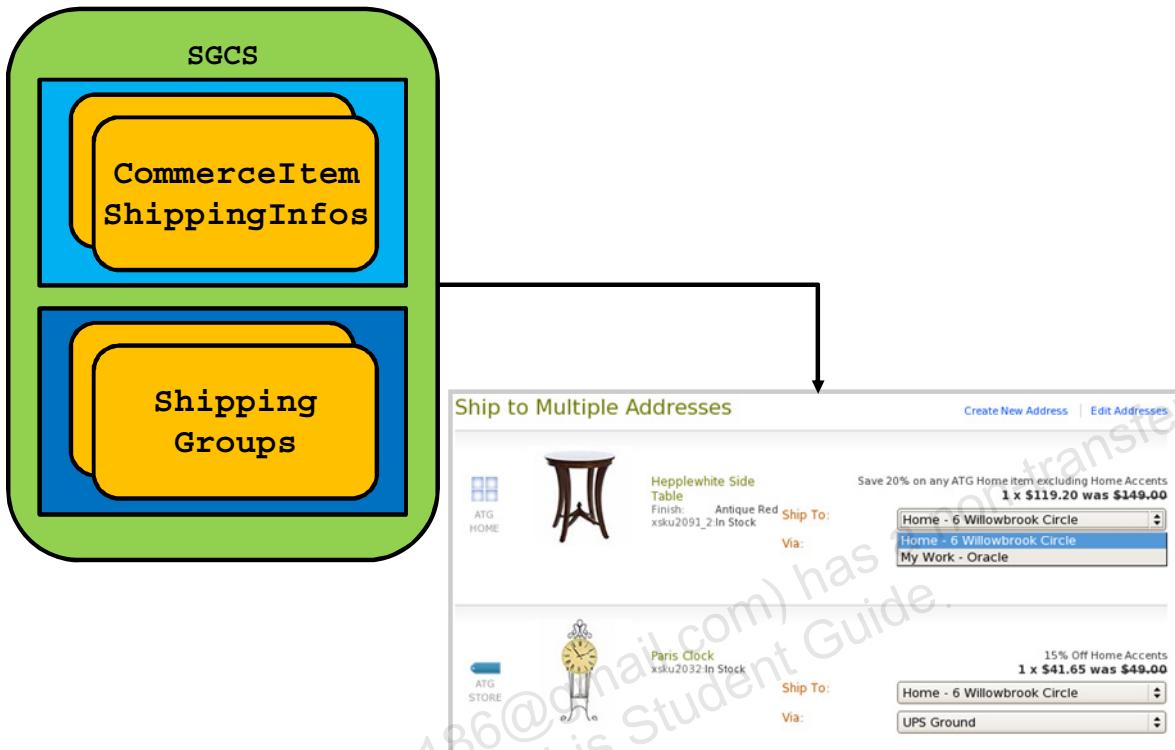


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The CISI holds the commerce item and quantity with the shipping method and shipping group name. When you ship a commerce item to multiple addresses, the CISI holds the information during the process of splitting the commerce item across shipping groups.

SGCS Data Is Used to Populate the Shipping Page

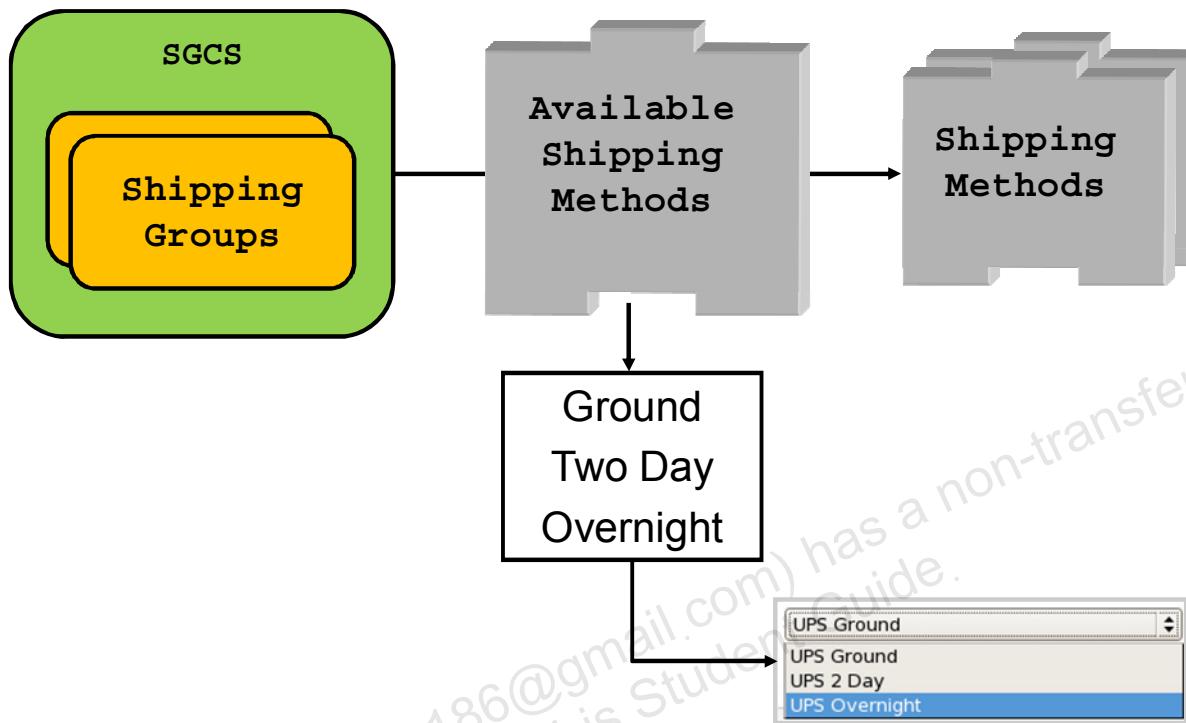


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The information from the two maps, `ShippingGroupMapContainer` and `CommerceItemShippingInfoContainer`, is used to populate the shipping page with the commerce items, the addresses from the profile, and the available shipping methods by commerce item. The screenshot in the slide shows the CRS page options when you select to send the items to multiple addresses. The page shows the commerce items, the available addresses in a drop-down menu, and the available shipping methods in a drop-down menu. You will learn more about how the available shipping methods are retrieved by the JSP in the next slides.

Note: Each of these commerce items is its own form. You will learn more about the underlying form handler later in the lesson.

Determining the Available Shipping Methods in the Shipping JSP



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

To determine which shipping methods are available, the JSP calls the `/atg/commerce/pricing/AvailableShippingMethods` droplet. The next slide discusses the droplet in more detail. In brief, the droplet returns an array of strings gathered from the shipping calculators' `shippingMethod` property, and that array can be used to generate the HTML to display the options.

AvailableShippingMethods Droplet

- The AvailableShippingMethods droplet:
 - Determines which shipping methods are available for a particular shipping group
 - Returns an array of strings gathered from the shipping calculators' shippingMethod property
 - Is used to display available shipping methods for shoppers to choose from for each shipping group
- Parameters include:
 - Input parameters
 - shippingGroup (required)
 - locale (optional; used to override the default locale)
 - Output parameter: availableShippingMethods (string array)
 - Open parameter: output

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Shopper Selects Shipping Options

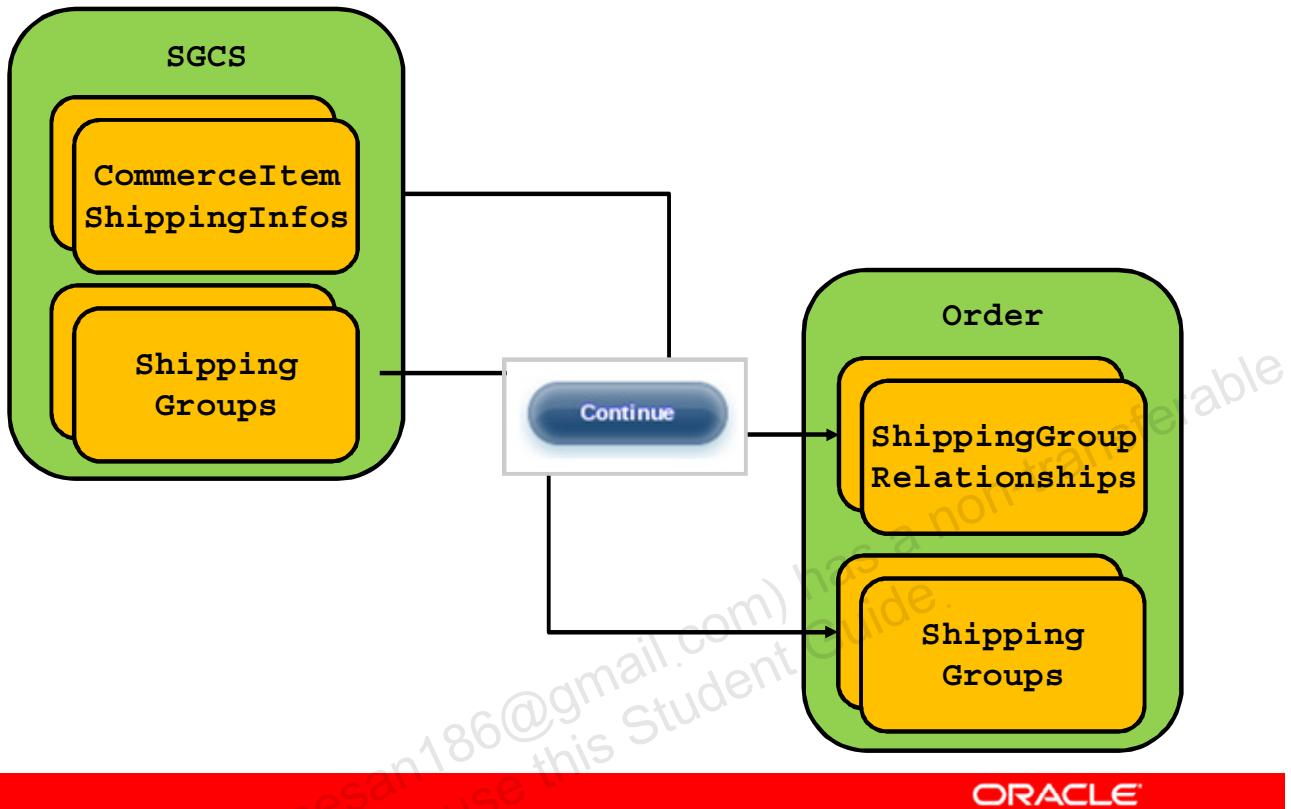


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

As the shopper splits shipping, selects shipping methods, and enters new shipping addresses, these changes are modeled in the `ShippingGroupContainerService`. The screenshots in the slide show the first and second pages of the simple shipping pages, where the shopper can select the address and then the shipping method for the entire order.

Shipping Options Are Changed in the Order



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

After shoppers have finalized their changes and clicked to move on to the billing page, their changes are made in the Order object. The CISIs are converted to ShippingGroupRelationships.

Example B: The Shopper Returns to the Shipping Page in an Order

Ship To: Alex Jackson
6 Willowbrook Circle
Boulder, CO 80302
United States
212-555-8860

Via: UPS Ground [Edit](#)

[Edit](#)

Select Shipping Address

Home (selected)
Alex Jackson
6 Willowbrook Circle
Boulder, CO 80302
United States
212-555-8860
[Edit](#) | [Delete](#)

My Work
Alex Jackson
Oracle
One Main Street
Cambridge, MA 02141
United States
617-555-5211
[Edit](#) | [Delete](#)

Create a Shipping Address

Nickname
this address

First Name

Last Name

Address
Line 1

Address
Line 2

City

State
or Province

ZIP
or Postal Code

Country

Phone

Save this address

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the shopper wants to change something in the shipping selections. When returning to the shipping page, the `ShippingGroupDroplet` needs to keep the current selections until the shopper makes any changes.

Shipping Information Is Not Cleared

Checkout: ① Shipping ② Billing ③ Confirm

Select Shipping Address or Create a Shipping Address

Home
Amy Scott
2130 Centre Street
Suite 200
Berkeley, CA 94720-4208
United States
212-555-8117
[Edit](#) | [Delete](#)

Work
Amy Scott
934 Cottonwood Ave
Berkeley, CA 94702
United States
212-555-1111
[Edit](#) | [Delete](#)

clear=false
initShippingGroups=false
initShippingInfos=false

Save this address

[Ship to this Address](#) [Ship to this Address](#)

When returning to the shipping page from later in the checkout process, do not clear the current settings.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

If the shipping page is being opened by a link from a page later in the purchase process (such as the confirmation page), the `ShippingGroupDroplet` should not reinitialize the `ShippingGroupContainerService`, because that would erase the information entered by the user.

ShippingGroupDroplet Input Parameters

- `clearShippingGroups`: Clears shipping groups
- `clearShippingInfos`: Clears CISIs
- `clear`: Clears both shipping groups and CISIs
- `initBasedOnOrder`: Initializes CISIs from relationships in the order
- `initShippingGroups` and `initShippingInfos`: Initialize the shipping groups or shipping infos from profile data and commerce items
- `createOneInfoPerUnit`: Creates one CSI for each unit of each SKU (set to false by default)
- `shippingGroupTypes`: Is a list of shipping group types to be initialized



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

All of the input parameters listed in the slide, with the exception of `shippingGroupTypes`, are Boolean parameters.

ShippingGroupDroplet Output and Open Parameters

- Output parameters:
 - shippingInfos: A map of CISIs
 - order: The current order
 - shippingGroups: A map of groups in ShippingGroupContainerService
- Open parameter:
 - output
 - The ShippingGroupFormHandler form can be inside this open parameter or appear later in the JSP.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

You will learn more about the ShippingGroupFormHandler in the next section of this lesson.

Quiz

The ShippingGroupDroplet always clears and initializes shipping groups and CommerceItemShippingInfos.

- a. True
- b. False



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: b

SGD clears and initializes shipping groups and CISIs when the customer visits the shipping page for the first time. When the customer returns to the order, you want to keep any information about shipping groups that the customer already entered. In that case, SGD does not clear or initialize shipping groups and CISIs.

Road Map

- Collecting shipping information
- Shipping form handlers
- Shipping groups
- Adding a custom shipping group type



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Adding New Shipping Groups

- Two form handlers allow shoppers to add new shipping groups:
 - CreateHardgoodShippingGroupFormHandler
 - CreateElectronicShippingGroupFormHandler
- Both extend PurchaseProcessFormHandler.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Splitting Shipping

- `ShippingGroupFormHandler` (SGFH) is used to gather shipping information or split shipping.
 - Has a `currentList` property that points to current CISIs
- Three handle methods:
 - `specifyDefaultShippingGroup`: Allows users to specify the shipping group for items not linked to other groups
 - `splitShippingInfos`: Reallocates amounts between CISIs
 - `applyShippingGroups`: Applies changes to the Order object



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `currentList` property contains all `CommerceItems`, and loops through the CISIs on each `CommerceItem` (a nested loop) to create the forms.

ShippingInfoFormHandler

- ShippingInfoFormHandler is the CRS extension that extends ShippingGroupFormHandler.
- It is a simplified class that handles:
 - Creating new shipping groups
 - Allocating shipping
 - Setting the shipping method
 - Initializing billing address from selected shipping address, if desired
- The handle methods are listed in slide notes.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The handle methods available for ShippingInfoFormHandler are:

- addShippingAddress
- cancel
- createAndMoveToBilling
- editShippingAddress
- initEditAddressForm
- initMultipleShippingForm
- initSingleShippingForm
- moveToBilling
- removeShippingAddress
- updateShippingMethod

Example C: The Shopper Ships the Same Item to Multiple Addresses

Ship to Multiple Addresses

Create New Address | Edit Addresses

Hepplewhite Side Table
Finish: Antique Red Ship To:
xsku2091_2 In Stock

Save 20% on any ATG Home item excluding Home Accents
1 x \$119.20 was \$149.00

Home - 6 Willowbrook Circle

UPS Ground

Hepplewhite Side Table
Finish: Antique Red Ship To:
xsku2091_2 In Stock

Via:

1 x \$149.00

Home - 6 Willowbrook Circle

UPS Ground

Paris Clock
xsku2032 In Stock

Ship To:
Via:

Home

UPS Gr

The SGFH's
applyShippingGroups
handle method is used to
apply changes to the
Order

Continue

cisi0
shippingGroupName=Home
quantity=2
splitQuantity=0
splitShippingGroupName=null

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the shopper has two tables and one clock in the cart. When splitting shipping in CRS, each quantity of a commerce item appears on its own line. You can then determine by line the shipping address and shipping method. In this example, the shopper wants to ship one of the tables to the home address and the other to the work address. The form handler sets up the form and the CISI holds the information on what is happening before, during, and after the split. The next slides show what happens with the CISI during the process.

Splitting Shipping and CISIs: During the Split

The SGFH's `splitShippingInfos` handle method is used in this process.

cisi0

```
shippingGroupName=Home
quantity=2
splitQuantity=1
splitShippingGroupName=My Work
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The shopper has selected the work address for the second line of the table commerce item. During the process, the CISI changes. There are only two of this table, and one will be shipped to an address different from the default address. The CISI's `splitQuantity` changes from 0 to 1 and the `splitShippingGroupName` changes from null to "My Work."

The SGFH's `splitShippingInfos` handle method reallocates amounts between CISIs. The two CISIs that are created by the example in the slide appear in the next slide.

Splitting Shipping and CISIs: After the Split

The quantities of the commerce item have been split into two CISIs:

```
cisi0  
shippingGroupName=Home  
quantity=1  
splitQuantity=0  
splitShippingGroupName=null
```

```
cisi1  
shippingGroupName=My Work  
quantity=1  
splitQuantity=0  
splitShippingGroupName=null
```

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

- Collecting shipping information
- Shipping form handlers
- **Shipping groups**
- Adding a custom shipping group type



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Shipping Groups

- Shipping groups represent the shipping information for an Order, including the address and shipping method.
- Orders may have more than one shipping group.
 - Example: Ship some items to the home and some to the office
 - Not all stores allow shoppers to split shipping.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

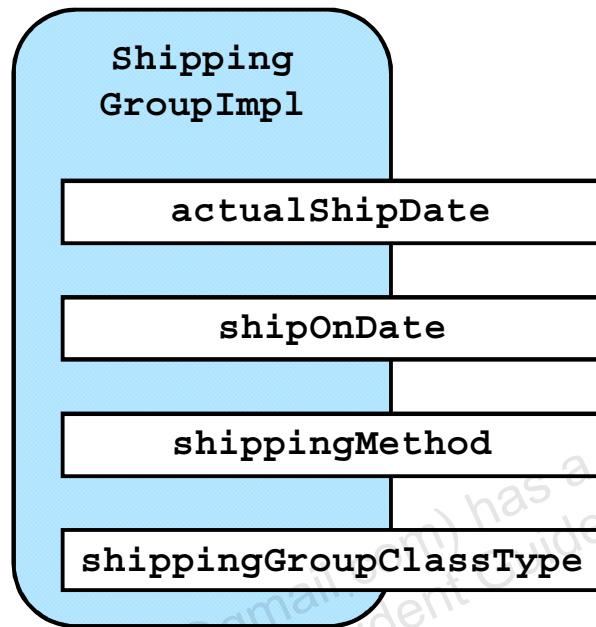
Existing Shipping Group Types

- Commerce includes two shipping group classes:
 - HardgoodShippingGroup
 - ElectronicShippingGroup
- Both extend the ShippingGroupImpl class.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ShippingGroupImpl Properties



ORACLE®

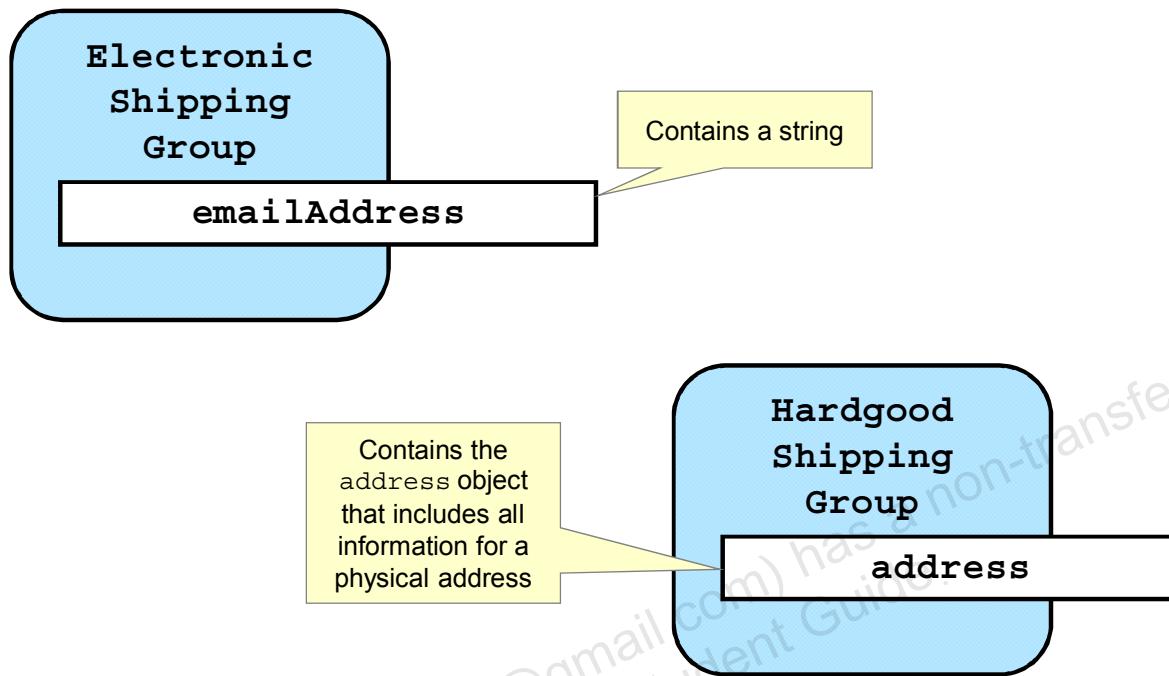
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The properties in the slide are:

- `actualShipDate`: The date of shipment, as set by the fulfillment system
- `shipOnDate`: The scheduled shipping date. This can be entered by the customer, if desired.
- `shippingMethod`: The shipping method, such as “Ground” or “Next Day Air”
- `shippingGroupClassType`: The property that maps the shipping group to the class

These are just a few of the available properties. For more information, see the `atg.commerce.order.ShippingGroupImpl` class in the *API Platform Reference*.

Additional Shipping Group Properties



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ElectronicShippingGroup adds the `emailAddress` property.
HardgoodShippingGroup adds the `address` property.

ShippingGroupManager

- ShippingGroupManager has methods to:
 - Create shipping groups
 - Add shipping groups to orders
 - Remove shipping groups from orders
 - Associate shipping groups and payment groups
- CommerceItemManager has methods for adding items to shipping groups.
- The CRS extension is StoreShippingGroupManager.
 - Adds methods to deal with gift items



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the component is /atg/commerce/order/ShippingGroupManager. The form handlers call methods on ShippingGroupManager.

Road Map

- Collecting shipping information
- Shipping form handlers
- Shipping groups
- Adding a custom shipping group type



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Steps to Add a Custom Shipping Group Type

1. Create a shipping group class that extends `ShippingGroupImpl` or one of the existing shipping group classes.
2. Create an item descriptor in the Order Repository that is a subtype of `shippingGroup`, or one of its subtypes.
3. Edit the `OrderTools` component:
 - Add the new item descriptor to `beanNameToItemDescriptorMap`
 - Add the class to `shippingClassMap`
 - Map the new shipping group class to the appropriate address class in `shippingGroupShippingAddressClassNameMap`



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

An example of an address class to map the shipping address to is `atg.commerce.order.RepositoryContactInfo`.

Steps to Add a Custom Shipping Group Type

4. Create an initializer class that implements `ShippingGroupInitializer`.
5. Create an initializer component and add it to `ShippingGroupDroplet's` `shippingGroupInitializers` property.
6. Extend the `validateShippingGroup` pipeline:
 - Create a pipeline processor to validate the shipping group for checkout.
 - Add the pipeline processor to the pipeline.
7. If shoppers will be adding these groups to their order themselves, create a form handler that extends `PurchaseProcessFormHandler`.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Extending pipelines will be covered in more detail in the lesson titled “Pipelines and Shipping Methods.”

Creating a ShippingGroupInitializer

To create a ShippingGroupInitializer:

- Implement the following interfaces:
 - ShippingGroupInitializer
 - ShippingGroupMatcher
- Define key methods as shown in the next slide



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ShippingGroupMatcher catches redundancies.

Defining the ShippingGroupInitializer Methods

- In `initializeShippingGroups()`:
 - Create one or more new shipping groups using `ShippingGroupManager.createShippingGroup()`
 - Retrieve the data needed to populate the shipping groups
 - Add the new shipping groups to the `pShippingGroupMapContainer` parameter
- In `getNewShippingGroupName()`, calculate the appropriate name for the new shipping groups of this type.
- In `matchShippingGroup()`, calculate whether given shipping group matches any groups in the given `ShippingGroupMapContainer`.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe the process for collecting shipping information
- Add custom shipping groups

For More Information

ATG Commerce Guide to Setting Up a Store

- Implementing Order Retrieval
- Adding Shipping Information to Shopping Carts

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Practice 15 Overview: Creating a Custom Shipping Group

This practice covers the following topics:

- Creating a custom shipping group type for physical stores
- Creating a shipping group initializer
- Displaying the new shipping group type in your store



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Pipelines and Shipping Methods

16

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Extend pipelines
- Configure shipping methods
- Add custom shipping methods

Road Map

- Extending pipelines
- Configuring and extending shipping methods

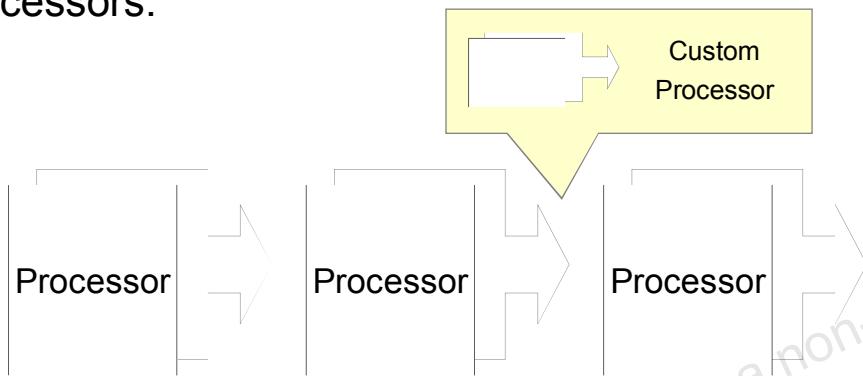


ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Review: Pipeline Basics

- ATG Commerce has several pipeline chains, typically called by the business layer classes, made up of processors.



- Pipeline Manager components, configured with an XML file, run processor chains.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

You first encountered pipelines in the lesson titled “Extending Orders.” Pipelines are also important when creating new shipping groups (and payment groups, covered in a subsequent lesson). A pipeline chain is made up of processors, each performing a specific function. You can easily customize a pipeline chain by inserting your own processors.

The `PipelineManager` is responsible for the creation, registration, and removal of pipeline chains.

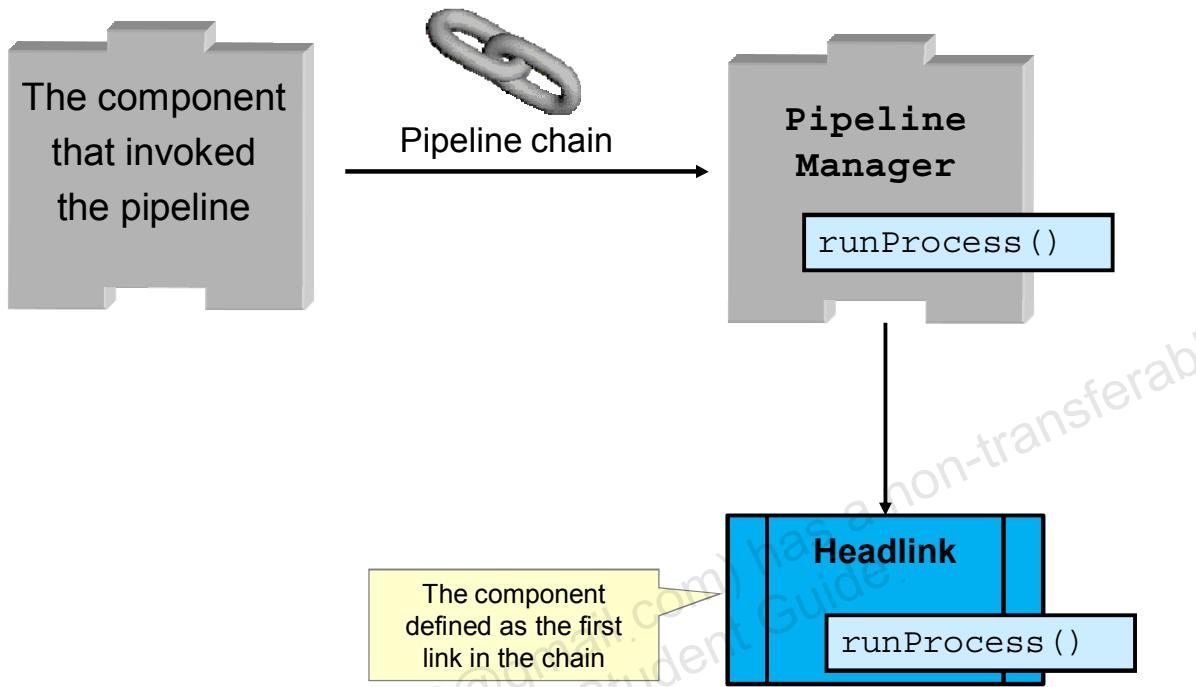
`PipelineManager` uses the Java Transaction API. Each processor in the chain can mark the chain to be rolled back. If none mark the transaction, it is committed when the chain finishes execution.

`PipelineManager` calls its `runProcess()` method with the chain's ID to execute a chain.

Processors return status codes, which may determine what processor to execute next.

Chain IDs are found in pipeline XML files. Examples: `processOrder`, `validateForCheckout`, `validateShippingInfo`.

Invoking a Pipeline

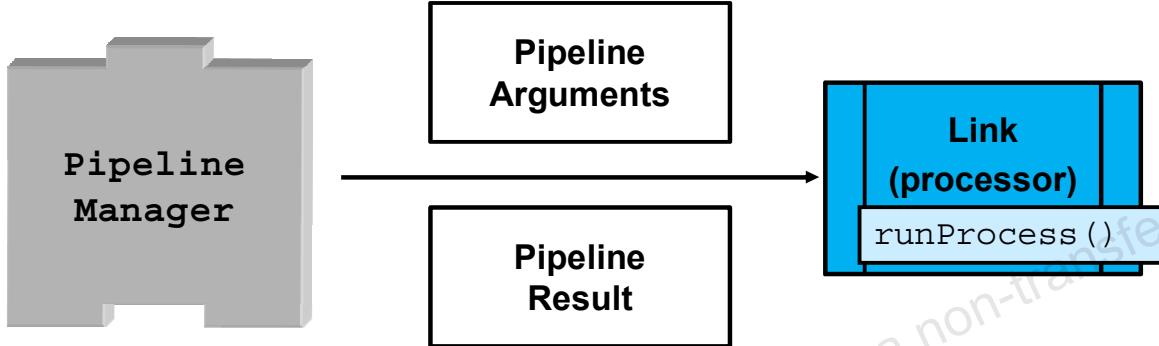


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When a component needs to run a pipeline, it calls the `runProcess` method of `PipelineManager`, with the name of the chain to be run. `PipelineManager` then calls the `runProcess` method of the component defined as the first link in the chain (also known as the headlink).

Calling the `runProcess()` Method of Pipeline Links



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Each time `PipelineManager` calls the `runProcess` method of a pipeline link, it passes the link references to an arguments object and a result object.

Pipeline Arguments

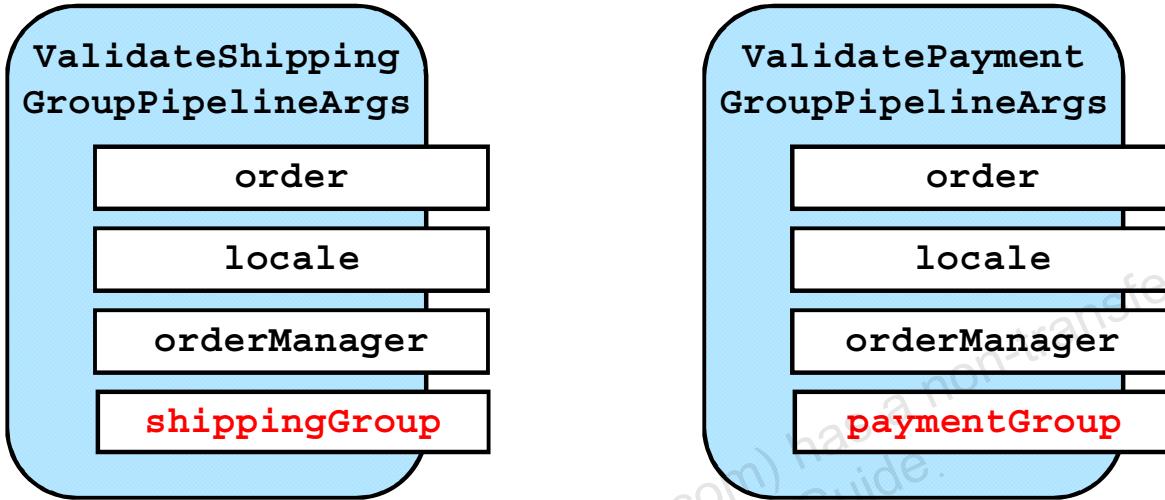
- Pipeline arguments contain information needed for the pipeline link to run.
- Different pipelines are given different classes of argument objects.
 - Order validation pipelines use a subclass of ValidationPipelineArgs
 - ValidateShippingGroupPipelineArgs
 - ValidatePaymentGroupPipelineArgs



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

This lesson discusses ValidateShippingGroupPipelineArgs. You will learn more about ValidatePaymentGroupPipelineArgs in the lesson titled “Payment Groups.”

Default Subclasses of ValidatePipelineArgs



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

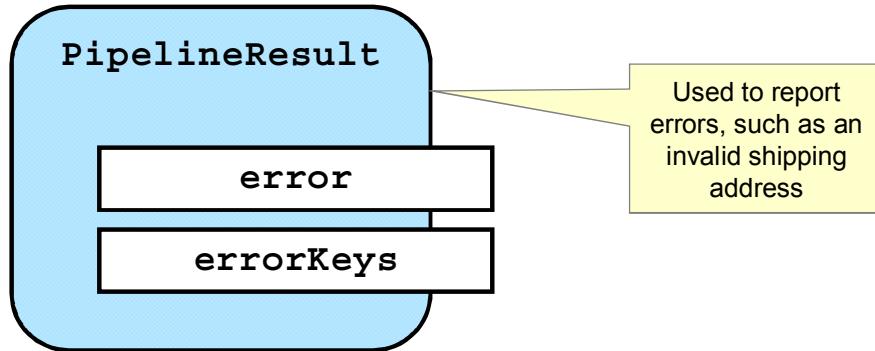
The full path to the class is

`atg.commerce.order.processor.ValidateShippingGroupPipelineArgs`. Three of the properties are inherited from `ValidatePipelineArgs`:

- `order`: The order whose payment or shipping group is being validated
- `locale`: The locale to use for user-visible error message
- `orderManager`: The order manager that invoked the current pipeline chain

`ValidateShippingGroupPipelineArgs` adds `shippingGroup`, which is the shipping group being validated. Similarly, `ValidatePaymentGroupPipelineArgs` adds `paymentGroup`.

Pipeline Result Object



- Pipeline links call the `addError()` method, such as

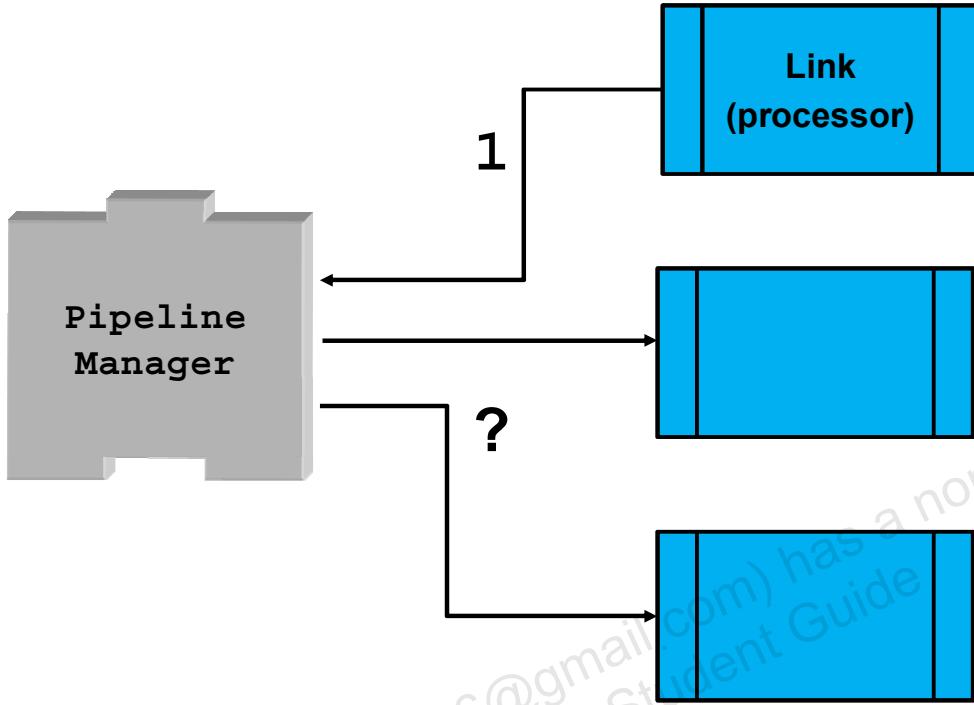
```
pResult.addError("MyErrorKey", "The text of  
my error");
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The default class is `atg.service.pipeline.PipelineResultImpl`. The `addError()` method adds an error object mapped to a key. There is also a `removeError()` method which removes the error object mapped to a key from the `PipelineResult`. The errors are passed along the chain. At end of the chain, you get all of the errors.

Moving to the Next Link in a Chain

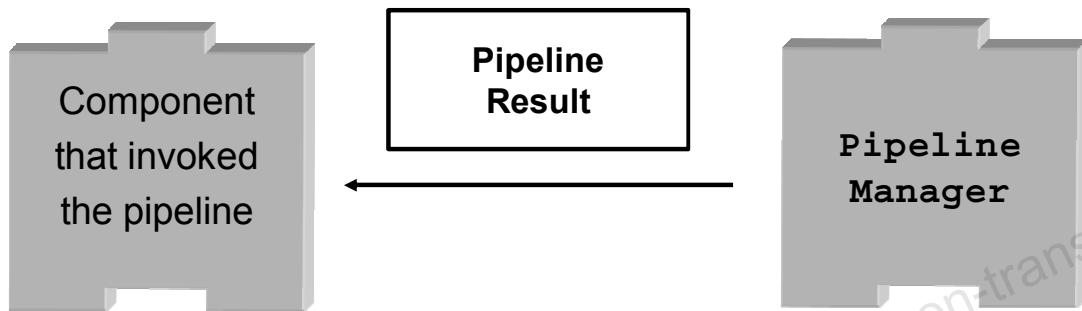


ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When the link is done running, it returns an integer to PipelineManager.
PipelineManager uses this integer to determine which link to execute next.

Ending the Pipeline



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When the pipeline is finished, `PipelineManager` returns the `PipelineResult` object to the calling component.

Pipeline Manager Components

- Most ATG Commerce pipelines are configured in two components:
 - /atg/commerce/PipelineManager
 - The pipelines for validating and processing orders
 - /atg/commerce/payment/PaymentPipelineManager
 - The pipelines for processing payments
- Use the Component Browser to see details on the live pipelines.
 - Click the `definitionFile` property to see the combined XML configuration.



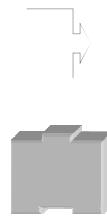
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Other pipeline manager components exist for other modules (for example, the Fulfillment module). The Fulfillment module's pipeline manager component is `/atg/commerce/fulfillment/FulfillmentPipelineManager`.

Extending a Pipeline

Steps to customize a pipeline chain:

1. Create a processor that implements the `atg.service.pipeline.PipelineProcessor` interface.
2. Create a nucleus component for your processor.
3. Extend the existing pipeline definition XML:
 - Add a transition from the previous pipeline link to yours
 - Add your pipeline link



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Many of the default pipeline classes are able to have new return codes configured in their properties, making it easy to add new links.

PipelineProcessor Class

- Implement the `retCodes []` data array property to give the possible integer return codes for this processor.
- Implement `runProcess ()`:
 - Extract the needed information from `pParams`.
 - Add errors as necessary to `pResult`.
 - Return one of the defined integer return codes.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Example: Extending a Pipeline

The examples that follow show a new shipping validator processor called myLink.

- Shipping validator example: The code excerpts from the new processor class
- Extending the validateShippingGroup chain: The XML to add the link to an existing chain
- Adding the return code and shipping group type



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Shipping Validator Example: Resource Bundle and Return Code

```
static final String RESOURCE_NAME =
"atg.commerce.order.OrderResources";

private static ResourceBundle sResourceBundle =
    LayeredResourceBundle.getBundle(RESOURCE_NAME,
        atg.service.dynamo.LangLicense.getLicensedDefault());

private final int SUCCESS = 1;
protected int[] mRetCodes = { SUCCESS };
public int[] getRetCodes() {
    return mRetCodes;
}
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

In the code example in the slide, the first section shows how the default resource bundle is instantiated and populated with the correct resource bundle for the current locale. The processor code continues by defining a constant, SUCCESS, which is equal to 1. This is the processor's only possible return code.

Shipping Validator Example: runProcess and Error Messages

```
public int runProcess(Object pParam, PipelineResult  
pResult) throws Exception {  
    ValidateShippingGroupPipelineArgs args =  
(ValidateShippingGroupPipelineArgs) pParam;  
  
    ShippingGroup shippingGroup = args.getShippingGroup();  
    if (shippingGroup == null)  
        throw new  
InvalidParameterException(ResourceUtils.getMsgResource  
("InvalidShippingGroupParameter", RESOURCE_NAME,  
sResourceBundle));  
  
    if (!(shippingGroup instanceof HardgoodShippingGroup))  
        throw new  
InvalidParameterException(ResourceUtils.getMsgResource  
("InvalidShippingGroupParameter", RESOURCE_NAME,  
sResourceBundle));
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the `runProcess()` method, the needed information is extracted from `ValidateShippingGroupPipelineArgs`. If the shipping group is null or not of the type that this processor is intended to run on, an exception is thrown using messages from the default resource bundle.

Shipping Validator Example: Returning SUCCESS

```
...code to validate shipping group...
...add error to pResult if any problems found...

return SUCCESS;
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `runProcess()` method then goes on to perform whatever operations are needed to validate the shipping group. If a problem is found, an error is added to `pResult`. Regardless of whether or not the shipping group passes the validation, the method returns the `SUCCESS` constant to indicate that `runProcess()` finished.

Extending the validateShippingGroup Chain

/atg/commerce/commercepipeline.xml

```
<pipelinemanager>
  <pipelinechain name="validateShippingGroup">
    <pipelinelink name="dispatchOnSGType">
      <transition returnvalue="5000"
        link="mylink"/>
    </pipelinelink>
    ...
    <pipelinelink name="mylink"
      transaction="TX_MANDATORY">
      <processor jndi=
        "/com/mystore/MyProcessorComponent"/>
    </pipelinelink>
  </pipelinechain>
</pipelinemanager>
```

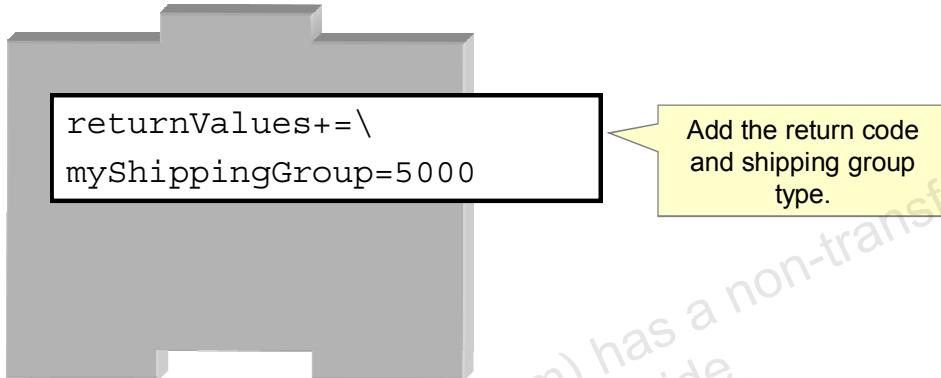


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The validateShippingGroup chain is configured in /atg/commerce/commercepipeline.xml. You want to add the transition to the dispatchOnSGType link. You then add a link for your new processor. The example in the slide assumes that you have created a custom processor component based on your new shipping validator class called /com/mystore/MyProcessorComponent.

Adding the Return Code and Shipping Group Type

ValidateShippingGroupByType



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full name of the component is /atg/commerce/order/processor/ValidateShippingGroupByType. The example in the slide assumes that you have created a custom shipping group type called myShippingGroup. It has been added to the default component with its return code (as shown in the code example in the previous slide).

Road Map

- Extending pipelines
- Configuring and extending shipping methods



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Shipping Methods

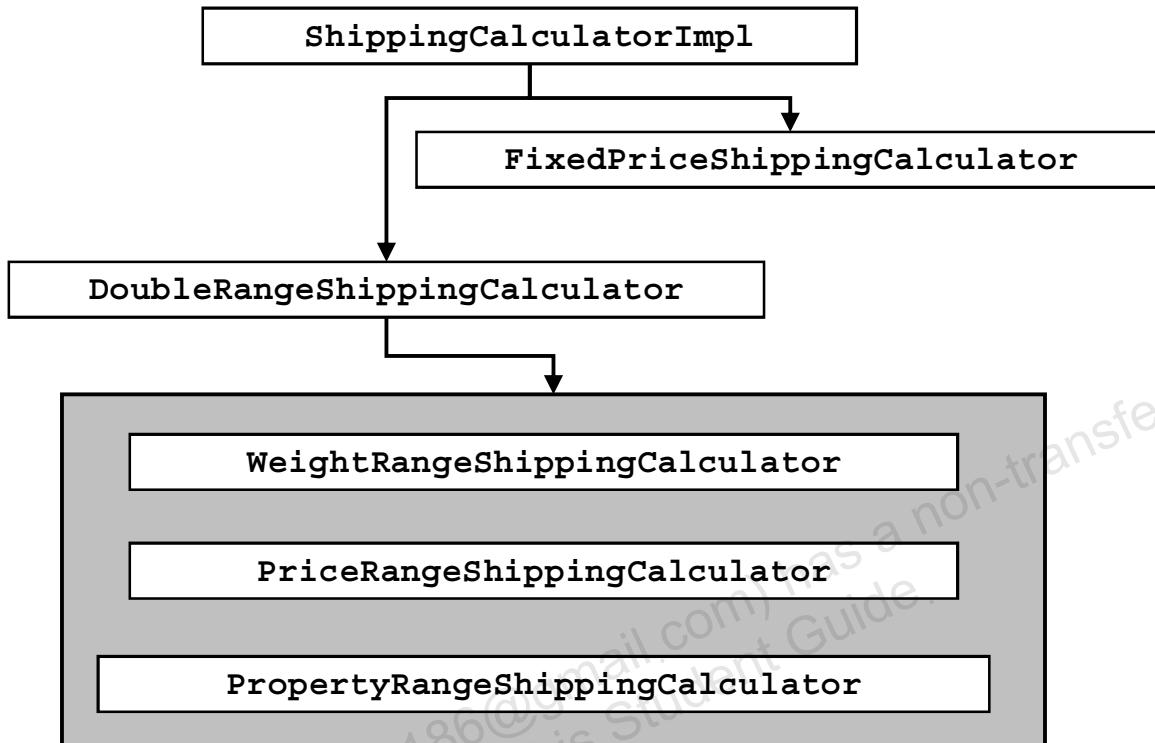
- Shipping methods allow shoppers to choose how each shipping group will be shipped.
 - Examples: Ground, 2-day courier, overnight
- Each shipping method is a `ShippingCalculator` component.
- Shipping methods can be extended to:
 - Add new types of pricing
 - Restrict when the method is available

 ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The main purpose of a `ShippingCalculator` component is to calculate the price of shipping.

Shipping Calculator Class Hierarchy



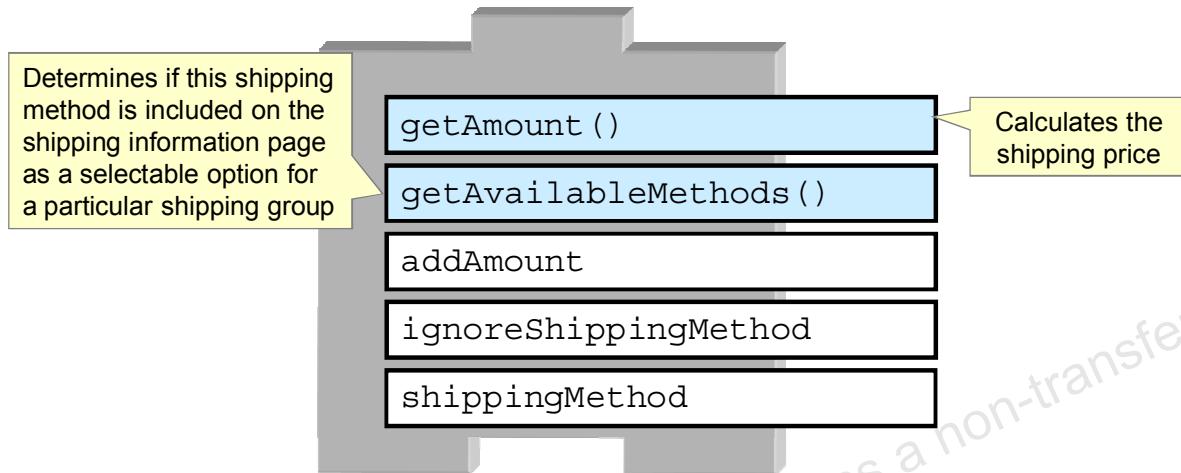
ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

All of the classes shown in the slide are in the `atg.commerce.pricing` package. The base class, `ShippingCalculatorImpl`, implements the `atg.commerce.pricing.ShippingPricingCalculator` interface.

ShippingCalculatorImpl Properties and Methods

ShippingCalculatorImpl class



- Both methods are passed a wide range of information, including the order, the relevant shipping group, the locale, and the shopper's profile.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `atg.commerce.pricing.ShippingCalculatorImpl` class is an abstract class that is a starting point for some general functionality on calculating pricing for shipping groups.

The properties of the `ShippingCalculatorImpl` class include:

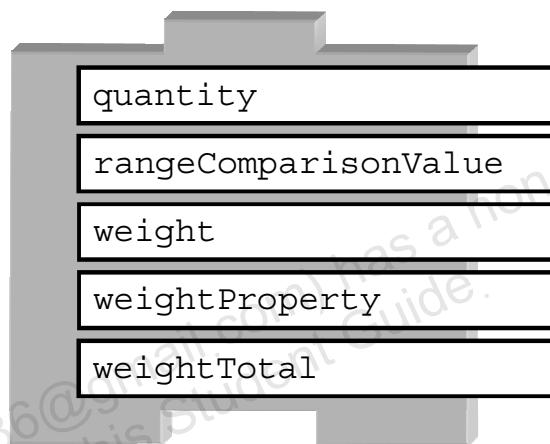
- `addAmount`: A Boolean. If it is true, the result of the `getAmount()` method is added to the existing amount in the `ShippingPriceInfo.amount`. This allows surcharges to be tacked on.
- `ignoreShippingMethod`: A Boolean. If it is true, this calculator does not expose a shipping method name (through `getAvailableMethods`). In addition, this calculator always attempts to perform pricing. This option is available if the user is not given a choice of different shipping methods.
- `shippingMethod`: This property should be set to the name (String) of a particular delivery process, such as UPS Ground, UPS 2-day, or UPS Next Day.

One example of a shipping method that may not be an option on a shipping information page is a courier shipping method that is not available for PO boxes.

Existing Shipping Calculators

- FixedPriceShippingCalculator
- PriceRangeShippingCalculator
- WeightRangeShippingCalculator
- PropertyRangeShippingCalculator

**PriceRange
Shipping
Calculator**



ORACLE®

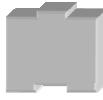
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

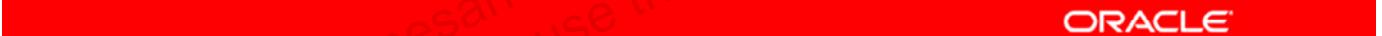
- FixedPriceShippingCalculator sets the fixed price, configured in the amount property. For example, the ground shipping is always \$10.
- PriceRangeShippingCalculator sets the price based on the total cost of the items in the shipping group. The shipping prices are configured in the ranges property. For example, if the shipping group has a total item cost between 0 and \$100, the shipping will be \$10; between \$101 to \$200, \$20.
- WeightRangeShippingCalculator sets the price based on the total weight of the items in the shipping group. It requires that a weight property be added to SKUs and the name of that property be configured in the calculator's weightProperty property. It can either calculate the total price or add a surcharge based on its addAmount property.
- PropertyRangeShippingCalculator works similarly to WeightRangeShippingCalculator, except it can be based on any SKU property (such as volume). You configure the SKU property name in the component's propertyName property.

The example in the slide shows some of the properties of PriceRangeShippingCalculator:

- **quantity**: The quantity of a particular item that is being shipped by this shipping group
- **rangeComparisonValue**: The value that will be used in the range comparison
- **weight**: The weight associated with an instance of a commerce item
- **weightProperty**: The property that specifies the weight of an item
- **weightTotal**: The total weight of the items being shipped to the specified ShippingGroup

Creating a New Shipping Method

- 1. If needed, create a class that extends `ShippingCalculatorImpl`, `DoubleRangeShippingCalculator`, or an existing calculator.**
 - For an example, see the `atg.projects.store.pricing.SurchargeShippingCalculator` class in CRS.
- 2. Create a component based on the new `ShippingCalculator` class.** 
- 3. Add the component to `ShippingPricingEngine.preCalculators`.** 

 ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Restricting Existing Shipping Methods

To restrict an existing shipping method:

- Extend the relevant shipping calculator
 - Override the `getAvailableMethods()` to determine when the shipping method can be used.
- Configure the existing shipping method component(s) to use the new class



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Quiz

Which of the following statements describe shipping methods?
(Select all that apply.)

- a. Shipping methods allow shoppers to choose how each shipping group will be shipped.
- b. You must create a class that extends `ShippingCalculatorImpl`, `DoubleRangeShippingCalculator`, or an existing calculator for your shipping method.
- c. You add your shipping method component to the `ShippingPricingEngine`'s `preCalculators` property.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Answer b: You only need to create a custom class for your shipping method when one of the default shipping calculator classes does not meet your needs.

Summary

In this lesson, you should have learned how to:

- Extend pipelines
- Configure shipping methods
- Add custom shipping methods

For More Information

- *ATG Commerce Guide to Setting Up a Store*
 - Implementing Order Retrieval
 - Adding Shipping Information to Shopping Carts
- *ATG Commerce Programming Guide*
 - Processor Chains and the Pipeline Manager



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Practice 16 Overview: Customizing Shipping Methods

This practice covers the following topics:

- Creating a custom ship-to-store shipping method
- Restricting the ship-to-store shipping method to only ship-to-store shipping groups
- Extending the shipping group validation pipeline to validate ship-to-store shipping groups



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The exercises in this practice continue the work you started in the previous practice.

17

Payment Groups

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the default and CRS payment collection processes
- Use the provided payment group types
- Add custom payment group types
- Describe and extend the processing of payments for completed orders

Road Map

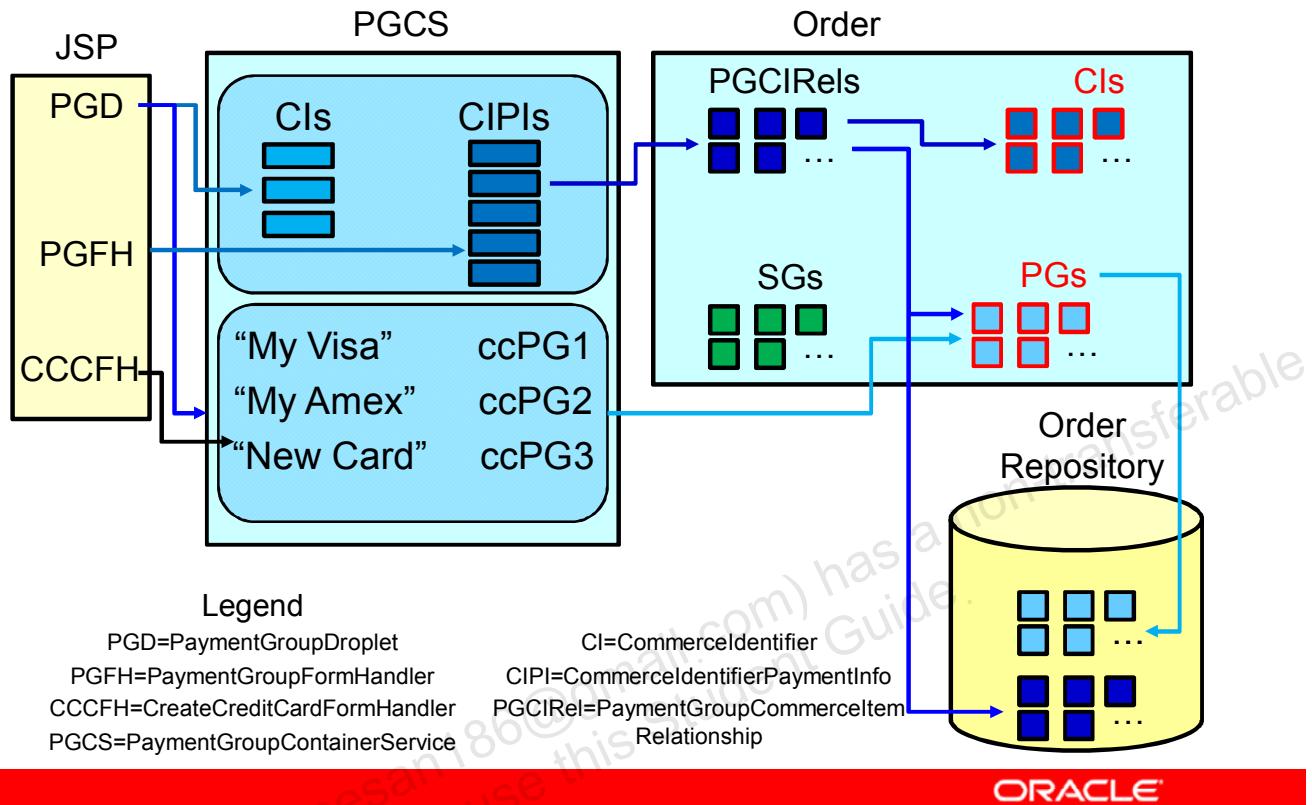
- Collecting payment information
- Payment groups and payment managers
- Claimable payment groups
- Payment form handlers
- Processing payments
- Adding a custom payment group



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Payment Architecture



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

The PaymentGroupContainerService (PGCS) is analogous to the ShippingGroupContainerService, and it contains two maps, stored in the CommerceIdentifierPaymentInfoMap and PaymentGroupMap properties. Respectively, these properties contain:

- CommerceIdentifiers (Cl)s mapped to CommerceIdentifierPaymentInfos (CIPPIs)
- Text keys mapped to PaymentGroups (in the diagram above, CreditCard payment groups only)

CommerceIdentifier is an interface with one property, `i.d.` Subinterfaces include Order, ShippingGroup, CommerceItem, PaymentGroup, and Relationship, among others. For each of the classes implementing these subinterfaces (such as OrderImpl and ShippingGroupImpl), there is a correspondingly named CIPPI class. Thus, entries in the Cl-to-CIPPI map may contain, at any point in time, one or more of the following types of entries:

- The current Order object mapping to one or more (if payment has been split at the order level) OrderPaymentInfo instances
- The current Order object mapping to one or more (if payment has been split at the tax level) TaxPaymentInfo instances

- A `CommerceItem` from the current Order mapping to one or more (if payment has been split across this `CommerceItem`) `CommerceItemPaymentInfo` instances
- A `ShippingGroup` from the current Order mapping to one or more (if payment has been split across this `ShippingGroup`) `ShippingGroupPaymentInfo` instances

Exactly what is in the map at a particular moment depends on whether split payment is supported (and if so, at what level), what point in the checkout process the customer has reached, and the particular choices for updating or splitting payment the customer has made. The map's contents are used to present a summary of the current state of the Order, and to allow users to change a CI's payment method or, if it is supported, split a CI across multiple methods. (User-selected methods are stored in `PaymentGroups`.)

In the example in the slide, text keys in `PaymentGroupMap` are read from the current customer's `defaultCreditCard` and `creditCards` repository properties. The latter is a one-to-one mapping of String keys, provided by the user during registration or afterwards, to `creditCard` repository items. In B2B implementations, derived properties, such as `derivedCreditCards`, `allCostCenters`, and `defaultCostCenter`, are often used (whose derivations are from `parentOrganization`). `PaymentGroupMap` is used to present payment options to the customer.

`PaymentGroupDroplet` (PGD) initializes both containers. Keys from (in the example in the slide) `Profile.creditCards` are mapped to `CreditCard` payment group objects, created by the appropriate initializer from the corresponding `creditCard RepositoryItem` on the `Profile` map. For a new Order, any CI being added to the `CIPIMap` during checkout is mapped initially to one `CIPIMap`. For updates to an existing order, any re-mappings should be based on that Order's current state: If a payment for a CI has been split, that CI should map to multiple `CIPIMaps`.

The `CreateCreditCardFormHandler` (CCCFH) lets users add a new credit card, updating `PaymentGroupMap` accordingly. That card then becomes available in the `PaymentGroupFormHandler` (PGFH) form as a payment option. The PGFH allows customers to change the payment group for a CI, and, if desired, split payments across a CI, updating the `CIPIMap` to reflect their choices.

After the user is satisfied with the payment choices, a handle method on PGFH transfers information from the updated PGCS maps to the Order. `CIPIMaps` are translated into `PaymentGroupCommerceIdentifierRelationships` (PGCIRels). The Order contains the (non-Order) CIs, the PGs, the `ShippingGroups` (SGs), and other relationships, as well as the PGCIRels.

Default Payment Collection Process

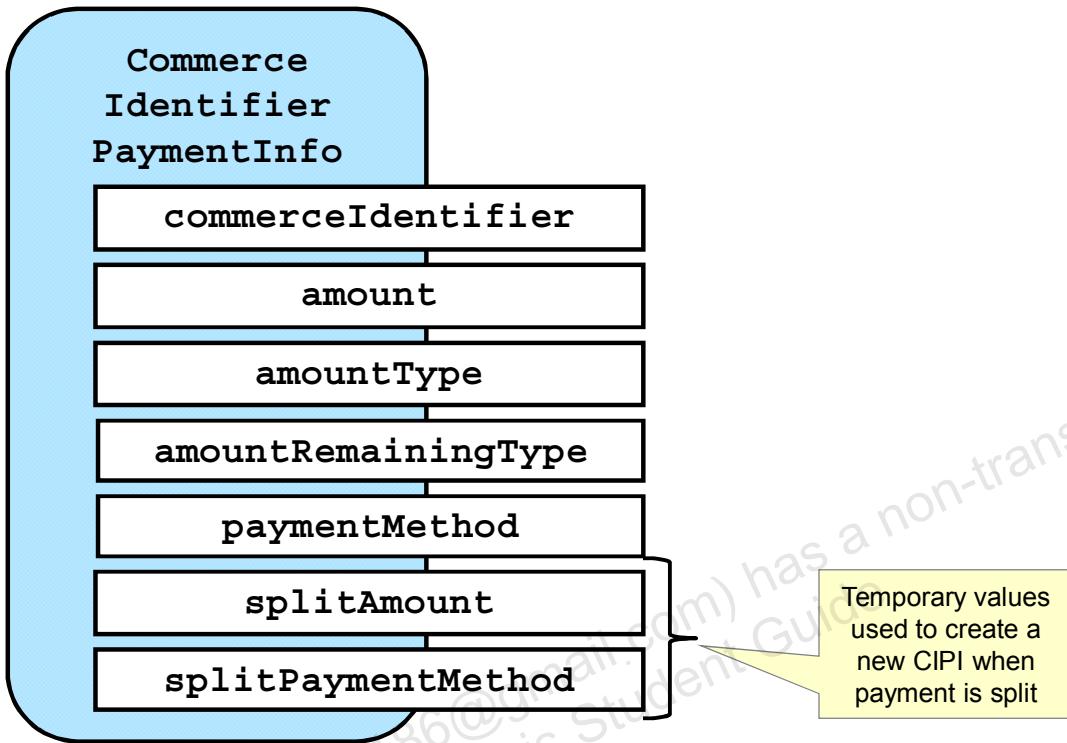
The default payment process is very similar to the process for collecting shipping information:

- PaymentGroupDroplet is used to initialize credit card and store credit payment groups as well as CommerceIdentifierPaymentInfos in PaymentGroupContainerService.
- The shopper uses PaymentGroupFormHandler to allocate payment.
- Payment selections are copied to the Order when the shopper moves to the confirmation page.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

CommerceIdentifierPaymentInfo



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `atg.commerce.order.purchase.CommerceIdentifierPaymentInfo` represents the association between a `CommerceIdentifier` and its payment information, and includes properties that allow the cost of a given quantity or even a single item to be spread across multiple payment groups. These objects store the information needed to create payment Relationships for the Order. Subclasses are `CommerceItemPaymentInfo`, `OrderPaymentInfo`, `ShippingGroupPaymentInfo`, and `TaxPaymentInfo`. The properties listed in the slide are:

- `commerceIdentifier`: The `CommerceIdentifier` for this CIPI
- `amount`: The amount for this CIPI
- `amountType`: The amount relationship type String based on the `CommerceIdentifier` type
- `amountRemainingType`: The amount remaining relationship type String based on the `CommerceIdentifier` type
- `paymentMethod`: The `PaymentMethod` being used for this CIPI
- `splitAmount`: The amount that will be split to another CIPI when splitting payment for this CIPI across multiple payment groups
- `splitPaymentMethod`: The `PaymentMethod` for another CIPI when splitting payment

CRS Payment Collection Process

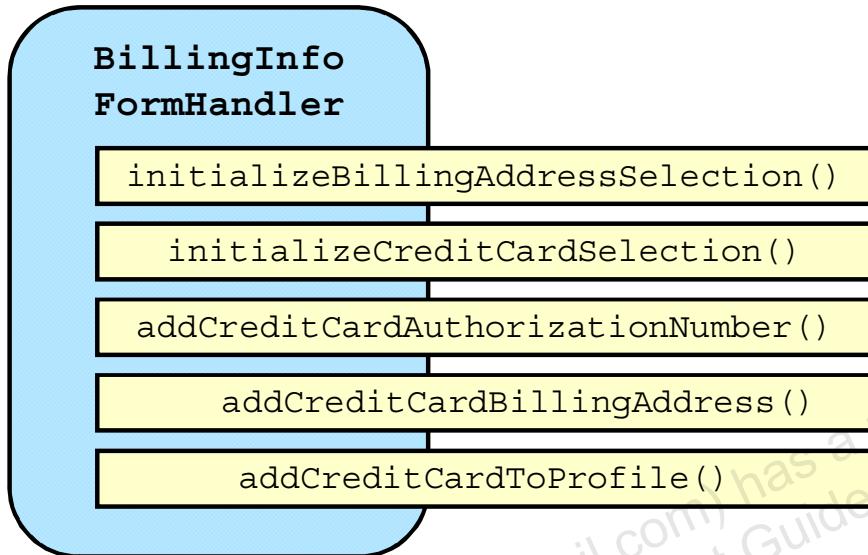
- CRS does *not* use PaymentGroupDroplet or PaymentGroupContainerService.
- The BillingInfoFormHandler class uses BillingProcessHelper to:
 - Initialize payment groups directly in the Order
 - Assign payment amounts to payment groups in the Order by type
- Payment types are used in the following order in CRS:
 - Store credits
 - Gift certificates
 - Credit card for any remaining amount



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The atg.projects.store.order.purchase.BillingInfoFormHandler class extends the atg.commerce.order.purchase.PurchaseProcessFormHandler abstract class.

BillingInfoFormHandler Methods



ORACLE

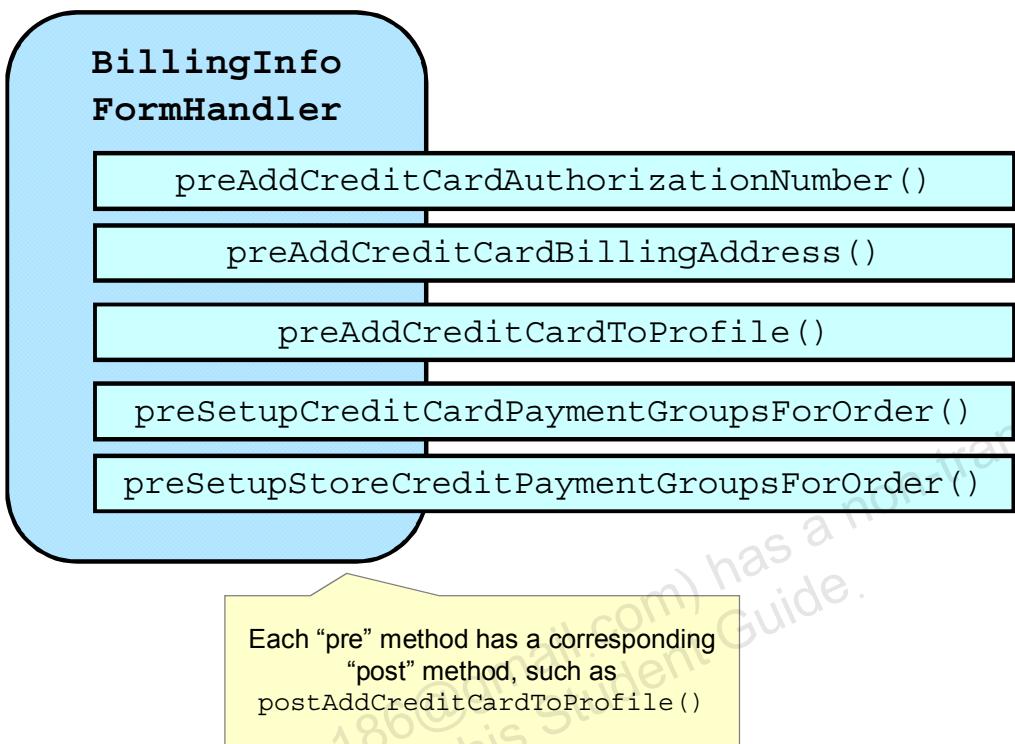
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Key methods of the class perform the following functions:

- `initializeBillingAddressSelection`: Sets initial values to the `storedAddressSelection` and `usingSavedAddress` fields based on the order
- `initializeCreditCardSelection`: Sets initial values to the `storedCreditCardName` and `usingProfileCreditCard` fields based on the order
- `addCreditCardAuthorizationNumber`: Adds and validates the credit card authorization number of the credit card if order payment is paid by the credit card
- `addCreditCardBillingAddress`: Adds and validates the billing Address to the credit card if the order payment is paid by the credit card
- `addCreditCardToProfile`: Used for any processing required after the billing information is entered

For more information on the properties and additional methods of this class, see the *API Reference* for CRS.

BillingInfoFormHandler Pre and Post Methods



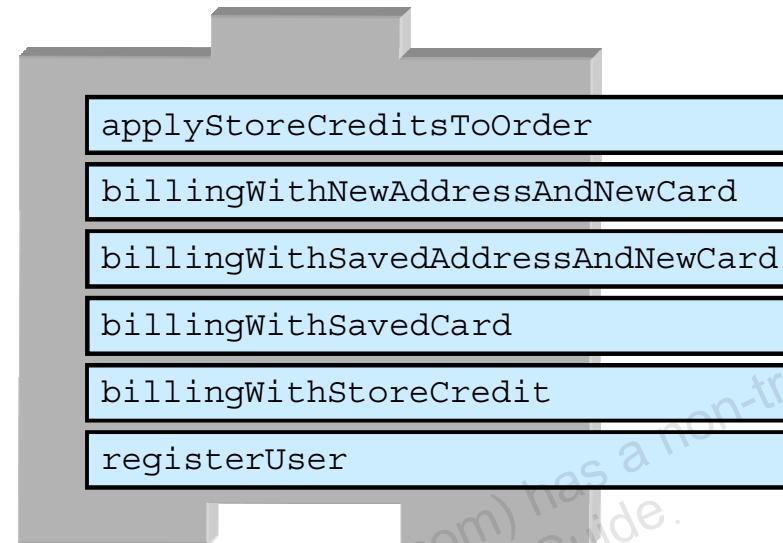
ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Similar to the out-of-the-box form handlers, **BillingInfoFormHandler** includes empty pre and post methods to allow developers to easily extend its functionality. The methods listed in the slide are just a few of the available methods. For a full list of the pre and post methods and their functions, see the *API Reference* for CRS.

BillingFormHandler Handle Methods (CRS Only)

BillingFormHandler



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

/atg/store/order/purchase/BillingFormHandler is the CRS component based on the atg.projects.store.order.purchase.BillingInfoFormHandler class. It has a number of handle methods that advance the shopper to the confirmation page:

- applyStoreCreditsToOrder: Apply available store credits to the order
- billingWithNewAddressAndNewCard: Move to confirmation using the new billing address and new credit card information
- billingWithSavedAddressAndNewCard: Move to confirmation using the saved billing address and new credit card information
- billingWithSavedCard: Move to confirmation using the saved credit card
- billingWithStoreCredit: Move to confirmation using only store credit as the payment method
- registerUser: Register an account and link all the saved information like the addresses and credit card information

The CRS billing page, store.war/checkout/billing.jsp, includes the store.war/checkout/gadget/billingForm.jsp gadget, which contains the form used for specifying payment options. This gadget uses BillingFormHandler to create the form and apply the payment options the customer specifies.

Road Map

- Collecting payment information
- Payment groups and payment managers
- Claimable payment groups
- Payment form handlers
- Processing payments
- Adding a custom payment group



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Payment Groups

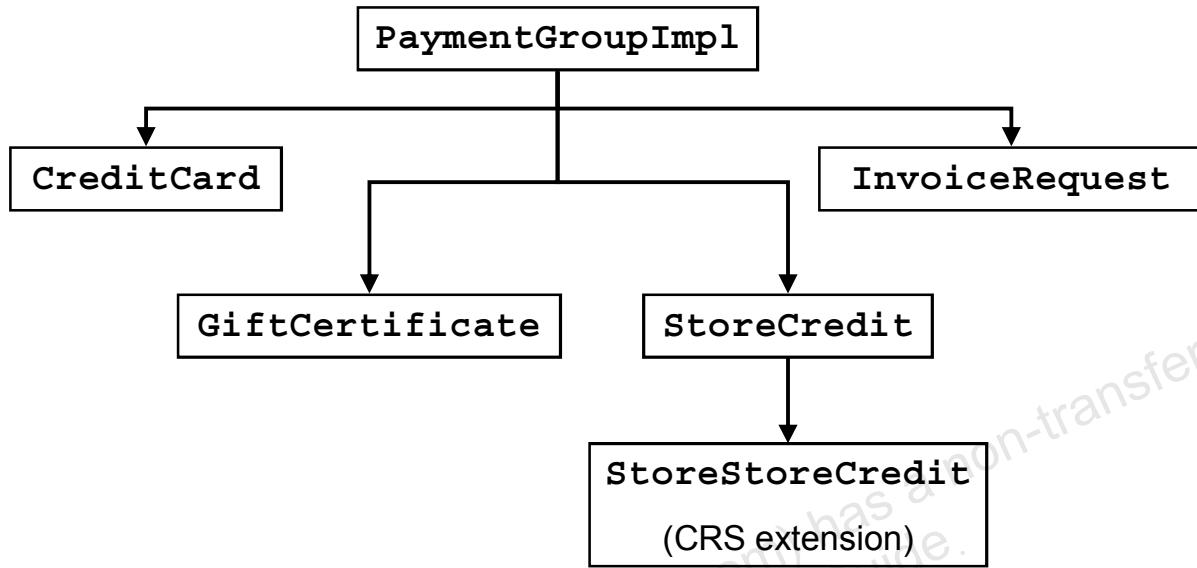
- Four payment group types available out of box:
 - CreditCard
 - StoreCredit
 - GiftCertificate
 - InvoiceRequest
 - For use with purchase orders
- You can add custom payment methods, such as Bill Me Later or PayPal.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

All the payment group classes are in the `atg.commerce.order` package.

Payment Group Class Hierarchy



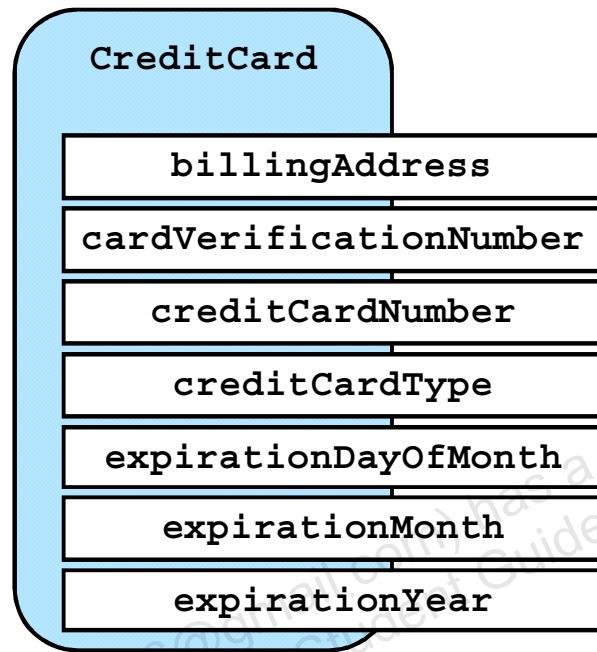
ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `InvoiceRequest` object holds information necessary for purchase orders, such as purchase order number, payment terms, and the billing address. For more information, see the *API Platform Reference*.

The default store credit payment group is extended by CRS to contain the amount applied to the order.

CreditCard Properties

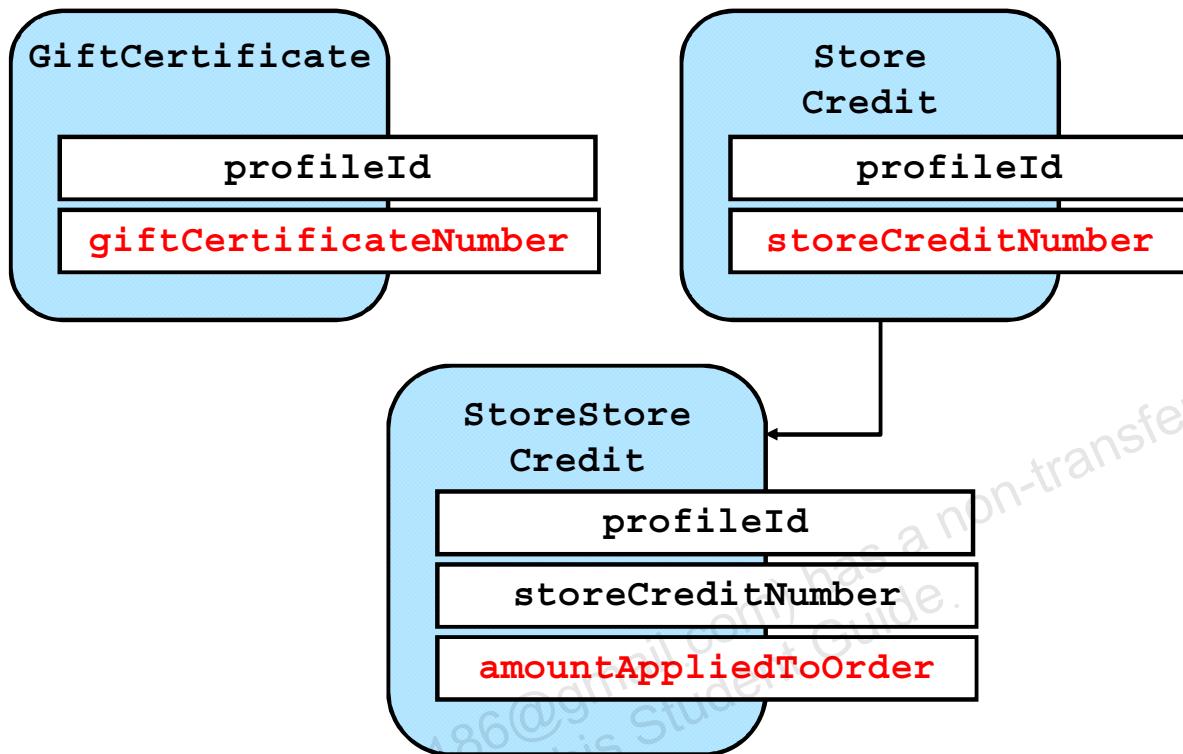


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `atg.commerce.order.CreditCard` class has the properties listed in the slide.

GiftCertificate and StoreCredit Properties



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `atg.commerce.order.GiftCertificate` and `atg.commerce.order.StoreCredit` classes have two properties: the `profileId` and the appropriate number (`giftCertificateNumber` or `storeCreditNumber`).

The `atg.projects.store.payment.StoreStoreCredit` class extends `StoreCredit` and adds the `amountAppliedToOrder` property.

Payment Manager Classes

- PaymentGroupManager has methods to:
 - Create and initialize payment groups
 - Add payment groups to orders
 - Remove payment groups from orders
- CommerceItemManager and OrderManager contain methods for allocating item or order amounts to payment groups.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

- Collecting payment information
- Payment groups and payment managers
- **Claimable payment groups**
- Payment form handlers
- Processing payments
- Adding a custom payment group



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

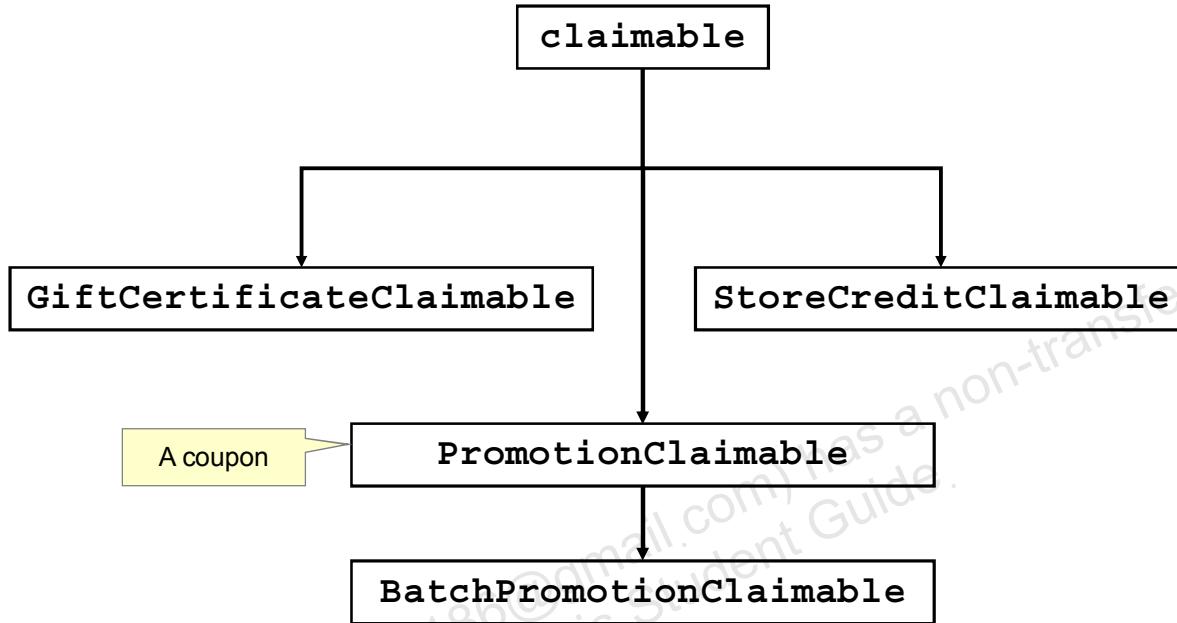
Claimable Payment Groups

- Gift certificates and store credits are “claimable” payment types.
 - Both store an amount that can be claimed in full or in part, either by a gift certificate recipient or a customer with credit.
- Claimable items are stored in the Claimable Repository.
- ClaimableManager and ClaimableTools contain methods for working with claimable repository items.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Claimable Repository Item Descriptors

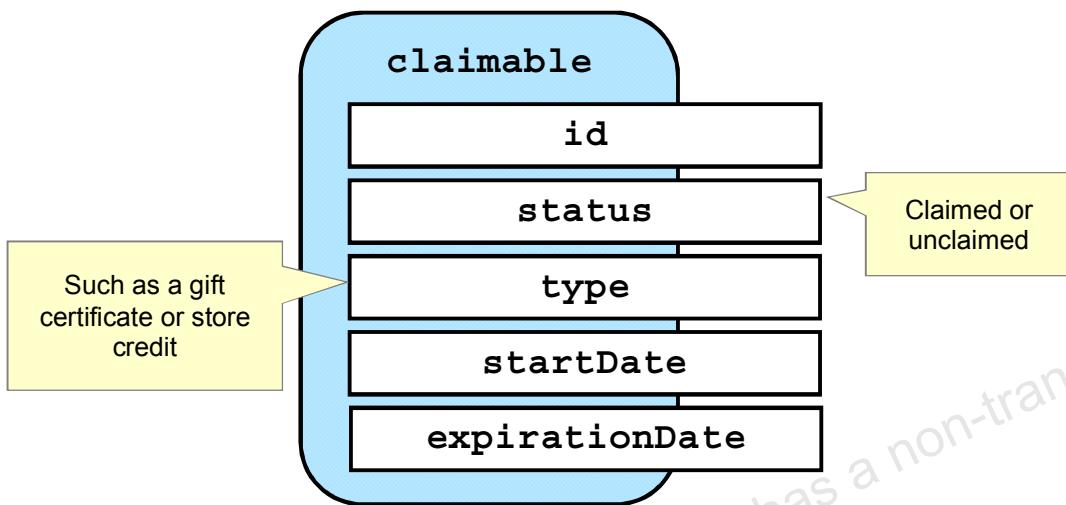


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `/atg/commerce/claimable/ClaimableRepository` has several item descriptors, including the ones shown in the slide. Coupons are also a type of claimable and are defined in the `PromotionClaimable` item descriptor. Coupons are a way of delivering promotions to a user by providing a code for the user to enter in order to claim the promotion. `BatchPromotionClaimable` is a subtype of `PromotionClaimable` and adds properties related to a batch of coupons.

Claimable Item Descriptor



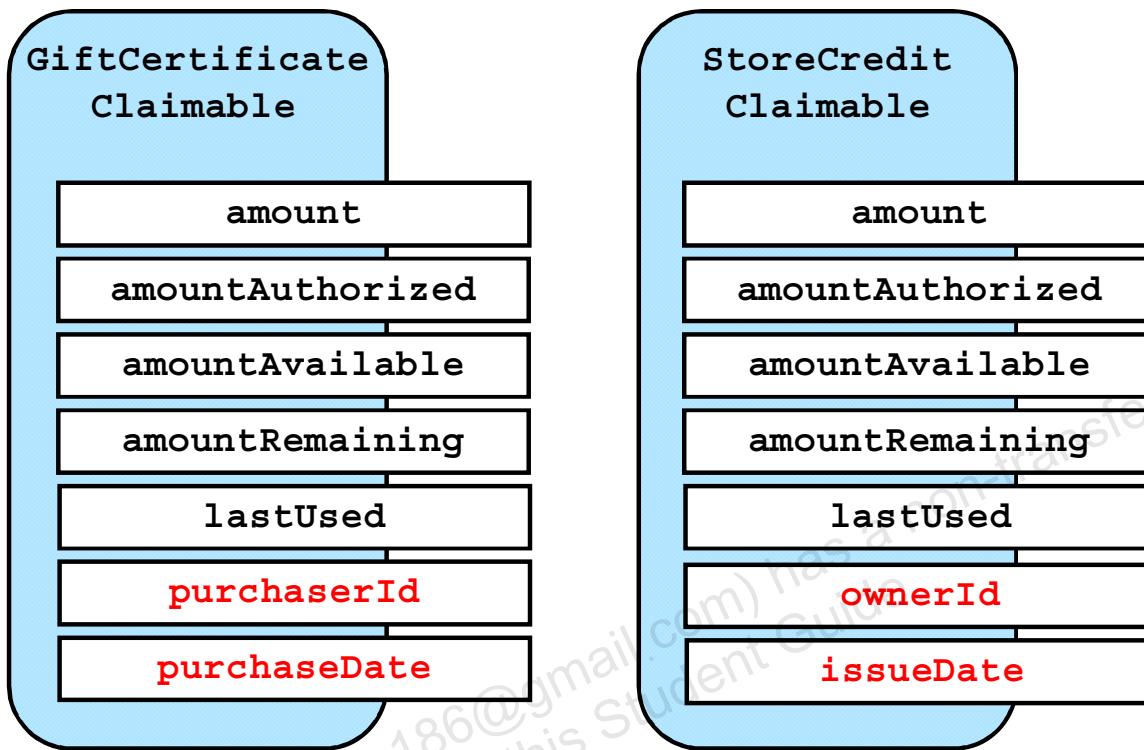
ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Valid claimable types defined out-of-the-box are:

- GiftCertificateClaimable
- PromotionClaimable (**coupon**)
- StoreCreditClaimable
- DeployablePromotionClaimable
- CouponBatch (**the batch of coupons**)
- BatchPromotionClaimable (**an individual coupon in a batch**)

GiftCertificateClaimable and StoreCreditClaimable Item Descriptors



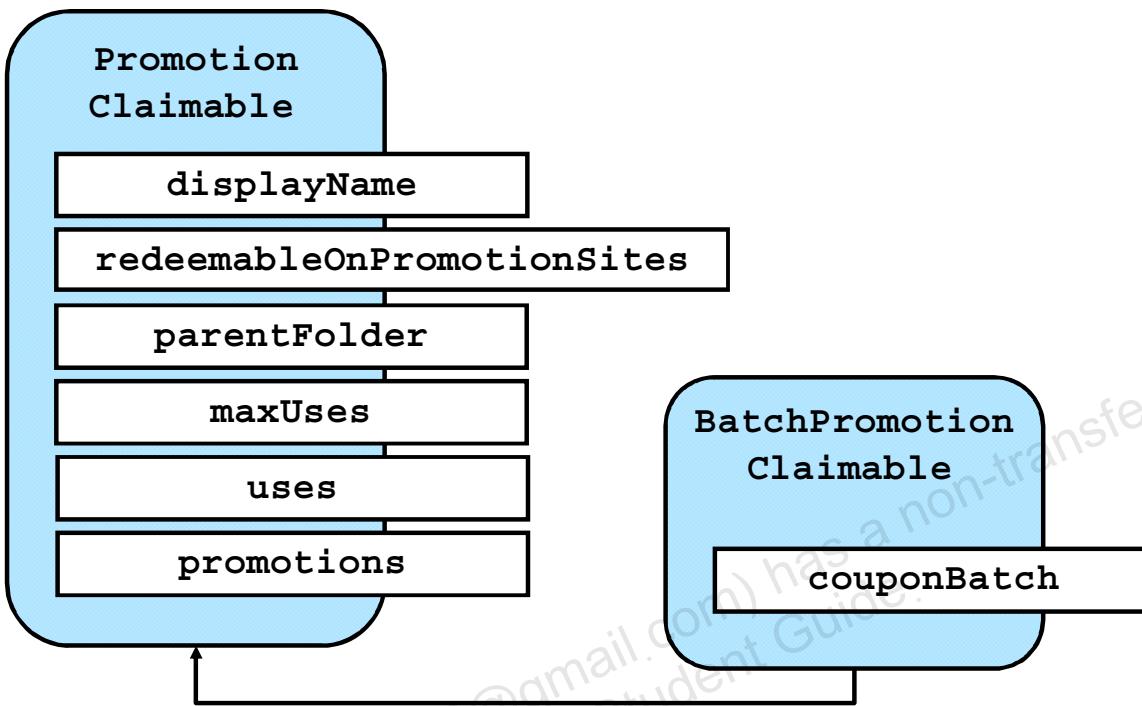
ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The gift certificate and store credit claimable item descriptors share the amount-related and date properties, but then define two of their own properties, as shown in the slide. The properties include:

- amount: The original amount of the credit
- amountAuthorized: The amount of money that has been authorized for debit. In other words, this is the amount that has already been used on other orders.
- amountRemaining: The current amount of the credit remaining for debit (the amount minus the amountAuthorized)
- lastUsed: The date the credit was last accessed
- purchaserId: The ID of the person who purchased the gift certificate
- purchaseDate: The date the gift certificate was purchased
- ownerId: The ID of the user or organization for which the store credit was issued
- issueDate: The date the store credit was issued

The PromotionClaimable Item Descriptor and the BatchPromotionClaimable Sub-Type



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Properties of the PromotionClaimable item descriptor include:

- displayName: The name of the coupon, which is also used as the claim code
- redeemableOnPromotionSites: The sites on which this coupon is valid
- parentFolder: The content folder in which this coupon appears in the BCC (a way of organizing coupons for the business user)
- maxUses: The maximum number of times that a person can claim a coupon. If maxUses is -1, the coupon can be claimed an unlimited number of times.
- uses: A property that increments when a customer tries to claim a coupon. If uses is greater than maxUses, the coupon has no more remaining uses and cannot be claimed anymore.
- promotions: One or more promotions that are associated with (granted by) this coupon.

A BatchPromotionClaimable claimable item type is created when a coupon is claimed from a coupon batch. The item descriptor extends the PromotionClaimable item descriptor and adds an additional couponBatch property. The couponBatch property is a reference to the CouponBatch that created the BatchPromotionClaimable.

Road Map

- Collecting payment information
- Payment groups and payment managers
- Claimable payment groups
- Payment form handlers
- Processing payments
- Adding a custom payment group



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Adding New Payment Groups

- Two default form handlers allow users to add new payment groups to an order:
 - CreateCreditCardFormHandler
 - CreateInvoiceRequestFormHandler
- The CRS extension is BillingInfoFormHandler.
 - Allows shoppers to add both credit cards and gift certificates
- All extend PurchaseProcessFormHandler.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The base Commerce product, without Commerce Reference Store, does not include a form handler that allows shoppers to add gift certificate payment groups to an order. Without CRS, the ShoppingCartModifier component must be used. This component is based on the SoftGoodFormHandler class. Use of this class is ordinarily deprecated.

Store credits do not have an associated form handler because shoppers are not expected to add them manually to an order. Store credit payment groups will be added to the order based on the store credits associated with the shopper's profile.

CreateCreditCardFormHandler and CreateInvoiceRequestFormHandler are the names for both the class and the component. The components are found in the /atg/commerce/order/purchase/ namespace and the classes in the atg.commerce.order.purchase package.

BillingInfoFormHandler and its corresponding BillingFormHandler component were discussed earlier in the lesson.

Changing Allocated Amounts

- Some stores may allow shoppers to split payment manually, similar to splitting shipping.
- Example:
 - The shopper has a \$100 gift certificate and a credit card.
 - Initially, the entire order amount is allocated to the credit card.
 - The shopper alters allocation so that \$50 of the order amount is paid by gift certificate and the remainder by credit card.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Another example would arise in B2B stores where shoppers may be using more than one purchase order to pay for a given order.

PaymentGroupFormHandler

- PaymentGroupFormHandler is used to create a form that allows shoppers to alter payment allocations.
 - Has a currentList property that points to current CommerceIdentifierPaymentInfos (CIPIs) items
- Three handle methods:
 - specifyDefaultPaymentGroup: Allows the user to specify the payment group to cover amounts not linked to other groups
 - splitPaymentInfos: Re-allocates amounts between CIPIs
 - applyPaymentGroups: Applies changes to the Order object

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

PaymentGroupFormHandler Example

- Splitting payment groups using PaymentGroupFormHandler is not exposed in CRS.
- Example JSPs can be found in the Sample Catalog in DCSSampleCatalog\j2ee-apps\sampleCatalog.ear\sampleCatalog.war\complex_billing.jsp.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

- Collecting payment information
- Payment groups and payment managers
- Claimable payment groups
- Payment form handlers
- Processing payments
- Adding a custom payment group



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Payment Process

- Payment happens in two stages:
 1. Authorization.
 - Verifies that the needed amount is available and, if necessary, reserves that amount
 2. Debit.
 - Is done when items are shipped
 - Is handled by the Fulfillment module out of the box
- Both operations are handled by PaymentManager.

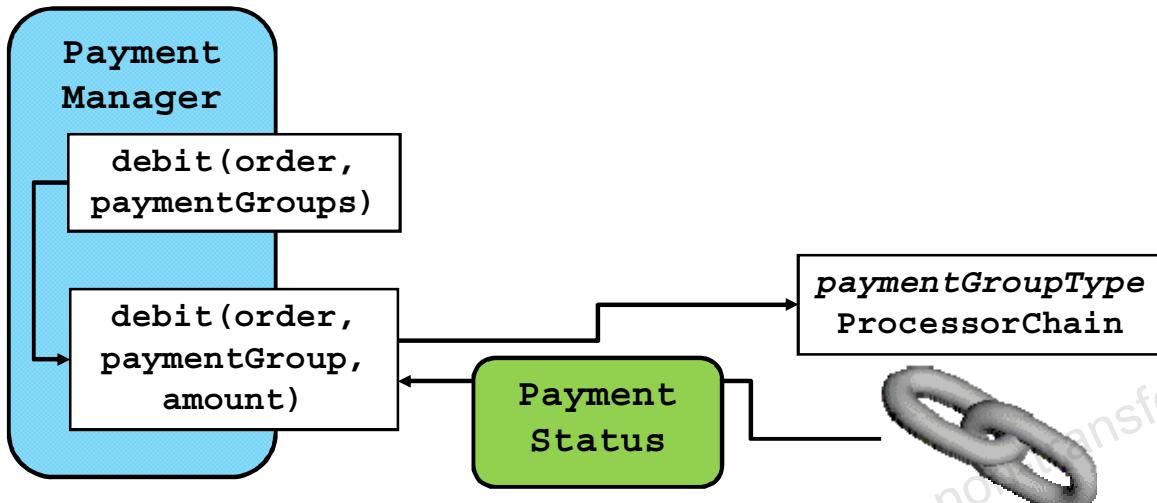


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The authorization stage is done after an order is submitted. The debit stage is done when the items are shipped, and is handled by the Fulfillment module out of the box.

PaymentManager also handles credit operations when those are necessary, for instance, when an item is returned.

Payment Operation Sequence



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

This slide illustrates the sequence of methods and objects used when a payment group is debited. The same sequence applies for authorizations and credits.

The `debit(order, paymentGroups)` method of `PaymentManager` is called. This method calls the `debit(order, paymentGroup, amount)` method for each payment group in the order, including the amount to be debited.

The debit method uses the `PaymentManager.paymentGroupToChainMap` property to determine which pipeline chain to use for a particular type of payment group. It then uses `PaymentPipelineManager` to invoke this chain. The pipeline chain returns a `PaymentStatus` object which indicates the result of the attempted payment operation.

Road Map

- Collecting Payment Information
- Payment groups and payment managers
- Claimable payment groups
- Payment form handlers
- Processing payments
- Adding a custom payment group



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Steps to Add a Custom Payment Group Type

1. Create a payment group class that extends `PaymentGroupImpl` or one of the existing payment group classes.
2. Create an item descriptor in the Order Repository that is a subtype of `paymentGroup`, or one of its subtypes.
3. Edit the `OrderTools` component:
 - Add the new item descriptor to `beanNameToItemDescriptorMap`.
 - Add the class to `paymentTypeClassMap`.
4. Create an initializer class that implements `PaymentGroupInitializer`.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

An example of an address class to map the shipping address to is `atg.commerce.order.RepositoryContactInfo`.

Steps to Add a Custom Payment Group Type

5. Create an initializer component and add it to PaymentGroupDroplet's paymentGroupInitializers property.
6. Extend pipelines:
 - Create a pipeline processor to authorize, debit, and credit the payment group.
 - Configure a new pipeline with that processor.
 - Add the pipeline to PaymentManager.paymentGroupToChainMap.
 - Create a pipeline processor to validate the payment group for checkout.
 - Add the validation processor to the validatePaymentGroup and validatePaymentGroupPreConfirmation pipelines.

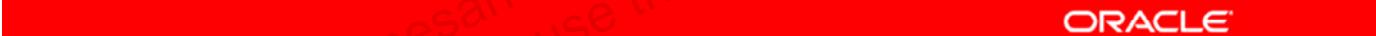


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Extending pipelines was covered in detail in the lesson titled “Pipelines and Shipping Methods.” The step to extend pipelines is an integration point.

Steps to Add a Custom Payment Group Type

7. If shoppers will be adding these groups to their order manually, create a form handler that extends PurchaseProcessFormHandler.

A solid red horizontal bar spanning most of the page width.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe the default and CRS payment collection processes
- Use the four provided payment group types
- Add custom payment group types
- Describe and extend the processing of payments for completed orders

For More Information

ATG Commerce Guide to Setting Up a Store

- Implementing Order Retrieval
- Adding Payment Information to Shopping Carts

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Practice 17 Overview: Creating a Custom Payment Group Type

This practice covers the following topics:

- Creating a custom payment method to pay for reward orders with points and an initializer
- Extending `BillingInfoFormHandler` and `BillingProcessHelper` to initialize and allocate order amounts to points payment groups
- Extending the payment group validation pipelines to validate points payment groups and the payment pipeline to award points after order has been shipped



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

18

Other Purchase Process Form Handlers



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to use default form handlers in JSPs that allow shoppers to:

- Cancel orders
- Submit orders
- Use express checkout
- Save incomplete orders
- Claim coupons

Road Map

- Cancelling and committing orders
- Express checkout
- Saving incomplete orders
- Claiming coupons



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

CancelOrderFormHandler

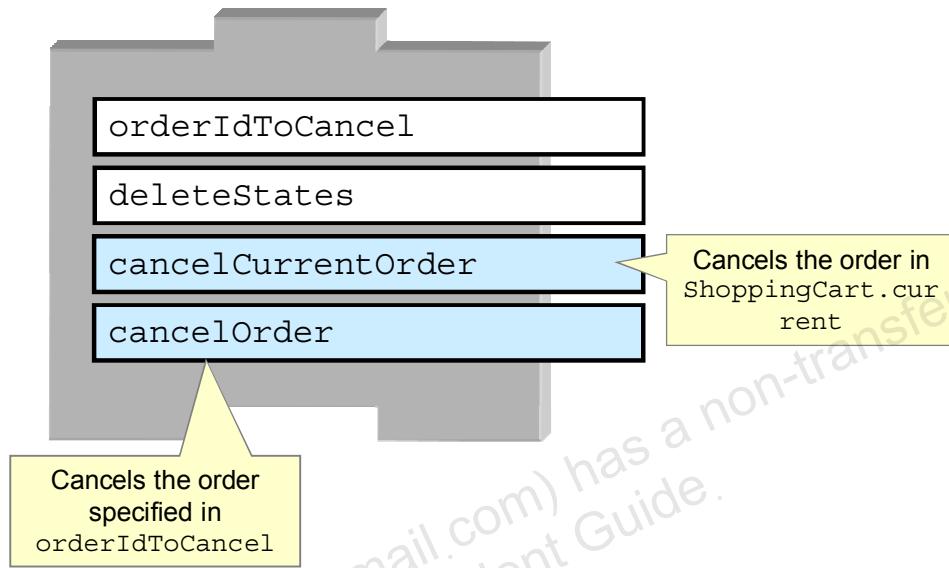
CancelOrderFormHandler:

- Is used on any purchase process page to allow the user to cancel the order
- Clears the Order object and deletes the Order from the repository
- May also be used after the Order has been submitted, if permitted
 - You can configure which order states allow cancellation.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

CancelOrderFormHandler: Properties and Handlers



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Key properties of the `/atg/commerce/order/purchase/CancelOrderFormHandler` include:

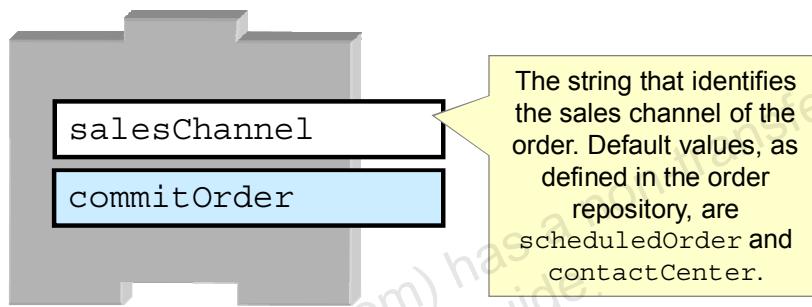
- `orderIdToCancel`: This is an optional property that is set in the form in the JSP. It is used in conjunction with the `cancelOrder` method.
- `deleteStates`: This property contains the list of order states that allow cancellation. By default, it is set to the `/atg/commerce/order/CancelOrderService` component, which has `deleteStates= INCOMPLETE, PENDING_APPROVAL`. `CancelOrderFormHandler` has a similar property called `ignoreStates` that is configured the same way. It relates to the `CancelOrderService.ignoreStates` property (set to `PENDING_REMOVE`).

The component is based on the `atg.commerce.order.purchase.CancelOrderFormHandler` class, which extends `atg.commerce.order.purchase.PurchaseProcessFormHandler`.

CommitOrderFormHandler

CommitOrderFormHandler is used on the confirmation page to place the order.

- Calls OrderManager.processOrder()
- Sets ShoppingCart.current to null



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The /atg/commerce/order/purchase/CommitOrderFormHandler submits the user's current Order for checkout. Its handleCommitOrder() method (shown as the handler "commitOrder" in the slide) ensures that the user is not trying to double-submit the order by checking if the ID of the current Order is equal to the ID of the user's last Order (in ShoppingCart.last). If the IDs are not equal, the current Order can be submitted. The handle method then calls OrderManager.processOrder(), which executes the processOrder pipeline.

The salesChannel property returns the sales channel which is used to submit the order. The default values are scheduledOrder and contactCenter, and are defined in the order repository. Additional properties on the form handler include orderId, allowEmptyOrders (set to true by default), siteID, and error and success URL properties. To see the full list of properties, see the atg.order.purchase.CommitOrderFormHandler class in the *API Platform Reference*. The class extends atg.commerce.order.purchase.PurchaseProcessFormHandler.

Road Map

- Cancelling and committing orders
- Express checkout
- Saving incomplete orders
- Claiming coupons



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ExpressCheckoutFormHandler

- ExpressCheckoutFormHandler provides simple and quick checkout functionality.
 - Allows a single hardgood shipping group and a single credit card payment group to be added to the Order.
 - Can be used for:
 - “One-click” checkout for registered users with default shipping and billing information
 - A single purchase process page that combines both shipping and payment gathering
- The CRS extension is StoreExpressCheckoutFormHandler, which gets the billing address from the credit card instead of the user's profile.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path for ExpressCheckoutFormHandler is /atg/commerce/order/purchase/ExpressCheckoutFormHandler, which is based on the atg.commerce.order.purchase.ExpressCheckoutFormHandler class (extends PurchaseProcessFormHandler). The ExpressCheckoutFormHandler is used to expedite the checking out of an Order. This supports creating a maximum of one Profile derived HardgoodShippingGroup and one Profile derived CreditCard, followed by committing the Order.

The class for the CRS extension is

atg.projects.store.order.purchase.StoreExpressCheckoutFormHandler.

For more information about the properties and handle method, see the classes in the *API Platform Reference Guide* and the *API Reference* (under the CRS documentation set).

Road Map

- Cancelling and committing orders
- Express checkout
- Saving incomplete orders
- Claiming coupons



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Saving Incomplete Orders

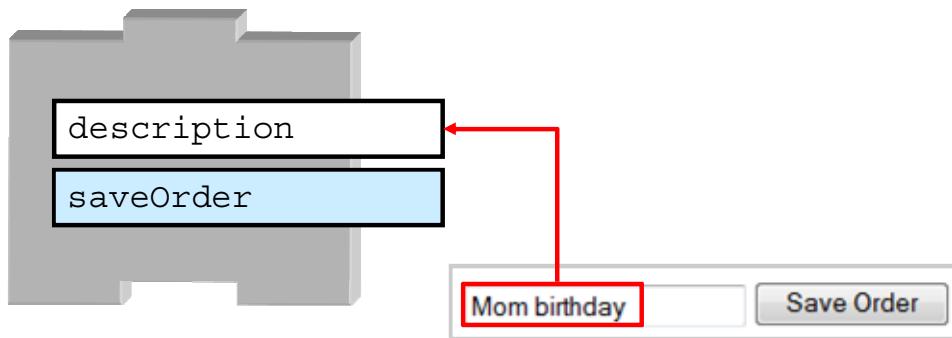
- Stores may allow customers to explicitly save incomplete orders.
 - Shoppers can have many saved orders.
 - Only one order can be active at a time.
- Orders can be saved at any stage of the purchase process, and will store any shipping and billing information provided so far.
- Your store must provide pages to save and reactivate orders.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Keep in mind that the current order for a registered user is automatically saved, by default. The feature discussed in the slide allows shoppers to *explicitly* save orders.

SaveOrderFormHandler



- The Order is moved from ShoppingCart.current to ShoppingCart.saved.
- In the Order Repository, the order's explicitlySaved property is set to true.
 - This prevents the order from being marked as abandoned.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full name of the component is /atg/commerce/order/purchase/SaveOrderFormHandler. It is based on the atg.commerce.order.purchase.SaveOrderFormHandler class, which extends PurchaseProcessFormHandler. The description property is used to specify the identifying name of the order for the shopper's use. In the example in the slide, the shopper wants to name the order "Mom birthday." That will be the identifier for the order when the shopper wants to see the list of saved orders. If the shopper does not supply a description, one will be created based on the user's locale and the date and time.

ShoppingCart Component

- ShoppingCart is used to “revive” a saved order.
- Key properties of the ShoppingCart component for saved orders are:
 - saved
 - handlerOrderId
- Handlers include:
 - switch
 - delete

The ShoppingCart component can also act as a form handler.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Key properties of the ShoppingCart component include:

- saved: A collection of Order objects that are the user’s saved list of shopping carts (orders)
- handlerOrderId: The property that identifies the order

Handlers include:

- switch: Takes the order ID as specified in the handlerOrderId property and makes that the current order by invoking the switchCurrentOrder() method
- delete: Takes the order ID as specified in the handlerOrderId property and deletes that order by invoking the deleteOrder method

The /atg/commerce/ShoppingCart component is an instance of the atg.commerce.order.OrderHolder class.

Road Map

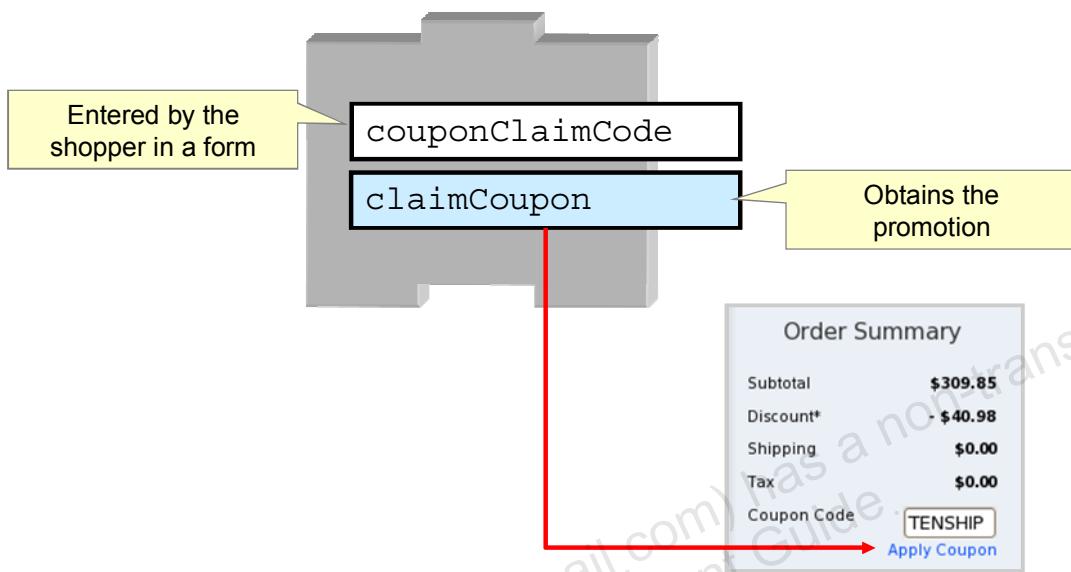
- Cancelling and committing orders
- Express checkout
- Saving incomplete orders
- Claiming coupons



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

CouponFormHandler



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Although /atg/commerce/promotion/CouponFormHandler does not extend PurchaseProcessFormHandler, it is often used on the shopping cart page, and affects the pricing of an order. This form handler is used to “claim” a promotion and then add this promotion to the users’ current list of promotions. A coupon is just another mechanism to deliver a promotion to the user, by having the customer explicitly enter the coupon claim number.

The handleClaimCoupon() method is responsible for actually doing the work and obtaining the promotion.

CouponFormHandler is based on the atg.commerce.promotion.CouponFormHandler class, which extends GenericFormHandler.

Summary

In this lesson, you should have learned how to use helpful form handlers in your JSPs to:

- Cancel or commit orders
- Complete an order using express checkout
- Save incomplete orders
- Claim coupons

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable
license to use this Student Guide.

Inventory and Fulfillment

19

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to describe:

- ATG's default inventory management and fulfillment systems
- The options for replacing the default systems

Road Map

- Inventory
- Fulfillment



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Inventory

- ATG Commerce expects an Inventory Manager component.
 - Based on the class that implements the `InventoryManager` interface
- By default, stock is “picked” and reserved after checkout when the order reaches the Fulfillment system.
- By default, inventory levels are stored separately from the rest of the catalog data.
- `InventoryDroplet` can be used to query inventory for display to shoppers.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The Fulfillment system will be covered in more detail in a later section.

The full path to the `InventoryManager` interface is
`atg.commerce.inventory.InventoryManager`.

The full path to `InventoryDroplet` is
`/atg/commerce/inventory/InventoryDroplet`.

Inventory Manager

The InventoryManager interface defines methods to:

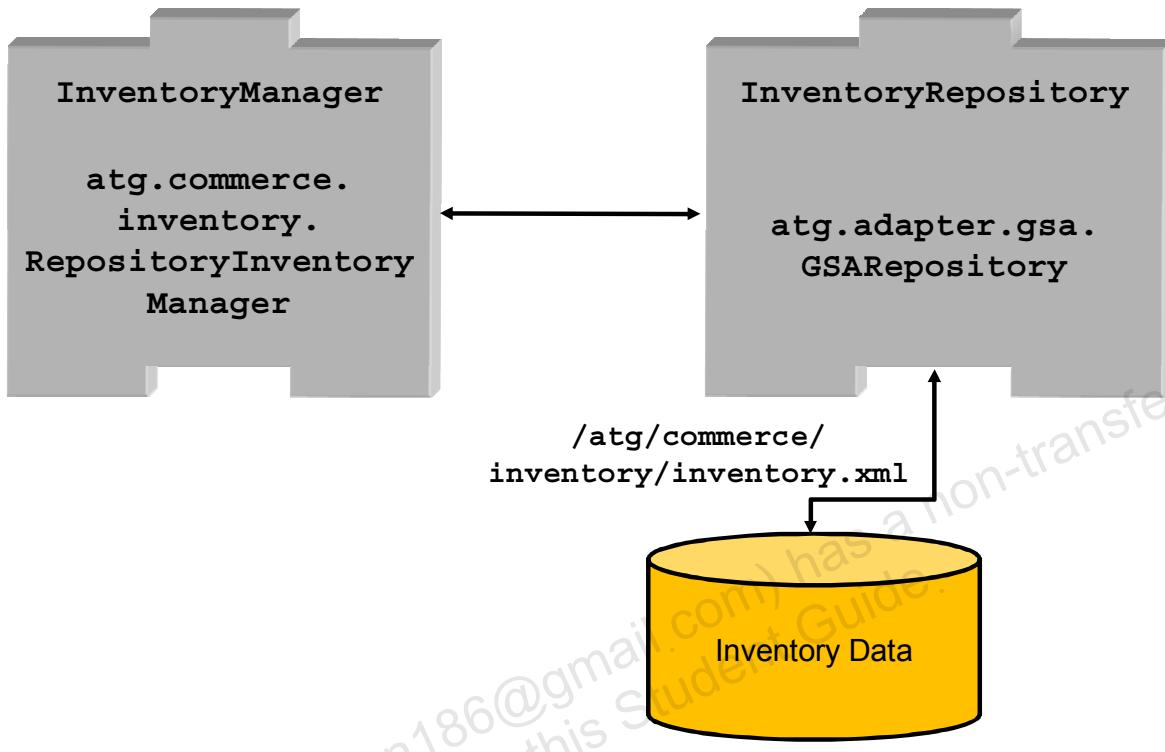
- Query, set, increase, and decrease regular, preorder, and backorder stock levels
- Query and set the availability status and availability date of an item
- Purchase items from regular, backorder, or preorder stock
- Notify other systems of inventory changes



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The purchase method is called when items are purchased from regular stock.

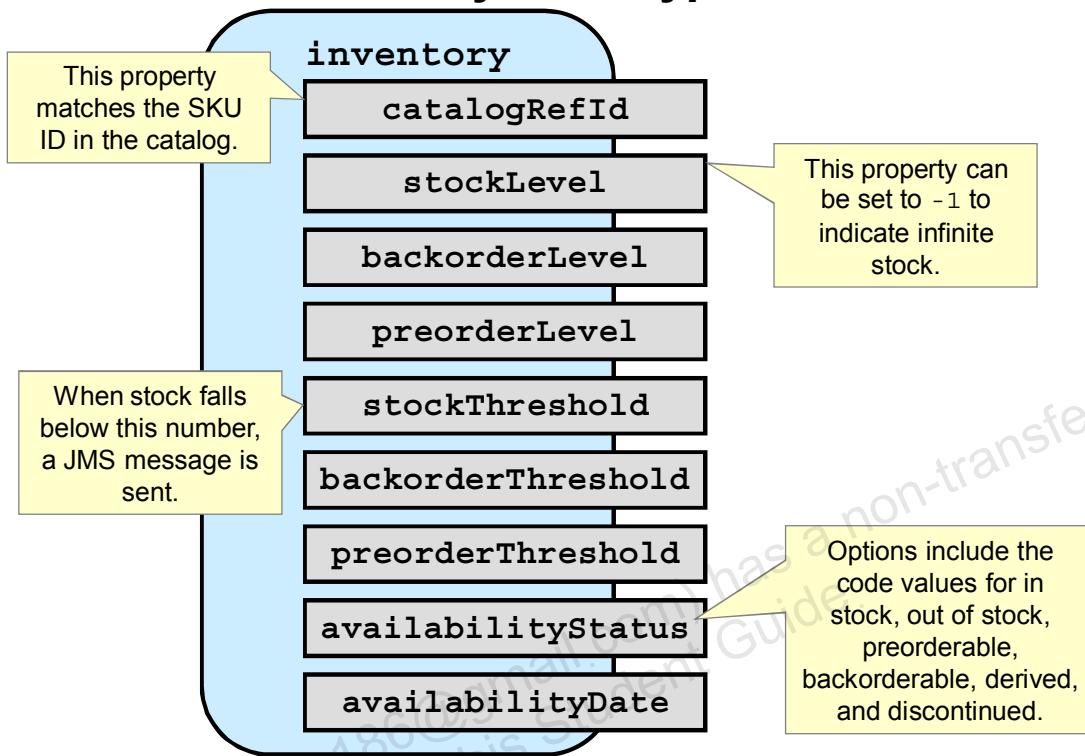
Default Inventory Implementation: Architecture



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Default Inventory Implementation: Inventory Item Type

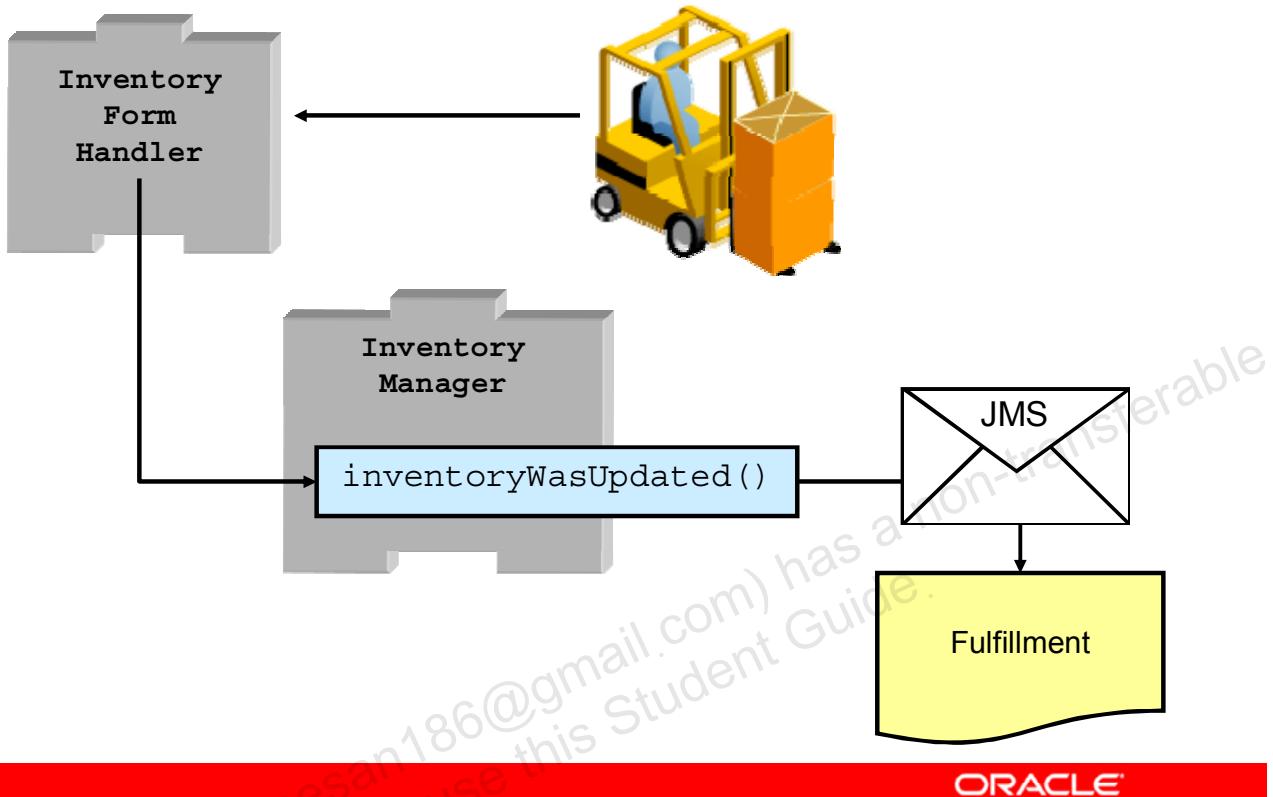


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Stores do not need to create inventory objects for SKU bundles. By default, `RepositoryInventoryManager` will use the individual SKUs' inventory levels to calculate the stock level and availability of a bundle. If any one of the SKUs in the bundle is out of stock, the entire bundle is considered out of stock.

Default Inventory Implementation: Updating Inventory



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When any part of the ATG application updates the inventory numbers outside of a purchase (such as when stock is returned or new stock arrives), the code that processes the change (in the example in the slide, the `InventoryFormHandler`) should call the `inventoryWasUpdated` method of the `Inventory Manager` component. By default, this method sends a JMS message to the `Fulfillment` system alerting it to the availability of new stock. The `Fulfillment` system will locate and process any orders that were waiting for out-of-stock items to arrive.

Customizing Inventory Management

You can customize your inventory management by choosing one of the following options:

1. Modify `inventory.xml` to represent your inventory database.
2. Model a custom inventory repository.
3. Modify the `InventoryManager` component to use one of the other provided classes (detailed later).
4. Write your own implementation of the `InventoryManager` interface.
 - For example, if inventory will be queried and modified using a web service



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Option 1: Modify `inventory.xml`

- You can override the `inventory.xml` file in your application layer to map to your database columns.
- As long as the `inventory` repository item definition remains the same, no changes are needed in ATG components or classes.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Option 2: Model a Custom Inventory Repository

- With this option, you model a custom repository representing your inventory data.
- The modeled object must include properties representing all the data shown earlier in the lesson.
 - Note: Properties do not have to have the same names.
- The names of inventory repository item properties are configured in `RepositoryInventoryManager`.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `atg.commerce.inventory.RepositoryInventoryManager` class is a repository-based implementation of `atg.commerce.inventory.InventoryManager`. It implements all the methods defined by the `InventoryManager` API. It is a thin wrapper around a repository that contains the inventory information. This allows a maximum amount of flexibility for potential third-party integrators. Integrators can simply implement a repository containing the required properties for cooperation with the `RepositoryInventoryManager`. The `RepositoryInventoryManager` can then be configured to extract inventory manager information from the third-party repository.

This class also is a message source. It can send `UpdateInventory` messages if there is new inventory available for a previously unavailable item.

The class has setter methods for many of the inventory-related properties, such as `availabilityStatus`, `availabilityDate`, `backorderLevel`, and `backorderThreshold`. For more information on the class, see the *API Platform Reference*.

Option 3: Modify the InventoryManager Class

- ATG provides several implementations of the `InventoryManager` interface:
 - `LocalizingInventoryManager`: Locale-specific inventory management
 - `RepositoryInventoryManager`: Repository-based inventory management
 - `NoInventoryManager`: No inventory management
 - `CachingInventoryManager`: Partial caching of inventory data
- All three are in the `atg.commerce.inventory` package.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

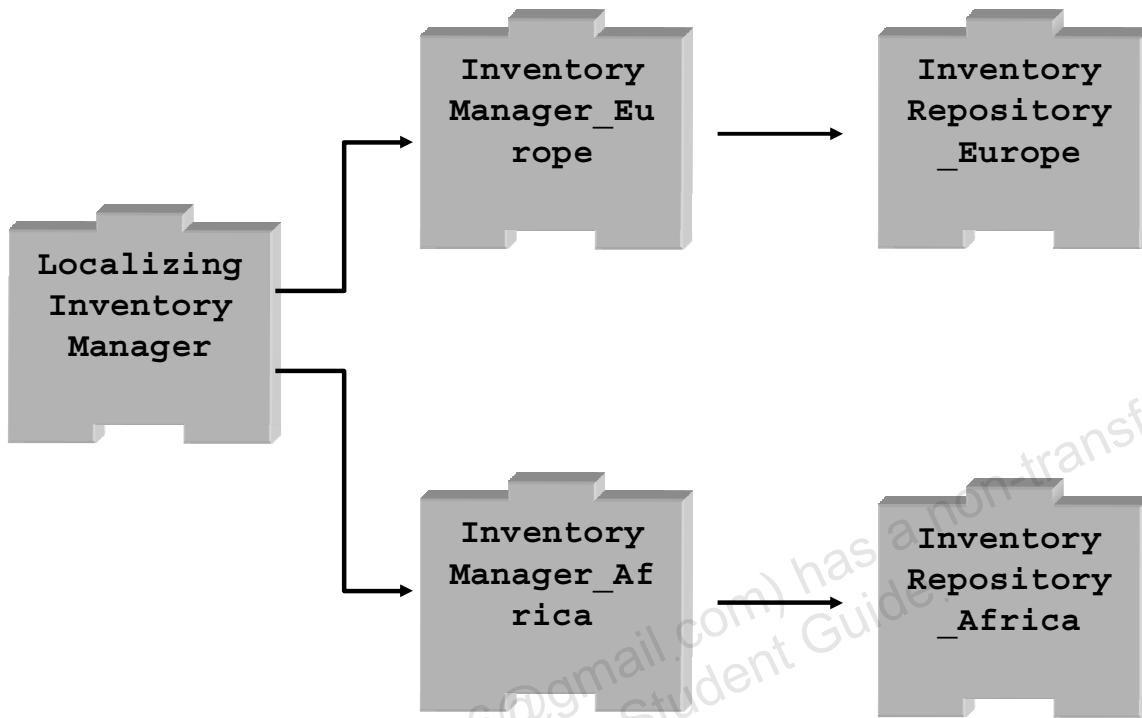
`LocalizingInventoryManager` is used when you want to have more than one set of inventory data (and therefore, more than one `InventoryManager`). It determines which data/manager set to use based on the user's locale.

`RepositoryInventoryManager` is a thin wrapper around a repository that contains the inventory information. This allows a maximum amount of flexibility for third-party integrators. You can implement a repository that contains certain required properties (discussed in the “`RepositoryInventoryManager`” section of the *ATG Commerce Programming Guide*) for cooperation with the `RepositoryInventoryManager`. The CRS extension to the class is `atg.projects.store.StoreInventoryManager`. It is responsible for writing to the inventory repository, which should only happen from the fulfillment server.

`NoInventoryManager` is useful in cases where no `InventoryManager` functionality is required, but an inventory manager of some kind is needed for a property setting. It allows components that require an `InventoryManager` to function without having an inventory system on the back end.

`CachingInventoryManager` sends cached data in response to any call of a query method, such as `queryStockLevel`. Other methods use up-to-date data. This differs from standard repository caching, which would use cached data for all purposes.

Example: Localizing Inventory Manager



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `LocalizingInventoryManager` has a list of the `InventoryManager` components and the locales that should be routed to them. When customers shop on the site, they will see the appropriate inventory data for their locale. Each `InventoryManager` component is connected to its local inventory data. In this example, the local `InventoryManager` components are based on `RepositoryInventoryManager`.

Option 4: Write a New InventoryManager Class

- Your class must implement the `InventoryManager` interface or the `AbstractInventoryManagerImpl` class.
- You must implement the `purchase()` method.
 - Other methods can be empty.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Design Decision: Inventory

- Does your organization want to maintain inventory levels in ATG Commerce?
- Will inventory levels be displayed to users?
- How will inventory levels be stored?
 - In the default repository?
 - In a custom repository?
 - In a third-party system?
- Will any customizations be needed?
 - For example, when do you want to reserve stock and when do you mark it as picked?

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Will any customizations be needed?

- For example, having stock picked and reserved before checkout

Road Map

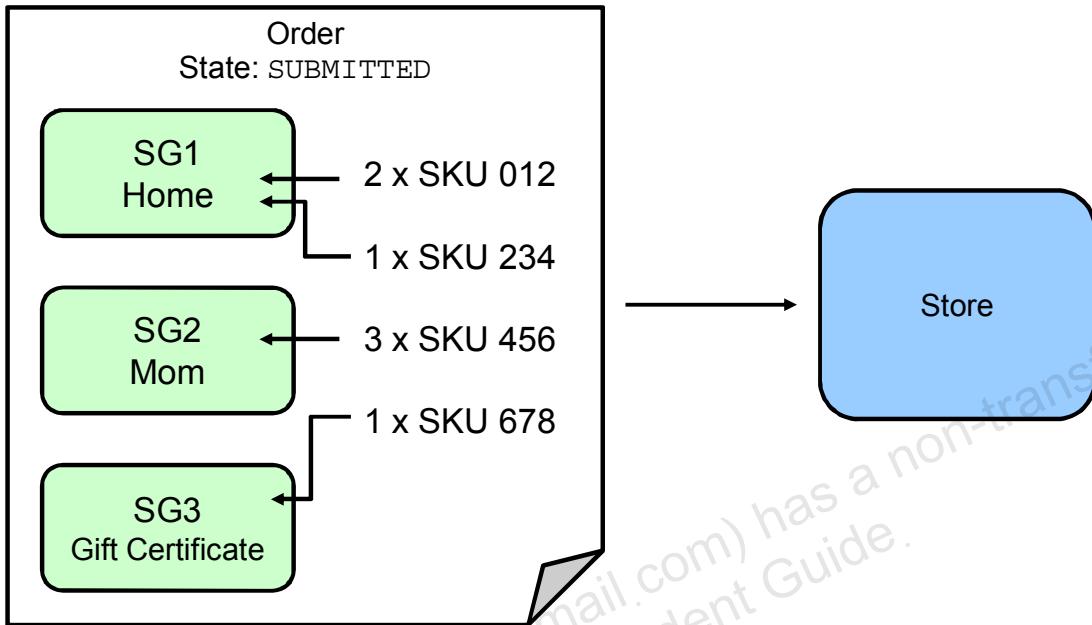
- Inventory
- Fulfillment



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Default Fulfillment Process: Submitting the Order

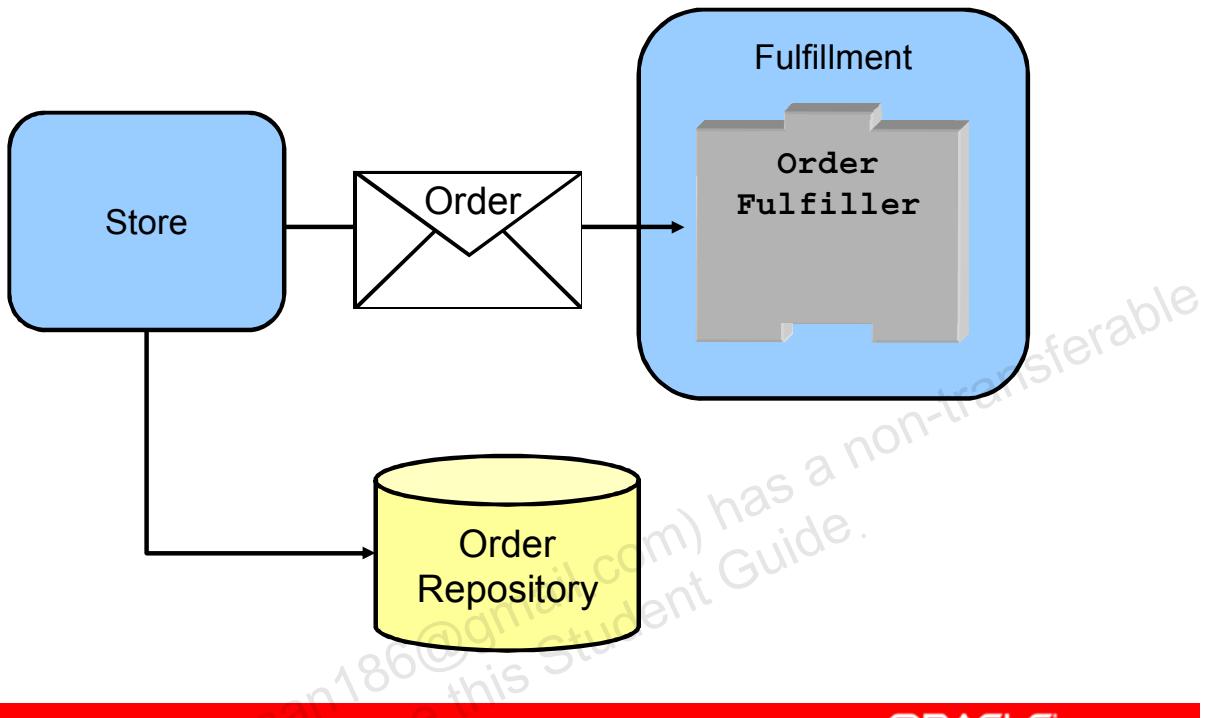


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

A new order is submitted to a store server. In the example in the slide, there are 4 SKUs. SKU 678 is a gift certificate. The other SKUs are all hard goods. The shipping has been split so that SKU 012 and SKU 234 are being shipped to the purchaser's home (SG1) and SKU 456 is going to the purchaser's mother's address (SG2). The gift certificate is a soft good, and is assigned its own shipping group (SG3).

Default Fulfillment Process: Notifying Fulfillment

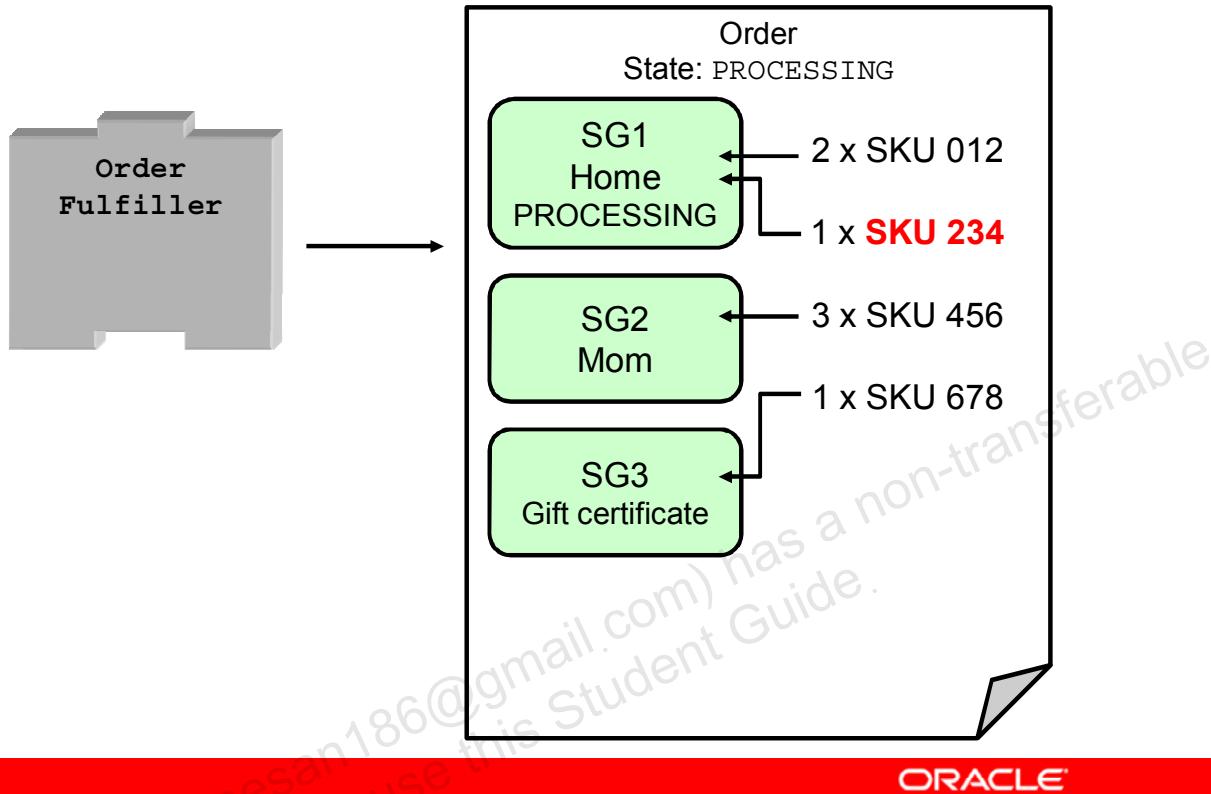


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The Store server persists the order in the Order Repository and also sends a JMS message to the OrderFulfiller component in the Fulfillment module. The Fulfillment module is fully separate from the other Commerce modules so that it can be run on a separate server, if desired. The JMS message contains the complete Order object.

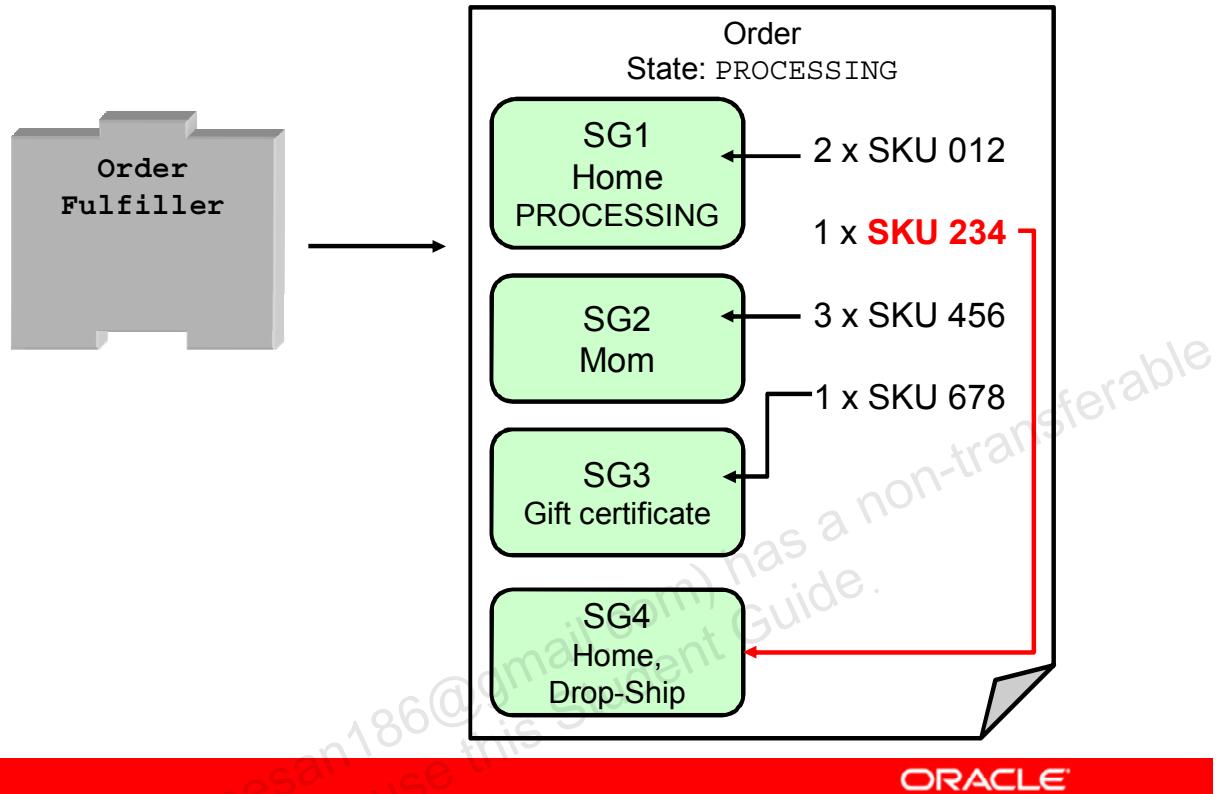
Default Fulfillment Process: Processing the Order



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The OrderFulfiller component examines and reorganizes the shipping groups, if needed. These changes are persisted in the Order Repository. In the example in the slide, SKU 012 is shipped from the store warehouse, but SKU 234 is drop-shipped from the manufacturer. SG1 needs to be split into two shipping groups. SG3 is a purchase of a gift certificate, which is a soft good. The order and the shipping groups are all marked as PROCESSING.

Default Fulfillment Process: Processing the Order

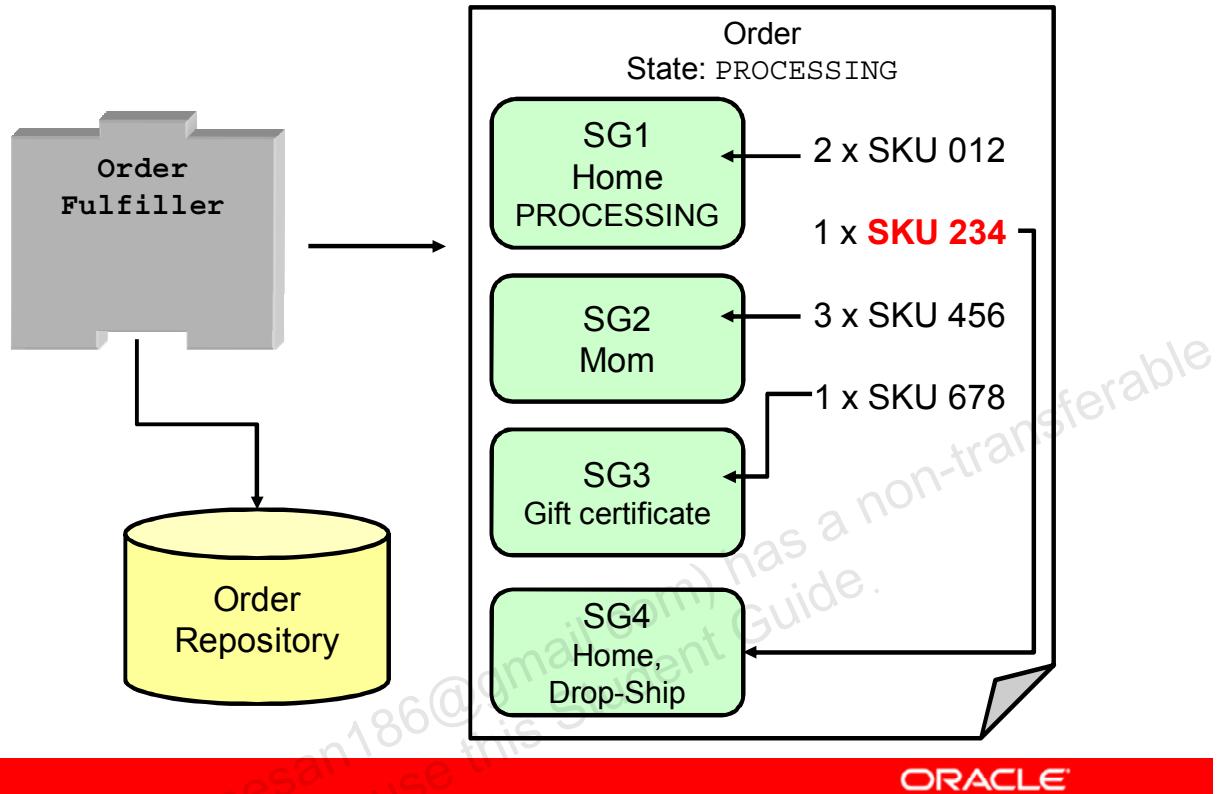


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

In the example in the slide, a new shipping group is created, SG4, to handle SKU 234, which will be drop-shipped to the purchaser's home.

Default Fulfillment Process: Processing the Order

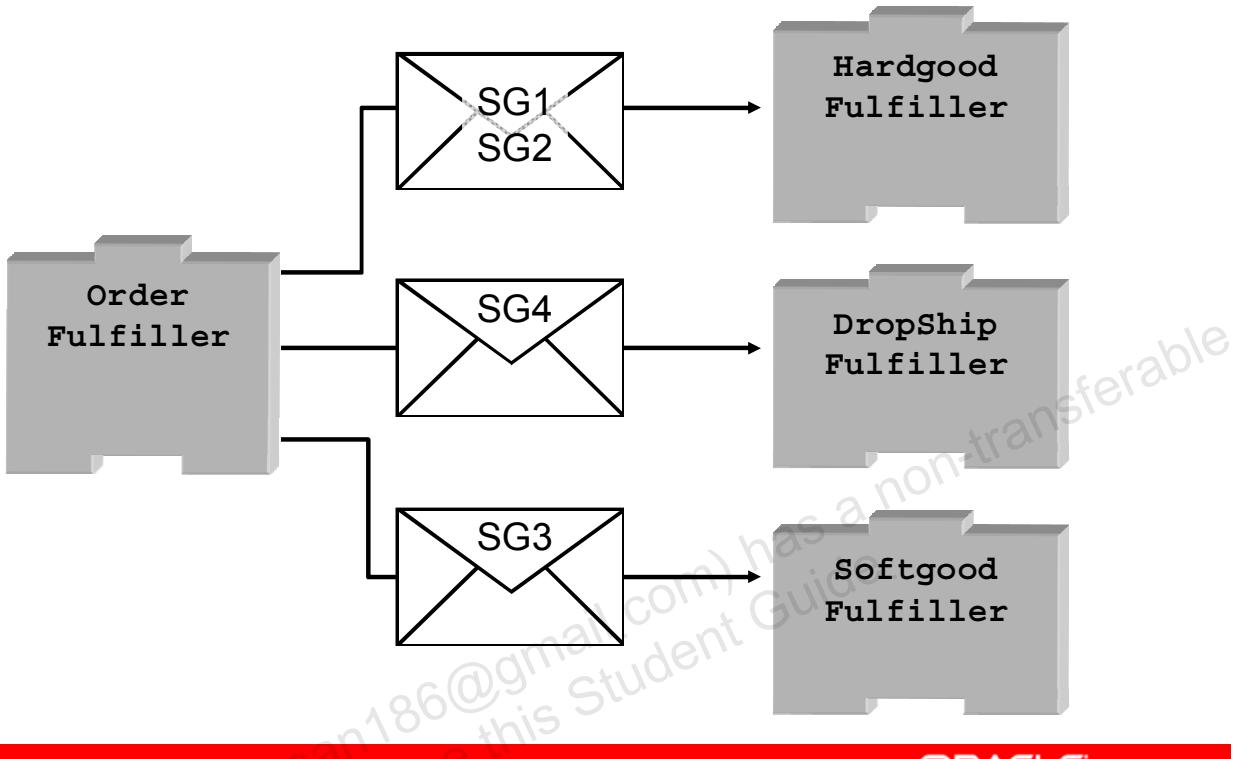


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

These changes are persisted in the Order Repository.

Default Fulfillment Process: Sending FulfillOrderFragment Messages



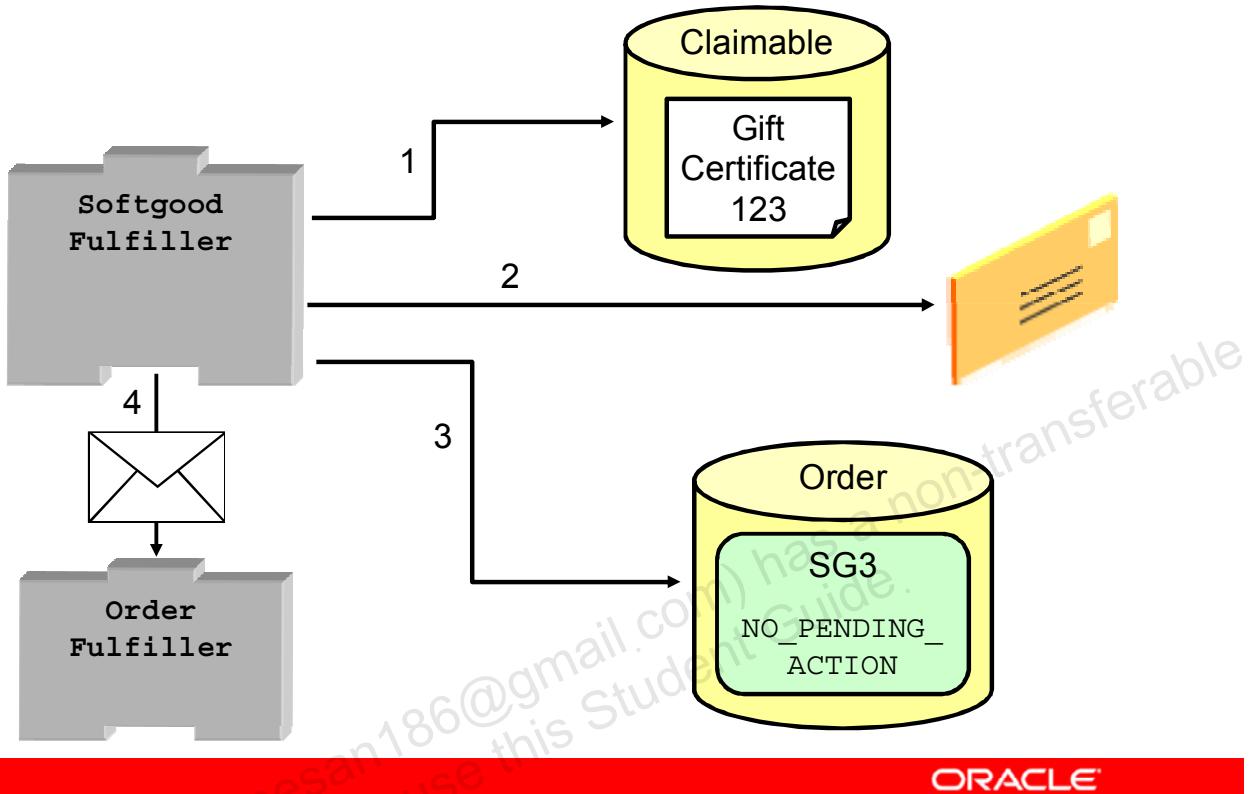
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The OrderFulfiller then sends a JMS message called a FulfillOrderFragment message to each fulfills component that will be involved in fulfilling the order. These messages contain the Order object and the list of shipping group IDs for which the fulfills is responsible.

In the example in the slide, HardgoodFulfiller and SoftgoodFufiller are out-of-the-box components. DropShipFulfiller is a custom component.

Default Soft Good Fulfillment Process



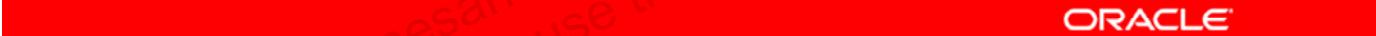
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The ElectronicFulfiller class provided out of the box is designed to fulfill purchases of gift certificates. It creates a new gift certificate item in the Claimable Repository (1), sends an email to the recipient with the new gift certificate number (2), and marks the shipping group as NO_PENDING_ACTION (3). Lastly, it sends a ModifyOrderNotification message to the OrderFulfiller component to notify it that the shipping group has shipped. This message is also used to notify the scenario engine.

Design Decision: Soft Goods

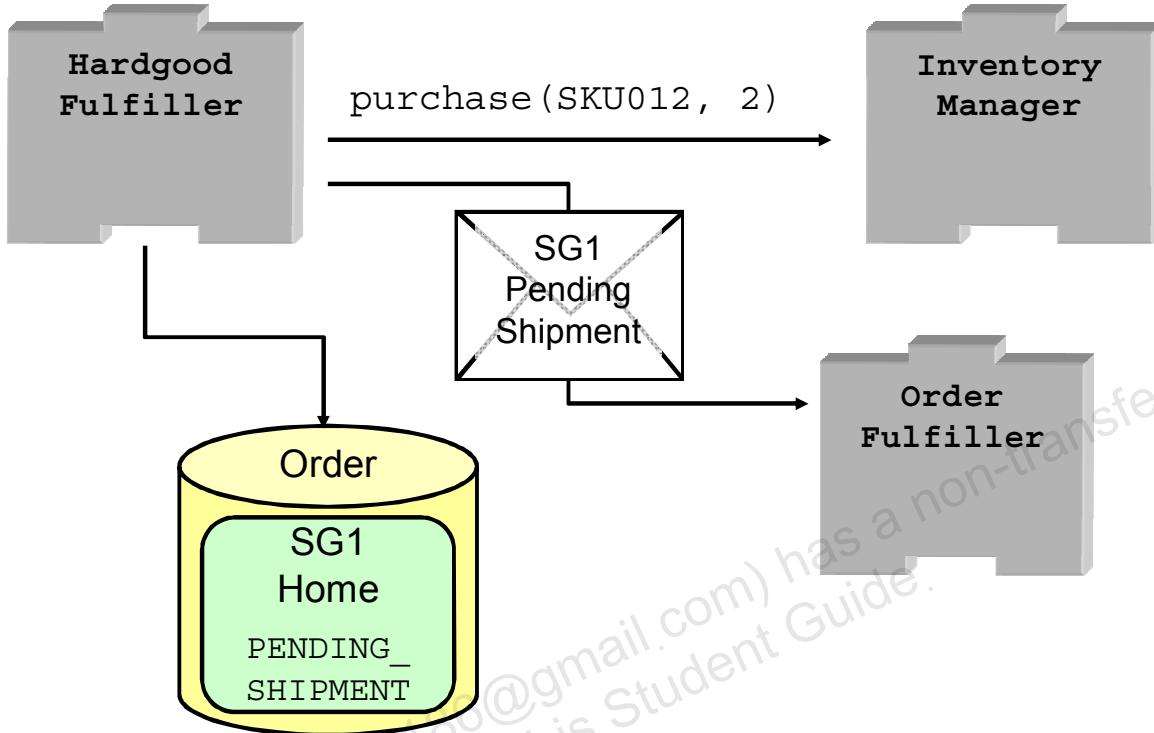
- Does your organization sell “soft” goods?
- How will those soft goods be “delivered?”
 - By email?
 - By download?
- What customizations will be needed to support soft good fulfillment?
 - For example, custom profile properties



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Default Hardgood Fulfillment Process: Picking Stock



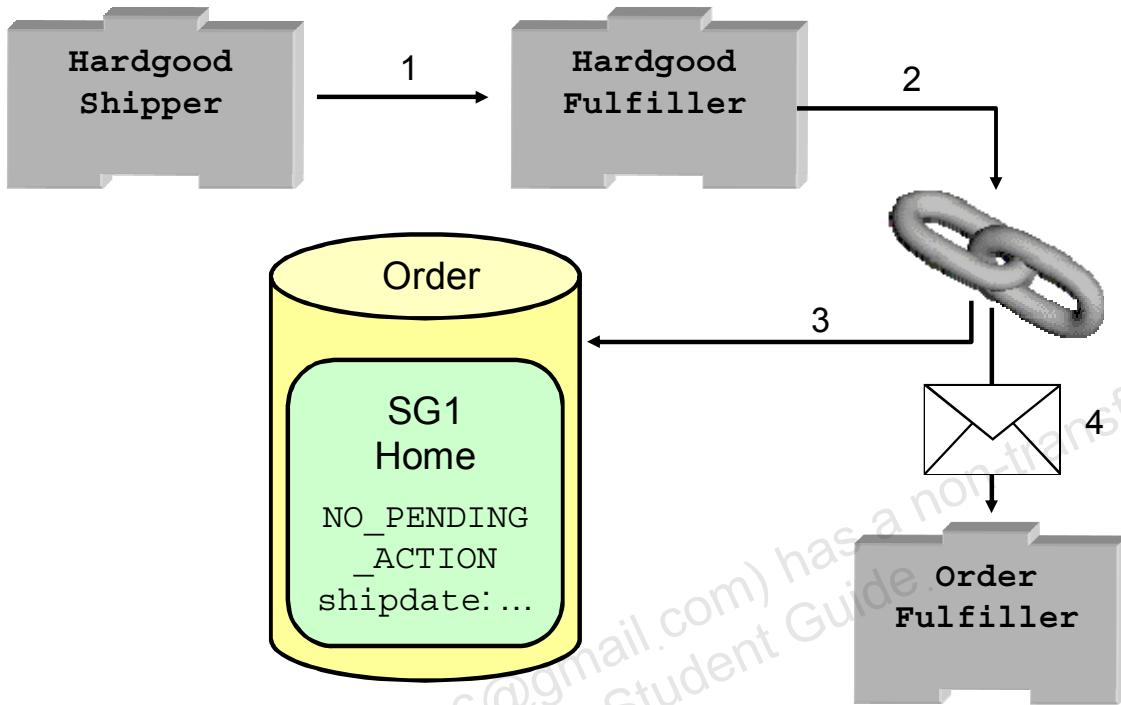
ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The main job of a `HardgoodFulfiller` component is to pick and reserve stock. Depending on the status of the item, it will call the `purchase()`, `purchaseOffBackorder()`, or `purchaseOffPreorder()` method of `InventoryManager`. If there is not enough stock to fulfill a shipping group, the fulfills component will either hold the entire group until it is notified that the needed item is back in stock, or ship the items that are in stock separately. This behavior is determined by the `allowPartialShipment` property of the fulfills.

After stock has been picked, if all the items in the shipping group are in stock, the fulfills component modifies the shipping group's state to `PENDING_SHIPMENT` and sends a JMS message reflecting the change.

Default Hardgood Fulfillment Process: Shipping a Shipping Group



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

After the shipping group has been shipped out, the **OrderFulfiller** needs to be notified of the shipment. **Note:** Depending on how you have integrated your shipping system with the Fulfillment module, not all of these steps may occur. Integrating with your shipping system will be discussed shortly.

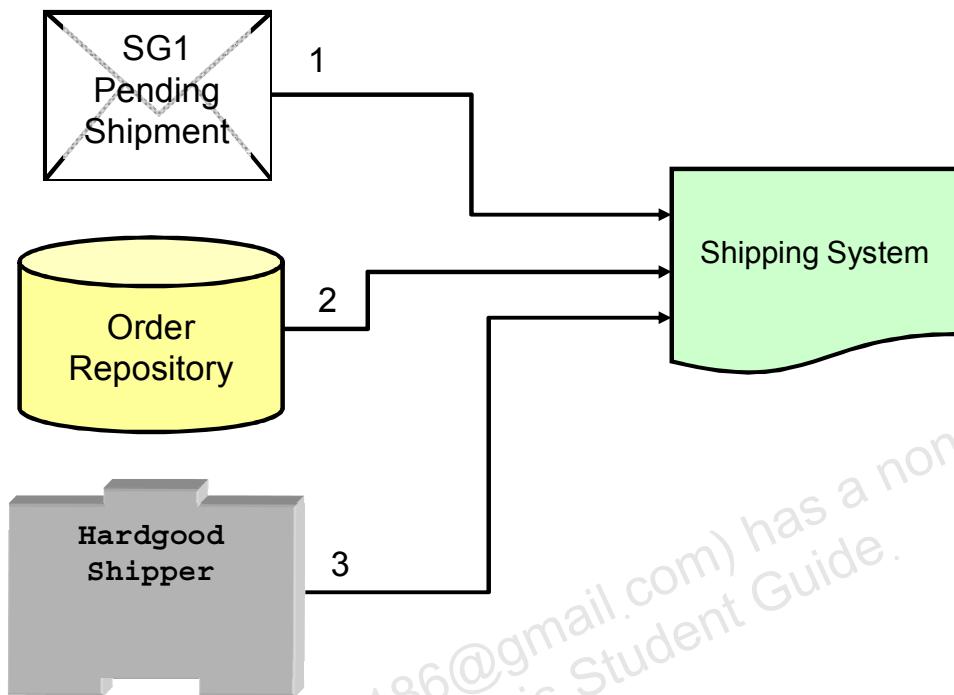
Step 1: The **HardgoodShipper** component calls the `shippingGroupHasShipped()` method of **HardgoodFulfiller**.

Step 2: The **HardgoodFulfiller** component calls the `shippingGroupHasShipped` pipeline.

Step 3: The `shippingGroupHasShipped` pipeline changes the shipping group's status to **NO_PENDING_ACTION** and sets its **shipdate**. It also sends a `ModifyOrderNotification` message to alert any listeners that the shipping group has been modified. The **OrderFulfiller** component is one of those listeners.

Step 4: When all of the shipping groups have been marked as **NO_PENDING_ACTION**, the **OrderFulfiller** will change the order state from **PROCESSING** to **NO_PENDING_ACTION**.

Integrating with a Shipping System: Options for Notifying the Shipping System



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

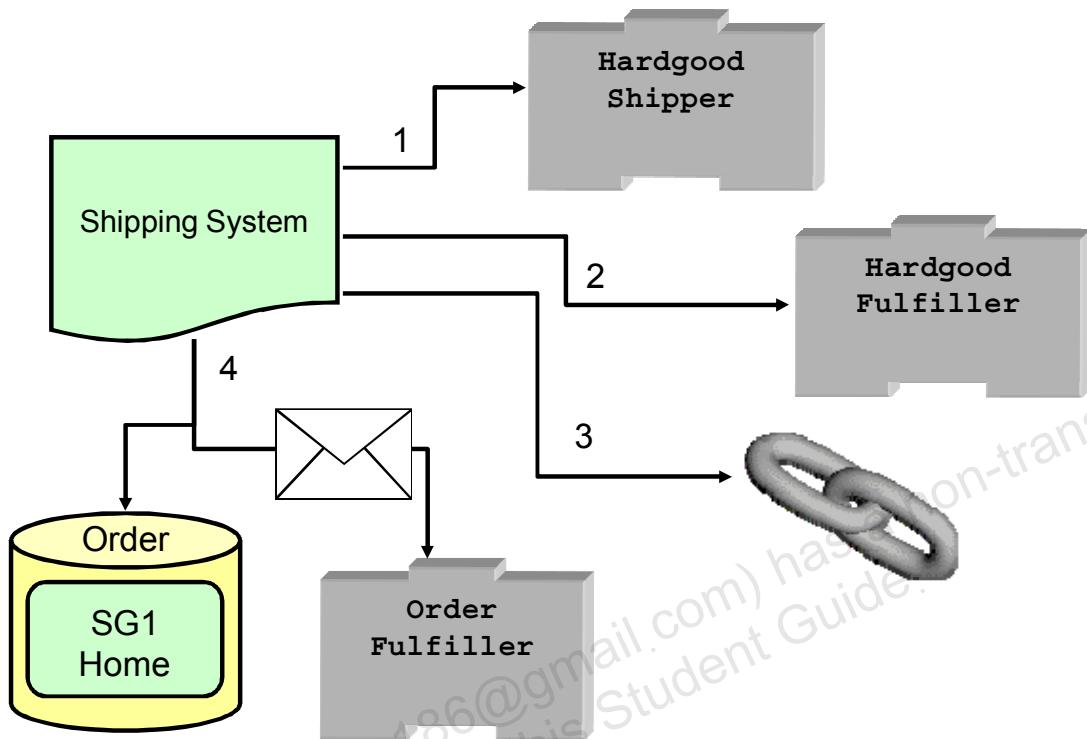
The out-of-the-box Fulfillment system does not include shipping functionality. There are a number of ways to have your shipping system be notified of groups waiting to be shipped.

Option 1: Configure a component in your shipping system as a JMS listener for `ModifyOrderNotification` messages of type `PENDING_SHIPMENT`.

Option 2: Have your shipping system poll the Order Repository, or the database directly, for shipping groups that are marked `PENDING_SHIPMENT`, either periodically or on demand.

Option 3: Extend the `atg.commerce.fulfillment.HardgoodShipper` class and replace the default `HardgoodShipper` component. Out of the box, this is a dummy component which polls for shipping groups that are pending shipment, then calls the `shippingGroupHasShipped()` method of `HardgoodFulfiller`.

Integrating with a Shipping System: Options for Communicating with ATG



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The shipping system needs to notify ATG's Fulfillment module when shipping groups have shipped so that ATG can continue on to processing payment for the order and appropriately update the order status. Whenever a shipping group has shipped, two things should happen: the properties of the shipping group should be modified accordingly (status set to NO_PENDING_ACTION and shipdate set to the date the items were shipped), and a ModifyOrderNotification message should be sent to alert the OrderFulfiller and the scenario engine that the group was shipped. There are several ways to accomplish this.

Option 1: Extend the HardgoodShipper class and replace the default HardgoodShipper component. Your code will somehow become aware that a shipping group has shipped, then call the shippingGroupHasShipped() method of HardgoodFulfiller.

Option 2: Call the shippingGroupHasShipped() method of HardgoodFulfiller.

Option 3: Use the PipelineManager component to call the shippingGroupHasShipped pipeline.

Option 4: Alter the shipping group and send the ModifyOrderNotification message directly from your code.

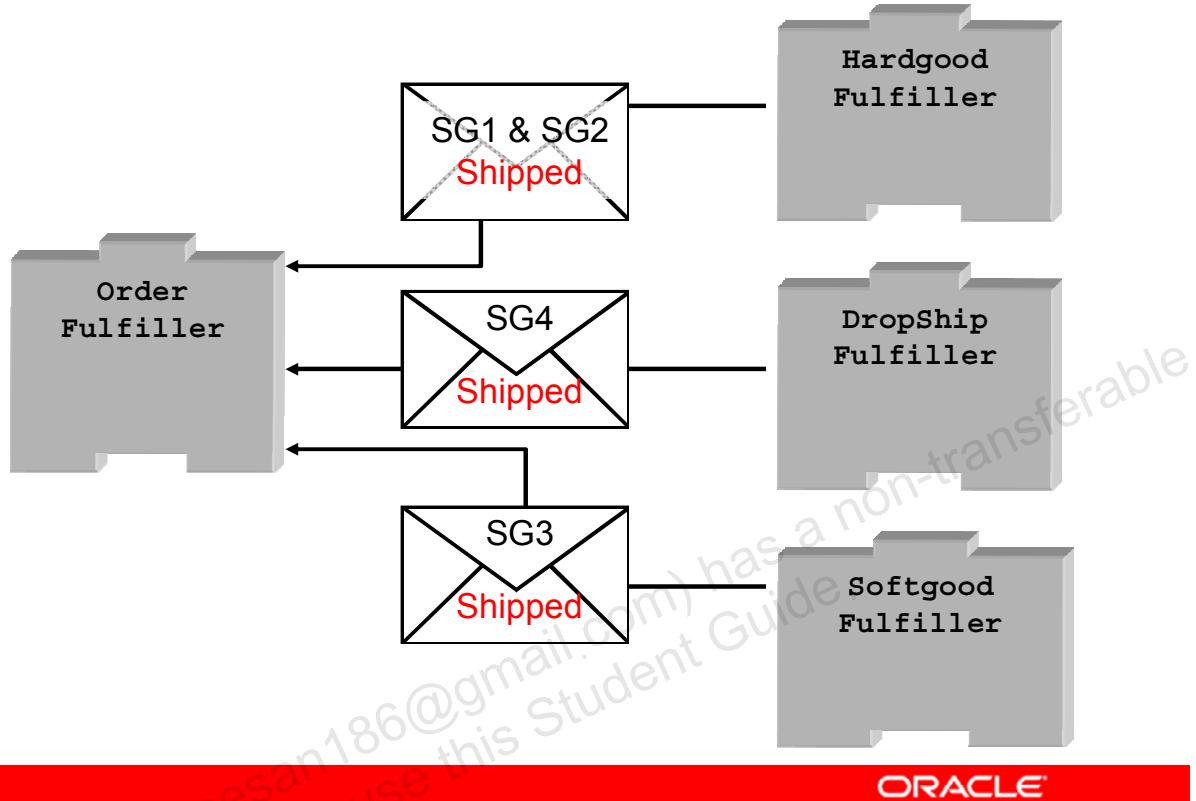
Design Decisions: Hard Goods

- Does your organization sell hard goods that need to be shipped?
- If so, how will you integrate the ATG Fulfillment system with your shipping system?
 - How will the shipping system be notified of groups needing to be shipped?
 - How will the shipping system notify ATG that a group has shipped?

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Default Payment Processing: Notifying OrderFulfiller of Shipment

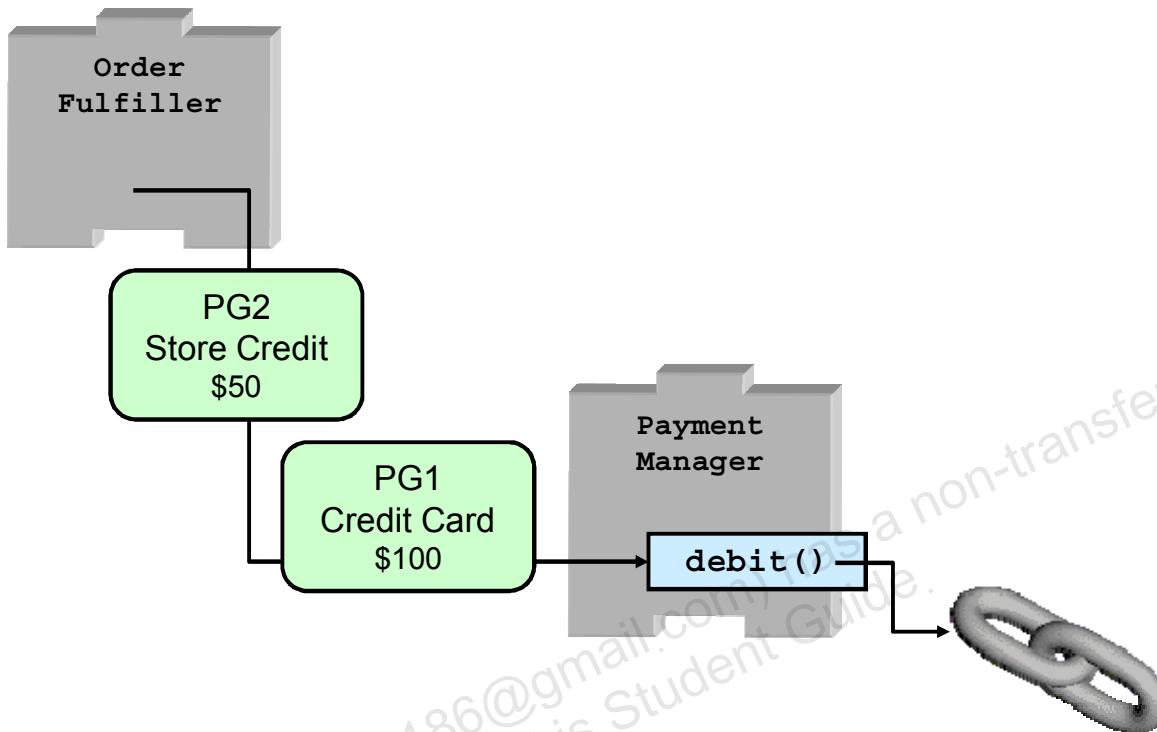


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

As each shipping groups is shipped, the fulfiller component responsible will send a message to OrderFulfiller notifying it of the shipment.

Default Payment Processing: Calling the PaymentManager.debit() Method



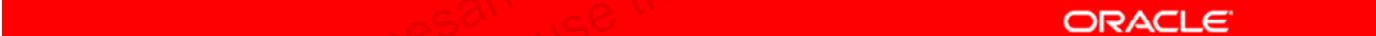
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `OrderFulfiller` component will then call the `PaymentManager's debit()` method, passing it references to all the payment groups in the order. The `Payment Manager` will call the appropriate pipeline for each particular payment type.

Options for Customizing Fulfillment

- JMS
 - All of the communication between Fulfillment components is done by JMS.
 - Sources and sinks can be added to customize or integrate with other systems.
- Pipelines
 - Like other order processing, most fulfillment tasks are carried out by pipelines.
 - Links can be added easily to pipelines.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Fulfillment Options

- Partial shipment:
 - Items are shipped separately as available, or held until all items are available.
- Payment timing:
 - Payment for the full order occurs after the first shipment, or after all shipments.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Design Decisions: Fulfillment

- Will the store use the ATG Fulfillment module?
 - If not, at what point will the non-ATG system take control of the order?
- Will any customizations to the ATG Fulfillment module be needed? Examples include:
 - Adding more listeners to Fulfillment messages
 - Adding links to the pipelines



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to describe:

- ATG's default inventory management and fulfillment systems
- The options for replacing the default systems

For More Information

ATG Commerce Programming Guide:

- Inventory Framework
- Configuring the Order Fulfillment Framework



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Oracle Commerce Technical Training

ATG Web Commerce Courses:

- *Foundations of ATG Application Development* (3 days)
- *Extending the ATG Platform* (2 days)
- *Implementing Your ATG Commerce Solution* (5 days)
- *ATG Personalization and Scenario Development* (3 days)
- *Configuring and Customizing Oracle ATG Web Commerce Service Center* (3 days)

Endeca Commerce Courses:

- *Implementing Endeca Commerce Core Technologies Rel 3.1* (3 days)
- *Developing Endeca Commerce Experience Manager Applications Rel 3.1* (2 days)



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The first three courses listed in the slide appear as gray instead of black. The first two, *Foundations of ATG Application Development* and *Extending the ATG Platform*, are the prerequisites for this course (the third in the list). ATG Web Commerce offers two additional courses for developers to further your knowledge. For more information on these courses, and to see the list of courses for business users and system administrators, visit <http://education.oracle.com>.

If you are implementing Endeca Commerce as part of your Oracle Commerce solution, you may want to attend the following the courses:

- *Implementing Endeca Commerce Core Technologies Rel 3.1*: In this course, students learn the fundamental knowledge of Endeca Guided Search and the tasks involved in extending and customizing the core functionality. This course is a prerequisite for the next course.
- *Developing Endeca Commerce Experience Manager Applications Rel 3.1*: This course teaches students how to develop the components needed to create applications using Endeca Commerce Experience Manager Rel 3.1. Students learn about the Assembler and how to use and extend the out-of-the-box cartridges and cartridge handlers, and how to process results from the Assembler within a web application.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable
license to use this Student Guide.

Displaying the Catalog and Breadcrumbs



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this appendix, you should be able to:

- Display top-level categories in JSPs
- Build drilldown pages for categories and products
- Use ATG's breadcrumb trail components

Road Map

- Displaying the catalog
- Managing the breadcrumb trail



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Catalog Browsing Flow

In general, a site displays top-level categories, then allows shoppers to drill down to subcategories and products.



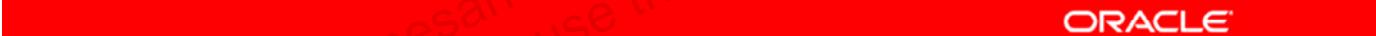
ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows the navigation from the top-level Gift Shop category. That category has two subcategories called For Him and For Her. It also has some featured products. When you click the Gift Shop category link, you see the featured products. If you click one of the products, you drill down to that product's page.

Top-Level Categories: Non-CRS

- Out of the box, without the CRS running, top-level categories are those with no parent categories.
- Use `Profile.catalog.allRootCategories` to retrieve the list for display.

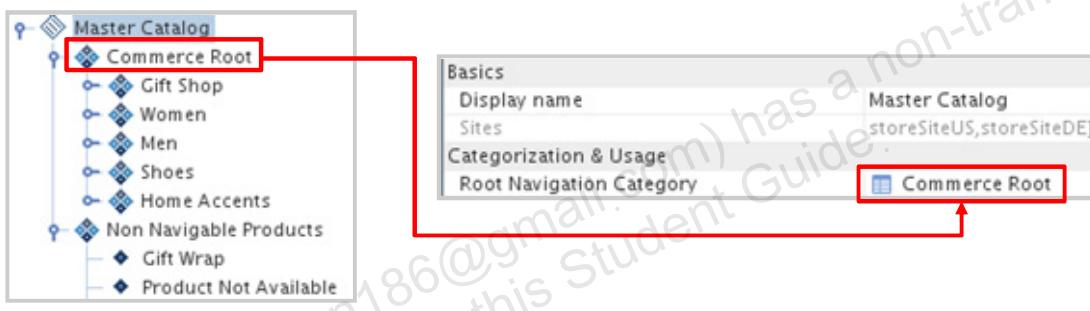


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Top-Level Categories: CRS

- CRS has extended catalogs to distinguish between navigable and non-navigable products.
- Top-level navigation categories are children of a specified root navigation category.
- Use the following to display top-level navigation categories if you are using CRS:
`Profile.catalog.rootNavigationCategory.childCategories`



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

An example of a non-navigable product is gift wrap. In the example in the slide, you are seeing the view from the ATG Control Center (ACC). The Master Catalog has a root navigation category called Commerce Root. The categories that appear below Commerce Root are the “top-level” categories that appear in the ATG Store site.

Drilldown Pages

- Display information about categories or products
- Use a lookup droplet to retrieve a category or product
 - The ID of the desired item must be passed in from the link on the previous page.
- Generally include links to subcategories or products, images, descriptive text, and more



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Commerce Lookup Droplets

- ATG Commerce provides four lookup droplet components:
 - CategoryLookup
 - ProductLookup
 - SKULookup
 - MediaLookup
- All of these droplets are based on the `ItemLookupDroplet` class, which provides a mechanism to retrieve items from a repository.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The Commerce-related `ItemLookupDroplet` components are located in `/atg/commerce/catalog`.

ItemLookupDroplet

- Input Parameter
 - id (required)
- Open Parameters
 - output: Rendered if item was found
 - empty: Rendered if item was not found
 - error: Rendered if exception occurred
- Output Parameter
 - element: Set to the `RepositoryItem` returned by the repository lookup



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

These are only some of the parameters supported by the droplets based on the `ItemLookupDroplet` class. See Appendix B: “ATG Servlet Beans” in the *Page Developer’s Guide* for full documentation.

Links to Categories and Products

Categories and products have a template property that specifies which JSP should be used for display of that object.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Links to Categories and Products: Non-CRS

```
<c:forEach var="el" ...>
  ...
  <dsp:getvalueof var="templateURL"
    idtype="String"
    param="el.template.url" />
  <dsp:a page="${templateURL}">
    <dsp:param name="itemId"
      param="el.repositoryId"/>
    <dsp:valueof
      param="el.displayName"/>
  </dsp:a>
  ...
</c:forEach>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the `ForEach` droplet is being used to loop through the category or product list. It uses the `var` attribute to rename the element output parameter to `el`, which represents the current object for this iteration.

Links to Categories and Products: CRS

CRS includes the CatalogItemLink and ProductLookupItemLink droplets.

- They are based on the ItemLink class.
- They translate URLs appropriately when being spidered by a search engine (SEO).
- For shoppers, they generate URLs by using template patterns.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

For more details about URL recoding for SEO, refer to the chapter titled “Search Engine Optimization” in the *Commerce Reference Store Overview*.

CatalogItemLink

CatalogItemLink uses the following templates to generate URLs for shoppers:

- For categories:
`{item.template.url}?categoryId={item.id}`
- For products:
`{item.template.url}?productId={item.id}&categoryId={item.parentCategory.id}`



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the component is /atg/repository/seo/CatalogItemLink.

ProductLookupItemLink

- ProductLookupItemLink uses the following template to generate URLs for shoppers:
 - {item.template.url}?productId={item.id}&category=categoryId
- The template includes the ID of the current category, rather than the product's default parent category.
- It cannot be used out of the box for category links.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the component is /atg/repository/seo/ProductLookupItemLink.

Example: CatalogItemLink

```
<c:forEach var="el" ...>
  ...
  <dsp:droplet name="CatalogItemLink">
    <dsp:param name="item" param="el"/>

    <dsp:oparam name="output">
      <dsp:getvalueof id="finalUrl"
        idtype="String" param="url"/>

      <dsp:a page="${finalUrl}">
        <dsp:valueofparam="el.displayName"/>
      </dsp:a>

    </dsp:oparam>
  </dsp:droplet>
  ...
</c:forEach>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The example in the slide has been simplified. It does not include Cascading Style Sheet (CSS) formatting or internationalization. Again, the element object has been renamed to el and is the output of the loop that encloses this code.

Road Map

- Displaying the catalog
- Managing the breadcrumb trail



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Breadcrumb Trail

The breadcrumb trail:

- Helps users traverse category hierarchy by keeping a “stack” of catalog elements
- Is sometimes called “Historical Navigation”
- Is needed because there may be more than one path to a product or subcategory



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Key Breadcrumb Trail Objects

- The CatalogNavHistory component:
 - Stores a stack of catalog repository items in the navHistory property
 - Uses the navCount property to track if the stack is still in sync with the user
 - Example: If the browser's back and forward buttons have been used
- The CatalogNavHistoryCollector droplet is used to “push” and “pop” items on and off stack.



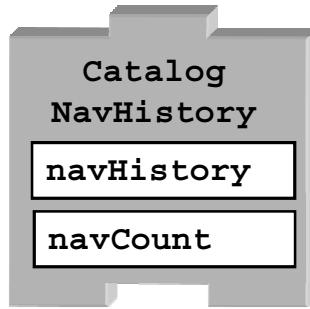
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full component name is /atg/commerce/catalog/CatalogNavHistory, and is based on the atg.commerce.catalog.CatalogNavHistory class.

CatalogNavHistoryCollector is also in /atg/commerce/catalog/, and is based on atg.repository.servlet.NavHistoryCollector.

CatalogNavHistory Component

- The CatalogNavHistory component is used to store a stack of catalog repository items.



- The CatalogNavHistoryCollector droplet is used to “push” and “pop” items on and off stack.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

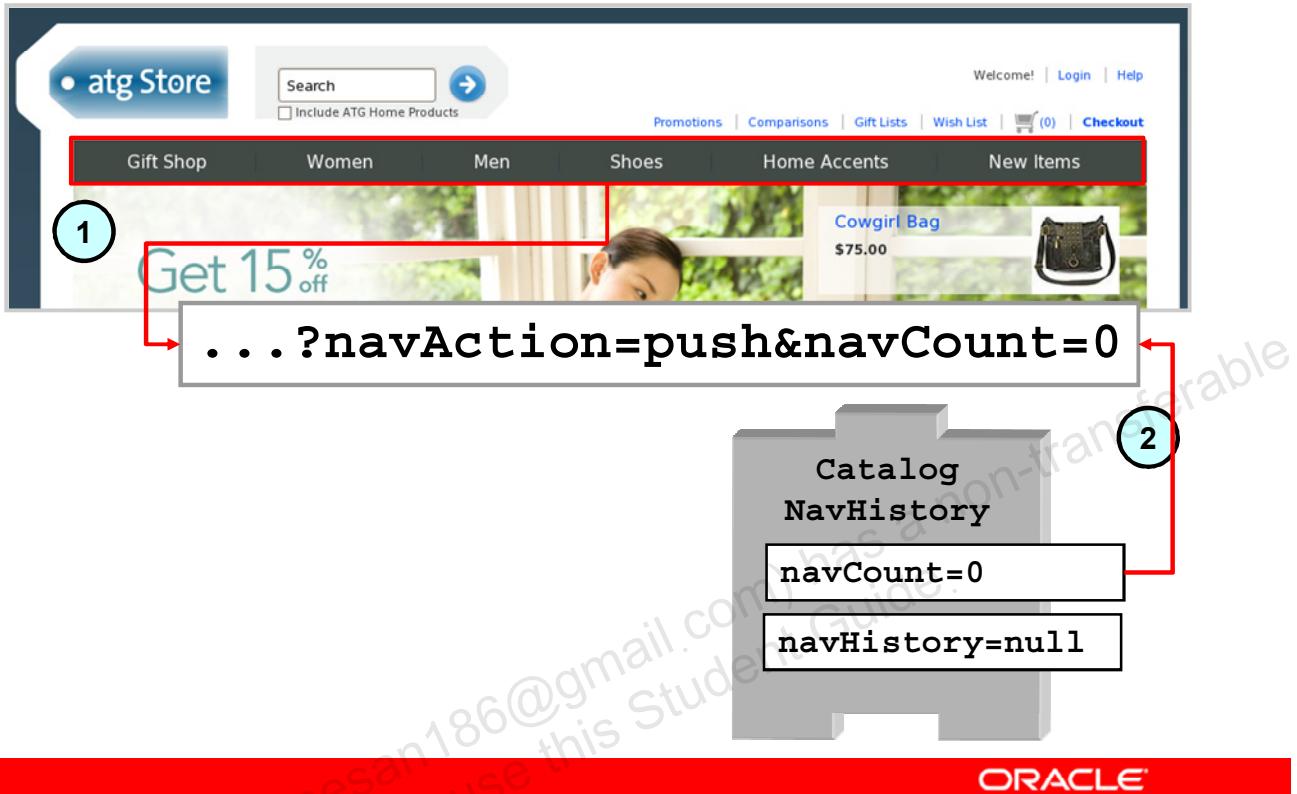
The CatalogNavHistory component uses two properties to keep track of the navigation history:

- It stores the stack of catalog repository items in the `navHistory` property.
- It uses the `navCount` property to track if the stack is still in sync with the user, such as when the browser’s back and forward buttons have been used.

The full component name is `/atg/commerce/catalog/CatalogNavHistory`, and is based on the `atg.commerce.catalog.CatalogNavHistory` class.

CatalogNavHistoryCollector is also in `/atg/commerce/catalog/`, and is based on `atg.repository.servlet.NavHistoryCollector`.

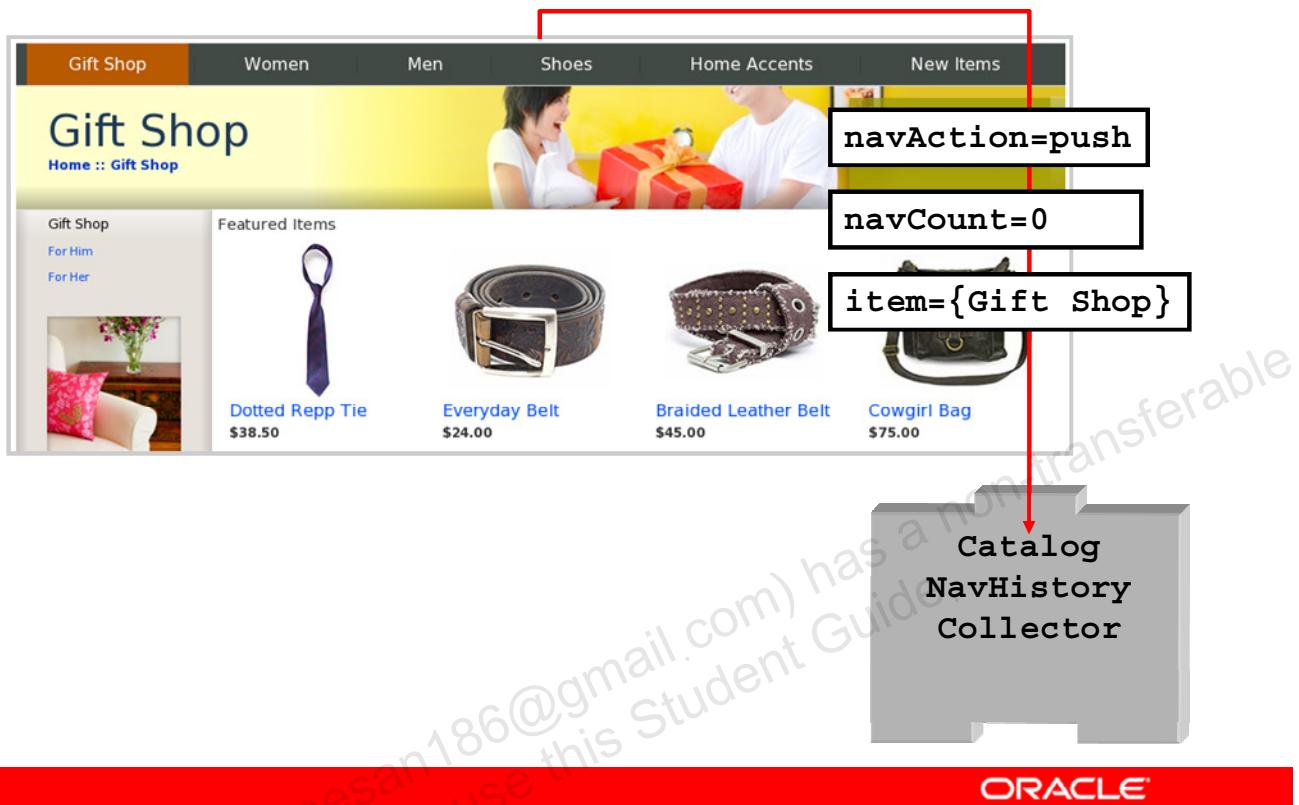
Breadcrumb Trail In Action: Step One



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The first time a shopper opens a page that calls on the CatalogNavHistory component, a session-scoped instance is created for that shopper. The navCount is initialized to 0, and the navHistory is null. You may notice that the navAction variable is set to "push."

Breadcrumb Trail In Action: Clicking a Link



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When a shopper clicks a link to view a category, the request includes the `navCount` number and the desired action ("push"), as shown in the previous slide. This information is passed to the `CatalogNavHistoryCollector` at the top of the category details page.

Example: Maintaining the Stack

```
<dsp:droplet  
name="/atg/commerce/catalog/CategoryLookup">  
<dsp:param name="id" param="itemId"/>  
<dsp:oparam name="output">  
...  


```
<dsp:droplet name="CatalogNavHistoryCollector">
 <dsp:param name="item" param="element"/>
 <dsp:param name="navAction" param="navAction"/>
 <dsp:param name="navCount" param="navCount"/>
</dsp:droplet>
```

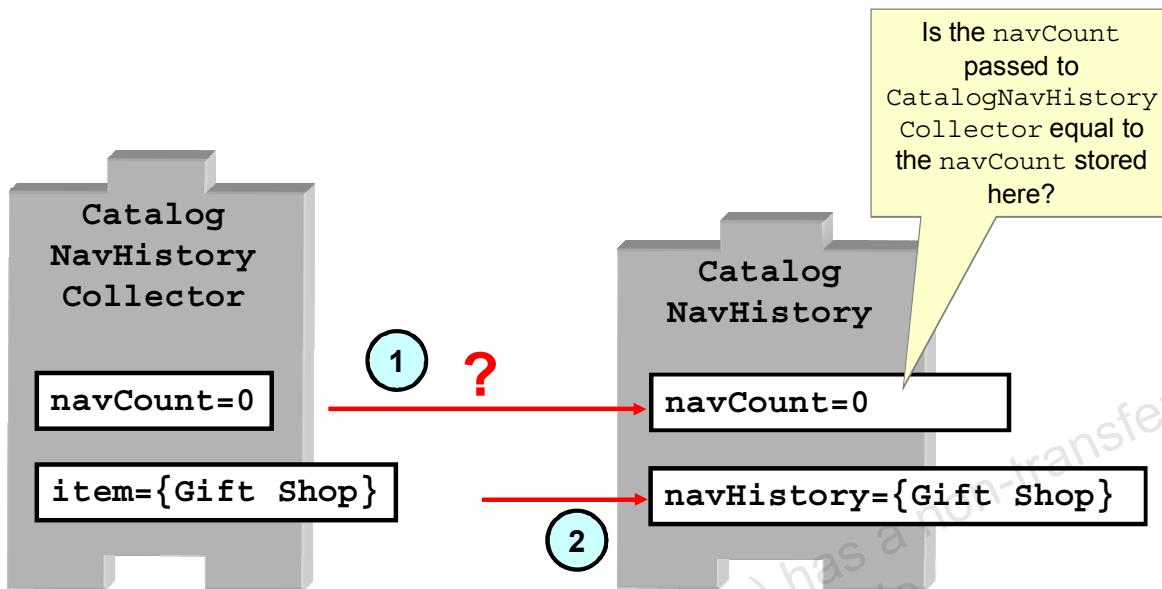
  
...  
</dsp:oparam>  
</dsp:droplet>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The CatalogNavHistoryCollector droplet should be called on every page that displays a catalog item, using a call similar to the example shown in the slide.

Breadcrumb Trail In Action: Incrementing NavCount



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

CatalogNavHistoryCollector checks to see if the navCount passed to it is equal to the navCount value stored in CatalogNavHistory. If it is, the application knows that the shopper has not used the back or forward buttons. The catalog object that was passed is then added to the stack, and the navCount is incremented.

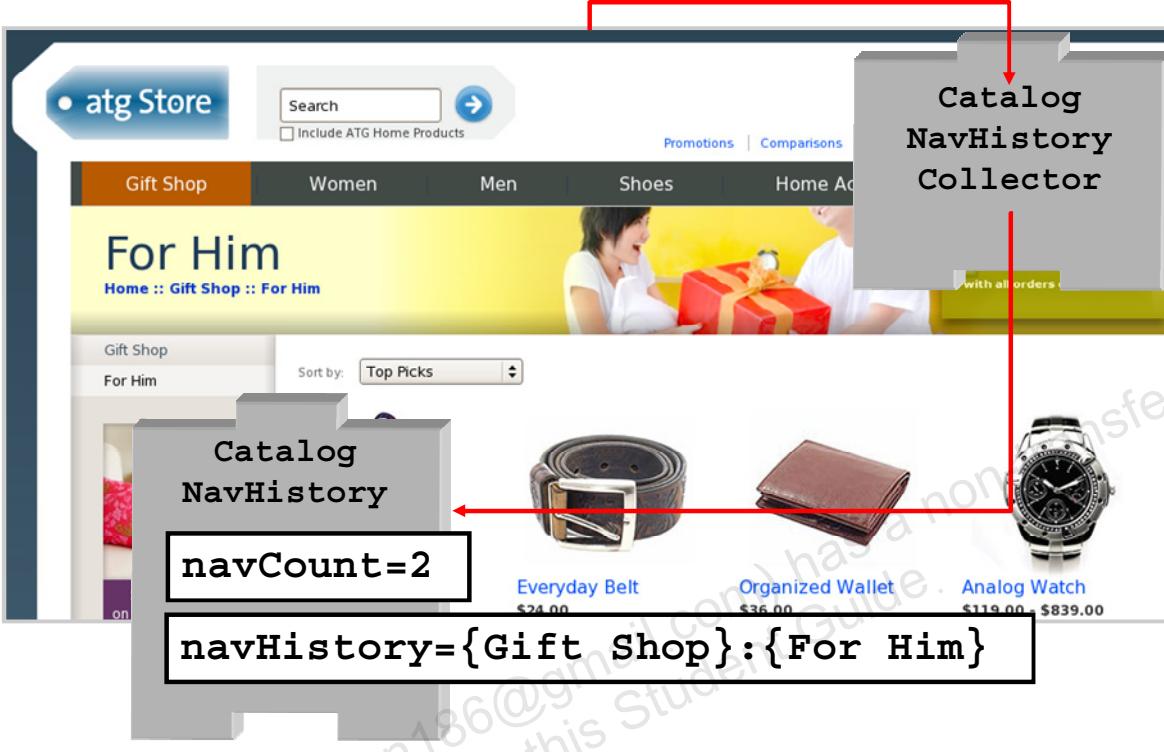
Breadcrumb Trail In Action: Drilling Down to a Subcategory



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Drilldown links on the category details page will also include the `navAction` parameter set to "push," and the current `navCount` value, which is now set to 1.

Breadcrumb Trail In Action: Subcategory

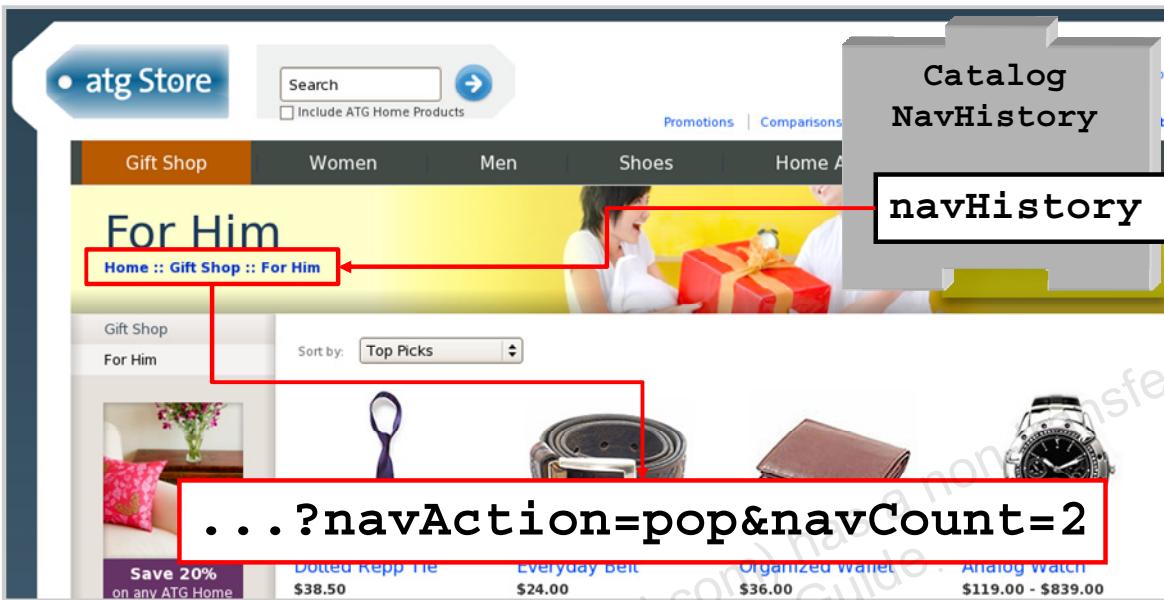


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

On the next subcategory page, the process is repeated: CatalogNavHistoryCollector is called, which compares the value of navCount from the request with the value stored in CatalogNavHistory and adds the next category to the stack.

Breadcrumb Trail In Action: Breadcrumb Display



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The breadcrumb trail itself is displayed by accessing and looping through the objects in the `navHistory` property. These links should use the “pop” `navAction`.

Breadcrumb Trail In Action: Pop navAction



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When a link from the breadcrumb trail is clicked (Gift Shop in the example in the slide), the "pop" request is passed to the CatalogNavHistoryCollector, which passes it on to CatalogNavHistory. The relevant items are removed from the stack, and the navCount is again incremented.

Using the Back Button: Part One



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

In the example in the slide, the shopper has navigated down to the “For Him” category, and the current navCount is 2. The browser’s back button is then used to get back to the “Gift Shop” page.

Using the Back Button: Part Two

The screenshot shows a web browser displaying a "Gift Shop" page. The top navigation bar includes links for "Gift Shop", "Women", "Men", "Shoes", "Home Accents", and "New Items". A promotional banner on the right side offers a "Free Scarf" with orders over \$200. On the left, there's a sidebar with a "Gift Shop" heading and links for "For Him" and "For Her". The main content area is titled "Featured Items" and displays four products: a "Dotted Repp Tie" (\$38.50), an "Everyday Belt" (\$24.00), a "Braided Leather Belt" (\$45.00), and a "Cowgirl Bag" (\$75.00). A red box highlights the "For Him" link in the sidebar, and a red arrow points from this box to a red-bordered text box containing the URL parameter ". . . ?navAction=push&navCount=1".

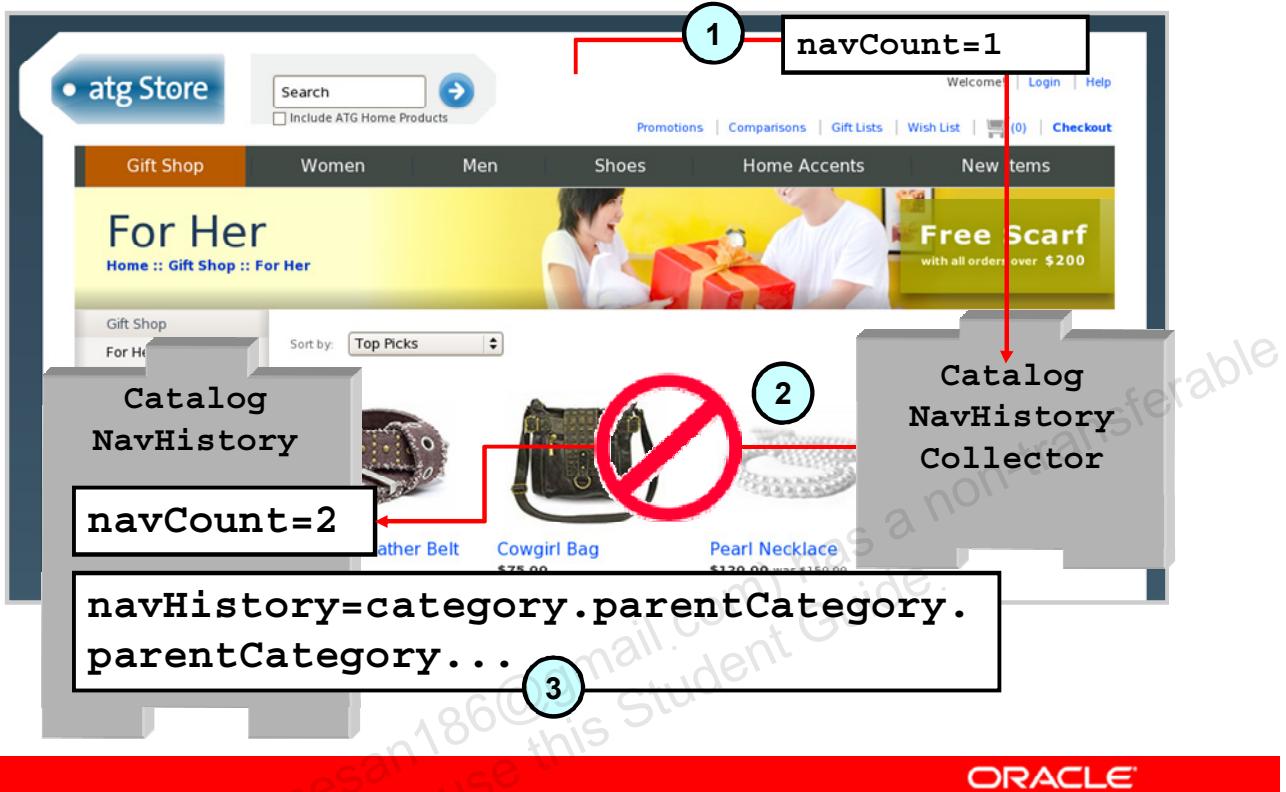
...?navAction=push&navCount=1

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Because the Gift Shop page has been loaded from the browser's memory, the navCount parameter on its links is still set to 1.

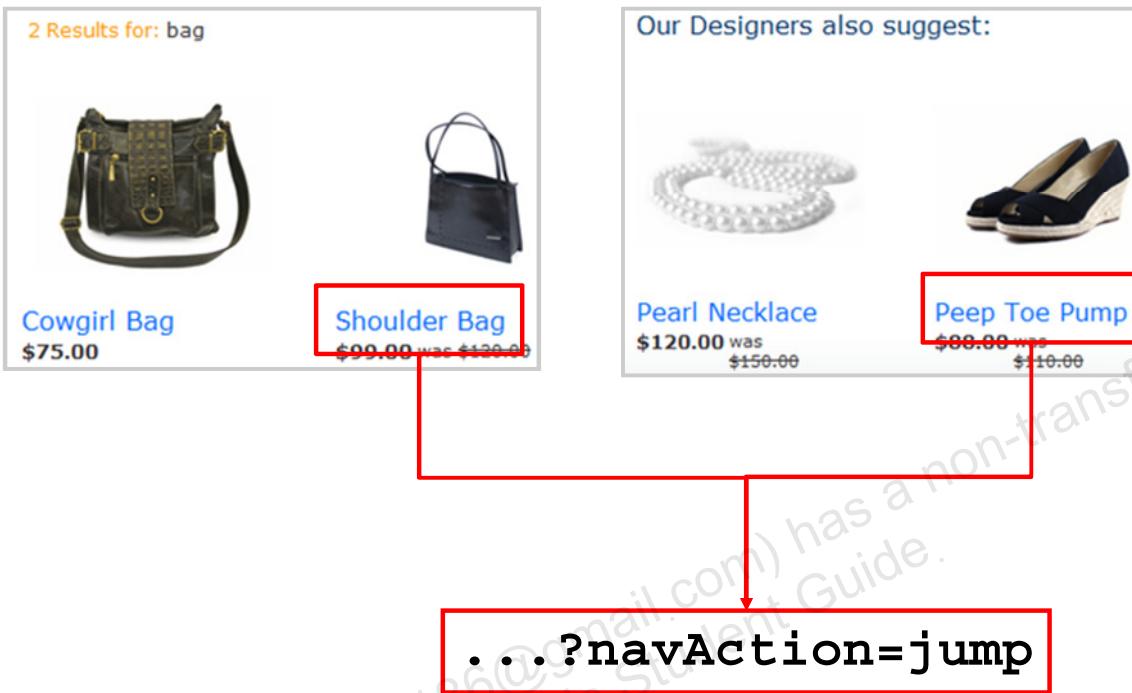
Using the Back Button: Part Three



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When the shopper clicks a link on that page, the navCount passed to CatalogNavHistoryCollector will not match the one stored in CatalogNavHistory. This prompts the CatalogNavHistory component to rebuild the navHistory from the default parent categories of the catalog objects.

Jumping to Products or Categories



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

There are a number of ways that a product or category page might be opened without the shopper navigating through the hierarchy. For example, the shopper clicks a link from a search page, from a bookmark, or from a related product link. In those cases, the `navAction` should be "jump."

Example: Jumping to Products or Categories



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `navAction "jump"` causes `CatalogNavHistory` to rebuild the `navHistory`.

Summary

In this appendix, you should have learned how to:

- Display top-level categories in JSPs
- Build drilldown pages for categories and products
- Use ATG's breadcrumb trail components

For More Information

ATG Commerce Guide to Setting Up a Store

- ATG Commerce Catalog Administration
- Catalog Navigation and Searching
- Appendix: ATG Commerce Servlet Beans



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

B **Additional Multisite Information**

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this appendix, you should be able to:

- Describe how to set the site context
- Create new item mappings for site templates

Road Map

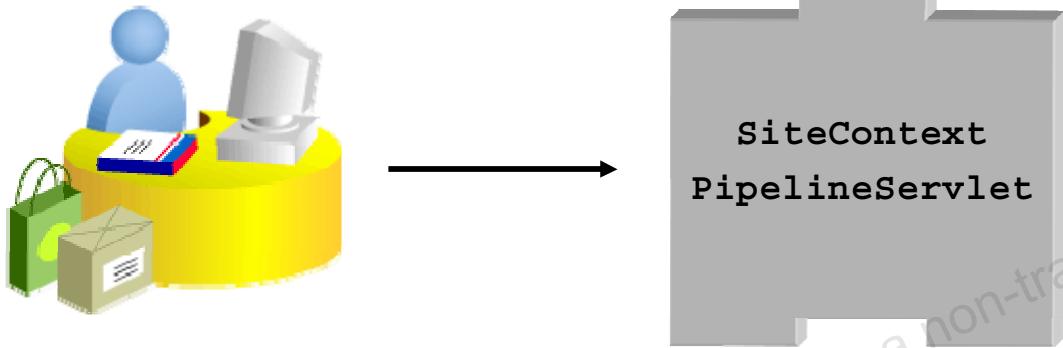
- Setting the site context
- Creating new item mappings for site templates



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

SiteContextPipelineServlet

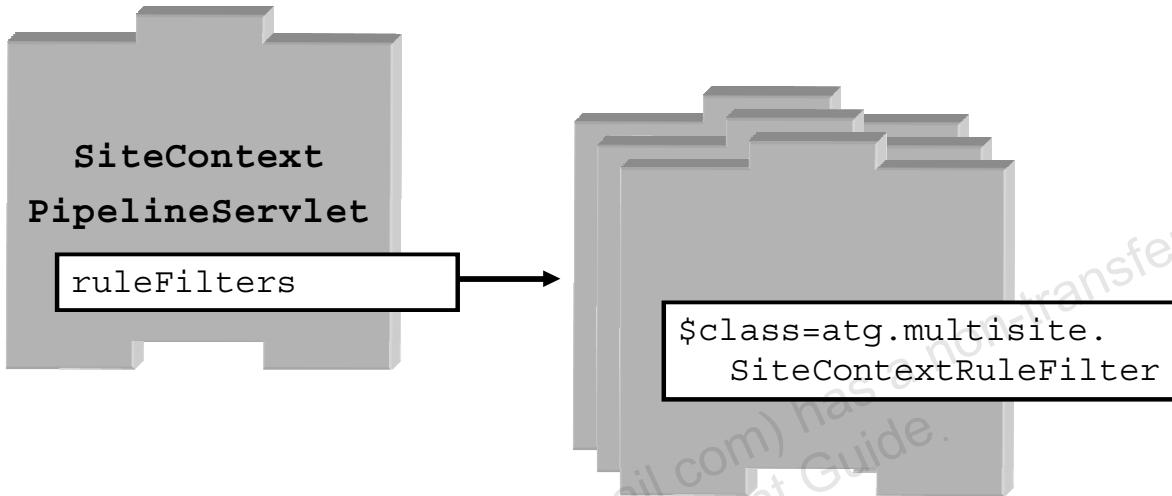


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The first site-related servlet in the servlet pipeline is SiteContextPipelineServlet.

SiteContextRuleFilters



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The SiteContextPipelineServlet has an array of SiteContextRuleFilter components in its ruleFilters property. Each filter is queried in order until a site is determined. Developers could add their own filters to this property to customize how sites are assigned to requests. SiteContextRuleFilter components are based on classes that implement atg.multisite.SiteContextRuleFilter.

Out-of-the-Box Rule Filters

1. RequestParameterRuleFilter
2. URLPatternMatchingRuleFilter
3. DefaultSiteContextRuleFilter



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When determining the site, the ATG server uses three out-of-the-box rule filters, in the order shown in the slide, to determine the site context. If one rule filter succeeds, the server skips the remaining filters.

RequestParameterRuleFilter

Sets the site to the site named in the pushSite request parameter

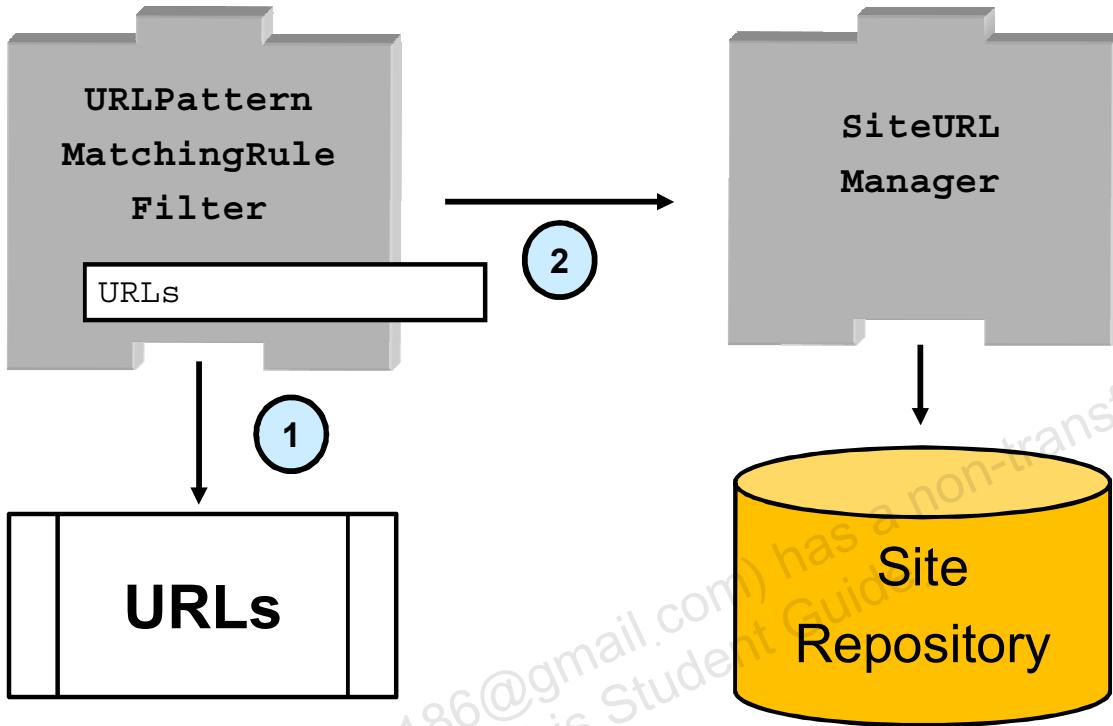
- The parameter name is configured in the pushSiteParamName property.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The RequestParameterRuleFilter component is based on atg.multisite.PushedSiteParamFilter. The full path to the component is /atg/multisite/RequestParameterRuleFilter. This filter checks the request for a specific parameter that indicates that the entire request should take place in the context of the given site.

URLPatternMatchingRuleFilter



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The URLPatternMatchingRuleFilter component is based on `atg.multisite.URLPatternMatchingRuleFilter`. The full path to the component is `/atg/multisite/URLPatternMatchingRuleFilter`. The steps for determining the URL are the following:

1. The filter checks its own map of URLs first (which is manually populated by developers) for a match, or a “starts with” match against the current request’s URL.
2. If no match is found, the filter asks the SiteURLManager for a match. The SiteURLManager builds a cached map of URLs from the Site Repository.

Two properties govern which checks are done:

- `enableSimpleAlgorithm`
- `enableSiteManagerAlgorithm`

DefaultSiteContextRuleFilter

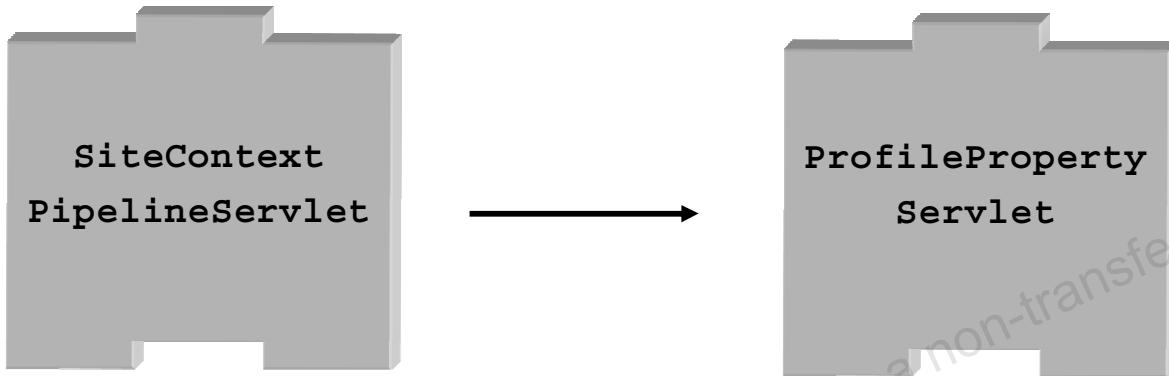
- This filter sets the site to the site configured in the defaultSiteId property.
- Out of the box, defaultSiteId is null.
 - When running CRS, defaultSiteId is set to storeSiteUS.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The DefaultSiteContextRuleFilter component is based on atg.multisite.DefaultSiteContextRuleFilter. The full path to the component is /atg/multisite/DefaultSiteRuleFilter. This filter method simply returns a default SiteContext name. It should be used as the last in the filter chain, so that it only executes if all other methods of determining the site context name fail.

ProfilePropertyServlet



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

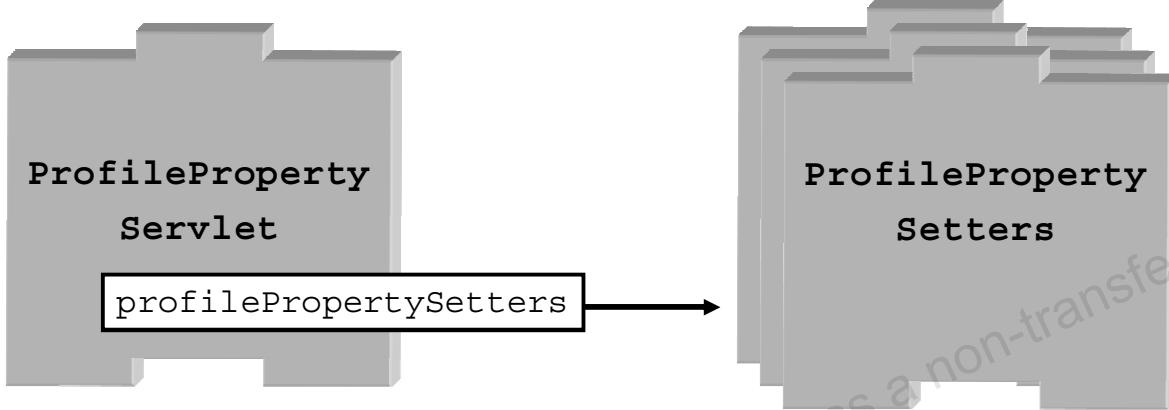
After the site context has been selected, the ProfilePropertyServlet uses it to set site-specific properties on the current profile.

(Note that the diagram in the slide is a simplification; there are other servlets in the pipeline between SiteContextPipelineServlet and ProfilePropertyServlet.)

The full path of the component is

/atg/dynamo/servlet/dafpipeline/ProfilePropertyServlet. It is based on atg.userprofiling.ProfilePropertyServlet.

ProfilePropertySetters



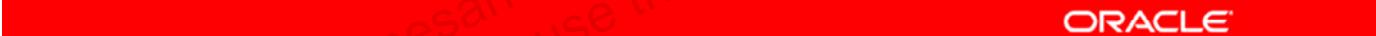
ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The ProfilePropertyServlet has an array of ProfilePropertySetters in its profilePropertySetters property. Developers may add their own ProfilePropertySetters, if needed. ProfilePropertySetter components are based on classes that extend atg.userprofiling.ProfilePropertySetter.

Out-of-the-Box Property Setters

1. SiteProfilePropertySetter
 - Added by DAS
2. CatalogProfilePropertySetter
 - Added by DCS
3. PriceListProfilePropertySetter
 - Added by DCS
4. StoreProfilePropertySetter
 - Added by CRS



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

SiteProfilePropertySetter

- This property setter sets profile properties from values of site properties.
- `siteProfilePropertyMap` maps site property names to profile property names.
- By default, `siteProfilePropertyMap` is null.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the component is `/atg/userprofiling/SiteProfilePropertySetter`.
It is based on `atg.userprofiling.SiteProfilePropertySetter`.

CatalogProfilePropertySetter

- This property setter sets the catalog from the catalog specified for a site.
- It can be disabled using the `enabled` property.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the component is
`/atg/userprofiling/CatalogProfilePropertySetter`. It is based on
`atg.commerce.catalog.custom.CatalogProfilePropertySetter`.

PriceListPropertySetter

- This property setter sets the price list from the price list specified for a site.
- It can be disabled using the enabled property.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the component is
/atg/userprofiling/PriceListProfilePropertySetter. It is based on
atg.commerce.pricing.priceLists.PriceListProfilePropertySetter.

StoreProfilePropertySetter

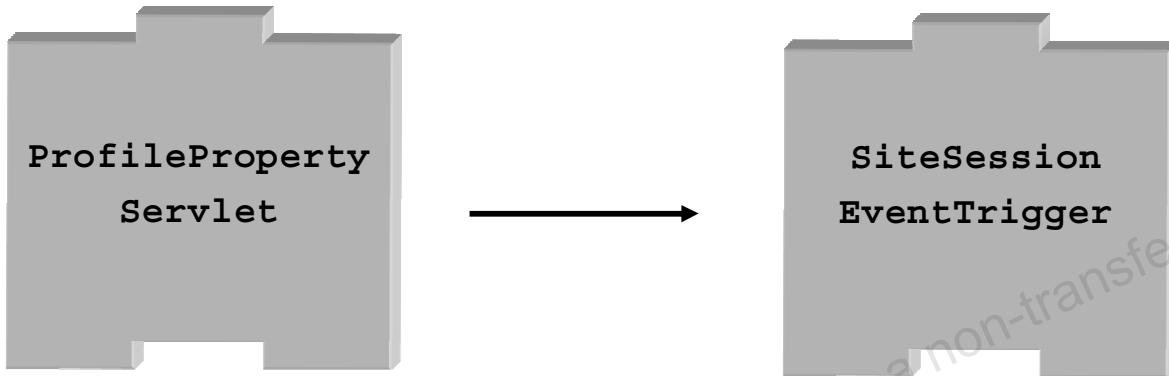
- This property setter does nothing out of the box.
- Developers using CRS can extend its `setProperties` method to add custom code.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full path to the component is
`/atg/userprofiling/StoreProfilePropertySetter`. It is based on
`atg.projects.store.servlet.pipeline.StoreProfilePropertySetter`.

SiteSessionEventTrigger



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

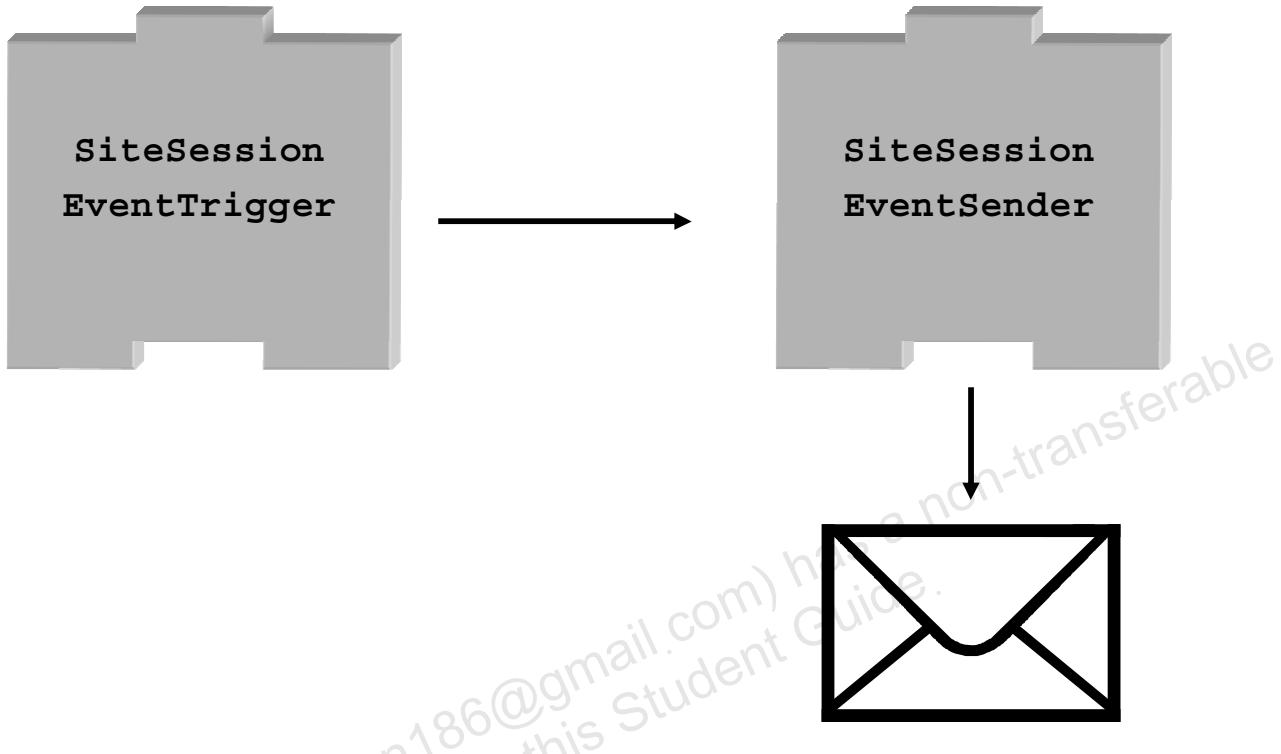
The SiteSessionEventTrigger servlet is later in the servlet pipeline.

(Note that this diagram is a simplification; there are other servlets in the pipeline between ProfilePropertyServlet and SiteSessionEventTrigger.)

The full path of the component is

/atg/dynamo/servlet/dafpipeline/SiteSessionEventTrigger. It is based on atg.multisite.SiteSessionEventTriggerPipelineServlet.

SiteSessionEventSender



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

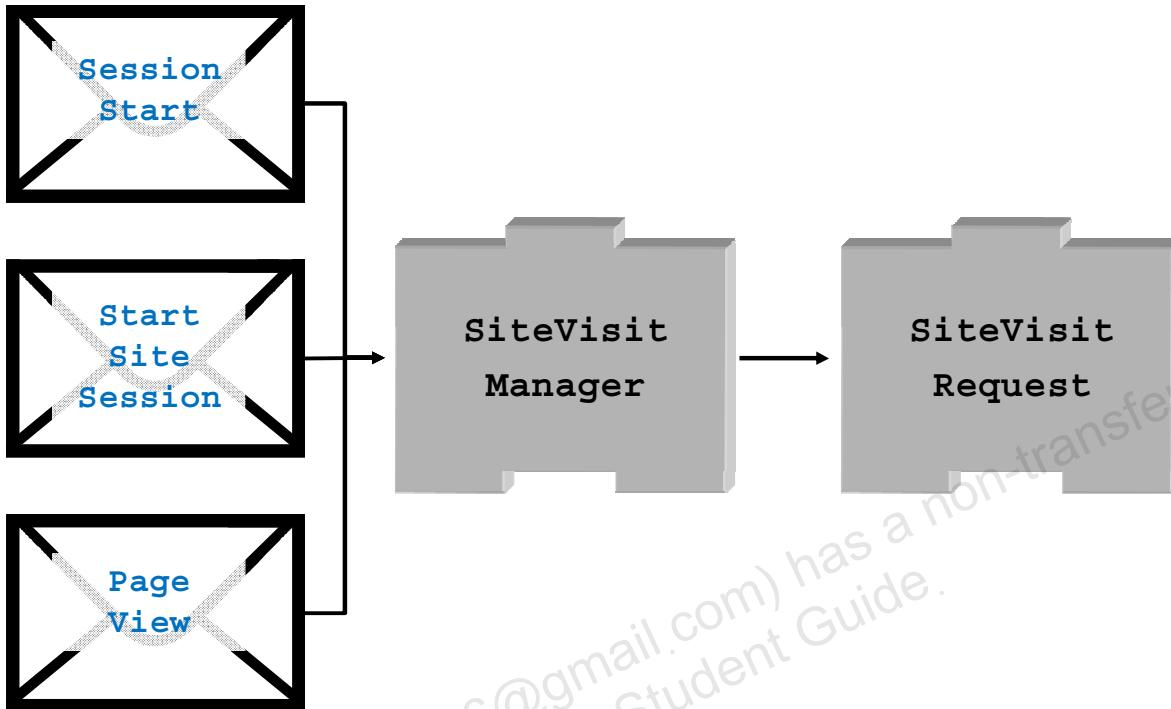
If the site context has changed, the `SiteSessionEventTrigger` triggers the `SiteSessionEventSender` component to send an event (JMS message). This functionality can be disabled by setting `SiteSessionEventTrigger`'s `sendSiteConfigurationEvents` property to `false`.

`SiteSessionEventSender` sends two JMS messages:

1. `StartSiteSession` is sent when the first site context is assigned. It contains the site ID and the referring URL.
2. `SiteChanged` is sent, if the site changes during a session. It contains the new and old site IDs.

Both messages are sent to local topic destinations. Developers can add their own listeners.

SiteVisitManager



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The SiteVisitManager component listens for SessionStart, StartSiteSession, and PageView events. It then calls methods on SiteVisitRequest to update session-scoped site-related information. The full path to the SiteVisitManager component is /atg/reporting/datacollection/userprofiling/SiteVisitManager. It is based on atg.userprofiling.dms.SiteVisitManager.

The full path to the SiteVisitRequest component is /atg/reporting/datacollection/userprofiling/SiteVisitRequest. This component:

- Is session-scoped
- Stores site-related information, such as:
 - currentSite
 - entrySite
 - siteCount
 - siteVisitSessions
- Sends SiteVisit message at the end of a user's session
- Contains all information stored in SiteVisitRequest

Disabling Multisite

- There are two ways to disable multisite:
 - Set `SiteContextPipelineServlet.enabled` to `false`, or
 - Set the web application context parameter `atg.multisite.SiteContextEnabled` to `false`
- The component property takes precedence over the web application parameter.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

- Setting the site context
- Creating new item mappings for site templates



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Creating a New Site Template

1. Create a new item mapping in the View Mapping Repository, if needed.
 - Site templates may be created solely to provide different default values, in which case a new item mapping is not necessary
2. Create a new site category item in the Site Repository.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

This section focuses on creating a new item mapping in the View Mapping Repository. The lesson titled “Multisite and ATG Applications” covered the second step.

Creating a New Item Mapping

1. Create a new item mapping in the View Mapping Repository.
 - Recommendation: Use `SiteCategory : mySiteTemplateName` as the format for the ID
2. Create new item view mappings, item views, property view mappings, and property views, as needed.
3. Associate the new item mapping with new or existing item view mappings, item views, property view mappings, and property views.

Tip: View or copy an existing default item mapping and its associated items.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Developers can use the ATG Control Center (ACC) on the Content Administration (CA) server to create and view items in the View Mapping Repository.

Item Mappings

- Item mappings are used to dynamically determine Java Server Pages (JSPs) or Flex elements to use to render a content asset in the Business Control Center (BCC).
- Each combination of asset type (such as a repository item or a file) and state (such as edit or view) is tied to an item mapping.
- Specific asset types (such as `siteConfiguration`) may have designated item mappings, rather than using the default for their asset type.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Item Views

- Item views:
 - Represent JSP fragments or Flex elements used to render assets
 - Are a container for various property views
- Assets may be mapped to more than one item view if there is more than one tab of properties.

Property View Mappings

- Property mappings are used to map property types to JSP fragments or Flex elements.
- As with item mappings, each property type (for example, a string) is tied to a default property view mapping.
- Specific properties (such as `longDescription`) may have specific property view mappings, used to override the default rendering or set custom options for rendering.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Property Views

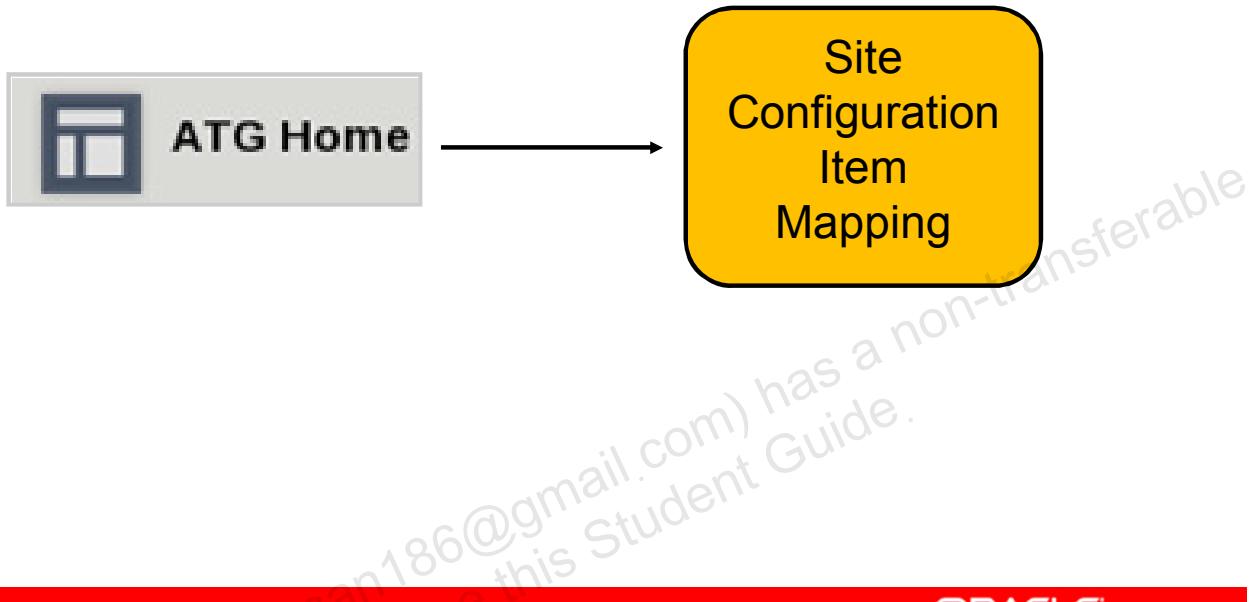
Property views:

- Represent JSP fragments or Flex elements used to render *individual asset properties*
- May have attributes defined that can be used to customize the rendering of the element
 - Examples: Width, header text
 - The `isReadOnly` property can be used to make a property read-only on a certain template.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

siteConfiguration Item Mapping

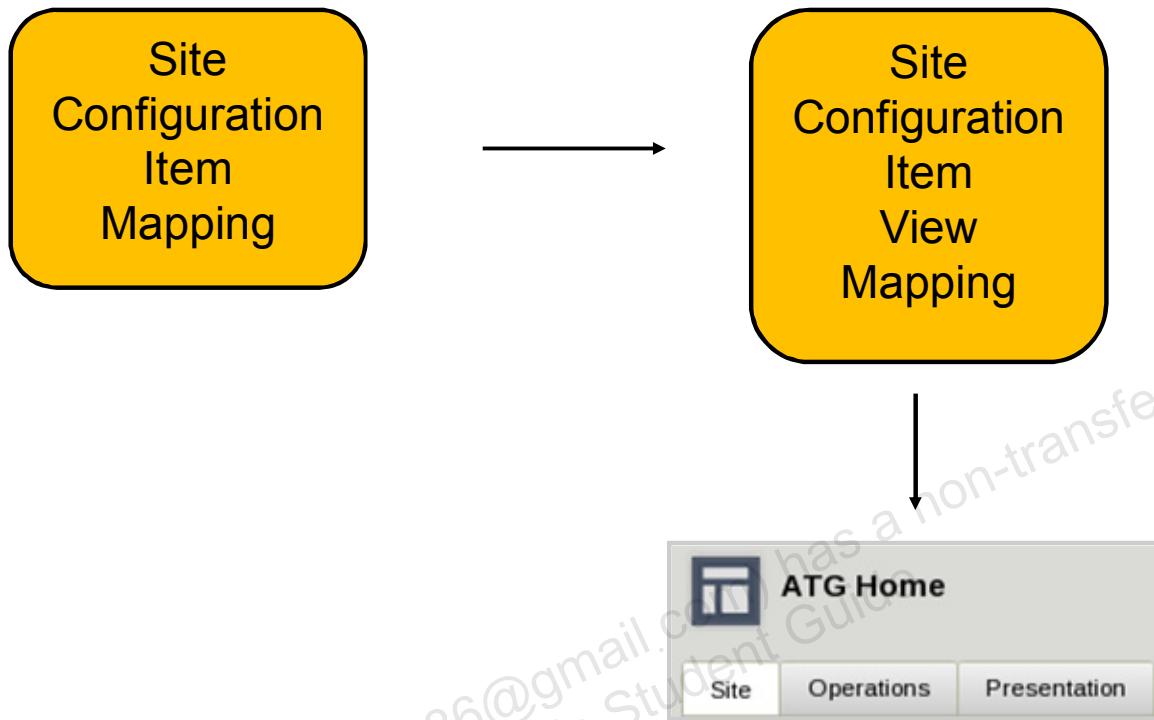


ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

When a siteConfiguration asset needs to be rendered in the BCC, its item mapping is retrieved.

Item Mapping to Item View Mapping

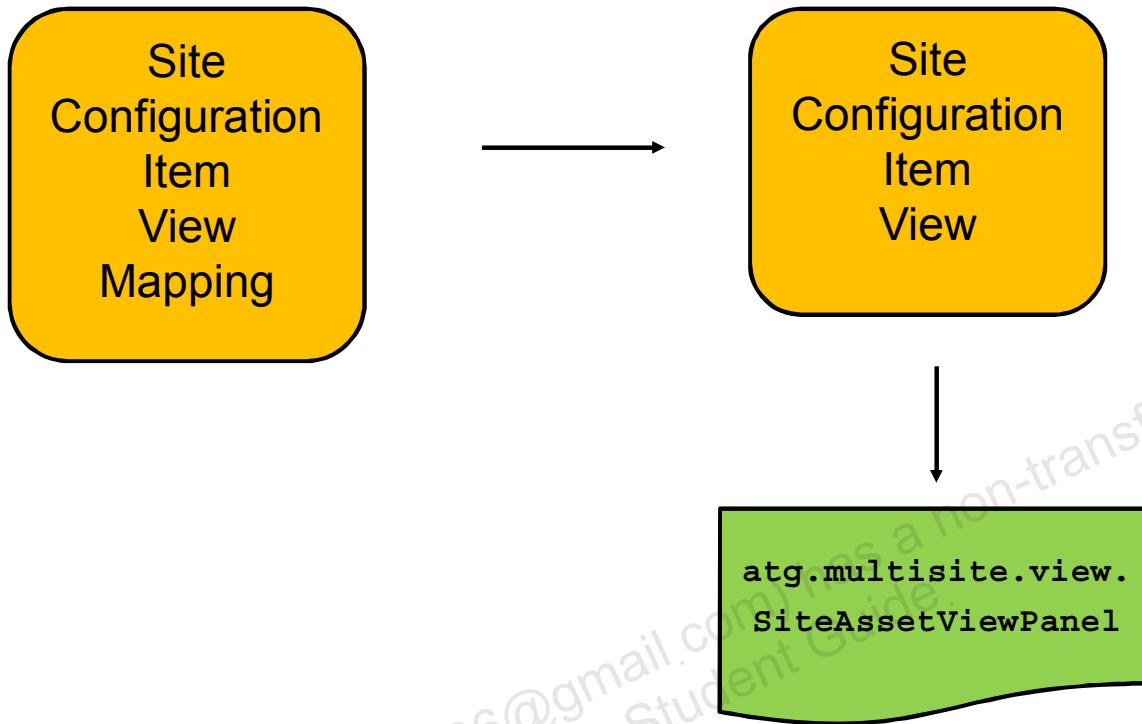


Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

The item mapping for `siteConfigurations` has multiple item view mappings, with one for each tab. The text for the tab name is determined from the item view mapping's properties. The example in the slide follows the Site tab mappings.

Item View Mapping to Item View

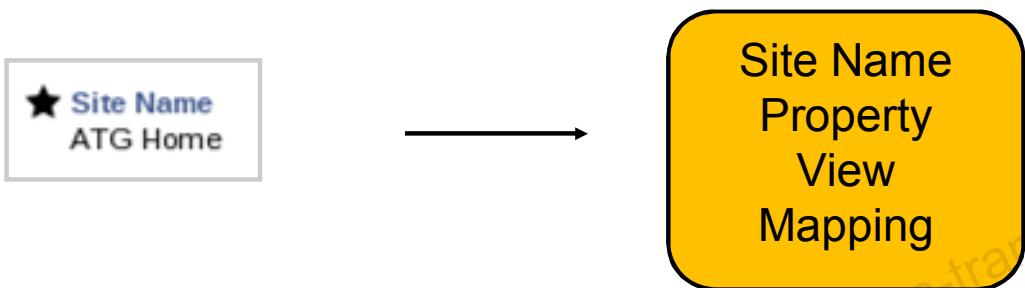


ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The item view mapping is linked to an item view repository item, which points to a Flex element.

Property View Mapping

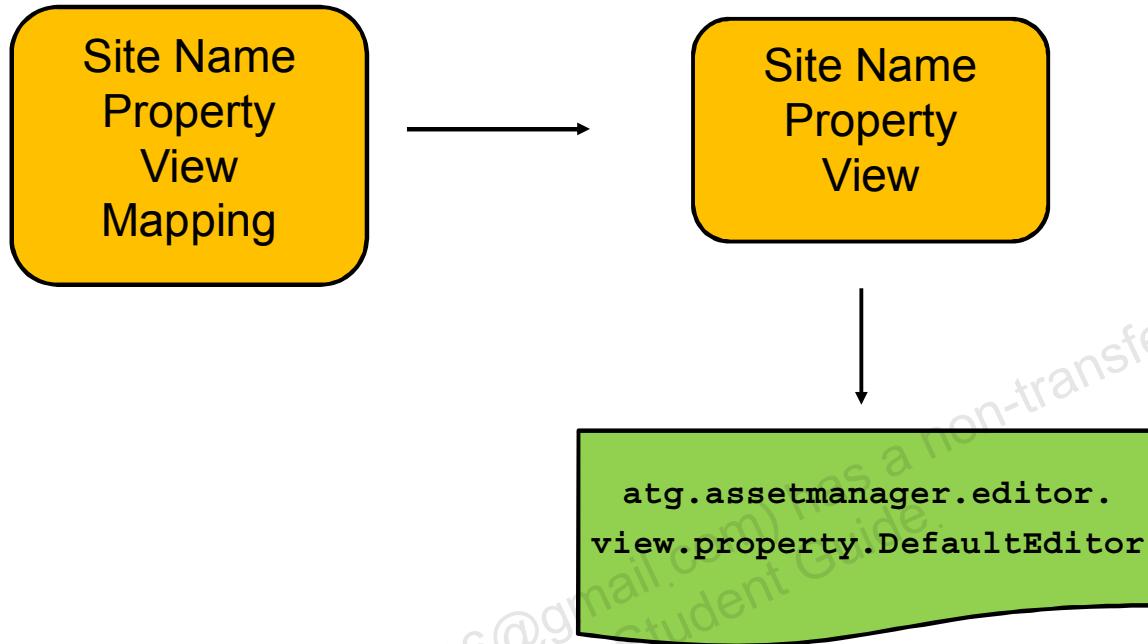


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

For each property to be rendered, the applicable property view mapping is determined. `siteName` has a specific property view mapping, rather than using the default for text properties.

Property View



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The property view mapping is linked to a property view repository item, which points to a Flex element.

For More Information

- *ATG Commerce Programming Guide*
 - Configuring Commerce for Multisite
- *ATG Programming Guide*
 - Multisite Request Processing
- *ATG Multisite Administration Guide*



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

Ganesan Sree (ganesan186@gmail.com) has a non-transferable
license to use this Student Guide.

Advanced Promotions Topics

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this appendix, you should be able to:

- Import and export promotions
- Extend PMDL
- Describe the architecture of:
 - Gift with Purchase (GWP) promotions
 - Stacking rules
 - Application messaging with GWP promotions and stacking rules
- Implement the `CalculatorInfoProvider` interface



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

- Importing and exporting promotions
- Extending PMDL
- GWP promotions
- Stacking rules
- Application messaging
- CalculatorInfoProvider interface

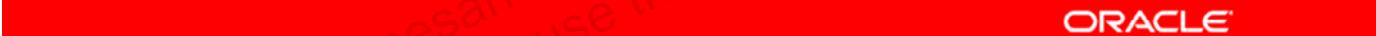


ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Promotion Import and Export API

- ATG Commerce is often integrated with other systems that are involved with pricing and promotions.
- When ATG Commerce is the system of record, promotions need to be exported to other systems.
- Conversely, when ATG Commerce is not the system of record, promotions need to be imported to ATG Commerce.
- An API provides methods to make writing integration code easier.
 - The custom integration code itself is not provided.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Import and Export Templates

- Promotions can be imported using a template.
- Promotions are imported by specifying the relevant template and required placeholder values.
- Template may or may not be available to end-users.
 - The `available-in-ui` property can be set to false to hide templates in the BCC.
 - Imports may or may not be available for editing in BCC.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Imports can also be done without a template, although this makes the import code more complex.

Importing and Content Administration

- Importing may be done on either the production or Content Administration cluster.
- Imports done on the production cluster go live immediately.
- Imports done on the CA cluster are automatically done in context of a CA project.
- The project is created and published by the PublishingWorkflowAutomator component.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The /atg/epub/PublishingWorkflowAutomator component is configured in the Publishing.base module.

Configuring PublishingWorkflowAutomator Properties

- `userName`
 - The default is publishing.
- `workflowName`
 - The default is /Content Administration/import-review.wdl.
 - import-review.wdl does not exist out of the box. You can create a custom workflow or use CA.wdl.
- `taskOutcomeId`
 - The default is 4.1.1.
- `projectNameStub`
 - The default is Promotion Import.
 - The date and time are appended to this stub to name each import project.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `taskOutcomeId` property should be set to the ID of the workflow task to be triggered when the import is finished.

PromotionImportExport Component

- Is used to import or export promotions
- Includes methods to:
 - Start and end import or export sessions
 - Maintain the status of sessions
 - Import promotions and return the status of individual promotion imports
 - Export promotions by ID or by RQL query
- Includes properties to:
 - Configure the import batch size
 - Add custom code via integrator components



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

RQL stands for Repository Query Language.

PromotionImportExportTools

- The PromotionImportExportTools utility component is used by the PromotionImportExport component.
- If custom promotion types are created, add them to PromotionImportExportTools component's typeCodeToItemDescriptorMap property.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Import Batch Size

- The import batch size is set in `PromotionImportExport.batchSize`.
- The batch size equals the number of promotions.
 - Assets related to a promotion are automatically bundled with the promotion.
- By default, the batch size is 1. Therefore, by default, each transaction includes one promotion and all its related assets.
- A lower batch size eases error recovery. A higher batch size improves performance.
- Recommendation: Leave the default setting if you are importing fewer than 1000 promotions.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Info Classes

- Each promotion and each of its related assets is represented as an “Info” object.
- The info classes are
 - PromotionImportExportInfo
 - ClosenessQualifierImportExportInfo
 - CouponImportExportInfo

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

PromotionImportExportInfo Properties: Part One

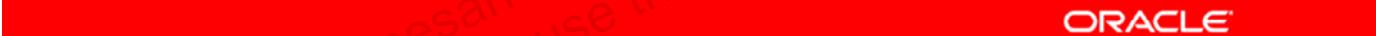
- action indicates whether the promotion is new or existing.
 - It has two possible values: ACTION_ADD or ACTION_UPDATE.
 - It is required; all other properties are optional.
- promotionPropertyValues is a map of strings representing promotion repository item values.
 - The key is the repository property name.
 - It is used when importing without a template or when exporting.
- promotionFolderPath is a string indicating the subfolder for the promotion in BCC.
 - If the specified folders do not exist, they are created automatically.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

PromotionImportExportInfo Properties: Part Two

- `templateId` is the ID of the template to use for importing (a string).
- `templateValues` is a map of strings representing the values to import through the template.
 - The key is the placeholder name.
- `closenessQualifiers` contains a list of `ClosenessQualifierImportExportInfo` objects.
- `coupons` contains a list of `CouponImportExportInfo` objects.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ClosenessQualifierImportExportInfo Properties

- action has three possible values: ACTION_ADD, ACTION_UPDATE, or ACTION_DELETE.
 - If the promotion action is ACTION_ADD, you may use only ACTION_ADD.
 - It is required. All other properties are optional.
- closenessQualifierPropertyValues is a map of strings representing closeness qualifier repository item values.
 - The key is the repository property name.
- closenessQualifierTemplateValues is a map of strings representing values to import through template.
 - The key is the placeholder name.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

CouponImportExportInfo Properties

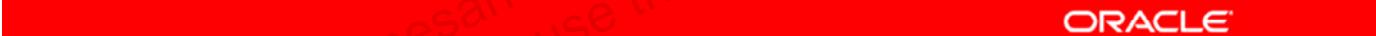
- **action** has three possible values: ACTION_ADD, ACTION_UPDATE, or ACTION_DELETE.
 - If a promotion action is ACTION_ADD, you may only use ACTION_ADD or ACTION_UPDATE.
 - It is required. All other properties are optional.
- **couponPropertyValues** is a map of strings representing coupon repository item values.
 - The key is the repository property name.
- **couponFolderPath** is a string indicating the subfolder for the coupon in the BCC.
 - If the specified folders do not exist, they are created automatically.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Accessing the PromotionImportExport Component

- From a custom ATG component:
 - Add a `PromotionImportExport` property to the custom component class
 - Specify `/atg/commerce/promotion/PromotionImportExport` as the value in the properties file
- From code outside of ATG, use ATG REST Web Services.
 - For more information about using REST with ATG, see the *Web Services and Integration Framework Guide* and My Oracle Support (support.oracle.com).



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Starting and Ending Import and Export Sessions

```
...get reference to PromotionImportExport component...
PromotionImportExportSession session;
try {
    session =
        pie.startImportExportSession();
...import or export promotions...
} //end try
catch (PromotionException) {
...code to handle errors...
} //end catch
finally {
    try {
        pie.endImportExportSession(session);
    }
    catch (PromotionException){}
} //end finally
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The details about how to import or export promotions are covered in the next few slides. The code in the following slides should be located within the try block shown in this slide, as indicated.

Import Code

```
PromotionImportExportInfo promol = new  
    PromotionImportExportInfo();  
...set values of promotion as needed...  
  
...create related ClosenessQualifierImportExportInfo and  
CouponImportExportInfo objects as needed...  
  
PromotionImportExportStatus status =  
    pie.importPromotion (session,  
        promol)  
...repeat for each promotion...
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The code in the slide should be placed inside the try block shown in slide C-17.

Exporting Promotions by ID

```
List<String> promoIds = new List<String>;
...assemble list of promos to be exported...

List<PromotionImportExportInfo> promoInfos =
pie.exportPromotionsById(promoIds);

...extract values from
PromotionImportExportInfo.promotionPropertyValues as
needed...
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The code in the slide should be placed inside the try block shown in slide C-17.

Exporting Promotions by RQL Query

```
String query = "ALL";  
  
List<PromotionImportExportInfo> promoInfos =  
pie.exportPromotionsByRQLQuery(query);  
  
...extract values from  
PromotionImportExportInfo.promotionPropertyValues as  
needed...
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The code in the slide should be placed inside the try block shown in slide C-17. In the code example in this slide, all the promotions are being exported. Other RQL queries that are more specific could be used. For more information about writing RQL queries, see the *Repository Guide*.

Integrator Components

Integrator components are:

- Used to add custom code to the import or export process
- Based on classes that implement the `PromotionImportExportIntegrator` interface
- Listed in `PromotionImportExport.integrators`
- Called by `PromotionImportExport` in the following methods:
 - `preImportExportSession`
 - `postImportExportSession`
 - `preImportPromotion`
 - `postImportPromotion`



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Integrator Components and PromotionImportExport Methods

Each method of `PromotionImportExport` calls the corresponding method of the same name in every integrator.

- `preImportExportSession` and `postImportExportSession` are passed the `PromotionImportExportSession` object.
- `preImportPromotion` and `postImportPromotion` are passed the session and the `PromotionImportExportInfo` object.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

- Importing and exporting promotions
- Extending PMDL
- GWP promotions
- Stacking rules
- Application messaging
- CalculatorInfoProvider interface



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Extending the PMDL

- The Pricing Model Definition Language (PMDL) DTD makes it possible to add custom information without altering the DTD.
 - A number of “generic” tags make this possible.
- PMDL changes are fully backwards-compatible.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Attribute Tag

- All PMDL elements can have any number of attribute sub-elements with any name and value.
 - Example: `<attribute name="myvariable" value="myvalue" />`
- These name and value pairs could be accessed in custom calculators or custom qualifier filters.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Generic Expression Tags

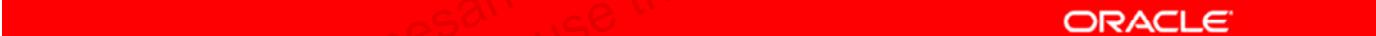
- The following generic expression tags have been added:
 - iterator
 - quantifier
 - operator
 - comparator
- Generic tags are differentiated by the name attribute.
 - Example: <operator name="xor">
- Tags are mapped by name to a custom ElemHandler class.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Creating Custom Expression Tags

1. Create a class to evaluate the expression that subclasses atg.xml.beans.ElemHandler.
2. Add mapping from the name to the custom class in /atg/commerce/pricing/definition/pmdlParser.xml.
3. Use the custom tag in your promotion template.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Example pmdlParser Mapping

```
<pmdl>
  <group name="quantifiers">
    <bean name="my-quantifier"
      class="com.MyStore.MyQuantifier">
      ...map attributes to bean properties...
      <attribute name="myAttribute"
        property="myBeanProperty"/>
      ...map properties to bean properties...
      <property name="collection-name"
        property="myBeanCollectionName"/>
    </bean>
  </group>
</pmdl>
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The full code example with additional properties:

```
<pmdl>
  <group name="quantifiers">
    <bean name="my-quantifier" class="com.MyStore.MyQuantifier">
      <attribute name="myAttribute" property="myBeanProperty"/>
      <property name="collection-name"
        property="myBeanCollectionName"/>
      <property name="element-name"
        property="myBeanElementName"/>
      <property name="element-quantity-property"
        property="myBeanElementQuantityProperty"/>
    </bean>
  </group>
</pmdl>
```

Older ATG Version Expression Tags

- The out-of-the-box expression tags that were available prior to ATG 10.0 can be used with the new syntax or the old syntax.
- Example:
 - `<comparator name="equals">...</comparator>`
 - `<equals>...</equals>`

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Mapping PMDL Elements to Java Objects

- When the PMDL is parsed for evaluation, each tag is represented as a Java object.
- The mapping is done in /atg/commerce/pricing/definition/pmdlParser.xml file.
- This file can be layered like any XML file in the config path.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

- Importing and exporting promotions
- Extending PMDL
- GWP promotions
- Stacking rules
- Application messaging
- CalculatorInfoProvider interface



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Gift with Purchase (GWP) Promotions: Overview

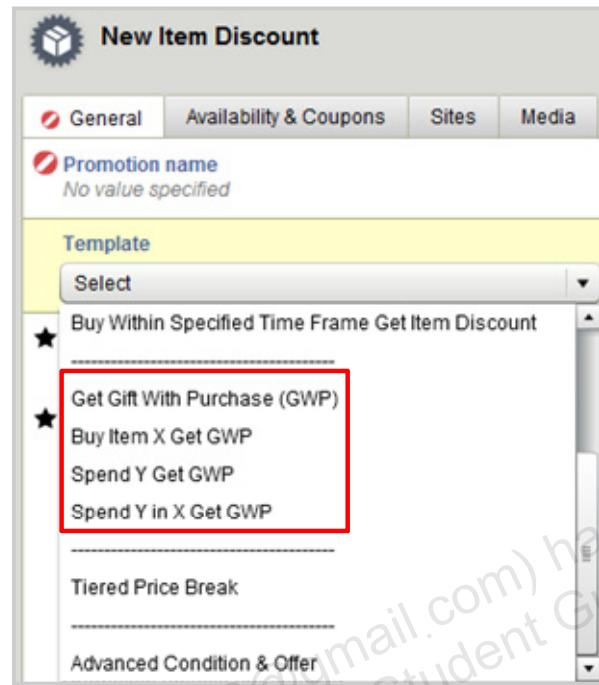
- GWP promotions allow you to give customers a free item as part of their order.
- Free gift criteria can be specified as any of the following asset types:
 - SKU
 - Product
 - Category
 - Content group
- The merchandiser determines what happens to the gift in the cart when the promotion no longer qualifies.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

SKU bundles cannot be used as gift items.

Default GWP Templates



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Example: Spend Y Get GWP Template

The screenshot shows the 'Template' section titled 'Spend Y Get GWP'. Under the 'Condition and offer' tab, the 'Condition' section has a 'Spend' field set to 200. Below it, a note says, 'Any related promotion upsells can be accessed via the "Promotion upsells" editor.' The 'Offer' section lists a 'Gift with Purchase' item: 'Cashmere Scarf' (sku: xsku60325). There are 'Replace...' and 'Clear' buttons. A note below states, 'Gifts are automatically added when a cart meets the specified condition. Gifts already in the cart are repriced as free. Items with a zero price cannot be used as a gift with purchase.' A red box highlights the 'Auto Remove from Cart?' dropdown, which has two options: 'Yes, remove' (selected) and 'No, keep in cart and reprice'. A note below says, 'Only automatically added items can be auto removed.' A callout box points to this section with the text: 'The merchandiser specifies what happens to the free gift in the shopping cart when the customer no longer qualifies for the GWP promotion.'

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The text provided in the template explains what happens to gifts that are already in the cart when the customer qualifies for the GWP promotion (repriced as free). The merchandiser can choose what happens to the gift item when the customer no longer qualifies for the promotion. Items that were automatically added will either be automatically removed or repriced, based on the setting in the template.

Note: Items that were manually added to the cart will be repriced, regardless of the template setting.

Gift Pricing in the Shopping Cart

- If a customer qualifies for a promotion, the gift will either be added to the cart or the customer will choose from gift options.
 - Gifts automatically added to the cart are priced at 0.
 - Gifts selected from the gift options are priced at 0.
- If the gift was manually added to the cart before qualifying for the promotion, and then the customer qualifies, the gift item will be repriced to 0.
- If the customer no longer qualifies, the behavior depends on several factors:
 - Automatically added items will be repriced or removed, based on the promotion's settings.
 - Manually added items will be repriced, regardless of the promotion's settings.

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

GWPManger Component

- The /atg/commerce/promotion/GWPManger component provides the API to manage the GWP promotion functionality to add and remove gifts from the cart.
- Key methods include:
 - processAutoAdd() : Checks whether some quantity of the gift can be automatically added to the cart
 - processGWPInfos() : Processes the given map of GWPInfo objects that was created during item pricing
 - makeGiftSelection() : Makes a single gift selection for a single GWP gift

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Storing Information About GWP Promotions

- `GWPInfo` objects are used for temporary storage of the gift item status for the current item pricing operation.
 - There is an instance of `GWPInfo` for each GWP promotion that qualifies and free gift selection given.
 - They are processed by `GWPManger`.
 - Any information that needs to be persisted is stored in markers against the order.
 - The map of `GWPInfo` objects is cleared for each pricing operation.
- Order markers store the GWP information against an order.
- There is a one-to-one mapping between `GWPInfo` objects and order markers.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

For a given GWP promotion, an instance of the `GWPInfo` class is created for each gift with purchase promotion that qualifies and each free gift selection given. For example, if a promotion gives away one shirt and three items in a GWP category, there will be two `GWPInfo` objects.

Order Repository Extensions: `gwpOrderMarker` and `gwpCommerceItemMarker`

- `gwpOrderMarker` represents a single gift selection.
- `gwpCommerceItemMarker` tracks the quantity of that commerce item that was auto-added, selected, or targeted by a GWP promotion.
- A `gwpOrderMarker` may have more than one `gwpCommerceItemMarker` that refers to it.
 - Example: A GWP promotion offers a free gift of two watches. The shopper selects two different watches, creating two commerce items. Each commerce item has an associated `gwpCommerceItemMarker`, each of which refers to the same `gwpOrderMarker`.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

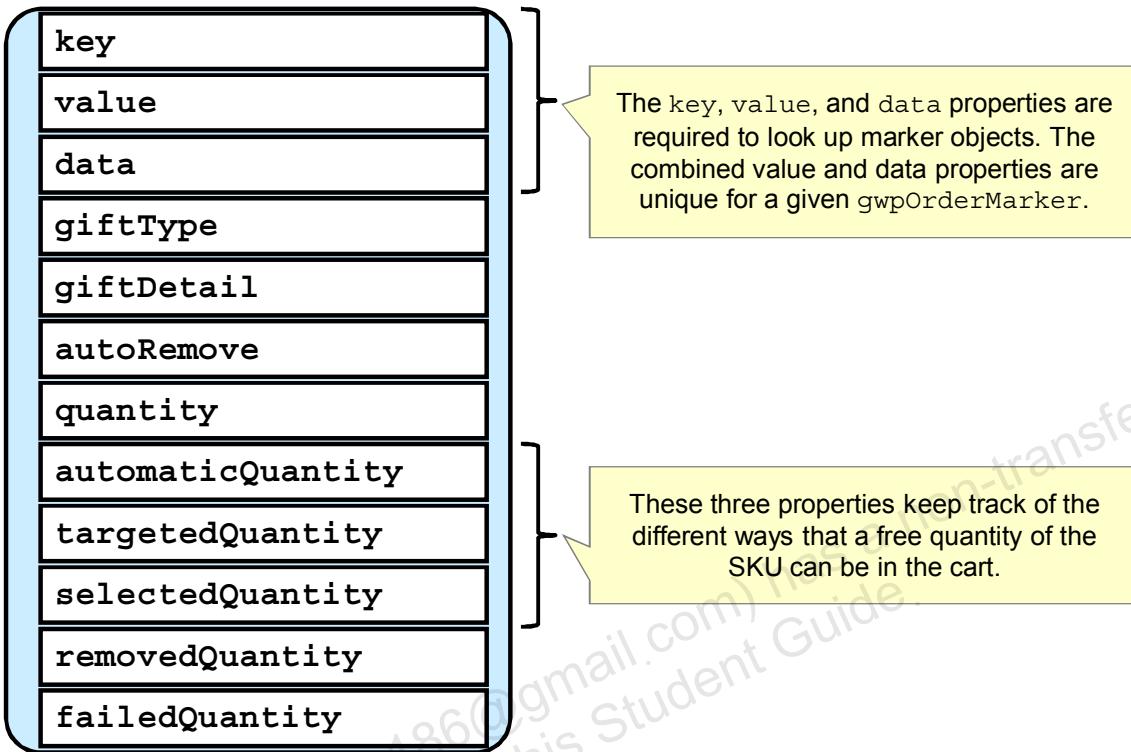
The Gift with Purchase architecture relies on order and commerce item markers. It extends the base marker repository definitions to provide the following item types:

- `gwpOrderMarker`: Each `gwpOrderMarker` represents a single gift selection.
- `gwpCommerceItemMarker`: Each `gwpCommerceItemMarker` tracks the quantity of that commerce item that was auto-added, selected, or targeted by a GWP promotion.

GWP markers only persist while the order still qualifies for the promotion. If the promotion no longer applies, associated markers are removed.

These markers provide a way to keep track of what items have been added by the customer, added automatically, or still need to be added to the order. This mechanism helps to signal when the customer is missing a gift item that needs to be selected, for example.

gwpOrderMarker Item Descriptor



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The gwpOrderMarker object has a number of properties:

- key: This value is always atg.gwp.
- value: The promotionId
- data: Hash code to track which gift selection in the promotion this is. It allows the commerce item marker to link to this marker.
- giftType: The type of gift, which can be SKU, product, category, skuContentGroup, or productContentGroup
- giftDetail: A string identifying the gift. This is usually the repository ID of the gift type.
- autoRemove: A Boolean flag to indicate whether free gifts should be auto removed if the promotion no longer qualifies. This is configured by the merchandiser when the promotion is created.
- quantity: The total quantity of the SKU for this gift selection. This is calculated as the quantity from the PMDL multiplied by the number of times the offer applied (if it applied more than once).
- automaticQuantity: The quantity of the free SKU for this gift selection that has already been auto added to the order.

- `targetedQuantity`: The quantity of the free SKU for this gift selection that has already been targeted and made free by the calculator in the order
- `selectedQuantity`: The quantity of the free SKU for this gift selection that has already been selected by the shopper and added to the order
- `removedQuantity`: The amount of the free quantity that has been manually removed by a shopper. Keeping track of removed quantities prevents them from being automatically re-added in future pricing operations. The assumption is that the customer does not want the free item
- `failedQuantity`: Tracks any gifts that should have been automatically added but failed, for example, due to a site conflict with the promotion

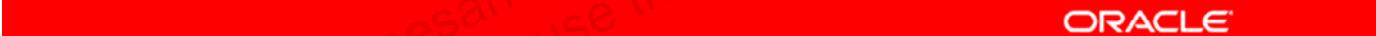
The quantity of SKU the shopper still needs to select via a placeholder item can be determined via the following expression:

$$\text{quantity} - (\text{automaticQuantity} + \text{targetedQuantity} + \text{selectedQuantity} + \text{removedQuantity} + \text{failedQuantity})$$

The `gwpCommerceItemMarker` repository item has similar properties. For more details about the properties of `gwpCommerceItemMarker`, see the section titled “Gift with Purchase Promotions” in Chapter 12: Understanding Promotions of the *ATG Commerce Programming Guide*.

giftWithPurchase Event Attribute

- When a new commerce item is added or removed or the quantity is changed due to gift with purchase selections, the following events are used to convey the change information:
 - ItemAddedToOrder
 - ItemRemovedFromOrder
 - ItemQuantityChanged
- These events have a giftWithPurchase Boolean flag that is set to true when the event fires due to gift with purchase actions.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Pricing with the GWPManager

- The pricing operation is controlled by the PricingTools component, which is responsible for:
 - Calling GWPManager at the end of item pricing
 - Restarting pricing if the GWPManager believes it is necessary
- The PricingTools.priceItemsForOrderTotal() method has been extended to call GWPManager's processGWPInfos() method after it calls the item pricing engine's priceItems() method.
 - If processGWPInfos() returns true, priceItemsForOrderTotals() loops around to start again.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The GWPManager.processGWPInfos() method might return true. In that case, the priceItemsForOrderTotal() method needs to run again. For example, if an item in the cart is no longer targeted to be free, this change of price could mean other item promotions might be affected.

GWP Droplets and Form Handlers

Two droplets and a form handler help page developers configure the gift selection functionality in the shopping cart page:

- `GiftWithPurchaseSelectionsDroplet` retrieves all GWP details on an order or for a single commerce item.
- `GiftWithPurchaseSelectionChoicesDroplet` retrieves all of the products and SKUs available for selection for a GWP gift.
- `GiftWithPurchaseFormHandler` is used to submit a gift selection for a GWP promotion.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The three components listed in the slide are located in the `/atg/commerce/promotion/Nucleus` name space.

The droplets can be used in conjunction with one another. Where `GiftWithPurchaseSelectionsDroplet` returns the gift selection details (including quantities available), `GiftWithPurchaseSelectionChoicesDroplet` will translate the quantities available (by using the gift type and detail) into actual products and SKUs, such as a hat or scarf.

GiftWithPurchaseSelectionsDroplet

- **Input parameters:**
 - `order` (required): The order to be inspected (required)
 - `item`: The commerce item to be inspected. If specified, the selections apply only to this item.
 - `onlyOutputAvailableSelections` (Boolean): For order inspections, if true then only selections where `quantityAvailableForSelection` is > 0 will be output. Defaults to false.
- **Output parameters:**
 - `selections`: A collection of `GiftWithPurchaseSelection` beans
 - `errorMsg`
- **Open parameters:** `output`, `empty`, and `error`



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `atg.commerce.promotion.GiftWithPurchaseSelection` bean represents a single gift selection available for a GWP promotion. Its properties are similar to the properties in the order marker for that gift. The `GiftWithPurchaseSelectionsDroplet` will output a collection of these for an order. It contains the various quantities associated with the promotion and gift in the order or item.

For more information about the droplet and the `GiftWithPurchaseSelection` bean, see the *API Platform Reference*.

GiftWithPurchaseSelectionChoicesDroplet

- Input parameters:
 - selection: A `GiftWithPurchaseSelection` bean for the free gift. If not specified, the `giftType` and `giftDetail` parameters must be supplied.
 - `giftType`
 - `giftDetail`
 - `alwaysReturnSkus` (Boolean): If `true`, the choices returned will always include the SKUs. Defaults to `false`.
- Output parameters:
 - `choices`: A collection of `GiftWithPurchaseSelectionChoice` beans
 - `errorMsg`
- Open parameters: `output`, `empty`, and `error`



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `GiftWithPurchaseSelectionChoice` bean holds GWP promotion selection choices. It holds a product and, possibly, the child SKU items available for selection. By default, if there is a single SKU or SKU content group gift type, the SKU will be returned. Otherwise, only the products will be returned. The droplet provides a Boolean input parameter called `alwaysReturnSkus` which defaults to `false`. If it is set to `true`, the choices returned will always include SKUs. It also ensures that SKUs will be populated for product and category `giftType`.

The choices are output as a collection of `atg.commerce.promotion.GiftWithPurchaseSelectionChoice` beans. The properties of the beans are:

- `product` (`RepositoryItem`): A valid product choice
- `skus`: (Collection of `RepositoryItem`): A collection of valid child SKUs (only provided for SKU content groups)

Note: The `skus` property will be null unless the `giftType` input or the `giftType` in the `GiftWithPurchaseSelection` bean is a SKU content group. For other gift types, it can be assumed that all of a product child SKUs are available for selection and do not need to be provided here.

GiftWithPurchaseFormHandler

- Key properties:
 - currentSelectedItemId
 - promotionId
 - giftHashCode
 - quantity
 - productId
 - skuId
- Redirects:
 - makeGiftSelectionSuccessURL
 - makeGiftSelectionErrorURL
 - sessionExpirationURL
- Submit handler: makeGiftSelection



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The /atg/commerce/promotion/GiftWithPurchaseFormHandler has a number of properties you can set in the page. For the full list, see Chapter 5: Creating and Maintaining Promotions in the *ATG Commerce Guide to Setting Up a Store*.

Note: Although the documentation discusses the *GWPFormHandler*, the actual name is *GiftWithPurchaseFormHandler* and the class is `atg.commerce.promotion.GiftWithPurchaseFormHandler`.

The `giftSelection.jsp` gadget provides an example of using *GiftWithPurchaseFormHandler* in the Commerce Reference Store (from `CommerceReferenceStore's store.war` file in the `\cart\gadgets` folder).

The form handler includes the following configurable properties:

- `currentSelectedItem`: Specify an item ID here to replace an existing gift selection
- `promotionId`: The GWP promotion for which to make the selection
- `giftHashCode`: The gift item within the promotion for which to make the selection
- `quantity`: Amount of the new selection
- `productId`: Product ID for the new selection
- `skuId`: SKU ID for the new selection

The form handler also includes two flags that control whether a new gift selection should first replace a removed or failed quantity before replacing a selected item:

- `replaceRemovedQuantity`
- `replaceFailedQuantity`

Both flags are `false` by default.

How the Droplets and Form Handler Work Together

The process flow is typically as follows:

1. Use the `GiftWithPurchaseSelectionsDroplet` to get all the GWP gift details, with one `GiftWithPurchaseSelection` bean per gift.
2. For each `GiftWithPurchaseSelection` bean that still has quantity to be selected (`selectedQuantity > 0`): When the shopper chooses to select their gift, call into the `GiftWithPurchaseSelectionChoicesDroplet` to get the possible gift options for that gift.
3. After the shopper chooses the gift options, call into the `GiftWithPurchaseFormHandler` to make the selection happen.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Commerce Reference Store GWP Extensions

The Commerce Reference Store extends the GWP architecture to add functionality:

- Extensions to the `GWManager.processAutoAdd()` method to handle gifts that are out of stock
- Logic needed to handle gifts appearing on separate lines in the cart



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Road Map

- Importing and exporting promotions
- Extending PMDL
- GWP promotions
- Stacking rules
- Application messaging
- CalculatorInfoProvider interface



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Stacking Rules: Overview

- Stacking rules prevent customers from taking advantage of unintended synergy among your promotions.
- Each stacking rule asset includes one or more promotions that you want to treat as a group.
- Each promotion can only be associated with a single stacking rule.
- You can specify the maximum number of promotions an order can contain from a stacking rule.



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Each stacking rule asset includes one or more promotions that you want to treat as a group. The rules are intended to prevent simple combination strategies, for example, to keep customers from combining BOGO offers with Gift with Purchase offers.

Each promotion can only be associated with a single stacking rule. Each stacking rule can include any number of promotions up to the number set in the “Maximum promotions per stacking rule” property. Within a stacking rule, promotions are applied to orders according to the promotion’s priority.

Stacking rules have the following behavior:

- If an order qualifies for more than one promotion, and those promotions are linked to mutually exclusive stacking rules, the promotion with the highest priority is applied, and all other promotions are skipped.
- If you specify that Stacking Rule X is not combinable with Stacking Rule Y, Y cannot be combined with X.
- If you change the combine list on one stacking rule, the affected stacking rule is automatically updated.
- Any promotion that is linked to a stacking rule that is excluded from use with the current stacking rule cannot be applied.

Pricing Engine Maps Related to Stacking Rules and Included and Excluded Promotions

A number of maps support the evaluation and tracking of stacking rules in the `PricingEngineService` class:

Map	Key	Value
<code>includedPromotions</code>	Included promotion	Stacking rule of the included promotion
<code>excludedPromotions</code>	Excluded promotion	Excluding promotion
<code>excludedStackingRules</code>	Excluded stacking rule	Excluding promotion
<code>stackingRuleUsage</code>	Stacking rule	Number of uses
<code>stackingRuleAppMessages</code>	ID of excluded promotion	Application message



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

Stacking Rule Tracking Information

The evaluation of stacking rules is supported by a number of maps:

- `includedPromotions`: Promotions that can be applied (with the promotion currently being evaluated)
- `excludedPromotions`: Promotions that should be prevented from being applied (with the promotion currently being evaluated)
- `excludedStackingRules`: Any promotion that belongs to a stacking rule that is in this map should be prevented from being applied.
- `stackingRuleUsage`: Tracks the usage of each stacking rule, which is in turn used to determine if a promotion should be prevented from being applied
- `stackingRuleAppMessages`: A map of application messages relating to those promotions that have been excluded and would normally have been applied

Pricing Engine Updates: `setupStackingRuleCollection()` Method

- The key method of the `PricingEngineService` class is:
`setupStackingRuleCollections(PricingContext pPricingContext, java.util.Map pExtraParameters)`
- The method sets up the stacking rule info collections:
 1. The collections are created, if they do not exist.
 2. The collections are populated with information related to previously applied promotions.
 3. Messages relating to previously applied promotions are generated.
- Tracking collections are created in the `pExtraParameters` map in order to be made available to all pricing engines.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The `pExtraParameters` map is made available to all pricing engines, because it is possible for an exclusion to be set in the item pricing engine that takes effect in the order pricing engine. You can learn more about the `atg.commerce.pricing.PricingEngineService` class in the *API Platform Reference*.

DID_PROMOTION_APPLY Parameter

pExtraParameters has an important parameter called DID_PROMOTION_APPLY.

- A Boolean used by the pricing engine to determine if a promotion applied, even if no pricing adjustment was made (such as can happen with a Gift with Purchase promotion)



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

PricingEngineService Methods Related to Stacking Rules

- The PricingEngineService class has three methods that pertain to stacking rules:
 - didPromotionApply: Determines if a particular promotion has been applied, based on a pricing adjustment
 - updateStackingRuleTracking: Updates the stacking rule tracking information, such as exclusion details
 - wasPromotionSkipped: Determines if a promotion should be skipped, based on the tracking details
- To invoke the stacking rule tracking map updates, one of the following must be true:
 - A pricing adjustment has been applied
 - pExtraParameters.DID_PROMOTION_APPLY=true

ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Pricing Engines and the applyPromotions () Method: Part One

The applyPromotions () method in each pricing engine iterates over each promotion:

- If stacking rule evaluation is enabled, it calls wasPromotionSkipped (), which evaluates the stacking rule details to see if the promotion was excluded.
 - If true, an application message may be sent and no further processing within applyPromotions () occurs for this promotion.
 - If false, a different method call evaluates the conditions and offer.
 - If the promotion qualifies, calculators are applied.
 - If it can determine that the promotion applied, DID_PROMOTION_APPLY is set to true.
 - If not, DID_PROMOTION_APPLY remains null.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The methods will be called from within the applyPromotions () method of:

- ItemPricingEngineImpl
- OrderPricingEngineImpl
- ShippingPricingEngineImpl
- TaxPricingEngineImpl

The applyPromotions () method in each pricing engine iterates over each promotion. It first calls wasPromotionSkipped () (if the stacking rule evaluation is enabled), which evaluates the stacking rule details to see if this promotion was excluded. If wasPromotionSkipped () returns true, an application message may be sent and no further processing within applyPromotions () occurs for this promotion. If wasPromotionSkipped () returns false, a different method call evaluates the conditions and offer. If the promotion qualifies, calculators are applied. If it can determine that the promotion applied, DID_PROMOTION_APPLY will be set to true. If not, DID_PROMOTION_APPLY remains null.

Pricing Engines and the applyPromotions () Method: Part Two

At the end, the applyPromotions () method checks for the presence of the DID_PROMOTION_APPLY parameter in the pExtraParameters map.

1. If DID_PROMOTION_APPLY=true, it calls the updateStackingRuleTracking () method.
2. If DID_PROMOTION_APPLY is null, it determines if the promotion applied by checking each item.
 - It calls the didPromotionApply () method each time.
 - If didPromotionApply () returns true, it calls the updateStackingRuleTracking () method.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

At the end of applyPromotions (), it checks for the presence of the DID_PROMOTION_APPLY parameter in the pExtraParameters map. If DID_PROMOTION_APPLY=true, it calls the updateStackingRuleTracking () method. If DID_PROMOTION_APPLY is null, it determines if the promotion applied by calling didPromotionApply () on each item. If didPromotionApply () returns true, it calls the updateStackingRuleTracking () method.

The goal is to avoid calling the didPromotionApply () method on every item of a promotion unless you really need to do so.

Additional Stacking Rules Information

- Stacking rule logic will not be invoked if the pExtraParameters map is null.
- You can bypass stacking rule evaluation logic by setting the Java constant /atg/commerce/pricing/Constants.DISABLE_STACKING_RULE_PROCESSING in pExtraParameters.



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Within each pricing engine implementation (item, shipping, order, tax), if the call to applyPromotions() is passed a null value for the pExtraParameters map, the stacking rule logic will not be invoked. The map is required in order to pass the tracking maps between engines.

Road Map

- Importing and exporting promotions
- Extending PMDL
- GWP promotions
- Stacking rules
- Application messaging
- CalculatorInfoProvider interface



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

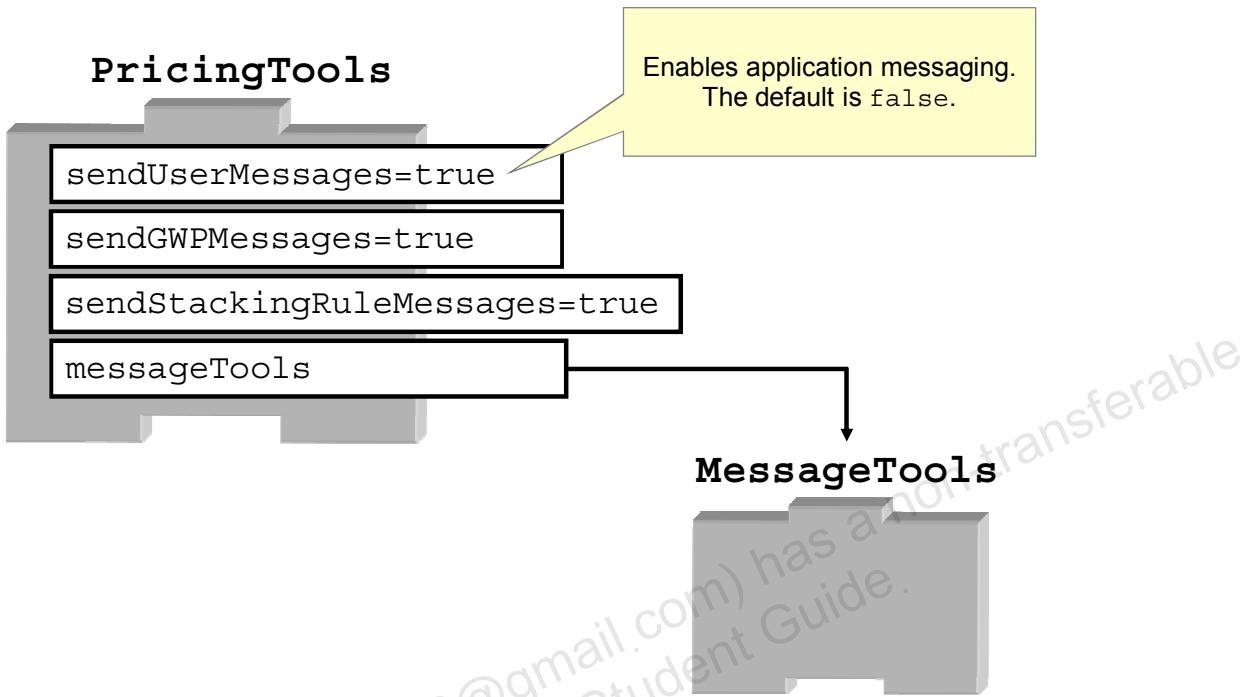
Application Messaging: Stacking Rules and GWP

- ATG Commerce provides a messaging system that lets you configure multiple message types: slots and JMS.
- This messaging system is used by GWP promotions and stacking rules to communicate with customers using slots.
- You enable application messaging in PricingTools.

 ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

PricingTools Messaging Properties



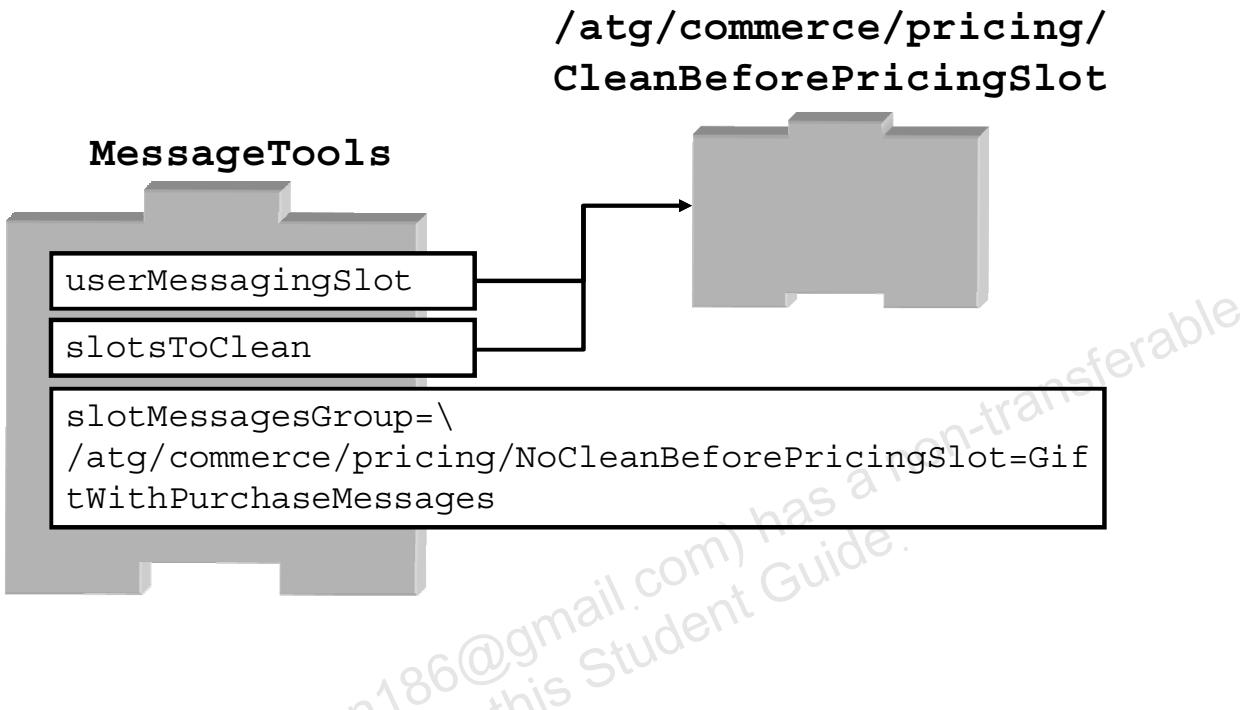
Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

The `/atg/commerce/pricing/PricingTools` property's `sendGWPMessages` and `sendStackingRuleMessages` properties are both set to `true` by default in the code. To enable application messaging, set `sendUserMessages` to `true`. This will enable messaging for GWP promotions and stacking rules as well. The `messageTools` property points to the `/atg/commerce/pricing/MessagingTools` component. `MessagingTools` includes a `sendUserMessage()` method that resolves the `MessageTools` component and sends the message to it.

For more information about application messaging, see the “Application Messaging” section in Chapter 16: Customizing Purchase Process Externals of the *ATG Commerce Programming Guide*. That section covers slot messages, JMS messages, and how to extend the framework.

MessageTools Properties



ORACLE

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The `userMessagingSlot` property is your default message slot and is the default for stacking rule messages. By default, this property is set to `CleanBeforePricingSlot`. Messages are not removed from this slot as they are retrieved. However, by default, the slot is configured in the `slotsToClean` property, which holds the list of slots to be cleared automatically at the start of each pricing operation.

The `slotMessagesGroup` property maps slot components to a user message group string. If a user group mapping is added, any messages with that message group are sent to the slots configured in the map. By default, this property specifically routes all GWP messages to the `NoCleanBeforePricing` slot. Messages are removed from this slot as they are retrieved.

Messages

The message takes the form of a UserMessage, which is part of the messaging framework. Key properties include:

- identifier
- priority: 10, 5, 0, -5 or -10
- type: error, confirmation, warning, or information
- params

Example GWP message

Property	Value
identifier	GWPQualified
priority	-10
type	information
params	[promotion ID]

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

The properties of UserMessage are:

- identifier (String): The unique message identifier
- summary (String): A string description for the message (not shown in the slide or message example)
- priority (int): An integer priority, Validation=10, Form Error (not validation)=5, Error (default)=0, Confirmation=0, Warning=-5, Information=-10
- type (String): Message type: error (default), confirmation, warning, information
- params (Object[]): Optional message-specific array of object parameters that relate to the message
- messageGroup (String): A group to which this message belongs, such as GiftWithPurchaseMessages (not shown in the slide or message example)

Road Map

- Importing and exporting promotions
- Extending PMDL
- GWP promotions
- Stacking rules
- Application messaging
- CalculatorInfoProvider interface



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

When to Implement CalculatorInfoProvider

When creating a pricing calculator class, you may need to implement the `CalculatorInfoProvider` interface.

- Is necessary if your custom calculator requires complex input
- Requires a single method, `getCalculatorInfo`, which returns a `CalculatorInfo` object (see the next slide for details)
- Is used to render the BCC UI and create PMDL



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The CalculatorInfo Object

- The CalculatorInfo object models the PMDL discount-structure and discount-detail attributes used for a particular discount type.
- Out of the box, it is used for volume discounts.
- Its calculatorType property should be set to match the key used in the pricing engine's calculatorTypeCalculators property.

ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

Constructing a CalculatorInfo Object

```
CalculatorInfo info = new CalculatorInfo(pCalculatorType);
DiscountAttributeInfo structAtts[] =
new DiscountAttributeInfo[2];
structAtts[0] = new DiscountAttributeInfo
(BANDING_PROPERTY_ATTRIBUTE);
structAtts[0].setDataType(String.class);
structAtts[0].setRequired(false);
...
info.setDiscountStructureAttributeInfos(structAtts);
DiscountAttributeInfo atts[] = new
DiscountAttributeInfo[2];
atts[0] = new DiscountAttributeInfo(BAND_ATTRIBUTE);
atts[0].setDataType(Number.class);
atts[1] = new DiscountAttributeInfo(ADJUSTER_ATTRIBUTE);
atts[1].setDataType(Double.class);
info.setDiscountDetailAttributeInfos(atts);
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

The code in the slide is from `atg.commerce.pricing.BandingDiscountCalculatorHelper`, which constructs the `CalculatorInfo` objects for all the volume discount calculators. The `BANDING_PROPERTY_ATTRIBUTE` constant is set to `bandingProperty`. The `pCalculatorType` variable, passed in by the calling calculator, is always `bulk`. `BAND_ATTRIBUTE` is set to `band`, and `ADJUSTER_ATTRIBUTE` to `adjuster`.

Extracting Values from Custom Attributes

Values of custom PMDL discount attributes can be retrieved using the key defined in the `CalculatorInfo` object.

```
attributes = discountStructure.getAttributes();  
  
bandingProperty = (String) attributes.get  
(BANDING_PROPERTY_ATTRIBUTE);
```



Copyright© 2013, Oracle and/or its affiliates. All rights reserved.

This example in the slide mirrors the example given earlier for setting a custom attribute name for the discount structure in the `CalculatorInfo` object.

Summary

In this lesson, you should have learned how to:

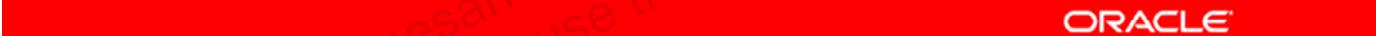
- Import and export promotions
- Extend PMDL
- Describe the architecture of
 - Gift with Purchase (GWP) promotions
 - Stacking rules
 - Application messaging with GWP promotions and stacking rules
- Implement the `CalculatorInfoProvider` interface



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

For More Information

- *ATG Commerce Guide to Setting Up a Store*
 - Creating and Maintaining Promotions
- *ATG Commerce Programming Guide*
 - Understanding Promotions
 - Using and Extending Pricing Services
- *API Platform Reference*
- *ATG Merchandising Guide for Business Users*
 - Creating and Managing Promotions and Coupons



ORACLE®

Copyright© 2013, Oracle and/or its affiliates. All rights reserved.