



Integrated Cloud Applications & Platform Services

Java SE 7: Develop Rich Client Applications

Student Guide - Volume II

D67230GC10

Edition 1.0 | December 2016 | D77354

Learn more from Oracle University at education.oracle.com

Authors

Michael J. Williams
Paromita Dutta
Anjana Shenoy
Cindy Church

Technical Contributors and Reviewers

Debra Masada
Ajay Bharadwaj
Aurelio Garcia-Ribeyro
Steve Watt
Nancy Hildebrandt
Alla Redko
Tom McGinn
Matt Heimer
Sharon Zahkour
Nicolas Lorain
Brian Goetz
Joe Darcy
Alessandro Leite
Jai Suri
Richard Bair
Jasper Potts

Editors

Daniel Milne
Anwesha Ray
Richard Wallis

Graphic Designer

Seema Bopaiyah

Publishers

Pavithran Adka
Durga Ramesh

Copyright © 2012, 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 to the Course

- Course Objectives 1-2
- Audience 1-4
- Prerequisite Skills 1-5
- Use JavaFX to Create Rich-Client Applications 1-6
- Rich-Client Application 1-7
- Course Roadmap 1-8
- Schedule 1-9
- Course Environment 1-12
- Quiz 1-14
- Facilities in Your Location 1-15
- Summary 1-16

2 Introduce Rich Client Applications

- Objectives 2-2
- BrokerTool Application Overview 2-3
- BrokerTool Application Customer Problem Statement: Requirements 2-4
- BrokerTool Application Customer Problem Statement: Resources & Constraints 2-5
- BrokerTool Application: Customer Table 2-6
- BrokerTool Application: Shares Table 2-7
- BrokerTool Application: Stock Table 2-8
- BrokerTool Application: Broker Table 2-9
- Your Assignment 2-10
- Two-Tier Application Design 2-11
- BrokerTool Application Design: Three-Tier 2-12
- HenleyApp Car Sales Application 2-13
- Summary 2-14
- Lesson 2 Practice Overview: Set up environment and use BrokerTool 2-15

3 JavaFX

- Objectives 3-2
- Topics 3-3
- JavaFX Features 3-4
- Overview of JavaFX in the Enterprise 3-5
- Swing 3-6

Topics	3-7
Stage and Scene	3-8
JavaFX Scene Graph: Stage and Scene	3-9
Stage, Scene, and Nodes	3-10
JavaFX Scene Graph: Nodes Hierarchy	3-11
JavaFX Scene Graph	3-12
Rich Client Application	3-13
Minimum JavaFX Application	3-14
JavaFX Application Types	3-15
Simple JavaFX Application	3-16
JavaFX Java Class	3-17
JavaFX FXML	3-18
JavaFX FXML Application Format	3-19
JavaFX FXML Application Format: JavaFXFXMLApplication.java	3-20
JavaFX FXML Application Format: Sample.fxml	3-21
JavaFX FXML Application Format: Sample.java	3-22
JavaFX Preloader	3-23
Quiz	3-24
Practice 3: Overview	3-26
Topics	3-27
Scene Graph API	3-28
Swing Containment Hierarchy	3-30
Download and Install JavaFX	3-31
JavaFX Default Installation Directory	3-32
JavaFX API Documentation	3-34
Download JavaFX Samples	3-35
Resources	3-36
Summary	3-38

4 Generics and JavaFX Collections

Objectives	4-2
Topics	4-3
Generics	4-4
Simple Cache Class Without Generics	4-5
Generic Cache Class	4-6
Generics in Action	4-7
Generics with Diamond Operator	4-8
ArrayList Without Generics	4-9
Generic ArrayList	4-10
Quiz	4-11
Topics	4-12

JavaFX Collections and Interfaces 4-13
ObservableList 4-14
ObservableMap 4-15
FXCollections Classes 4-16
ObservableList from Strings 4-17
ObservableMap from Strings 4-18
Bid Example: Demo 4-19
Quiz 4-20
Summary 4-21
Practice 4: Overview 4-22

5 UI Controls, Layouts, Charts, and CSS

Objectives 5-2
Topics 5-3
JavaFX Scene Graph and UI Elements 5-4
Topics 5-5
UI Controls 5-6
Control Is a Node 5-7
Henley Car Sales Button 5-9
Button Created in HenleyClient.fxml 5-10
Button Example in a Java Class 5-11
Tooltip 5-12
Images and Shapes 5-13
Imageview Is a Node 5-14
Henley Car Sales Image 5-15
Image Created in HenleyClient.fxml 5-16
Shape Is a Node 5-17
Clock Example Uses Circles 5-18
Clock Example: Circles 5-19
Group Is a Node 5-20
Group of Circles 5-21
Layout Containers 5-22
A Layout Container Is a Node 5-23
Types of Layout Containers 5-24
StackPane 5-25
BorderPane 5-26
HBox 5-27
VBox 5-28
GridPane 5-29
Henley Car Sales Form Layout 5-30
Resizability 5-32

Quiz 5-34
Topics 5-35
Skinning UI Controls 5-36
CSS in Henley Sales Application 5-38
CSS Syntax 5-39
Topics 5-40
Events in JavaFX 5-41
Button Event 5-42
Choice Box 5-43
Listener 5-44
Topics 5-45
Charts 5-46
Chart Is a Node 5-47
Chart Example in a Java Class 5-48
Chart Example in FXML 5-49
Defining the X Axis and Y Axis 5-50
Chart Data 5-51
Apply Effects 5-52
Style Charts 5-53
Animate Charts 5-54
Quiz 5-55
Summary 5-56
Practice 5: Overview 5-57

6 Visual Effects, Animation, WebView, and Media

Objectives 6-2
Topics 6-3
Animation: Timelines and Transitions 6-4
Using Animation in JavaFX: Introduction to Computer Animation 6-5
Animation Is Applied to a Node 6-7
Timeline Animation 6-8
Using the Transition Classes 6-9
Animated Transition: Path Transition 6-10
Effects 6-11
Effect Is Applied to a Node 6-12
Effect: Example 6-13
Quiz 6-14
Topics 6-15
Media 6-16
Building a Media Player 6-17
MediaView Is a Node 6-18

MediaView	6-19
Topics	6-20
WebView	6-21
WebView Is a Node	6-22
WebView: Example	6-23
Use Cases for WebView	6-24
Quiz	6-25
Summary	6-26
Practice 6: Overview	6-27

7 JavaFX Tables and Client GUI

Objectives	7-2
Topics	7-3
Tables in JavaFX	7-4
TableView is a Node	7-5
Creating a Table	7-6
Creating a Table: Code Example	7-7
TableCell<S,T>	7-8
Cell<T>	7-9
Cell Factory	7-10
Custom Cell Factory: FXML Example	7-11
Custom Cell: Example	7-12
Table Data Model	7-13
Topics	7-15
Benefits of Using CSS with Tables	7-16
CSS and Tables	7-17
Topics	7-19
BrokerTool Application	7-20
JavaFX Development Practices	7-21
Application in MVC Terms	7-22
Login Window	7-23
Main Window	7-24
Order Form Window	7-25
Quiz	7-26
Summary	7-27
Practice 7: Overview	7-28

8 JavaFX Concurrency and Binding

Objectives	8-2
Topics	8-3
Concurrency and JavaFX	8-4

Worker Interface	8-5
Task Class	8-6
Example: Create the Task	8-7
Example: Run the Task	8-8
Service Class	8-9
Service Class Execution	8-10
Example: Create the Service	8-11
Example: Run the Service	8-12
Topics	8-13
Data Binding in JavaFX	8-14
Quiz	8-16
Summary	8-17
Practice 8: Overview	8-18

9 Java Persistence API (JPA)

Objectives	9-2
Topics	9-3
Java Persistence API: Overview	9-4
Benefits of Using JPA	9-5
Features of JPA	9-6
Topics	9-7
JPA Components	9-8
Object-Relational Mapping (ORM)	9-9
JPA Entities	9-11
Creating an Entity	9-12
Entity Mapping	9-14
Primary Keys	9-15
Entity Component Primary Key Association	9-16
Overriding Mapping	9-17
Transient Fields	9-18
Persistent Fields Versus Persistent Properties	9-19
Persistence Data Types	9-21
Entity Manager	9-22
Persistence Unit	9-23
Persistence Unit: Definition	9-24
Obtaining an Entity Manager	9-25
Using JPA in a Java SE Application	9-26
Quiz	9-27
Topics	9-28
What Is a Transaction?	9-29
ACID Properties of a Transaction	9-30

Transferring Without Transactions	9-31
Successful Transfer with Transactions	9-32
Unsuccessful Transfer with Transactions	9-33
Using JPA for Transactions	9-34
Quiz	9-35
Topics	9-36
Entity Instance Life Cycle	9-37
Persisting an Entity	9-39
Finding an Entity	9-40
Updating an Entity	9-41
Deleting an Entity	9-42
Detach and Merge	9-43
Queries with JPQL	9-44
Example: @NamedQuery	9-45
Quiz	9-46
Summary	9-48
Practice 9: Overview	9-49

10 Applying the JPA

Objectives	10-2
Topics	10-3
Entity Relationships	10-4
Relationship Direction	10-5
Unidirectional One-to-One Relationships	10-6
Many-to-One and One-to-Many Relationships	10-7
Employee and Department Entities	10-8
Bidirectional Many-to-Many Relationships	10-9
Employee and Project Entities	10-10
Quiz	10-11
Topics	10-12
The Criteria API	10-13
Steps to Create a Criteria Query	10-14
A JPQL Query Versus a Criteria Query	10-15
Using the Criteria API in the HenleyApp	10-16
Quiz	10-17
Topics	10-19
HenleyApp: Two-Tier Architecture	10-20
Using the JPA in the HenleyApp Two-Tier Application	10-21
Using a Persistent Provider Implementation	10-22
NetBeans IDE Configuration	10-23
Entity Relationships in the HenleyApp	10-24

Using the API	10-27
Defining a Query in the HenleyApp	10-28
The Criteria API in the HenleyApp	10-29
Packaging and Deployment	10-30
The persistence.xml File	10-31
Data Access Objects	10-34
Applying the DAO Design Pattern	10-35
Applying the DAO in the HenleyApp	10-36
HenleyApp Code Example	10-37
Summary	10-38
Practice 10: Overview	10-39

11 Implementing a Multi-tier Design with RESTful Web Services

Objectives	11-2
Topics	11-3
Two-Tier Architecture	11-4
HenleyApp Two-Tier Design	11-5
BrokerTool Two-Tier Design	11-6
Advantages of Two-Tier Design	11-7
Disadvantages of a Two-Tier Design	11-8
Three-Tier Architecture	11-9
HenleyApp Three-Tier Design	11-10
Extending HenleyApp to Three-Tier	11-11
BrokerTool Three-Tier Design	11-13
Advantages of Three-Tier Design	11-14
Disadvantages of Three-Tier Design	11-15
Quiz	11-16
Topics	11-17
Implementing Three-Tier Design	11-18
Web Services	11-19
Types of Web Services	11-20
Which Type of Web Service to Use?	11-21
When to Use REST	11-22
Principles of a RESTful Web Service	11-23
Relationships Between SQL and HTTP Verbs	11-24
Developing a RESTful Web Service with JAX-RS	11-25
RESTful Web Service with JAX-RS: An Example	11-26
Quiz	11-28
Topics	11-30
Three-Tier Architecture Using REST	11-31
Steps to Generate RESTful Web Services in NetBeans	11-32

Examine the RESTful Web Services in HenleyServer	11-33
Examining the DailySales Web Service Code	11-34
Testing DailySales RESTful Service	11-35
Quiz	11-36
Summary	11-37
Practice 11: Overview	11-38

12 Connecting to a RESTful Web Service

Objectives	12-2
Topics	12-3
Testing a RESTful Web Service	12-4
Testing HenleyServer's RESTful Web Services	12-5
Topics	12-9
Steps to Develop a Restful Web Service Client	12-10
Examine the Generated Web Service Client Code	12-11
Topics	12-12
Apply the Web Service Client Code in HenleyClient	12-13
Verify the Output in HenleyClient	12-14
Sales Form to Insert Details	12-15
Examining the Code for the POST/Insert Operation	12-16
Quiz	12-17
Summary	12-18
Practice 12: Overview	12-19

13 Packaging and Deploying Applications

Objectives	13-2
Topics	13-3
Packaging Introduction	13-4
Java jar Files	13-5
Application Example	13-6
jar Command Line	13-7
The Manifest File	13-8
Modifying the Manifest	13-9
Tools for Making .jar Files	13-10
Quiz	13-11
Topics	13-12
Deploying Applications	13-13
Stand-Alone Deployment	13-14
Installers and Wrappers	13-15
Applets/Embedded	13-16
Applet Security	13-17

Java Web Start	13-18
Java Web Start Benefits	13-19
Quiz	13-20
Topics	13-21
Deployment Toolkit	13-22
Deployment Toolkit Methods	13-23
Java Web Start Deployment	13-24
Java Web Start HTML Link	13-25
JNLP Application Descriptor	13-26
Web Browser Deployment	13-27
Deployment Toolkit Callbacks	13-28
Application Startup Process	13-29
Preloaders	13-30
Default Information	13-31
Packaging Tools	13-32
Breaking Out of the Sandbox	13-33
Quiz	13-34
Summary	13-35
Practice Overview	13-36

14 Developing Secure Applications

Objectives	14-2
Topics	14-3
Security Overview	14-4
Confidentiality	14-5
Integrity	14-6
Availability	14-7
Topics	14-8
Developing Secure Software	14-9
Fundamental Security Concepts: Part I	14-10
Fundamental Security Concepts: Part II	14-11
Topics	14-12
Types of Security Threats	14-13
Injection and Inclusion	14-14
XSS Example	14-15
Cross-Site Scripting	14-16
SQL Injection Example	14-17
SQL Injection	14-18
OS Command Injection Example	14-19
OS Command Injection	14-20
Uncontrolled Format String Example	14-21

Uncontrolled Format String	14-22
Resource Management	14-23
Denial-of-Service Examples	14-24
Confidential Exception Example	14-25
Confidential Log Example	14-26
Confidential Information	14-27
Accessibility and Extensibility	14-28
Minimize Mutable Classes	14-29
Immutability Example	14-30
Steps to Make a Class Immutable	14-31
Quiz	14-32
Topics	14-34
Security Resources on the Net	14-35
Summary	14-36
Practice 14: Overview	14-37

15 Signing an Application and Authentication

Objectives	15-2
Topics	15-3
Key Security Concepts	15-4
Caesar Cipher	15-5
Types of Encryption	15-6
Symmetric Key Encryption	15-7
Public Key Encryption	15-8
Quiz	15-9
Topics	15-10
Applying Encryption Technologies	15-11
Digital Certificates	15-12
Certificate Sample	15-13
Message Digests/Hashes	15-14
Digital Signatures I	15-15
Digital Signatures 2	15-16
Certificate Authority	15-17
Quiz	15-18
Topics	15-19
Code Signing with a CA	15-20
Steps for Getting a Certificate	15-21
Generating Public and Private Keys with keytool	15-22
Additional keytool Information	15-23
Sign a jar	15-24
Quiz	15-25

Topics	15-26
SSL	15-27
Generating Secure Connections	15-28
SSL Server Authentication	15-29
Quiz	15-30
Topics	15-31
Java EE Application Security Mechanisms	15-32
Securing Web Applications	15-33
Applying Security in GlassFish Server	15-34
HTTP Basic Authentication Mechanism	15-35
Authentication in an Application	15-36
Authentication in the BrokerTool Application	15-37
Deployment Descriptor of BrokerToolServer	15-38
Set Up Users and Groups on the GlassFish Server	15-39
Deployment Descriptor: glassfish-web.xml	15-40
Programmatic Security	15-41
Quiz	15-42
Summary	15-43
Practice 15: Overview:	15-44

16 Logging

Objectives	16-2
Topics	16-3
Logging Overview	16-4
Using the Java Logging API	16-5
Topics	16-6
Configure the Logger Handler	16-7
Configure the Logger Formatter	16-8
Configure the Logger Level	16-9
Quiz	16-10
Topics	16-11
Logger Example: main()	16-12
Logger Example: logMessages()	16-13
Logger Example: basic.log	16-14
Logger Example: SimpleLogging.java	16-15
Logger Example: simple.log	16-16
Quiz	16-17
Topics	16-18
File-Based Configuration	16-19

Custom Log Handler 16-20
Summary 16-21
Practice 16: Overview 16-22

17 Implementing Unit Testing and Using Version Control

Objectives 17-2
Topics 17-3
What Is Unit Testing? 17-4
Test Cases and Their Uses 17-5
Features of JUnit 17-6
Test Driven Development 17-7
Quiz 17-8
Annotations in JUnit 4.x 17-10
Steps to Write a Test Case 17-11
Writing a Test 17-12
Assert Statements 17-13
Test a Method That Throws an Exception 17-14
Run the Tests 17-16
Creating a Test Suite 17-17
Quiz 17-18
Topics 17-20
JUnit Support in NetBeans 17-21
Generating JUnit Test Classes in NetBeans 17-23
Running Tests in NetBeans 17-24
Quiz 17-25
Topics 17-26
Version Control System 17-27
Advantages of Using a Version Control System 17-28
Features of the Subversion (SVN) Tool 17-29
Using NetBeans to Manage Code in SVN 17-30
Using NetBeans to Commit Code to SVN 17-31
Using NetBeans to Commit to SVN 17-32
Quiz 17-33
Summary 17-34
Practice 17: Overview 17-35

18 Oracle Cloud

Agenda 18-2
What is Cloud? 18-3
What is Cloud Computing? 18-4
History – Cloud Evolution 18-5

Components of Cloud Computing	18-6
Characteristics of Cloud	18-7
Cloud Deployment Models	18-8
Cloud Service Models	18-9
Industry Shifting from On-Premises to the Cloud	18-13
Oracle IaaS Overview	18-15
Oracle PaaS Overview	18-16
Oracle SaaS Overview	18-17
Summary	18-18

19 Oracle Application Container Cloud Service Overview

Objectives	19-2
Oracle Application Container Cloud Service	19-3
Oracle Application Container Cloud	19-4
Polyglot Platform	19-5
Open Platform	19-6
Container-based Application Platform as a Service	19-7
Elastic Scaling	19-8
Profiling	19-9
Manageable	19-10
Deploy—Application Archive (Zip)	19-12
Application Deployment	19-13
Application Container Cloud Architecture	19-14
Load Balancer	19-15
Oracle Developer Cloud Service	19-16
Developer Cloud Service – Easy Adoption/Integration	19-17
Application Container Cloud Service Advantages	19-19
Summary	19-20

A Development Methodologies and Design Patterns

Topics	A-2
Agile Development	A-3
Scrum	A-4
Scrum Terminology	A-5
Scrum Roles	A-6
Topics	A-7
Design Patterns: An Introduction	A-8
Builder Design Pattern	A-9
Builder Pattern Example	A-10
Observer Design Pattern	A-11

Observer Pattern Example A-12

The MVC Design Pattern A-13

B JavaFX History and Architecture

The Story of JavaFX B-2

JavaFX Architecture and APIs B-3

AWT and Glass B-5

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Adolfo De+la+Rosa (adolfodelarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.

10

Applying the JPA

ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Identify relationships in an application
- Build and deploy a JPA application in a Java SE environment
- Apply a two-tier design in the HenleyApp application



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Topics

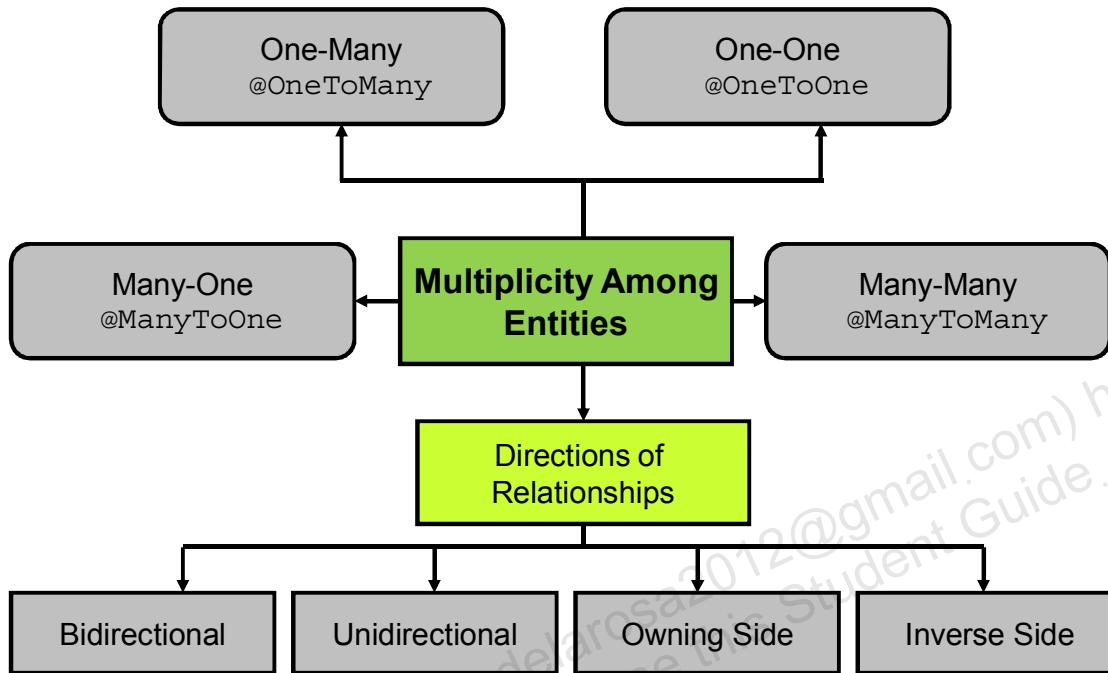
- Entity relationships
- The Criteria API
- Applying the JPA in the HenleyApp two-tier application



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Entity Relationships



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Entities must have relationships with other entities, which helps to develop the domain model for an enterprise application.

Relationship mapping can be:

One-to-many: An entity instance can be related to multiple instances of the other entities. One-to-many relationships use the javax.persistence.OneToMany annotation on the corresponding persistent property or field.

Many-to-one: This multiplicity is the opposite of a one-to-many relationship. Many-to-one relationships use the javax.persistence.ManyToOne annotation on the corresponding persistent property or field.

Many-to-many: The entity instances can be related to multiple instances of each other. Many-to-many relationships use the javax.persistence.ManyToMany annotation on the corresponding persistent property or field.

Directions of a relationship is bidirectional or unidirectional.

- Bidirectional relationships:
 - Must be managed by the application
 - Have an owning side and an inverse side
- The owning side controls the database write.
- The inverse side must specify the owning side by using the `mappedBy` element.

Relationship Direction

Unidirectional	Bidirectional
Only one entity has a pointer to the other.	Both entities point to one another.
The relationship has only an owning side.	The relationship has both an owning side and an inverse side.
<ul style="list-style-type: none">• The owning side of a relationship determines how the Persistence runtime makes updates to the relationship in the database.• The owning side controls the database write.	

ORACLE

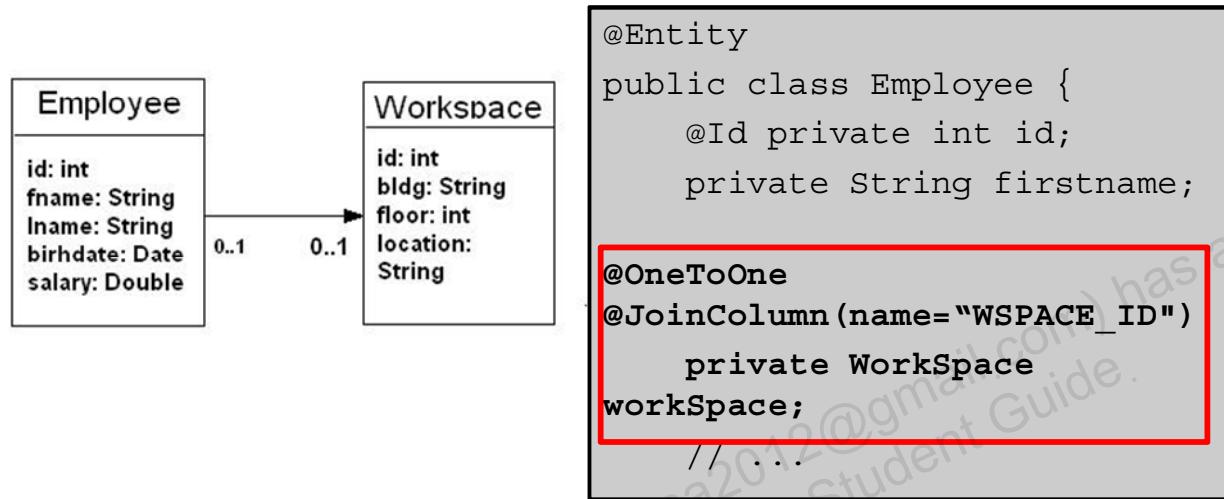
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Identifying the relationship direction is usually a data modeling decision, not a Java programming decision, and it would likely be decided based on the most frequent direction of traversal.

The subsequent slides have examples.

Unidirectional One-to-One Relationships

Each entity instance is related to a single instance of another entity.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Example of a one-to-one association would be an employee who has a workspace. Every employee gets assigned his or her own workspace; therefore, a one-to-one relationship from Employee to WorkSpace can be created as shown in the figure in the slide.

0..1 indicates the range.

In the Employee and Workspace table mapping, The foreign key column in the EMPLOYEE table will have WSPACE_ID that refers to the WORKSPACE table. However, since the direction is unidirectional, the Workspace entity will not have an Employee field or property in it. Employee knows about Workspace, but Workspace does not know which LineItem instances refer to it.

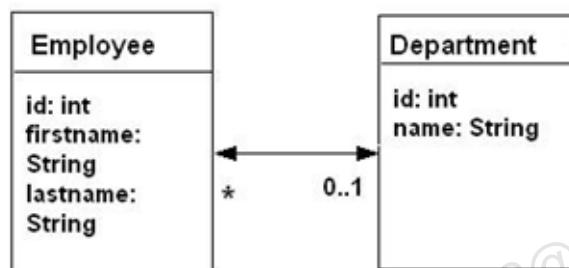
One-to-one relationships use the javax.persistence.OneToOne annotation on the corresponding persistent property or field. The code snippet in the slide uses the @OneToOne and @JoinColumn annotations.

This is a unidirectional one-to-one mapping. The entity table that contains the join column determines the entity that is the owner of the relationship. The other entity table (that is, Workspace) does not need a join column at all.

However, in one-to-one bidirectional relationships, the owning side corresponds to the side that contains the corresponding foreign key.

Many-to-One and One-to-Many Relationships

- In many-to-one, multiple instances of an entity can be related to a single instance of the other entity.
- In one-to-many, an entity instance can be related to multiple instances of the other entities.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Employee and Department example shows a many-to-one mapping from Employee to Department and a one-to-many mapping from Department to Employee. They are equivalent, because bidirectional many-to-one relationships imply a one-to-many mapping back from the target to source, and vice versa.

Because Department knows which Employee instances it has and Employee knows which Department it belongs to, they have a bidirectional relationship.

Employee and Department Entities

```

@Entity public class Employee {
    @Id protected Integer id;
    @ManyToOne
    @JoinColumn (name="DEPT_ID") → Owning side
    protected Department dept;
}

@Entity public class Department {
    @Id private Integer id;
    @OneToMany (mappedBy="dept") → Inverse side has mappedBy
    private Collection<Employee> emps;
}

```



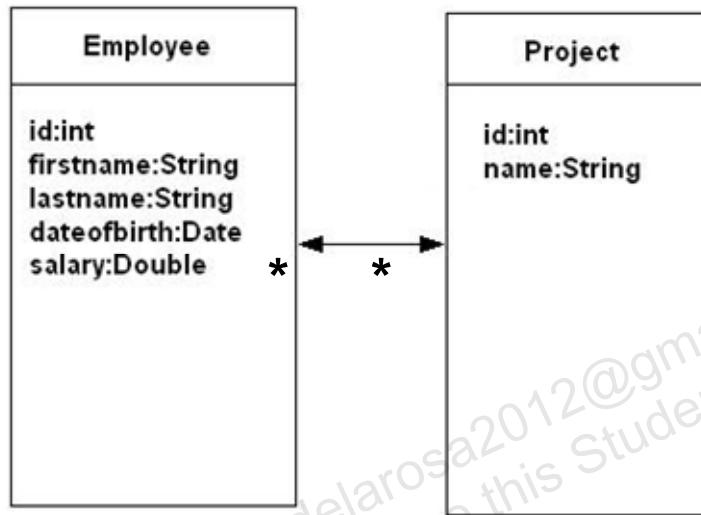
Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

In the code snippet in the slide, the many-to-one relationship between Employee and Department entities are shown.

- The many side is always the owning side of the relationship. Here Employee is the owning side. The JoinColumn is specified at the owning side.
- The inverse side of a bidirectional relationship must refer to its owning side by using the mappedBy element. In the code snippet in the slide, the @OneToMany annotation provides the mappedBy element in the Department class. Department is the inverse side.
- The mappedBy element designates the property or field in the entity that is the owner of the relationship. In the code snippet, mappedBy refers to “dept,” which is the Department field declared in the Employee class.
- The many side of many-to-one bidirectional relationships must **not** define the mappedBy element. In the Employee class, @ManyToOne annotation does not have the mappedBy element.
- A @JoiningColumn cannot be defined in this case. A separate join table has to be created in such a scenario.

Bidirectional Many-to-Many Relationships

When two entities are associated with each other by means of collections



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

When one or more entities are associated with a collection of their entities, we must model it as a many-to-many relationship. Each of the entities on each side of the relationship has a collection-valued association that contains entities of the target type. The figure in the slide shows a many-to-many relationship between Employee and Project.

Each employee can work on multiple projects, and each project can be worked on by multiple employees. This is a bidirectional many-to-many relationship.

Employee and Project Entities

```
@Entity public class Employee {  
    @Id protected Integer id;  
    @ManyToOne protected Department dept;  
    @ManyToMany protected Collection<Project> projects;  
}
```

Owning side

```
@Entity public class Project {  
    @Id private Integer id;  
    @ManyToMany (mappedBy="projects")  
    private Collection<Employee> emps;  
}
```

Inverse side has a mappedBy attribute.

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

In the code snippet in the slide, two entities are considered: Employee and Projects.

Both sides have `@ManyToMany` mappings. There is no `@JoinColumn` in this relationship. As it is a bidirectional relationship, you have to choose an owning side and an inverse side.

In this example, Employee is chosen as the owning side. Project is the inverse side and has the `mappedBy` element.

The only way to implement a many-to-many relationship is with a separate join table.

Quiz

State whether the following statements are true:

- a. For many-to-many bidirectional relationships, either side may be the owning side.
- b. The owning side of a bidirectional relationship controls the database write.
- c. For many-to-many bidirectional relationships, there is no need to have an owning side.
- d. The many side of a bidirectional relationship is always the owning side of the relationship.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, d

Topics

- Entity relationships
- The Criteria API
- Applying the JPA in the HenleyApp two-tier application



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Criteria API

- The Criteria API, an alternative method for constructing queries, uses a Java programming language API instead of JPQL or native SQL.
- Criteria API allows generics and thus removes the need for casting.
- Criteria queries set the SELECT, FROM, and WHERE clauses by using Java programming language objects, so the query can be created in a typesafe manner.
- Criteria API :
 - Standardizes many of the programming features that exist in proprietary persistence products
 - Applies programming best practices of the proprietary models
 - Makes full use of the Java programming language features



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Criteria API and JPQL are closely related and are designed to allow similar operations in their queries. Developers familiar with JPQL syntax will find equivalent object-level operations in the Criteria API. The Criteria API was introduced in JPA 2.0.

In spite of JPQL, programming APIs are still used to enable features not yet supported by the standard query language. The Criteria API allows you to find, modify, and delete persistent entities by invoking Java Persistence API entity operations.

Steps to Create a Criteria Query

1. Use an EntityManager instance to create a CriteriaBuilder object.
2. Create a query object by creating an instance of the CriteriaQuery interface.
3. Set the query root by calling the FROM method on the CriteriaQuery object.
4. Specify what the type of the query result will be by calling the SELECT method of the CriteriaQuery object.
5. Prepare the query for execution by creating a TypedQuery<T> instance, specifying the type of the query result.
6. Execute the query by calling the getResultList method on the TypedQuery<T> object.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

For a particular CriteriaQuery object, the root entity of the query, from which all navigation originates, is called the *query root*. It is similar to the FROM clause in a JPQL query.

A JPQL Query Versus a Criteria Query

JPQL	Query Using Criteria API
<pre>SELECT emp FROM Employee emp WHERE emp.firstname = 'John'</pre>	<pre>1 CriteriaBuilder cb = em.getCriteriaBuilder(); 2 CriteriaQuery<Employee> cq = cb.createQuery(Employee.class); 3 Root<Employee> emp = cq.from(Employee.class); 4 cq.select(emp) .where(cb.equal(emp.get("firstname"), "John")); 5 TypedQuery<Employee> tq = em.createQuery(cq); 6 List<Employee> allemps = tq.getResultList();</pre>



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The query in the slide, written using the Criteria API, can be explained as follows:

1. To create a CriteriaBuilder instance, call the getCriteriaBuilder method on the EntityManager instance: CriteriaBuilder cb = em.getCriteriaBuilder();
2. The query object is created by using the CriteriaBuilder instance:
`CriteriaQuery<Employee> cq = cb.createQuery(Employee.class);`
The query will return instances of the entity, so the type of the query is specified when the CriteriaQuery object is created to create a typesafe query.
3. The FROM clause of the query is set, and the root of the query specified, by calling the from method of the query object: `Root<Employee> emp = cq.from(Employee.class);`
4. The SELECT clause of the query is set by calling the select method of the query object and passing in the query root: `cq.select(emp);`
5. The query object is now used to create a TypedQuery<T> object that can be executed against the data source. The modifications to the query object are captured to create a ready-to-execute query: `TypedQuery<Employee> tq = em.createQuery(cq);`
6. This typed query object is executed by calling its getResultList method, because this query will return multiple entity instances. The results are stored in a List<Employee> collection-valued object: `List<Employee> allemps = tq.getResultList();`

Using the Criteria API in the HenleyApp

```
public List<T> findAll() {  
    javax.persistence.criteria.CriteriaQuery cq =  
        getEntityManager().getCriteriaBuilder().createQuery();  
    cq.select(cq.from(entityClass));  
    return  
        getEntityManager().createQuery(cq).getResultList();  
}  
  
public List<T> findRange(int[] range) {  
    javax.persistence.criteria.CriteriaQuery cq =  
        getEntityManager().getCriteriaBuilder().createQuery();  
    cq.select(cq.from(entityClass));  
    javax.persistence.Query q =  
        getEntityManager().createQuery(cq);  
    q.setMaxResults(range[1] - range[0]);  
    q.setFirstResult(range[0]);  
    return q.getResultList();  
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The slide shows some examples of using Criteria queries in the HenleyApp application.

Quiz

Which of the following statements are true about Criteria API?

- a. The Criteria API uses the Java programming language API.
- b. The Criteria API does not allow the use of generics.
- c. Criteria queries set the SELECT, FROM, and WHERE clauses by using Java programming language objects, so the query can be created in a typesafe manner.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Quiz

Identify the classes of the Criteria API from the following list:

- a. CriteriaBuilder
- b. CriteriaQuery
- c. EntityManager
- d. NamedQuery



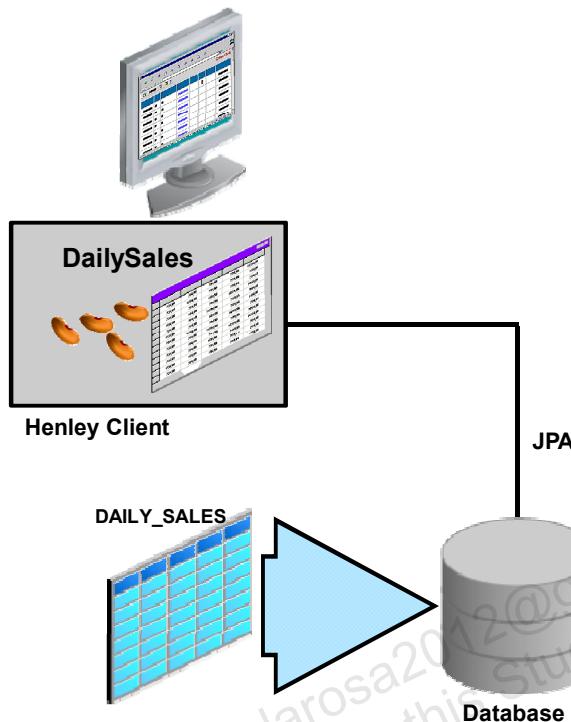
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Topics

- Entity relationships
- The Criteria API
- Applying the JPA in the HenleyApp two-tier application

HenleyApp: Two-Tier Architecture



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Henley client contains the front-end components and data-access components to manage data that is stored in the back-end Java DB database server. The front-end components use Java and JavaFX APIs to design user interface screens. The data access components use JPA to perform the database operations. The database server used for this application is Java DB.

The development environment or IDE used is NetBeans IDE.

Using the JPA in the HenleyApp Two-Tier Application

1. Download the required persistent provider implementation.
2. Configure NetBeans IDE for using the JPA for development.
3. Figure out the entity relationships in the HenleyApp application.
4. Examine methodical usage of EntityManagerFactory and EntityManager instances in the HenleyApp application.
5. Examine the usage of named queries and the Criteria API in the HenleyApp application.
6. Package the HenleyApp JPA application.
7. Examine the information that goes to the persistence.xml file of HenleyApp.
8. Examine how DAO pattern can be applied while using JPA in HenleyApp.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

This slide shows what will be covered in the subsequent slides. The subsequent slides show how JPA has been applied to the HenleyApp two-tier application.

Using a Persistent Provider Implementation

- Persistence provider implementations may be optimized for specific database vendors.
- One important reason for using the JPA is to decouple your application from the underlying database technology.
- After downloading and installing the reference implementation, you must configure your development environment to use these files in your project. The goals for configuration are to:
 - Include the reference implementation in the Java language compiler classpath
 - Include the reference implementation in the Java runtime environment (JRE) classpath



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

You can download the Java Persistence API reference implementation from the GlassFish project. Although originally part of the enterprise application server, the reference implementation works well in desktop applications too.

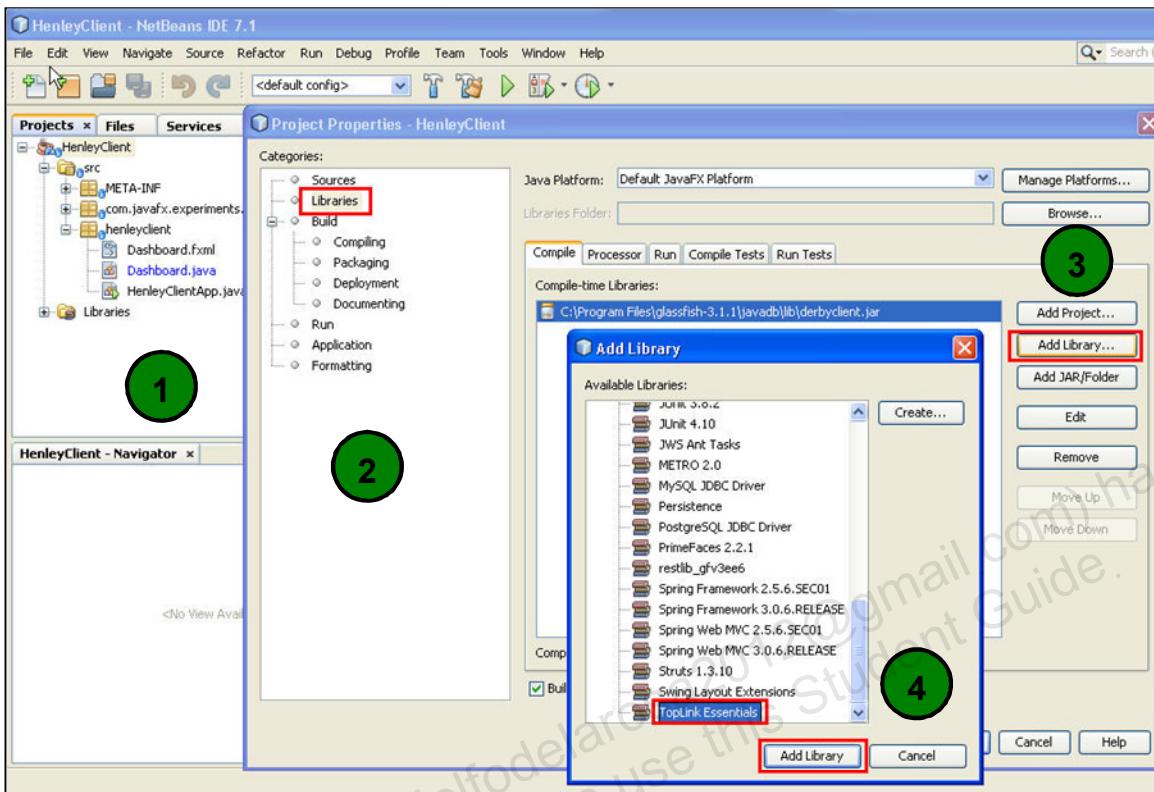
You can easily use and package the reference implementation in your desktop applications. Using the reference implementation, you can simplify your desktop applications to use simple POJOs for both application logic and persistence. Adding the API to your development environment usually involves just including the provider's JAR files in your compiler and runtime environment classpaths. An IDE can help you automate the process of packaging an API implementation with your application.

You should avoid using proprietary extensions. Avoiding proprietary extensions ensures that you can use the widest range of database technologies now and in the future. You should contact your provider vendor for information about what databases they support.

GlassFish is an industrial-strength, open-source application server. Being the reference implementation of the Java EE 6 standard, GlassFish also provides a Java Persistence API implementation. You do not have to download the entire GlassFish product to get the API implementation. The filename is `glassfish-persistence-installer-v2-b46.jar`, but you should expect slightly different filenames as the version changes.

[TopLink Essentials JPA implementation](#) is the reference implementation provided by GlassFish.

NetBeans IDE Configuration



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The simplest way to install and configure a JPA reference implementation is to use an integrated development environment (IDE). NetBeans or Eclipse are both examples of popular IDEs that will help you include the provider implementation JAR files in your compiler and JRE classpaths.

In NetBeans:

- Add the persistent provider, which is EclipseLink (JPA 2.0) to your project from the properties window, as shown in the slide.
- In order to use the Java DB database client, you must add the file `derbyclient.jar` to the project. `derbyclient.jar` is available in your Java installation location (for example, `D:\Program Files\Java\jdk1.7.0\db\lib`). This JAR file can be added from the project properties window.

Entity Relationships in the HenleyApp

```
@Entity  
@Table(name = "DAILY_SALES", catalog = "", schema  
= "HENLEY")  
.....  
public class DailySales implements Serializable {  
....  
@JoinColumn(name = "REGION_ID",  
referencedColumnName = "REGION_ID")  
    @ManyToOne  
    private Region regionId;  
  
@JoinColumn(name = "PRODUCT_ID",  
referencedColumnName = "PRODUCT_ID")  
    @ManyToOne  
    private Product productId;  
.....
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Having identified the entities of HenleyApp in the previous lesson, in this and the next two slides, we will identify the entity relationships.

Consider the following entities: DailySales, Product, Region.

In NetBeans, examine DailySales.java, Product.java, and Region.java of the HenleyApp_2Tier project.

Entity Relationships in the HenleyApp

```
@Entity  
@Table(name = "PRODUCT", catalog = "", schema =  
"HENLEY")  
.....  
public class Product implements Serializable {  
    @OneToMany(mappedBy = "productId")  
    private Collection<DailySales>  
    dailySalesCollection;  
    @JoinColumn(name = "PRODUCT_ID",  
    referencedColumnName = "PRODUCT_ID")  
    @ManyToOne  
    private Product productId;  
.....
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The code snippet in the slide from `Product.java` shows the entity relationship between `Product` and `DailySales`.

Entity Relationships in the HenleyApp

```
@Entity  
@Table(name = "REGION", catalog = "", schema =  
"HENLEY")  
....  
public class Region implements Serializable {  
....  
    @OneToMany(mappedBy = "regionId")  
    private  
    Collection<DailySales> dailySalesCollection;  
....}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The code snippet in the slide from Region.java shows the entity relationship with DailySales.

Using the API

How to Use the API

- To work with entities, you must use the `javax.persistence` package.
- To work with an entity, you need an `EntityManager` instance.
- To create an `EntityManager` object, you need an `EntityManagerFactory` instance.
- To get the factory, you must use the `Persistence` class.
- To create queries and transactions in a desktop environment, you can use the `EntityManager` object.

Code Example

```
// declaring instances  
EntityManagerFactory emf;  
EntityManager em;  
  
//creating objects  
emf =  
Persistence.createEntityManagerFactory("henley");  
em = emf.createEntityManager();  
  
//creating transactions  
em.getTransaction().begin();  
em.persist(dailysls);  
em.getTransaction().commit();
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

In this slide, you identify the packages and classes of JPA, which you will commonly use.

- The `javax.persistence` package is one of the important packages of the JPA .
- One of the first things you need is an `EntityManager` instance.
- An `EntityManager` provides methods to begin and end transactions, to persist and find entities in the persistence context, and to merge or even remove those entities.
- Additionally, an `EntityManager` instance can create and execute queries.
- The `Persistence` class is the bootstrap class used in Java SE environments.

Defining a Query in the HenleyApp

```
@Entity  
 @Table(name = "CUSTOMER", catalog = "", schema =  
 "HENLEY")  
 @XmlRootElement  
 @NamedQueries({  
     @NamedQuery(name = "Customer.findAll", query =  
 "SELECT c FROM Customer c"),  
     @NamedQuery(name = "Customer.findByCustomerId",  
 query = "SELECT c FROM Customer c WHERE c.customerId =  
 :customerId"),  
     @NamedQuery(name = "Customer.findByFirstName", query  
 = "SELECT c FROM Customer c WHERE c.firstName =  
 :firstName"),  
     @NamedQuery(name = "Customer.findByLastName", query  
 = "SELECT c FROM Customer c WHERE c.lastName =  
 :lastName")})  
 public class Customer implements Serializable {
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The name of the query must be unique within the entire persistence unit. A common practice is to prefix the query name with the entity name. For example, the "findAll" query for the Customer entity would be named "Customer.findAll".

The Criteria API in the HenleyApp

```
public List<T> findAll() {  
    javax.persistence.criteria.CriteriaQuery cq =  
        getEntityManager().getCriteriaBuilder()  
            .createQuery();  
    cq.select(cq.from(entityClass));  
    return getEntityManager().createQuery(cq)  
        .getResultList();  
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

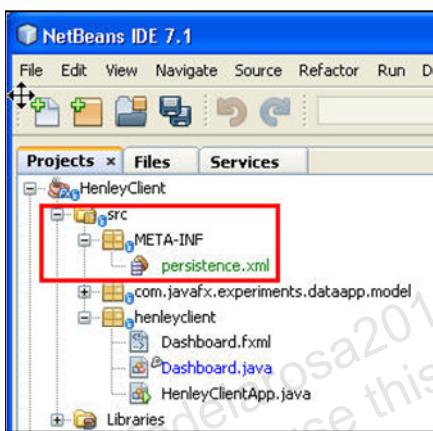
The simplest form of query execution is via the `getResultList()` method. It returns a collection containing the query results. If the query did not return any data, the collection is empty. The return type is specified as a `List` instead of a `Collection` in order to support queries that specify a sort order.

For queries that return values, the developer may choose to call either `getSingleResult()` if the query is expected to return a single result or `getResultList()` if more than one result may be returned. The `executeUpdate()` method is used to invoke bulk update and delete queries.

The code in the slide shows the use of the Criteria API.

Packaging and Deployment

- You must define your application's persistence unit in a configuration file called `persistence.xml`.
- This file should exist alongside your application in a `META-INF` directory.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

This slide and the following few slides cover how to package and deploy a JPA application to a Java SE environment.

Packaging means integrating the parts of the application so that it can be interpreted correctly and the application works accurately when deployed to an application server or run in a stand-alone JVM.

Deployment is the process of getting the application into an execution environment and running it. There are some obvious differences between deploying in a Java EE server and deploying to a Java SE runtime environment; for example, some of the Java EE container services will not be present.

The set of entities in your application is called a persistence unit.

You can put the `META-INF` subdirectory within your project's source directory as shown in the diagram in the slide.

The `persistence.xml` file lets the persistence API implementation know about entities.

The persistence.xml File

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" ..... >
  1  <persistence-unit name="henley" transaction-
    type="RESOURCE_LOCAL">
  2  <provider>oracle.toplink.essentials.ejb.cmp3
    EntityManagerFactoryProvider</provider>
  3  <class>com.javafx.experiments.dataapp.model.DailySales
    </class>
    <class>com.javafx.experiments.dataapp.model.Address
    </class>
    <class>com.javafx.experiments.dataapp.model.Customer
    </class>
    .......
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The persistence.xml file has many functions, but its most important task in the desktop environment is to list all the entities in your application and to name the persistence unit. Listing entity classes is required for portability in Java SE environments. The file may contain one or more persistence unit configurations that are separate and distinct from one another.

Tools like the NetBeans IDE can create the outline for you.

The important elements of the file number listed in the slide are as follows:

1. persistence-unit

The <persistence-unit> tag defines the name of the persistence unit used in the EntityManagerFactory method, and transaction-type defines the transaction resource type—either local (RESOURCE_LOCAL) or container (JTA). The default for Java SE applications is resource local transactions.

2. Provider

The <provider> tag identifies the persistence provider. As mentioned previously, the reference implementation for JPA 2.0 is EclipseLink.

3. Class

Use the class element to list the entity class names in your application.

4. Property (shown in the next slide)

The <properties> key in the persistence unit definition includes properties named with the prefix javax.persistence., such as:

- jdbc.driver (the JDBC driver class)
- jdbc.url (the JDBC URL to load)
- jdbc.user and jdbc.password (the username and password for the database defined in the URL)

The JDBC properties are now standard in JPA 2.0.

The persistence.xml File

4

```
<properties>
    <property name="toplink.jdbc.user"
    value="henley"/>
    <property name="toplink.jdbc.password"
    value="henley"/>
    <property name="toplink.jdbc.url"
    value="jdbc:derby://localhost:1527/henley"/>
    <property name="toplink.jdbc.driver"
    value="org.apache.derby.jdbc.ClientDriver"/>
</properties>
</persistence-unit>
</persistence>
```

database connection properties



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

In Java SE desktop environments, you should put database connection properties in the persistence.xml file if Java Naming and Directory Interface (JNDI) lookups are not possible.

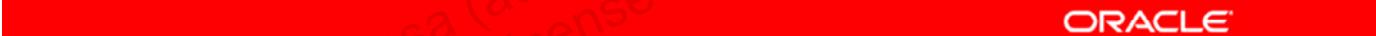
Database connection properties include:

- The username and password for the database connection
- The database connection string
- Driver classname

Additionally, you can even include persistence provider properties like options to create or drop-create new tables.

Data Access Objects

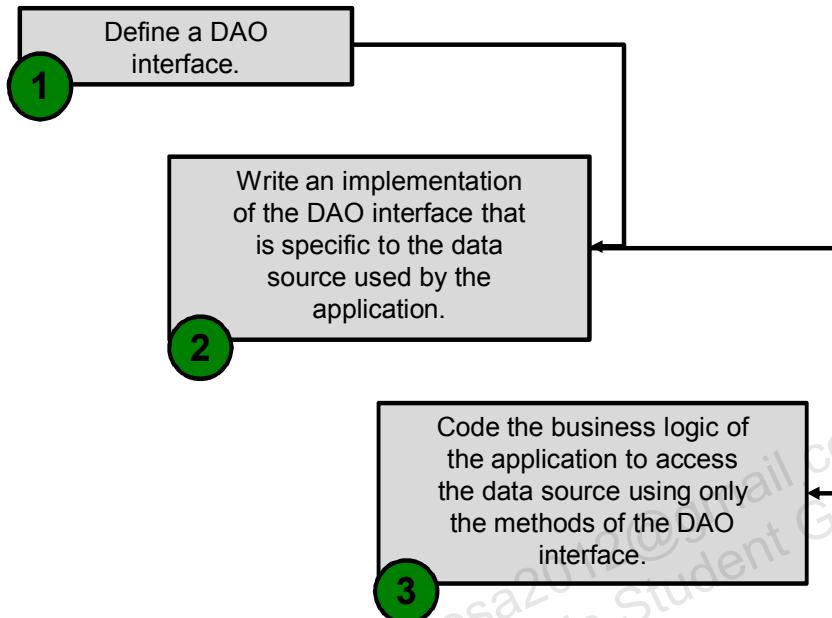
- The role of a data access object (DAO) is to segregate completely the persistence logic from business or presentation logic.
- This ensures that the user of DAOs is unaware of the database, its physical representation, and the relationships between the objects of the database.
- The primary advantage of using the DAO design pattern is isolating the code that needs to be changed whenever the application's data source changes.
- The DAO pattern is suitable for introducing JPA into an existing application.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Applying the DAO Design Pattern



ORACLE

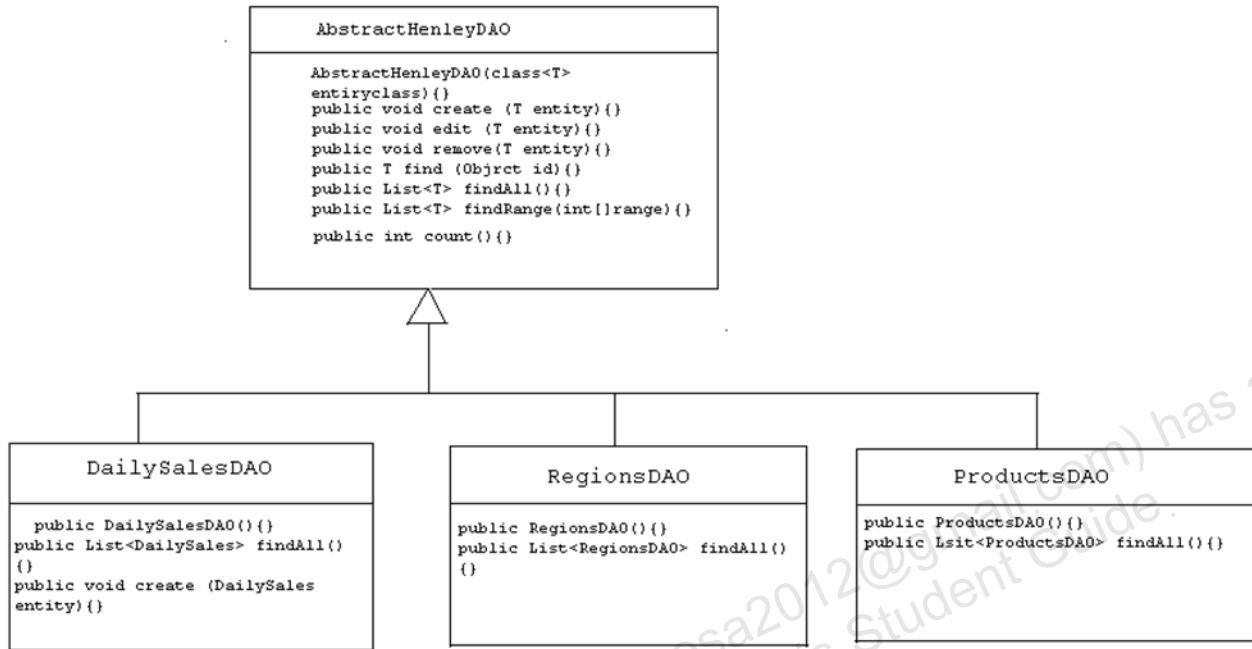
Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The DAO design pattern is implemented using the steps listed in the slide:

- The DAO interface contains methods for interacting (reading and writing) with a data source. These methods signatures must be generic. This would enable these methods not to contain any aspects that would bind them to a specific database.

Directly exposing persistence APIs to other application tiers is something to be avoided. Therefore, a well-designed data access object implements a simple persistence manager interface by delegating to a particular persistence mechanism.

Applying the DAO in the HenleyApp



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

AbstractHenleyDAO class is an abstract class and it provides all the data access methods. The methods use generics to create, edit, remove, and find data in the database.

DailySalesDAO, **RegionsDAO**, and **ProductsDAO** extend the **AbstractHenleyDAO** class and override the required methods of the abstract class.

The entity class instances (**DailySales**, **Products**, and **Regions**) are used to represent tables of the database.

HenleyApp Code Example

```
public abstract class AbstractHenleyDAO<T> {  
    private Class<T> entityClass;  
    protected HenleyEntityManager HenleyEnM;  
  
    public AbstractHenleyDAO(Class<T> entityClass) {  
        this.entityClass = entityClass;  
        HenleyEnM=new HenleyEntityManager();  
        HenleyEnM.create();  
    }  
  
    public void create(T entity) {  
        HenleyEnM.getEntityManager().getTransaction().begin();  
  
        HenleyEnM.getEntityManager().persist(entity);  
        HenleyEnM.getEntityManager().getTransaction().commit();  
        HenleyEnM.close();  
    }  
}
```

1
AbstractHenleyDAO
class

2
DailySalesDAO

```
public class DailySalesDAO extends AbstractHenleyDAO<DailySales> {  
  
    public DailySalesDAO() {  
        super(DailySales.class);  
    }  
  
    @Override public List<DailySales> findAll() {  
        return super.findAll();  
    }  
  
    @Override  
    public void create(DailySales entity) {  
        super.create(entity);  
    }  
}
```

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

In the first code snippet given in the slide, we can see that the create method of the AbstractHenleyDAO class provides the logic for creating a new record in a table. The method uses generics so that it can be used for creating or persisting any entity.

In the second code snippet, the overridden create method of DailySalesDAO is shown. It invokes its super classes' create method and provides the DailySales entity in the constructor.

Summary

In this lesson, you should have learned how to:

- Identify relationships in an application
- Build and deploy a JPA application in a Java SE environment
- Apply a two-tier design in the HenleyApp application



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Practice 10: Overview

- 10-1: Identifying Entity Relationships in the BrokerTool Application
- 10-2: Implementing Database Connectivity in the BrokerTool Application by Using the JPA



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Adolfo De+la+Rosa (adolfodelarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.

11

Implementing a Multi-tier Design with RESTful Web Services

ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Compare the HenleyApp two-tier design and the HenleyApp three-tier design
- Describe a RESTful web service
- List the web services used in the HenleyApp application
- Describe how the RESTful web services were developed in the HenleyServer application



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

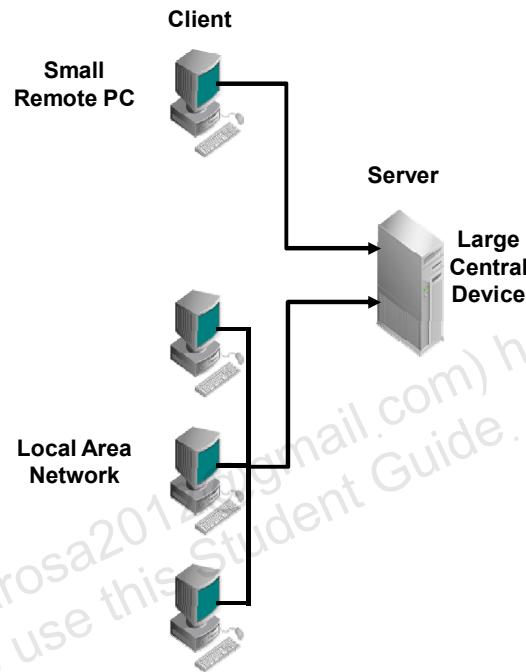
Topics

- Three-tier design versus two-tier design
- JAX-RS web services
- JAX-RS web services in the HenleyServer application

Two-Tier Architecture

Two-tier client/server architectures have two essential components:

- A client application
- A database server



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Two-Tier Considerations

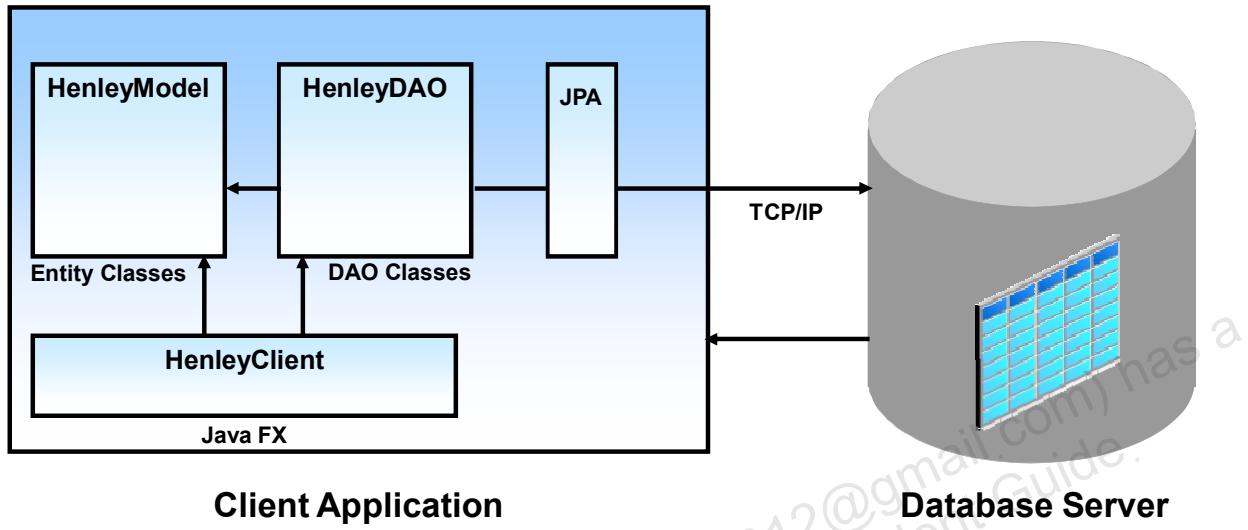
Client application accesses database directly.

- Requires a code change to port to a different database
- Potential bottleneck for data requests
- High volume of traffic due to data shipping

Client application executes application logic.

- Limited by processing capability of client workstation (memory, CPU)
- Requires application code to be distributed to each client workstation

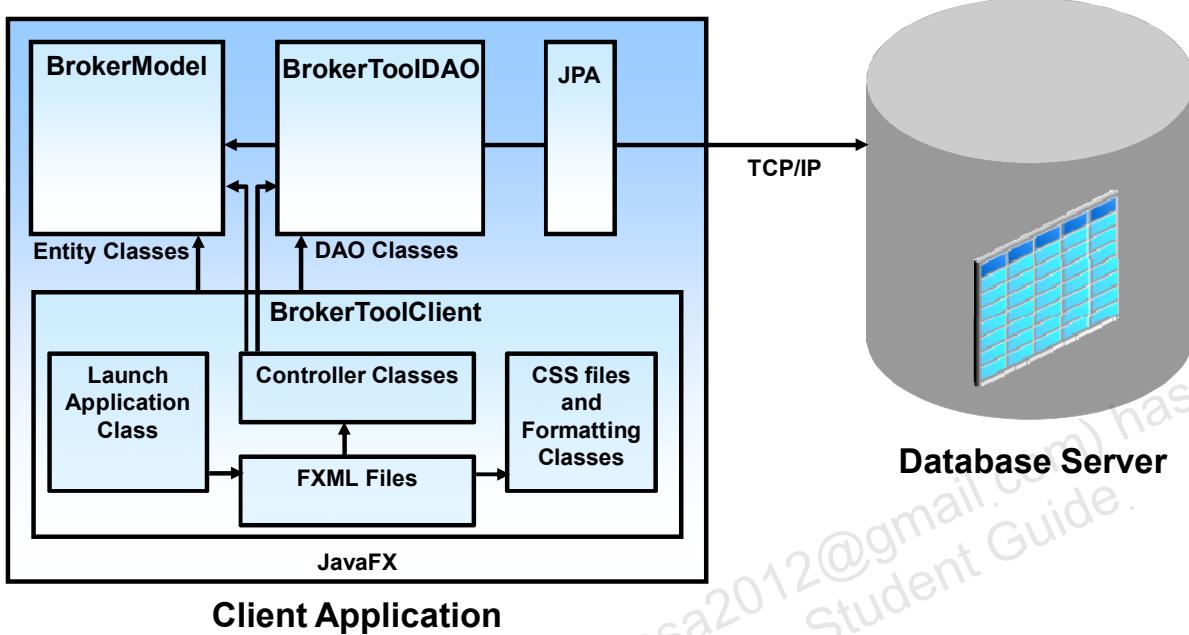
HenleyApp Two-Tier Design



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

BrokerTool Two-Tier Design



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Client Application

The client application as shown in the slide comprises BrokerToolClient, BrokerModel, BrokerToolDAO, and JPA components.

- **BrokerToolClient**: This component contains FXML files representing the front-end screens, corresponding controller classes that perform event handling, a class to launch the BrokerTool application, and CSS files, along with a few formatting classes that are used to format the front-end screens.
- **BrokerModel**: This component contains the entity classes that represent the tables of the BrokerTool database.
- **BrokerToolDAO**: This component contains the classes that perform the database operations using the BrokerModel classes.
- **JPA**: This component is the JPA 2.0 library that is used to connect to the database and perform database operations smoothly.

Database Server

Java DB is the database server that is used to store the tables of the BrokerTool database.

Advantages of Two-Tier Design

A two-tier design:

- Is more extensible than a one-tier design
- Combines presentation logic, business logic, and data resources into a single system
- Can have a client on any host as long as that host is connected by a network to the data-resource tier, unlike a one-tier design
- Has fewer points of failure than a three-tier design



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Disadvantages of a Two-Tier Design

A two-tier design has the following disadvantages:

- Any changes to the business strategy result in changes in the business logic, which requires redeployment of all clients. This can be very costly and time-consuming.
- Each client requires a connection to the data resource.
- This design restricts or complicates the addition of mirroring, caching, proxy services, or secure transactions to the data resource.
- The business logic is in the client tier. Therefore, all the database data that is used by the application is exposed on the network.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Three-Tier Architecture

A three-tier client-server architecture has the following components:

- A client application
- An application server
- A database server



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Three-Tier Architecture Considerations

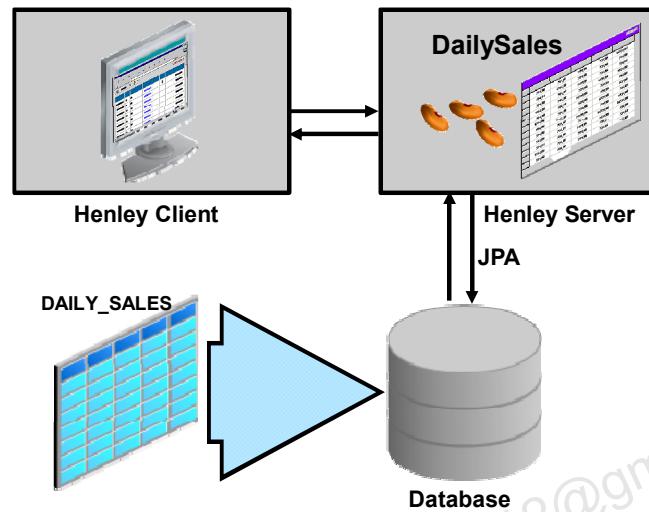
The client application contains presentation logic only; therefore:

- Less resources are needed for client workstation.
- No client modification is needed if the database location changes.
- There is less code to distribute to client workstations.

One server handles many client requests; therefore:

- More resources are available for the server application.
- Data traffic on the network is reduced.

HenleyApp Three-Tier Design



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Extending HenleyApp to Three-Tier

- Develop a HenleyServer application as a web application (WAR file).
 - It will contain the business logic to perform CRUD operations with the HenleyApp database on the Java DB server.
 - Its components will be published as web services.
 - The web services will be developed using Jersey, which is a reference implementation of JAX-RS.
 - It will be deployed on the GlassFish application server and can be consumed over HTTP.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The list in this slide and the next provides an overview of how to extend the HenleyApp application to a three-tier application.

Create a HenleyApp Server application and deploy it on a GlassFish application server. The server application is a collection of RESTful web services that are available over HTTP. The web services use JPA to talk to the HenleyApp database stored on a Java DB database server.

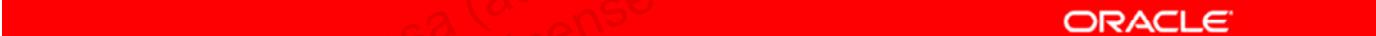
GlassFish will provide infrastructure services for both JPA and JAX-RS.

If the Java DB server is deployed in a separate address space, the three-tier architecture can be extended to a multi-tier one.

Subsequent slides discuss why JAX-RS was chosen as the web services API.

Extending HenleyApp to Three-Tier

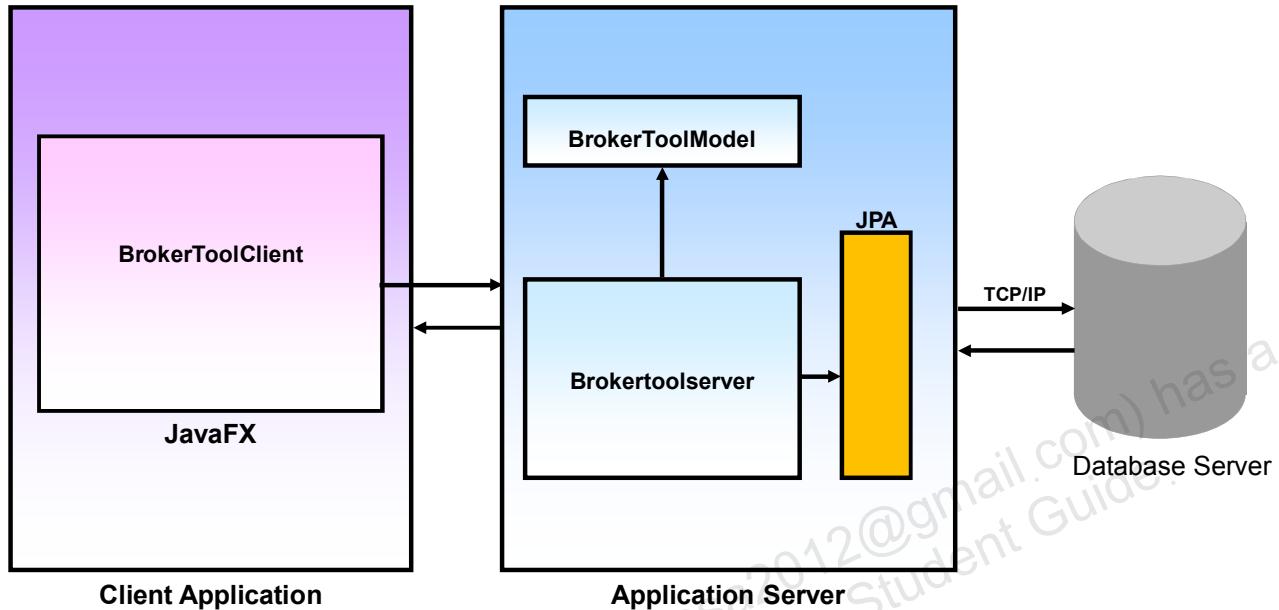
- Separate the model and the view component into two tiers.
 - HenleyClient will comprise the front-end user interface and controller component developed in JavaFX.
 - HenleyModel will contain the Entity classes that will be used by JPA for database persistence.
 - HenleyModel classes will be used by HenleyServer. Therefore, HenleyModel's JAR will be added to the HenleyServer library.
- Modify the JavaFX front-end components of HenleyClient to consume the web services published by HenleyServer.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

BrokerTool Three-Tier Design



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Client Application

The client application as shown in the slide comprises BrokerToolClient. The BrokerToolClient is the front end of the BrokerTool application. It primarily comprises FXML files and their corresponding controller classes.

Application Server

The Application Server contains the BrokerToolModel, BrokertoolServer, and JPA components that represent the business logic of the BrokerTool application.

- **BrokerToolModel:** This contains the entity classes class that represent the tables of the BrokerTool database.
- **BrokertoolServer:** This component's classes provide server capability and use the BrokerToolModel classes.
- **JPA:** This component is the JPA 2.0 library that is used by the BrokertoolServer to connect to the database and perform database operations smoothly.

Database Server

The enterprise storage system is placed on the database server. Java DB is the database server that is used to store the tables of the BrokerTool database.

Advantages of Three-Tier Design

A three-tier design has the following advantages:

- It enables efficient use of data resource connections by using connection pooling.
- It is possible to change business logic without redeploying client software.
- It is architecturally better suited for scaling and load balancing than other designs.
- Scaling affects primarily the middle tier.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Disadvantages of Three-Tier Design

A three-tier design has the following disadvantages:

- This design has increased network traffic.
- This design has multiple points of failure.
- Business objects must be designed to manage transactional integrity.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Quiz

Select the advantages of three-tier design from the following list:

- a. The business logic is in the client tier. Therefore, all the database data that is used by the application is exposed to the network.
- b. It is possible to change business logic without redeploying client software.
- c. It is architecturally better suited for scaling and load balancing than other designs.
- d. It combines presentation logic, business logic, and data resources into a single system



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

Topics

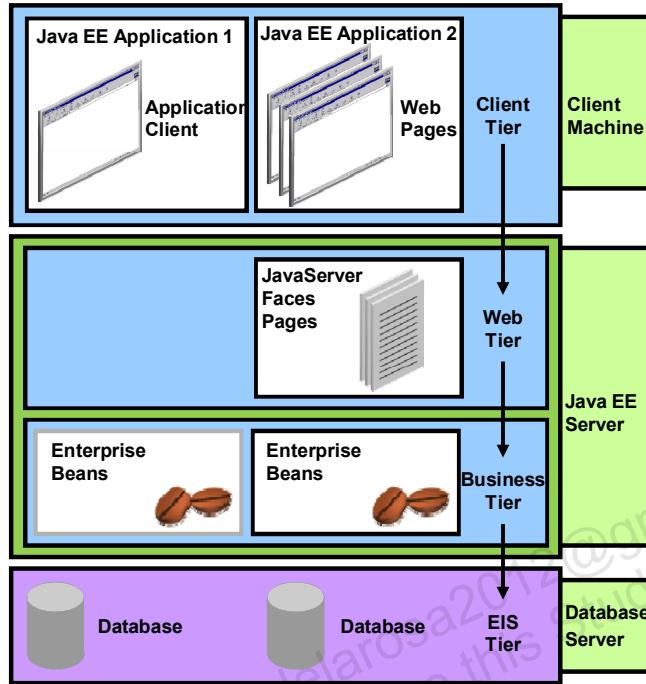
- Three-tier design versus two-tier design
- **JAX-RS web services**
- JAX-RS web services in the HenleyServer application



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Implementing Three-Tier Design



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Java EE Technologies Used to Implement Three-Tier Design

- **Java EE Web Components**
- **Java EE – EJB components**
- **Java EE web services**
- **Java EE server:** The runtime portion of a Java EE product. A Java EE server provides EJB and web containers.
- **Enterprise JavaBeans (EJB) container:** Manages the execution of enterprise beans for Java EE applications. Enterprise beans and their container run on the Java EE server.
- **Web container:** Manages the execution of web pages, servlets, and some EJB components for Java EE applications. Web components and their container run on the Java EE server.
- **Application client container:** Manages the execution of application client components. Application clients and their container run on the client.
- **Applet container:** Manages the execution of applets. Consists of a web browser and Java Plug-in running on the client together.
- **EE EJB** is also designed for such things, but it can be a pretty heavy architecture for a simple client-server app.

Web Services

Web services:

- Are applications that communicate over the World Wide Web's (WWW) HyperText Transfer Protocol (HTTP)
- Provide a standard means of interoperating between software applications running on a variety of platforms and frameworks
- Are interoperable and extensible because of using XML
- Can be combined in a loosely coupled way to achieve complex operations



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A web service is a software component provided through a network-accessible endpoint. By using web services, your client server two-tier application can be modified to a three-tier networked application that can operate over the web. Thus your three-tier application will become extensible as the use cases expand and it will become interoperable with different kinds of client applications.

Types of Web Services

The two types of web services are Simple Object Access Protocol (SOAP)-based web services and Representational State Transfer (REST)ful web services:

JAX-WS	API for SOAP-based web services
JAX-RS	API for RESTful web services



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

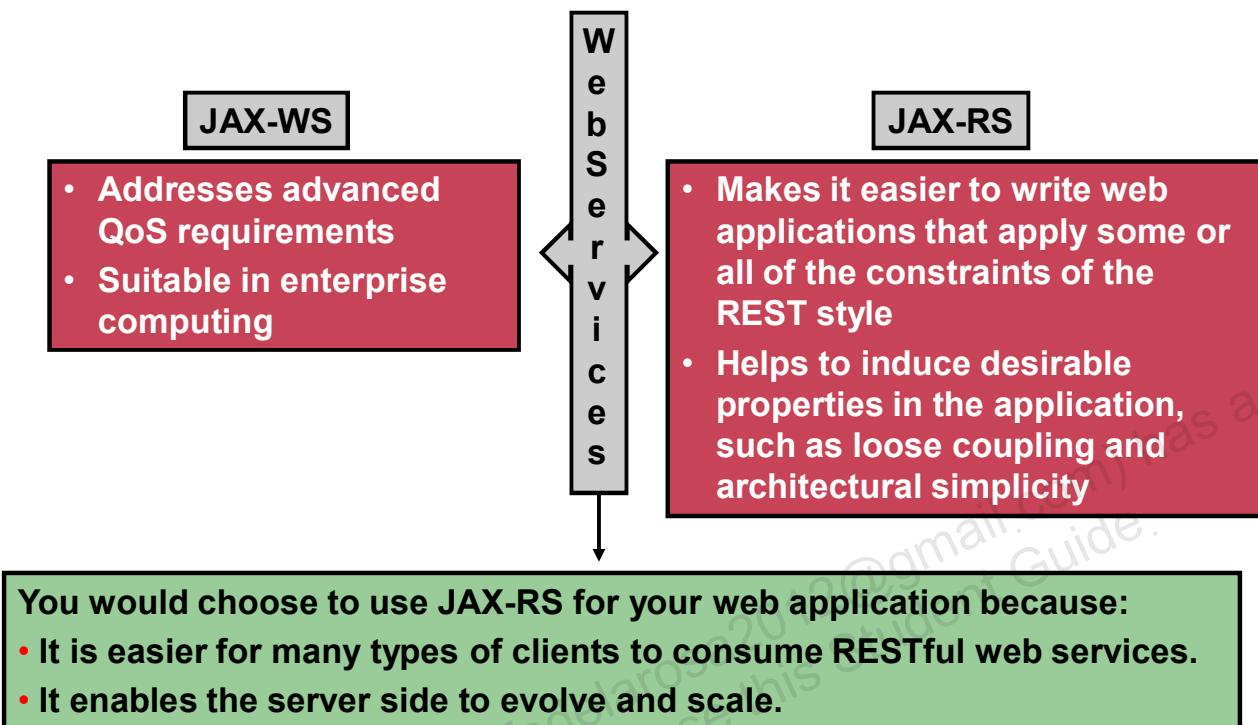
SOAP based web services use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Such systems often contain a machine-readable description of the operations offered by the service, written in the Web Services Description Language (WSDL), an XML language for defining interfaces syntactically.

REST is well suited for basic, ad hoc integration scenarios. RESTful web services, often better integrated with HTTP than SOAP-based services are, do not require XML messages or WSDL service API definitions.

RESTful web services are based on the JSR-311 specification, JAX-RS API. Jersey is a reference implementation of JAX-RS .

RESTful web services use existing well-known W3C and Internet Engineering Task Force (IETF) standards (HTTP, XML, URI, MIME) and have a lightweight infrastructure that allows services to be built with minimal tooling. Developing RESTful web services is inexpensive and, therefore, has a very low barrier for adoption. You can use a development tool such as NetBeans IDE to further reduce the complexity of developing RESTful web services.

Which Type of Web Service to Use?



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

You would want to use RESTful web services for integration over the web and use SOAP-based web services in enterprise application integration scenarios that have advanced quality of service (QoS) requirements.

When compared to JAX-RS, JAX-WS makes it easier to support the WS-* set of protocols, which provide standards for security and reliability, among other things, and interoperate with other WS-*–conforming clients and servers.

You would choose to use JAX-RS for your web application because it is easier for many types of clients to consume RESTful web services while enabling the server side to evolve and scale. Clients can choose to consume some or all aspects of the service and mash it up with other web-based services, because it is easy to modify the server without breaking existing clients.

When to Use REST

A RESTful design may be appropriate when:

1. The web services are completely stateless.
2. A caching infrastructure can be leveraged for performance.
3. The service producer and service consumer have a mutual understanding of the context and content being passed along.
4. Bandwidth is particularly important, and limited.
5. Web service delivery or aggregation into existing websites can be enabled easily with a RESTful style.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

This slide provides an analysis for using REST with design point of view. Developers must decide when this particular style is an appropriate choice for their applications. Each point in the slide is elaborated below:

1. A good test to check whether the web service is stateless is to consider whether the interaction can survive a restart of the server.
2. If the data that the web service returns is not dynamically generated and can be cached, then the caching infrastructure that web servers and other intermediaries inherently provide can be leveraged to improve performance. However, the developer must take care because such caches are limited to the HTTP GET method for most servers.
3. Because there is no formal way to describe the web services interface, both parties must agree on the schemas that describe the data being exchanged and on ways to process it meaningfully.
4. REST is particularly useful for devices such as PDAs and mobile phones, for which the overhead of headers and additional layers of SOAP elements on the XML payload must be restricted.
5. Rather than starting from scratch, services can be exposed with XML and consumed by HTML pages without significantly refactoring the existing website architecture.

Principles of a RESTful Web Service

RESTful applications are simple, lightweight, and fast because:

- **Resources** are identified by URIs, which provide a global addressing space.
- **A uniform interface** is used for resource manipulation.
- **Self-descriptive messages** or metadata about the resource is available and used.
- **Stateful interactions through hyperlinks** are based on the concept of explicit state transfer.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

REST is a key design idiom that represents a stateless client-server architecture. A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients. Resources are identified by URIs, which provide a global addressing space for resource and service discovery.

Resources are manipulated by a fixed set of four create, read, update, and delete operations: PUT, GET, POST, and DELETE.

Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, and JSON. Metadata about the resource is available and used, for example, to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.

Every interaction with a resource is stateless; that is, request messages are self-contained. Stateful interactions are based on the concept of explicit state transfer. Several techniques exist to exchange state, such as URI rewriting, cookies, and hidden form fields. State can be embedded in response messages to point to valid future states of the interaction.

Web service clients that want to use these resources access a particular representation by transferring application content using a small, globally defined set of remote methods that describe the action to be performed on the resource.

Relationships Between SQL and HTTP Verbs

For resource manipulation, an analogy to operations in SQL is made, which also relies on a few common verbs, as shown in this table:

Action	SQL	HTTP
Create	Insert	PUT
Read	Select	GET
Update	Update	POST
Delete	Delete	DELETE



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

With RESTful web services, there is a natural mapping between the HTTP methods and most CRUD-like business operations that many services expose. Though there are no hard-and-fast rules, the following general guidelines are applicable for most cases:

- GET is used to retrieve data or to perform a query on a resource. The data returned from the web service is a representation of the requested resource.
- PUT is used to create a new resource. The web service may respond with data or a status indicating success or failure.
- POST is used to update existing resources or data.
- DELETE is used to remove a resource or data.

In some cases, the update and delete actions may be performed with POST operations as well (for example, when the services are consumed by browsers that do not support PUT or DELETE).

Sometimes, the following mapping might be applicable for PUT and POST:

- Create = PUT if you are sending the full content of the specified resource (URL)
- Create = POST if you are sending a command to the server to create a subordinate of the specified resource, using some server-side algorithm
- Update = PUT if you are updating the full content of the specified resource
- Update = POST if you are requesting the server to update one or more subordinates of the specified resource

Developing a RESTful Web Service with JAX-RS

- JAX-RS, a Java programming language API:
 - Is designed to make it easy to develop applications that use the REST architecture
 - Uses Java programming language annotations to simplify the development of RESTful web services
 - Uses annotations that are runtime annotations, and therefore, runtime reflection will generate the helper classes and artifacts for the resource
- In the Java programming language class files, you use JAX-RS annotations to define resources and the actions that can be performed on those resources.
- Project Jersey is the reference implementation for the JAX-RS specification.

 ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Jersey implements support for the annotations defined in the JAX-RS specification, making it easy for developers to build RESTful web services with Java and the Java Virtual Machine (JVM).

A Java EE application archive containing JAX-RS resource classes will have the resources configured, the helper classes and artifacts generated, and the resource exposed to clients by deploying the archive to a Java EE server.

Detailed information about RESTful web services can be obtained from
<http://docs.oracle.com/javaee/6/tutorial/doc/giepu.html>.

RESTful Web Service with JAX-RS: An Example

```
import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import javax.ws.rs.Path;
//The Java class will be hosted at the URI path
//"/hello"
@Path("/hello")
public class Hello {
    // The Java method will process HTTP GET requests
    @GET
    // The Java method will produce content identified by the
    //MIME Media type "text/plain"
    @Produces("text/plain")
    public String sayHello() {
        // Return some textual content
        return "Hello World";
    }
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The code sample in the slide is a very simple example of a root resource class that uses JAX-RS annotations.

The annotations used in the example can be explained as follows:

- The `@Path` annotation's value is a relative URI path. The Java class will be hosted at the URI path `/helloworld`.
 - This is an extremely simple use of the `@Path` annotation, with a static URI path.
 - Variables can be embedded in the URIs.
 - *URI path templates* are URIs with variables embedded within the URI syntax.
- The `@GET` annotation is a request method designator, along with `@POST`, `@PUT`, `@DELETE`, and `@HEAD`, defined by JAX-RS and corresponding to the similarly named HTTP methods. In the example, the annotated Java method will process HTTP GET requests. The behavior of a resource is determined by the HTTP method to which the resource is responding.
- The `@Produces` annotation is used to specify the MIME media types that a resource can produce and send back to the client. In this example, the Java method will produce representations identified by the MIME media type "text/plain".

- The @Consumes annotation is used to specify the MIME media types that a resource can consume that were sent by the client.
 - The example could be modified to set the message returned by the sayHello method, as shown in this code example:

```
@POST @Consumes("text/plain")
public void sendMessage(String message)
{ // Store the message
}
```

Adolfo De+la+Rosa (adolfo.delarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.

Quiz

What is a web service?

- a. An application that communicates over the World Wide Web's (WWW) Hyper Text Transfer Protocol (HTTP)
- b. An application that provides a standard means of interoperating between software applications running on a variety of platforms and frameworks
- c. An application that is interoperable and extensible because of using XML
- d. A software component provided in the client tier

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, c

Quiz

Identify the statements that are true about RESTful web services:

- a. REST is a key design idiom that represents a stateful client-server architecture.
- b. A RESTful web service exposes a set of resources that identify the targets of the interaction with its clients.
- c. Resources are defined as tables that provide a global addressing space for resource and service discovery.
- d. Resources are manipulated by a fixed set of four create, read, update, and delete operations: PUT, GET, POST, and DELETE.

 ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b, d

Topics

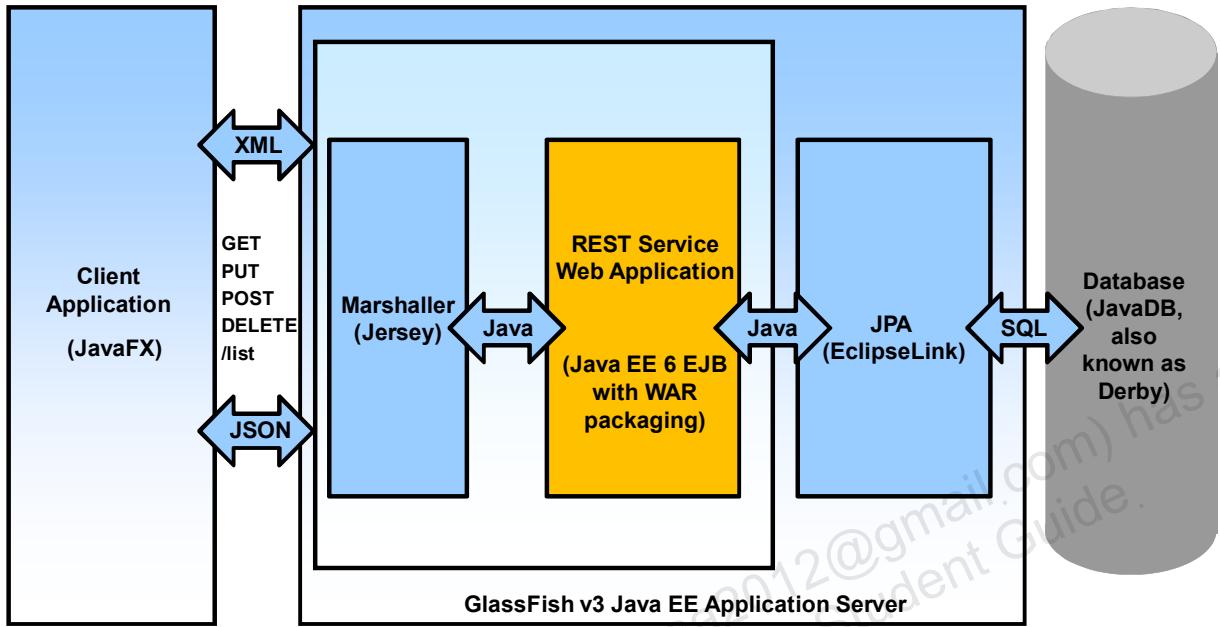
- Three-tier design versus two-tier design
- JAX-RS web services
- JAX-RS web services in the HenleyServer application



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Three-Tier Architecture Using REST



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Steps to Generate RESTful Web Services in NetBeans

1. Ensure that the prerequisites are met:
 - Add Jersey 1.8 to the project library.
 - Add JAX-RS 1.1 API to the project library.
 - Add the HenleyModel project with Entity classes to the HenleyServer project.
 - Ensure that the connection pool and JDBC data source has been created on the GlassFish server.
 - Ensure that the persistence unit has been created and configured in the project.
 - Ensure that the JAXB annotations are added to the JPA Entity classes.
2. Generate web services:
 - Create RESTful web services from the entity classes.
 - Validate the generated web service classes.
 - Validate the configuration in the web.xml file.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

This slide provides an overview of the steps involved in generating RESTful web services using NetBeans.

JAXB annotations are added directly to JPA entity classes.

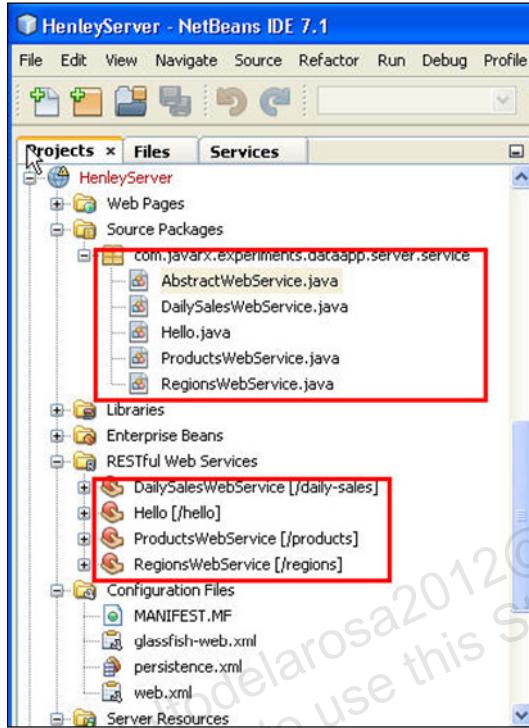
Services are generated as EJB session facades.

A session facade is a design pattern. As stated in the core J2EE pattern catalog, it attempts to resolve common problems that arise in a multi-tiered application environment, such as:

- Tight coupling, which leads to direct dependence between clients and business objects
- Too many method invocations between client and server, leading to network performance problems
- Lack of a uniform client access strategy, exposing business objects to misuse

A session facade abstracts an application's underlying business object interactions and provides a service layer that exposes only the required functionality. Thus, the session façade hides from the client's view the complex interactions between the objects. The session bean also manages the life cycle of business objects. The session bean creates, locates, modifies, and deletes objects as required by the workflow.

Examine the RESTful Web Services in HenleyServer



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

RESTful web services in Java rely on the JPA to communicate with a database. Specifically, RESTful web services rely on *entity classes* and a *persistence unit*, as defined in the Persistence API. Entity classes are Java classes that map to objects in a relational database.

The screenshot in the slide shows the list of RESTful web services generated in the HenleyServer application.

By creating web services as session facades, NetBeans IDE creates one service class for each entity class. The figure shows the project structure of a Java EE 6 RESTful service created from the entity classes, but with the service classes generated as session facades.

The other service classes extend the abstract session facade and call on its methods for a specific entity class. NetBeans IDE creates an abstract session facade class. This abstract class contains the code used by all service classes for managing entities.

Examining the DailySales Web Service Code

```
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
@Stateless
@Path("/daily-sales")
public class DailySalesWebService extends
AbstractWebService<DailySales> {
    public DailySalesWebService() {
        super(DailySales.class);
    }
@GET
@Override public List<DailySales> findAll() {
    return super.findAll();
}
@POST
@Override public void create(DailySales entity) {
    super.create(entity);
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The code in the slide shows the DailySalesWebService class. The use of session facades results in simpler code. Customizing and maintaining this code should also be much easier.

The DailySalesWebService class calls the methods of AbstractWebService, replacing the generic class parameters with references to the DailySales entity class.

The service class is easy to read, consisting of HTTP method-annotated method calls that refer back to the abstract class.

Testing DailySales RESTful Service

- Deploy and run the HenleyServer project.
- Test the web service URL in a web-browser:

```
http://localhost:8080/HenleyServer/resources/daily-sales
```

- The result of the get operation shows all the records of the Daily-Sales table in XML format.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Quiz

Select the correct statements about applying JAX-RS in your application:

- a. You would choose to use JAX-RS for your web application because it is easier for many types of clients to consume RESTful web services while enabling the server side to evolve and scale.
- b. RESTful web services in Java rely on HTTP to communicate with a database.
- c. Jersey is the JAX-RS RI and is used to implement RESTful web services in your application.
- d. RESTful web services rely on *entity classes* and a *persistence unit*, as defined in the Persistence API.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, c, d

Summary

In this lesson, you should have learned how to:

- Compare the HenleyApp two-tier design and the HenleyApp three-tier design
- Describe a RESTful web service
- List the web services used in the HenleyApp application
- Describe how the RESTful web services were developed in the HenleyServer application



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Practice 11: Overview

- Practice 11-1: Reviewing Basic Concepts of Java Web Services
- Practice 11-2: Examining BrokerToolServer's Web Services



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

12

Connecting to a RESTful Web Service

ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe how to test a RESTful Web Service
- Identify how to develop a Jersey RESTful Client
- Review the implementation of web service clients in the HenleyApp application



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Topics

- Test RESTful web services
- Steps to develop JAX-RS web service clients
- Web service clients of HenleyApp



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Testing a RESTful Web Service

Check the following while testing a RESTful web service:

- That the URL address correctly represents the service deployment endpoint and the method annotations
- That the server requests(GET, PUT, DELETE, or POST) that are generated invoke the appropriate methods of the web service.
- That the methods return acceptable data

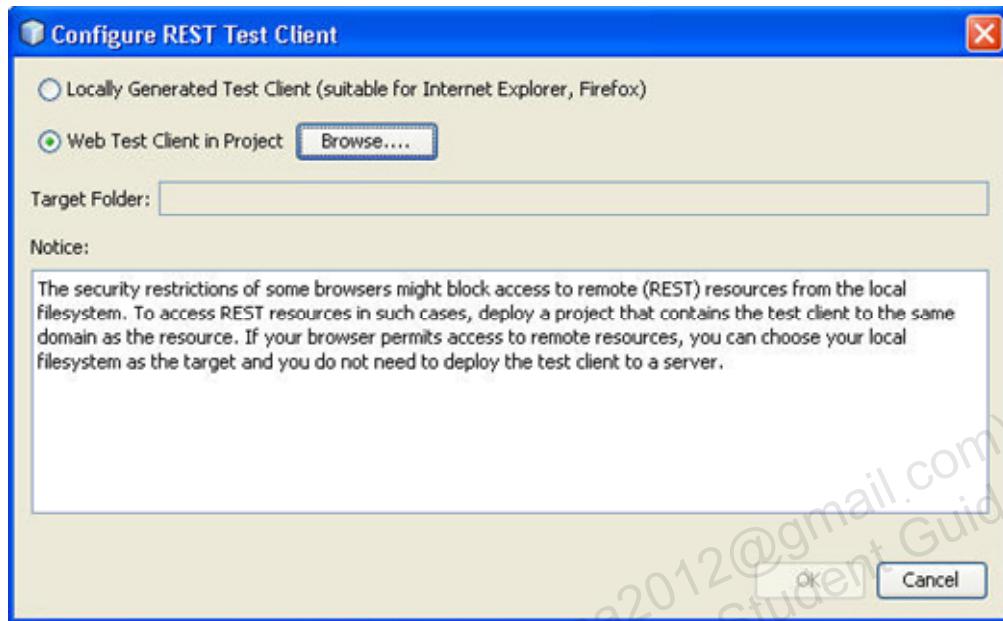


Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

There are various tools and software to test a RESTful web service. NetBeans IDE also provides various options to test a RESTful web service within the same project.

Later in this lesson, a slide shows the service requests that are generated for each resource.

Testing HenleyServer's RESTful Web Services



ORACLE®

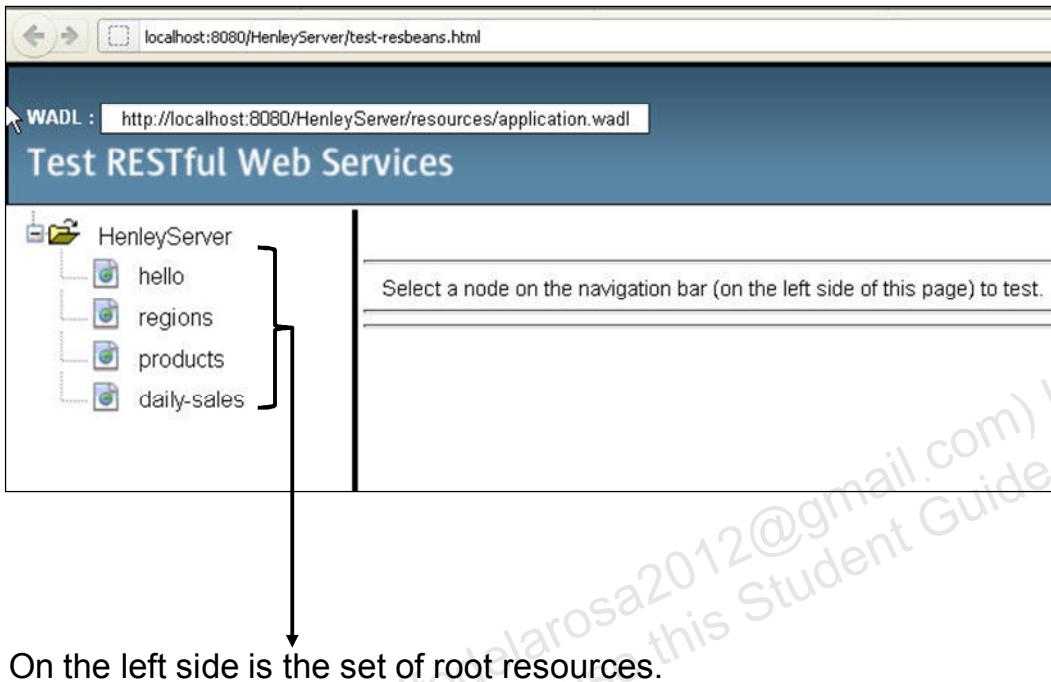
Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

To test a RESTful web service in NetBeans:

- Right-click the project node and select Test RESTful Web Services. A dialog box opens asking whether you want to generate the test client inside the service project or in another Java web project. This option lets you work around security restrictions in some browsers. You can use any web project, as long as it is configured to deploy in the same server domain as the HenleyServer project.

The screenshot in the slide shows the two options for the target location of the generated test client.

Testing HenleyServer's RESTful Web Services

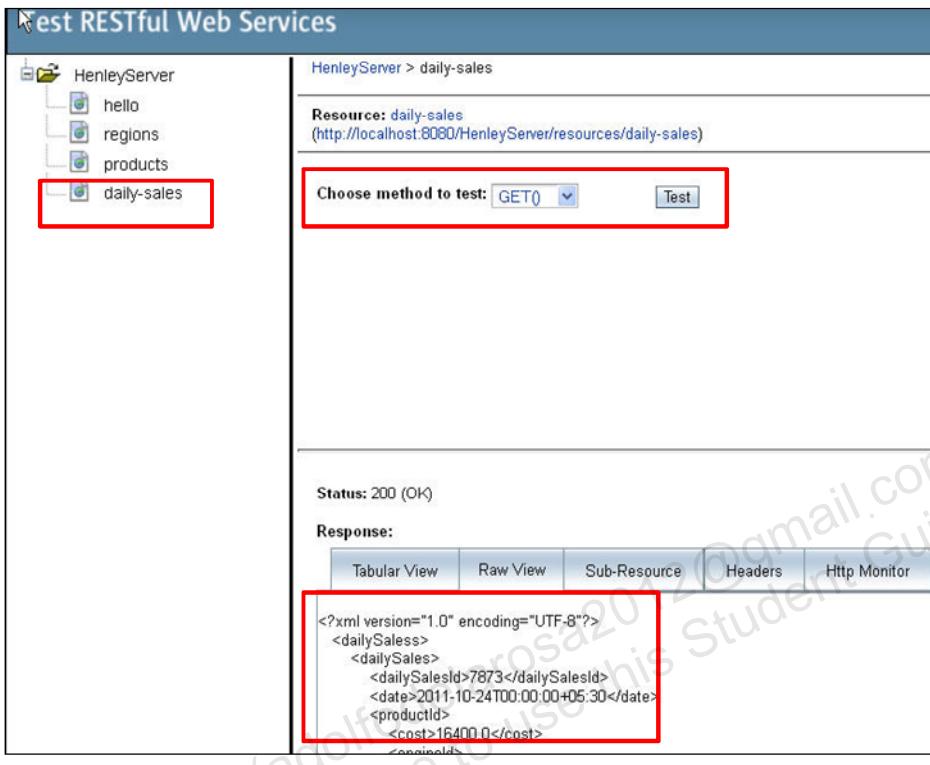


ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

After you have selected where to generate the test client, click OK. The server starts and the application is deployed. When deployment is complete, the browser displays your application, with a link for each of the web services.

Testing HenleyServer's RESTful Web Services



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Click one resource node. In the "Choose method to test" field, select either GET or POST, and then click Test. The test client sends a request and displays the result in the Test Output section.

The test client displays the Raw View by default. The image in the slide shows the response to an application/JSON request.

Testing HenleyServer's RESTful Web Services

Tab Name	Description
Tabular View	A flattened view that displays all the URIs in the resulting document. Currently, this view only displays a warning that Container-Containee relationships are not allowed.
Raw View	Displays the actual data returned. Depending on which mime type you selected (application/xml), the data displayed will be XML format, respectively.
Sub-Resource	Shows the URLs of the root resource and sub-resources. When the RESTful web service is based on database entity classes, the root resource represents the database table, and the sub-resources represent the columns.
Headers	Displays the HTTP header information
Http Monitor	Displays the actual HTTP requests and responses sent and received



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

There are five tabs in the Test Output section of the Test Client window, as shown in the table in the slide.

Exit the browser and return to the IDE.

Topics

- Test RESTful web services
- Steps to develop JAX-RS web service clients
- Web service clients of HenleyApp



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Steps to Develop a Restful Web Service Client

1. Ensure that the project has the required libraries added:
 - JAX-RI
 - Jersey
2. Identify the GUI window and control where the results of the web service invocation will be displayed.
3. Create a new RESTful service client using a wizard. The following information is important:
 - The URL of the RESTful service
 - The package name
 - The class where the client code will be generated
4. Invoke the generated code in the GUI window appropriately.

 ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The slide lists the steps to develop a web service client. You can write code to consume a RESTful web service in any Java SE/Java web or in a JavaFX application. NetBeans provides a wizard to easily do the same.

You must know the URL of the service that you want to consume in your client program.

Examine the Generated Web Service Client Code

```
...
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.UniformInterfaceException;
import com.sun.jersey.api.client.WebResource;
public class HelloClient {
    private WebResource webResource;
    private Client client;
    private static final String BASE_URI =
        "http://localhost:8080/HenleyServer/resources";

    public HelloClient() {
        com.sun.jersey.api.client.config.ClientConfig config = new
        com.sun.jersey.api.client.config.DefaultClientConfig();
        client = Client.create(config);
        webResource = client.resource(BASE_URI).path("hello");
    }

    public String sayPlainTextHello() throws UniformInterfaceException {
        WebResource resource = webResource;
        return
            resource.accept(javax.ws.rs.core.MediaType.TEXT_PLAIN).get(String.cl
        ass);
    }

    public void close() {
        client.destroy();
    }
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The code snippet in the slide shows NetBeans-generated code.

Topics

- Test RESTful web services
- Steps to develop JAX-RS web service clients
- Web service clients of HenleyApp



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Apply the Web Service Client Code in HenleyClient

```
public class GetDailySalesTask extends Task<ObservableList<DailySales>> {
    private static final ClientConfig CONFIG = new DefaultClientConfig();
    static {
        CONFIG.getFeatures().put(JSONConfiguration.FEATURE_POJO_MAPPING,
        Boolean.TRUE);
    }
    @Override protected ObservableList<DailySales> call() throws Exception
    {
        Client client = Client.create(CONFIG);
        WebResource webResource =
client.resource("http://localhost:8080/HenleyServer/resources").path("dail
y-sales");
        ClientResponse response = webResource.get(ClientResponse.class);
        GenericType<List<DailySales>> genericType = new
GenericType<List<DailySales>>() {};
        // Return an ObservableList backed by the ArrayList of DailySales
        from the web service
        return
FXCollections.observableArrayList(response.getEntity(genericType));
    }
}
```

The code to access the web service URL and invoke the GET operation

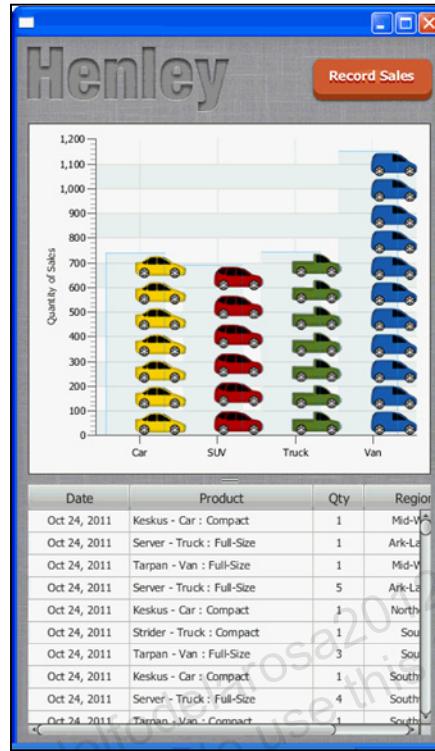


Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The lines of code in the slide is from GetDailySalestask.java of the HenleyClient4 project.

The webResource.get (ClientResponse.class) invokes the findAll method of the daily-sales web service.

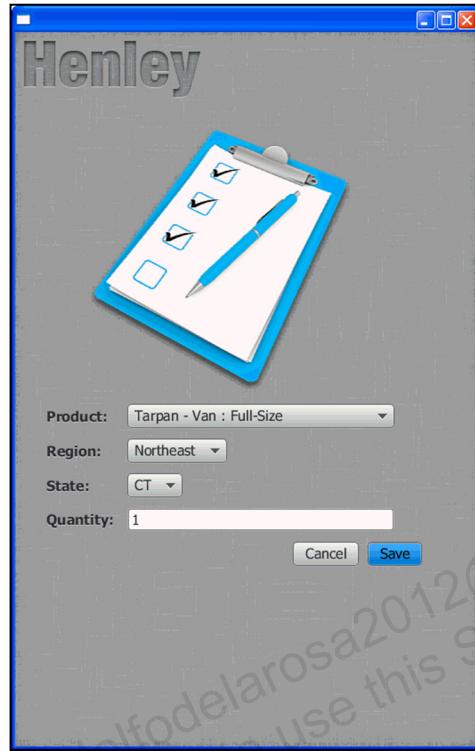
Verify the Output in HenleyClient



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Sales Form to Insert Details



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The user provides the sales information in the sales form (`SalesForm.fxml`) as shown in the slide, and then clicks the Save button. The `SalesForm.java` class is executed.

Examining the Code for the POST/Insert Operation

```
...  
final Task<List<Product>> saveSaleTask = new Task() {  
    @Override protected Object call() throws Exception  
{  
    Client client = Client.create(CONFIG);  
    WebResource dailySalesWebResource =  
client.resource("http://localhost:8080/HenleyServer/resou  
rces").path("daily-sales");  
  
    dailySalesWebResource.type(javax.ws.rs.core.MediaType.APP  
PLICATION_JSON).post(sale);  
    return null;  
}  
}
```

The daily-sales web service URL

The POST operation/create method/insert statement (for a daily_sale record) is invoked.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The code snippet in the slide is from SalesForm.java of HenleyClient4 project:

1. A task is started, which invokes the call method.
2. The daily-sales web service is consumed through its URL.
3. The create or the post method of the daily-sales service is invoked.
4. The create method inserts the from data as a record in the DailySales table.

These steps execute when the user clicks the Save button of the SalesForm.fxml form.

By invoking the Dashboard.fxml file, you can verify the record insertion.

Quiz

What do you need to create a RESTful client? Select from the list below:

- a. The URL of the RESTful service
- b. Names of all the methods that are exposed by the web service
- c. JAX-RS RI and Jersey libraries
- d. A HTML web page to invoke the web service



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Summary

In this lesson, you should have learned how to:

- Describe how to test a RESTful Web Service
- Identify how to develop a Jersey RESTful Client
- Review the implementation of web service clients in the HenleyApp application



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Practice 12: Overview

- Practice 12-1: Testing RESTful Web Services
- Practice 12-2: Creating a RESTful Web Service Client



ORACLE

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Adolfo De+la+Rosa (adolfodelarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.

Packaging and Deploying Applications

13

ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe a JAR file and the steps for creating one
- Create a manifest file that uses headers
- Create a JAR by using development tools
- Deploy a stand-alone JAR
- Deploy a JAR as an applet
- Deploy a JAR by using Java Web Start

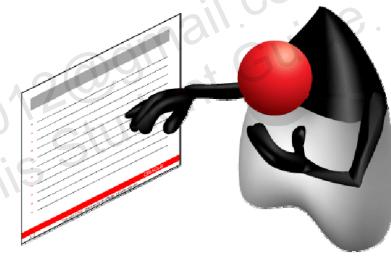


ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Topics

- Packaging
- Deployment
- JavaFX Deployment



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Packaging Introduction

- Running a Java class
 - `java classname`
 - `java package.dir.classname`
- Examples:
 - `java BaseUi`
 - `java com.example.ui.BaseUi`
- What is the problem?
 - Not practical with hundreds of `.class` files
 - Not portable



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Running a Java application is pretty straightforward. In a command line, type `java` and the class name: `java BaseUi`.

Note that if you include the `.class` extension, you will receive an error message, because Java interprets `BaseUi.class` as the "class" class in the `BaseUi` package.

Running your application this way works fine on a small scale, but what happens when you have dozens, if not hundreds, of class files? Then you need a better solution for organizing and distributing your files.

Java jar Files

- The `jar` utility is used to create JAR (or “`.jar`”) files.
- What is a `.jar` file?
 - Essentially a `.zip` file
 - Contains all your `.class` files
 - Contains any resources needed by your program
 - Any special Java stuff
- How to you use the `jar` utility?
 - Command line
 - From another tool

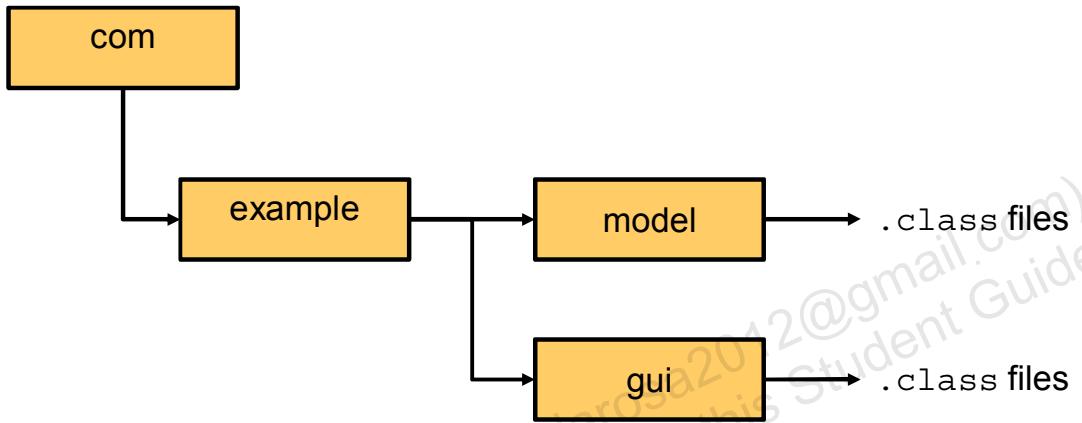
ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Java applications are packaged and deployed using `.jar` files. These files are created using the `jar` tool. You can use the `jar` tool from the command line. However, typically the `jar` tool is most often used by other tools to create `.jar` files.

Application Example

A typical application and its packages:



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This picture shows a typical directory structure for an application. Notice that the domain name precedes the directories, which contain the .class files.

jar Command Line

- Basic command line:
 - jar cvf jarFileName dirName
- Example:
 - jar cvf BasicUi.jar com
- Options:
 - c: Create the .jar file.
 - v: Produce verbose output.
 - f: Send the output to a file instead of to standard output.
- Running a .jar file:
 - java -jar MyJar.jar
 - Double-clicking the .jar file works in most situations, too.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The slide shows the basic command line options for `jar`. If you refer to the previous slide, the command in the example creates a .jar file from the "com" directory and all its subdirectories. If there are no other subdirectories in your current directory, you could also have used:

```
jar cvf BasicUi.jar *
```

However, how do we determine which class file to run?

The Manifest File

- A manifest file is created whenever a .jar is created.
 - Located in META-INF directory
 - Named MANIFEST.MF
- Sample contents:

```
Manifest-Version: 1.0
Created-By: 1.7.0
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

When you create a .jar file, the jar utility automatically creates a manifest file unless you specify one. The slide shows sample contents of a manifest. Each line consists of a header, the text before the colon, and its value. The first line says that the file conforms to version 1.0 of the manifest file standard. The second line says that the jar file was created by JDK 7.

Modifying the Manifest

- Example command line:
 - jar cfm BaseUi.jar Manifest.txt com
- Common headers:
 - Main-Class:
 - Class-Path:
- Example:

```
Main-Class: com.example.ui.BaseUi  
Class-Path: HelperLib.jar HelperLib2.jar
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

You can provide your own manifest file settings in a regular text file. Just pass the file on the command line and the headers will be added to the manifest in your .jar. Two of the most common headers are included in the slide.

Main-Class: Sets the class that should be run when the .jar file is executed.

Class-Path: Specifies the .jar files to be included in the CLASSPATH. This saves you from having to specify the .jar files in the command line.

Tools for Making .jar Files

Commonly, tools are used to create .jar files.

- Ant
 - "Another Neat Tool"
 - A Java build, test, and deployment tool
 - Written in Java
- JavaFX Packager
- NetBeans
 - Automatically creates the .jar file for you
 - Relies on Ant scripts for building and packaging



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Quiz

Which manifest header allows you to specify the class you should launch at run time?

- a. Run-Class:
- b. Launch-Class:
- c. Main-Class:
- d. Super-Class:

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Topics

- Packaging
- Deployment
- JavaFX Deployment



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Deploying Applications

- Distribute the .jar file.
 - Just make the .jar file available.
- Use an installer.
 - Use a professional installer to set up and install your jar.
- Embedded
 - Embed applications in a web page.
- Java Web Start
 - Launch a Java Application from a URL.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

There are a number of ways to deploy an application. Some methods are very manual, other are much more automated.

Stand-Alone Deployment

- Double-click the .jar file on most platforms.
- Command line:
 - `java -jar jarfilename.jar`
- Put the .jar in an .exe wrapper.
- Security:
 - Full access to the system



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Many Java applications are deployed as stand-alone applications. NetBeans is a great example of this type of deployment. When deployed stand-alone, a Java application has the same security access to your system resources as any other application (specifically, access to things like files, directories, and network shares).

On most operating systems, if Java (JDK or JRE) is installed, you can simply double-click the .jar file to launch it.

Installers and Wrappers

- **Installers**
 - install4j – Commercial
 - <http://www.ej-technologies.com/products/install4j/overview.html>
 - Izpack - Open Source (Apache)
 - <http://izpack.org/>
- **Wrappers**
 - launch4j - Open Source (BSD, MIT)
 - <http://launch4j.sourceforge.net/>
 - jsSmooth -
 - <http://jssmooth.sourceforge.net/>

Note: The inclusion of products in this list is not an endorsement by Oracle.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Typically, many popular tools, such as NetBeans or Eclipse, are written in Java, but are installed just like a normal application. This is achieved through the use of an installation program or a wrapper program.

An installation program provides a complete framework for installing an application on a platform.

A wrapper program allows you to add platform-specific execution to your application. For example, you can execute your Java program on Windows by using an .exe file. In addition, wrapper programs can provide more functionality, such as detecting whether a JRE is present and auto-installing one if necessary.

Applets/Embedded

- Allows Java code to run within a web browser
- Introduced in the first version of Java
 - Very popular at the time (1995)
 - One of the main reasons for early Java adoption
 - Flash and AJAX caused popularity to diminish over time
- Applet functionality is still available in Java FX.
 - Applets can still be used in JDK 7.
 - However, JavaFX can be embedded in web pages.
 - Embedding Java FX is the preferred method for delivering Java in a browser.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

One cannot talk about Java deployment without mentioning applets. Applets are Java-based programs, typically GUIs, that are designed to run in a web browser. Applets were released with the first version of Java in 1995 and were a big reason for the growth of interest in the language and platform.

Although applets are included in JDK 7, they are not the preferred method for including a Java FX application in a web browser.

Applet Security

- Applets introduced the concept of a sandbox.
- Unsigned applets have very limited access.
 - Cannot access the local file system
 - Cannot access another server
 - Cannot load native libraries
 - Cannot change the security manager or create a ClassLoader
- Signed applets have access to the local system.
- Java web-based deployment technologies continue to follow this model.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Applets introduced the concept of a sandbox. This sandbox greatly limits the applications access to the local system. Limitations include:

- They cannot access client resources such as the local filesystem, executable files, system clipboard, and printers.
- They cannot connect to or retrieve resources from any third-party server (any server other than the server it originated from).
- They cannot load native libraries.
- They cannot change the SecurityManager.
- They cannot create a ClassLoader.
- They cannot read certain system properties.

The only way to get access to the local system is by signing the applet. The security model is still followed for newer systems like Java Web Start and the Deployment Kit.

Java Web Start

- Designed to launch full-featured Java applications from the Internet
- Complete client applications, such as a spreadsheet or editor
- How it works:
 - A web page
 - A link to a Java Network Launch Protocol (JNLP) file
 - An installation of Java on the client machine
- Uses a security model like applets



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

In contrast to applets, Java Web Start (JWS) was designed to launch full Java clients from a web page, not run in the web page.

Java Web Start Benefits

- The application can be deployed from a browser.
- The application is cached locally after the initial download.
- Shortcuts can be created to the locally cached apps.
- Updates can be distributed from the network automatically.



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Java Web Start offers a number of benefits. Instead of being limited to a browser windows, Java Web Start can be used to distribute full-fledged applications to the desktop. Applications can be distributed using the network and cached locally. This makes distribution and updating very easy.

Quiz

Which two deployment methods run their applications in a sandbox by default?

- a. Browser-based deployment
- b. Java Web Start
- c. Stand-alone deployment
- d. Wrapping your .jar in an .exe file



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Topics

- Packaging
- Deployment
- JavaFX Deployment

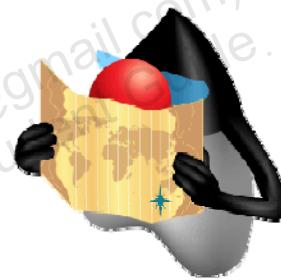


ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Deployment Toolkit

- A set of JavaScript functions to help deploy Rich Internet Applications (RIA) written in Java
 - Correct HTML based on browser version and operating system
 - Checks that the required JRE is present
 - Prompts for installation if it is not
 - Introduced in Java 6 Update 10
- The JavaScript file is located at:
 - <http://www.java.com/js/deployJava.js>
 - <https://www.java.com/js/deployJava.js>
- For NetBeans development:
 - ./web-files/dtjava.js



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Deployment Toolkit helps deploy rich Internet applications written in Java. It corrects all the HTML so that Java and JavaFX apps will run correctly on any browser or operating system. The Deployment Toolkit is used to deploy a JavaFX application in a browser or by using Java Web Start.

If your application is deployed in an Internet-accessible environment, use the java.com URL to refer to the toolkit. However, when developing an application, NetBeans will include a copy of the Deployment Toolkit in the local directory.

Deployment Toolkit Methods

- Key JavaScript deployment methods:
 - `dtjava.embed(app, platform, callbacks)`
 - Embeds the application into the browser
 - `dtjava.launch(app, platform, callbacks)`
 - Launches applications outside of the browser
- Utility methods:
 - `dtjava.install(platform, callbacks)`
 - Initiates installation of required components
 - `dtjava.validate(platform)`
 - Validates that the user environment satisfies platform requirements



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

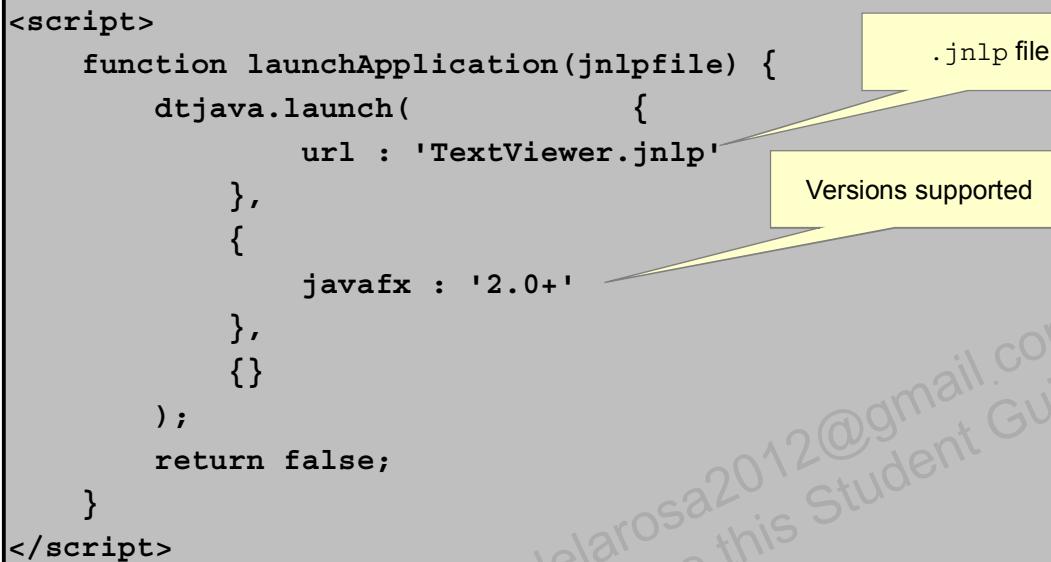
The Deployment Toolkit API provides several important methods:

- `dtjava.embed(app, platform, callbacks)` embeds the application into the browser based on a given application descriptor. If Java Runtime or JavaFX Runtime installation is required, it will invite the user to click to install it.
- `dtjava.launch(app, platform, callbacks)` launches applications that are not embedded in the browser, based on a given application descriptor. If Java Runtime or a JavaFX Runtime installation is required, then it makes attempts to start the installer.
- `dtjava.install(platform, callbacks)` initiates the installation of required components according to platform requirements.
- `dtjava.validate(platform)` validates that the user environment satisfies platform requirements (for example, if a required version of JavaFX is available). It returns `PlatformMismatchEvent`, which describes problems, if any.

Java Web Start Deployment

Here is an example of the JavaScript included in the browser:

```
<script>
    function launchApplication(jnlpfile) {
        dtjava.launch(
            {
                url : 'TextViewer.jnlp'
            },
            {
                javafx : '2.0+'
            },
            {}
        );
        return false;
    }
</script>
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Deployment Toolkit uses JavaScript object literal notation to specify the key information needed to launch the application. Note that the URL to the .jnlp file is included. In addition, you must specify which version of JavaFX is supported; in this case, any version 2.0 or later of JavaFX.

Java Web Start HTML Link

HTML launch link:

```
<b>Webstart:</b>
<a href='TextViewer.jnlp' onclick="return
launchApplication('TextViewer.jnlp');">click to launch
this app as webstart</a>
<br/>
<hr/>
<br/>
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Even with the launch script included in the header of the HTML document, you must still add a link to launch the application in the body of the document. This link in the body of the document calls the JavaScript and launches your JavaFX application.

JNLP Application Descriptor

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0" xmlns:jfx="http://javafx.com" href="TextViewer.jnlp">
  <information>
    <title>TextViewer</title>
    <vendor>Your Name Here</vendor>
    <description>Sample JavaFX 2.0 application.</description>
    <offline-allowed/>
  </information>
  <resources os="Windows">
    <jfx: javafx-runtime version="2.0+"
      href="http://javapl.sun.com/webapps/download/GetFile/javafx-latest/windows-
      i586/javafx2.jnlp"/>
  </resources>
  <resources>
    <j2se version="1.6+" href="http://java.sun.com/products/autodl/j2se"/>
    <jar href="TextViewer.jar" size="17431" download="eager" />
  </resources>
  <applet-desc width="800" height="600" main-
    class="com.javafx.main.NoJavaFXFallback" name="TextViewer" />
  <jfx: javafx-desc width="800" height="600" main-
    class="com.example.gui.TextViewer" name="TextViewer" />
  <update check="background"/>
</jnlp>
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

This slide shows the JNLP application descriptor for the TestViewer application. Note that elements are included for embedded launch (the `applet-desc` tag) and Java Web Start. The screen size is also included here.

Web Browser Deployment

```
<script>
    function javafxEmbed() {
        dtjava.embed(
            {
                url : 'TextViewer.jnlp',
                placeholder : 'javafx-app-placeholder',
                width : 800,
                height : 600,
                jnlp_content : 'PD94bWwgdmVyc2lvbj0iMS4w'
            },
            {
                javafx : '2.0+'
            },
            {}
        );
    }
    <!-- Embed FX application into web page once page is loaded -->
    dtjava.addOnloadCallback(javafxEmbed);
</script>
```

Base64 encoded
JNLP file



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Here is the JavaScript for deploying your application in a browser. Notice a callback method is used to launch the application once the page loads. Notice the "jnlp content" attribute in the middle of the page. This allows you to Base64 encode your .jnlp file and include it in your page. This should give you a minor speed improvement on initial page load, because the browser does not have to make a second request to get the .jnlp file.

Note: In this example, the Base64 data has been truncated so that the code example fits in the slide.

Deployment Toolkit Callbacks

The deployment toolkit includes callback functions. Here are a few of them:

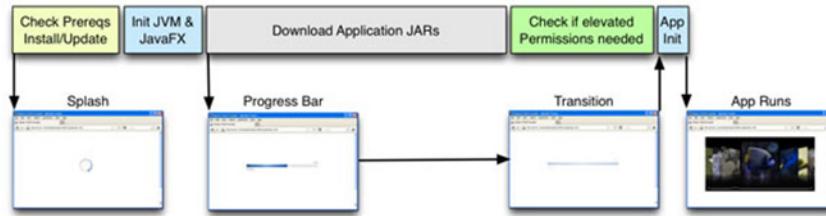
- `onDeployError: function(app, mismatchEvent)`
 - Called when platform requirements are not met
- `onInstallFinished: function(placeholder, component, status, relaunchNeeded)`
 - Called once installation of a required component is completed
- `onInstallNeeded: function(app, platform, cb, isAutoinstall, needRelaunch, launchFunc)`
 - Called if embedding or launching an application needs additional components to be installed
- `onInstallStarted - function(placeholder, component, isAuto, restartNeeded)`
 - Called before installation of the required component is triggered



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Callback JavaScript functions are included to allow you handle a number of event and situations that might arise when trying to deploy your application. The examples provided here are just a subset of what you can use. See the *Java FX Deployment Guide* for details:
<http://docs.oracle.com/javafx/2.0/deployment/jfxpub-deployment.htm>

Application Startup Process



- Initialization
- Loading and preparation
- Application-specific initialization
- Application execution

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

JavaFX application startup can be broken up into the following phases.

Phase 1: Initialization

Initialization of Java Runtime and an initial examination identifies components that must be loaded and executed before starting the application.

Phase 2: Loading and preparation

The required resources are loaded from either the network or a disk cache, and validation procedures occur. All execution modes see the default or a custom preloader.

Phase 3: Application-specific initialization

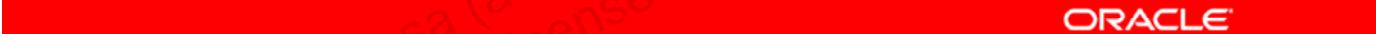
The application is started, but it may need to load additional resources or perform other lengthy preparations before it becomes fully functional. An example of this is checking whether elevated permissions are needed and displaying the appropriate request for permission to the user.

Phase 4: Application execution

The application is displayed and is ready to use.

Preloaders

- The application provides visual information that your application is loading.
- Java FX includes default preloaders:
 - Splash screens
 - Progress bars
- JavaFX preloader is customizable.
 - Can be customized with CSS

The red bar contains the word "ORACLE" in white capital letters.

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Preloaders are small applications that provide visual information that your application is loading. Typically, they provide splash screens, usually with a logo, and progress bars, indicating that your application is loading.

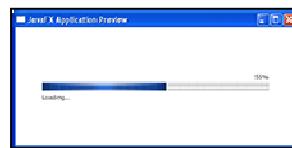
JavaFX provides a default preloader. However, you can customize the preloader with your own logos and progress bars. In addition, as with other JavaFX components, you can style the preloader with CSS.

Default Information

- Java Web Start

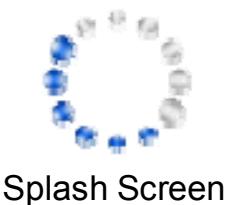


Splash Screen



Loading Dialog

- HTML



Splash Screen



Loading Dialog



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Pictured are some of the default images displayed depending upon your deployment option.

Packaging Tools

- NetBeans
 - Builds a Web Start and web browser deployment for each app
 - Totally automated
 - Digital signing
- Ant
 - For more control and customization options
 - Can create multiple JARs
 - Custom HTML and preloaders
 - Digital signing
- JavaFX Packager
 - A convenience utility
 - Does not provide as much flexibility or as many options as Ant tasks

 ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

NetBeans automatically creates deployment files for Web Start and web browser distribution. You can always start with these files and customize them for your needs.

If you need more power, Ant tasks are also provided with JavaFX, giving you much finer-grained control of your `.jar` files.

Both NetBeans and Ant can be used to digitally sign your applications.

Breaking Out of the Sandbox

How do you break out of the sandbox?

- Digitally sign each application.
- What steps are involved?
 - Use public/private key encryption.
 - Publish your public key with a known certificate authority.
 - Requires a yearly fee
 - Generate a certificate
- Covered in detail in the security lessons.

ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Quiz

Which two components are required to launch a JavaFX application from a web page?

- a. A JavaScript launch script
- b. RSA encryption
- c. A security certificate
- d. A .jnlp file



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, d

Summary

In this lesson, you should have learned how to:

- Describe a JAR file and the steps for creating one
- Create a manifest file that uses headers
- Create a JAR by using development tools
- Deploy a stand-alone JAR
- Deploy a JAR as an applet
- Deploy a JAR using Java Web Start



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Practice Overview

- Practice 13-1: Deploying an Application in a Web App
- Practice 13-2: Deploying an Embedded Application Only (Browser Only)
- Practice 13-3: Deploying Java Web Start Only



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

14

Developing Secure Applications

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe aspects of security
- Describe fundamental software security concepts
- Avoid common injection and inclusion attacks
- Avoid common confidential data errors
- Limit class accessibility, extensibility, and mutability
- Define immutability
- Create classes that are immutable
- List security resources available on the Internet

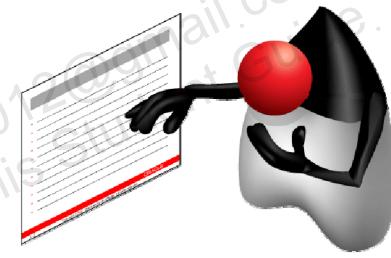


ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Topics

- Security overview
- Secure software overview
- Attacks and best practices
- Security resources



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Security Overview

The goal of computer security is to protect information from theft, corruption, or natural disaster while making it accessible for its intended purpose.

- Security is always a systemic solution, which includes:
 - Computer systems
 - Computer software
 - Human systems
- Aspects of security:
 - Confidentiality
 - Integrity
 - Availability



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The goal of computer security is to protect information from theft, corruption, or natural disaster while making it accessible for its intended purpose. Security is often a balancing act. For example, the best way to make an application completely secure from the Internet would be to not connect to the Internet at all. However, your application would then have no availability to your users.

All computer systems are made up of many parts. You have operating systems, network systems, software, and finally, people systems. Each of these has an influence on overall security.

To further define security terms, we look at three aspects of security: confidentiality, integrity, and availability.

Confidentiality

Confidentiality is the concealment of information resources.

- Most systems require some degree of secrecy.
- Mechanisms that support confidentiality:
 - Encryption
 - Passwords
 - Access control
 - Limiting resource access to a few people
 - Administrator versus user access



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Pretty much all systems today rely on some degree of secrecy. Anything from passwords to your system configuration system configuration needs protecting.

Encryption is the process of converting information into an unreadable code. With a key, information can be encrypted for safe storage, and then decrypted so the information can be read when needed.

Access control is the process of limiting access to important resources. For example, a system administrator is the only user on a system who can reset another user's password.

Integrity

Integrity means that your data is protected from unauthorized change.

- Authentication
 - A user must provide credentials.
 - Only authenticated users have access.
- Mechanisms that support integrity:
 - Prevention mechanisms
 - Blocking unauthorized users
 - Passwords
 - Certificates
 - Detection mechanisms
 - Access logs
 - Analyzing patterns



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Integrity means that your data is protected from unauthorized change. This is normally accomplished through authentication. A user must provide some sort of credential (user name and password) to indicate they are who they say they are. This controls who and how changes are made to data.

However, authentication is not enough. Detection systems must be in place in case the authentication systems fails; for example, logging the IP address of a user to identify unusual changes in behavior.

Availability

Availability is the ability to use a system or resource when desired.

- A system must be accessible by its users.
 - An offline system cannot be used.
- Denial-of-service attacks:
 - Exploit system features to negatively affect performance
 - Deny users access to the system



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A practical approach must be used when it comes to availability. If you were to design a truly secure system, you would put it in a bunker with armed guards and provide no network access. However, if network access is a requirement for your application, then there must be some compromise to make the system available to its users. Thus, the mere fact that a system is on a network makes it vulnerable to denial-of-service attacks.

Denial-of-service attacks block access to a system. They prevent users from accessing a system or using it at a normal level of performance. Systems should be flexible enough to be able to detect such attacks and respond to them.

Topics

- Security overview
- Secure software overview
- Attacks and best practices
- Security resources



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Developing Secure Software

The focus of this lesson is on developing more secure software.

- Any system can be a target.
 - Theft
 - Sabotage
 - Controlling resources
- Avoiding common mistakes.
 - Be aware of common attacks.
 - Learn from others' experience.

This is an introduction, not comprehensive.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The goal of this lesson is to introduce some of the principles needed to develop more secure software. Any system that contains confidential information is likely to be a target of attackers. Whether the attacker is attempting theft, sabotage, or just wants another zombie for his/her bot net, your software should take into account possible exploitation attempts.

This lesson is an introduction to the most common security vulnerabilities found in software. It is not a substitute for a full security course or book on the subject. Links to recommended resources are provided at the end of the lesson.

Fundamental Security Concepts: Part I

- Prefer obviously no flaws, rather than no obvious flaws.
- Design APIs to avoid security concerns.
- Avoid duplication.
- Restrict privileges.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Prefer obviously no flaws: Design and write code that does not require clever logic to see that it is safe. Avoid overly complex code.

Secure APIs: It is much easier to design code to be secure from the ground up. Trying to retrofit existing code is difficult and error prone.

Example: Making a class final prevents a malicious subclass from adding finalizers, cloning, and overriding random methods.

Duplication: Code that is duplicated tends not to be treated the same way consistently when copied. In addition, this violates the Don't Repeat Yourself (DRY) principle from Agile programming.

Restrict privileges: If the code is operating with reduced privileges, then exploitation of any flaws is likely to be thwarted. For example, leaving applets and JWS jar files unsigned means the application runs in a sandbox and will not be able to execute any potentially dangerous code.

Fundamental Security Concepts: Part II

- Establish trust boundaries.
- Minimize security checks.
- Encapsulate.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Trust boundaries: Establish boundaries between different parts of your application. For example, any data coming from a web browser to a web application should be sanitized before use.

Security checks: Perform security checks at a few defined points and return an object (a capability) that client code retains so that no further permission checks are required.

Encapsulate: Allocate behaviors and provide succinct interfaces. Fields of objects should be private and accessors avoided. The interface of a method, class, package, and module should form a coherent set of behaviors, and no more.

Topics

- Security overview
- Secure software overview
- Attacks and best practices
- Security resources



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Types of Security Threats

There are lots of potential security threats to a program you write. Here are a few categories:

- Injection and inclusion, input validation
- Resource management
 - Buffer overflows
 - Denial of service
- Confidential information
- Accessibility and extensibility
- Mutability



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

These are the various vulnerability categories as defined by the security personnel at Oracle. This lesson will try to highlight a few examples from each. However, the examples that follow are introductory and not exhaustive.

Injection and Inclusion

- An attack that causes a program to interpret data in an unexpected way and results in an unanticipated change of control
- Any data from an untrusted source must be validated.
 - Unfiltered data can lead to a number of attacks.
- Common forms of attack
 - Cross-site scripting (XSS)
 - SQL Injection
 - OS Command Injection
 - Uncontrolled format string

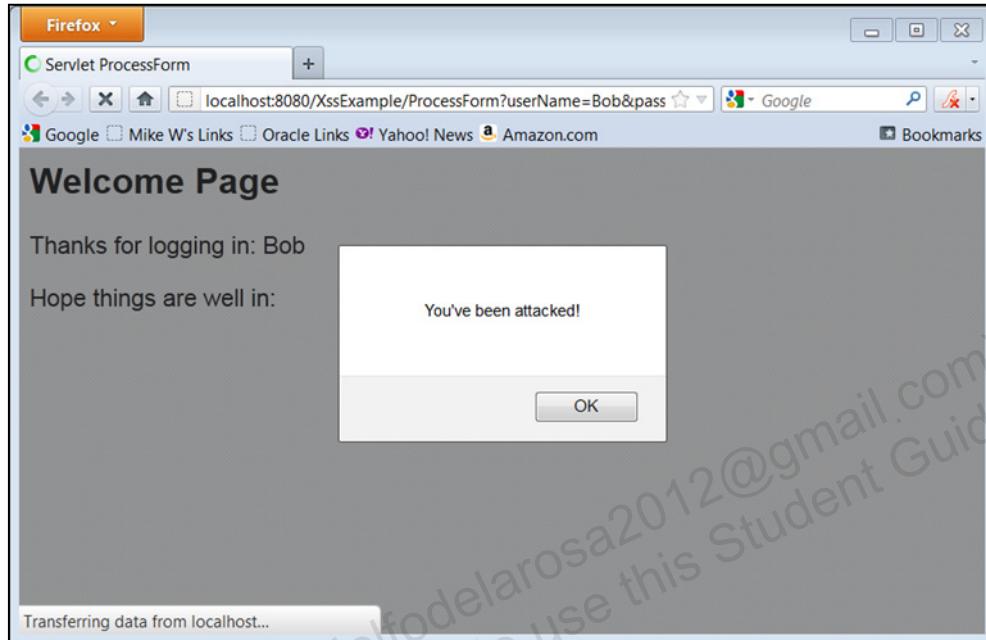


ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

XSS Example

Inserting a script into form data



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The picture shows an example of a servlet with a malicious script inserted into the input text.

Cross-Site Scripting

- Cross-site scripting (XSS) vulnerabilities occur when:
 1. Untrusted data enters a web application
 2. The web application dynamically generates a web page
 3. Exploit is sent to browsers
 4. A victim visits the generated web page
 5. The victim's web browser executes the malicious script
 6. This effectively violates the intention of the browser's same-origin policy
- A very common type of attack

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Cross-site scripting (XSS) vulnerabilities occur when:

1. Untrusted data enters a web application, typically from a web request.
2. The web application dynamically generates a web page that contains this untrusted data.
3. During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, or ActiveX.
4. Using a web browser, a victim visits the generated web page, which contains malicious script that was injected with the untrusted data.
5. Because the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.
6. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

CERT Reference: <http://cwe.mitre.org/data/definitions/79.html>

SQL Injection Example

Given the following code fragment, could anything go wrong?

```
try {
    Connection con =
DriverManager.getConnection("jdbc:derby://localhost:1527/Simple
DBDemo", "demo", "demo");
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM DEMO.Table1
        WHERE NAME=' " + nameParam + "' AND AGE ='" +
        ageParam + "'");

    while (rs.next()) {
        String s = rs.getString("Name");
        float n = rs.getFloat("Age");
        System.out.println(s + " " + n);
    }
} catch (SQLException e) {
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This is a typical piece of database code. Parameters are passed so that a search based on Name and Age can be done.

SQL Injection

An attack that relies on unfiltered data to modify SQL data results.

- The SQL statement is trying to do this:

```
SELECT * FROM DEMO.Table1 WHERE NAME=<nameParam>  
AND AGE=<ageParam>
```

- However, what if an attacker submits:

```
someValue' OR 'a' = 'a'
```

- This make the statement always true. The result is:

```
SELECT * FROM DEMO.Table1
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

By modifying the input data, attackers can access confidential information or even modify data stored in the database. Therefore, any input data should be filtered before being used.

CERT reference: <http://cwe.mitre.org/data/definitions/89.html>

OS Command Injection Example

Given the following code fragment, what could go wrong?

```
try {
    Process p = Runtime.getRuntime()
        .exec("cmd /c dir C:\\\\Users\\\\" + userName);

    BufferedReader stdInput = new BufferedReader(
        new InputStreamReader(p.getInputStream()));
    BufferedReader stdError = new BufferedReader(
        new InputStreamReader(p.getErrorStream()));

    // read the output from the command
    System.out.println("Here is your home directory:");
    while ((curLine = stdInput.readLine()) != null) {
        System.out.println(curLine);
    }
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

This piece of code attempts to read the contents of a user's directory based on data passed in by a user. Could anything go wrong with this?

OS Command Injection

An attack that relies on unfiltered data to modify an operating system command

- The example looks harmless enough: Just get a directory listing.
 - dir C:\Users\username
- However, an attacker could submit the following:
 - username; &&del *.*;
- Which would result in the loss of a lot of data



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

This is just another example of how a seemingly harmless command could be turned into something very bad. Never trust unvalidated user input.

CERT Ref: <http://cwe.mitre.org/data/definitions/78.html>

Uncontrolled Format String Example

In this code fragment, the programmer is trying to print the results when two values did not match. What could happen if a format string is submitted instead of the month?

```
static Calendar c =  
    new GregorianCalendar(1995, GregorianCalendar.MAY, 23);  
  
public static void main(String[] args) {  
    String param = "%1$tY";  
    // Example param is the credit card expiration date  
    // If param contains %1$tm, %1$te or %1$tY as malicious  
    // args  
    // Attacker and display value being compared.  
    System.out.printf(param  
        + " did not match! HINT: It was issued on %1$terd  
        of some month\n", c);  
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Even Java format strings can be exploited if code is not written properly and input is not checked.

Uncontrolled Format String

In the example, the idea was to compare two month values.

- Instead of submitting a month value, the attacker submits a format string:
 - %1\$tY
- The attacker has now figured out the year the card expires from a format string.
- Java displays the output for both format strings on the line.

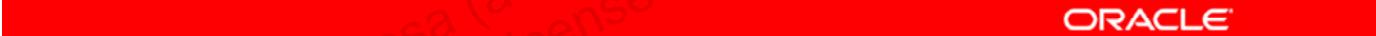


Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

CERT Ref: <https://www.securecoding.cert.org/confluence/display/java/IDS06-J.+Exclude+user+input+from+format+strings>

Resource Management

- Buffer overflows
 - Copy an input buffer to an output buffer without checking the size.
 - Result: An attacker can execute arbitrary code outside the normal program.
- Java and buffer overflows
 - Java is immune to these sort of attacks because of its automatic memory management.
- Denial of service
 - Still possible in Java
 - Relate to inappropriate use of resources



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Denial-of-Service Examples

Here are some examples of potential denial-of-service attacks:

- Zip bombs
 - Unzipping the file breaks down the system.
- Billion laughs attack
 - XML entity expansion causes the document to grow dramatically during expansion.
- XPath expressions can consume arbitrary amounts of processing time.
- Constructing object graphs



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Zip bomb: A zip file that is relatively small but contains many nested zip files within it. Unzipping the files can tie up CPUs and eat up disk space. A relatively small file can contain petabytes worth of data. Limit the size and amount of processing that can be done on a resource like this.

Billion laughs attack: If you want to use the DOM API with an XML document, you must load the entire document into memory. An attack like this can eat up available memory and bring a system to its knees.

XPath: XPath is a language for querying and traversing XML documents. Some queries have a recursive nature that can return a lot more data than expected.

Object Graph: An object graph constructed by parsing a text or binary stream may have memory requirements many times that of the original data.

Confidential Exception Example

Is there anything wrong with this exception?

```
Cannot find main configuration file:  
C:\config\badfile.properties (The system cannot find the path  
specified)  
java.io.FileNotFoundException: C:\config\badfile.properties  
(The system cannot find the path specified)  
at java.io.FileInputStream.open(Native Method)  
at java.io.FileInputStream.<init>(FileInputStream.java:138)  
at java.io.FileInputStream.<init>(FileInputStream.java:97)  
at com.example.sec.FileException.readProps  
(FileException.java:20)  
at com.example.sec.FileException.main  
(FileException.java:12)  
Java Result: -1
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Providing an exception like this gives an attacker the exact location of your application's configuration file.

Confidential Log Example

Could this log information be improved?

```
Feb 08, 2012 10:55:14 AM com.example.sec.DetailedLogger
logMessages
SEVERE: ID 123-45-6778 for User John Adams does not
match.

Feb 08, 2012 10:55:14 AM com.example.sec.DetailedLogger
logMessages
SEVERE: User John Adams located at 123 Drury Lane,
Quincy, MA

Feb 08, 2012 10:55:14 AM com.example.sec.DetailedLogger
logMessages
SEVERE: Cannot find exp date for credit card 1111-1111-
1111-1111
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Confidential Information

- Confidential data should:
 - Be readable only within a limited context
 - Not be exposed to tampering
 - Only provide users with the information that they need
 - Not be hard-coded into source
- Examples
 - Purge sensitive data from exceptions.
 - Do not log highly sensitive information.
 - Consider purging memory when done with sensitive information.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Keeping confidential information private seems like a common sense idea. However, if you do not think about security when creating your code, these sorts of leaks could easily happen. Confidential data should not be included in exceptions or log files. In addition, you should not hard-code user names and passwords in source code. Instead, a properties file should be used to store this kind of information.

Accessibility and Extensibility

Limit the accessibility of classes.

- Classes and APIs
 - Any class that is part of a public API should be declared public.
 - Any other class should be declared package-private (no modifier in front of class).
- Class fields, methods, and APIs
 - Declare fields and methods public, protected, etc. as appropriate for the API.
 - All other methods and fields should be declared private.
- Packages in .jar files should be sealed.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Package-private at the class level means that a class can interact with classes only in its own package. Thus, you can not import a Package-Private class from another package.

A jar file can be sealed by adding:

Sealed:true

to the manifest file. The heading by itself seals the entire file. You can seal specific packages by using Sealed with Name:

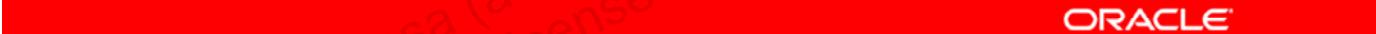
Name: com/example/sec/

Sealed:true

By sealing a .jar file, only classes contained in that .jar may be loaded by the class loader. This prevents naming conflicts when other .jar files contain packages with the same name.

Minimize Mutable Classes

- An object is considered immutable if its state cannot change after it is constructed.
 - Think of it as a read-only feature for classes.
 - Once an object is created, it cannot be changed.
 - It is good to combine this with static factory initializers.
- Provides benefits for security and concurrency:
 - Hiding constructors allows more flexibility in instance creation and caching.
 - Making classes `final` prevents classes from being extended.
 - This prevents internal data from being changed outside the class.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Immutability Example

Transforming an Employee class into an immutable employee class.

- Employee01
- Employee02
- Employee03



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Steps to Make a Class Immutable

- Set the class to final.
- Use a static initializer.
- Make copies of any mutable objects passed to an initializer.
- Set the fields to final.
- Return copies of any mutable objects.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Quiz

Most security vulnerabilities are a result of:

- a. Not validating/filtering input
- b. Poor encryption
- c. Slow CPUs
- d. Buffer overflows

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Quiz

Which is true about a final field that is a mutable object?

- a. Once an object is assigned, its values are immutable.
- b. Once an object is assigned, its values are mutable.
- c. Once an object is assigned, some of its values are mutable.
- d. You cannot mark a mutable object field as final.

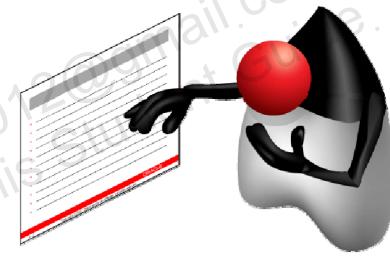


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Topics

- Security overview
- Secure software overview
- Attacks and best practices
- Security resources



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Security Resources on the Net

- CERT
 - Computer Emergency Response Team
 - Cert reports
 - <https://www.securecoding.cert.org/confluence/display/java/The+CERT+Oracle+Secure+Coding+Standard+for+Java>
- Secure Coding Guidelines for Java
 - <http://www.oracle.com/technetwork/java/seccodeguide-139067.html#conclusion>
- Books
 - Gary McGraw. *Software Security: Building Security In.* Boston: Addison-Wesley, 2006.
 - Joshua Bloch. *Effective Java Programming Language Guide.* 2nd ed. Addison-Wesley Professional, 2008.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The slide lists a few resources that you can use to learn more about security.

Summary

In this lesson, you should have learned how to:

- Describe the aspects of security
- Describe fundamental software security concepts
- Avoid common injection and inclusion attacks
- Avoid common confidential data errors
- Limit class accessibility, extensibility, and mutability
- Define immutability
- Create classes that are immutable
- List security resources available on the Internet



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Practice 14: Overview

There are no practices for this lesson.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Adolfo De+la+Rosa (adolfodelarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.

15 **Signing an Application and Authentication**

ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe cryptography, encryption, and cipher
- Differentiate a symmetric key from an asymmetric key
- Describe public/private key encryption
- Describe digital certificates
- Describe hash algorithms
- Sign an application by using Java tools
- Describe how SSL works
- Explain the security concepts that are applied in BrokerTool



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Topics

- Key security concepts
- Applying encryption technologies
- Code signing
- SSL
- Explain the security concepts that are applied in BrokerTool



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Key Security Concepts

- Cryptography (from Latin *cryptographia*, “hidden writing”)
 - The practice and study of securing communications between two parties
 - Dates back as early as Julius Caesar
- Encryption
 - Process of converting plain text into ciphertext (unintelligible text)
- Cipher
 - The algorithm used to encrypt and decrypt the text
- Key
 - A piece of information that determines the output of the cipher



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Cryptography is the practice and study of securing communication between two parties so that third parties cannot read the communication.

Caesar Cipher

- Convert plain text message
 - Hi, how are you?
- Substitution cipher
 - ABCDEFGHIJKLMNOPQRSTUVWXYZ
 - DEFGHIJKLMNOPQRSTUVWXYZABC
- Encrypted message
 - Kl, krz duh bry?



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Cryptography was known to be used by Julius Caesar to send messages of military significance. He used a simple substitution cipher to encrypt his messages.

Types of Encryption

- Symmetric key encryption
 - The same key is used to encrypt and decrypt data.
- Asymmetric key encryption
 - Separate keys can be used to encrypt and decrypt the data.
- Public key encryption
 - An asymmetric key encryption system
 - A private key encrypts the data that can be decrypted with a public key.
 - Anyone with the public key can read the message.
 - A public key is used to encrypt data that can be decrypted by the private key.
 - Used for digital signatures and code signing



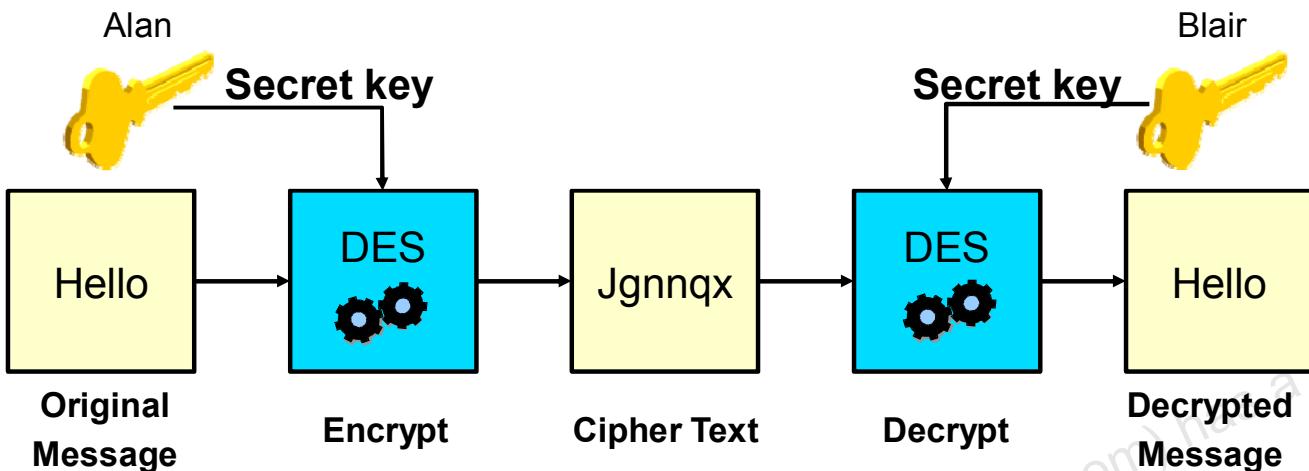
Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Symmetric key encryption is a faster, more efficient way to encrypt and decrypt data. However, given how powerful computers have become today, it is not very secure and can be defeated using a brute force attack (that is, by trying every possible combinations of letters).

Public key encryption uses two keys to encrypt and decrypt data. One key, called the private key, is encrypted using a password known only to the owner and is never transmitted by any means. The other key, called the public key, is provided freely to anyone with whom the key pair owner wishes to communicate.

Any message encrypted by the private key can be decrypted by the public key, and vice versa. If the private key is lost, then any message encrypted using the public key is lost as well.

Symmetric Key Encryption



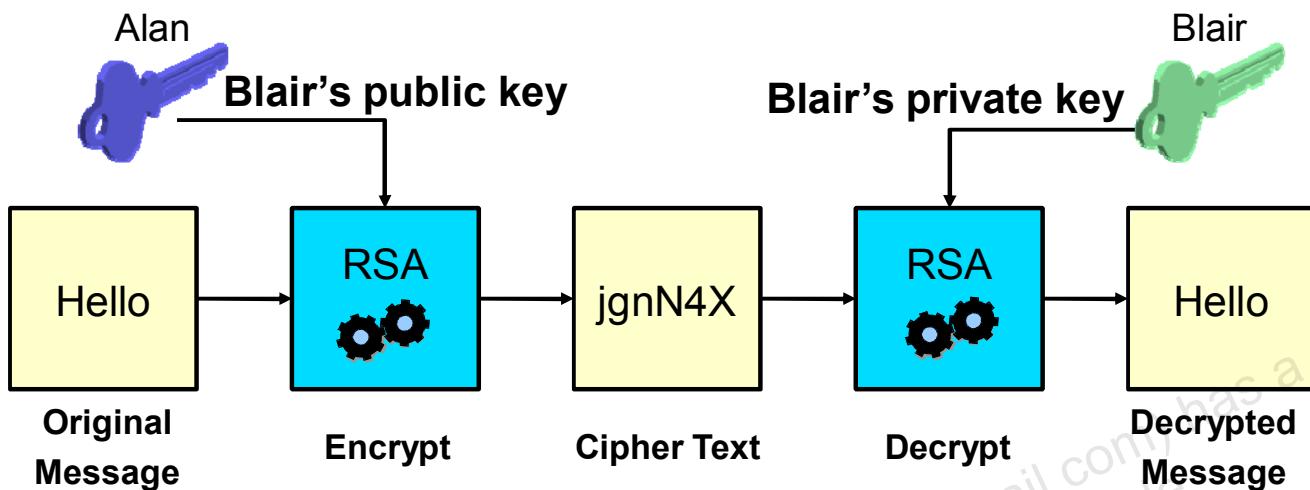
The same key is used for encryption and decryption.

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

In the example, data is encrypted using the DES cipher. Data Encryption Standard (DES) was a symmetric key encryption standard developed by Horst Feistel at IBM around 1974. However, eventually it became evident that the 56-bit key used for DES was insufficient and the cipher was replaced with other techniques like Triple DES.

Public Key Encryption



The receiver's public key is used for encryption and the receiver's private key is used for decryption.

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This example shows how a message is sent from Alan to Blair. But how do we know that we really have Blair's public key?

Quiz

A message encrypted with a private key can be decrypted with:

- a. A symmetric key
- b. Both the public and private key
- c. A private key
- d. A public key

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Topics

- Key security concepts
- Applying encryption technologies
- Code signing
- SSL
- Explain the security concepts that are applied in BrokerTool



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Applying Encryption Technologies

Given these great tools, what can we do with them?

- Code signing
 - Verifying that software comes from you
 - Lets customer knows that software is safe
- SSL/TLS
 - Secure information between client and server
 - Once again provides trust to customers



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

So we have all these wonderful encryption tools, but what can we use them for?

First, we can use the technologies to digitally sign software. This allows customers to trust that the software came from you and is safe to use. The steps for signing software is covered next.

Finally, when using a web service, it is important that we secure communications point to point. This can be done using Secure Socket Layer (SSL), also called Transport Layer Security (TLS). The protocol uses a combination of techniques to keep your data private.

Digital Certificates

- Digital ID
- Proves your identity
- Typically includes
 - Owner's name
 - Owner's public key
 - Public key expiration date
 - Certificate authority name
 - Serial number
 - Certificate authority digital signature



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Digital certificates are the electronic counterparts to driver licenses, passports, and other forms of identification. You present your digital certificate electronically to prove your identity, or your right to access information or services online.

Digital certificates bind an entity's identity to a pair of digital keys that can encrypt and sign information. When used with encryption, digital certificates provide a more complete security solution, assuring the identity of all parties involved in electronic transactions.

Certificate Sample



Certificate:
Data:
Version: v3 (0x2)
Serial Number: 5 (0x5)
Signature Algorithm: PKCS #1 MD5 With RSA Encryption
Issuer: CN=example rootCA,OU=Certification Authority,O=example.com, C=US
Validity:
Not Before: Wed Nov 26 10:46:34 1997
Not After: Mon May 25 11:46:34 1998
Subject: E=scarter@example.com,CN=Sam Carter,UID=scarter,O=example.com, C=US
Subject Public Key Info:
Algorithm: PKCS #1 RSA Encryption
Public Key:
Modulus:
00:bd:83:73:ee:26:7c:6a:3d:be:0f:de:...
Public Exponent: 65537 (0x10001)
Extensions:
Identifier: Certificate Type
Critical: no
Certified Usage:
SSL Client
Secure E-mail
Identifier: Authority Key Identifier
Critical: no
Key Identifier:
ca:57:0c:29:d0:2b:31:b9:93:a5:f9:...
Signature:
Algorithm: PKCS #1 MD5 With RSA Encryption
Signature:
06:1d:fa:ae:3b:bf:1d:a0:63:a0:1b:24:f5:5b:...

A certificate contains a public key and information about its owner, with a certificate authority's guarantee about the accuracy of the information.

How does Alan know that the certificate was not altered?

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Certificates used in Java code signing follow the X.509 standard. Pictured is some of the sample information contained in the certificate. But how do we know that the information has not been altered?

Sample Certificate Information:

Certificate:

Data:

Version: v3 (0x2)

Serial Number: 5 (0x5)

Signature Algorithm: PKCS #1 MD5 With RSA Encryption

Issuer: CN=example rootCA,OU=Certification

Authority,O=example.com, C=US

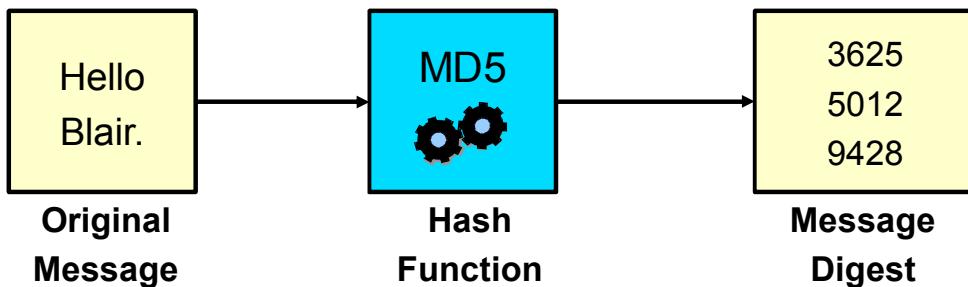
Validity:

Not Before: Wed Nov 26 10:46:34 1997

Not After: Mon May 25 11:46:34 1998

Subject: E=scarter@example.com,CN=Sam

Message Digests/Hashes



Message digests are unique hash values such that:

- Two documents are not likely to hash to the same value
- Given a hash, you cannot determine what the original message was (one-way hash)

How do I know that the message digest was not altered?

ORACLE

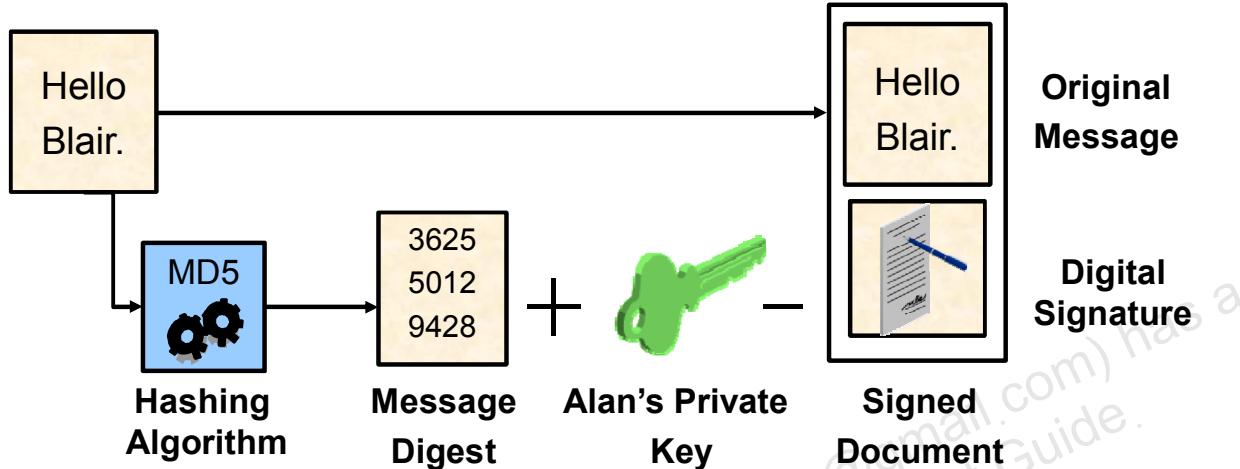
Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Message integrity is achieved by calculating a numerical value, called a message digest, which provides the original document's digital fingerprint. A message digest is created using a hash function that derives a unique numerical value from message text.

So, if you could compute a message digest before a message is sent and if you computed another message digest after the message is received, then if the BEFORE digest and the AFTER digest are the same, you can be sure the message has not been changed.

Digital Signatures I

Alan sends a message to Blair.



The message digest encrypted with the sender's private key is the digital signature.

ORACLE

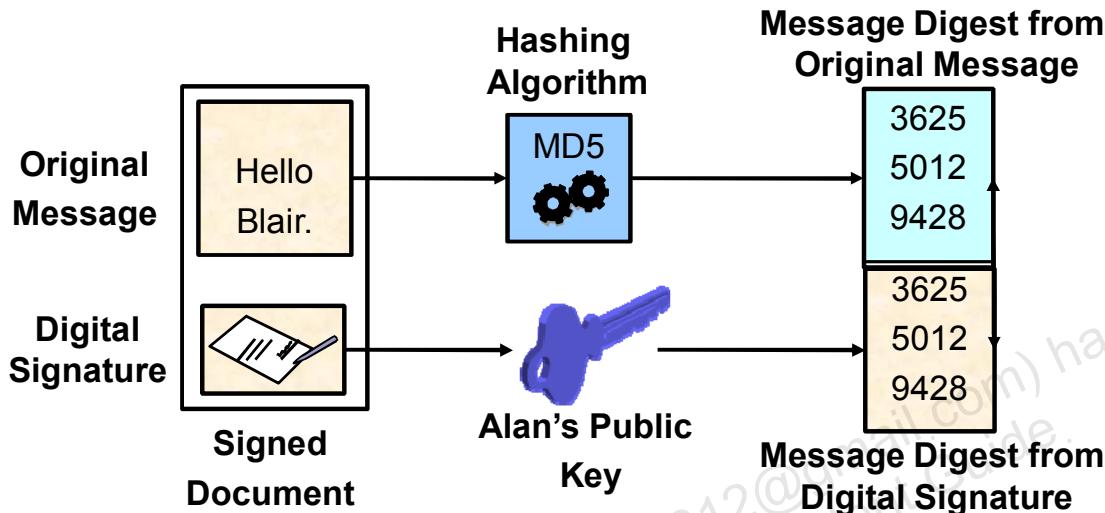
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A message digest provides the original document's digital fingerprint. When the message digest is encrypted using the sender's private key, it becomes a digital signature, which can be appended to an encrypted or plain-text message to guarantee the message's source and integrity.

A digital signature, in theory, is of greater validity than a physical signature on paper. If a ten-page paper document is signed on the last page, there is no guarantee that one of the previous pages was not altered. In contrast, any change to a digitally signed document changes the message digest.

Digital Signatures 2

Blair receives the message.



The recipient hashes the message and compares the resultant message digest with the one from the digital signature.

Whose certificate should I trust?

ORACLE

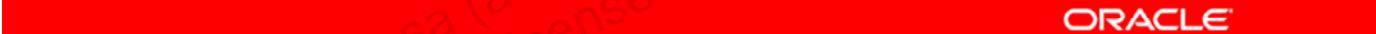
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To verify the data's integrity, the receiving software first uses the sender's public key to decrypt the message digest. It then uses the same hashing algorithm that generated the original hash to generate a new one-way hash of the same data. (Information about the hashing algorithm used is sent with the digital signature, although this information is not shown in this slide.)

Finally, the receiving software compares the new hash against the original hash. If the two hashes match, the data has not changed since it was signed. If they do not match, the data might have been tampered with because it was signed, or the signature might have been created with a private key that does not correspond to the public key presented by the signer.

Certificate Authority

- A trusted entity that issues and manages certificates
- Three ways to handle certificates:
 - External: CAs sell individual certificates
 - Outsourced: CAs manage certificate issuance and revocation
 - Internal: Use a certificate server



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

An X.509 certificate is issued by a third party called a certificate authority (CA), which takes responsibility for identifying a specific public key's owner. The method used by the CA to establish the owner's identity is the most important part of CA security policy. The effectiveness of Internet security depends on the care taken when issuing certificates.

Quiz

Which technology can you use to ensure that a message has not been tampered with?

- a. A digital signature
- b. A certificate
- c. A symmetric key
- d. Compression

 ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Topics

- Key security concepts
- Applying encryption technologies
- **Code signing**
- SSL
- Explain the security concepts that are applied in BrokerTool

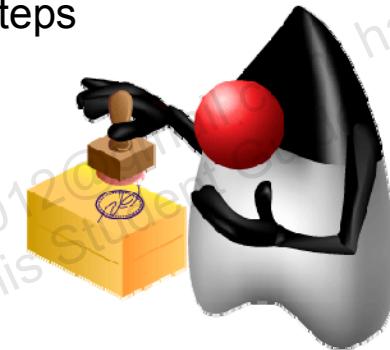


ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Code Signing with a CA

- For an application or applet distributed over the Internet
- Must purchase certificate from the CA
- Example Certificate Authorities
 - VeriSign
 - Thawte
 - Go Daddy
- Check with the vendor for specific steps



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

If you have a public application available over the Internet that you wish to have signed, you must purchase a certificate from a certificate authority. Each vendor has its own requirements, but the steps should be pretty similar.

Note: The companies listed are examples of vendors who offer these services. It is in no way an endorsement by Oracle.

Steps for Getting a Certificate

To get a certificate from a CA, you must perform the following steps:

- Create a keystore.
 - This will create a public and private key.
 - Use keytool, which comes with Java.
- Generate a certificate signing request.
 - Create a request form using your public key.
- Go to the vendor's website to purchase and submit the request.
- The vendor sends you a certificate.
- Store the certificate in your keystore.
 - Now you can use the certificate to sign your applications.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Once you have done all the steps outlined in the slide, you are ready to go. When you sign a .jar, your private key is used. The certificate is included with the .jar so that a user can verify that the file comes from you and has not been tampered with.

Generating Public and Private Keys with keytool

- **keytool**
 - It is a key and certificate management utility.
 - Users can create public/private key pairs.
 - Users can create certificates for use in self-authentication.
 - It allows users to cache the public keys (in the form of certificates).
- **Sample keytool command:**
 - `keytool -genkey -alias signFiles -keystore examplestore`

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

`keytool -genkey -alias signFiles -keystore examplestore`

What do each of the keytool subparts mean?

-genkey: Generates the keys

-alias signFiles: The alias used to refer to the keystore entry containing the keys that are generated

-keystore examplestore: The name (and optionally path) of the keystore that you are creating

You will also be prompted for a password for the keystore (the storepass) and for the private key (keypass). You can optionally make them the same password.

Additional keytool Information

In addition to the storepass and keypass values, you will be prompted for distinguished name information such as the following:

```
What is your first and last name?  
[Unknown]: John Adams  
What is the name of your organizational unit?  
[Unknown]: Purchasing  
What is the name of your organization?  
[Unknown]: ExampleCompany  
What is the name of your City or Locality?  
[Unknown]: Mountain View  
What is the name of your State or Province?  
[Unknown]: CA  
What is the two-letter country code for this unit?  
[Unknown]: US  
Is <CN=John Adams, OU=Purchasing, O=ExampleCompany,  
L=Mountain View, ST=CA, C=US> correct?  
[no]: y
```

ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

In addition to the passwords, you are prompted for additional information to uniquely identify this keystore.

Sign a jar

- Jarsigner
 - Is a command line tool
 - Can sign or verify a jar file
 - Creates a new jar and does not change the source jar
- jarsigner -keystore examplestore -signedjar sCount.jar Count.jar signFiles
- NetBeans
 - Project properties
 - Build > Deployment > Request Unrestricted Access

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To create a self-signed jar file, you can use the jarsigner command line tool. In addition, NetBeans includes support for creating self-signed .jar files.

jarsigner -keystore examplestore -signedjar sCount.jar Count.jar signFiles

-keystore: Specifies the keystore file

-signedjar: Specifies the jar that will be created from the command (sCount.jar)

The .jar file and alias for the keystore are the last two parameters in the command.

Quiz

Which tool can be used to create a keystore?

- a. Keystoretool
- b. keytool
- c. jarsigner
- d. keystore

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Topics

- Key security concepts
- Applying encryption technologies
- Code signing
- SSL
- Explain the security concepts that are applied in BrokerTool



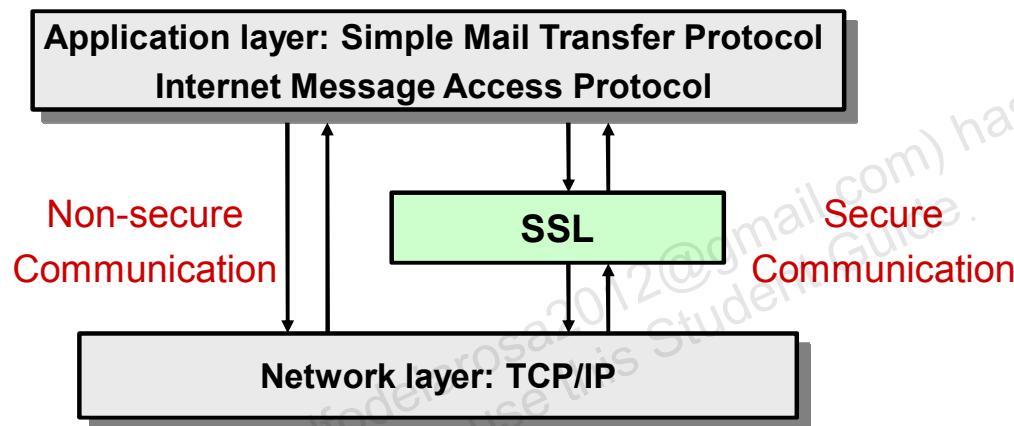
ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

SSL

Secure Socket Layer (SSL)

- Ensures safe and secure client-server transactions
- Data sent over the network is point-to-point encrypted.
- Also called Transport Layer Security (TLS)



ORACLE

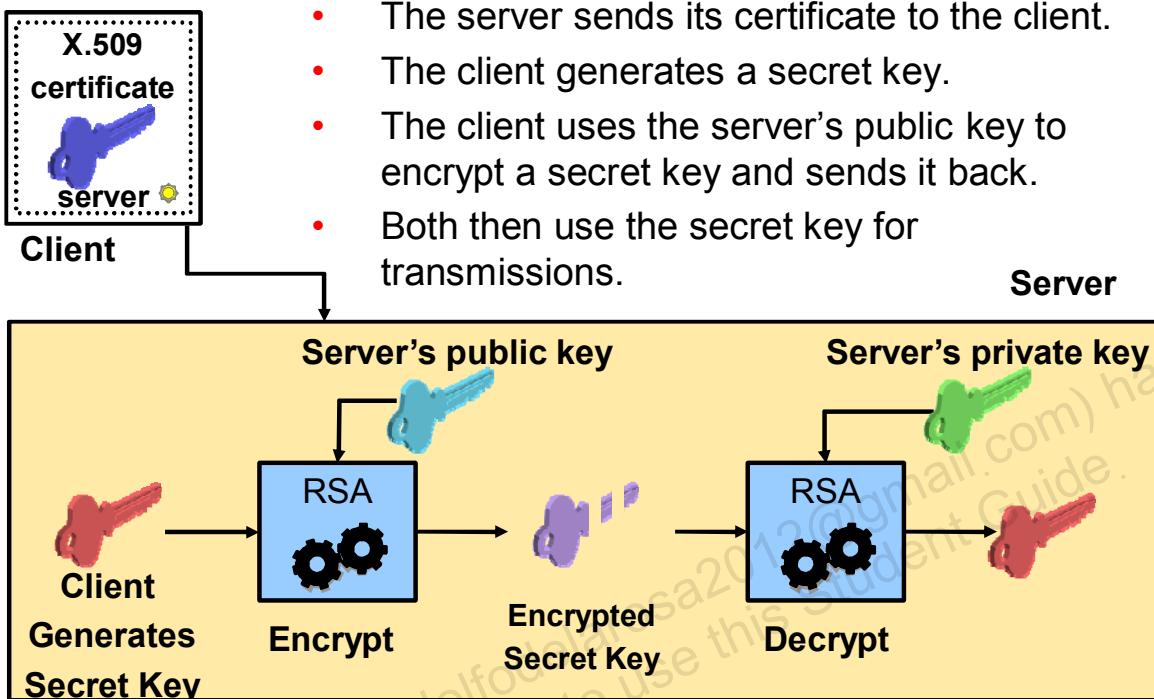
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To secure our web service connection, we would use secure sockets layer (SSL). SSL provides the following:

- **Authentication:** Using X.509 certificates and digital signatures encrypted using RSA public-key technology
- **Message privacy:** Using DES, RC2, or RC4 symmetric encryption
- **Message integrity:** Using MD5 or SHA-1 message digests

SSL forms an application-independent layer between TCP/IP and the network application layer. As a result, SSL can be used to provide secure communications for different network applications. For HTTP connections, Hypertext Transport Protocol, Secure (HTTPS) secures the connections between clients and servers.

Generating Secure Connections



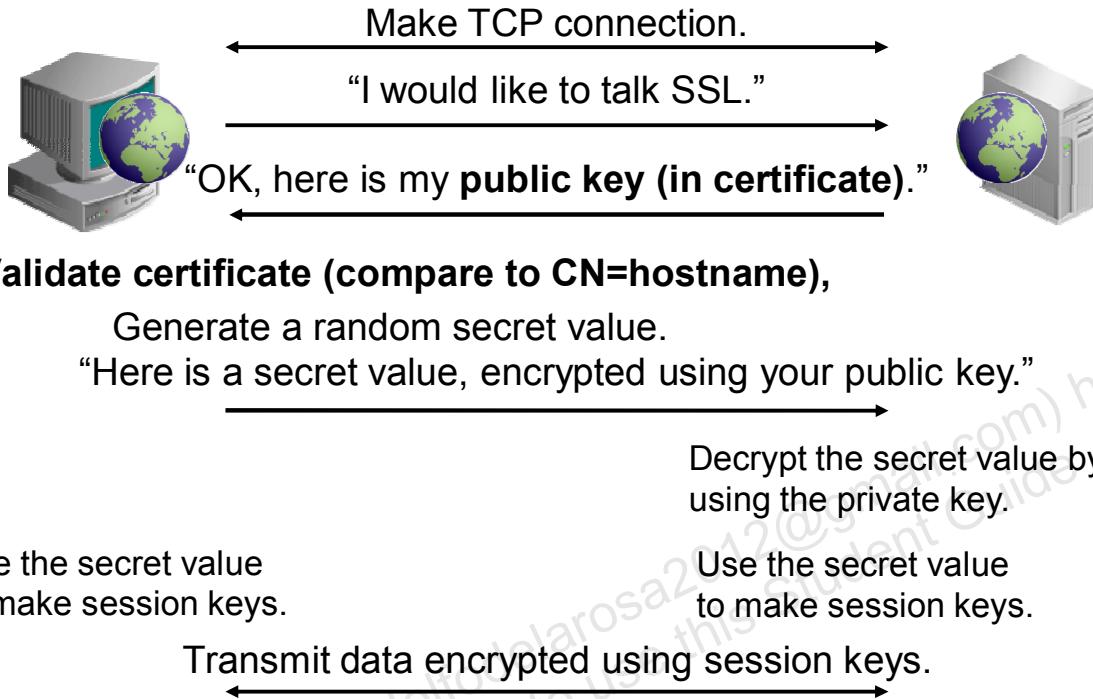
ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

By combining symmetric and asymmetric-key encryption and digital certificates, you can exchange symmetric keys. A secure connection is generated when using the following steps:

1. A client generates a temporary symmetric key. This is sometimes called the session key because it expires after communication between the client and the server is complete. It is used only for one session.
2. The client encrypts the temporary symmetric key with a server's public key. This means that only the server can decrypt this information.
3. The encrypted symmetric key is sent to the server in much the same way that an encrypted message is sent.
4. The server then decrypts the secret key by using its own private key, so now both the client and the server can use the newly generated symmetric key.
5. During the rest of the session, encrypted communication takes place using the faster, more efficient symmetric key.
6. When the session is over, the symmetric key is discarded, because it no longer has any use to either system.

SSL Server Authentication



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The following summarizes the steps in the SSL process when only the server is authenticated.

1. **Client hello:** The client sends a message that begins the handshake portion of the SSL protocol. The client tells the server which encryption algorithms it recognizes.
2. **Server hello:** The server responds with an X.509 certificate signed with its private key and tells the client which encryption algorithm is the strongest form of encryption that both of them recognize.
3. **Client key exchange:** The client calculates a master secret, encrypts it with the server's public key, and sends it to the server.
4. **Change cipher spec:** The client and the server create session keys from the master key and begin communicating securely.
5. **HTTPS request:** The client sends the HTTPS request, encrypted with the session key.
6. **HTTPS response:** The server responds, encrypting its response with the session key, and notifies the client that it is done.

Quiz

What is the purpose of SSL?

- a. To sign documents
- b. To create a secure connection between a client and a server
- c. To create unbreakable certificates
- d. To decrypt any document that has been encrypted using a public key



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Topics

- Key security concepts
- Applying encryption technologies
- Code signing
- SSL
- Explain the security concepts that are applied in BrokerTool



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Java EE Application Security Mechanisms

- Security of Java EE components are provided by their containers.
- A container provides two kinds of security:
 - Declarative
 - Programmatic
- Declarative security uses either deployment descriptors or annotations.
- Programmatic security is embedded in an application and is used to make security decisions.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Enterprise-tier and web-tier applications are made up of components that are deployed into various containers.

These components are combined to build a multitier enterprise application.

A deployment descriptor is an XML file that contains information about security roles, access control, and authentication requirements. NetBeans IDE provides tools for creating and modifying deployment descriptors.

Web components may use a web application deployment descriptor named `web.xml`.

Enterprise JavaBeans components may use an EJB deployment descriptor named `META-INF/ejb-jar.xml`, contained in the EJB JAR file.

Annotations, also called metadata, are used to specify information about security within a class file. When the application is deployed, this information can be either used by or overridden by the application deployment descriptor. Annotations save you from having to write declarative information inside XML descriptors. Instead, you simply put annotations on the code, and the required information gets generated.

Programmatic security is useful when declarative security alone is not sufficient to express the security model of an application.

Securing Web Applications

By using annotations or deployment descriptors, you can secure a web application by:

- Specifying security constraints
- Specifying authentication mechanisms
 - HTTP basic
 - Form-based
 - Digest
 - Client
 - Mutual
- Declaring security roles



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

A *security constraint* is used to define the access privileges to a collection of resources by using its URL mapping. Its sub-elements are:

- Web resource collection
- Authorization constraint
- User data constraint

A user-authentication mechanism specifies:

- The way a user gains access to web content
- With basic authentication, the realm in which the user will be authenticated
- With form-based authentication, additional attributes

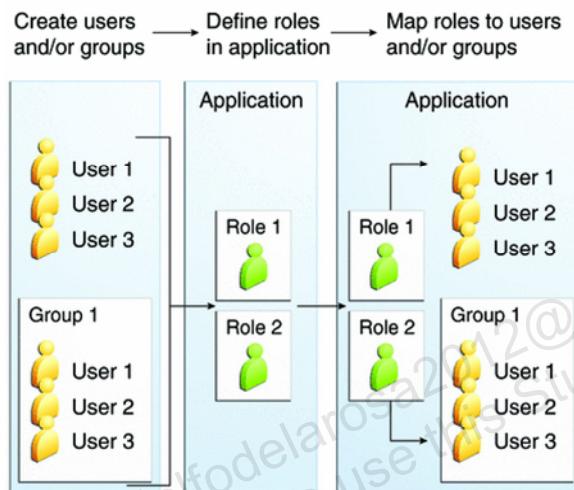
When an authentication mechanism is specified, the user must be authenticated before access is granted to any resource that is constrained by a security constraint. There can be multiple security constraints applying to multiple resources, but the same authentication method will apply to all constrained resources in an application.

Before you can authenticate a user, you must have a database of user names, passwords, and roles configured on your web or application server.

You can declare security role names used in web applications by using the security-role element of the deployment descriptor. Use this element to list all the security roles that you have referenced in your application.

Applying Security in GlassFish Server

- GlassFish Server supports the Java EE 6 security model.
- You can configure GlassFish Server for adding, deleting, or modifying authorized users and existing or custom realms.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

A realm is a security policy domain defined for a web or application server. A realm contains a collection of users, who may or may not be assigned to a group.

The figure in the slide shows how mapping roles to users and groups are done in GlassFish.

HTTP Basic Authentication Mechanism



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

A user authentication mechanism specifies the way a user gains access to web content.

With basic authentication:

1. A client requests access to a protected resource.
2. The web server returns a dialog box that requests the user name and password.
3. The client submits the user name and password to the server.
4. The server authenticates the user in the specified realm and, if successful, returns the requested resource.

Specifying HTTP basic authentication requires that the server request a user name and password from the web client and verify that the user name and password are valid by comparing them against a database of authorized users in the specified or default realm.

Authentication in an Application

You protect resources to ensure that only authorized users have access.

To authenticate a user, you need to perform the following basic steps:

1. The application developer writes code to prompt for a user name and password.
2. The application developer communicates how to set up security for the deployed application by use of a metadata annotation or deployment descriptor.
3. The server administrator sets up authorized users and groups on the GlassFish Server.
4. The application deployer maps the application's security roles to users, groups, and principals defined on the GlassFish Server.



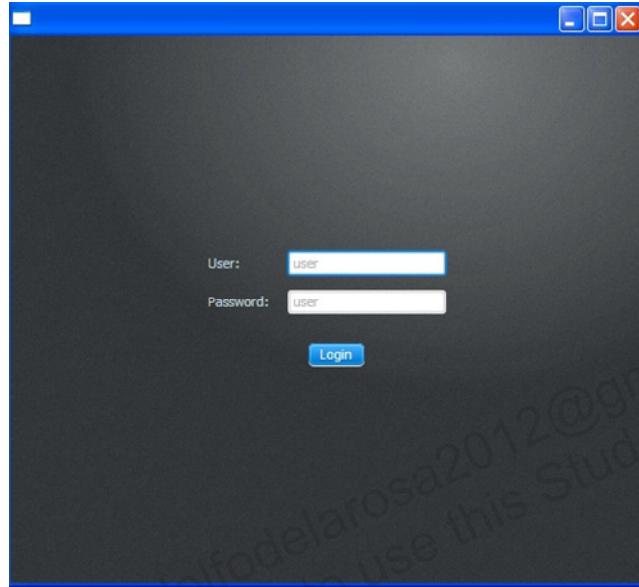
Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

An application will often prompt for a user name and password before allowing access to a protected resource.

After the user name and password have been entered, that information is passed to the server, which either authenticates the user and sends the protected resource or does not authenticate the user, in which case access to the protected resource is denied.

Authentication in the BrokerTool Application

Login.fxml of BrokerToolClient3 prompts for a user name and password.



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Login screen that will prompt for a user name and password.

Deployment Descriptor of BrokerToolServer

```
<security-constraint>
    <display-name>UserConstraint</display-name>
    <web-resource-collection>
        <web-resource-name>User</web-resource-name>
        <description/>
        <url-
pattern>/resources/com.brokertool.model.shares/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <description/>
        <role-name>UserRole</role-name>
    </auth-constraint>
</security-constraint>
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>file</realm-name>
    .....
    .....

```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

View the web.xml configuration file of the BrokerToolServer project where you can specify authentication mechanism.

The code snippet in the slide displays the security constraints declared in web.xml deployment descriptor file.

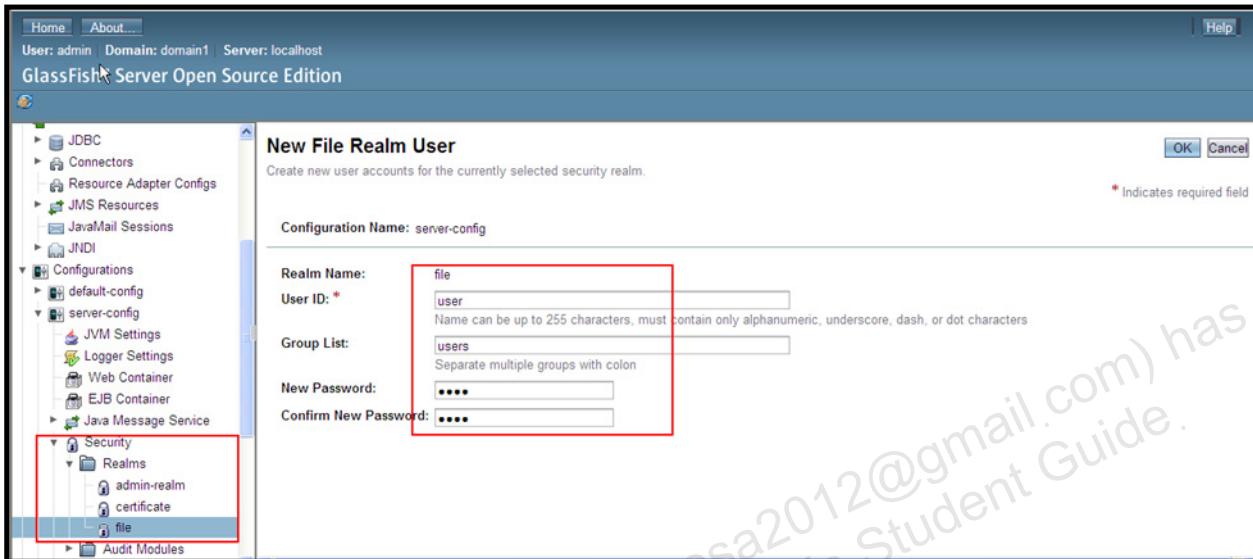
The auth-constraint element specifies which of the roles is authorized to access protected resources.

To specify an authentication mechanism, use the login-config element. It can contain the following sub-elements:

- The auth-method sub-element configures the authentication mechanism for the web application. The element content must be either NONE, BASIC, DIGEST, FORM, or CLIENT-CERT.
- The realm-name sub-element indicates the realm name to use when the basic authentication scheme is chosen for the web application.

You can declare security role names used in web applications by using the security-role element of the deployment descriptor. Use this element to list all the security roles that you have referenced in your application.

Set Up Users and Groups on the GlassFish Server



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

BrokerToolClient3 includes basic authentication for "user"/"user"; therefore, in order to run this client with BrokerToolServer, your GlassFish server must be configured to include this specific user.

1. Open the GlassFish Admin Console in NetBeans.
 - a. In the Services tab, open Services > GlassFish Server 3.1.1. Right-click and select View Domain Admin Console.
 2. In the Common Tasks tree, open Configurations > server-config > Security > Realms > file. Select "file". Edit Realm appears.
 3. Select Manage Users on the Edit Realm page.
 4. Add one user:
User ID = user
Group List = Users
Password = user
- The above is probably all you need, but another user has been added anyway:
- User ID = admin
Group List = Users
Password = admin

Deployment Descriptor: glassfish-web.xml

You will use `glassfish-web.xml` to map the application's security roles to users, groups, and principals defined on the GlassFish Server.

```
<glassfish-web-app error-url="">
    <security-role-mapping>
        <role-name>AdminRole</role-name>
            <principal-name>admin</principal-name>
        </security-role-mapping>
        <security-role-mapping>
            <role-name>UserRole</role-name>
            <principal-name>user</principal-name>
        </security-role-mapping>
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The role-name must match the role-name in the security-role element of the corresponding deployment descriptor `web.xml`.

Principal: An entity that can be authenticated by an authentication protocol in a security service that is deployed in an enterprise. A principal is identified by using a principal name and authenticated by using authentication data.

Security policy domain, also known as **security domain** or **realm**: A scope over which a common security policy is defined and enforced by the security administrator of the security service

Programmatic Security

```
@Override  
protected List<Customer> call() throws Exception {  
    Client client = Client.create(CONFIG);  
    client.addFilter(new  
        HTTPBasicAuthFilter(BrokerToolClientApp.getInstance().getUs  
er(), BrokerToolClientApp.getInstance().getPassword()));  
    WebResource productsWebResource =  
        client.resource("http://localhost:8080/BrokerToolServer/res  
ources").path("com.brokertool.model.shares");  
    ClientResponse response =  
        productsWebResource.get(ClientResponse.class);  
    int status = response.getStatus(); //status =  
    401 if login error  
    if (status == 401) {  
        throw new AuthenticationException();  
    }  
    return null;  
}
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The code from `login.java` of the `BrokerToolClient3` project demonstrates the use of programmatic security using Jersey API (`com.sun.jersey.api.client.filter.HTTPBasicAuthFilter`).

The code checks whether you are trying to access a web service resource without providing the authorized user credentials. If the user name and password is correct, the next screen appears; otherwise, the login screen appears.

This code snippet does the following:

- It retrieves the user name and password as provided by the user in the login screen.
- The resource `("http://localhost:8080/BrokerToolServer/resources").path("com.brokertool.model.shares")` is accessed.
- The status of `ClientResponse` determines whether the user has been authenticated.
- If its value is 401, an authentication exception is thrown.

Quiz

HTTP basic is an:

- a. Authentication mechanism
- b. Authority mechanism
- c. Agreement mechanism
- d. Amazing mechanism

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Describe cryptography, encryption, and cipher
- Differentiate a symmetric key from an asymmetric key
- Describe public/private key encryption
- Describe digital certificates
- Describe hash algorithms
- Sign an application by using Java tools
- Describe how SSL works
- Explain the security concepts that are applied in BrokerTool



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Practice 15: Overview:

- Practice 15-1: Signing an Application by Using keytool and jarsigner
- Practice 15-2: Signing and Deploying TextViewer by Using NetBeans



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

16

Logging

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Add a logging mechanism to your Java application
- Log messages to a file at an appropriate level
- Configure your logger
- Configure logging formatters and handlers



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Topics

- Logging API overview
- Logging configuration
- Logging examples
- Configuration files and custom handlers



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Logging Overview

For software applications, logging systems provide end users, system administrators, and developers with reports for troubleshooting and maintaining their systems. Logging systems are typically used for:

- Monitoring security
- Providing configuration information
- Reporting on errors or problems with a system
- Troubleshooting systems



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

When considering whether to use logging in your application, consider this statement from Brian W. Kernighan and Rob Pike from the book *The Practice of Programming*:

As personal choice, we tend not to use debuggers beyond getting a stack trace or the value of a variable or two. One reason for this is that it is easy to get lost in the details of complicated data structures and control flow; we find stepping through a program less productive than thinking harder and adding output statements and self-checking code at critical places. Clicking over statements takes longer than scanning the output of judiciously-placed displays. It takes less time to decide where to put print statements than to single-step to the critical section of code, even assuming that we know where that is. More important, debugging statements stay with the program; debugging sessions are transient.

Reference: <http://logging.apache.org/log4j/1.2/manual.html>

Using the Java Logging API

To use the Java Logging API you need to:

- Create a logger object
 - Logging objects are found in the `java.util.logging` package
- Configure the logger object
- Send messages to the logger object

```
Logger logger = Logger.getLogger("com.example.ClassName");
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Java Logging API classes can be found in the `java.util.logging` package.

Topics

- Logging API overview
- Logging configuration
- Logging examples
- Configuration files and custom handlers



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Configure the Logger Handler

The logging API is contained in the `java.util.logging` package. The core class in the API is the **Logger** class. Setting the level for a Logger determines what messages are logged. A Logger object is associated with one or more handlers. The handler is responsible for publishing the log message.

Predefined handlers include:

- StreamHandler
- ConsoleHandler
- FileHandler
- SocketHandler
- MemoryHandler



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A logger object is associated with one or more handlers. The handler is responsible for the publication of the log message. This responsibility includes specifying both the format and the output location. Examples of an output location are a console and a file.

The logging API provides the following handlers:

- StreamHandler: A simple handler for writing formatted records to an `OutputStream` object
- ConsoleHandler: A simple handler for writing formatted records to the `System.err` stream
- FileHandler: A handler that writes formatted log records either to a single file or to a set of rotating log files
- SocketHandler: A handler that writes formatted log records to remote TCP ports
- MemoryHandler: A handler that buffers log records in memory

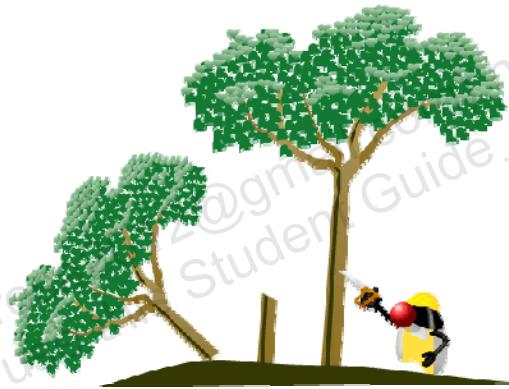
Note: The default handler associated with the logger instance is the `StreamHandler`.

Configure the Logger Formatter

Once a handler is assigned, set the Formatter that determines the format of the log file. There are two predefined formatters:

- SimpleFormatter
- XMLFormatter

The XMLFormatter is the default in Java 7.



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Java includes two standard Formatters:

SimpleFormatter: Writes brief “human-readable” summaries of log records.

XMLFormatter: Writes detailed XML-structured information.

As with Handlers, it is fairly straightforward to develop new Formatters.

Configure the Logger Level

Logging output is controlled by using levels to define the importance of messages.

- The predefined levels, from highest to lowest, are:
 - SEVERE
 - WARNING
 - INFO
 - CONFIG
 - FINE
 - FINER
 - FINEST
- Two special levels of ALL or OFF are also available.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

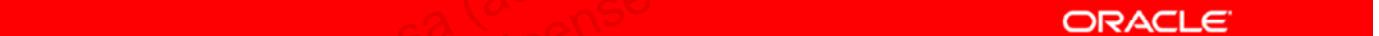
The level information is used in two ways: Firstly for recording logging error messages. SEVERE error would be the most serious, and probably should always be logged. FINEST would be the least important and probably should not be logged.

Setting the log lever for a logger. For example, `logger.setLevel(Level.INFO)` ; would log any messages that had their severity set to INFO, WARNING, or SEVERE.

Quiz

Setting the logging level:

- a. Determines the location and the name of the log file
- b. Has no effect on the log file
- c. Provides military grade security and encryption to your application, making it much tougher to hack
- d. Determines which messages are logged and, therefore, the amount of detail in the log

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Topics

- Logging API overview
- Logging configuration
- Logging examples
- Configuration files and custom handlers



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Logger Example: main()

```
public static void main(String[] args) {  
    BasicLogging bl = new BasicLogging();  
    bl.logger = Logger.getLogger("com.example.BasicLogging");  
  
    try {  
        bl.logger.addHandler(new FileHandler("Basic.log"));  
        bl.logger.setLevel(Level.INFO);  
        bl.logMessages();  
    } catch (IOException e) {  
        e.getMessage();  
    }  
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Logger Example: logMessages()

```
public void logMessages() {  
    logger.severe("A very severe problem has occurred");  
    logger.warning("Warning, something bad may be happening");  
    logger.log(Level.INFO, "Here is INFO you may want to know  
about");  
    logger.config("Here is some info about your CONFIG");  
    logger.fine("This is some really FINE info");  
    logger.finer("This is even FINER info");  
    logger.finest("This is the FINEST info");  
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Logger Example: basic.log

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE log SYSTEM "logger.dtd">
<log>
<record>
    <date>2011-12-21T19:11:51</date>
    <millis>1324519911801</millis>
    <sequence>0</sequence>
    <logger>com.example.BasicLogging</logger>
    <level>SEVERE</level>
    <class>com.example.BasicLogging</class>
    <method>logMessages</method>
    <thread>1</thread>
    <message>A very severe problem has occurred</message>
</record>
<!-- Additional records here -->
</log>
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

If you look at the file created from the program, only three messages are included the SEVERE, WARNING, and INFO messages. The other messages are not included, why?

Because of the logging level. Setting the logging level to INFO means that only messages of the INFO level and more severe are included.

Logger Example: SimpleLogging.java

```
public static void main(String[] args) {  
    SimpleLogging sl = new SimpleLogging();  
    sl.logger = Logger.getLogger("com.example.SimpleLogging");  
  
    try {  
        FileHandler fh = new FileHandler("simple.log");  
        fh.setFormatter(new SimpleFormatter());  
        sl.logger.addHandler(fh);  
        sl.logger.setLevel(Level.CONFIG);  
        sl.logMessages();  
    } catch (IOException e) {  
        e.getMessage();  
    }  
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Once you create a handler, you assign the formatter of your choice as shown in the slide.

Logger Example: simple.log

```
Dec 21, 2011 7:44:49 PM com.example.SimpleLogging logMessages
SEVERE: A very severe problem has occurred
Dec 21, 2011 7:44:49 PM com.example.SimpleLogging logMessages
WARNING: Warning, something bad may be happening
Dec 21, 2011 7:44:49 PM com.example.SimpleLogging logMessages
INFO: Here is INFO you may want to know about
Dec 21, 2011 7:44:49 PM com.example.SimpleLogging logMessages
CONFIG: Here is some info about your CONFIG
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The SimpleFormatter produces simple single-line messages like those shown in the slide. Notice that, because the logging level was set to CONFIG, four messages are included in this log file.

Quiz

Which two are default logging formatters?

- a. SimpleFormatter
- b. XMLFormatter
- c. BasicFormatter
- d. HTMLFormatter

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Topics

- Logging API overview
- Logging configuration
- Logging examples
- Configuration files and custom handlers

File-Based Configuration

Logging configuration can be set up using a configuration file.

- The file is read at startup.
- It is a standard `java.util.Properties` file.
- A default sample file can be found in:
`C:\ProgramFiles\Java\jdk1.7.0_03\jre\lib\logging.properties`



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Logging configuration can be controlled from a Java Properties file, `logging.properties`. An example file is included in:

`C:\ProgramFiles\Java\jdk1.7.0_03\jre\lib\logging.properties`

Custom Log Handler

You can write your own handler to format logging messages.

- Extend the `Handler` class.
- Implement the `publish`, `flush` and `close` methods.
- `CustomLogHandler.java`

Sample output:

```
0 Dec 21, 2011 8:53:33 PM SEVERE A very severe problem has  
occurred com.example.CustomLogging logMessages  
1 Dec 21, 2011 8:53:33 PM WARNING Warning, something bad may be  
happening com.example.CustomLogging logMessages  
2 Dec 21, 2011 8:53:33 PM INFO Here is INFO you may want to know  
about com.example.CustomLogging logMessages
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

If you want more control over the information included in the log, you can create your own log handler.

Summary

In this lesson, you should have learned how to:

- Add a logging mechanism to your Java application
- Log messages to a file at an appropriate level
- Configure your logger
- Configure logging formatters and handlers



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 16: Overview

Practice 16-1: Logging in Java Applications



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

17

Implementing Unit Testing and Using Version Control

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Set up a unit test system
- Write test cases
- Apply JUnit test framework
- Run unit tests against source code
- Create a test suite
- Use a version control system



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Topics

- Unit testing, test cases, and features of JUnit
- JUnit test cases
- NetBeans support for JUnit
- Version control system

What Is Unit Testing?

Unit testing:

- Refers to testing individual objects through their interfaces
- Is used to validate functionality of individual components
- Enables composition and debugging of code blocks



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Need for Unit Testing

To understand the need for unit testing, consider this classic example of building a bridge. The raw materials act as units or components and each of the individual components contributes to the construction of the bridge. You cannot build a strong bridge unless you can trust the characteristics of your components, such as cables, girders, cement, and reinforcing rods. Similarly, you cannot create a robust application without validating each of the individual components that contribute to the application development. By performing unit testing, you can ensure that the composition debugging is achieved because it enables you to decide if the bug is in a component or between components. A good unit test should depend (as much as possible) only on the component being tested and only interact with its public interfaces. Every important component should have unit tests.

Test Cases and Their Uses

A test case:

- Refers to a set of test data and test scripts and their expected results
- Can be created while developing code
- Validates one or more components' requirements
- Can be used for regression testing, as well as for validation testing



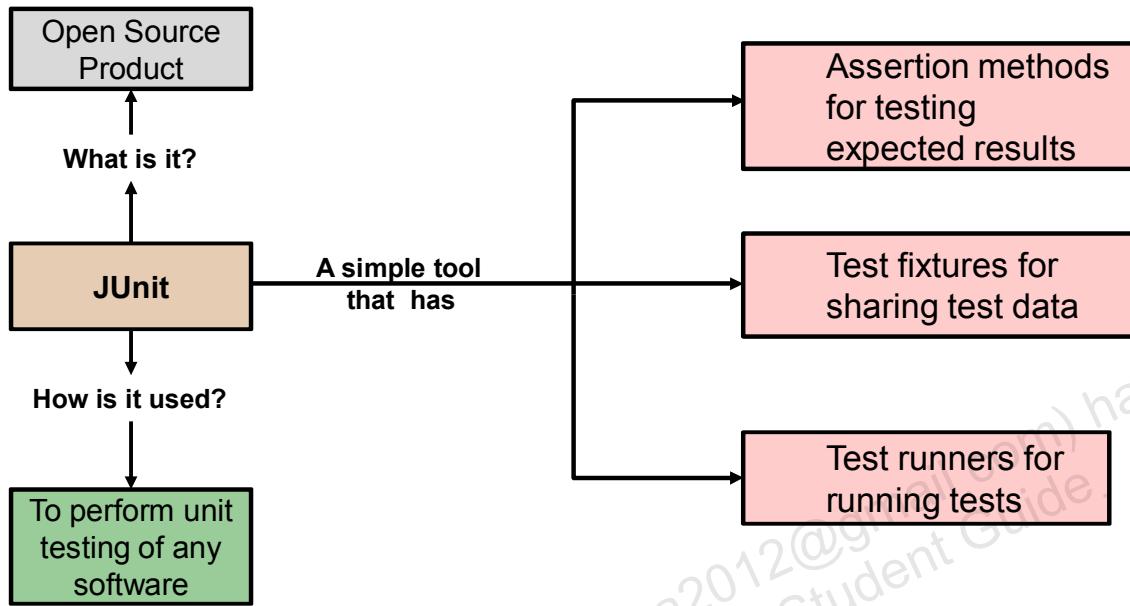
Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Understanding Test Cases and Their Uses

To perform unit tests, consider framing test cases. A test case refers to a set of test data and test scripts and their expected results. You can create test cases as you develop code. A test case validates one or more components' requirements and validates results in the form of a PASS or a FAIL value. Unit tests are coded with these test cases as reference baselines. Unit tests are typically used to test each functionality independently. However, they can be used for regression testing, as well as for validation testing.

Note: Regression testing is the process of testing changes to programs to make sure that the older program still works with the new changes. There are certain requirements of a test that must satisfy certain specifications of an application. These are referred to as validations, and the tests written for this purpose are called validation tests.

Features of JUnit



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A testing framework must provide the following:

- Automated testing
- Self-verification
- Simultaneously running

JUnit is a tool for developing, compiling, and running unit tests. It is an open source product.

JUnit helps you to write code for test cases faster by providing application programming interfaces (APIs) that describe attributes and variables for developing test cases. Because the test cases are based on an existing framework, they have a well-defined structure and are easy to use.

JUnit establishes synergy between coding and testing. JUnit can also test the whole application.

Tests need to run against the background of a known set of objects. This set of objects is called a test fixture. When you are writing tests, you will often find that you spend more time writing the code to set up the fixture than you do in actually testing values.

Test Driven Development

Advantages of test driven development:

- All code of the system is covered by tests.
- The system is loosely coupled, because it comprises independent objects.
- The system develops iteratively with steady progress as each component is tested during development.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Test-driven development is based on the concept of writing a test before writing code. There are several advantages of this style of development.

While writing the tests first, you refactor code as you write. As a result, you improve the design of your system as you build it, thereby reducing the cost of building new features. A good design is one that does not have any redundancy and has only the required classes and methods with self-explanatory names. A well-designed system is easy to modify, easy to extend, easy to understand, and, thus, easy to maintain.

Quiz

Which two of the following statements are true about unit testing and JUnit:

- a. A test case refers to a set of test data and test scripts and their expected results.
- b. The basic principle behind JUnit is the comparison of expected and actual results by a process called test fixtures.
- c. Unit testing refers to testing individual objects through their interfaces.
- d. JUnit is an open source product for developing, compiling, and running software programs.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Topics

- Unit testing, test cases, and features of JUnit
- **JUnit test cases**
- NetBeans support for JUnit
- Version control system



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Annotations in JUnit 4.x

The following annotations are available in JUnit 4.x:

- `@Test`
- `@Before`
- `@After`
- `@BeforeClass`
- `@AfterClass`
- `@Ignore`
- `@Test (expected = Exception.class)`
- `@Test(timeout=100)`

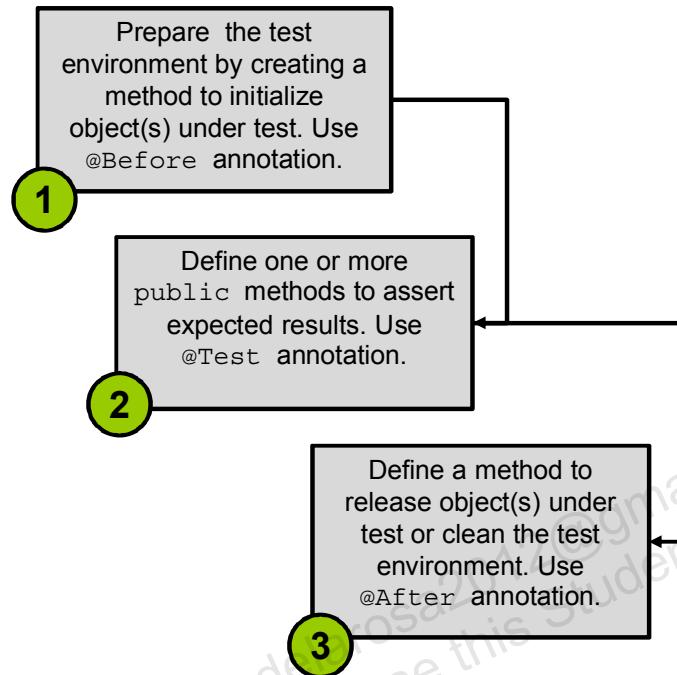


Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Annotation Description

- `@Test public void method()`: The annotation `@Test` identifies that a method is a test method.
- `@Before public void method()`: Execute the method before each test. This method can prepare the test environment (examples: read input data, initialize the class).
- `@After public void method()`: Execute the method after each test. This method can clean up the test environment (examples: delete temporary data, restore defaults).
- `@BeforeClass public void method()`: Execute the method once, before the start of all tests. This can be used to perform time-intensive activities (example: connect to a database).
- `@AfterClass public void method()`: Execute the method once, after all tests have finished. This can be used to perform clean-up activities (example: disconnect from a database).
- `@Ignore`: Ignore the test method. This is useful when the underlying code has been changed and the test case has not yet been adapted, or if the execution time of this test is too long to be included.
- `@Test (expected = Exception.class)`: Fail, if the method does not throw the named exception
- `@Test(timeout=100)`: Fail, if the method takes longer than 100 milliseconds

Steps to Write a Test Case



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

JUnit 4.x is a test framework that uses annotations to identify methods that are tests. JUnit assumes that all test methods can be executed in an arbitrary order. Therefore, tests should not depend on other tests.

Writing a Test

```
1 ...
2 public class StringTest {
3 protected String alpha;
4 protected String beta;
5 public void StringTest(String test){
6 }
7 @Before public void setUp() {
8 alpha="Code";
9 beta= "Test";
10 }
11 @Test public void testConcat() {
12 String actualResult = alpha.concat(beta);
13 String expectedResult = "CodeTest";
14 // Either line 13 or 15 could be used for assertion.
15 assertEquals(actualResult, expectedResult);
16 }
17 @After public void tearDown() {
18 alpha=null;
19 beta=null;
20 }
...
21 }
```

The diagram illustrates the basic steps for writing a simple test. It shows three numbered circles (1, 2, 3) connected by arrows to three specific code blocks in a Java class:

- Circle 1 points to the `@Before public void setUp()` method.
- Circle 2 points to the `@Test public void testConcat()` method.
- Circle 3 points to the `@After public void tearDown()` method.

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The basic steps used to write simple tests, as shown in the slide:

1. In the `@Before public void setup` method, create an object and put it in a known state.
2. In the `@Test testConcat` method:
 - Invoke a method that returns the “actual result.”
 - Create the “expected result,” which may be a primitive value or a more complex object.
 - Invoke `assertEquals(expectedResult, actualResult)`.
3. In the `@After public void tearDown` method, release the objects.

Assert Statements

- The assert statements enable easy comparison and assertion of results.
- You can verify the expected results by `assertTrue`, `assertEquals` and various other JUnit API methods within the predefined test methods.
- Each of the assertion methods accepts an optional first parameter a `String` , to display a failure message when the assertion fails.



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Assert Statement Types

- `fail(String)`: Let the method fail. Might be used to check that a certain part of the code is not reached, or to have failing test before the test code is implemented.
- `assertTrue(true) / assertTrue(false)`: Will always be true / false. Can be used to predefine a test result, if the test is not yet implemented. `assertTrue([message], boolean condition)` Checks that the Boolean condition is true.
- `assertEquals([String message], expected, actual)`: Tests that two values are the same. Note: for arrays the reference is checked not the content of the arrays.
- `assertEquals([String message], expected, actual, tolerance)`: Test that float or double values match. The tolerance is the number of decimals, which must be the same.
- `assertNull([message], object)`: Checks that the object is null
- `assertNotNull([message], object)`: Checks that the object is not null
- `assertSame([String], expected, actual)`: Checks that both variables refer to the same object
- `assertNotSame([String], expected, actual)`: Checks that both variables refer to different objects

Test a Method That Throws an Exception

```
@Test (expected=IllegalArgumentException.class)
public void checkExpectedException() {
    System.out.println("* UtilsJUnit4Test: test
method 3 - checkExpectedException()");
    final int factorialOf = -5;
    System.out.println(factorialOf + " ! = " +
        Utils.computeFactorial(factorialOf));
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

This test demonstrates how to test for an expected exception. The method fails if it does not throw the specified expected exception. In this case, you are testing that the `computeFactorial` method throws an `IllegalArgumentException` if the input variable is a negative number (-5).

Test a Method That Throws an Exception

```
24 ...
25     @Test (expected=IndexOutOfBoundsException.class)
26 public void testForLimits() {
27
28     Object obj = initialList.get(0);
29 }
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The slide has a code snippet showing another example of testing whether a method throws an expected exception.

Run the Tests

- The class `org.junit.runner.JUnitCore` provides the method `runClasses()`, which allows you to run one or several tests classes.
- An object of the type `org.junit.runner.Result` is returned, which can be used to retrieve information about the tests.

```
import org.junit.runner.JUnitCore; I
import org.junit.runner.Result;
import org.junit.runner.notification.Failure;
public class MyTestRunner {
    public static void main(String[] args) {
        Result result = JUnitCore.runClasses(MyClassTest.class);
        for (Failure failure : result.getFailures()) {
            System.out.println(failure.toString());
        }
    }
}
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Creating a Test Suite

The TestSuite class should contain the following lines of code:

```
package mypackage;  
import org.junit.runner.RunWith;  
import org.junit.runners.Suite;  
@RunWith(Suite.class)  
  
{@Suite.SuiteClasses(value={CurrencyJUnit4Test.class,  
BankJUnit4Test.class})}  
public class JUnit4TestSuite { }
```



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

When you run the test suite, the IDE will run the test classes in the order in which they are listed. If you develop another test later, you can add it to @Suite.SuiteClasses.

Quiz

Identify the options that are true about assert statements.

- a. The Assert class provides the methods that you can use to compare results or test objects.
- b. assertTrue is the only JUnit API method that can be used to verify expected results.
- c. Each of the assertion methods accepts an optional first parameter, a String, to display a failure message when the assertion fails.

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Quiz

Examine the following code snippet and identify the statements that are true.

```
@RunWith(Suite.class)
@Suite.SuiteClasses(value={CurrencyJUnit4Test.class,
BankJUnit4Test.class})
```

- a. The code snippet shows the contents of a Test Suite class.
- b. When you run the test suite, the IDE will run the test classes in the order in which they are listed.
- c. The code snippet shows how to write a Test Case.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Topics

- Unit testing, test cases, and features of JUnit
- JUnit test cases
- NetBeans support for JUnit
- Version control system



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

JUnit Support in NetBeans

The JUnit module in NetBeans:

- Provides a results window
- Generates a test code skeleton for each testable method
- Makes navigation between source files and corresponding test files easy
- Provides an easy “Run File” option to run the tests corresponding to an application
- Can generate Test packages for all project types



Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

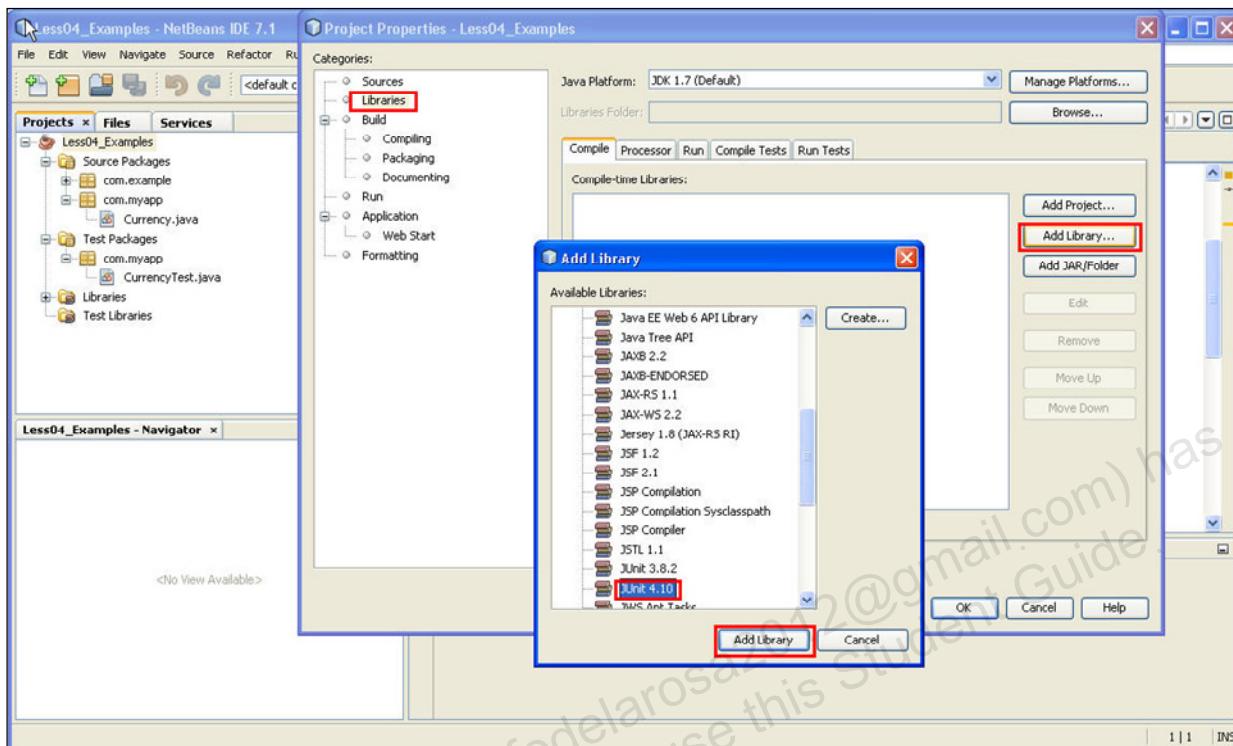
Netbeans has provided an implementation of the JUnit framework and supports JUnit for unit testing.

Netbeans, along with Junit, provides a conducive environment for testing applications, enabling the generation of test cases, and providing an environment for running the test cases.

The JUnit module in Netbeans makes creating and running JUnit tests easier. NetBeans support JUnit 3.8 and JUnit 4.1

- Test classes are kept in separate source directories.
- Each generated test class contains test methods for all accessible methods from the tested class. These methods contain either a simple skeleton suitable for comparing return values of tested methods with values expected by the tests (by default), or these methods can be left empty, in which case the body of the test method must be filled in by the developer to hold some reasonable test code.
- Generated test classes can be compiled and, by default, print the names of tested methods. It provides a results window for a quick overview of the status of executed tests.

JUnit Support in NetBeans

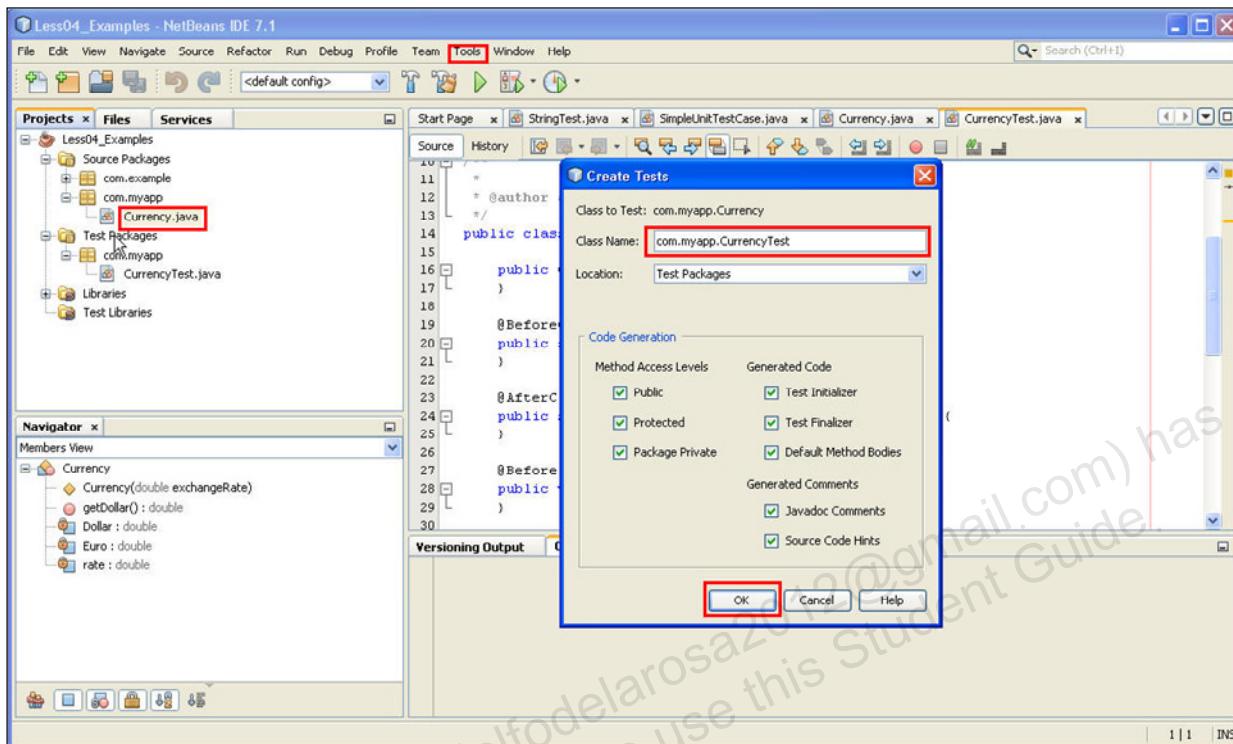


ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The JUnit module in Netbeans makes creating and running JUnit tests easier. You need to add the JUnit 4.1 library to the project.

Generating JUnit Test Classes in NetBeans

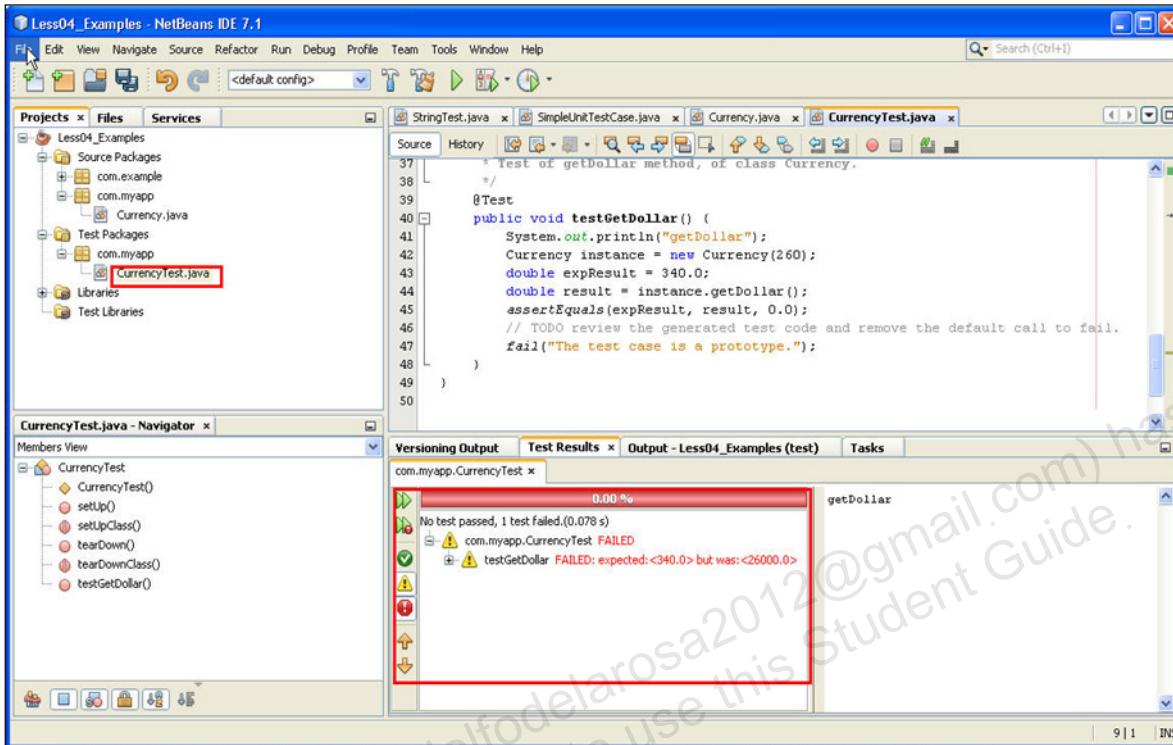


ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Test classes are kept in separate source directories. Each generated test class contains test methods for all accessible methods from the tested class. These methods contain either a simple skeleton suitable for comparing return values of tested methods with values expected by the tests (by default), or these methods can be left empty, in which case, the body of the test method must be filled in by the developer to hold some reasonable test code. Generated test classes can be compiled and, by default, print the names of tested methods. It provides a results window for a quick overview of the status of executed tests.

Running Tests in NetBeans



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

You can run a test class and view the output of the test methods in the TestResults window

Quiz

Identify the correct option for JUnit support in NetBeans.

- a. NetBeans can generate Test packages for specific project types.
- b. NetBeans provides a conducive environment for testing applications, enabling the generation of test cases, and providing an environment for running the test cases.
- c. Test classes are kept in same directories as the source code.

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Topics

- Unit testing, test cases, and features of JUnit
- JUnit test cases
- NetBeans support for JUnit
- Version control system



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Version Control System

Version control and source control is :

- An aspect of software configuration management (SCM)
- The management of changes to documents, programs, large websites, and other information stored as computer files
- Is most commonly used in an environment where a team of people may change the same files



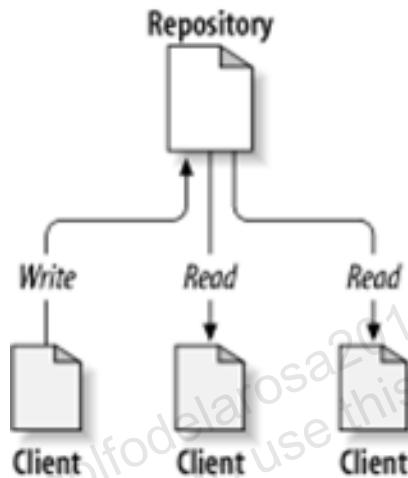
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Changes are usually identified by a number or letter code, termed the "revision number," "revision level," or simply "revision." For example, an initial set of files is "revision 1." When the first change is made, the resulting set is "revision 2," and so on. Each revision is associated with a timestamp and the person making the change.

A version control system (or revision control system) is a system that tracks incremental versions (or revisions) of files and, in some cases, directories over time. What makes a version control system useful is the fact that it allows you to explore the changes that resulted in each of those versions and facilitates the arbitrary recall of the same.

Advantages of Using a Version Control System

- Automatic backups
- Sharing on multiple computers
- Version control and branching
- Maintain history



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

A modern version control system (VCS) has four clear benefits over the “folder backup” method.

- If you accidentally delete some file (or part of a file), you can undelete it. If you change something and want to undo it, the VCS can do so.
- VCSes are designed to help multiple people collaboratively edit files. This makes sharing between multiple computers particularly easy. You do not need to worry whether you copied the latest version; the VCS does that for you. Even if you are offline and make changes to files on both computers, the VCS will merge the changes intelligently once you are online.
- If you fix bugs in one version of code, the VCS will merge them to the other versions.
- VCS allows you to keep track of the development of your source code, keeping track of the changes you make as you go along with all the changes being stored in a repository.

The figure in the slide shows a typical client server system. At the core of the version control system is a repository, which is the central store of that system's data. The repository usually stores information in the form of a *filesystem* tree, which is a hierarchy of files and directories. Any number of clients connect to the repository, and then read or write to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others. Popular version control systems are: Git, CVS, VCS, Subversion, TLA, Darcs, and Mercurial.

Features of the Subversion (SVN) Tool

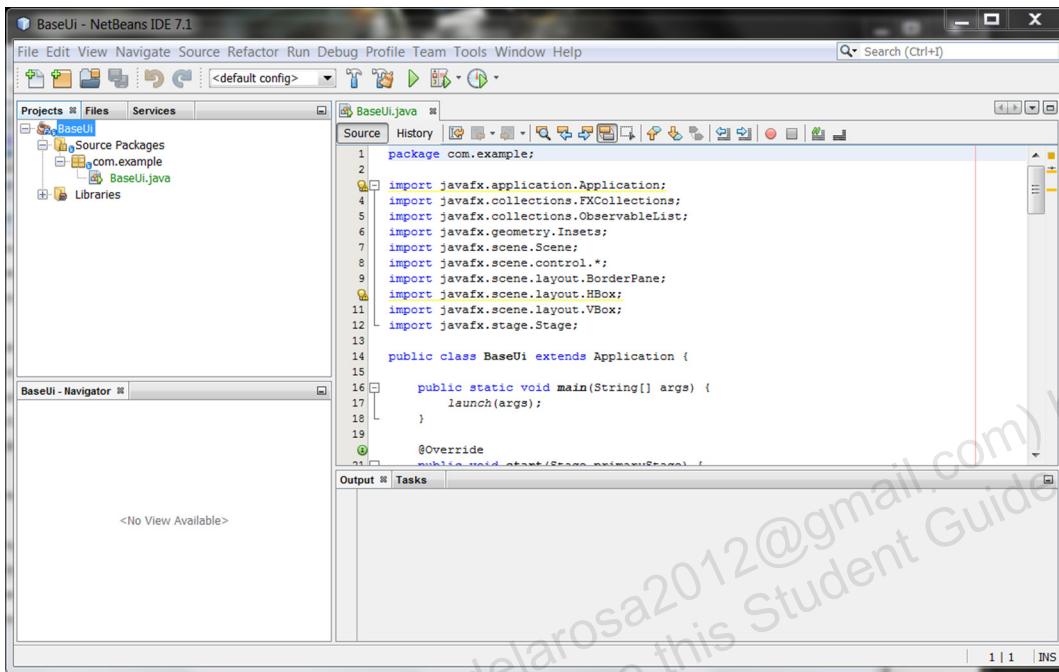
- Subversion is a modern, network-aware version control system.
- Subversion offers several different ways for its clients to communicate with its servers—the URLs used to address the repository differ subtly depending on which repository access mechanism is employed.
- Updates and commits are separate.

ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Schema	Access method
file:///	Direct repository access (on local disk)
http://	Access via WebDAV protocol to Subversion-aware Apache server
https://	Same as http://, but with SSL encryption
svn://	Access via custom protocol to an svnserve server
svn+ssh://	Same as svn://, but through an SSH tunnel

Using NetBeans to Manage Code in SVN

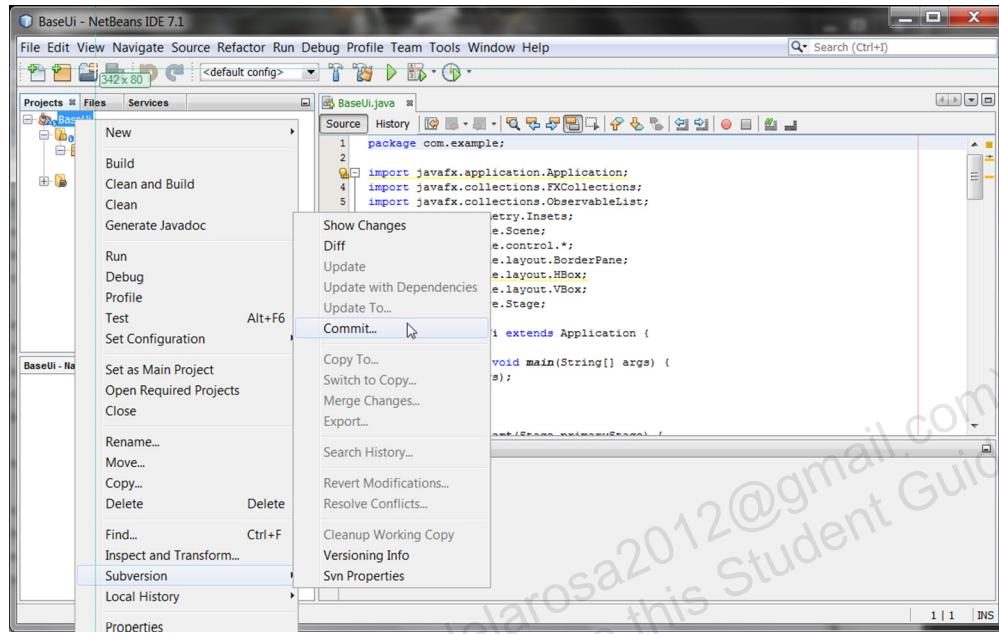


ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

You can install and configure an SVN client in your system. After the setup, you can use NetBeans to manage code or projects in SVN.

Using NetBeans to Commit Code to SVN

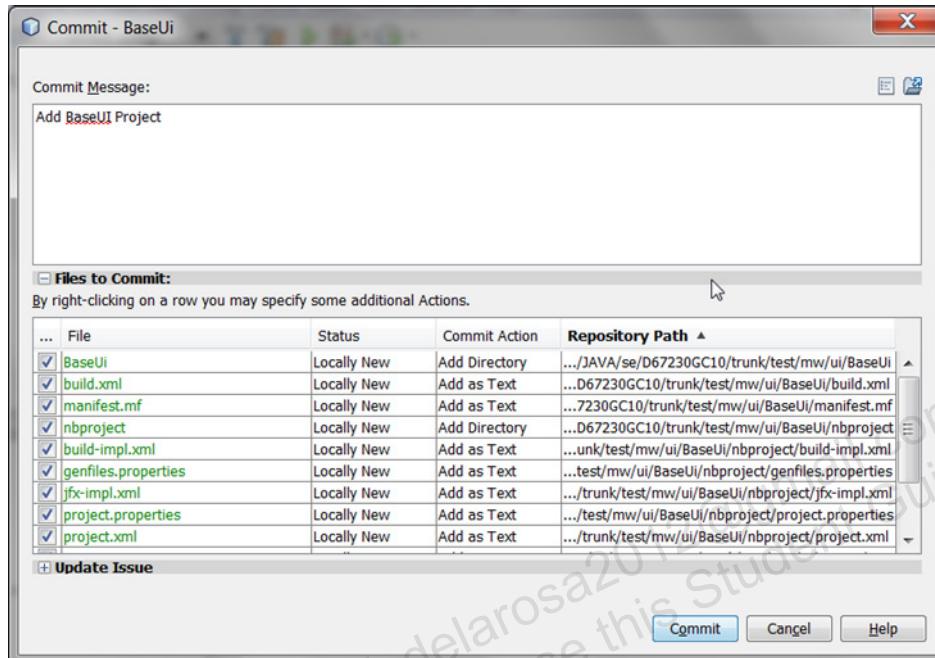


ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The screenshot in the slide shows the commit option in the Subversion context menu.

Using NetBeans to Commit to SVN



... File	Status	Commit Action	Repository Path
✓ BaseUi	Locally New	Add Directory	.../JAVA/se/D67230GC10/trunk/test/mw/ui/BaseUi
✓ build.xml	Locally New	Add as Text	...D67230GC10/trunk/test/mw/ui/BaseUi/build.xml
✓ manifest.mf	Locally New	Add as Text	...7230GC10/trunk/test/mw/ui/BaseUi/manifest.mf
✓ nbproject	Locally New	Add Directory	...D67230GC10/trunk/test/mw/ui/BaseUi/nbproject
✓ build-impl.xml	Locally New	Add as Text	...unk/test/mw/ui/BaseUi/nbproject/build-impl.xml
✓ genfiles.properties	Locally New	Add as Text	...test/mw/ui/BaseUi/nbproject/genfiles.properties
✓ jfx-impl.xml	Locally New	Add as Text	.../trunk/test/mw/ui/BaseUi/nbproject/jfx-impl.xml
✓ project.properties	Locally New	Add as Text	.../test/mw/ui/BaseUi/nbproject/project.properties
✓ project.xml	Locally New	Add as Text	.../trunk/test/mw/ui/BaseUi/nbproject/project.xml

Update Issue

Commit Cancel Help

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The screenshot in the slide shows the window that shows the code being committed to SVN directory.

Quiz

Identify the options that are true about version control.

- a. A version control system is the management of changes to documents, programs, large websites, and other information stored as computer files.
- b. Version control systems are designed to help multiple people collaboratively edit files.
- c. Subversion is the only version control system available today.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Summary

After completing this lesson, you should have learned how to:

- Set up a unit test system
- Write test cases
- Apply JUnit test framework
- Run unit tests against source code
- Create a test suite
- Use a version control system



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 17: Overview

- Practice 17-1: Creating and Executing Test Cases
- Practice 17-2: Writing Test Cases Using Additional JUnit 4 Annotations
- Practice 17-3: Creating a Test Suite
- Practice 17-4: Performing Parameterized Testing



ORACLE®

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Adolfo De+la+Rosa (adolfodelarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.



18



Oracle Cloud

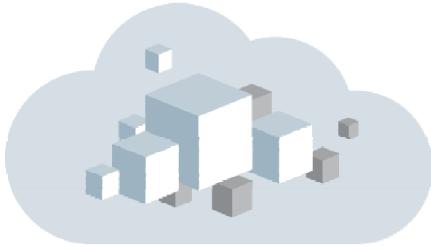
An Overview

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Adolfo De+la+Rosa (adolfo.delarosa02@gmail.com) has a
non-transferable license to use this presentation guide.

Agenda



- 1** What is Cloud Computing?
- 2** Cloud Evolution
- 3** Components of Cloud Computing
- 4** Characteristics and Benefits of Cloud
- 5** Cloud Deployment Models
- 6** Cloud Service Models
- 7** Oracle Cloud Services

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

What is Cloud?

The term Cloud refers to a Network or Internet.

It is a means to access any Software that is available remotely.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

What is Cloud Computing?

- It is a means to access any Software that is available remotely.
- Refers to the practice of using remote Servers hosted on Internet to store, manage and process data
- When you store your photos online instead of on your home computer, or use webmail or a social networking site, you are using a “cloud computing” service.

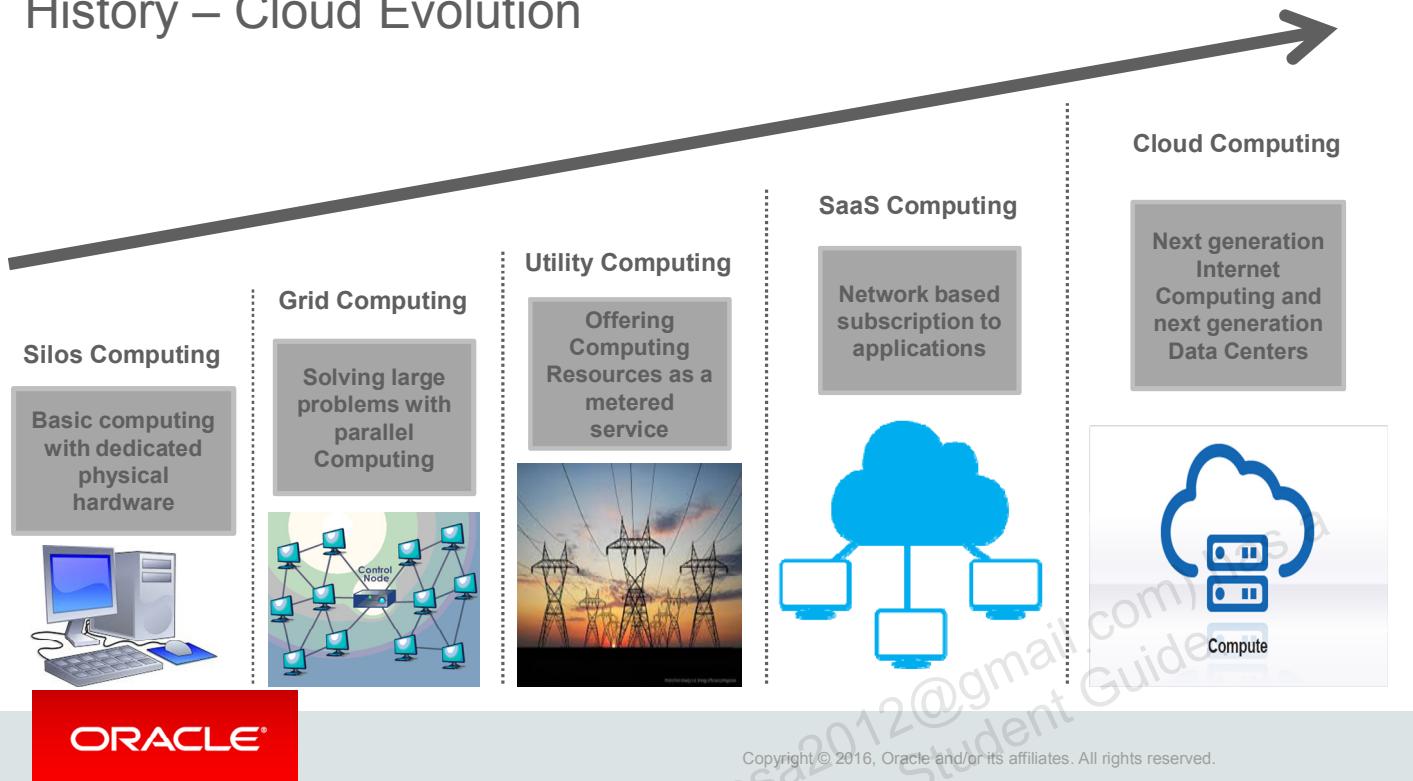


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

History – Cloud Evolution

Unauthorized reproduction or distribution prohibited. Copyright © 2020, Oracle and/or its affiliates.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Components of Cloud Computing

Client Computers



Devices that end user interact with cloud. Types of client Thick, Thin (Most popular), Mobile

Distributed Servers



Often Servers are in geographically different places, but server acts as if they are next to each other

Data Centers

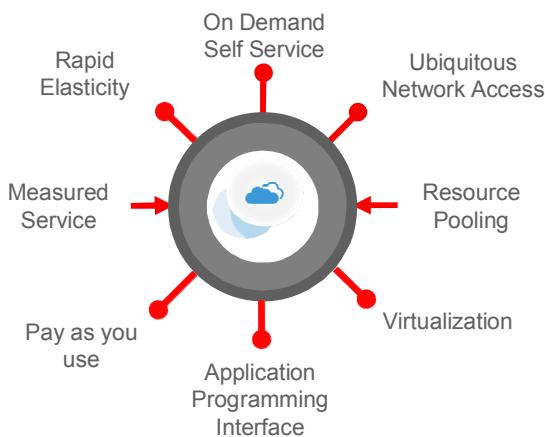


Collection of servers where application is placed and is accessed via Internet

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Characteristics of Cloud



Description

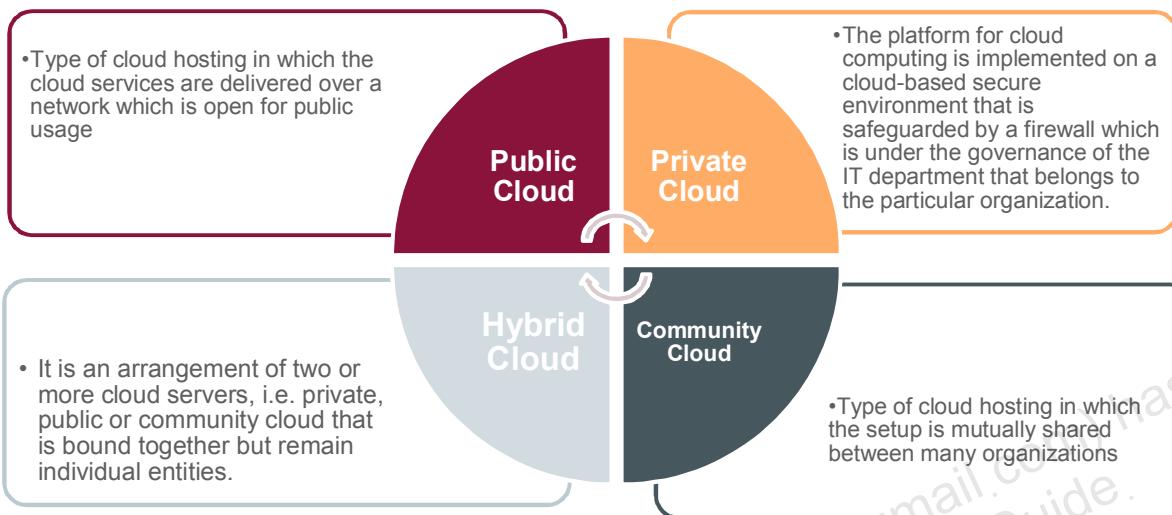
- Allows users to use the service on demand
- Anywhere, Anytime and Any Device
- Draw from a pool of computing resources, usually in remote data centers
- Request and manage own computing resources
- Service is measured and customers are billed accordingly
- Select a configuration of CPU, Memory and storage
- Services can be scaled larger or smaller

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Cloud Deployment Models

Deployment models define the type of access to the Cloud.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Cloud Service Models

All three tiers of computing delivered as Service via global network

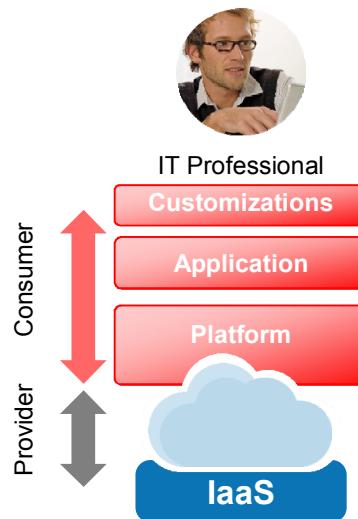
- **Applications:** Software as a Service - SaaS
- **Platform:** Database, Middleware, Analytics, Integration as a Service – Platform as a Service - PaaS
- **Infrastructure:** Storage, Compute, and Network as a service – Infrastructure as a Service - IaaS



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Cloud Service Models

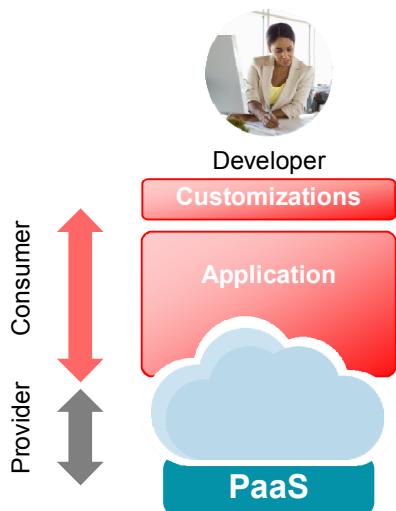


- Provides computer hardware (servers, networking technology, storage and data center space) as a web based service.
- Virtual Machines with pre-installed Operating System
- Target: Administrators
- Ready to Rent

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Cloud Service Models



- Provides platform to develop and deploy applications
- Up to Date Software
- Target: Application Developers
- Ready to Use

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Cloud Service Models



Business End User

Customizations



- Allows usage of the software remotely as a web based service
- Software are automatically Upgraded and Updated
- All Users are running the same version of the Software
- Target: End Users
- Ready to Wear

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Industry Shifting from On-Premises to the Cloud

Transition to the Cloud is driven by a desire for:

- **Agility:** Self-service provisioning – deploy a database in minutes
- **Elasticity:** Scale on demand
- **Lower cost:** Reduction in management and total cost – pay for what is used
- **Back to core business:** Focus on core activities
- **More mobility:** Access from any device



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle IaaS Overview

IaaS

Designed for large enterprises, which allow them to scale up their computing, networking, and storage systems into the cloud, rather than expanding their physical infrastructure.

- Allows large businesses and organizations to run their workloads, replicate their network, and back up their data in the cloud.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Develop, deploy, integrate and manage applications on cloud.
- Seamless integration across PaaS and SaaS Applications.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle SaaS Overview

SaaS

Delivers modern cloud applications that connect business processes across the enterprise.

- Only Cloud integrating ERP, HCM, EPM, SCM
- Seamless co-existence with Oracle's On-Premise Applications



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have:

- Got an overview of Cloud Computing, its Characteristics, History and Technology
- Understood the various components , Deployment Models and Service Models of Cloud Computing
- Understood the Oracle Cloud Services



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.



Oracle Application Container Cloud Service Overview

ORACLE®

19

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Get an overview of Oracle Application Container Cloud
- Understand the unique features of Oracle Application Container Cloud
- Understand how to build, zip, and deploy applications to the cloud



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Application Container Cloud Service



An open highly available
Docker container-based
elastic polyglot cloud
application platform

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Oracle Application Container Cloud

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

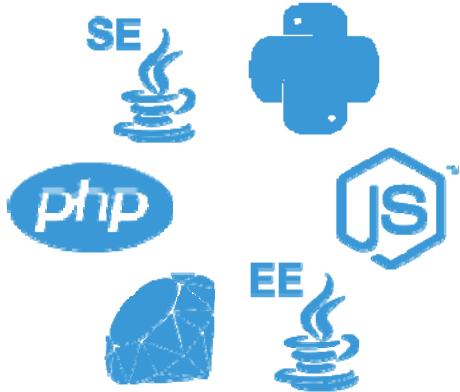


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

- Simple and easy to use deployment platform for Java SE & Node applications
- Open platform—use any application frameworks and libraries
- Runs applications in Docker containers for reliability and scalability

Polyglot Platform



ORACLE®

Runtime releases regularly updated to the latest

Deploy applications to a selection of popular language runtimes supported

- Latest release supports Java SE, Java EE Web Apps, Node.js, and PHP

Leverage unique Oracle Java SE features

- Immediate access to platform upgrades, security, platform optimizations
- Continued commercial support for Java SE versions no longer receiving public updates

Node access to Oracle DB with open source database driver

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Open Platform

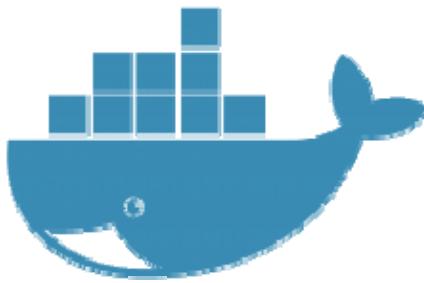
Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Use any of the thousands of open source or commercial Java or Node frameworks—no restrictions.

Container-based Application Platform as a Service



Applications run on Oracle Linux in Docker containers

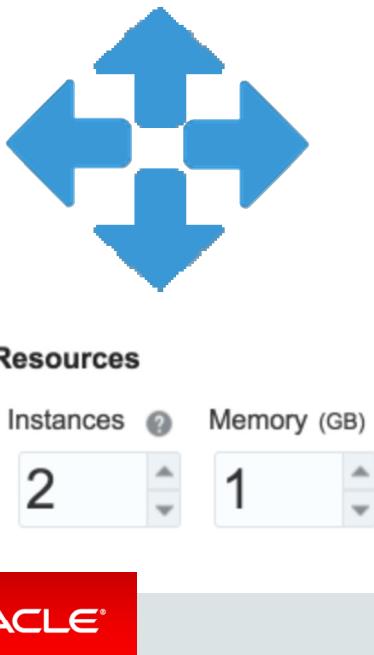
Stateless Applications

- Ephemeral disk
- Permanent storage through database or storage service

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Elastic Scaling



On demand elastic scaling either through the service console or using the service REST API

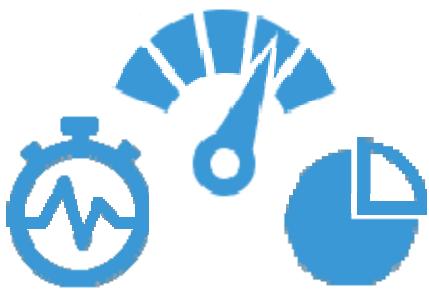
Scale out / in

- Add / remove application instances to handle workloads

Scale up / down

- Add / remove RAM to accommodate application memory requirements

Profiling



Java application can use Java Flight Recorder to monitor application and JVM behavior and analyze in Mission Control

Use Application Performance Monitoring Cloud Service for advanced use cases

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Manageable



New Java and Node releases published in the service console
One-click upgrade to the latest releases—applications are simply restarted to upgrade to new runtime

Updates

Current Version: Java SE 8u71

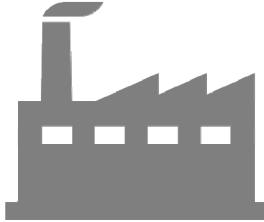
Available Updates

	Runtime: Java SE 8u91	Release Date: Jul 4, 2015 12:00:00 AM UTC	Update
	Release Notes	Description: This update contains new features as well as fix for critical issues. Refer to the 'Release Notes' for more details	

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Build Zip Deploy!



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Build

- Use your favourite or corporate standard build system to produce binaries and deployable resources.

Zip

- Zip up all binaries, scripts, html files, images, etc. that make up your application. The structure of the zip is entirely up to the user—we have no opinion on structure.

Deploy

- Deploy the application archive (zip) to the platform and tell us how to start the application. This could be “java –jar”, “java –classpath ... <main>”, “node myapp.js”, or “sh bootmyapp.sh”.

Deploy—Application Archive (Zip)

- All application binaries
- All required libraries
- Binaries of any container/embedded container
- Images files
- HTML files

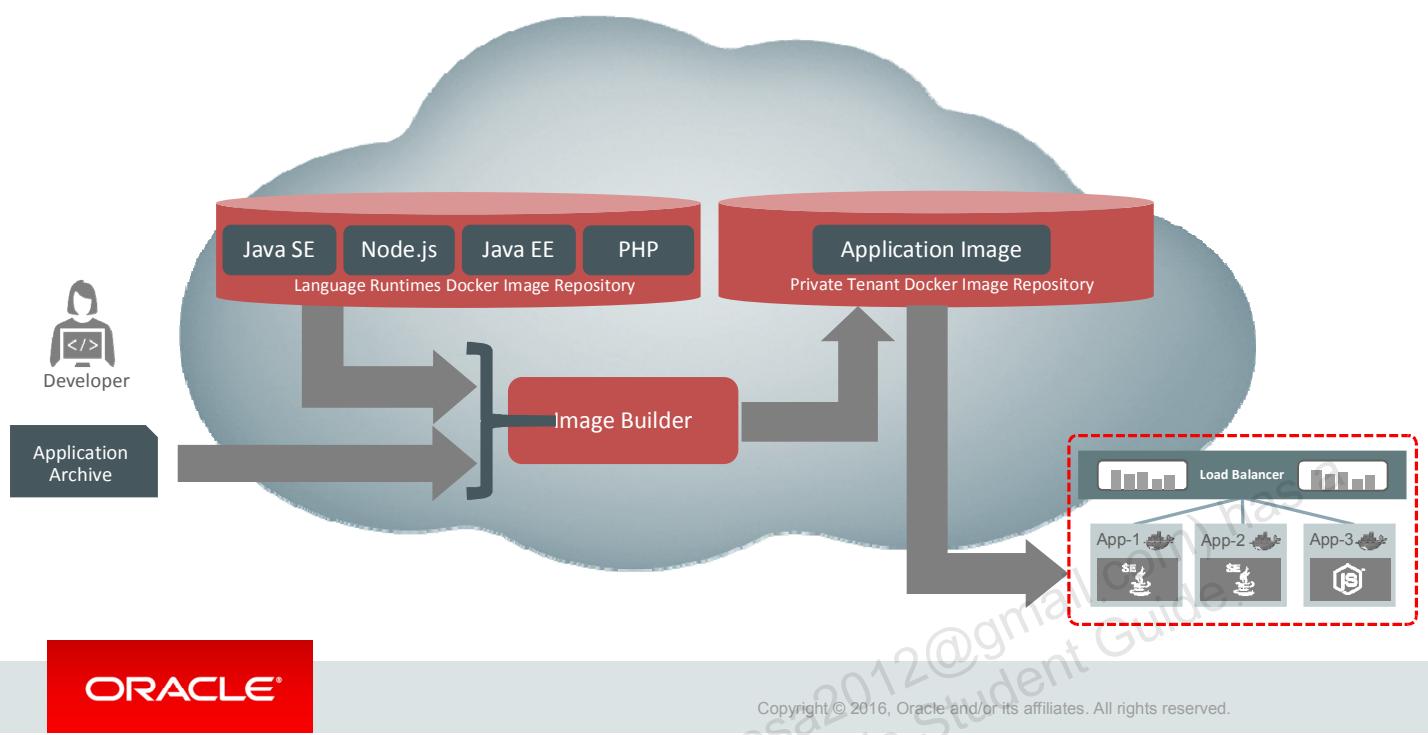
Everything you'd need to run your application on a virgin machine



Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Application Deployment

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

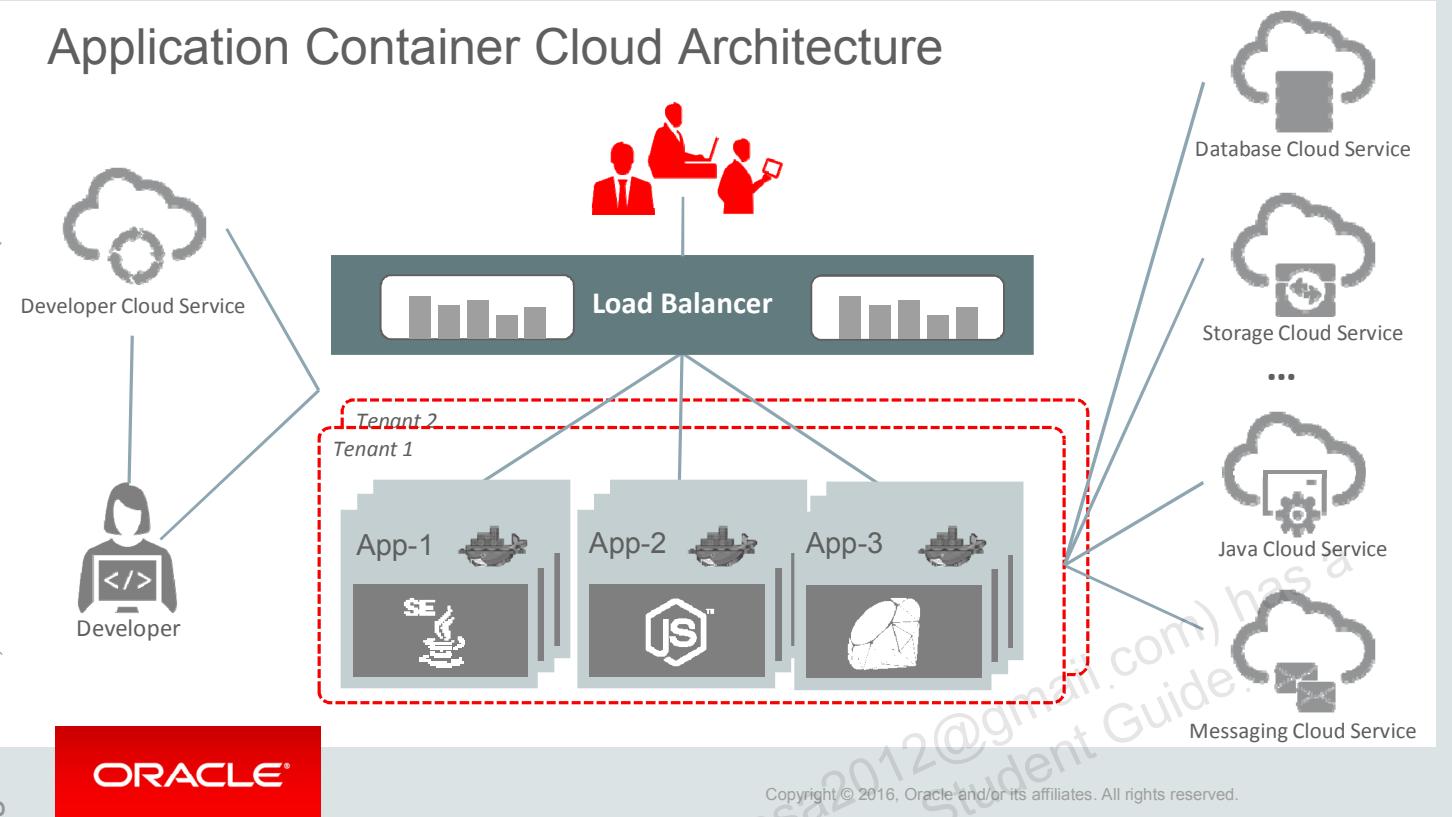


ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Application Container Cloud Architecture

Unauthorized reproduction or distribution prohibited. Copyright © 2020, Oracle and/or its affiliates.



- Tenant Isolation
- Polyglot
- Integrated
- Developer Friendly

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Load Balancer

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.



Fully automated—no user management required

Scale out or in and application instances are automatically registered/unregistered

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Vanity URL support (upcoming) will allow installation of certificates

Oracle Developer Cloud Service



**Source Control
Management**



Issue Tracking



**Hudson Continuous
Integration**



Wiki Collaboration

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Complete, Integrated Development Platform—as a Service

Application Lifecycle Management

Team Management

Entitlement with all Application Container Cloud services

Developer Cloud Service – Easy Adoption/Integration

Pre-integrated development technologies in the cloud

Standards Based

- Git, Maven, Hudson, Ant, Grunt, Gulp, etc.

Built-in IDE Integration

- Eclipse, NetBeans, JDeveloper

Flexible Source Location

- Hosted Git or GitHub

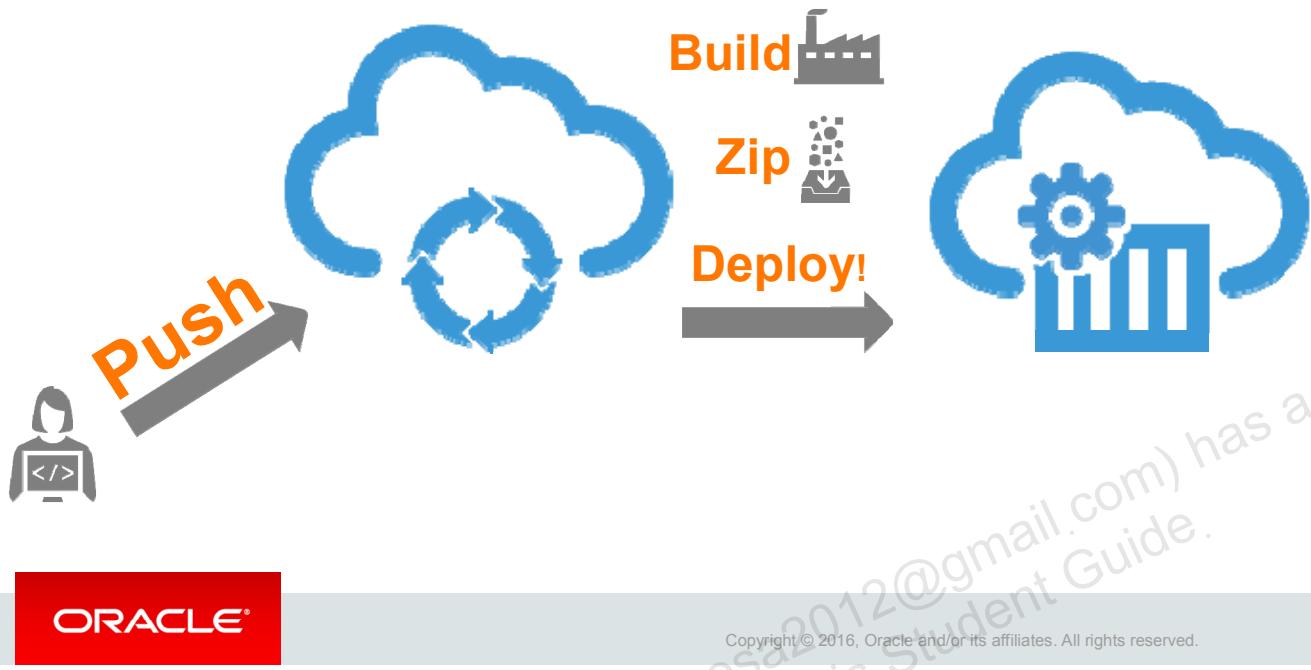
Choice of Deployment Target

- Oracle Cloud or on-premise



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Rather than build on-premise, use DevCS to perform continuous build, test, and deployment.

Application Container Cloud Service Advantages

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.



- Integrated **enterprise** ecosystem and services from IaaS to PaaS and SaaS
- Java SE Advanced – completely **unique** and unavailable on any other cloud platform
- Developer Cloud Service – **included** and **integrated**

ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have :

- Got an overview of Oracle Application Container Cloud
- Understood the unique features of Oracle Application Container Cloud
- Understood how to build, zip, and deploy applications to the cloud



ORACLE®

Copyright © 2016, Oracle and/or its affiliates. All rights reserved.

Development Methodologies and Design Patterns



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Adolfo De la Rosa (adolfo.delarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.

Topics

- Agile Development
- Design Patterns

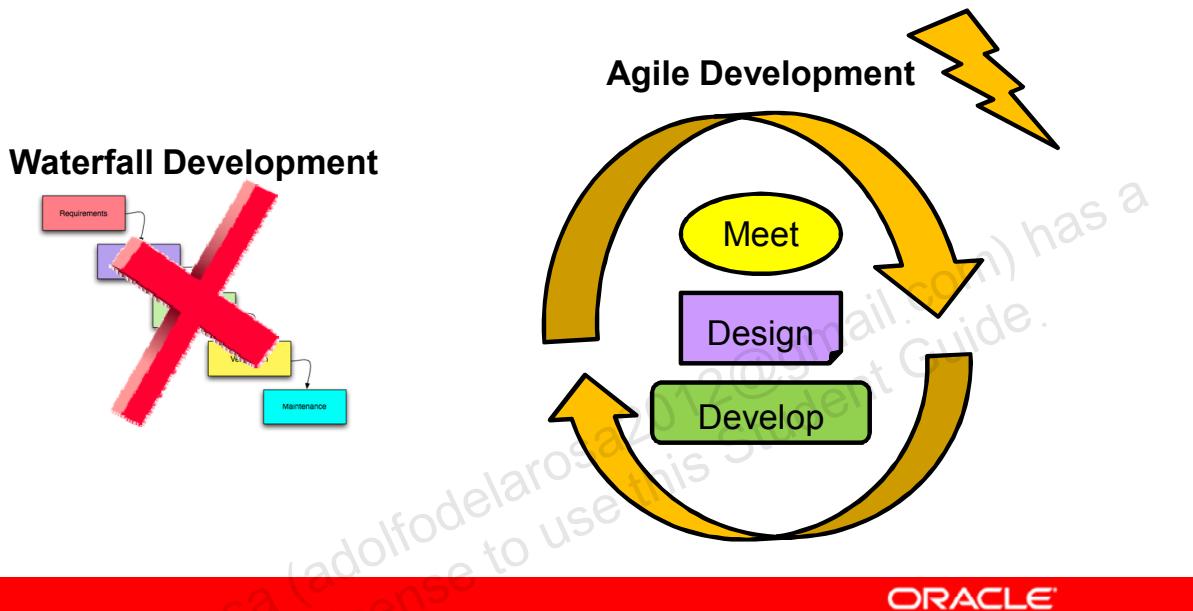


ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Agile Development

Agile development is a development process that emphasizes frequent status meetings, short-term goals, and good communication.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Agile development is a development process that emphasizes frequent status meetings, short-term goals, and good communication. In general, Agile development is a big improvement over the typical Waterfall development. The Waterfall development consists of discrete tasks that must be completed before the new phase can begin. With Agile development, developers do iterative development that lets them complete tasks without focusing on phases. The benefits of Agile development are:

- Reduces project overhead by streamlining process
- Creates better collaboration between team members
- Allows team members to contribute to the best of their abilities
- Promotes iterative development
- Delivers product earlier
- Catches issues sooner
- Receives end-user inputs during development
- Adjusts schedule sooner and creates a better estimate of work

Scrum

Scrum is a type of Agile development methodology, and it has an iterative development style which includes:

- Meeting two to five times a week
- Team members choosing tasks
- Team members working toward short-term goals called sprints
- Integrating builds often
- Putting incomplete tasks in a backlog



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A key principle of Scrum is its recognition that during a project, the customers can change their minds about what they want and need, and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. The key feature of scrum is that you iterate. So by meeting with customers on a regular basis, and showing them sample code on a regular basis, you avoid any surprises. As such, Scrum adopts an empirical approach—accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the group's ability to deliver quickly and respond to emerging requirements. Anything in the backlog at the end of the cycle is evaluated for inclusion in future releases.

Advantages:

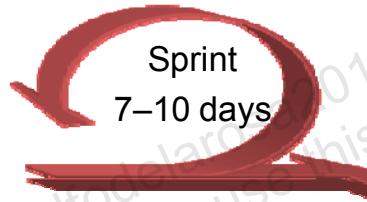
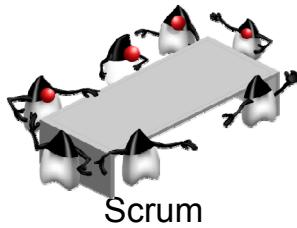
- Increased speed
- Increased flexibility

Disadvantages:

- Decreased design
- Decreased quality

Scrum Terminology

Term	Definition
Scrum	<ul style="list-style-type: none">• Regularly scheduled meeting to identify and assign tasks, two to five times a week• Traditionally limited to 15–30 minutes
Sprint	<ul style="list-style-type: none">• A development iteration• Traditionally a duration of 7–10 days• Designed for iterative tasks
Sprint Goals	<ul style="list-style-type: none">• Intermediate goals designed to demonstrate progress toward the high-level milestones• Typically short term goals



Sprint Goals

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Scrum Roles

Role	Responsibilities
Product Owner	<ul style="list-style-type: none">• Helps identify high level goals and milestones• Helps to resolve issues and remove roadblocks if necessary
Scrum Master (Lead Developer)	<ul style="list-style-type: none">• Leads scrum meetings• Maintains task list• Adapts to changing requirements and problems• Keeps Project Owner informed
Team Member (Developer)	<ul style="list-style-type: none">• Self organized• Volunteers for tasks during each sprint and completes them



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Topics

- Agile Development
- Design Patterns



ORACLE®

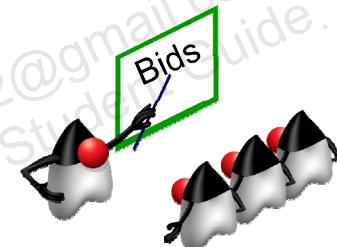
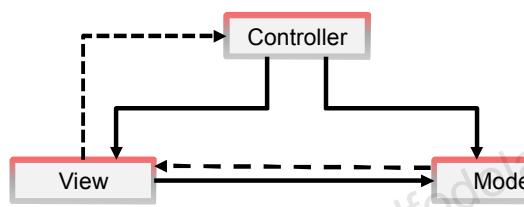
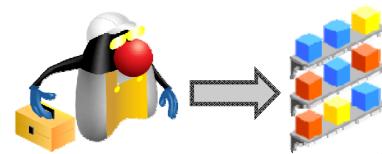
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Design Patterns: An Introduction

A design pattern is a solution to a common software problem.
Object-oriented design patterns show relationships and interactions between objects and classes

Three patterns are used in this course:

- Builder
- Observer
- Model-View-Controller (MVC)



ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

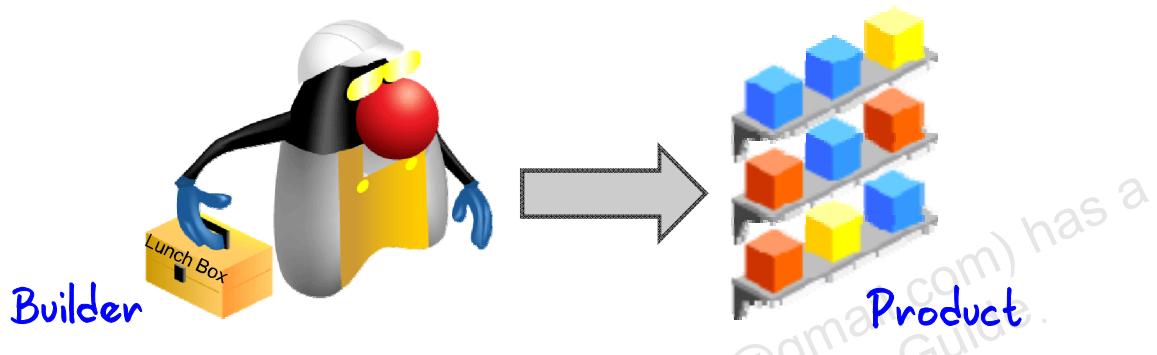
Although described in the early 1960s, design patterns became popular in the 1990s when the Gang of Four introduced design patterns in their book, *Design Patterns: Elements of Reusable Object-Oriented Software*. (The Gang of Four is Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.)

Design patterns provide solutions to common software problems, and they give generalized solutions in the form of templates that can be applied to real-world problems.

- The pattern is a proven and tested solution.
 - Development is more cohesive.
 - Isolates variability.
 - Makes the solution easy to understand.
- The pattern facilitates communication between designers and developers.
 - Both groups will have a mental picture of the high-level design when they think in terms of patterns.
- Can speed up the development process
- Templates can apply to real-world situations.

Builder Design Pattern

The Builder pattern separates the construction of a complex object from its representation and simplifies creation of complex objects.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Builder Pattern Example

```
1 ParallelTransitionBuilder.create()
2   .children(slideOutOld, slideInNew)
3   .onFinished(new EventHandler<ActionEvent>() {
4     @Override public void handle(ActionEvent t) {
      currentPage.setCache(false);
      getChildren().remove(currentPage);
      currentPage = nextPage;
    }
  })
  .build().play();
}
```

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Implementing the Builder Pattern

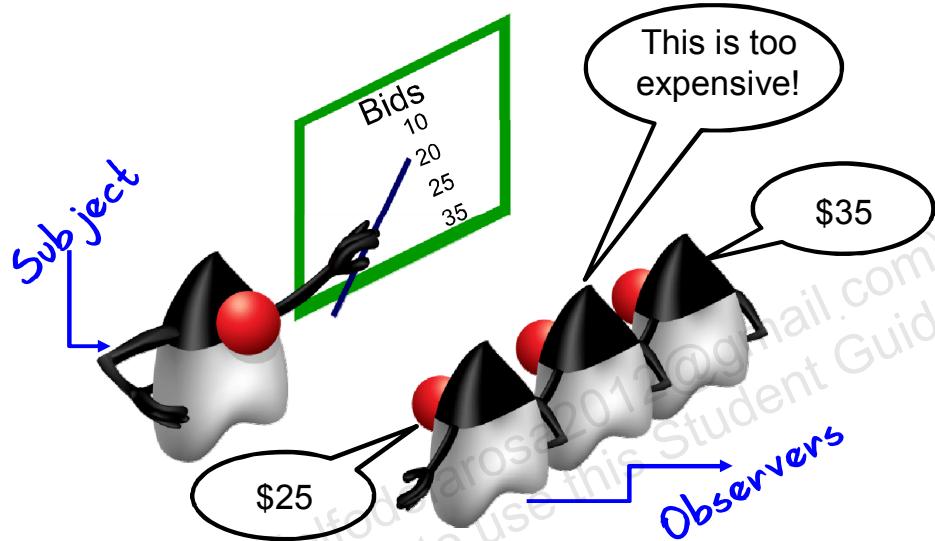
The Builder design pattern is applied to the Henley application animation as the following:

1. ParallelTransitionBuilder is the builder class
2. Defines which node is disappearing and what is appearing
3. Creates the event handler that runs when the animation finishes and resets the current page cache
4. Builds the animation and calls the `play()` method

Observer Design Pattern

The Observer pattern has a *subject* that maintains a list of *observers* and notifies them of changes.

Duke's Auction



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Observer pattern is a software design pattern in which an object, called the *subject*, maintains a list of its dependents, called *observers*, and notifies them automatically of any state changes, usually by calling one of their methods. The Observer pattern defines a one-to-many relationship so that when one object changes state, the others are notified and updated automatically. It is mainly used to implement distributed event-handling systems. The Observer pattern is also a key part in the familiar MVC architectural pattern.

Observer Pattern Example

```
final ObservableList<Integer> bidList = 1  
    FXCollections.observableArrayList(20, 25, 30);  
ListView bidView = new ListView(bidList); 2  
final TextField bidField = new TextField();  
  
bidList.addListener(new ListChangeListener<Change>() { 3  
    @Override  
    public void onChanged(ListChangeListener.Change  
change) {  
  
        sizeField.setText(Integer.toString(bidList.size()));  
    }  
}
```



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

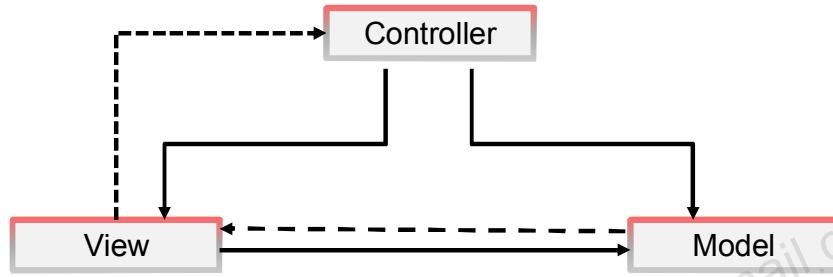
Implementing the Observer Pattern

In the code example in the slide, the `bidList` (1) is the subject. The `ListView` (2) and the `ListChangeListener` event handler (3) are observers.

The `ListView` is set up as one listener and automatically updates each time the list is updated. `ListChangeListener` event handler is a second listener. This displays the size every time you add a new bid. (`size` is the size of the `bidList`.)

The MVC Design Pattern

The pattern isolates the application logic for the user from the user interface.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- The pattern isolates “domain logic” (the application logic for the user) from the user interface (input and presentation), permitting independent development, testing, and maintenance of each (separation of concerns).
- Use of the Model/View/Controller (MVC) pattern results in applications that separate the different aspects of the application (input logic, business logic, and UI logic), while providing a loose coupling between these elements.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

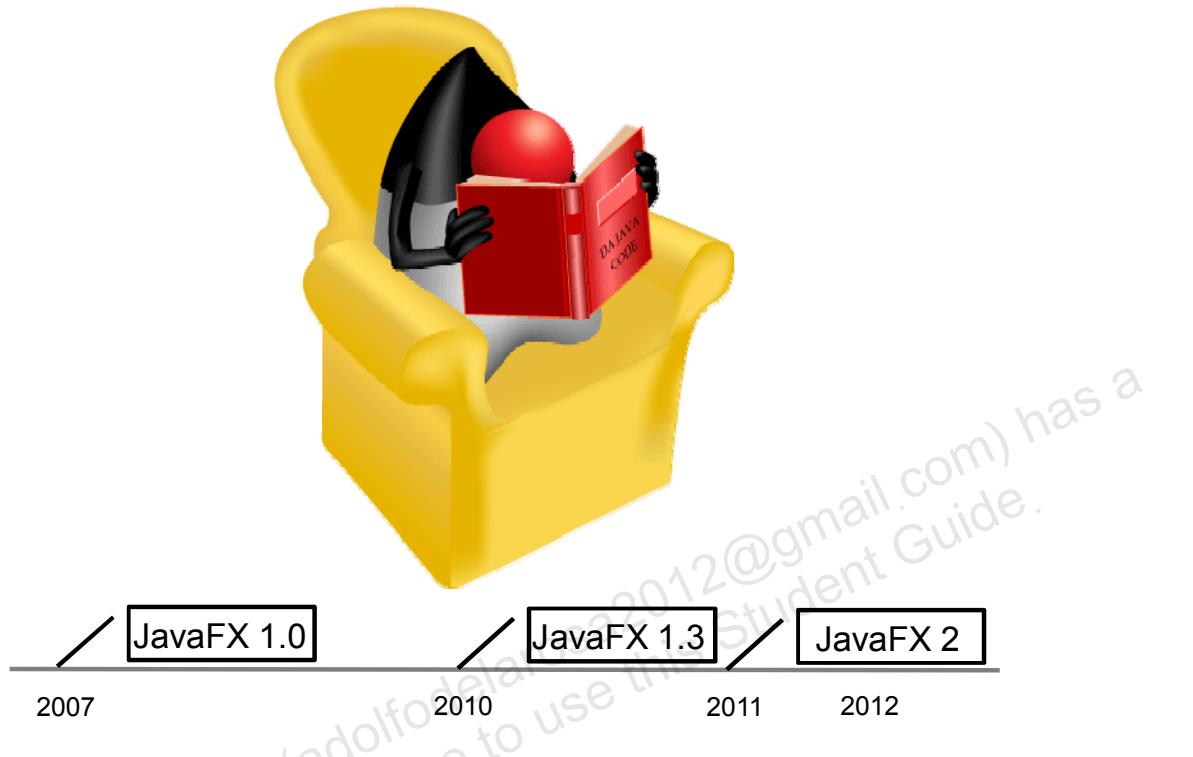
Adolfo De+la+Rosa (adolfodelarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.

JavaFX History and Architecture

ORACLE

Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

The Story of JavaFX



ORACLE

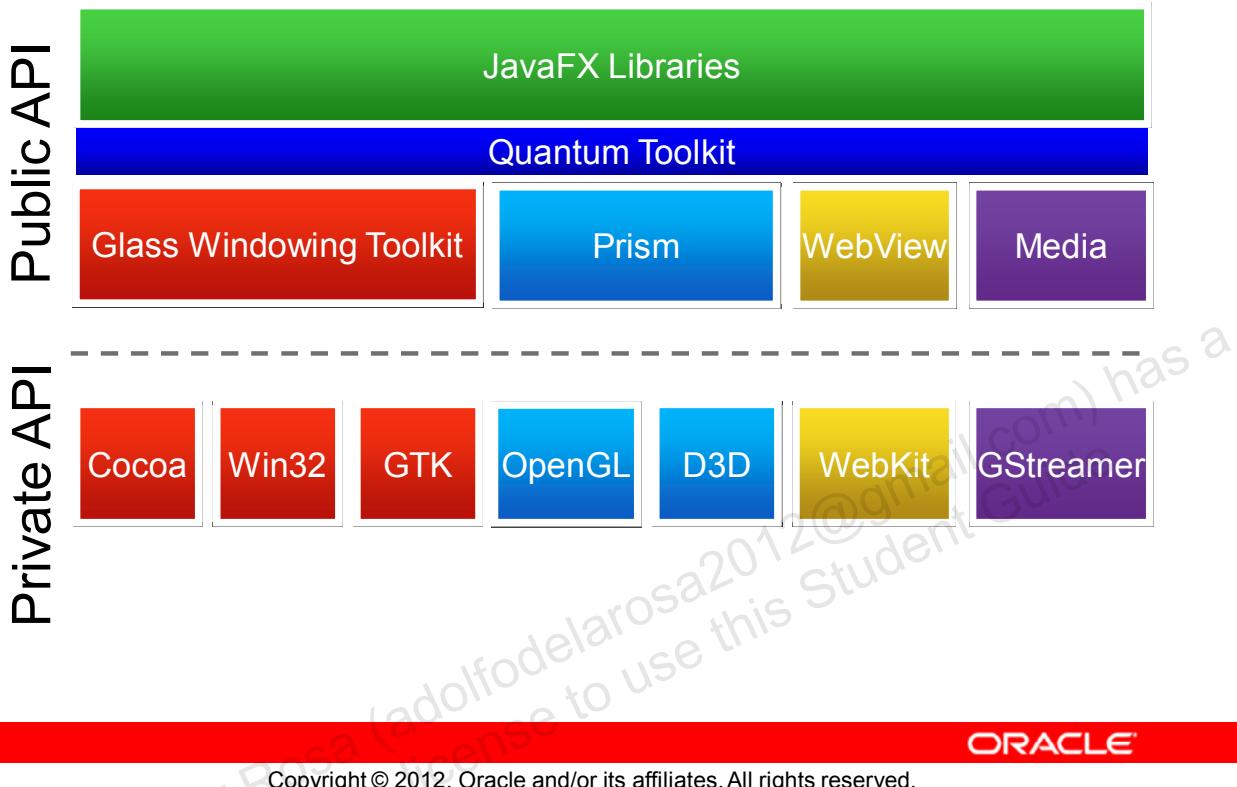
Copyright© 2012, Oracle and/or its affiliates. All rights reserved.

JavaFX started life as F3 (form follows function), which was the brainchild of Chris Oliver when he worked for a company named SeeBeyond. SeeBeyond was acquired by Sun Microsystems, who renamed F3 to JavaFX, and announced it at JavaOne 2007. Chris joined Sun during the acquisition and continued to lead the development of JavaFX. The first version of JavaFX Script was an interpreted language and was considered a prototype of the compiled JavaFX Script language.

At the JavaOne 2007 conference, Sun Microsystems introduced the JavaFX platform to help content developers and application developers to create content-rich applications for mobile devices, desktops, televisions, and other consumer devices. The initial offering consisted of the JavaFX Mobile platform and the JavaFX Script language. Multiple public releases were delivered after the initial announcement; the 1.3 version was released in April 2010.

After Oracle's acquisition of Sun Microsystems, Oracle announced during the JavaOne 2010 conference that support for the JavaFX Script language would be discontinued. However, it was also announced that the JavaFX Script APIs would be ported to Java and would be released as part of the JavaFX 2 platform. This announcement meant that the JavaFX capabilities would become available to all Java developers, without the need for them to learn a new scripting language. With this announcement, Oracle committed to making JavaFX the premier environment for rich client applications.

JavaFX Architecture and APIs



The diagram describes each component and how the parts interconnect. Below the JavaFX public APIs lies the engine that runs your JavaFX code. It is composed of subcomponents that include the new JavaFX high performance graphics engine, called Prism; the new small and efficient windowing system, called Glass; a media engine, and a web engine. In addition, there are several private APIs available through JavaFX, and although these components are not exposed publicly, their descriptions can help you to better understand what runs a JavaFX application.

Prism processes render jobs. It can run on both hardware and software renderers, including 3-D. It is responsible for rasterization and rendering of JavaFX scenes. The following multiple render paths are possible based on the device being used:

- DirectX 9 on Windows XP and Windows Vista
- DirectX 11 on Windows 7
- OpenGL on Mac, Linux, Embedded
- Java2D when hardware acceleration is not possible

The fully hardware-accelerated path is used when possible, but when it is not available, the Java2D render path is used because the Java2D render path is already distributed in all of the Java Runtime Environments (JREs). This is particularly important when handling 3-D scenes. However, performance is better when the hardware render paths are used.

Quantum Toolkit ties Prism and Glass Windowing Toolkit together and makes them available to the JavaFX layer above them in the stack. It also manages the threading rules related to rendering versus events handling.

The Glass Windowing Toolkit is the lowest level framework for the JavaFX 2 graphics stack. Its main responsibility is to provide native operating services, such as managing the windows, timers, and surfaces. It serves as the platform-dependent layer that connects the JavaFX platform to the native operating system.

The Glass toolkit is also responsible for managing the event queue. Unlike the Abstract Window Toolkit (AWT), which manages its own event queue, the Glass toolkit uses the native operating system's event queue functionality to schedule thread usage. Also unlike AWT, the Glass toolkit runs on the same thread as the JavaFX application. In AWT, the native half of AWT runs on one thread and the Java level runs on another thread. This introduces a lot of issues, many of which are resolved in JavaFX by using the single JavaFX application thread approach.

AWT and Glass

AWT	Glass Windowing Toolkit
Manages its own event queue	Uses native operating system's event queues
Multi-threaded approach (Native half of AWT runs on one thread and the Java level runs on another thread)	Single-threaded approach (runs on the same thread as the JavaFX application)

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Adolfo De+la+Rosa (adolfodelarosa2012@gmail.com) has a
non-transferable license to use this Student Guide.