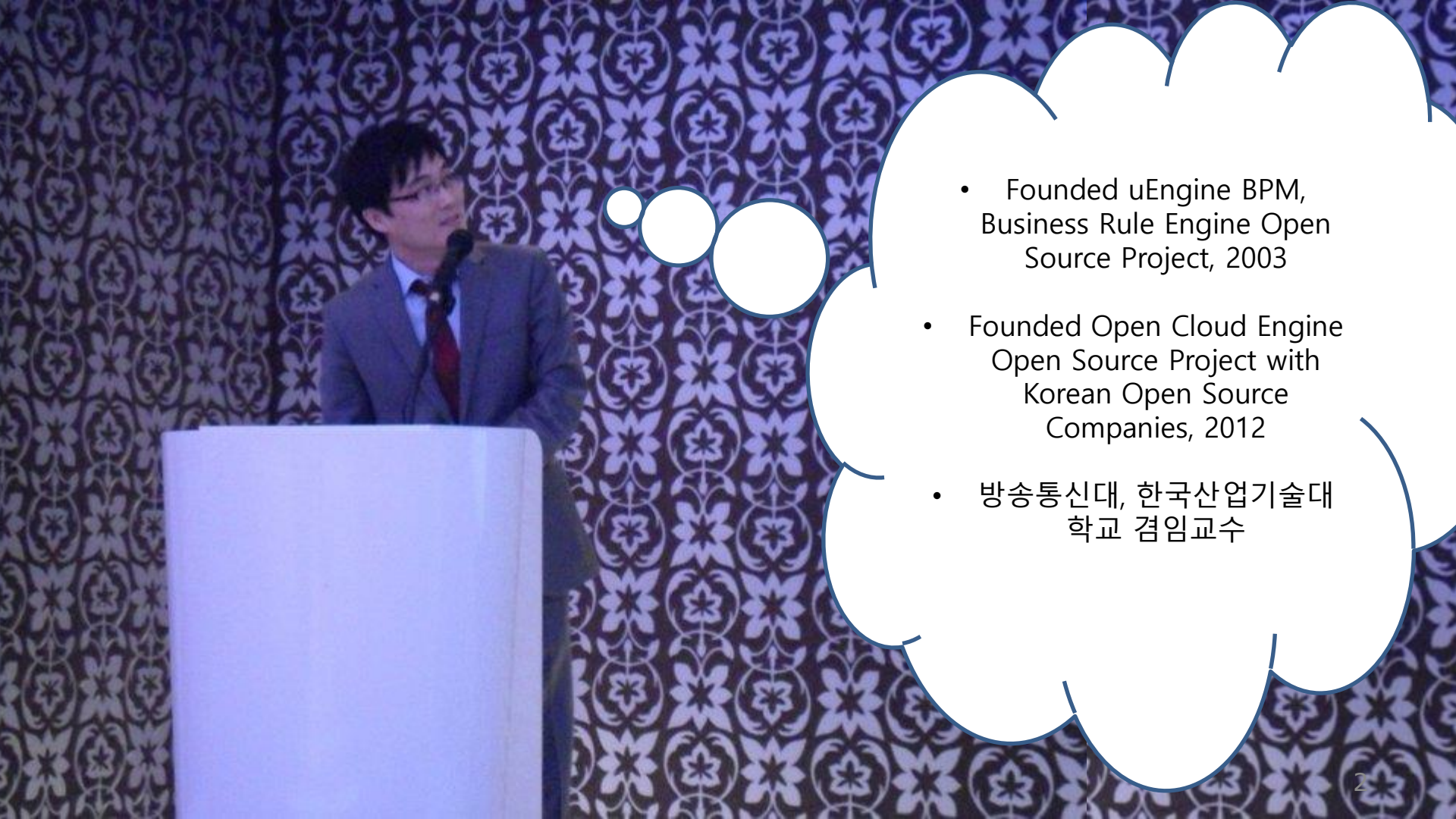
Business Analysis Project: Online Auto Insurance - Realization Part

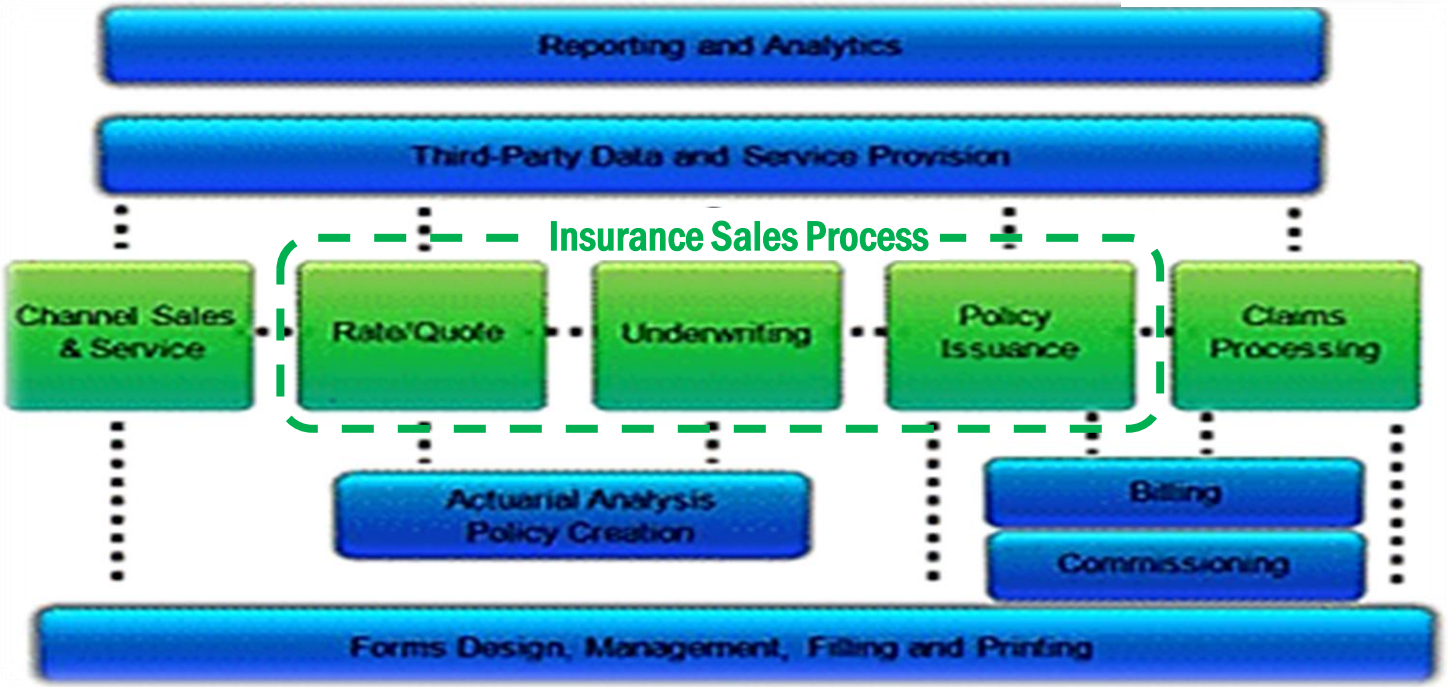
Jinyoung Jang
Jyjang at uEngine.org
2015



- Founded uEngine BPM, Business Rule Engine Open Source Project, 2003
- Founded Open Cloud Engine Open Source Project with Korean Open Source Companies, 2012
- 방송통신대, 한국산업기술대 학교 겸임교수

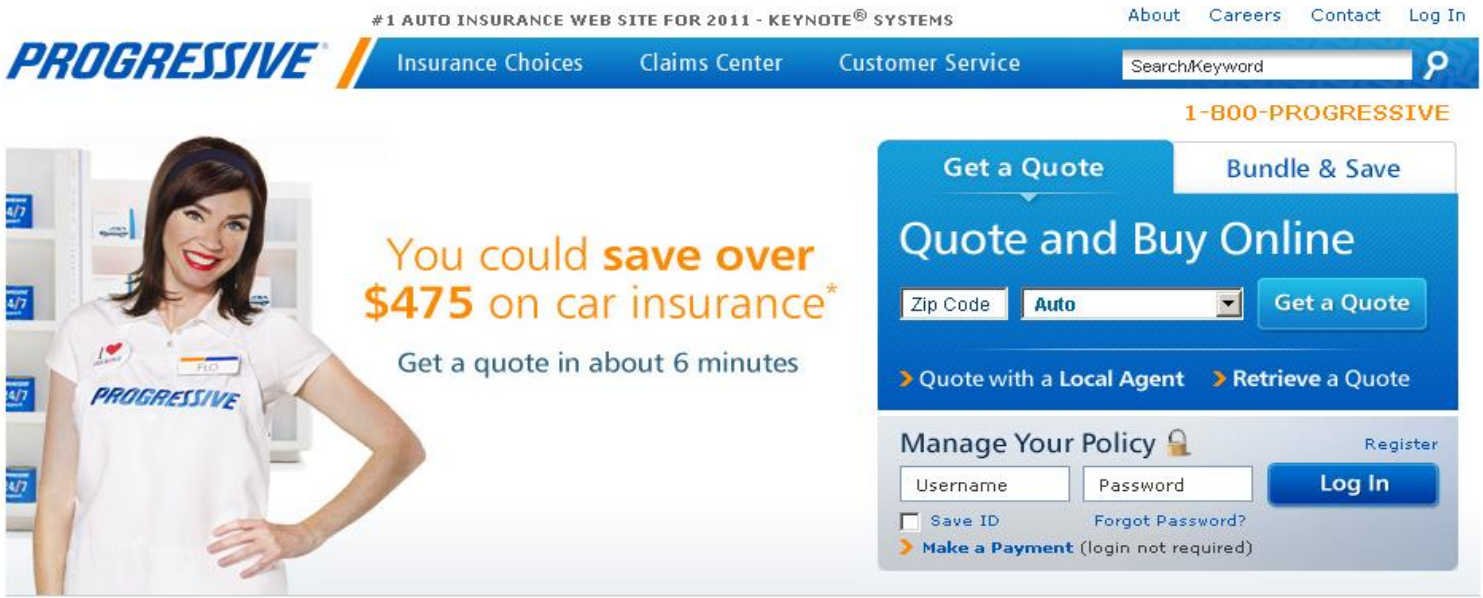
Remind: Problem Description

Auto Insurance Value Chain



Microsoft, Insurance Value Chain Integration Factory, 2007

Online Auto Insurance

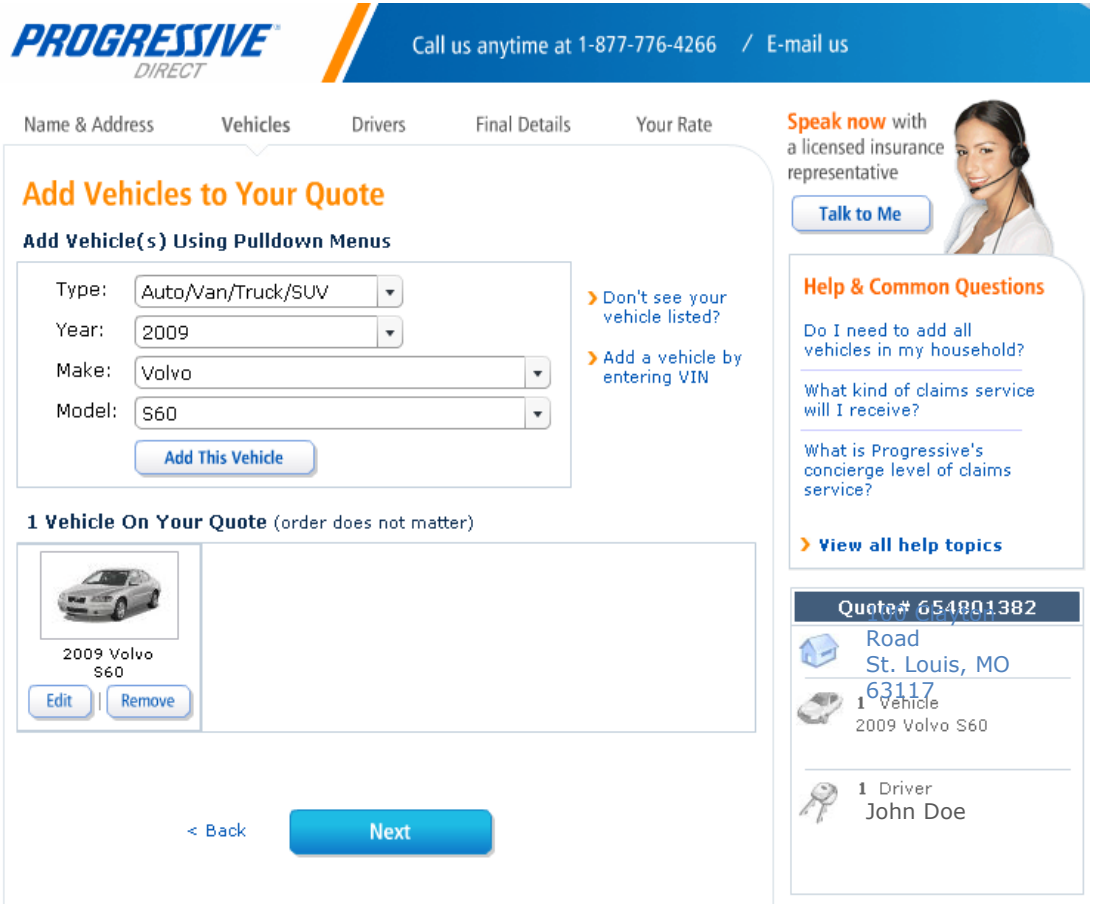


Auto Insurance Online Sales Process

1. Ask the customer to provide her demographic information such as name, address and birth date.
2. Check if the customer is already registered. If not, register the customer.
3. Get consent of the customer for using private information.
4. If customer refuses to provide private information, terminate the process.
5. Otherwise get the customer's credit history from the Credit Bureau and compute the credit rating.

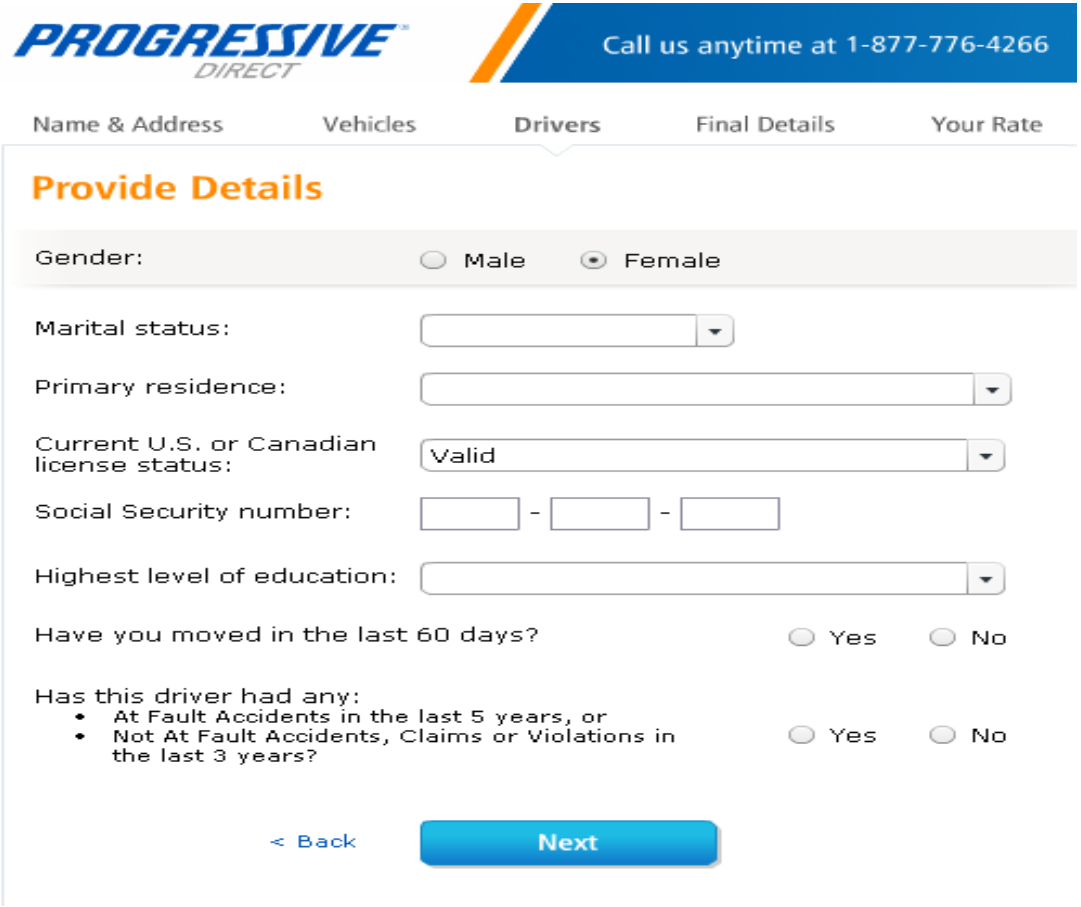
Auto Insurance Online Sales Process

6. Get from Dept of Motor Vehicles the customer's driver license number and the VIN, type, make, model and year for each of her owned vehicles.
7. Terminate the process if the customer has no record with the DMV.
8. Show the customer's owned vehicles and ask her to select the one to insure.
9. If the customer doesn't see her vehicle listed, allow her to add a vehicle entering VIN.
10. Create a quote and show it to the customer.



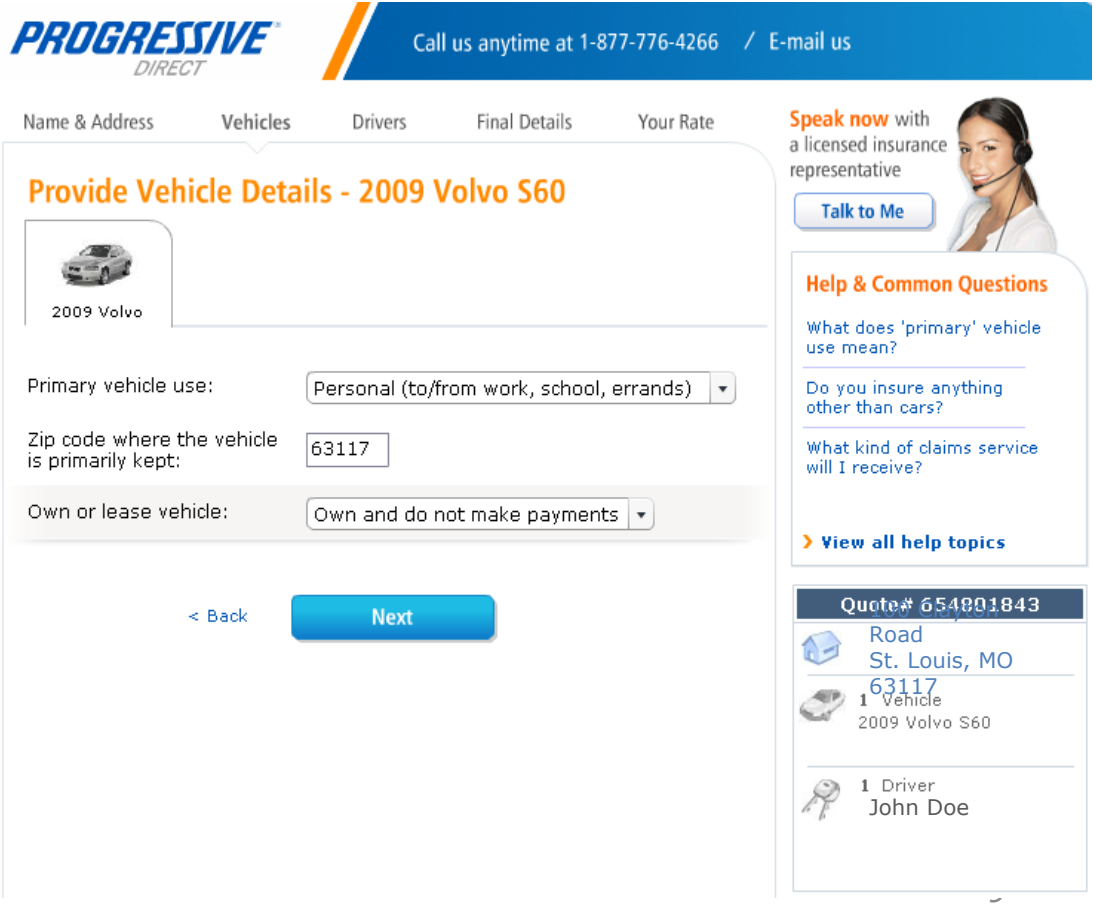
Auto Insurance Online Sales Process

11. Ask the customer to provide details about herself including gender, marital status, primary residence, driver license status, social security number, highest level of education, move history, and accident history.



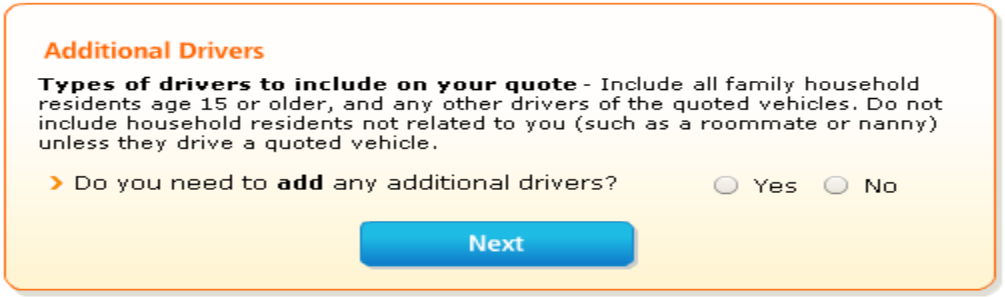
Auto Insurance Online Sales Process

12. Ask the customer to provide further information about the vehicle including primary use, zip code where the vehicle is primarily kept, and type of ownership.



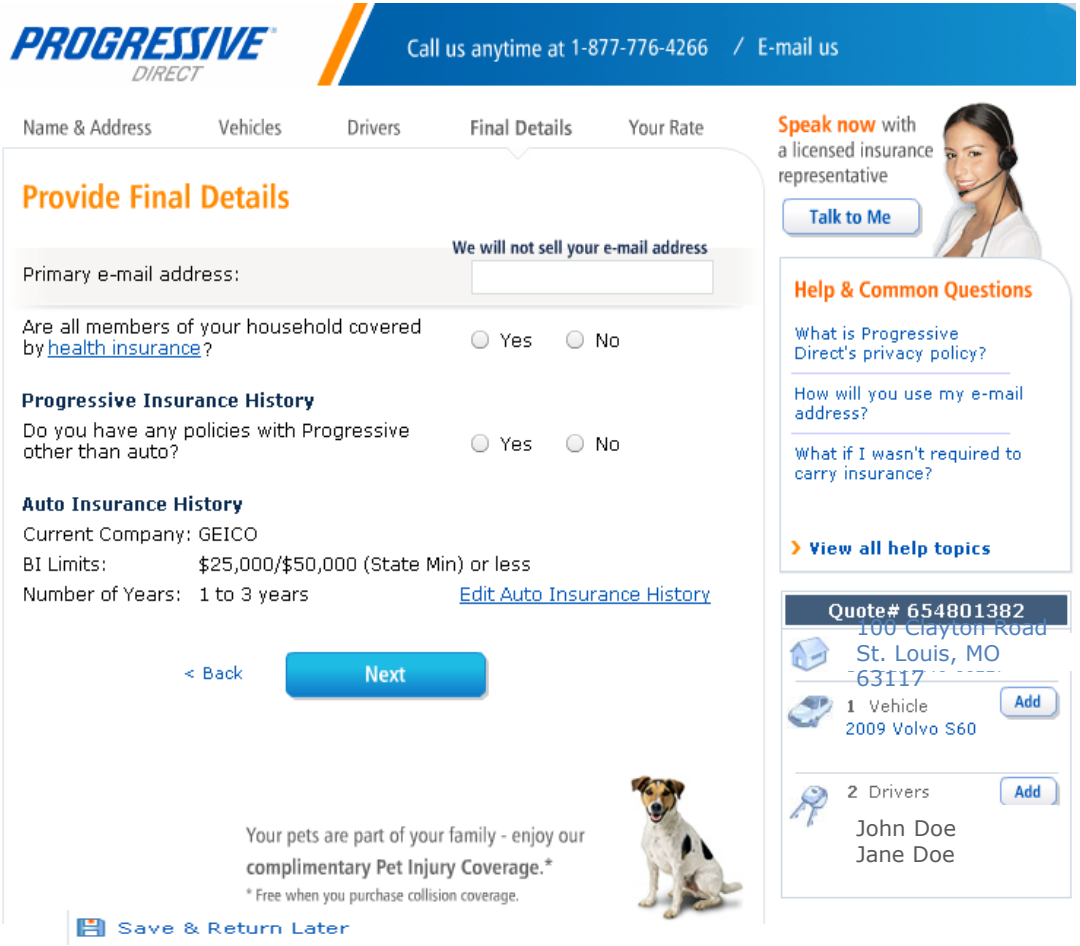
Auto Insurance Online Sales Process

13. Ask the customer if she wants to add additional drivers for the insured vehicle. If so, ask the customer to provide information about each additional driver including his name, social security number, gender, birth date, marital status, driver license number, and relationship to the customer.
14. Evaluate the eligibility of the customer for the applied policy.
15. If the application must be rejected, notify the result and terminate the process.



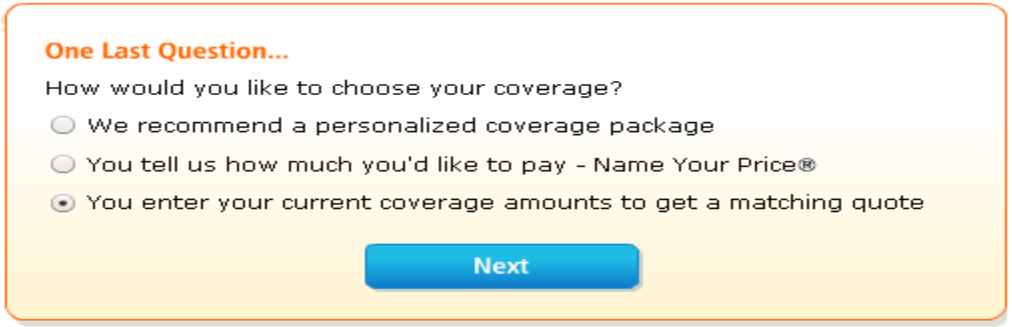
Auto Insurance Online Sales Process

16. Ask the customer to provide final details about her insurance status including her health insurance, and any insurance policies she holds with Progressive.
17. Get the customer's insurance history from the National Insurance Association.
18. Show the auto insurance policy she currently holds and allow her to edit it.



Auto Insurance Online Sales Process

19. Ask the customer to decide whether to accept a recommended personalized coverage package, or to suggest the price to match, or to set the coverage for herself for each loss event covered.



Auto Insurance Online Sales Process

20. Determine the 6-month coverage for each loss event covered and calculate the 6-month premium.

Get a Matching Quote Close

Choose Your Policy Level Coverage

Bodily Injury & Property Damage Liability	$25,000/$50,000/$10,000
Uninsured Motorist Bodily Injury	$25,000/$50,000
Underinsured Motorist Bodily Injury	No coverage
Medical Payments	No coverage

Continue

Get a Matching Quote Close

Choose Your 2009 Volvo S60 Coverage

Comprehensive	$500 deductible
Collision	$500 deductible
Rental Car Reimbursement	$30/day ($900 maximum)
Roadside Assistance	Coverage selected

Continue

Auto Insurance Online Sales Process

21. Ask the customer if she wants to buy the policy now, change the coverage options selected, or save the quote for a later visit.
22. If the customer wants to proceed to purchase, ask the date to start the coverage.

PROGRESSIVE DIRECT

Call us anytime at 1-877-776-4266 / E-mail us

Name & Address | Vehicles | Drivers | Final Details | Your Rate

Choose a Personalized Package

Our Choice package (6-monthly)

$113.53 — 6 Monthly Payments

$640 — Paid in full (save $46)

Basic $45.18 | Recommended $122.86

or **Name Your Price®**

Tell us what you want to pay

$ [ ] per month

See Your Package

6-month Policy Coverages

Bodily Injury & Property Damage Liability	Edit	$25,000 per person $50,000 per accident $10,000 property damage	$258
Uninsured Motorist Bodily Injury	Edit	$25,000 per person $50,000 per accident	$10
Underinsured Motorist Bodily Injury	Edit	No coverage	$0
Medical Payments	Edit	No coverage	$0

Speak now with a licensed insurance representative

Talk to Me

Compare Auto Rates

In 3 minutes or less, get additional rates to compare to your Progressive Direct rate quote:

› Get Comparison Rates

Compare Auto Rates | Other Quotes | Home & Rental

Your Quote Summary

Pay in full: $640.00

Policy term: 6 months

Continue to Buy

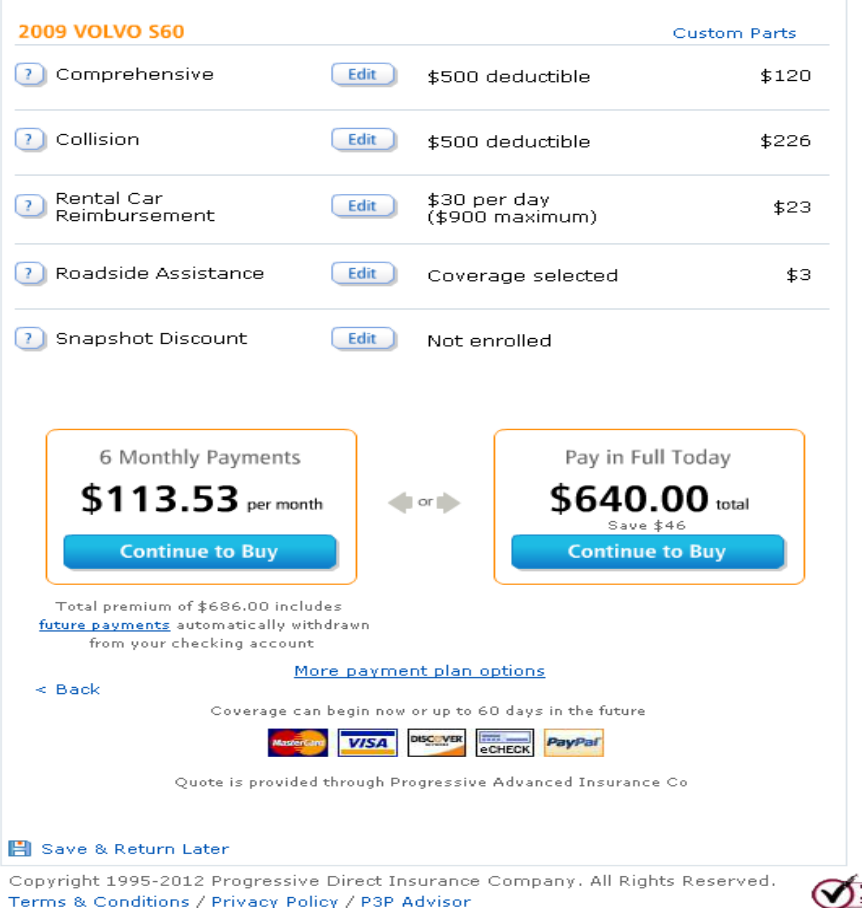
Print Quote Summary

Save & Return Later

Quote# 654801382

Auto Insurance Online Sales Process

23. Ask the customer to choose the bill plan among (i) pay in full today at a discount, (ii) automatic withdrawal from her checking account, or (iii) other options such as online, phone or mail.
24. Ask the customer to provide information needed for payments.
25. Get the payment.
26. Ask the customer sign the application online.
27. Send the signed insurance policy via email.
28. Complete the sales.



Overview of this Course

Day-1: Application Generation

- Generate Entity Classes
- Generate UI Modules

Day-2: Make the Application As A Service

- Expose Services through REST
- Orchestration with BPM

Day-3: Cloud Deployment

- Multi-tenancy
- Deploy to a PaaS (CloudFoundry)

URL for sample project

https://github.com/TheOpenCloudEngine/sw-modeling-tutorial

TheOpenCloudEngine / sw-modeling-tutorial

Unwatch 3

Modeling Driven Development Tutorial using OCE — Edit

5 commits | 1 branch | 0 releases | 1 contributor

branch: master | sw-modeling-tutorial / +

Merge branch 'master' of https://github.com/TheOpenCloudEngine/sw-mod...

admin authored 22 hours ago | latest commit 54259a9d3a

lib	no message	22 hours ago
src/main	first commit	22 hours ago
vpproject	first commit	22 hours ago
README.md	Update README.md	22 hours ago
autoinsurance.ddl	first commit	22 hours ago
pom.xml	first commit	22 hours ago

README.md

Welcome to Model Driven Software Development using OCE and Visual-paradigm

Building and running the tutorial example

```
mvn tomcat7:run-war
```

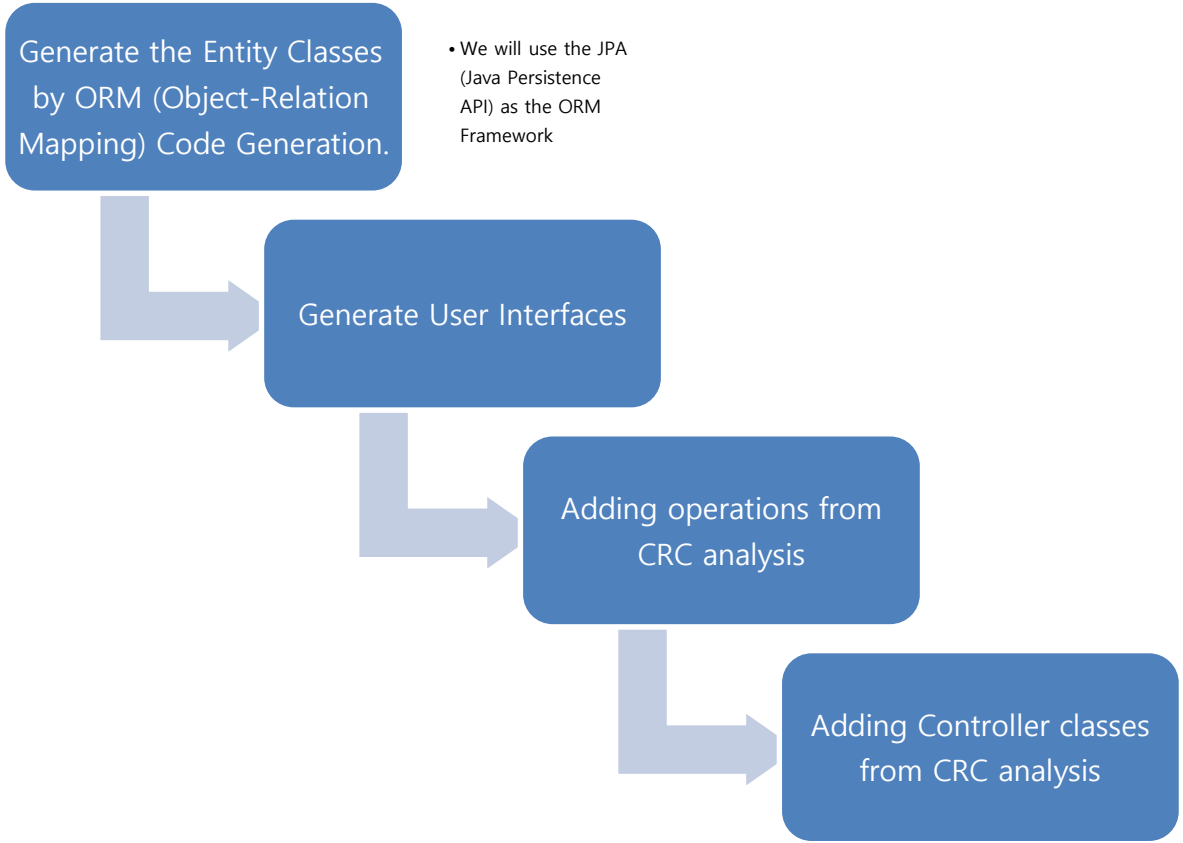
after seeing the last log - INFO: Starting ProtocolHandler ["http-bio-8089"] - then you can now go to following url:

```
http://localhost:8089/auto-insurance/runner.html?className=controllers.NameAndAddressTa
```

Congratulations!

Day-1: Application Generation

Overview



- We will use the JPA (Java Persistence API) as the ORM Framework

Prerequisites

- Apache Maven 3.x
  - Will import following dependencies automatically:
    - Mysql Java Library
    - ORM-related Libraries including JPA

- Integrated Development Environment:
  Intelli-J CE 12 or higher

- Database-related tools:
  Mysql Server, Mysql workbench
  (Mysql Java Library will be automatically imported by maven)

Create a maven project

1. **Create a maven project with Metaworks3 Archetype**
   Just copy and paste following command:

   ```
   mvn archetype:generate ₩
   -DarchetypeGroupId=org.uengine ₩
   -DarchetypeArtifactId=metaworks-sample-archetype ₩
   -DarchetypeVersion=1.1-SNAPSHOT ₩
   -DarchetypeRepository=https://oss.sonatype.org/content/repositories/snapshots
   ```

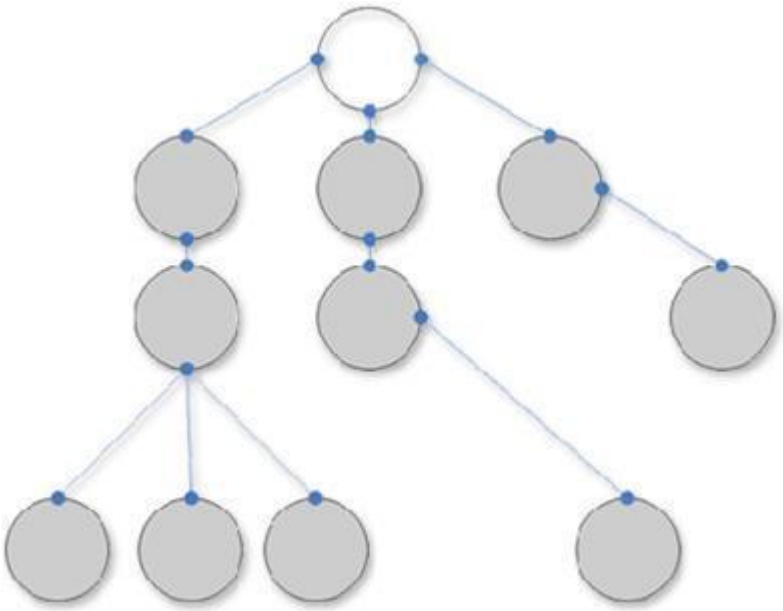
2. **Following the instruction, input these values for the options:**

   - groupId : oai
   - artifiactId : autoinsurance

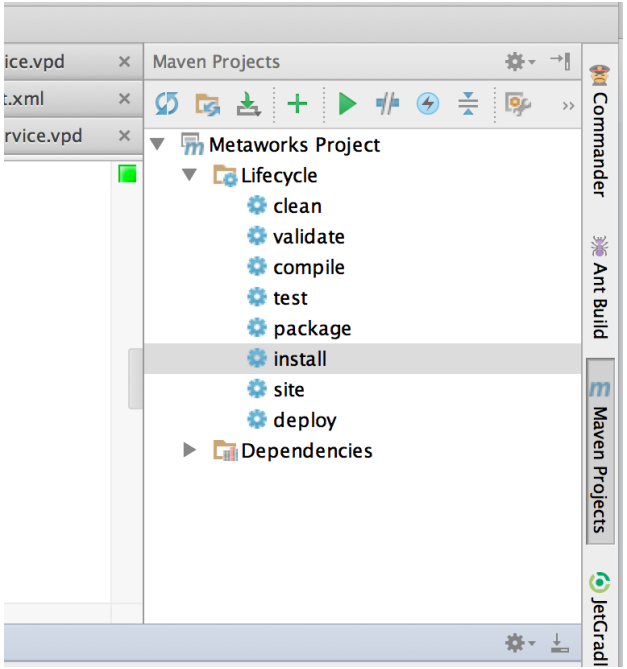
3. **[In Eclipse] Import As Maven Project, [In IntelliJ] Just Open it…**

What is Maven?

- Automatic management of complex dependencies of software components

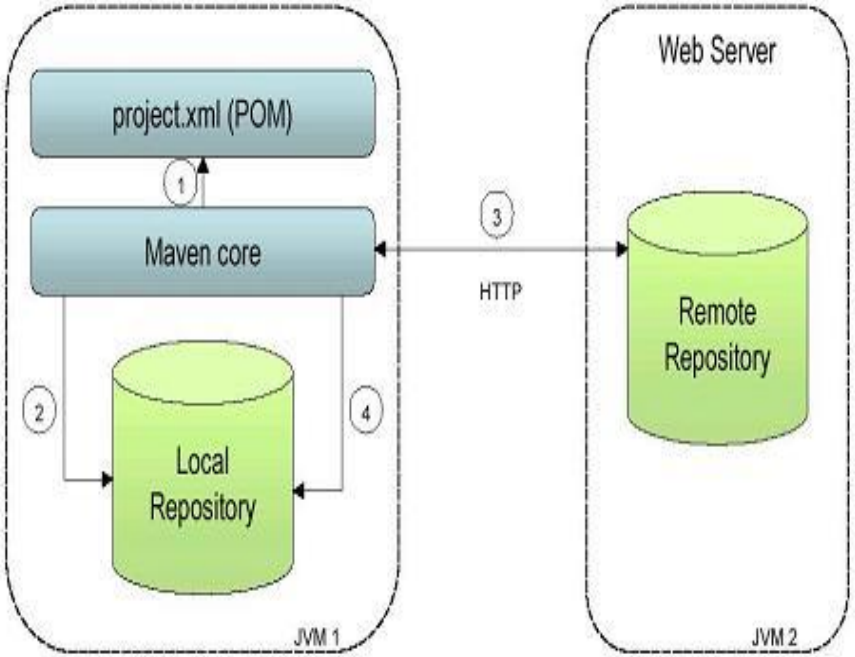


- Maven's default lifecycle of sw component



Maven Repository

- Your components will be managed in your local repository by default.
- If there's no component found in your local, maven will try to import them remotely from the maven remote repository.
- There is a Central Maven Repository and there are so many repositories operated by open source projects
- Also you can build your own (Sonatype's nexus is free and open source)



Basic Maven Commands

- mvn dependency:tree
- mvn install
- mvn tomcat7:run-war

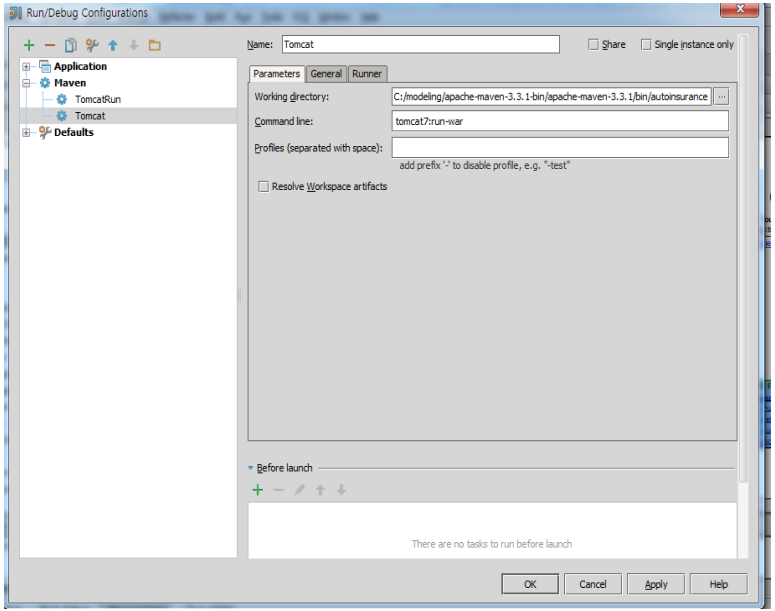
Solution for JVMBind Exception

- Change the tomcat port number specified in the pom.xml:

```
 <plugin>
....
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <version>2.2</version>
                <configuration>
                    <url>http://localhost:8081/user</url>
                    <port>8081</port>
                </configuration>
            </plugin>
```

Tomcat run

- Run > Edit Configuration
- Press “+” button in the Maven group
- Enter “Tomcat Run” something in the Name entry
- Enter “tomcat7:run-war” in the command line
- Press “Ok”
- Press Run button with ‘Tomcat Run’ selected



[!]

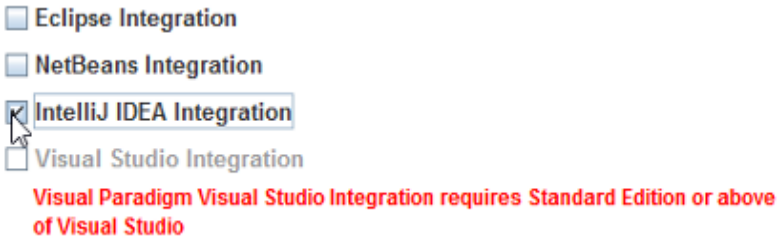
[인텔리J에서 메이븐 명령 실행시] maven.multiModuleProjectDirectory 오류가 나시면

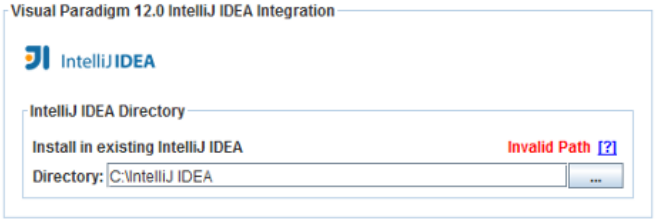
- File > Settings > Maven >> Runner 에서
- VMOptions에 다음을 입력

-Dmaven.multiModuleProjectDirectory=[MAVEN HOME]

IntelliJ and VP integration

1. In Visual Paradigm, select **Windows > Integration > IDE Integration...** from the toolbar. (for some version, **Tools > IDE Integration**)
2. Select IntelliJ IDEA. You can run Visual Paradigm in multiple IDEs. In other words, if you need you can select multiple IDEs here. Click **Next**.
3. Specify the folder path of IntelliJ IDEA. Click **Next** to start copying files to your IDE.





(Currently VP supports only IntelliJ 12 or under)

IntelliJ and VP integration

IntelliJ IDEA File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help Jinyoung Jang

Customer.java – [autoinsurance] – autoinsurance – [~/IdeaProjects/OAI/autoinsurance]

Unnamed

autoinsurance › src › main › java › autoinsurance › Customer

Project Explorer Project Explorer

Model Explorer | Logical View

- autoinsurance
  - autoinsurance
    - Coverage_Item
    - Coverage_Item_Option
    - Coverage_Item_Option_Selected
    - Customer
    - Customer
    - Glossary
    - Impact Analysis
    - Insurance_Policy
    - Insurance_Policy_Payment
    - Insurance_Product
    - Insurance_Quote
    - Insured_Driver
    - Policyholder
    - ui
      - ui
    - Use Cases
    - Vehicle
  - Auto Insurance Online Sales Process
  - coverage_item
  - Coverage_Item
  - coverage_item_code
  - coverage_item_name

NameAndAddressTab.java | VehiclesTab.java | CustomerUI.java | Customer.java

```
@DiscriminatorValue("Customer")
public class Customer implements Serializable {
    public Customer() {
    }

    @Column(name="ID", nullable=false)
    @Id
    @GeneratedValue(generator="AUTOINSURANCE_CUSTOMER_CUSTOMER_ID_GENERATOR")
    @org.hibernate.annotations.GenericGenerator(name="AUTOINSURANCE_CUSTOMER_CUSTOMER_ID_GENERATOR", strateg
    private String customer_ID;

    @Column(name="Customer_social_security_number", nullable=true, length=255)
    private String customer_social_security_number;

    @Column(name="Customer_first_name", nullable=true, length=255)
    private String customer_first_name;

    @Column(name="Customer_middle_initial", nullable=true, length=255)
    private String customer_middle_initial;

    @Column(name="Customer_last_name", nullable=true, length=255)
    private String customer_last_name;

    @Column(name="Customer_suffix", nullable=true, length=255)
    private String customer_suffix;

    @Column(name="Customer_mailing_address", nullable=true, length=255)
    private String customer_mailing_address;

    @Column(name="Customer_apt_unit_number", nullable=true, length=255)
    private String customer_apt_unit_number;

    @Column(name="Customer_city", nullable=true, length=255)
```

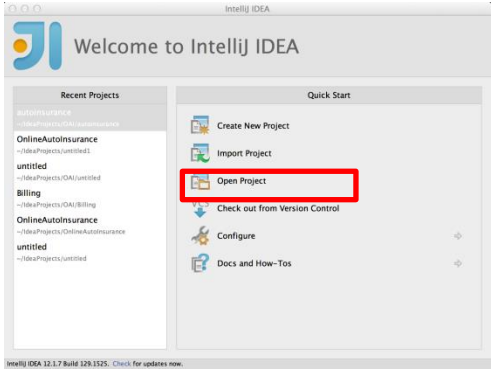
Message Pane Message Pane

6: TODO | Message Pane | Event Log

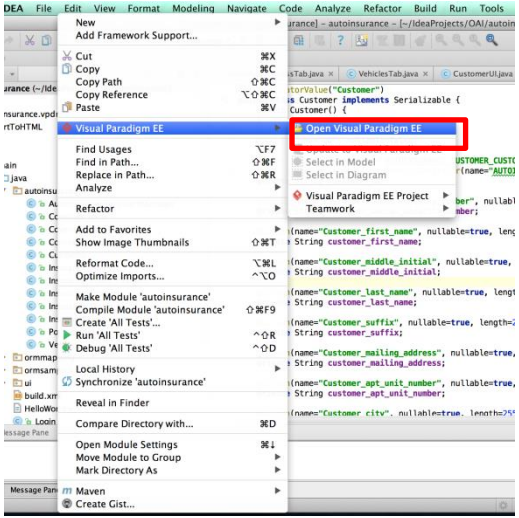
50:17 | LF | UTF-8 | 194M of 666M

Using Visual Paradigm In IntellJ

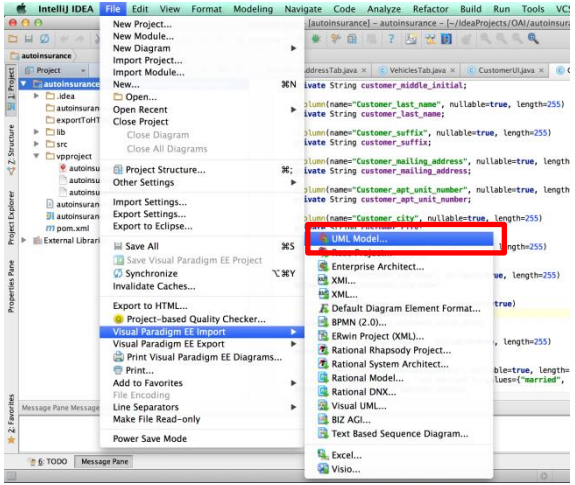
1. Open the created maven project



2. Opening a VP in IntellJ



3. Import existing model



Importing Existing Model

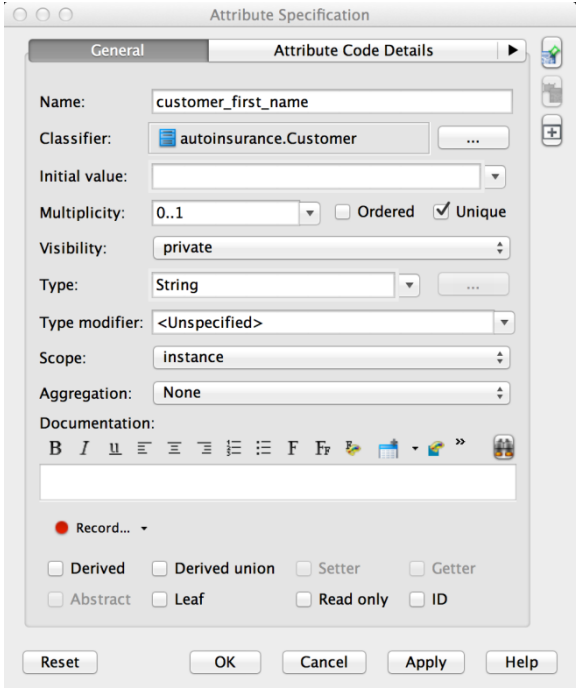
- File > Visual Paradigm EE Import > UML Model
- Choose the existing .vpp file

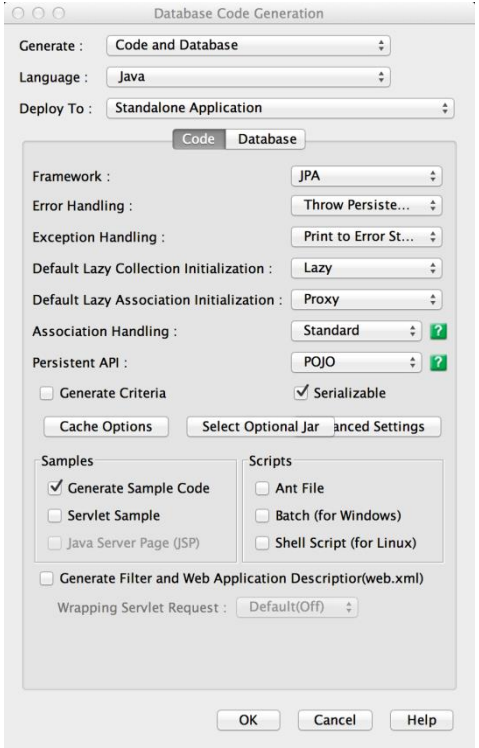
Entity-Relationship Diagram

Generate ERD and Entity Classes (ORM)

- Open the Class Diagram in the Project Explorer > Diagram Navigator > ClassDiagram
- Set the type of attributes
  - String for most types
  - boolean for gender
  - int for options
  - double for money
  - java.util.Date for Dates, timestamp
- Modeling > ORM > Generate Code
- Database Code Generation Options:
  - Framework: JPA
  - Persistence API: POJO

- Be careful:
  - All attributes must have its type
  - All classes should be belong to any package





Generated Files

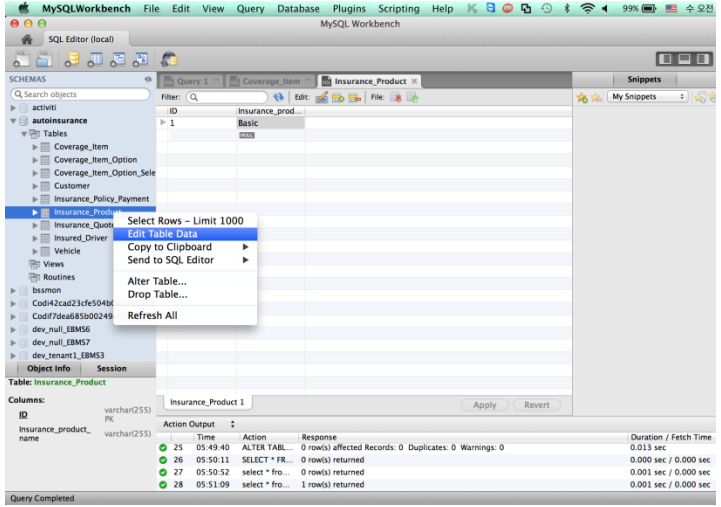
Autoinsurance.cfg.xml – [autoinsurance] – autoinsurance – [~/IdeaProjects/OAI/autoinsurance]

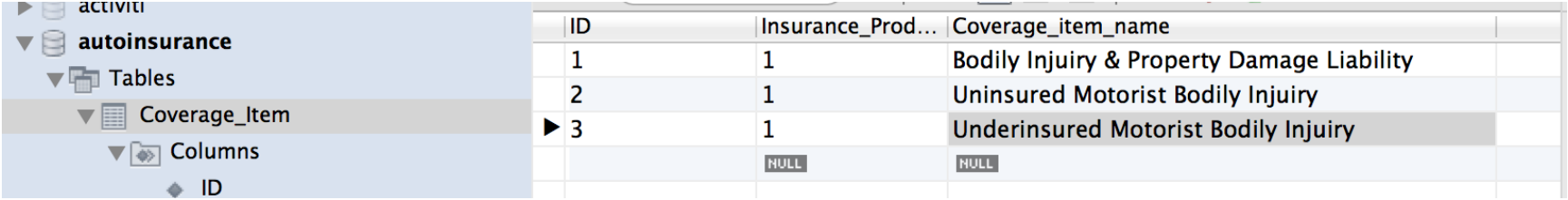
Project

- autoinsurance (~/IdeaProjects/OAI/autoinsurance)
  - .idea
  - autoinsurance.vpdm
  - exportToHTML
  - lib
  - src
    - main
      - java
        - autoinsurance
        - ormmapping
          - Autoinsurance.cfg.xml
          - Autoinsurance.erd.xml
          - database.cfg.xml
        - ormsamples
        - services
        - ui
        - build.xml
        - Capability
        - HelloWorld.ejs
        - Login
        - Main
      - resources
      - webapp
  - vpproject
    - autoinsurance.vpp
    - autoinsurance.vpp~3
    - autoinsurance.vpp~4
  - autoinsurance.ddl
  - autoinsurance.iml
  - pom.xml

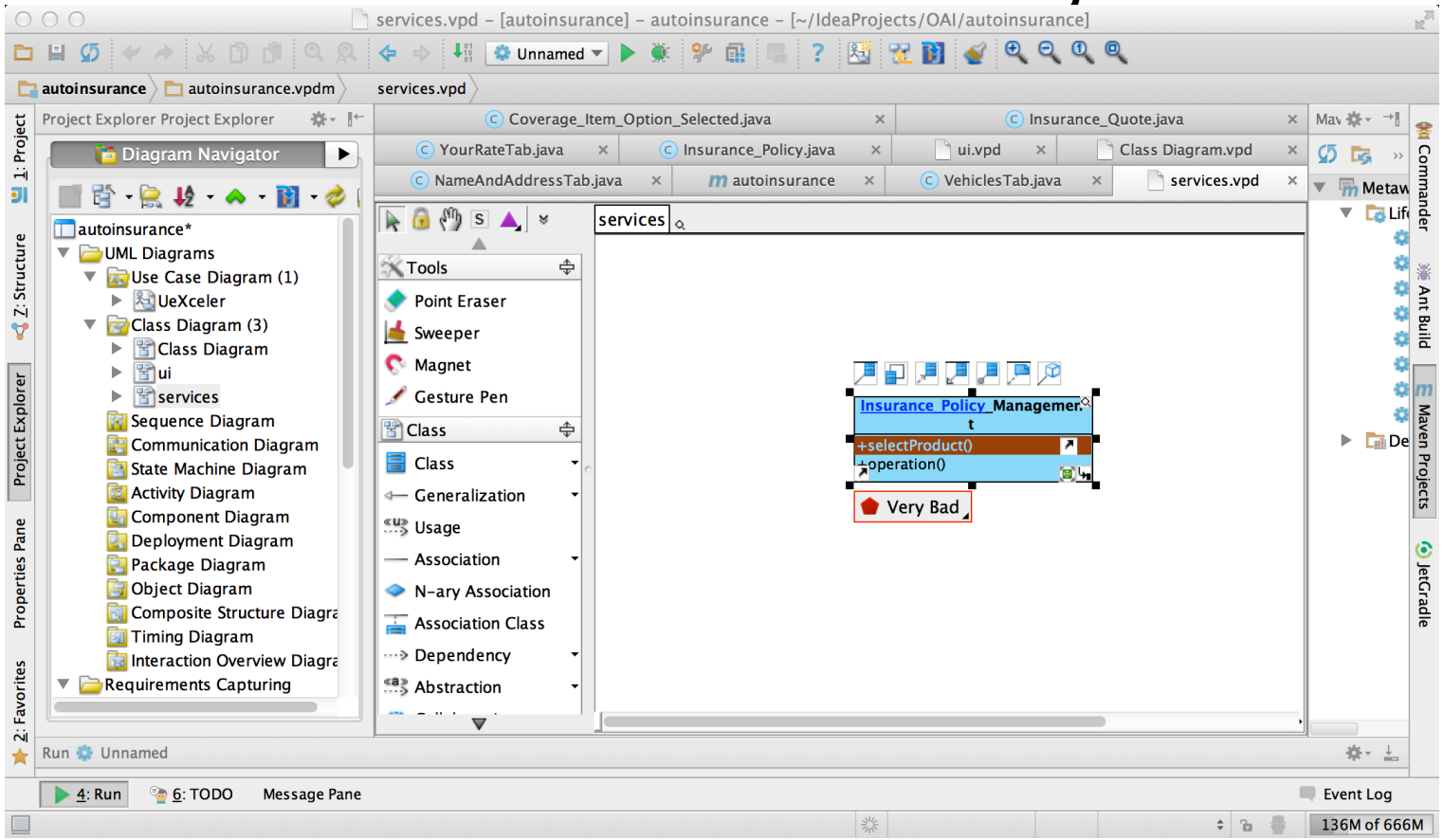
```
<?xml version="1.0" encoding="utf-8" ?>
<!--
Licensee: uengine
License Type: Purchased
-->
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
    <!-- properties -->
        <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/autoinsurance</property>
        <property name="connection.username">root</property>
        <property name="connection.password"></property>
        <property name="hibernate.connection.provider_class">org.hibernate.service.jdbc.connecti
        <property name="hibernate.c3p0.acquire_increment">1</property>
        <property name="hibernate.c3p0.idle_test_period">0</property>
        <property name="hibernate.c3p0.min_size">1</property>
        <property name="hibernate.c3p0.max_size">15</property>
        <property name="hibernate.c3p0.timeout">0</property>
        <property name="show_sql">false</property>
        <property name="query.substitutions">true 1, false 0, yes 'Y', no 'N'</property>
        <!-- mapping files -->
    </session-factory>
</hibernate-configuration>
```

Inserting Product Master Data



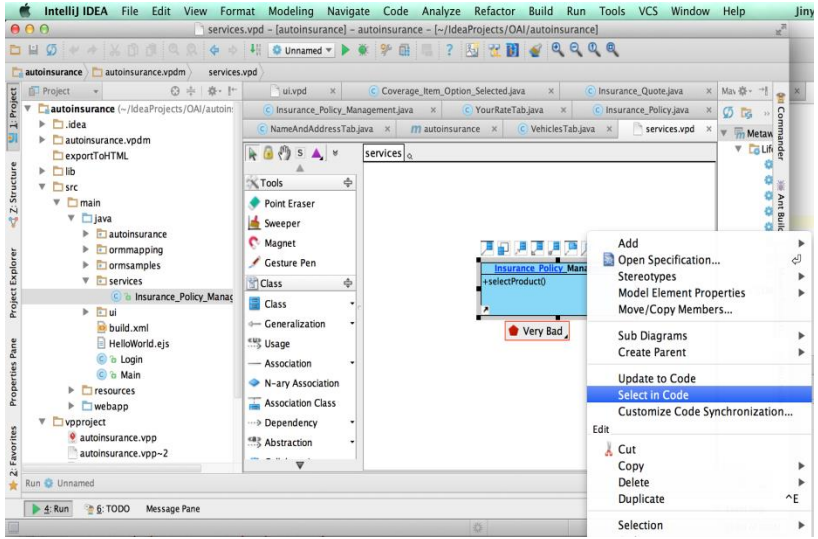


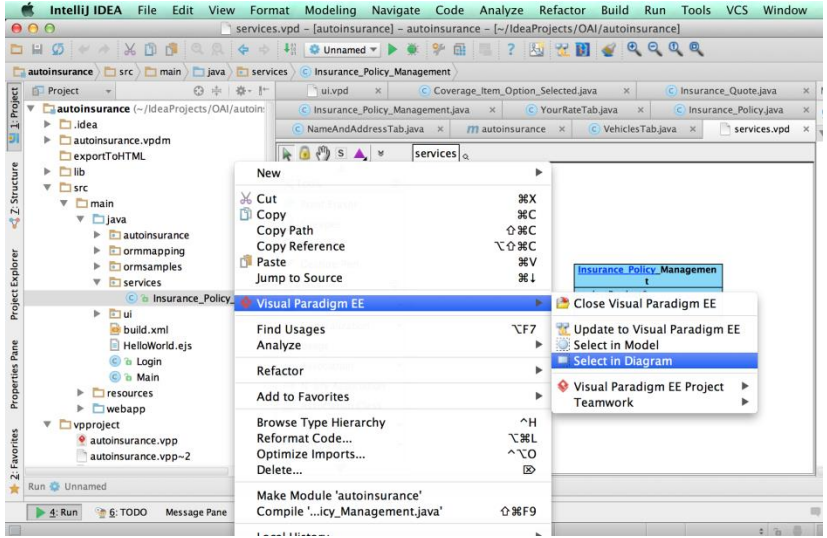
Create a Class Diagram and Generate Java Code in IntelliJ-VP integration



Navigating between Model and Code

Jumping from diagram to Code:
Right-click on the class > Select in Code

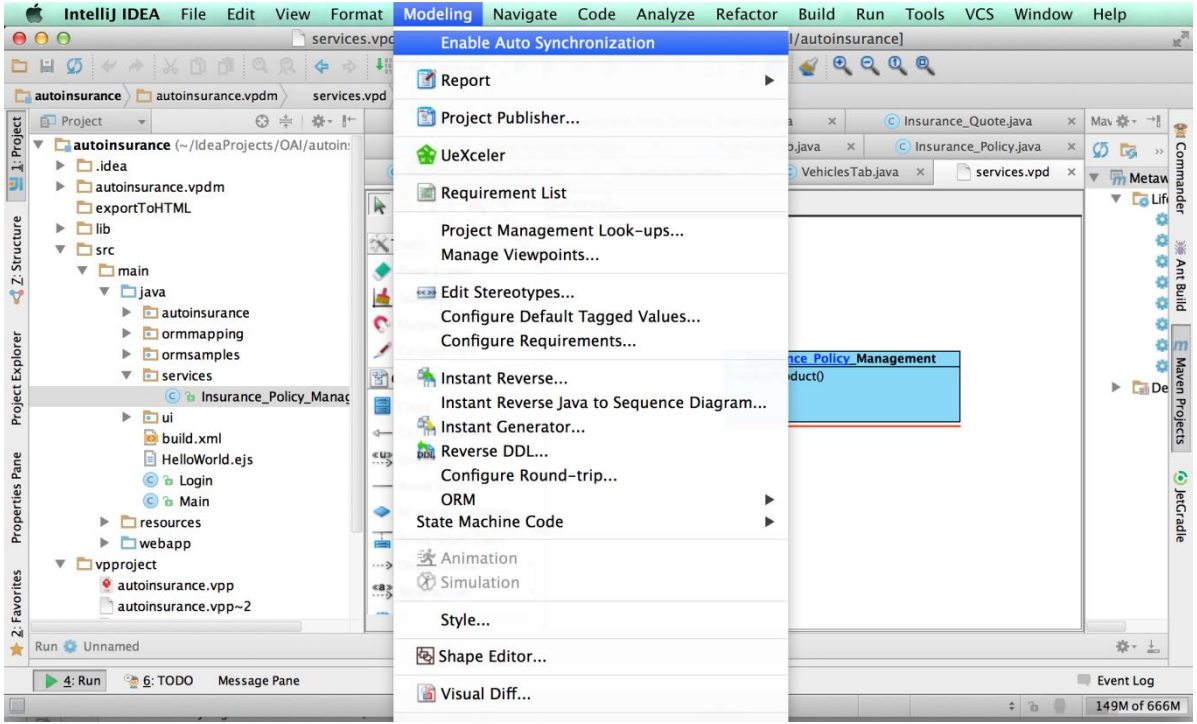




Jumping from Code to Diagram:
Right-click on the source file > Visual Paradigm EE > Select in Diagram

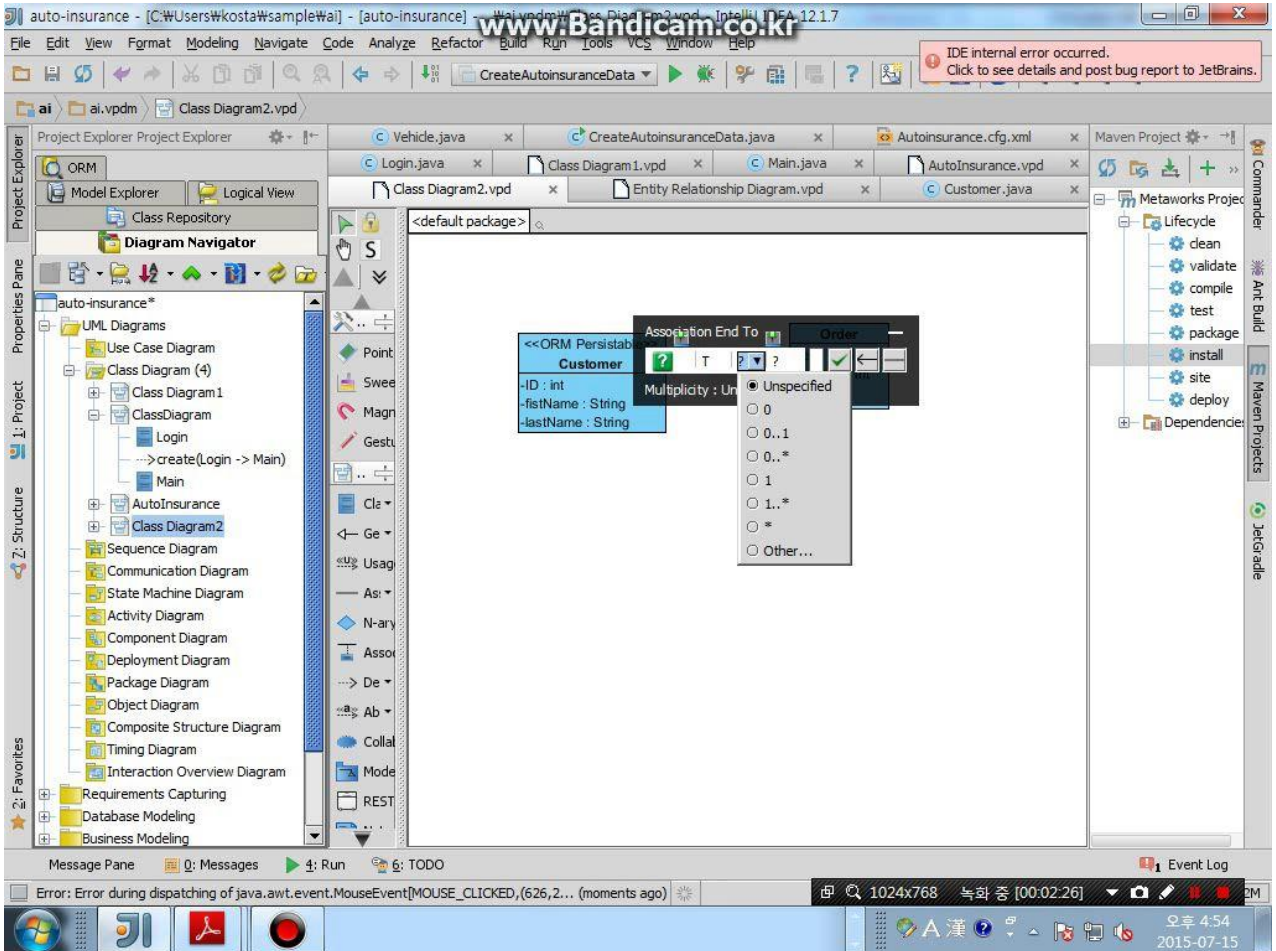
Automatic Code Synchronization

- Modeling > Enable Auto Synchronization

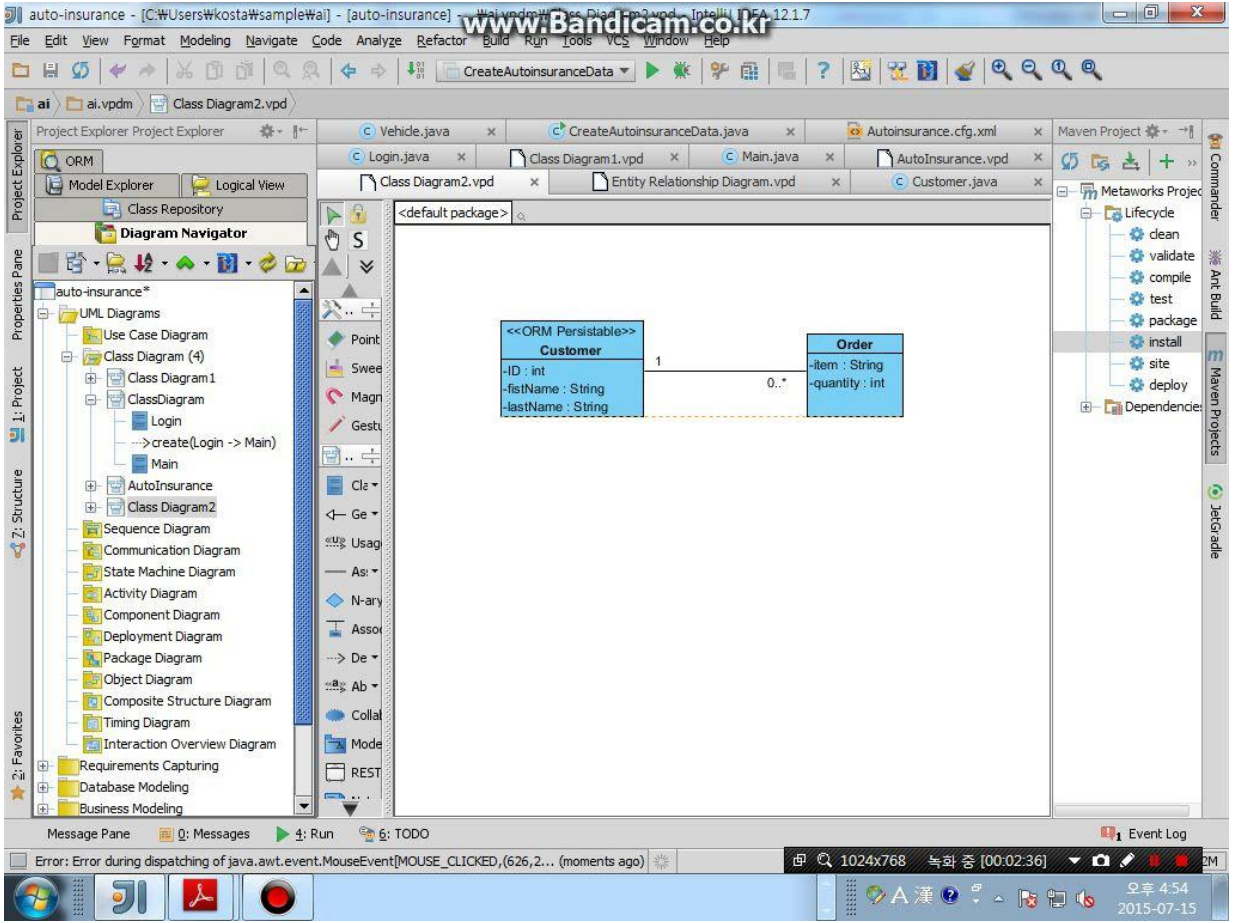


EXERCISE : CREATING ORM AND DATABASE CONNECTION

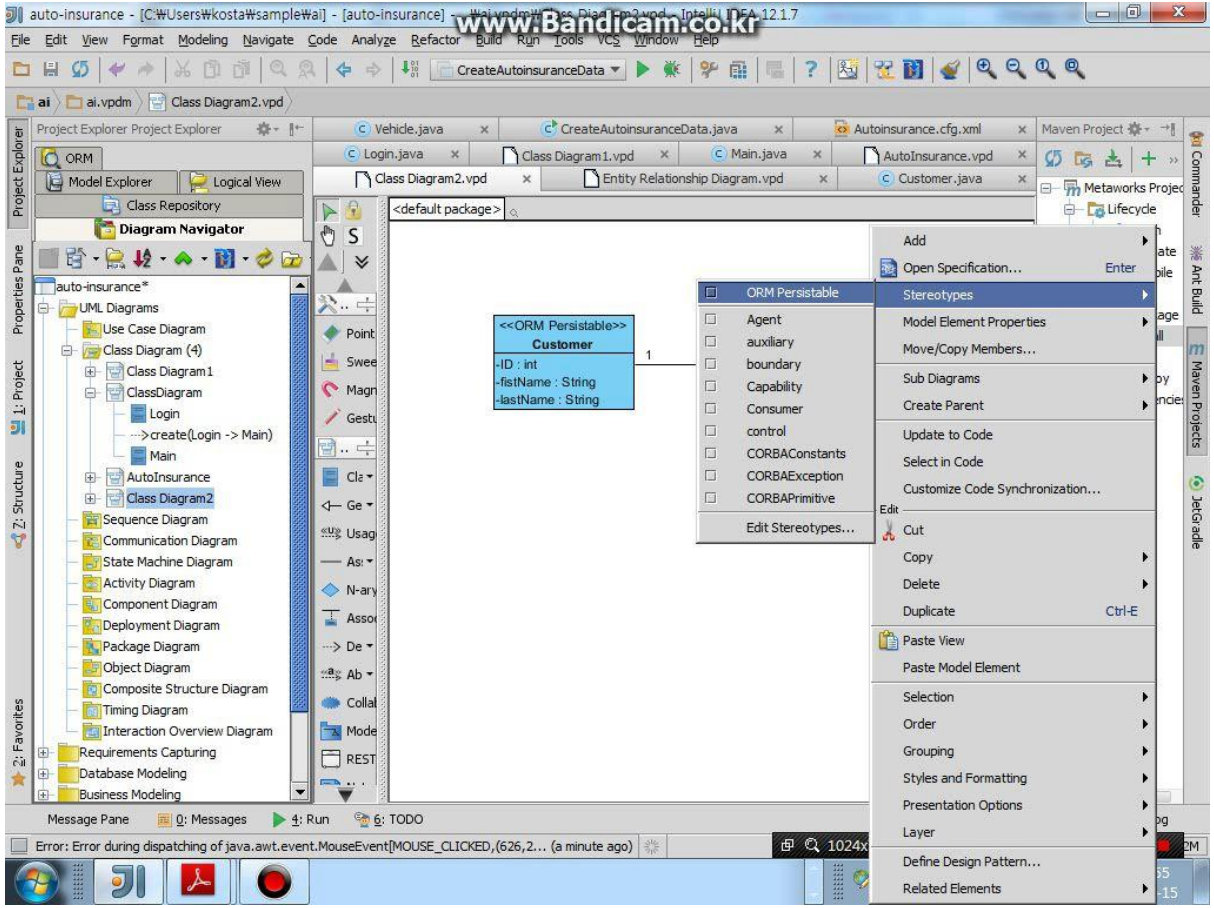
Create a new class diagram



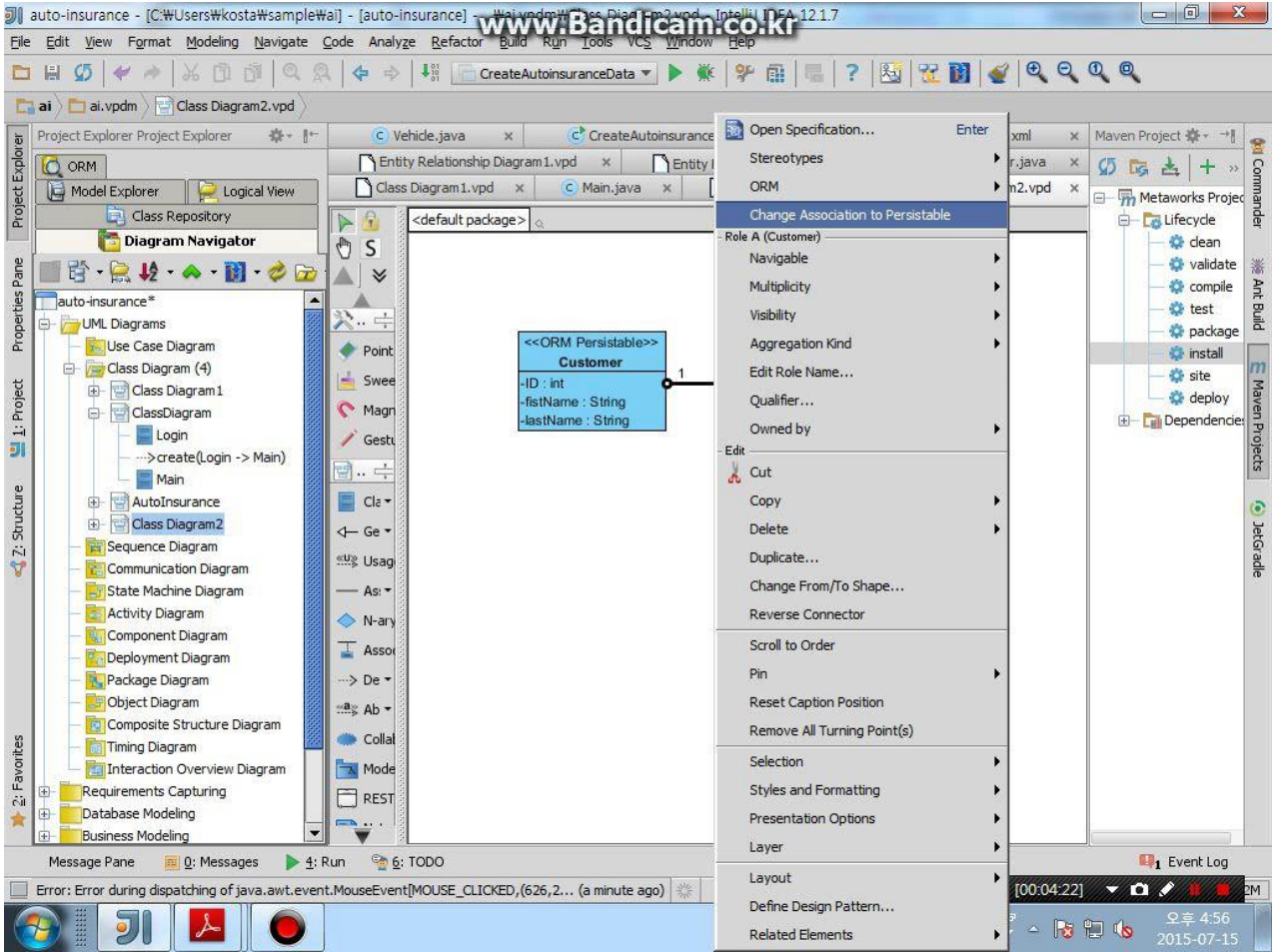
1:0..* Multiplicity



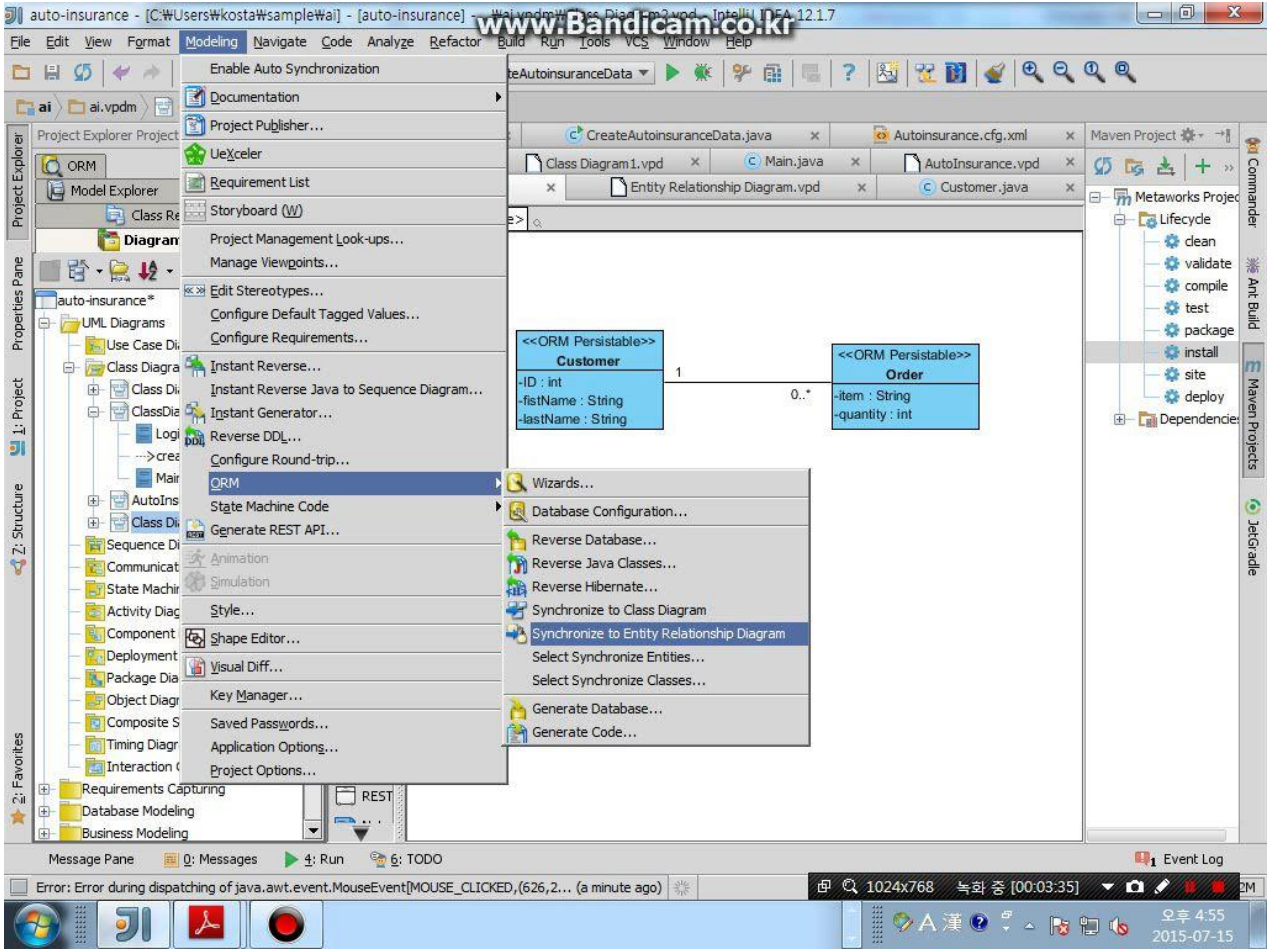
All classes must be ORMPersistable

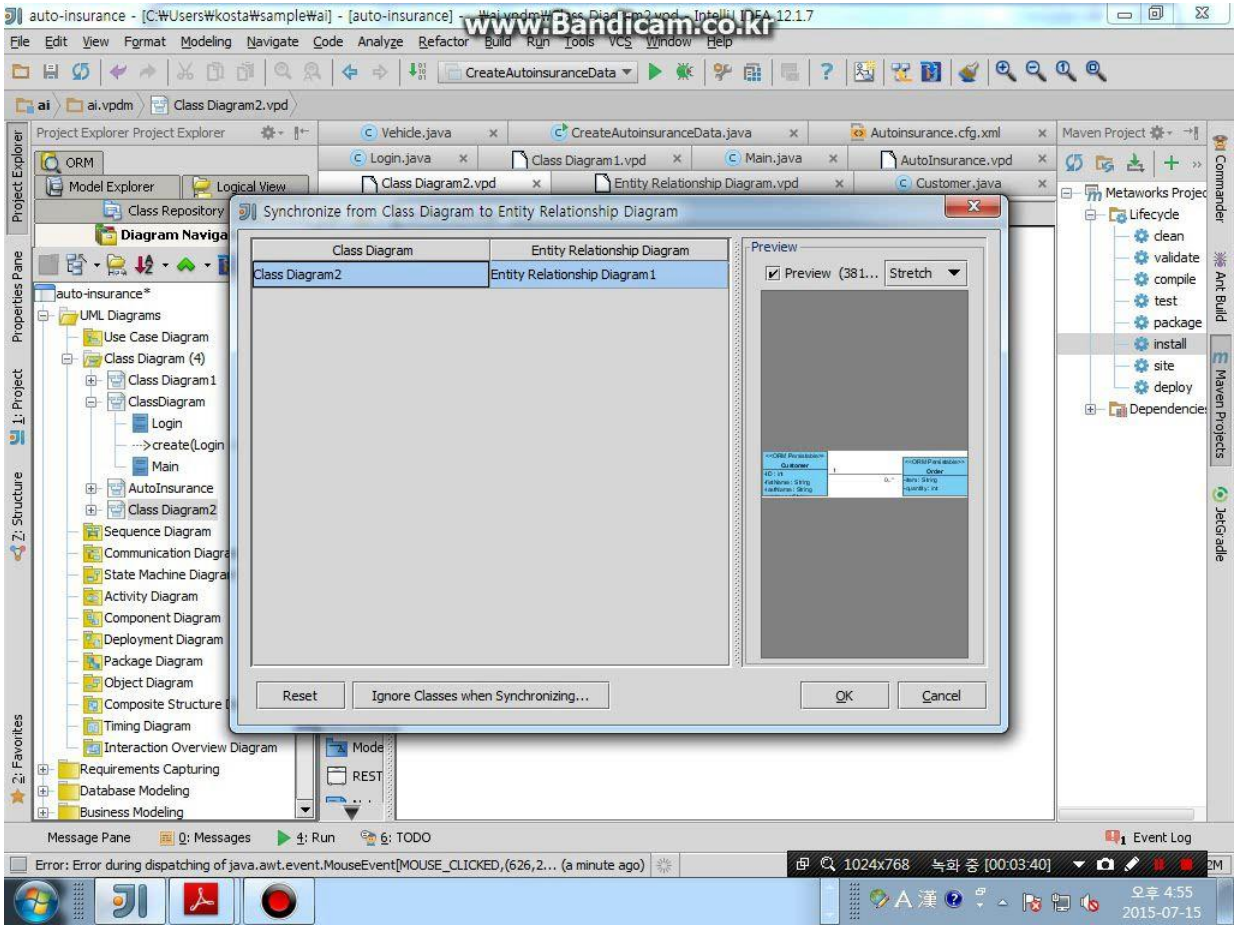


All relations must be Persistable

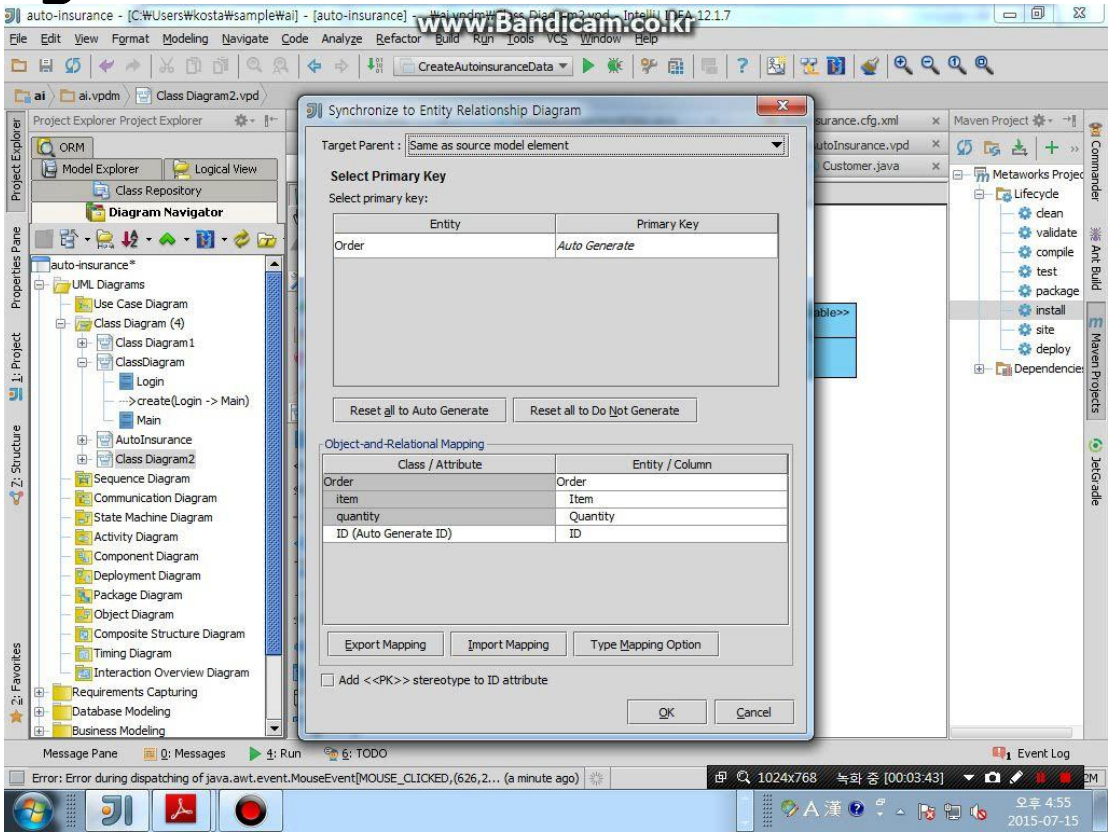


Then, they can be converted to ERD

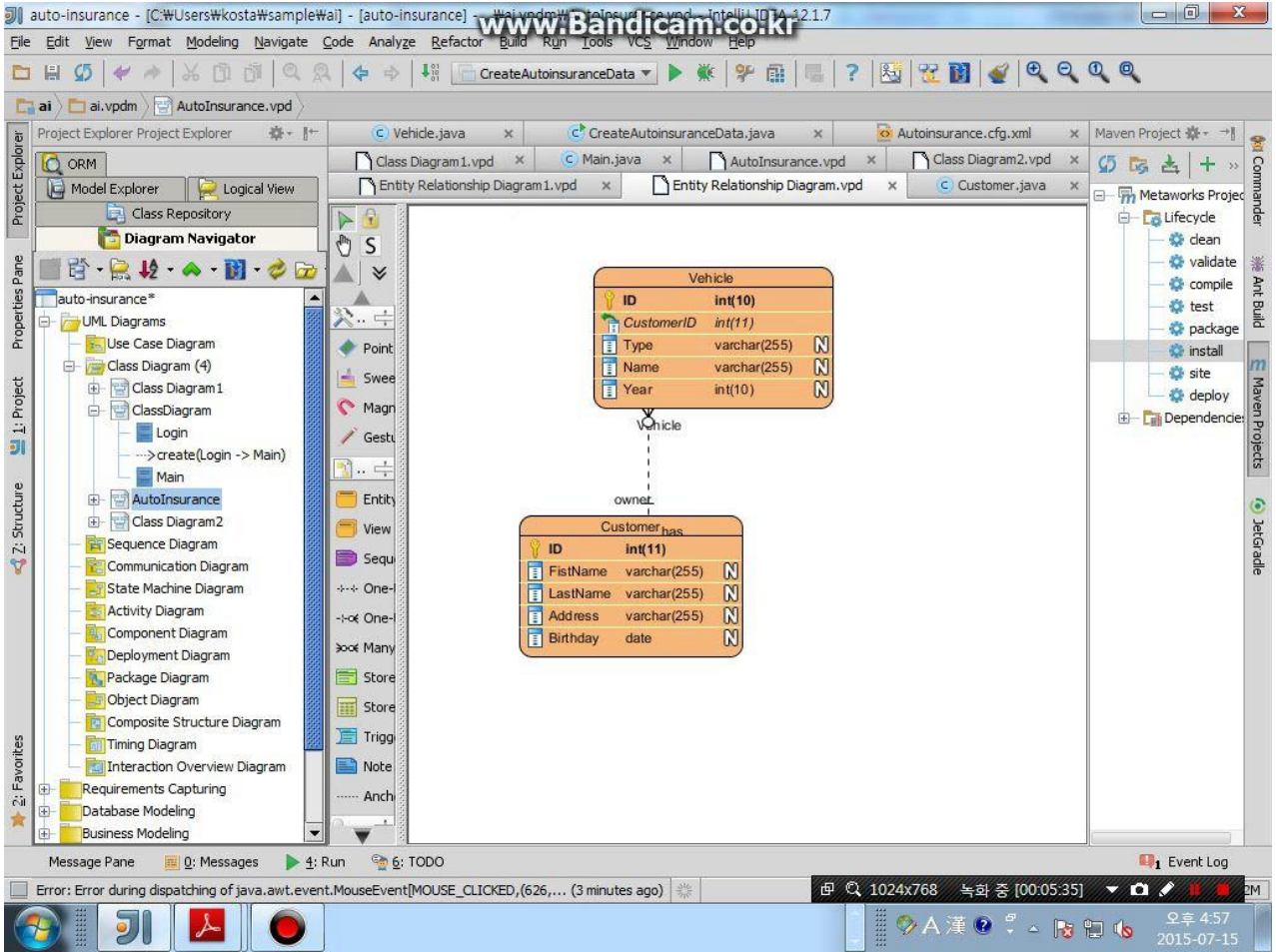



auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
www.Bandicam.co.kr
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
CreateAutoinsuranceData
ai
ai.vpdm
Class Diagram2.vpd
Project Explorer Project Explorer
Vehicle.java
CreateAutoinsuranceData.java
Autoinsurance.cfg.xml
Maven Project
Login.java
Class Diagram1.vpd
Main.java
AutoInsurance.vpd
Class Diagram2.vpd
Entity Relationship Diagram.vpd
Customer.java
ORM
Model Explorer
Logical View
Class Repository
Diagram Naviga
auto-insurance*
UML Diagrams
Use Case Diagram
Class Diagram (4)
Class Diagram1
ClassDiagram
Login
create(Login
Main
AutoInsurance
Class Diagram2
Sequence Diagram
Communication Diagra
State Machine Diagra
Activity Diagram
Component Diagram
Deployment Diagram
Package Diagram
Object Diagram
Composite Structure
Timing Diagram
Interaction Overview Diagram
Requirements Capturing
Database Modeling
Business Modeling
Synchronize from Class Diagram to Entity Relationship Diagram
Class Diagram
Entity Relationship Diagram
Class Diagram2
Entity Relationship Diagram1
Preview
Preview (381...
Stretch
Reset
Ignore Classes when Synchronizing...
OK
Cancel
Metaworks Projec
Lifecycle
clean
validate
compile
test
package
install
site
deploy
Dependencies
Commander
Ant Build
Maven Projects
JetGradle
Properties Pane
1: Project
7: Structure
2: Favorites
Message Pane
0: Messages
4: Run
6: TODO
Event Log
Error: Error during dispatching of java.awt.event.MouseEvent[MOUSE_CLICKED,(626,2... (a minute ago)
1024x768
녹화 중 [00:03:40]
오후 4:55
2015-07-15

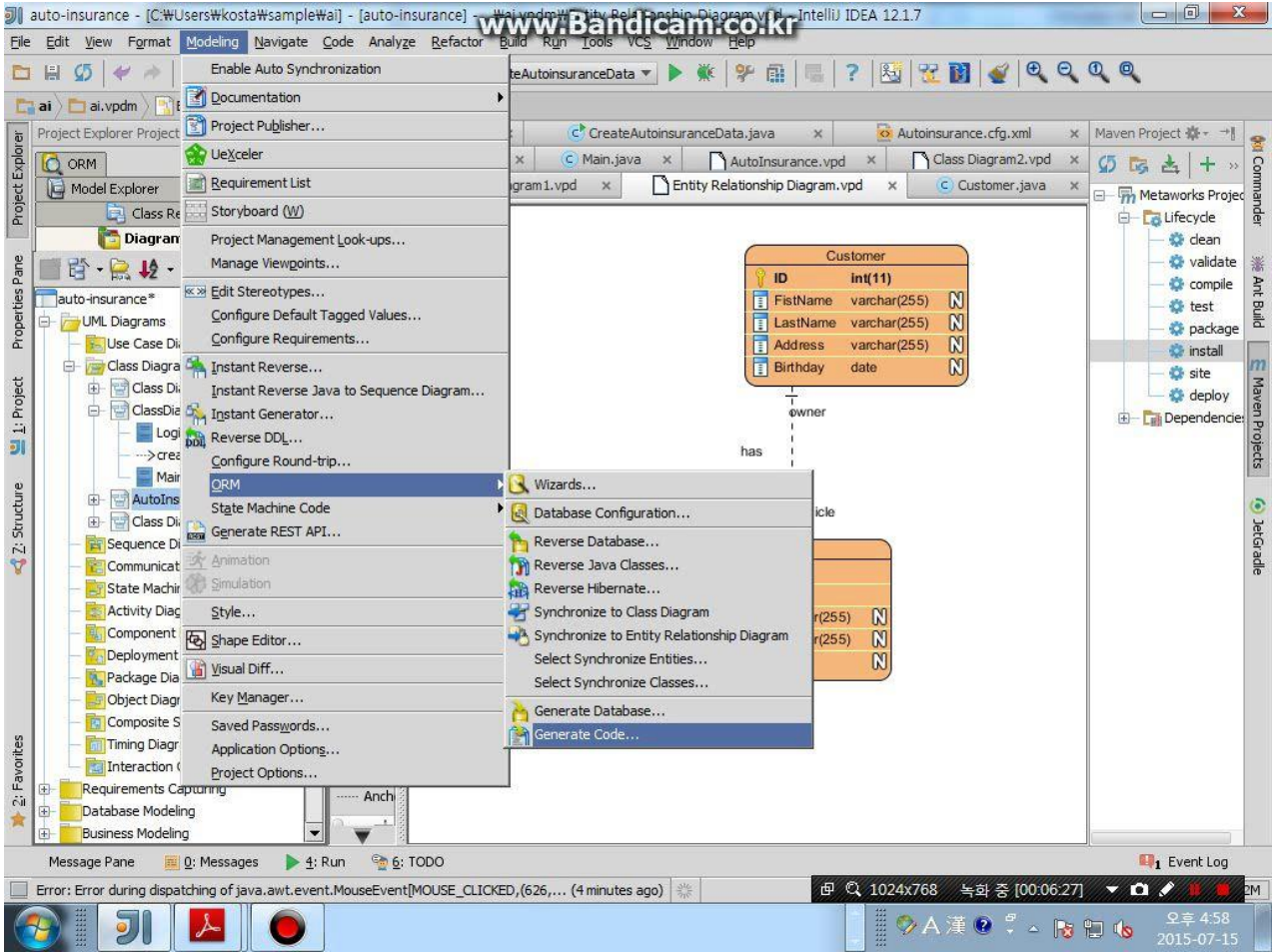
If there is no ID field (primary key), VP will generate them automatically



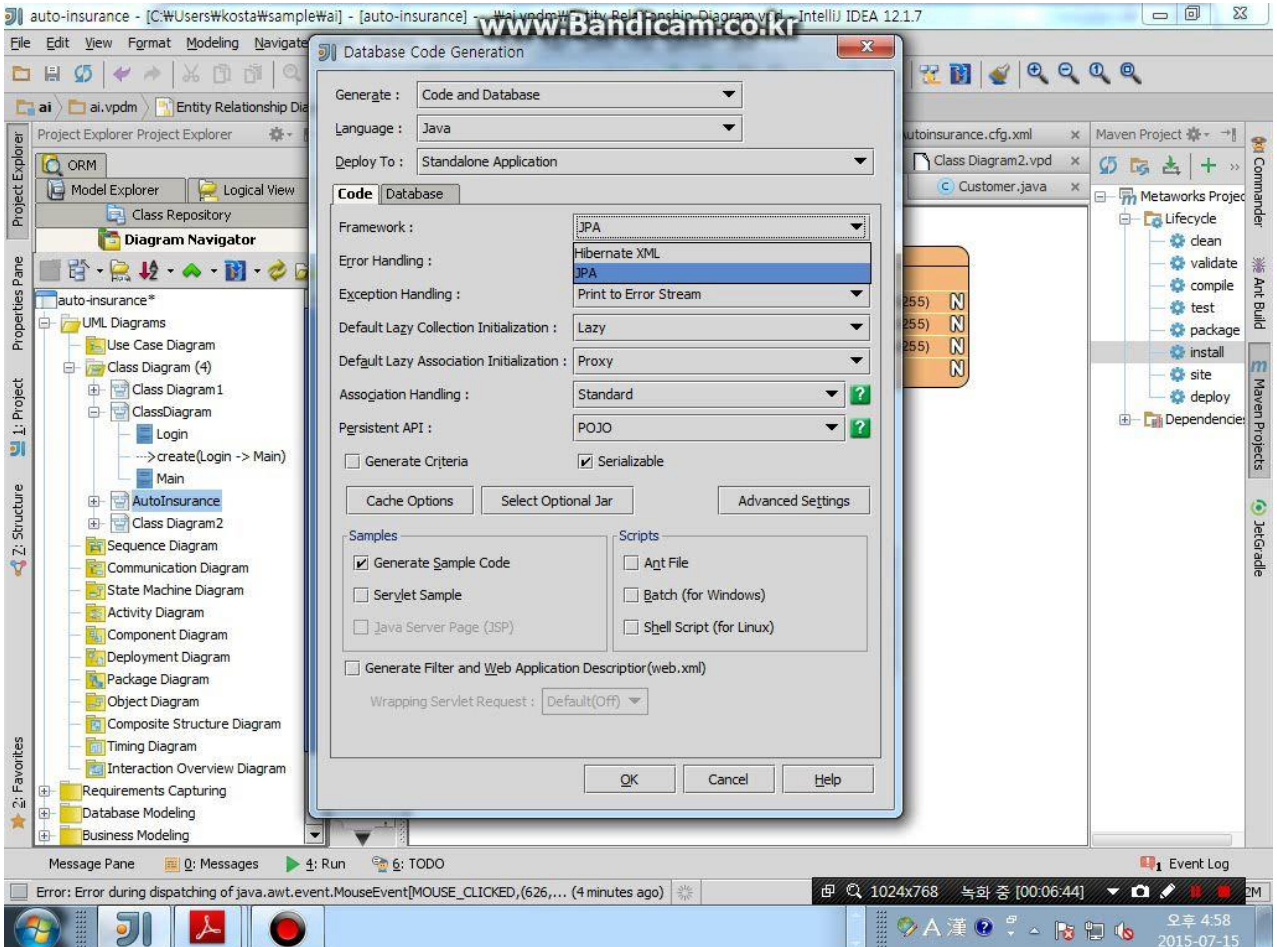
Generated ERD



Generating ORM Code



Settings: JPA and POJO



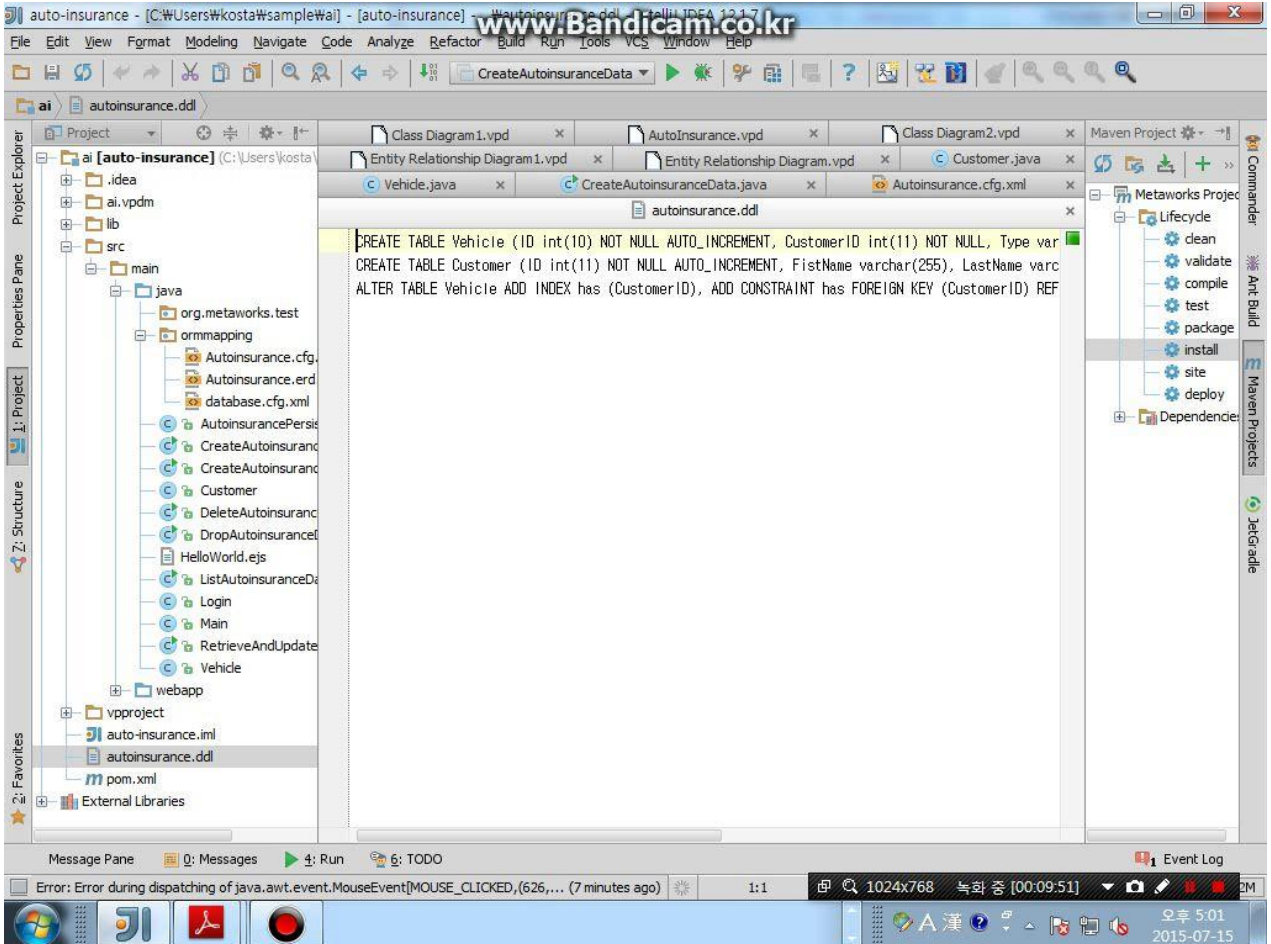
Generated JPA model code

```
*/
import java.io.Serializable;
import javax.persistence.*;
@Entity
@org.hibernate.annotations.Proxy(lazy=false)
@Table(name="Customer")
public class Customer implements Serializable {
    public Customer() {
    }

    @Column(name="ID", nullable=false, length=11)
    @Id
    @GeneratedValue(generator="CUSTOMER_ID_GENERATOR")
    @org.hibernate.annotations.GenericGenerator(name="CUSTOMER_ID_GENERATOR", strat
    private int ID;

    @Column(name="FistName", nullable=true, length=255)
    private String fistName;

    @Column(name="LastName", nullable=true, length=255)
    private String lastName;

    @Column(name="Address", nullable=true, length=255)
    private String address;

    @Column(name="Birthday", nullable=true)
    @Temporal(TemporalType.DATE)
```

Generated Sample CRUD Code

```
/**...*/
import org.orm.*;
public class CreateAutoinsuranceData {
    public void createTestData() throws PersistentException {
        PersistentTransaction t = AutoinsurancePersistentManager.instance().getSession(
        try {
            Customer customer = new Customer();          // Initialize the properties of
            AutoinsurancePersistentManager.instance().getSession().save(customer);

            Vehicle vehicle = new Vehicle();             // TODO Initialize the prop
            AutoinsurancePersistentManager.instance().getSession().save(vehicle);

            t.commit();
        }
        catch (Exception e) {
            t.rollback();
        }

    }

    public static void main(String[] args) {
        try {
            CreateAutoinsuranceData createAutoinsuranceData = new CreateAutoinsuranceDa
            try {
                createAutoinsuranceData.createTestData();
            }
            finally {
```

Generated SQL (DDL) Code



Generated Entity Classes (JPA)

- All Entity Codes are POJO Classes that are described with Java Annotations meaning entity field descriptions and relationships.

```
@Entity
@Table(name="Customer")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="Discriminator", discriminatorType=DiscriminatorType.STRING)
@DiscriminatorValue("Customer")
public class Customer implements Serializable {
        public Customer() {}

        @Column(name="ID", nullable=false)
        @Id
        @GeneratedValue(generator="AUTOINSURANCE_CUSTOMER_CUSTOMER_ID_GENERATOR")
        @org.hibernate.annotations.GenericGenerator(name="AUTOINSURANCE_CUSTOMER_CUSTOMER_ID_GENERATOR", strategy="native")
        private String customer_ID;

        @Column(name="Customer_social_security_number", nullable=true, length=255)
        private String customer_social_security_number;

        @Column(name="Customer_first_name", nullable=true, length=255)
        private String customer_first_name;

        @Column(name="Customer_middle_initial", nullable=true, length=255)
        private String customer_middle_initial;

        @Column(name="Customer_last_name", nullable=true, length=255)
        private String customer_last_name;

        @OneToMany(mappedBy="owns", targetEntity=autoinsurance.Vehicle.class)
        @org.hibernate.annotations.Cascade({org.hibernate.annotations.CascadeType.SAVE_UPDATE, org.hibernate.annotations.CascadeType.LOCK})

        @org.hibernate.annotations.LazyCollection(org.hibernate.annotations.LazyCollectionOption.TRUE)
        private java.util.Set vehicles = new java.util.HashSet();

}
```

Why JPA?

- The (almost-final) Java Standard

- Why not iBatis or hibernate ?
  - JPA is a POJO Framework while the others not.
  - Also Hibernate engine supports JPA as a interface.

- Why POJO framework is goooooood?
  - To focus on the domain-driven design
  - To yield the sophisticated Java class system to solve your Doman-Specific problem.
  - A POJO frameworks would not limits your domain models, they only gives an '*advice*' through @Annotation or Aspect-oriented manner.
  - So, you can introduce one or more frameworks without changing the domain classes.

Adding Dependencies

```
 <dependency>
        <groupId>org.eclipse.persistence</groupId>
        <artifactId>javax.persistence</artifactId>
        <version>2.1.0</version>
</dependency>
<dependency>
        <groupId>org.uengine.vp</groupId>
        <artifactId>orm</artifactId>
        <version>1.1-SNAPSHOT</version>
</dependency>
```

Basic JPA annotations

```
@Entity
class Item {

    @Id @GeneratedValue
    private Long id;

    // Field access enabled automatically
}
```

- **@Entity** identifies this class is a ORM Class.
- **@Id** means this field is a primary key.
- **@JoinColumn** means a foreign value referenced by a foreign key.
- **@OneToMany** means this field is an accessor of related entity in 'One-to-many' relationship.

Key Generation Strategy

- Native/AUTO:

```
@Entity
public class Persona {

    @Id
    @GeneratedValue
    private int id;
```

- SEQUENCE:

```
@Id
@GeneratedValue(strategy = GenerationType.SEQUENCE)
private Long id;
```

Data Insertion Test

```
public static void main(String[] args) throws PersistentException {

    PersistentTransaction t = autoinsurance.AutoinsurancePersistentManager.instance().getSession().beginTransaction();

    try{

        Customer customer = new Customer();

        customer.setCustomer_ID("1");
        customer.setCustomer_first_name("Jane Doe");

        customer.setVehicle(new ArrayList());
        customer.getVehicle().add(new Vehicle());

        autoinsurance.AutoinsurancePersistentManager.instance().getSession().save(customer);

        t.commit();
    }
    catch (Exception e) {
        t.rollback();
        throw e;
    }
}
```

According to CRC, add operations

1. Responsibilities can be mapped with operations of the class

Insurance_Policy

Super Classes: Insurance_Quote

Sub Classes:

Description:

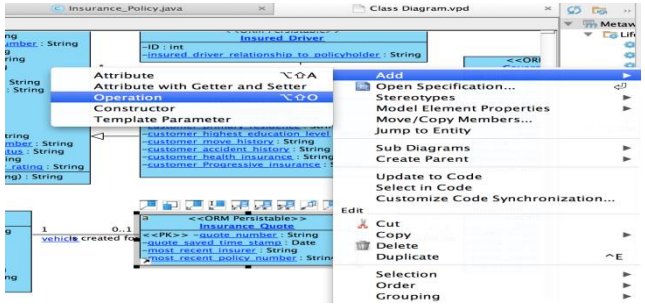
Attributes:

Name	Description
quote_number	
vehicle_VIN	
quote_saved_time_stamp	
policy_flag	
policy_number	
policy_coverage_start_date	
policy_bill_plan	
policy_sinature	
insurance_premium	

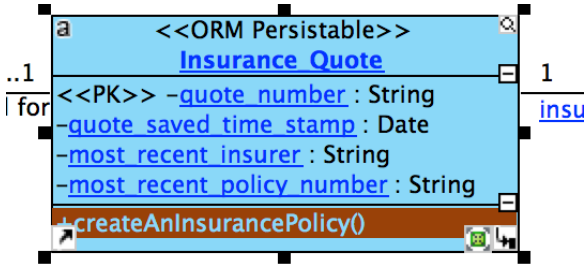
Responsibilities:

Name	Collaborator
Create an insurance policy	
Update an insurance policy	Insurance_Product, Coverage_Item, Coverage_Item_Option, Coverage_Item_Option_Selected
Produce the document of an insurance	Vehicle, Policyholder, Insured_Driver, Insurance_Product, Coverage_Item_Option_Selected

2. Right-click on the Class > Add > Operation



3. Name the operation as 'createInsurancePolicy()'



Generate code again and put the logic

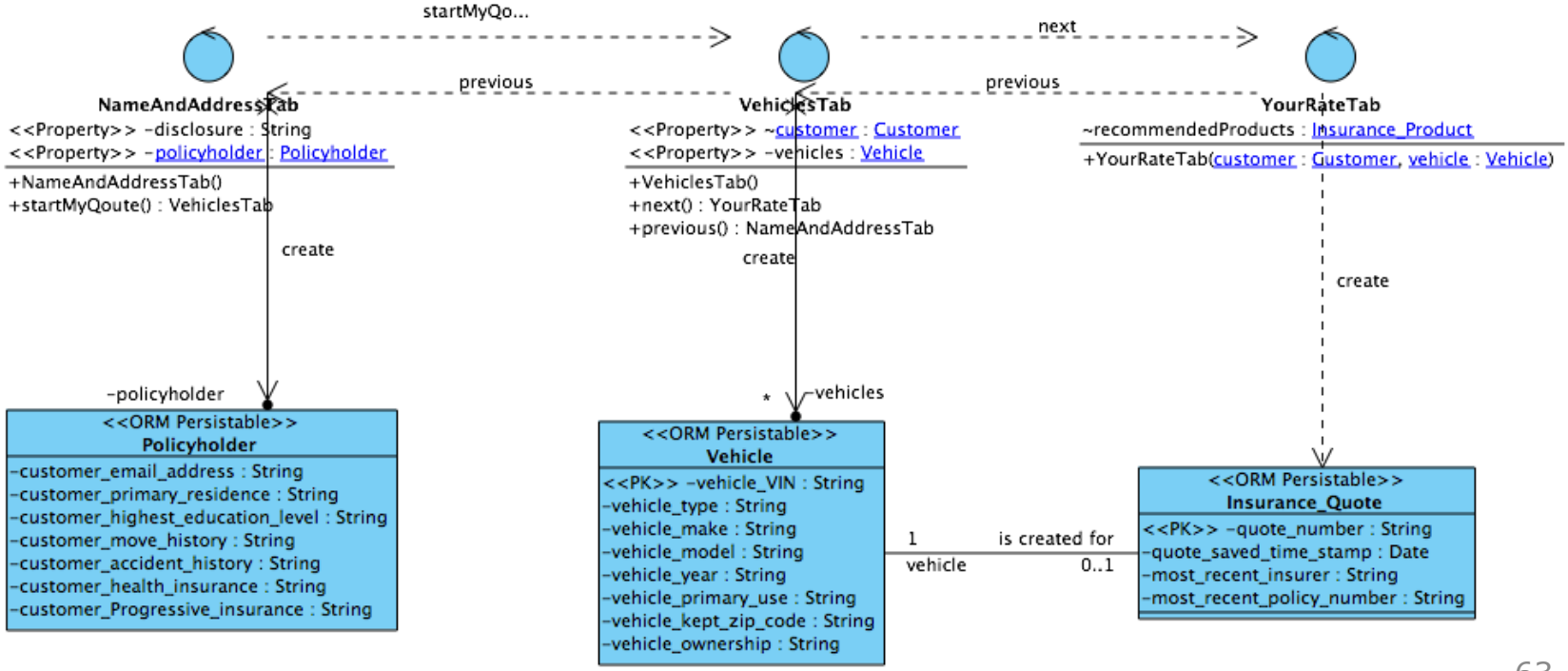
```
public void createInsurancePolicy() throws PersistentException {

        PersistentTransaction t =
autoinsurance.AutoinsurancePersistentManager.instance().getSession().beginTransaction();

        try{

            autoinsurance.AutoinsurancePersistentManager.instance().getSession().save(this);

            t.commit();
        }
        catch (Exception e) {

            t.rollback();

            throw e;
        }

}
```

Where are the new properties or methods inserted?

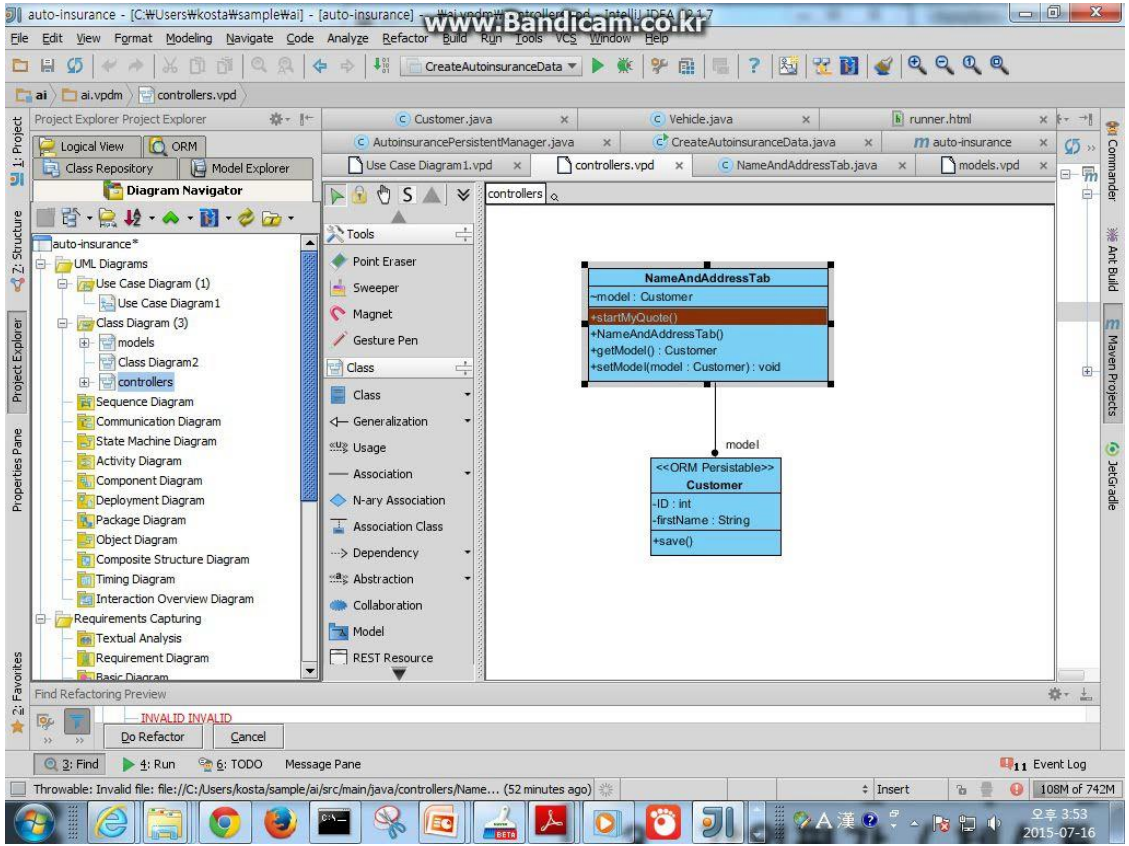
- To make it least message-calls between objects.
  - Most cohesive
  - Least coupling

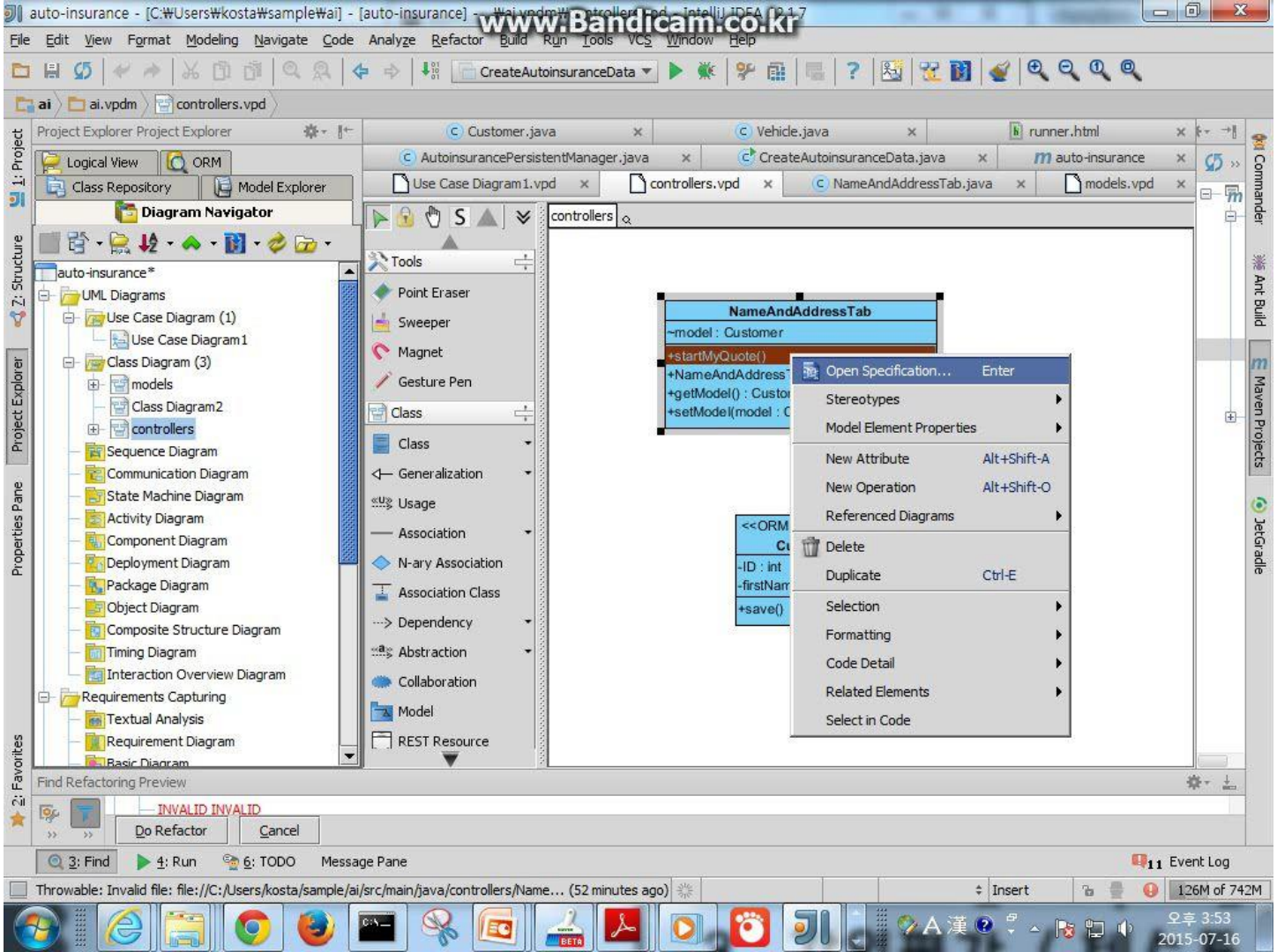
Creating Controller Classes

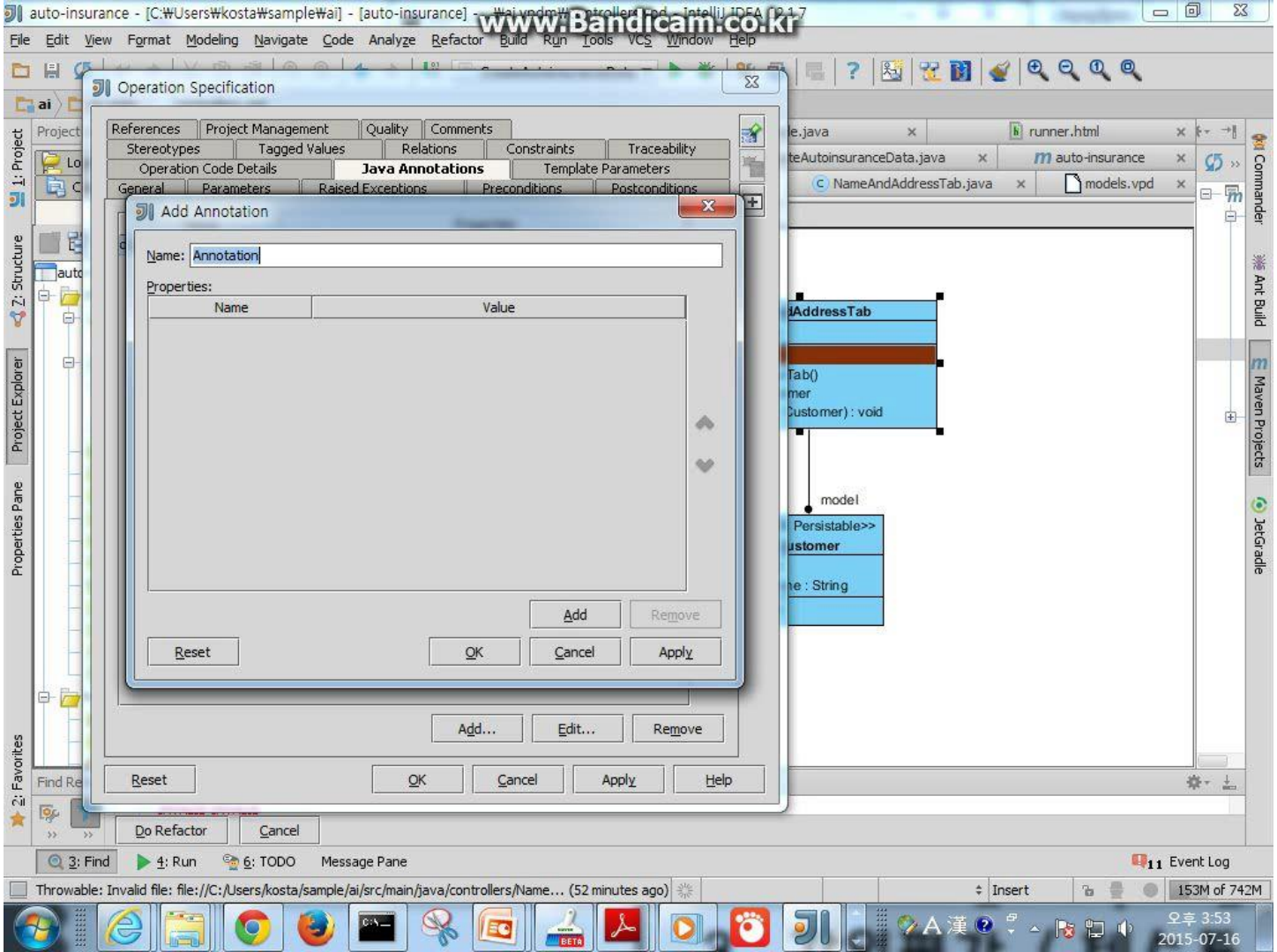
By separating entity and controller classes, the side-effects between them can be minimized.

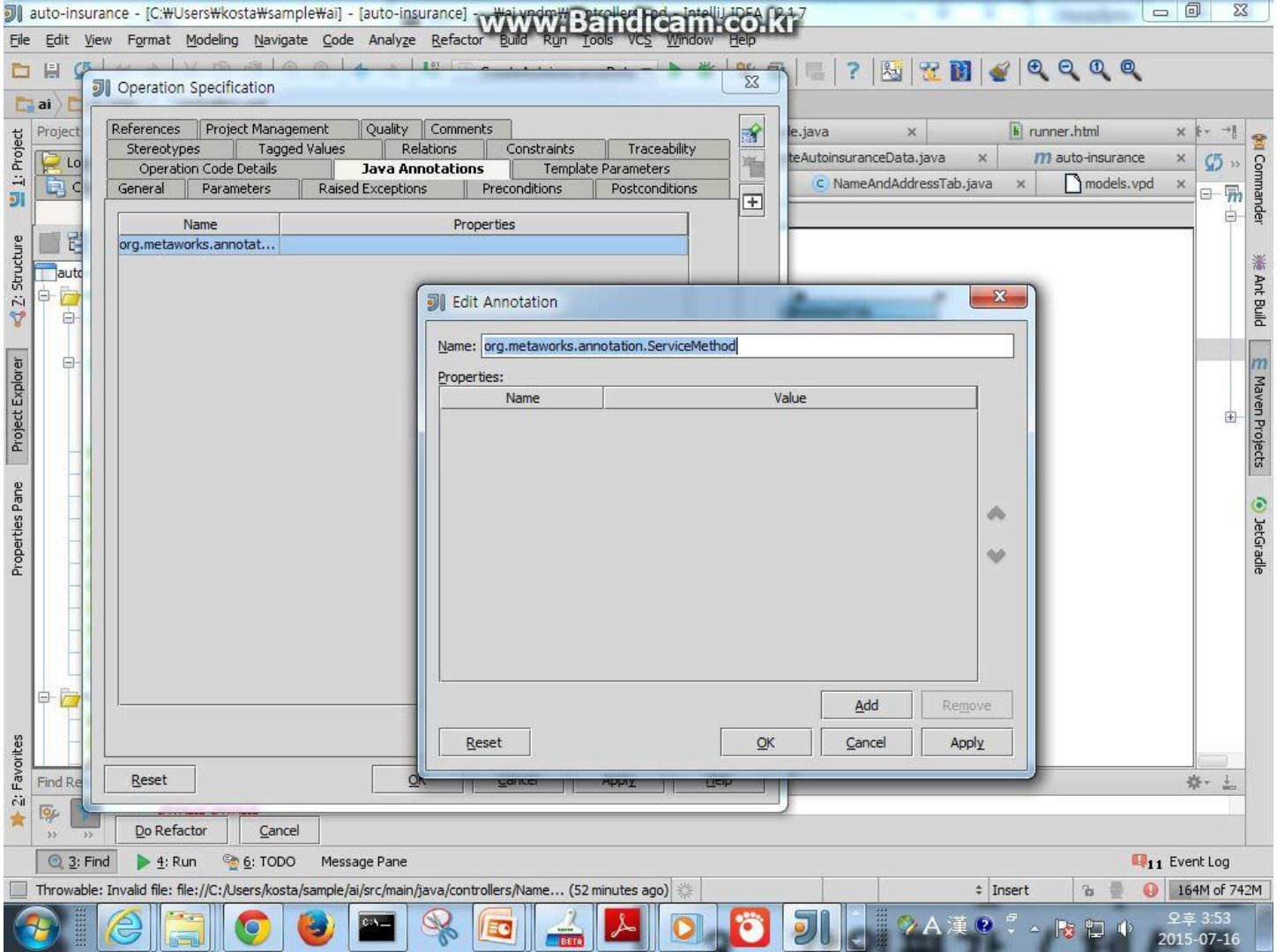


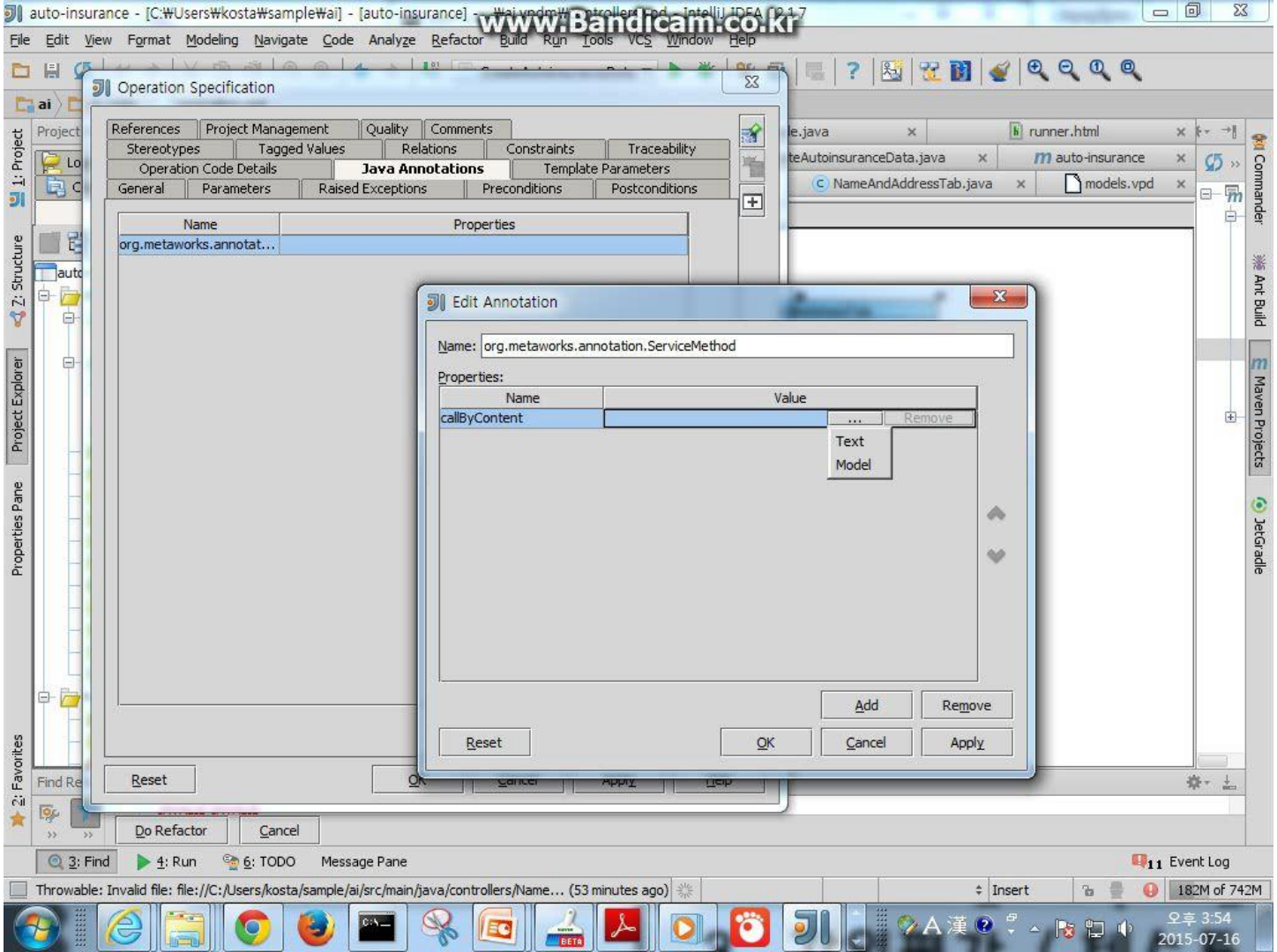
EXERCISE : CREATING CONTROLLERS AND USER INTERFACES

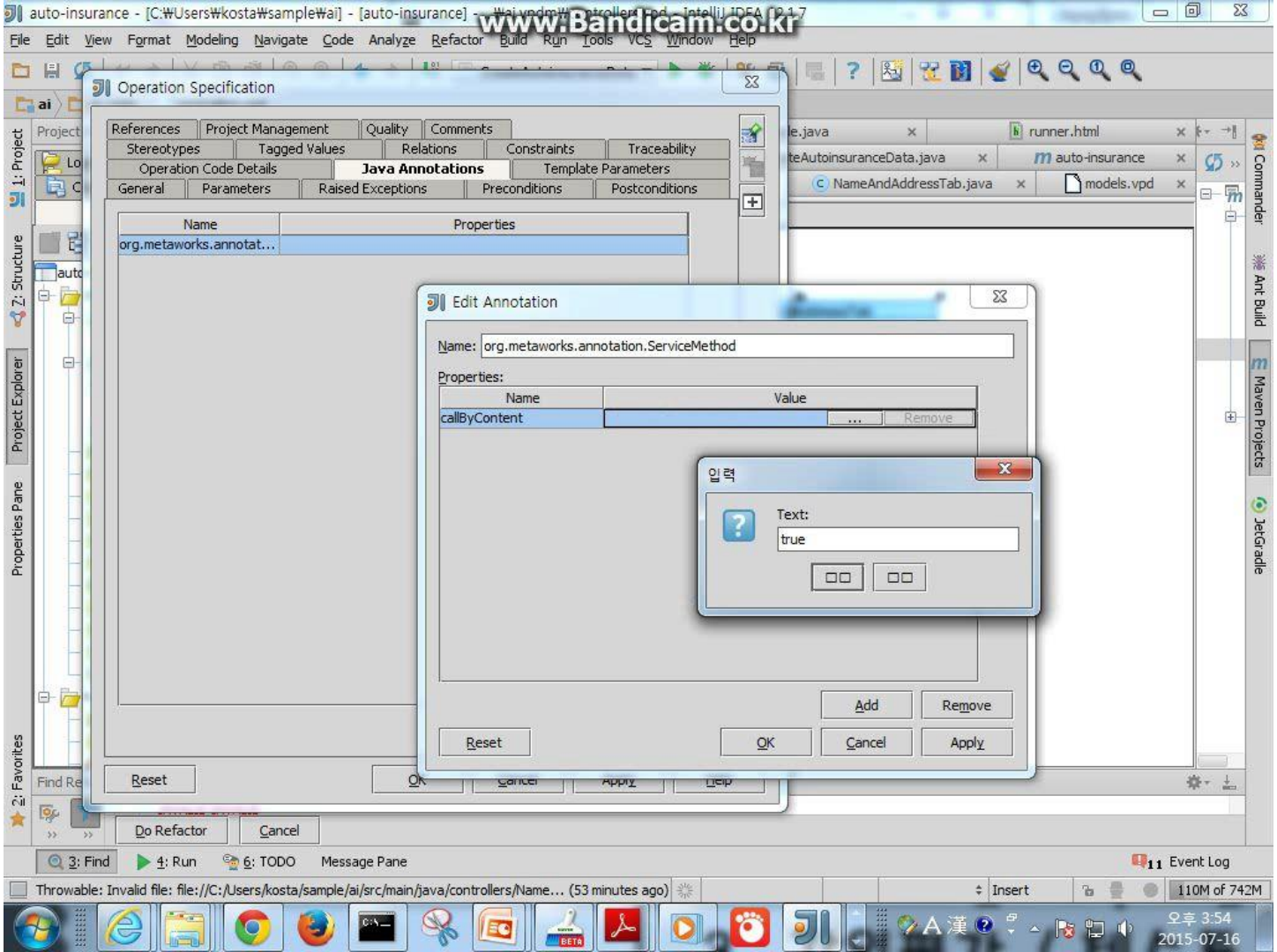

www.Bandicam.co.kr
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Project Explorer Project Explorer
Logical View
ORM
Class Repository
Model Explorer
Diagram Navigator
auto-insurance*
UML Diagrams
Use Case Diagram (1)
Use Case Diagram1
Class Diagram (3)
models
Class Diagram2
controllers
Sequence Diagram
Communication Diagram
State Machine Diagram
Activity Diagram
Component Diagram
Deployment Diagram
Package Diagram
Object Diagram
Composite Structure Diagram
Timing Diagram
Interaction Overview Diagram
Requirements Capturing
Textual Analysis
Requirement Diagram
Tools
Point Eraser
Sweeper
Magnet
Gesture Pen
Class
Generalization
Usage
Association
N-ary Association
Association Class
Dependency
Abstraction
Collaboration
Model
REST Resource
NameAndAddressTab
-model : Customer
+startMyQuote()
+NameAndAddressTab()
+getModel() : Customer
+setModel(model : Customer) : void
model
<<ORM Persistable>>
Customer
-ID : int
-firstName : String
+save()
Find Refactoring Preview
Do Refactor
Cancel
3: Find
4: Run
6: TODO
Message Pane
Event Log
Insert
108M of 742M
2015-07-16

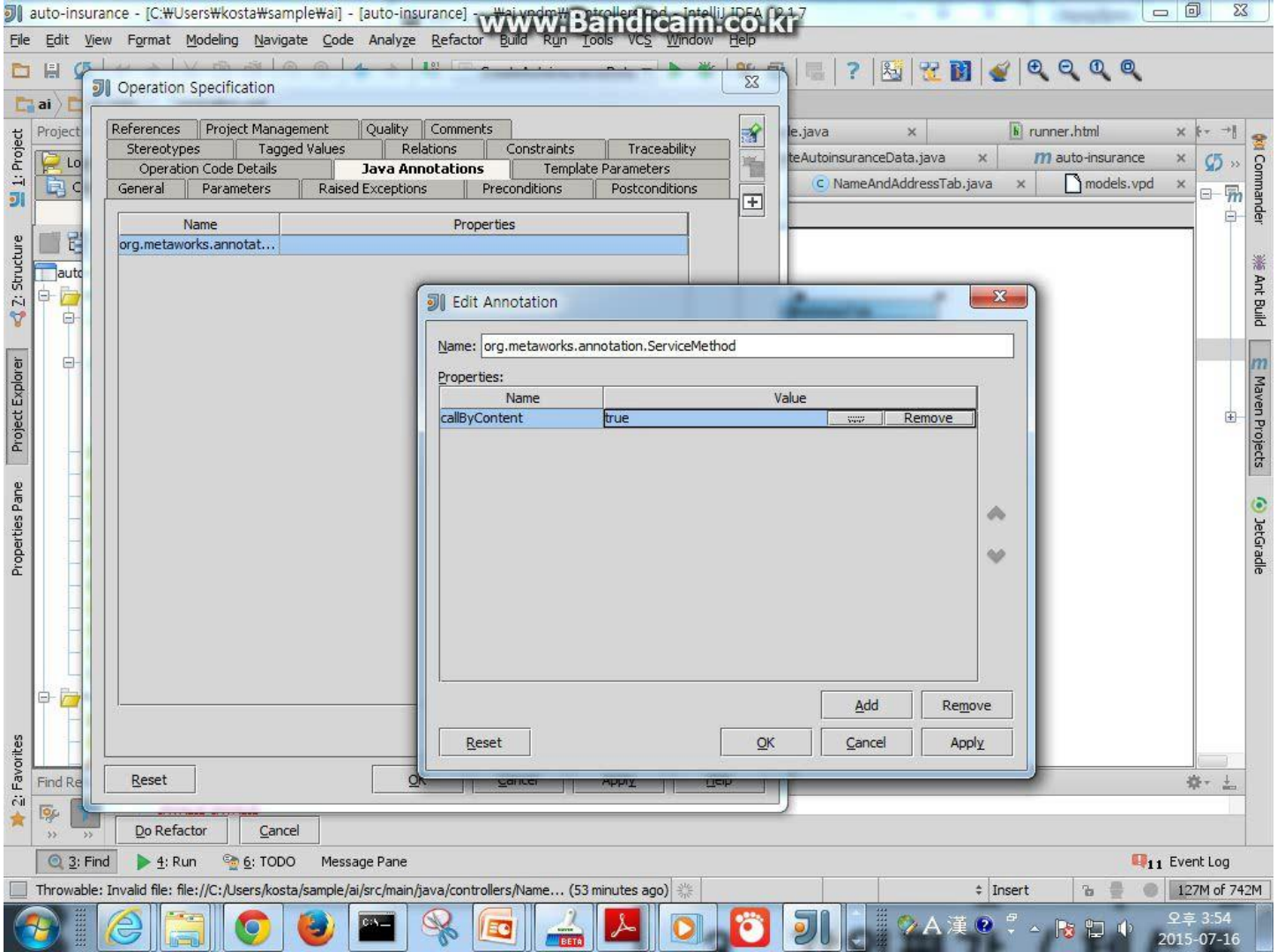

www.Bandicam.co.kr
auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
CreateAutoinsuranceData
ai
ai.vpdm
controllers.vpd
Project Explorer Project Explorer
Logical View
ORM
Class Repository
Model Explorer
Diagram Navigator
auto-insurance*
UML Diagrams
Use Case Diagram (1)
Use Case Diagram1
Class Diagram (3)
models
Class Diagram2
controllers
Sequence Diagram
Communication Diagram
State Machine Diagram
Activity Diagram
Component Diagram
Deployment Diagram
Package Diagram
Object Diagram
Composite Structure Diagram
Timing Diagram
Interaction Overview Diagram
Requirements Capturing
Textual Analysis
Requirement Diagram
Customer.java
Vehicle.java
runner.html
AutoinsurancePersistentManager.java
CreateAutoinsuranceData.java
auto-insurance
Use Case Diagram1.vpd
controllers.vpd
NameAndAddressTab.java
models.vpd
controllers
Tools
Point Eraser
Sweeper
Magnet
Gesture Pen
Class
Generalization
Usage
Association
N-ary Association
Association Class
Dependency
Abstraction
Collaboration
Model
REST Resource
NameAndAddressTab
~model : Customer
+startMyQuote()
+NameAndAddress
+getModel() : Custo
+setModel(model :
Open Specification... Enter
Stereotypes
Model Element Properties
New Attribute Alt+Shift-A
New Operation Alt+Shift-O
Referenced Diagrams
Delete
Duplicate Ctrl-E
Selection
Formatting
Code Detail
Related Elements
Select in Code
Find Refactoring Preview
INVALID INVALID
Do Refactor
Cancel
3: Find
4: Run
6: TODO
Message Pane
Event Log
Throwable: Invalid file: file://C:/Users/kosta/sample/ai/src/main/java/controllers/Name... (52 minutes ago)
Insert
126M of 742M
오후 3:53
2015-07-16

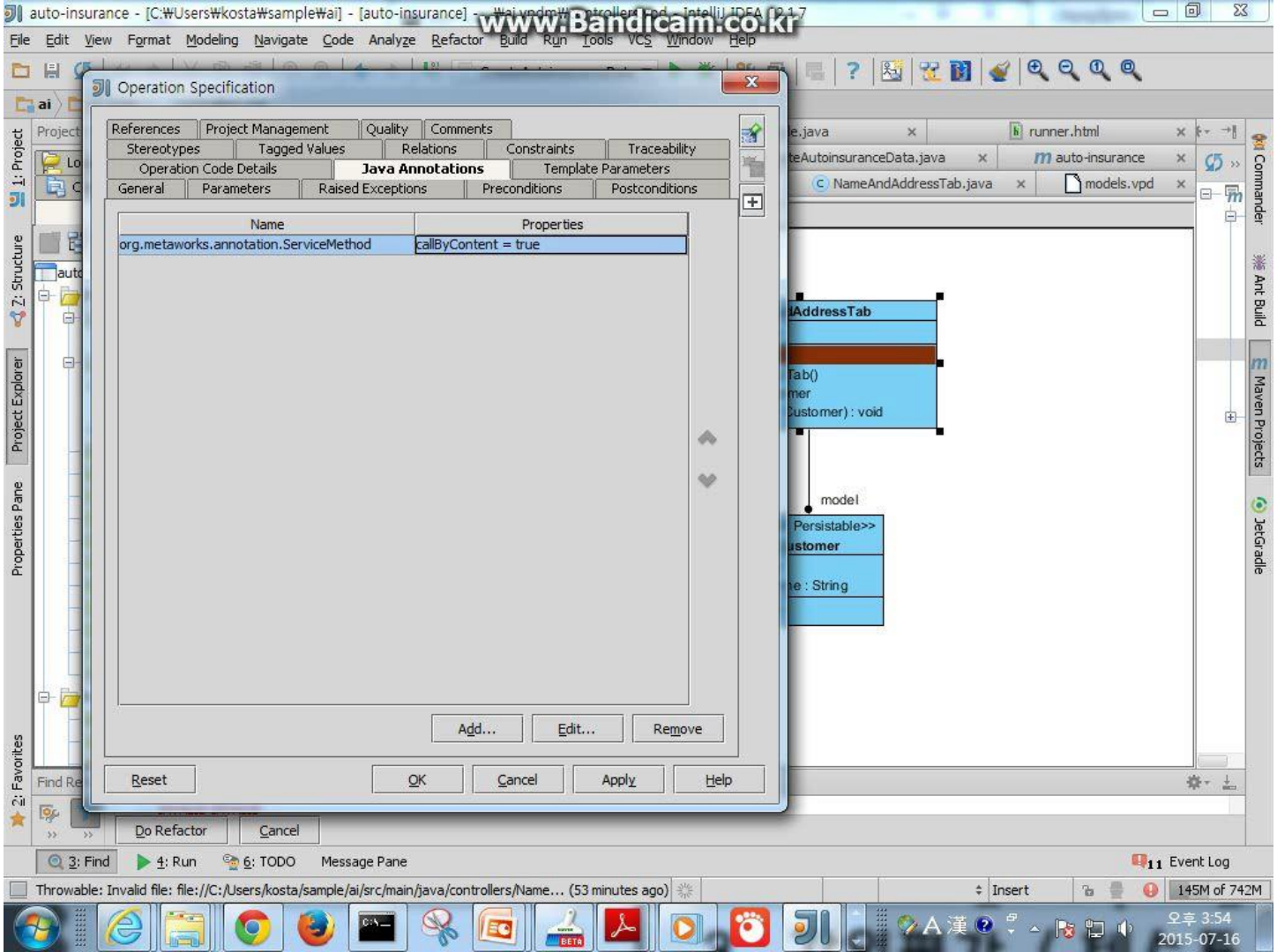

www.Bandicam.co.kr
auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Operation Specification
References
Project Management
Quality
Comments
Stereotypes
Tagged Values
Relations
Constraints
Traceability
Operation Code Details
Java Annotations
Template Parameters
General
Parameters
Raised Exceptions
Preconditions
Postconditions
Add Annotation
Name: Annotation
Properties:
Name
Value
Add
Remove
Reset
OK
Cancel
Apply
Add...
Edit...
Remove
Reset
OK
Cancel
Apply
Help
Do Refactor
Cancel
runner.html
auto-insurance
models.vpd
NameAndAddressTab.java
model
Persistable>>
Project Explorer
Properties Pane
Commander
Ant Build
Maven Projects
JetGradle
3: Find
4: Run
6: TODO
Message Pane
Event Log
Throwable: Invalid file: file://C:/Users/kosta/sample/ai/src/main/java/controllers/Name... (52 minutes ago)
Insert
153M of 742M
2015-07-16

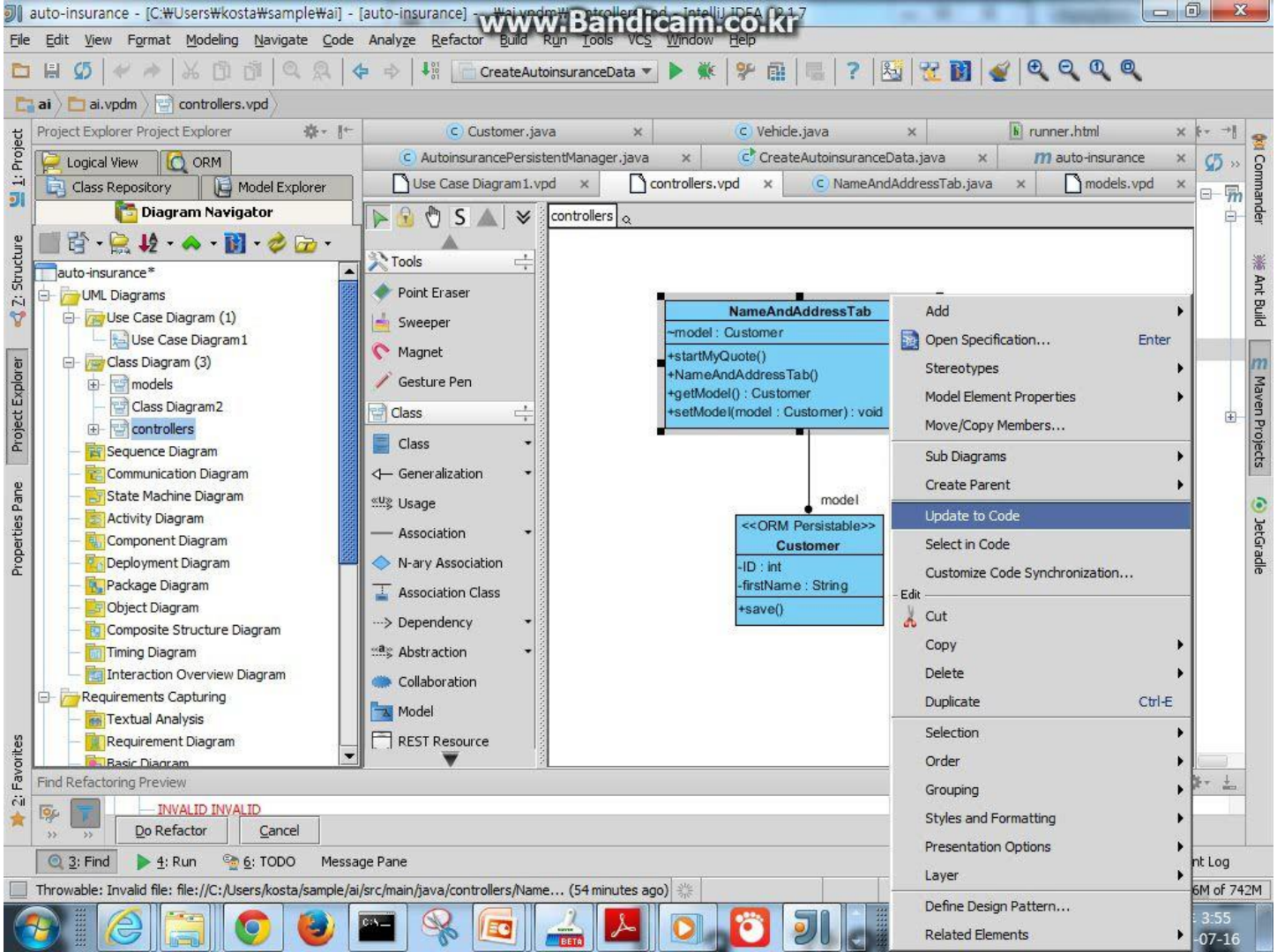

www.Bandicam.co.kr
auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Operation Specification
References
Project Management
Quality
Comments
Stereotypes
Tagged Values
Relations
Constraints
Traceability
Operation Code Details
Java Annotations
Template Parameters
General
Parameters
Raised Exceptions
Preconditions
Postconditions
Name
Properties
org.metaworks.annotat...
Reset
OK
Cancel
Apply
Help
Edit Annotation
Name: org.metaworks.annotation.ServiceMethod
Properties:
Name
Value
Add
Remove
Reset
OK
Cancel
Apply
runner.html
auto-insurance
NameAndAddressTab.java
models.vpd
Do Refactor
Cancel
3: Find
4: Run
6: TODO
Message Pane
Event Log
Throwable: Invalid file: file://C:/Users/kosta/sample/ai/src/main/java/controllers/Name... (52 minutes ago)
Insert
164M of 742M
오후 3:53
2015-07-16

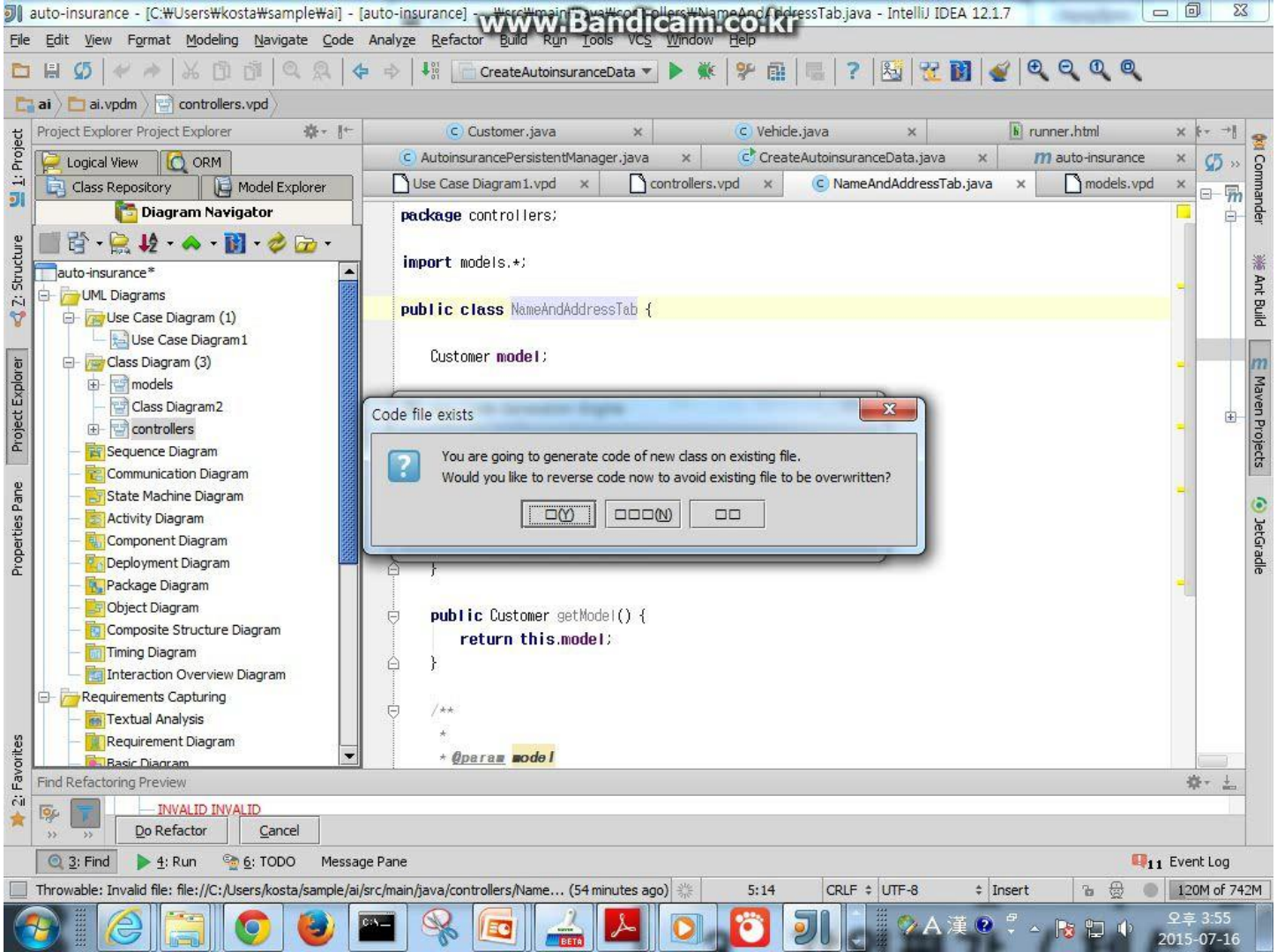

www.Bandicam.co.kr
auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Operation Specification
References
Project Management
Quality
Comments
Stereotypes
Tagged Values
Relations
Constraints
Traceability
Operation Code Details
Java Annotations
Template Parameters
General
Parameters
Raised Exceptions
Preconditions
Postconditions
Name
Properties
org.metaworks.annotat...
Reset
OK
Edit Annotation
Name: org.metaworks.annotation.ServiceMethod
Properties:
Name
Value
callByContent
...
Remove
Text
Model
Add
Remove
Reset
OK
Cancel
Apply
runner.html
auto-insurance
NameAndAddressTab.java
models.vpd
Do Refactor
Cancel
3: Find
4: Run
6: TODO
Message Pane
Event Log
Throwable: Invalid file: file://C:/Users/kosta/sample/ai/src/main/java/controllers/Name... (53 minutes ago)
Insert
182M of 742M
2015-07-16


www.Bandicam.co.kr
auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Operation Specification
References
Project Management
Quality
Comments
Stereotypes
Tagged Values
Relations
Constraints
Traceability
Operation Code Details
Java Annotations
Template Parameters
General
Parameters
Raised Exceptions
Preconditions
Postconditions
Name
Properties
org.metaworks.annotat...
Edit Annotation
Name: org.metaworks.annotation.ServiceMethod
Properties:
Name
Value
callByContent
Remove
입력
Text:
true
Add
Remove
Reset
OK
Cancel
Apply
Reset
OK
Cancel
Apply
Help
Do Refactor
Cancel
runner.html
auto-insurance
NameAndAddressTab.java
models.vpd
3: Find
4: Run
6: TODO
Message Pane
Event Log
Throwable: Invalid file: file://C:/Users/kosta/sample/ai/src/main/java/controllers/Name... (53 minutes ago)
Insert
110M of 742M
오후 3:54
2015-07-16

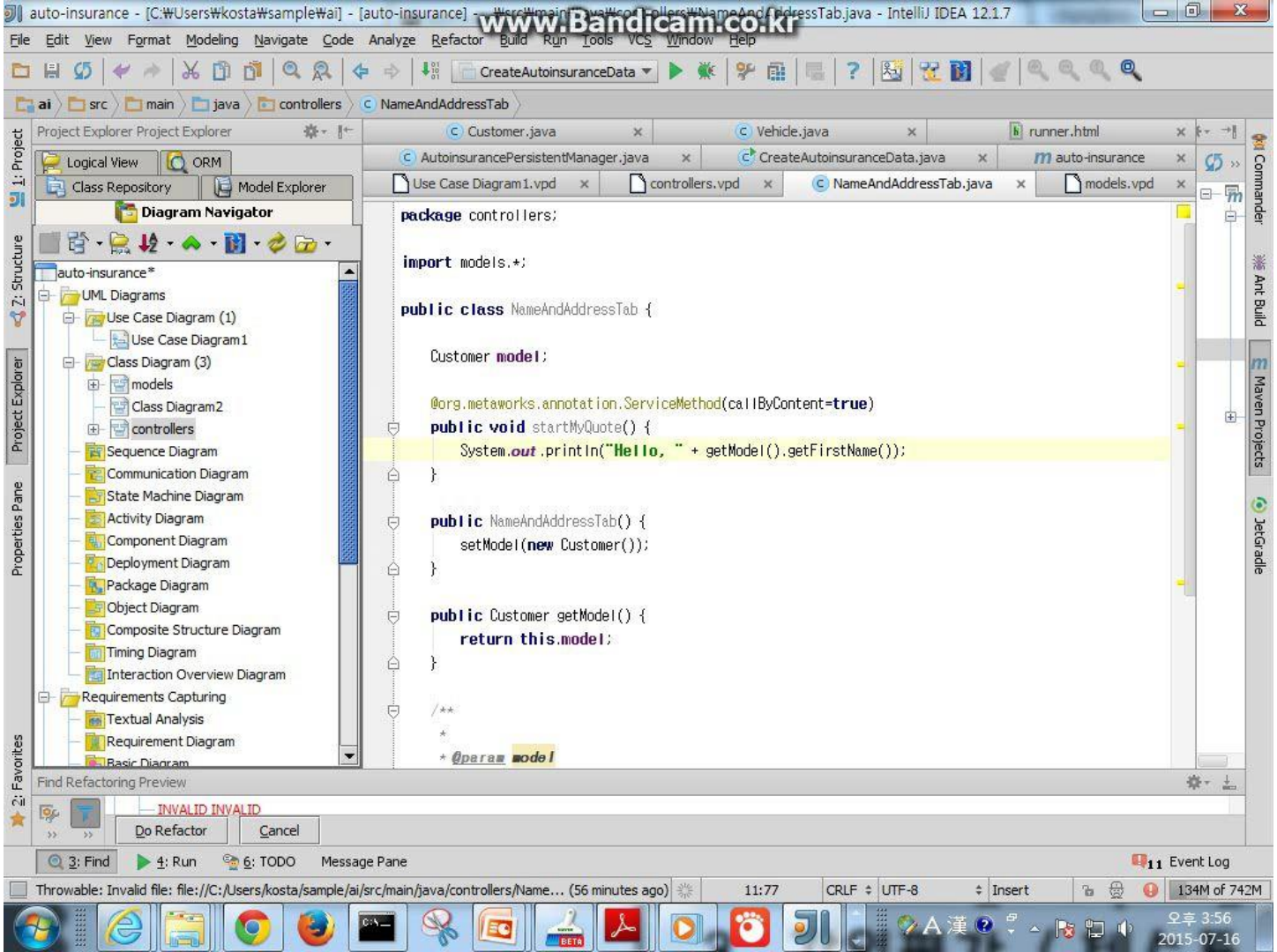

www.Bandicam.co.kr
auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Operation Specification
References
Project Management
Quality
Comments
Stereotypes
Tagged Values
Relations
Constraints
Traceability
Operation Code Details
Java Annotations
Template Parameters
General
Parameters
Raised Exceptions
Preconditions
Postconditions
Name
Properties
org.metaworks.annotat...
Reset
OK
Edit Annotation
Name: org.metaworks.annotation.ServiceMethod
Properties:
Name
Value
callByContent
true
Remove
Add
Remove
Reset
OK
Cancel
Apply
runner.html
auto-insurance
NameAndAddressTab.java
models.vpd
Do Refactor
Cancel
3: Find
4: Run
6: TODO
Message Pane
Event Log
Insert
127M of 742M
2015-07-16

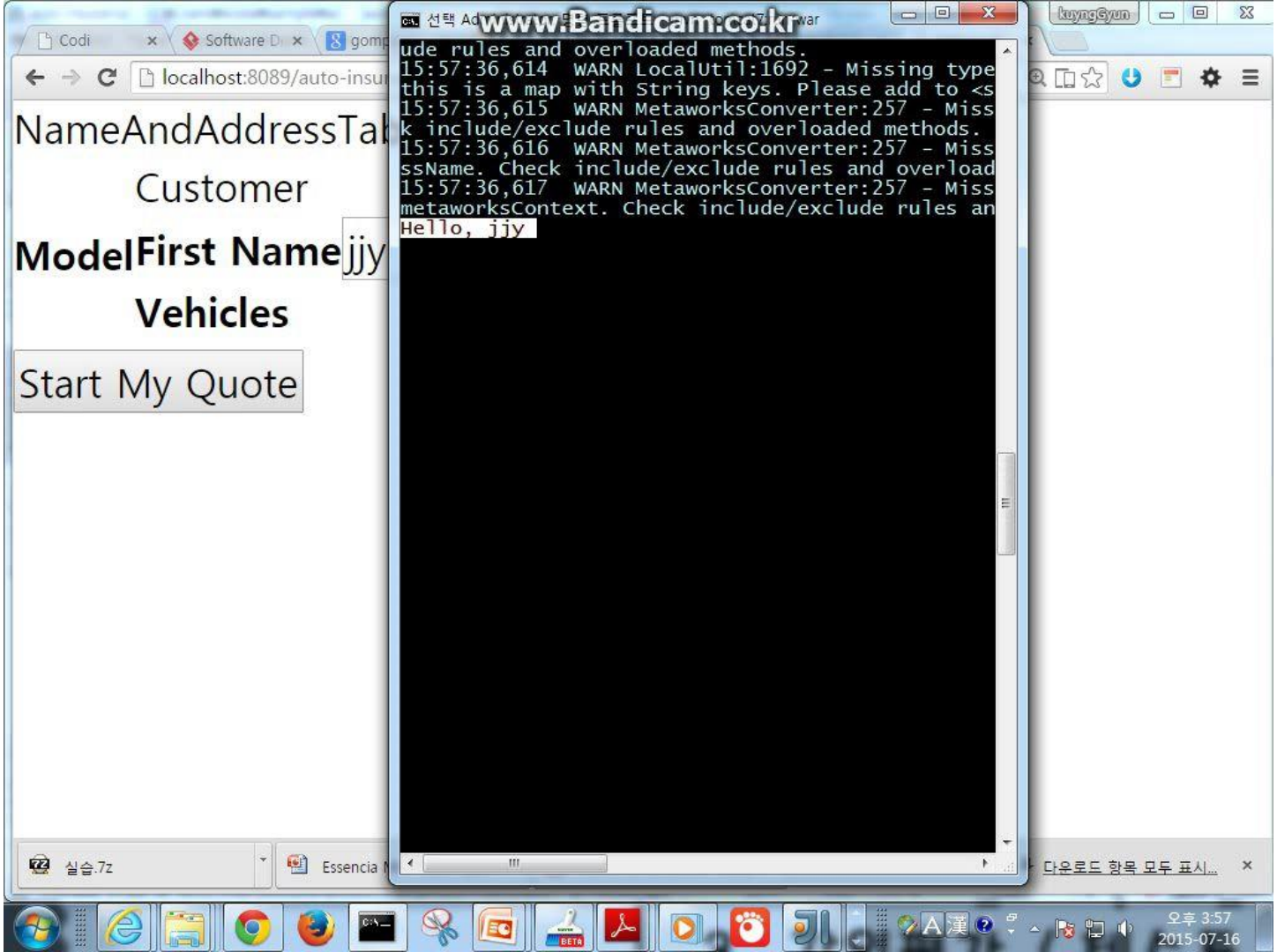

auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
www.Bandicam.co.kr
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Operation Specification
References
Project Management
Quality
Comments
Stereotypes
Tagged Values
Relations
Constraints
Traceability
Operation Code Details
Java Annotations
Template Parameters
General
Parameters
Raised Exceptions
Preconditions
Postconditions
Name
Properties
org.metaworks.annotation.ServiceMethod
callByContent = true
Add...
Edit...
Remove
Reset
OK
Cancel
Apply
Help
runner.html
auto-insurance
models.vpd
NameAndAddressTab.java
model
Do Refactor
Cancel
3: Find
4: Run
6: TODO
Message Pane
Event Log
Insert
145M of 742M
오후 3:54
2015-07-16


auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance]
www.Bandicam.co.kr
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
CreateAutoinsuranceData
ai
ai.vpdm
controllers.vpd
Project Explorer Project Explorer
Logical View
ORM
Class Repository
Model Explorer
Diagram Navigator
auto-insurance*
UML Diagrams
Use Case Diagram (1)
Use Case Diagram1
Class Diagram (3)
models
Class Diagram2
controllers
Sequence Diagram
Communication Diagram
State Machine Diagram
Activity Diagram
Component Diagram
Deployment Diagram
Package Diagram
Object Diagram
Composite Structure Diagram
Timing Diagram
Interaction Overview Diagram
Requirements Capturing
Textual Analysis
Requirement Diagram
Customer.java
Vehicle.java
runner.html
AutoinsurancePersistentManager.java
CreateAutoinsuranceData.java
auto-insurance
Use Case Diagram1.vpd
controllers.vpd
NameAndAddressTab.java
models.vpd
Tools
Point Eraser
Sweeper
Magnet
Gesture Pen
Class
Generalization
Usage
Association
N-ary Association
Association Class
Dependency
Abstraction
Collaboration
Model
REST Resource
NameAndAddressTab
~model : Customer
+startMyQuote()
+NameAndAddressTab()
+getModel() : Customer
+setModel(model : Customer) : void
model
<<ORM Persistable>>
Customer
-ID : int
-firstName : String
+save()
Add
Open Specification... Enter
Stereotypes
Model Element Properties
Move/Copy Members...
Sub Diagrams
Create Parent
Update to Code
Select in Code
Customize Code Synchronization...
Edit
Cut
Copy
Delete
Duplicate Ctrl-E
Selection
Order
Grouping
Styles and Formatting
Presentation Options
Layer
Define Design Pattern...
Related Elements
Find Refactoring Preview
INVALID INVALID
Do Refactor
Cancel
3: Find
4: Run
6: TODO
Message Pane
Throwable: Invalid file: file://C:/Users/kosta/sample/ai/src/main/java/controllers/Name... (54 minutes ago)



www.Bandicam.co.kr


www.Bandicam.co.kr
auto-insurance - [C:₩Users₩kosta₩sample₩ai] - [auto-insurance] - ₩src₩main₩java₩controllers₩NameAndAddressTab.java - IntelliJ IDEA 12.1.7
File Edit View Format Modeling Navigate Code Analyze Refactor Build Run Tools VCS Window Help
CreateAutoinsuranceData
ai src main java controllers NameAndAddressTab
Project Explorer Project Explorer
Logical View ORM
Class Repository Model Explorer
Diagram Navigator
auto-insurance*
UML Diagrams
Use Case Diagram (1)
Use Case Diagram1
Class Diagram (3)
models
Class Diagram2
controllers
Sequence Diagram
Communication Diagram
State Machine Diagram
Activity Diagram
Component Diagram
Deployment Diagram
Package Diagram
Object Diagram
Composite Structure Diagram
Timing Diagram
Interaction Overview Diagram
Requirements Capturing
Textual Analysis
Requirement Diagram
Customer.java
Vehicle.java
runner.html
AutoinsurancePersistentManager.java
CreateAutoinsuranceData.java
auto-insurance
Use Case Diagram1.vpd
controllers.vpd
NameAndAddressTab.java
models.vpd
package controllers;
import models.*;
public class NameAndAddressTab {
Customer model;
@org.metaworks.annotation.ServiceMethod(callByContent=true)
public void startMyQuote() {
System.out.println("Hello, " + getModel().getFirstName());
}
public NameAndAddressTab() {
setModel(new Customer());
}
public Customer getModel() {
return this.model;
}
/**
*
* @param model
Find Refactoring Preview
INVALID INVALID
Do Refactor
Cancel
3: Find
4: Run
6: TODO
Message Pane
Event Log
Throwable: Invalid file: file://C:/Users/kosta/sample/ai/src/main/java/controllers/Name... (56 minutes ago)
11:77
CRLF
UTF-8
Insert
134M of 742M
오후 3:56
2015-07-16


www.Bandicam.co.kr
localhost:8089/auto-insu
NameAndAddressTab
Customer
ModelFirst Name jjy
Vehicles
Start My Quote
ude rules and overloaded methods.
15:57:36,614 WARN LocalUtil:1692 - Missing type
this is a map with String keys. Please add to <s
15:57:36,615 WARN MetaworksConverter:257 - Miss
k include/exclude rules and overloaded methods.
15:57:36,616 WARN MetaworksConverter:257 - Miss
ssName. Check include/exclude rules and overload
15:57:36,617 WARN MetaworksConverter:257 - Miss
metaworksContext. Check include/exclude rules an
Hello, jjy

Creating UIs

- Developing UI Applications is very complex and costly:
  - Native UI: Swing
  - Web UI: JSP, Struts
  - N-Screen UI: AJAX / html5 / CSS

- Development Lifecycle is getting shorter and shorter:
  - Lean Startup
  - Early Prototyping

- So, we need something agile development approach such as:
  - Model-Driven Development
  - Application Frameworks
  - SOA and Process Orchestration Architecture

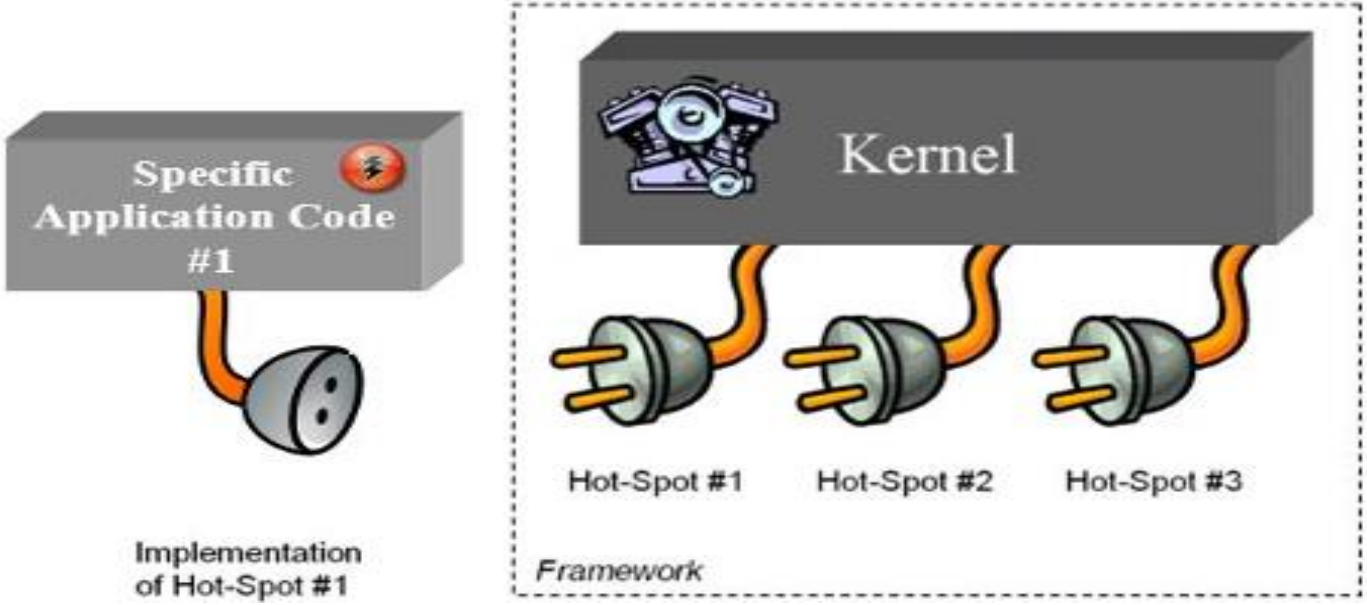
What is Framework?

Achieving 'Reuse of Design'

through

1. **Inversion of Control**
2. **Separation of Concerns**

Framework of Framework

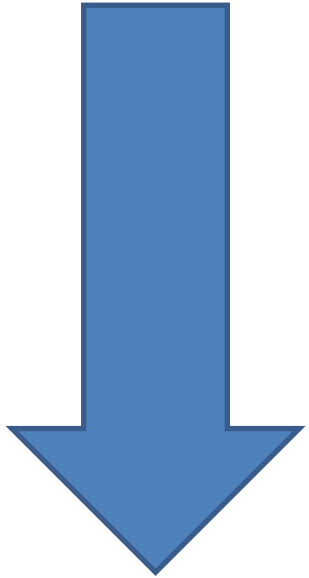


Frameworks, Toolkits and Libraries

	Libraries	Toolkits	Frameworks
Code Reuse	✓		✓
Code Generation		✓	
Design Reuse			✓

History of OO Frameworks and Methodologies

1. Smalltalk
2. STL (Standard Template Language) of C++
3. Adaptive Object Models – Ralf Johnson
4. Design Patterns – Gang of four
5. Object-Oriented Frameworks
6. OMG Reflection
7. Java – 1995
8. IBM's San francisco Framework
9. Java 2 Enterprise Edition – EJB 2.1
10. EJB Patterns



- ***Bigger and Complex Super Classes***
- ***Java doesn't Support Multiple Inheritance***

Domain Specific Application Frameworks

Example of Business Domain Specific Application Frameworks :

1. Car Simulation
2. Product Simulation
3. Or your Problem itself!!

→ abstracts business problems

Example of System Domain Specific Application Frameworks :

1. EJB
2. Struts
3. Swing
4. Apache Mesos

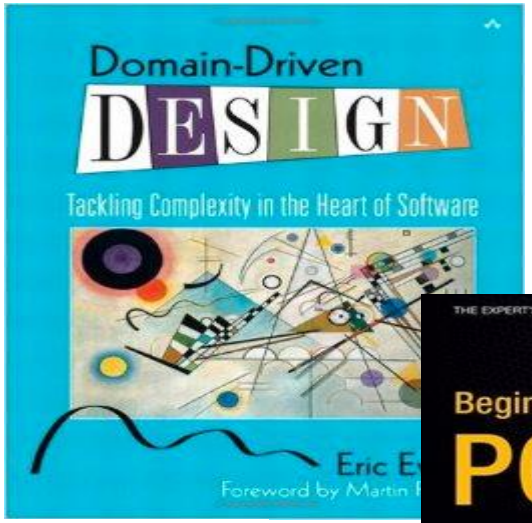
→ abstracts technical problems

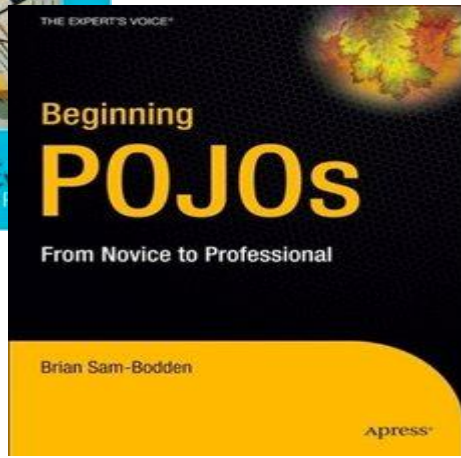
Merging two or more frameworks??

But We Can't Inherit Two Super Classes at the same time!!!

Return to the “Pure Java”

- Aspect Oriented Programming
- Java Annotation
- JPA
- JAX-RS
- EJB 3.1





spring

EJB 2.1 vs EJB 3.1

Java 클래스

```
public class AccountBean
implements javax.ejb.SessionBean {

      SessionContext ctx;
      DataSource accountDB;

      public void setSessionContext(SessionContext ctx) {
         this.ctx = ctx;
      }

      public void ejbCreate() {
           accountDB = (DataSource)ctx.lookup(
                            "jdbc/accountDB");

      }
      public void ejbActivate() { }
      public void ejbPassivate() { }
      public void ejbRemove() { }

      public void setAccountDeposit(int empId,
                                    double deposit) {
        ...
        Connection conn = accountDB.getConnection();
        ...
      }
   ...
}
```

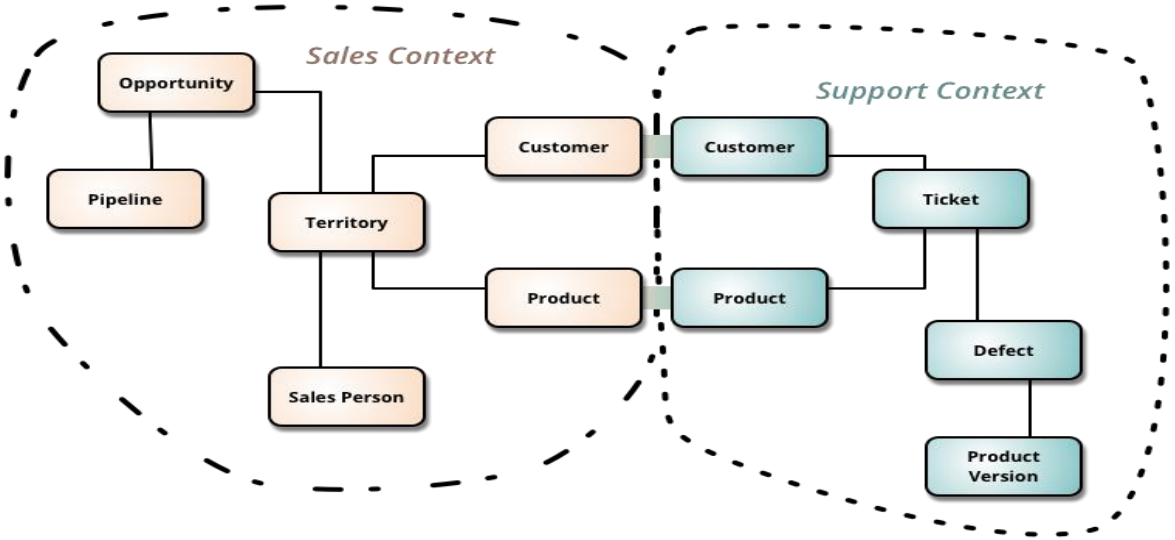
Java 클래스

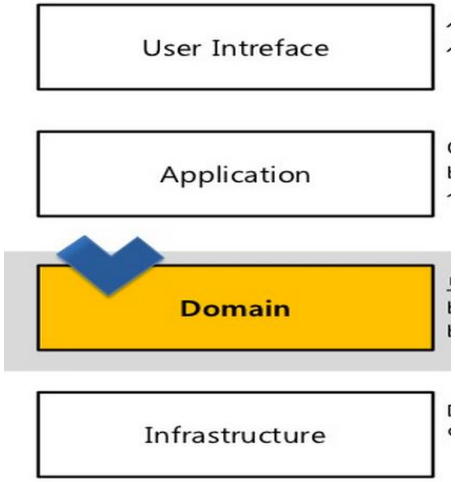
```
@Stateless
public class AccountBean implements Account
{
      @Resource private DataSource accountDB;

      public void setAccountDeposit(int customerId,
                                    double deposit) {
        ...
        Connection conn = accountDB.getConnection();
        ...
      }
   ...
}
```

출처: http://www-01.ibm.com/support/knowledgecenter/SS8PJ7_9.1.0/com.ibm.javaee.doc/topics/ceib3veib21.html?cp=SS8PJ7_9.1.0%2F1_1_0_1

Domain-Driven Design

1. Using 'Ubiquitous Language' so that the domain expert and developer can communicate





Metaworks

1. Metadata Oriented Application Framework
2. Developed by Jinyoung Jang in 1998
3. Inspired from the Adaptive Object Models, OMG Reflection, and MDA
4. Application Component Generation on the fly from metadata
5. Metaworks1: Generates database query and action, Simple Web Forms generation. written in ASP - 1998
6. Metaworks2: Generates Swing User interfaces written in Java, is adopted in uEngine Process Designer – 2003
7. Metaworks3: Generates Dynamic Web Uis. Metadata definition by annotation, is adopted in Process Codi SNS, Web based BPMN Modeler and the Cloud IDE of OCE garuda – 2011

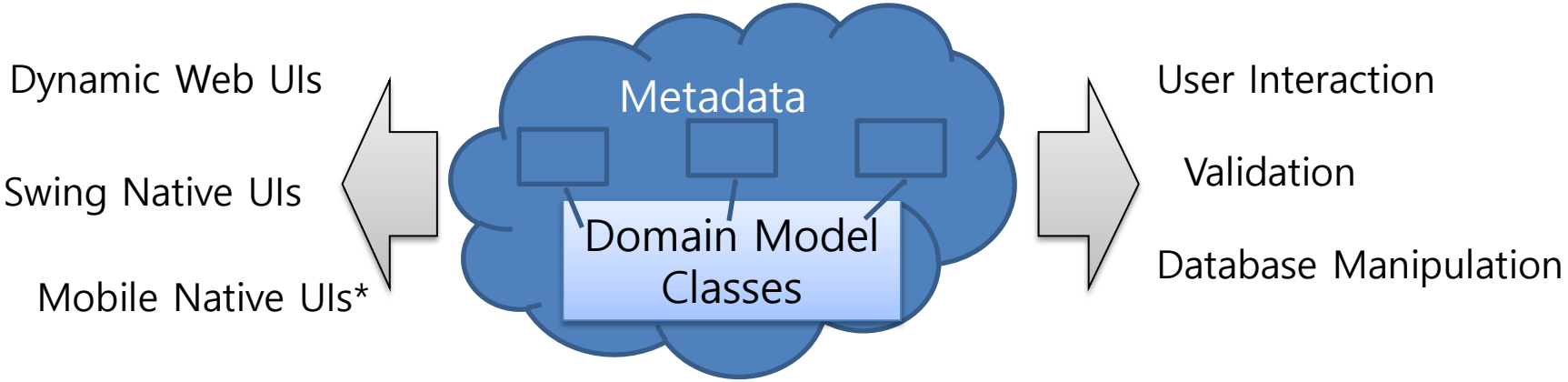
Metaworks

So Metaworks3…

1. Is A POJO framework
2. Encourages the Domain-Driven Design
3. Especially for developing model-driven applications

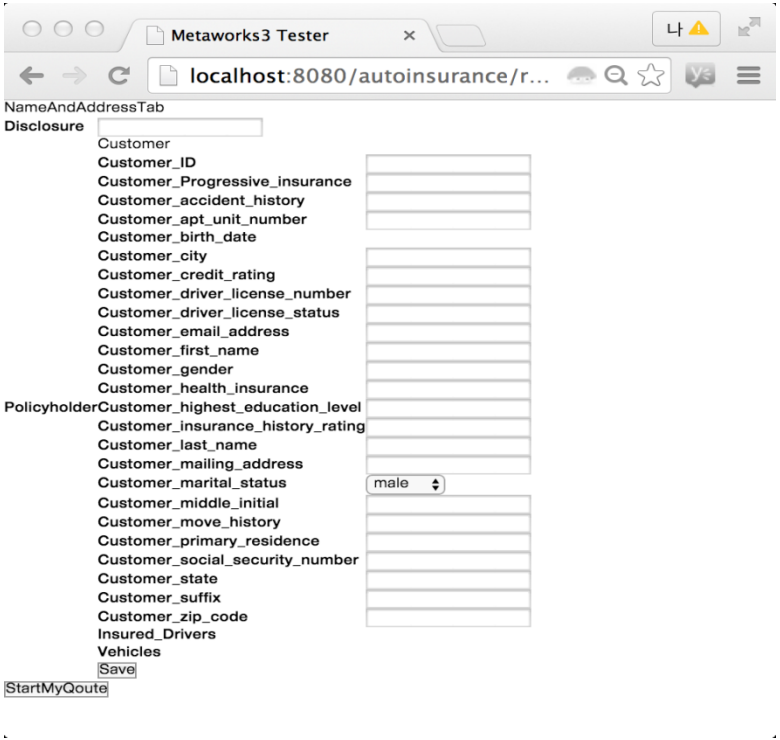
Metaworks3 for early prototyping

- Metaworks version 3 generates Dynamic Web Applications on the fly from your domain class model with minimal technical details.
- By reading the metadata declared by class model itself and the additional annotations.
- Metaworks doesn't generate any source codes so doesn't requires any round-trip job for source codes while MDA need such a job.



Creating a default application

- By adding some annotations and control code to your model, metaworks3 will generates the runnable application



Running generated UIs

Accessible with URL patterns:

- http://localhost:8080/autoinsurance/runner.html?classname=**<ClassName>**
  - Start Page

    http://localhost:8080/autoinsurance/runner.html?classname=ui.NameAndAddressTab

  - Test for Customer

    http://localhost:8080/autoinsurance/runner.html?classname=autoinsurance.Customer

Customizing the default application

- To make up the face, add EJS template file for each classes.

Provide Policyholder Name & Address

First Name :
Middle initial :
Last Name :
Suffix : Mr.
Mailing Address :
Apt./Unit # :
City : , MO
Zip code :
Check this box if this is a P.O.Box or military address
Data of birth : mm / dd / yyyy

Information Disclosure

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the U SPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 3 7 CFR 1.14. This collection is estimated to take 1 hour to complete, including gathe ring, preparing and submitting the completed application form to the USPTO. Time

OK - Start my Quote

No Thanks

NameAndAddress.ejs

Provide Policyholder Name & Address

First Name : <%= fields.policyholder.here() %>

Middle initial :

Last Name :

Suffix : Mr.

Mailing Address :

Apt./Unit # :

City : , MO

Zip code :

Check this box if this is a P.O.Box or military address

Data of birth : mm / dd / yyyy

Information Disclosure

This collection of information is required by 37 CFR 1.97 and 1.98. The information is required to obtain or retain a benefit by the public which is to file (and by the U SPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 3 7 CFR 1.14. This collection is estimated to take 1 hour to complete, including gathe ring, preparing and submitting the completed application form to the USPTO. Time

<%= methods.startMyQuote.here() %>

Get an MyQuote

No Thanks

Policyholder.ejs

First Name : <%= fields. customer_first_name.here() %>

Middle initial :

Last Name :

Suffix : Mr.

Mailing Address :

Apt./Unit # :

City : , MO

Zip code :

Check this box if this is a P.O.Box or military address

Data of birth : mm / dd / yyyy

```
@Range(options =
{ "Mr.",
 "Mrs. ",
 "Miss" },
values =
{ "Mr.",
 "Mrs. ",
 "Miss" })
```

Add vehicles to Your Quote

Add Vehicle(s) Using Pulldown Menus

Vehicles.java

Type : Auto/Van/Truck/SUV

Year : 2015

Make : Volvo

Model : S60

Add This Vehicle

1 Vehicle On Your Quote (order does not matter)

<%= VehiclesListFace %>

2009 Volvo S60

Edit | Remove

< Back

Next

Provide Details

DriversTab.java

<%= fields.policyholder.here() %>

Gender : Male Femail

Marital status : 미혼

Primary residence : 서울

Current U.S or Canadian License status : Valid

Social Security Number - -

Highest level of education : 대학원

Have you moved in the last 60 days? Yes No

Has this driver had any :

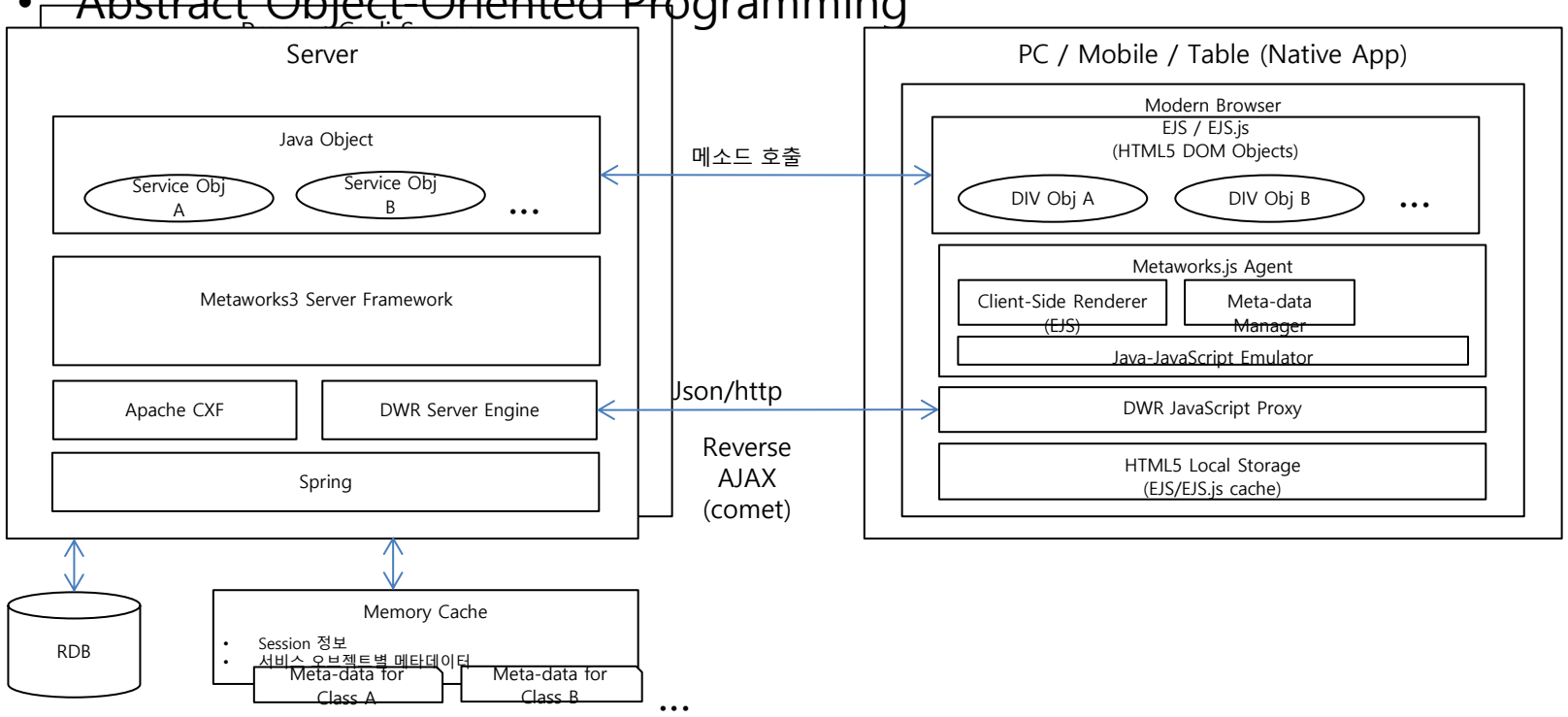
- At Fault Accidents in the last 5 years, or
- Not At Fault Accidents, Claims or Violations in the last 3 years?

Yes No

< Back Next

Metaworks Application Architecture

- Client-Side Rendering and JSON via HTTP
- Client Object Controlling through Reverse AJAX
- Abstract Object-Oriented Programming



Development Process

1. (Optional) UML을 이용한 설계
2. 도메인 자바 코드의 생성
3. 타 POJO Framework의 적용 - JPA를 통한 ORM 생성, JAX-RS를 통한 서비스 공개
4. Metaworks3 Framework의 적용을 통한 Default Application 의 생성
5. UI 구체화 / 커스터마이징
6. 1~5 반복

Separating domain language from technical implementation

```
@ServiceMethod(callByContent=true)
public VehiclesTab startMyQoute() throws org.orm.PersistentException {

        PersistentTransaction t = autoinsurance.MetaworksapptemplatePersistentManager.instance().getSession().beginTransaction();
        try {

            getPolicyholder().save();

            VehiclesTab vehiclesTab =  new VehiclesTab();
            vehiclesTab.setPolicyholder(getPolicyholder());

            t.commit();

            return vehiclesTab;
        }
        catch (Exception e) {
            t.rollback();

            throw new PersistenceException("고객정보입력시 오류", e);
        }
}
```

What if we can separate like this:

```
@Component
public class NameAndAddressTab {

....

@ServiceMethod(callByContent=true)
@NeedTransaction
public VehiclesTab startMyQoute() throws org.orm.PersistentException {
        getPolicyholder().save();

        VehiclesTab vehiclesTab =  new VehiclesTab();
        vehiclesTab.setPolicyholder(getPolicyholder());

        return vehiclesTab;
}
```

Aspect Oriented Programming: Advice on annotation

```
package framework;

import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.orm.PersistentException;
import org.orm.PersistentTransaction;

@Aspect
@Component
public class TransactionAdvice {

    @Before("@annotation(framework.NeedTransaction)")
    public void initiateTransaction() throws PersistentException {
        System.out.println("start tx");
        PersistentTransaction t = autoinsurance.MetaworksapptemplatePersistentManager.instance().getSession().beginTransaction();
    }

    @AfterReturning("@annotation(framework.NeedTransaction)")
    public void commitTransaction() throws Exception {
        System.out.println("commit");
        autoinsurance.MetaworksapptemplatePersistentManager.instance().getSession().getTransaction().commit();
    }

    @AfterThrowing("@annotation(framework.NeedTransaction)")
    public void rollbackTransaction() throws Exception {
        System.out.println("rollback");
        autoinsurance.MetaworksapptemplatePersistentManager.instance().getSession().getTransaction().rollback();
    }
}
```

applicationContext.xml

```
<context:annotation-config />
<aop:aspectj-autoproxy />

...
<context:component-scan base-package="autoinsurance" />
<context:component-scan base-package="ui" />
<context:component-scan base-package="framework" />
```

Or

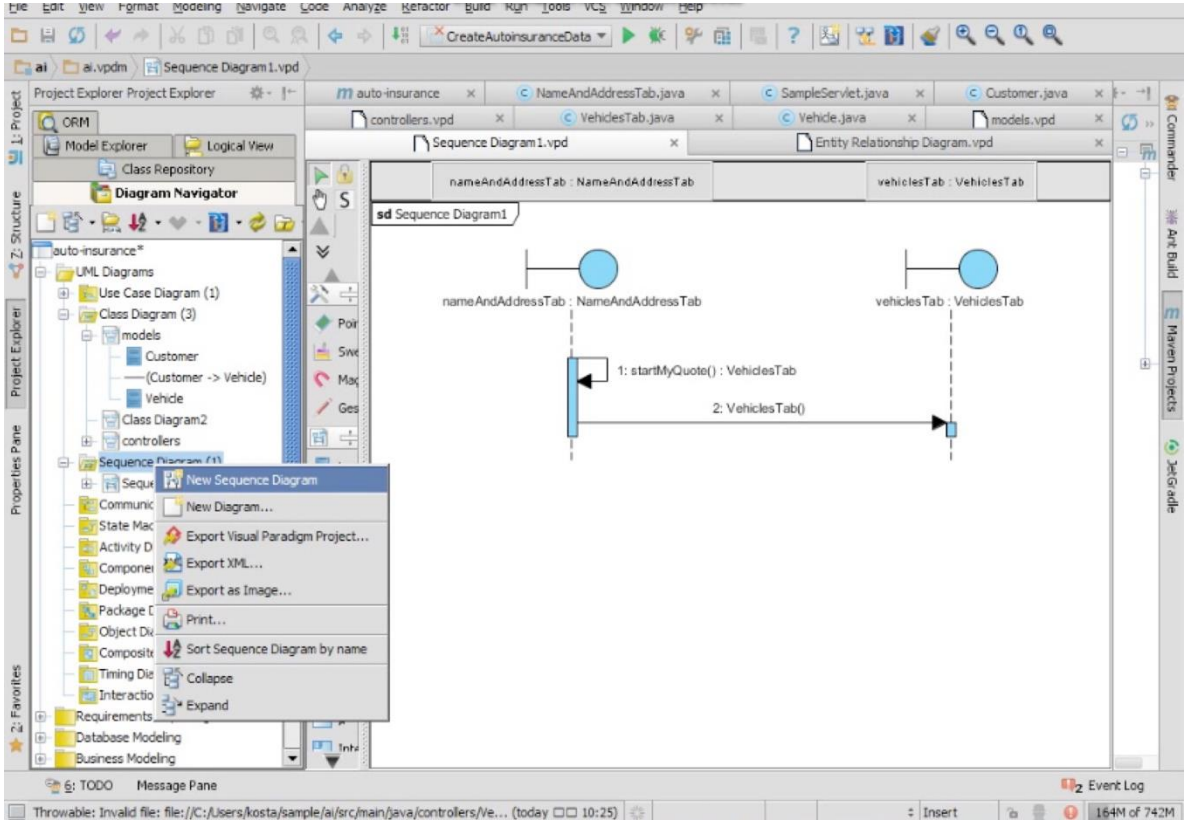
```
<bean id="transactionAdvice" class="framework.TransactionAdvice"/>
<bean id="nameAndAddressTab" class="ui.NameAndAddressTab"/>
```

Talking in domain specific language

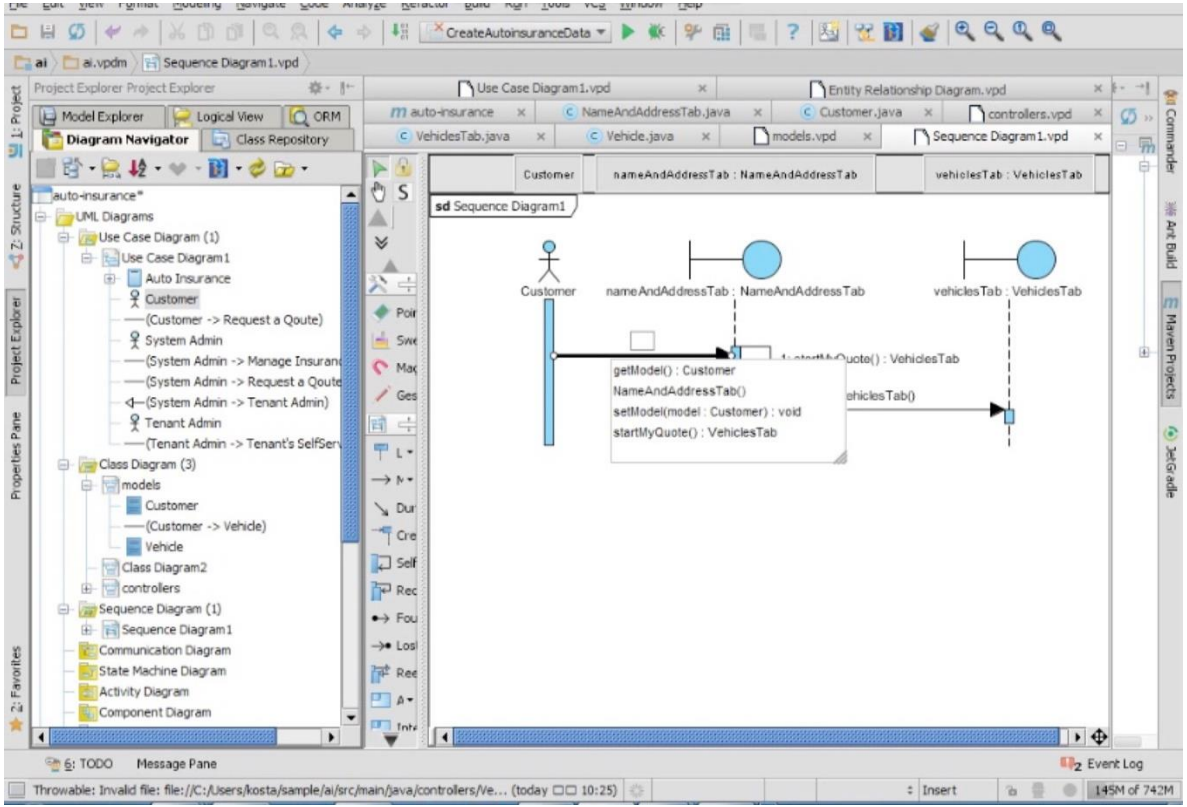
```
@Component
public class VehiclesTab {
....
    @ServiceMethod(callByContent=true)
    @NeedTransaction
    public YourRateTab next() throws Exception{

        for(Vehicle vehicle : getVehicles()){
            try {
                vehicle.setOwns(getPolicyholder());
                vehicle.save();
            } catch (PersistentException e) {
                throw new Exception("보유차량 정보 저장 중 오류가 발생하였습니다.", e);
            }
        }

        YourRateTab yourRateTab = new YourRateTab(getPolicyholder(), vehicles.get(0));
        return yourRateTab;
    }
    ....
}
```

EXERCISE: SEQUENCE DIAGRAM

Diagram Navigator > Sequence Diagram



Drag and drop message link



CASE STUDY: MODEL DRIVEN DEVELOPMENT ON SOFTWARE ENGINEERING STANDARD

OMG Standards and Tool Vendors

Fortune 500 기업의 70%가 고객

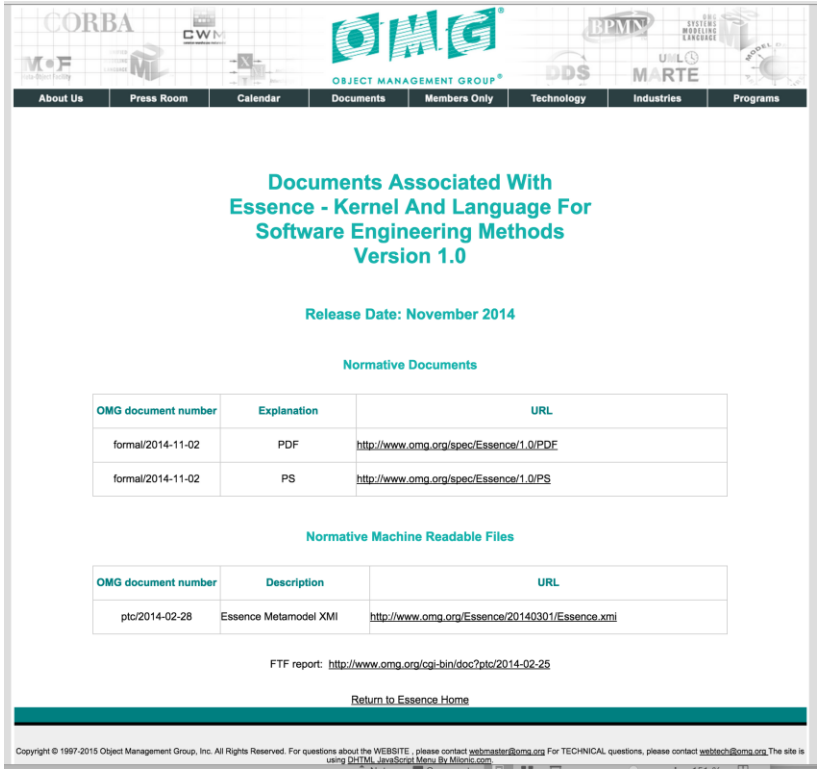
Name	Creator
AgileJ StructureViews	AgileJ
ArgoUML	Tigris.org
astah+	Change Vision, Inc.
ATL	Obeo, INRIA Free software community
Borland Together	Borland
BOUML	Bruno Pagès
CaseComplete	Serlio Software
Creately for UML	Cinergix, Pty Ltd
Dia	Alexander Larsson/GNOME Office
Eclipse UML2 Tools[4]	Eclipse Foundation
Enterprise Architect	Sparx Systems
GenMyModel	Axellience, INRIA
LucidChart	Lucid Software
MagicDraw	No Magic
Microsoft Visio	Microsoft
Modelio	Modeliosoft
MyEclipse	Genuitec
NClass	Balazs Tihanyi
NetBeans[6]	Oracle Corporation
Objecteering	Objecteering Software
objectiF	microTOOL
Open ModelSphere	Grandite
Papyrus	Commissariat à l'Énergie Atomique, Atos Origin
Poseidon for UML	Gentleware
PowerDesigner	Sybase
Prosa UML Modeller	Insoft Oy
Rational Rhapsody	IBM
Rational Rose XDE	IBM
Rational Software Architect	IBM
Rational Software Modeler	IBM
Rational System Architect	IBM
RISE	RISE to Bloome Software
Real Time Developer Studio	PragmaDev
Software Ideas Modeler	Dusan Rodina
StarUML	Plastic Software
Umbrello UML Modeller	Umbrello Team
UMLet	The UMLet Team
UModel	Altova
Visual Paradigm for UML	Visual Paradigm Int'l Ltd.
yEd	yWorks GmbH

Name	Creator
Activiti Modeler	Alfresco and the Activiti community
ADONIS (software)	BOC Information Technologies Consulting AG
Agiles BPMS & ECM	IMAGE Technology S.A.
Altova UModel	Altova
ARIS Express	Software AG
Bizagi BPM Suite	Bizagi
Bizagi Process Modeler	Bizagi
Bonita BPM	BonitaSoft
Borland Together	Borland
Camunda Modeler	Camunda
Cubetto	semture GmbH
Cubetto Toolset	semture GmbH
Enterprise Architect	Sparx Systems
Genexus WorkFlow	Artech
Yaoqiang BPMN Editor[8]	耀强基 (Bienta) (Sourceforge ID)
IBM BlueWorks Live	IBM
IBM Process Designer	IBM
iGrafx Flowcharter, iGrafx Process	iGrafx
INNOVATOR for Business Analysts	MID GmbH
jBPM	Redhat
LucidChart	Lucid Software Inc
MagicDraw	No Magic
Mega Process	MEGA
Microsoft Visio 2013	Microsoft
Modelio	Modeliosoft
OmniGraffle	Omni Group
QPR ProcessDesigner	QPR Software Plc.
QUAM	LINTRA Solutions GmbH
RunaWFE	Runa Consulting Group
SemTalk	Semtation GmbH
Signavio Process Editor	Signavio
Software Ideas Modeler	Dusan Rodina
SYDLE SEED Community	SYDLE Systems
TIBCO ActiveMatrix	TIBCO Software Inc.
Visible Analyst[14]	Visible Systems
yEd	yWorks

15,000,000 고객

프로젝트 목표

OMG 의 SW 공학 관련 표준인 Essence 의 모델을 준수하는 제품의 개발.



About Us | Press Room | Calendar | Documents | Members Only | Technology | Industries | Programs

Documents Associated With Essence - Kernel And Language For Software Engineering Methods Version 1.0

Release Date: November 2014

Normative Documents

OMG document number	Explanation	URL
formal/2014-11-02	PDF	http://www.omg.org/spec/Essence/1.0/PDF
formal/2014-11-02	PS	http://www.omg.org/spec/Essence/1.0/PS

Normative Machine Readable Files

OMG document number	Description	URL
ptc/2014-02-28	Essence Metamodel XMI	http://www.omg.org/Essence/20140301/Essence.xmi

FTF report: http://www.omg.org/cgi-bin/doc?ptc/2014-02-25

Return to Essence Home

Copyright © 1997-2015 Object Management Group, Inc. All Rights Reserved. For questions about the WEBSITE , please contact webmaster@omg.org For TECHNICAL questions, please contact webtech@omg.org The site is using DHTML JavaScript Menu By Milonic.com.

2.2 Practice Description Conformance

2.2.1 Overview

This conformance class applies to published practice descriptions defined using the Essence language, as specified in Clause 9. It provides a clear indication of what can be done with the practice description. One of three levels of conformance may be claimed for a practice description, as further described below.

NOTE: These practice description conformance levels are not associated with a practice; they are measure of the level of detail with which the practice has been described. It is quite possible for the same practice to be described at all the different conformance levels, for example Scrum could be described by different authors at different conformance levels. It is also possible for teams to use practices that are described at different conformance levels, for example a team could have their much used development and requirement practices at level 3 as these areas are important for them to monitor and track, and their project kick-off practices at level 1 as it is not as important to track their progress and they are typically only performed once by the team.

2.2.2 Level 1: Narrative

Practice descriptions defined at this conformance level use the conceptual elements of the Essence language as a framework for structuring their text. All of the elements in the practice are expressed correctly according to the language; for example all the work products appear as work products and all the activities appear as activities. Beyond this simple classification of the elements in the practice there are no other constraints or invariants.

Once published practices at this level can be referenced by other practices but cannot be exchanged between tools or automatically composed with other practices. Practices described at this level are typically just free format text and there is no XMI interchange format for sharing or composing them.

2.2.3 Level 2: Practice Description Interchange

Practice descriptions defined at this level use the full expressive quality of the language. Everything is typed properly and uses any applicable language element attributes and associations correctly; for example, all the elements will have names and brief descriptions conformant with the language rules and all associations between the elements will be queryable and traversable.

Level 2 practices can be exchanged between tools in XMI. This formal use of the language allows the practices to be composed with the kernel and other practices. Practice descriptions at this level are highly structured and will require specialist authoring or modeling tools to produce.

Level 2 practice descriptions add rigor and XMI interchange to Level 1. This provides the consistency and robustness to all tools to “do things” with them. They can read, manipulate, and compose the practices but a person is needed to “action” the resulting composition.

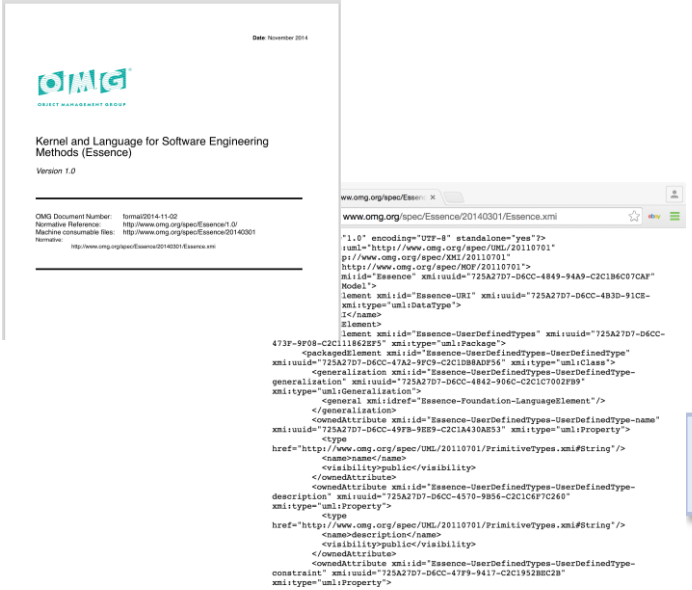
2.2.4 Level 3: Practice Actionable and Trackable

Practice descriptions defined at this level use the full power of the language to ensure they are prepared to be automatically actioned and tracked. For example there will always be an Alpha with a fully defined state machine with a complete set of checklists either contained in, or extended by the practice and all activities will be clearly related to the Alpha state progressions that they enable.

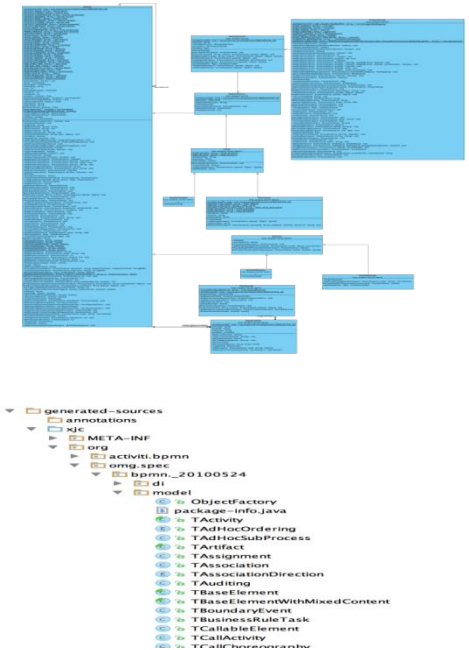
Generate from OMG Specifications

모델은 사람이 읽을 수 있는 pdf 형식과 기계 (모델링툴)가 읽을 수 있는 xmi 로 공개됨

Essence Standard model in XMI



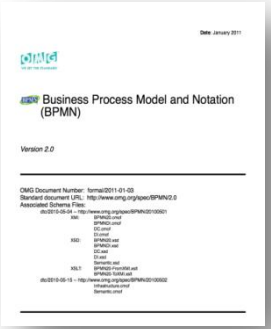
Essence Standard model in Java

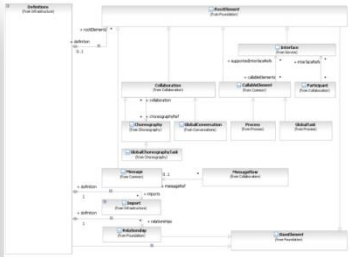


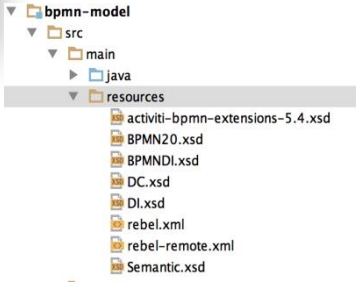
Generate from OMG Specifications

해당 과제에서는 OMG 의 또다른 표준인 BPMN 또한 지원해야 함. BPMN은 XML Schema 의 형태로 모델이 제공됨. 이는 jaxb 프레임워크로 자바로 생성한 후에 모델링 도구에서 Import 함.

BPMN Standard model in XSD

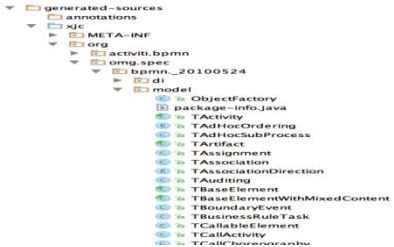






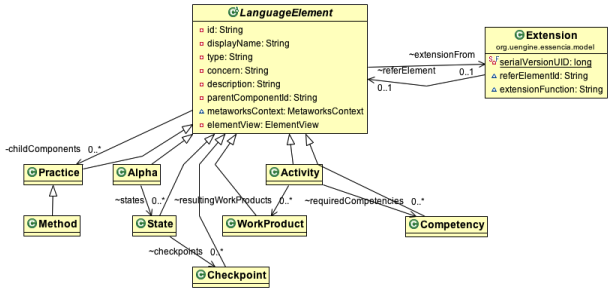
JAXB

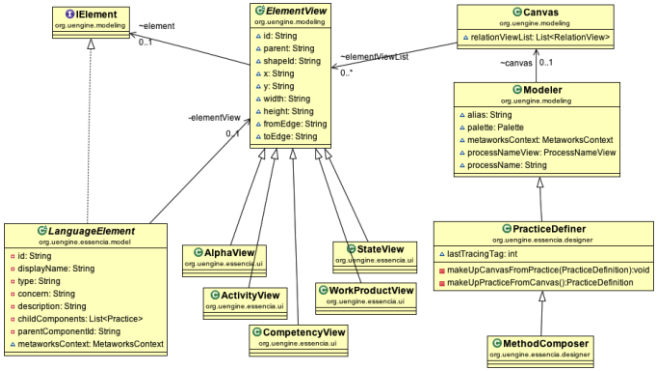
BPMN Standard model in Java

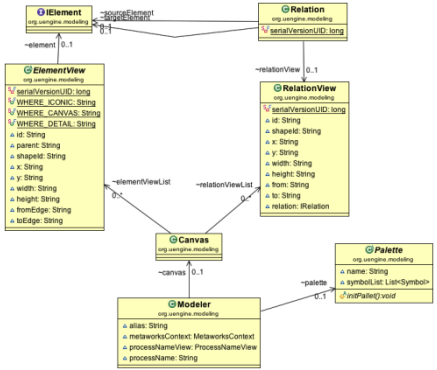


모델링 툴에서 읽어들인 Essence 모델

빠른 프로토타입을 만들기 위하여 초기에는 모델의 부분만을 추출하여 작은 프로젝트를 실시함.

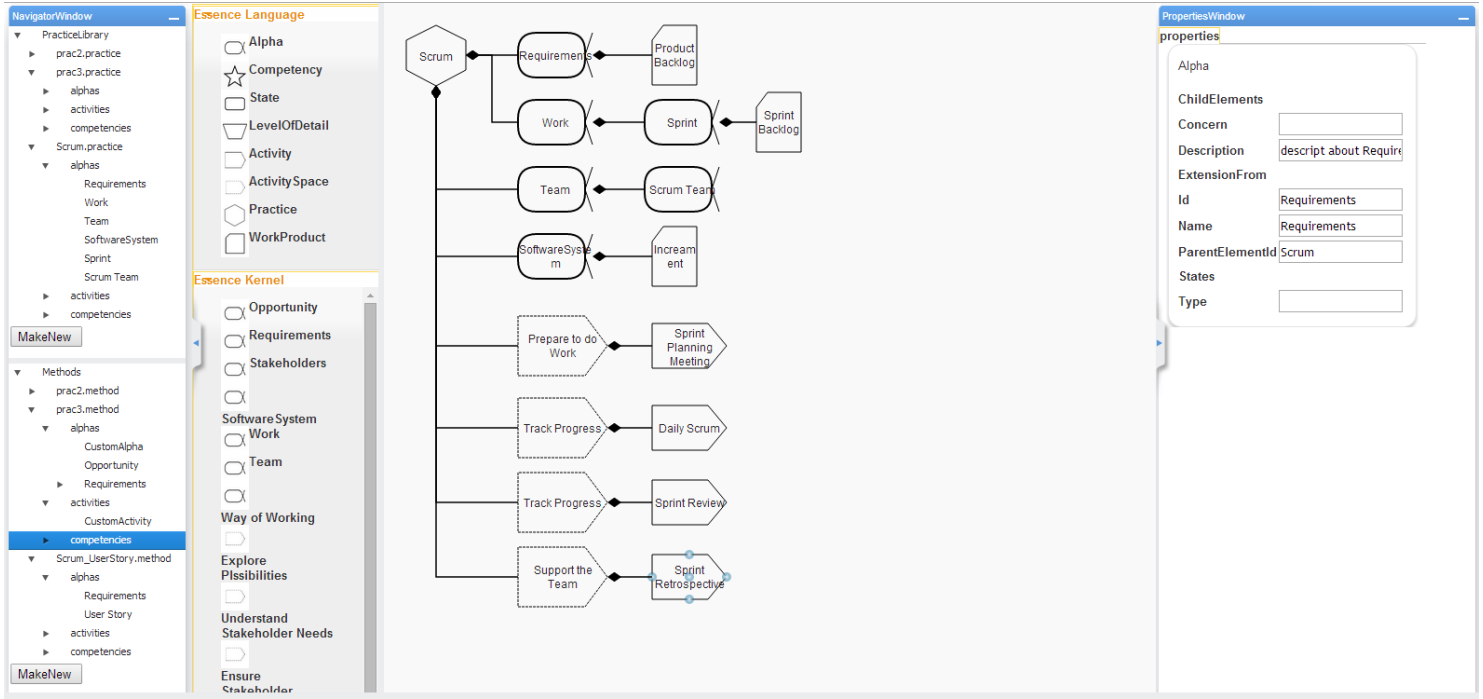






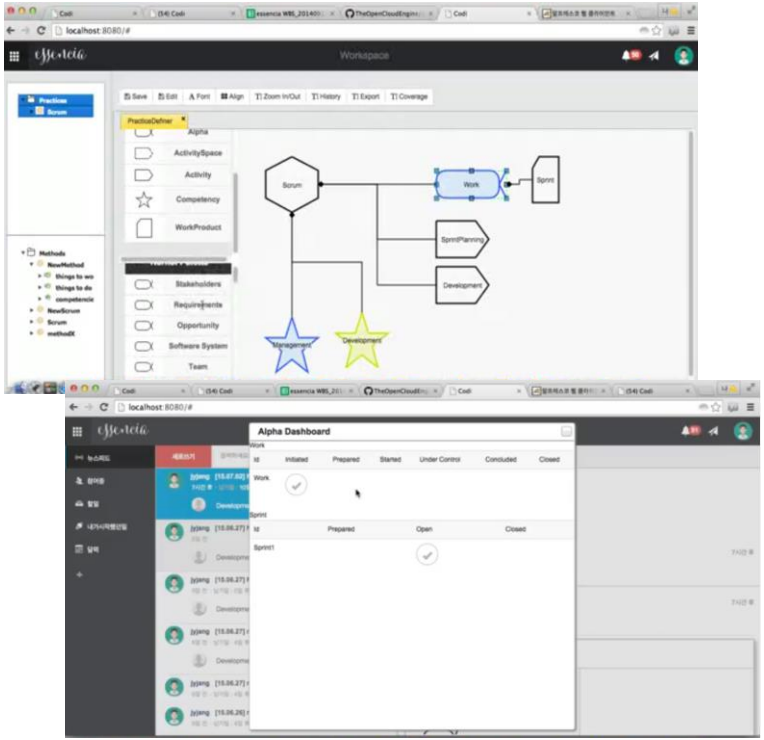
Essencia-Mini – Early Prototyping

모델 기반의 개발은 빠른 프로토타이핑이 가능하여 고객과 빠른 커뮤니케이션이 가능하다



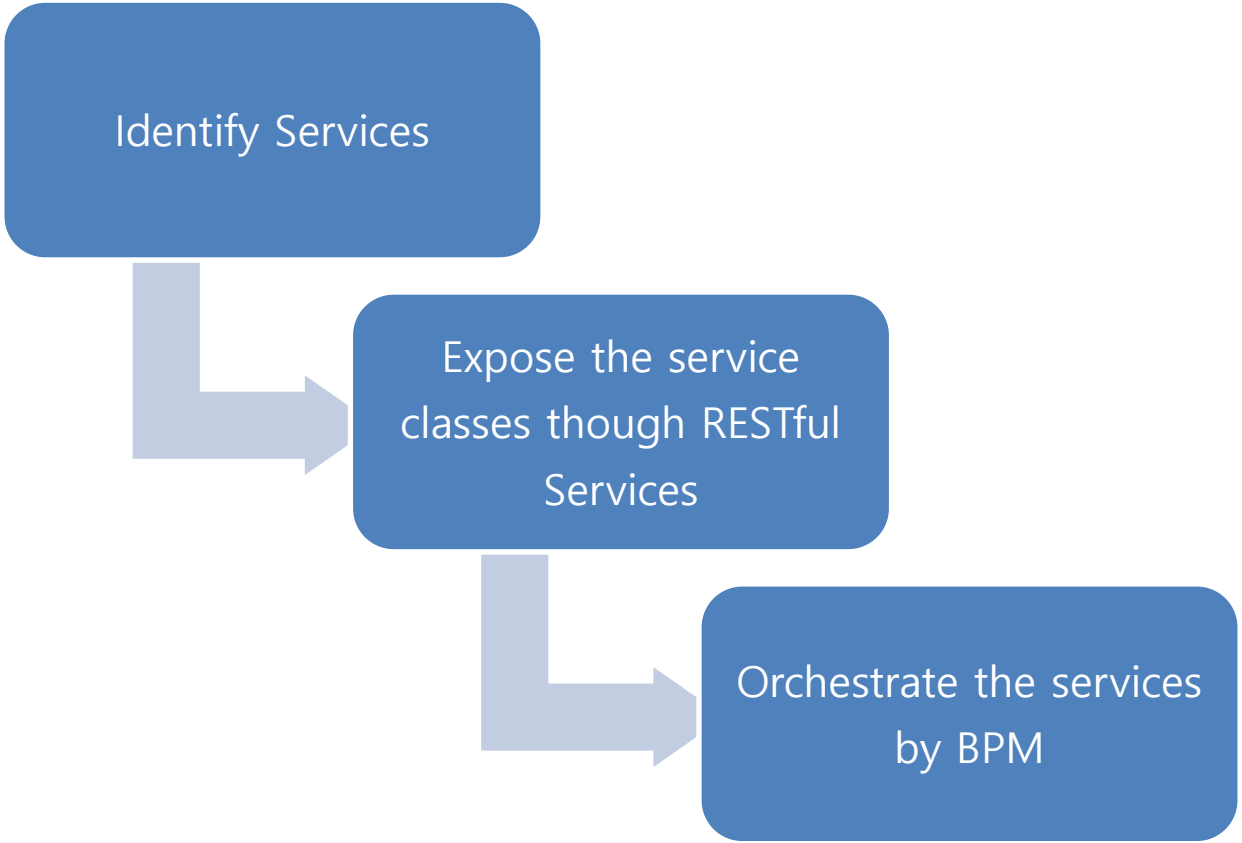
구현 현황

모델기반으로 완성한 기본 애플리케이션에서 빠르게 시작하여 지속적으로 고객과 논의해가면서 디자인 개선 작업, 구체화 해나가면서 시장의 Acceptance 를 이끌어냄



Day-2: Make the Application As A Service

Overview


Identify Services
Expose the service classes though RESTful Services
Orchestrate the services by BPM

Review: Service Identification

Use Case	Included Use Case	Customer	Policyholder	Insurance History	Credit History	Driver	Vehicle	Insurance Quote	Snapshot Info	Coverage Option	Insurance Policy	Payment Method	Payment	Insurance Product	Coverage Item	Underwriter
1. Determine the customer and the vehicle to be insured		U					C	C								
	5. Create and update customer	C														
2. Evaluate customer eligibility for the selected insurance product	6. Select insurance product							U								
		C	C	C	C	C	U									
	7. Evaluate customer eligibility							U								
3. Determine insurance coverage and premium								U	C	C						
	8. Calculate premium							U								
4. Get payment and send the signed policy		U									C					
	9. Get payment											C	C			

Select the product in UC 2

Expand the scope of UC 5

Get driver license numbers in UC 2

Services

- Customer Relationship Management
- Policyholder Management
- Vehicle Info Management
- Insurance Policy Management
- Payment Management

Service Specification for Insurance Policy Management

Service Name: Insurance Policy Management			
Service Interface	**Name**	**Tables (Owned)**	**Use Cases**
	Generate a Quote Number	Insurance Policy	Use Case 1 (Partially used)
	Select product based on customer and vehicle information	Insurance Policy	Use Case 6
	Evaluate customer eligibility for the insurance product	Insurance Policy	Use Case 7
	Determine insurance coverage	Insurance Policy Coverage Attribute Value	Use Case 3
	Calculate premium	Insurance Policy	Use Case 8
	Produce the insurance policy	Insurance Policy	Use Case 4 (Partially used)
Communication Protocol		WSDL, SOAP, ACORD	
Consumer Capabilities Required		Java Script or AJAX	
Provider Capabilities Required		Web Services, Java, Relational DBMS	
Service Objectives and Metrics		Number of customers applying an auto insurance online per month > 100 Number of online applicants signing a contract per month > 50	
Non-Functional Quality Req'ts		1. Average time to process an online insurance application until the issuance of the insurance policy should be less than 30 minutes. 2. 99.95 availability	
Service Policies		Ensure privacy and security of information provided by online applicants	
Service Dependency		Customer Relationship Management, Vehicle Info Management	
Service Domain Model		Refer to AOM-Generalization Model	

Process Data

3. Input/Output Data of Process Tasks

Process	Subprocess	Process Task	Input Data	Output Data
Auto Insurance Online Sales Process	Select product based on customer and vehicle information	Get customer demographic information	Customer (Policyholder) Name [First Name, Middle Initial, Last Name, Suffix], Address [Mailing Address, Apt Unit #, City, Zip Code, PO Box/Military Address Type], Birth date	
		Register customer		Customer ID
		Receive driver info from DMV	Driver license number, VIN, Vehicle Type, Vehicle Year, Vehicle Make, Vehicle Model	
		List owned vehicles		
		Ask the customer to enter a vehicle		
		Determine the vehicle to insure		
		Select insurance product		Product code
		Register quote		Quote Number

Exposing a service

- Through Web Services / SOAP
- Through RESTful Services / JSON
- You can directly expose your services using @Annotations since there are so many POJO-based Web Service Frameworks such as:
  - Apache CXF
  - Apache Wink
  - Jersey
  - Spring MVC

Apache CXF: What is it?

What is Apache CXF and what does it provide?

- An open-source web services framework
- Support for web service standards and JSR APIs.
- Tooling and configuration for building and consuming web services using the JAX-WS and JAX-RS frontend APIs.
- Spring namespace handlers for integration with the Spring Application Framework.

Apache CXF: A Robust Framework

Apache CXF provides robust support for producing and consuming web services:

- JSR APIs: JAX-WS, JAX-RS, JSR-181 annotations, SAAJ
- WS-* specifications for web service interoperability.
- A variety of message transports, protocol bindings, data bindings, and formats.
- Flexible, lightweight deployment in a variety of web application containers or stand-alone.
- Tooling for code and WSDL generation and validation.
- Multiple language support.

Apache CXF: Recommendations

The Java Stack recommends the following two basic feature sets for usin g CXF to produce and consume web services in Java applications:

- Option #1: JAX-WS, using the SOAP protocol over HTTP transport with JAXB data binding.
- Option #2: JAX-RS using REST over HTTP with JAXB, XML, or JSON da ta binding.

This training will focus on producing and consuming SOAP web services with JAX-WS. A future training session will cover JAX-RS.

SOAP vs. REST: An Overview

Both SOAP and REST are front-end technologies. That is, their ser vices are exposed within the view layer of an application.

SOAP

- Supports a variety of transports (HTTP, JMS, etc.)
- Integrates with a variety of web service standards.
- *Typically used to pass contractually structured data between ap plications.*

REST

- Simple point-to-point communication using well-established HT TP verbs, protocols, and standards.
- *Often used to faciliate dynamic HTML page creation.*

SOAP: Pros and Cons

Pros:

- Agnostic to language, platform, and transport.
- Designed for distributed environments and applications.
- Richer, more mature, standards-based support and integration.
- Built-in error-handling (faults).
- Extensible through integration with other transports and data b indings.

Cons:

- Heavier abstraction and steep learning curve.
- More verbose.
- Difficult to develop without tools.

REST: Pros and Cons

Pros:

- Agnostic to language and platform.
- Conceptually simpler to understand and develop.
- Less reliance on tools
- Closer in design and philosophy to the web.

Cons:

- Locked into the HTTP transport model. Cannot be scaled to non -HTTP communication models.
- Lack of standards support for security, policy, messaging, etc., making apps with richer requirements harder to develop.

* For a more detailed comparison of SOAP and REST, see the following excellent article: http://ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest or perform an internet search on "SOAP vs REST".

SOAP vs. REST: Summary

- Know your deployment environment and i nfrastructure.
  - Will your service be distributed?
  - Will communication need to pass through non-HTTP boundaries?
- Know your interoperability and scalability requirements.
  - Who will be consuming your service?
  - Will you need to support non-HTTP transports (e.g. JMS, SMTP, POP3) now or in the future?

JAX-WS: Defining a Service Endpoint

The JAX-WS specification uses Java annotations to define your web s ervice endpoint.

- CXF processes these annotations to produce your WSDL
- The annotations can be used to customize many aspects of how y our WSDL and web service schema are published.

JAX-WS: Annotations

@javax.jws.WebService

- Marks a Java interface as a Service Endpoint Interface.
- Marks a Java class as a web service endpoint implementation.

@javax.jws.WebMethod

- Optional annotation for customizing a SOAP web service operation

@javax.jws.WebParam

- Optional annotation for customizing the mapping of an individual parameter to a web service message part, or input parameter.

@javax.jws.WebResult

- Optional annotation for customizing how SOAP treats the output of a web operation.

JAX-WS: More Annotations

@javax.jws.soap.SOAPBinding

- Customizes the mapping of the web service onto the SOAP messa ge protocol.
- Its use is optional as the default settings are the recommended va lues and will make your web service the most portable.

Adding Dependencies

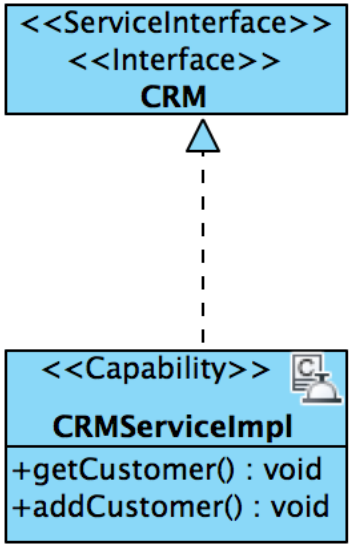
```
<dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxf-rt-frontend-jaxrs</artifactId>
        <version>3.0.3</version>
</dependency>
<dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxf-rt-frontend-jaxws</artifactId>
        <version>3.0.3</version>
</dependency>
<dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxf-rt-transports-http</artifactId>
        <version>3.0.3</version>
</dependency>
<dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxf-rt-transports-http-jetty</artifactId>
        <version>3.0.3</version>
</dependency>

<dependency>
        <groupId>org.apache.cxf</groupId>
        <artifactId>cxf-rt-bindings-soap</artifactId>
        <version>3.0.3</version>
        <exclusions>
                <exclusion>
                        <groupId>com.sun.xml.bind</groupId>
                        <artifactId>*</artifactId>
                </exclusion>
        </exclusions>
</dependency>

<dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-impl</artifactId>
        <version>2.2.10-b140310.1920</version>
</dependency>
<dependency>
        <groupId>com.sun.xml.bind</groupId>
        <artifactId>jaxb-core</artifactId>
        <version>2.2.10-b140310.1920</version>
</dependency>
<dependency>
        <groupId>org.apache.ws.xmlschema</groupId>
        <artifactId>xmlschema-core</artifactId>
        <version>2.0.1</version>
</dependency>
```

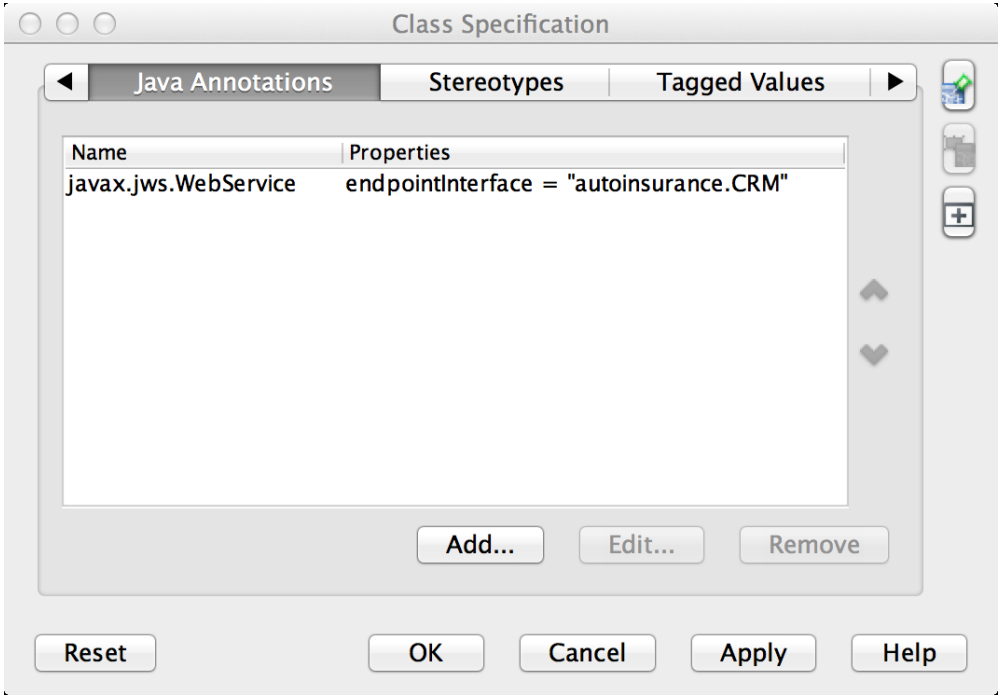
Adding CXF common servlets

#web.xml

```
<servlet>
    <servlet-name>CXFWebServices</servlet-name>
    <servlet-class>
    org.apache.cxf.transport.servlet.CXFServlet
</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>CXFWebServices</servlet-name>
    <url-pattern>/services/*</url-pattern>
  </servlet-mapping>
```

Exposing the CRM Service as a web service





JAX-WS: Defining a Service Endpoint

A Basic Example:

```
package autoinsurance;


@javax.jws.WebService(endpointInterface="autoinsurance.CRM")
public class CRMServiceImpl implements CRM {

        public void getCustomer() {
                throw new UnsupportedOperationException();
        }

        public void addCustomer() {
                throw new UnsupportedOperationException();
        }
}
```

Add the service to ApplicationContext.xml

```
<beans ...
xmlns:jaxws="http://cxf.apache.org/jaxws"
xmlns:jaxrs="http://cxf.apache.org/jaxrs"
...
xsi:schemaLocation="....
http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd
http://cxf.apache.org/jaxrs http://cxf.apache.org/schemas/jaxrs.xsd"
>

        <import resource="classpath:META-INF/cxf/cxf.xml" />
        <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />
        <jaxws:endpoint id="crm"
                implementor="autoinsurance. CRMServiceImpl"
        address="/crm" />
</beans>
```

The Web Services Exposed

- If your Web Services have been exposed successfully, you can find WSDL for the service:

localhost:8080/autoinsurance/services/crm?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
▼<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:tns="http://autoinsurance/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="CRMServiceImplService" targetNamespace="http://autoinsurance/">
  ▼<wsdl:types>
    ▼<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://autoinsurance/" elementFormDefault="unqualified"
     targetNamespace="http://autoinsurance/" version="1.0">
       <xs:element name="addCustomer" type="tns:addCustomer"/>
       <xs:element name="addCustomerResponse" type="tns:addCustomerResponse"/>
       <xs:element name="getCustomer" type="tns:getCustomer"/>
       <xs:element name="getCustomerResponse" type="tns:getCustomerResponse"/>
     ▼<xs:complexType name="addCustomer">
        <xs:sequence/>
      </xs:complexType>
     ▼<xs:complexType name="addCustomerResponse">
        <xs:sequence/>
      </xs:complexType>
     ▼<xs:complexType name="getCustomer">
        <xs:sequence/>
      </xs:complexType>
     ▼<xs:complexType name="getCustomerResponse">
        <xs:sequence/>
      </xs:complexType>
    </xs:schema>
  </wsdl:types>
  ▼<wsdl:message name="addCustomer">
     <wsdl:part element="tns:addCustomer" name="parameters"></wsdl:part>
  </wsdl:message>
  ▼<wsdl:message name="getCustomer">
     <wsdl:part element="tns:getCustomer" name="parameters"></wsdl:part>
  </wsdl:message>
  ▼<wsdl:message name="addCustomerResponse">
     <wsdl:part element="tns:addCustomerResponse" name="parameters"></wsdl:part>
  </wsdl:message>
  ▼<wsdl:message name="getCustomerResponse">
     <wsdl:part element="tns:getCustomerResponse" name="parameters"></wsdl:part>
  </wsdl:message>
  ▼<wsdl:portType name="CRM">
    ▼<wsdl:operation name="addCustomer">
       <wsdl:input message="tns:addCustomer" name="addCustomer"></wsdl:input>
       <wsdl:output message="tns:addCustomerResponse" name="addCustomerResponse"></wsdl:output>
     </wsdl:operation>
    ▼<wsdl:operation name="getCustomer">
       <wsdl:input message="tns:getCustomer" name="getCustomer"></wsdl:input>
       <wsdl:output message="tns:getCustomerResponse" name="getCustomerResponse"></wsdl:output>
     </wsdl:operation>
```

Configuring & using a Web Service Client

#applicationContext.xml

```
<jaxws:client id="crm" serviceClass="autoinsurance.CRM"
address="http://localhost:8080/autoinsurance/services/crm" />
```

#NameAndAddressTab.java

```
@Autowired
CRM crmService;

@ServiceMethod(callByContent = true)
public void save() throws PersistentException {
        crmService.addCustomer(this);

}
```

Proxy Pattern and Spring's DI

- Without changing the client's code, easy to change the real implementations
- Spring's dependency injection facility enables easy customization for various customers' requirements.

#applicationContext.xml for Remote CRM integration:

```
<jaxws:client id="crm" serviceClass="autoinsurance.CRM"
address="http://localhost:8080/autoinsurance/services/crm" />
```

#applicationContext.xml for Sugar CRM integration:

```
<bean id="crm" class="implementation.crm.SugarCRM"/>
```

Separating Requester/Provider machine

- Client’s Service Interface (CRM.java) should be shared or generated by the WSDL.
- Set the request URL in the client-side should be specified.
- Using UDDI, the services could be searched and then it can be dynamically bound.

Introduction to REST

REST stands for Representational State Transfer

- It is an architectural *pattern* for developing w eb services as opposed to a *specification*.
- REST web services communicate over the HT TP specification, using HTTP vocabulary:
  - Methods (GET, POST, etc.)
  - HTTP URI syntax (paths, parameters, etc.)
  - Media types (xml, json, html, plain text, etc)
  - HTTP Response codes.

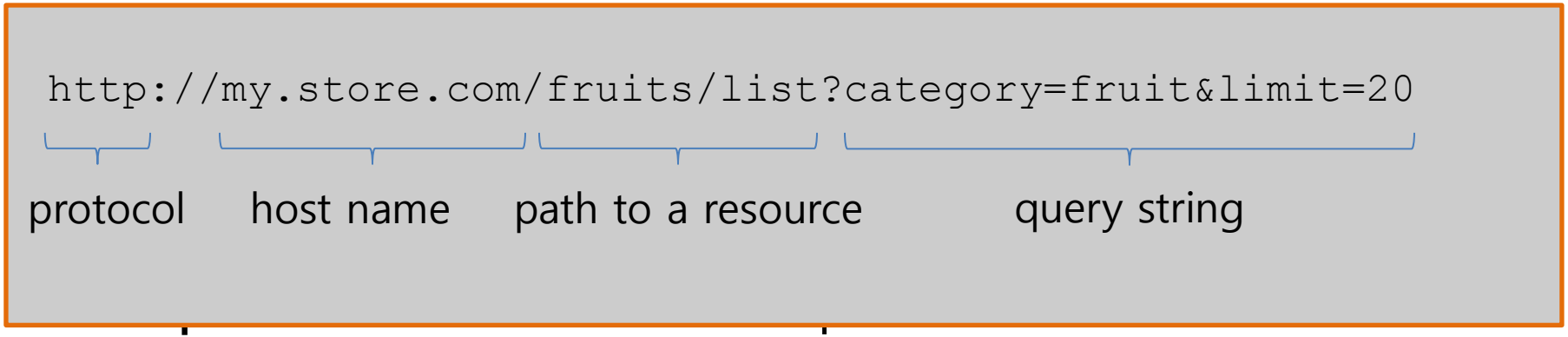
HTTP-REST Vocabulary

HTTP Methods supported by REST:

- GET – Requests a resource at the request URL
  - Should <u>not</u> contain a request body, as it will be discarded.
  - <u>May</u> be cached locally or on the server.
  - May produce a resource, but should not modify on it.
- POST – Submits information to the service for processing
  - Should typically return the new or modified resource.
- PUT – Add a new resource at the request URL
- DELETE – Removes the resource at the request URL
- OPTIONS – Indicates which methods are supported
- HEAD – Returns meta information about the request URL

HTTP-REST Vocabulary

A typical HTTP REST URL:



o process and respond to the request.

- The **host name** identifies the server address of the resource.
- The **path** and **query string** can be used to identify and customi ze the accessed resource.

JAX-RS Basics

An example REST service class:

```
package org.lds.tech.training.lab.ws;

import javax.ws.rs.GET;
import org.springframework.stereotype.Controller;

@Controller
public class HelloWebServiceRest {

    @GET
    public String sayHello() {
        return "Hello, World!";
    }
}
```

JAX-RS Basics

Example Spring configuration:

- Example location: WEB-INF/example-servlet.xml

```
<stack-rs:produce>
    <stack-rs:interfaces>
        <ref bean="helloWebServiceRest"/>
    </stack-rs:interfaces>
</stack-rs:produce>
```

- A reference to the JAX-RS annotated service bean is passed to the Stack RS “produce” namespace handler.
- Multiple service beans may be supplied under the Stack RS “interfaces” element.
- Each bean will be scanned by CXF for annotated resources that can be served up RESTfully.

JAX-RS Basics

An example WADL descriptor:

```
<application xmlns="http://wadl.dev.java.net/2009/02"
             xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <resources
      base="http://localhost:8080/example/Services/rest">
    <resource path="/">
      <method name="GET">
        <response>
          <representation mediaType="application/octet-stream">
            <param name="result" style="plain" type="xs:string"/>
          </representation>
        </response>
      </method>
    </resource>
  </resources>
</application>
```

JAX-RS Method Annotations

JAX-RS HTTP Method Annotations:

@GET	@POST
@PUT	@DELETE
@OPTIONS	@HEAD

- Applied to a Java method to bind it to an HTTP meth od.
- Only one HTTP annotation may be applied to a singl e Java method.
- Multiple Java methods may be given the same HTTP method annotation, assuming they are bound to diff erent paths.

JAX-RS @Path Annotation

- @Path annotations may be supplied to customize the reques t URI of resource.
- @Path on a class defines the base relative path for all resou rces supplied by that class.
- @Path on a Java class method defines the relative path for t he resource bound to that method.
- @Path on a method is relative to any @Path on the class.
- In the absence of @Path on the class or method, the resour ce is defined to reside at the root of the service.
- A leading forward slash (/) is unecessary as the path is alwa ys relative.

JAX-RS Annotations: Examples

```
@Path("example")
public class ExampleWebServiceRest {
    @GET
    public Item getItem(@QueryParam("itemId") Long id) {
          // @QueryParam example
    }

    @POST
    public Response editItem(@FormParam("itemId") Long id,
                             @FormParam("value") String value) {
          // @FormParam example
    }

    @GET
    @Path("{category}/{subcategory:[^/]*}")
    public List<Item> getItems(
        @PathParam("category") String category) {
        @PathParam("subcategory") String subcategory) {
          // @Path and @PathParam example
    }
}
```

JAX-RS Annotations

Examples of @Produces and @Consumes:

```
@Path("example")
public class ExampleRestService {

    @POST
    @Path("items")
    @Produces({"application/json", "application/xml"})
    @Consumes({"application/json", "application/xml"})
    public List<Item> editItems(List<Item> items) {
        // Does something and returns the modified list
    }
}
```

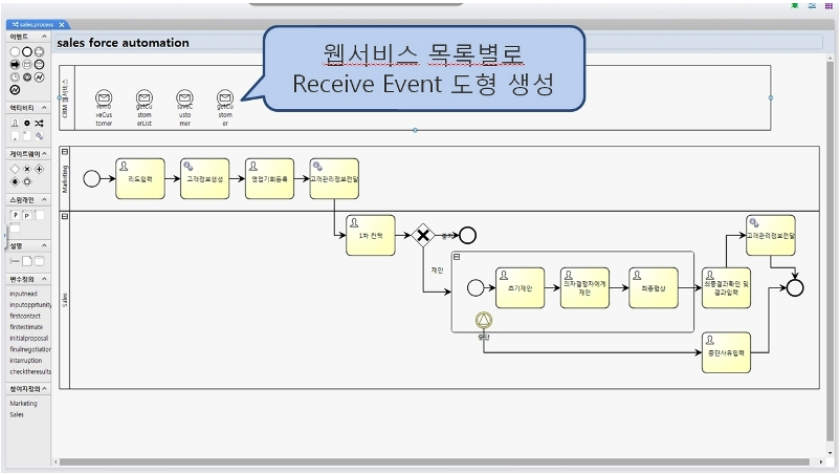
- The client *submits* JSON or XML content with the "Content-Type" header.
- The client *requests* either JSON or XML content through use of the HTTP "Accept" request header.

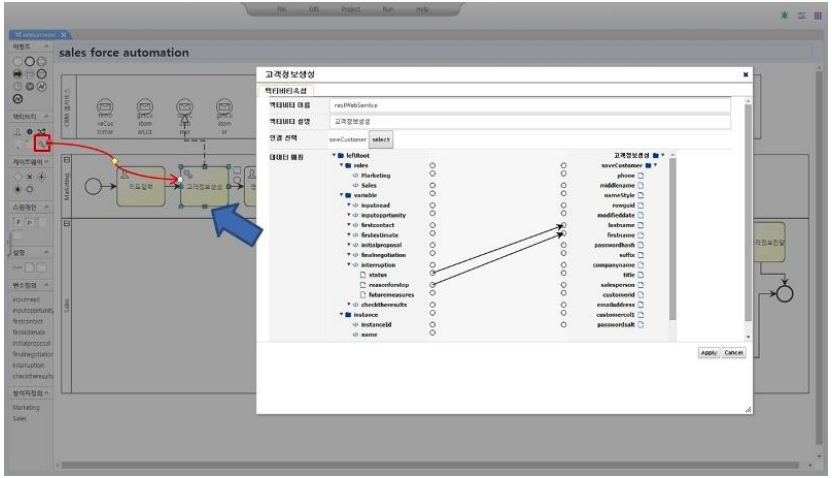
Operations behind the scene

- JSON(XML) serialization / deserialization
- Networking between Requester and Provider
- Dynamic method invocation

Service Orchestration with BPM

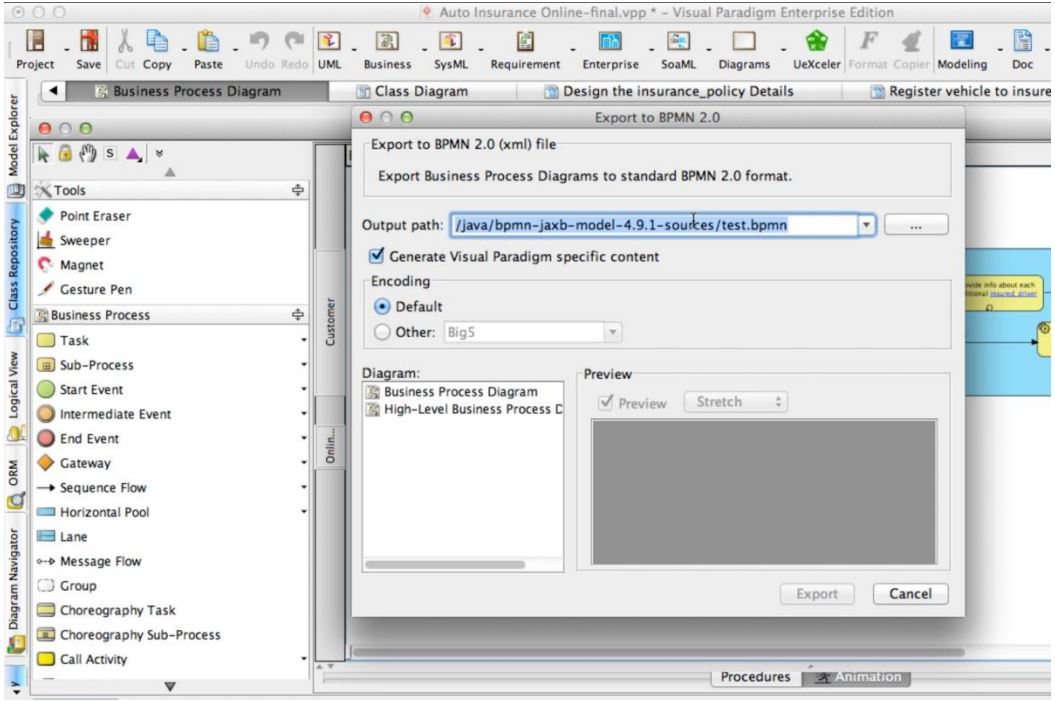
- uEngine BPMS is an open source software supporting BPMN 2.0
- Pools for service definition
- Using data mapper, detailed message inter-change between pools can be done





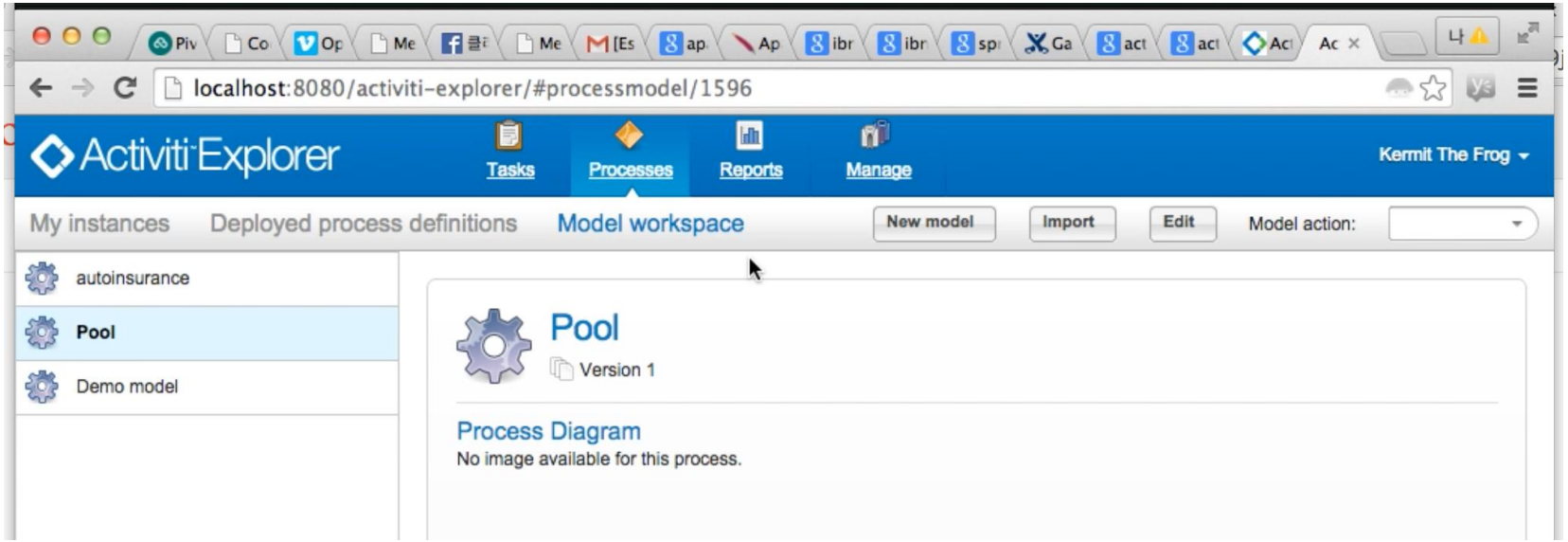
Export BPMN xml file from VP

- Exporting BPMN 2.0 xml
- File > Visual Paradigm Export > BPMN 2.0

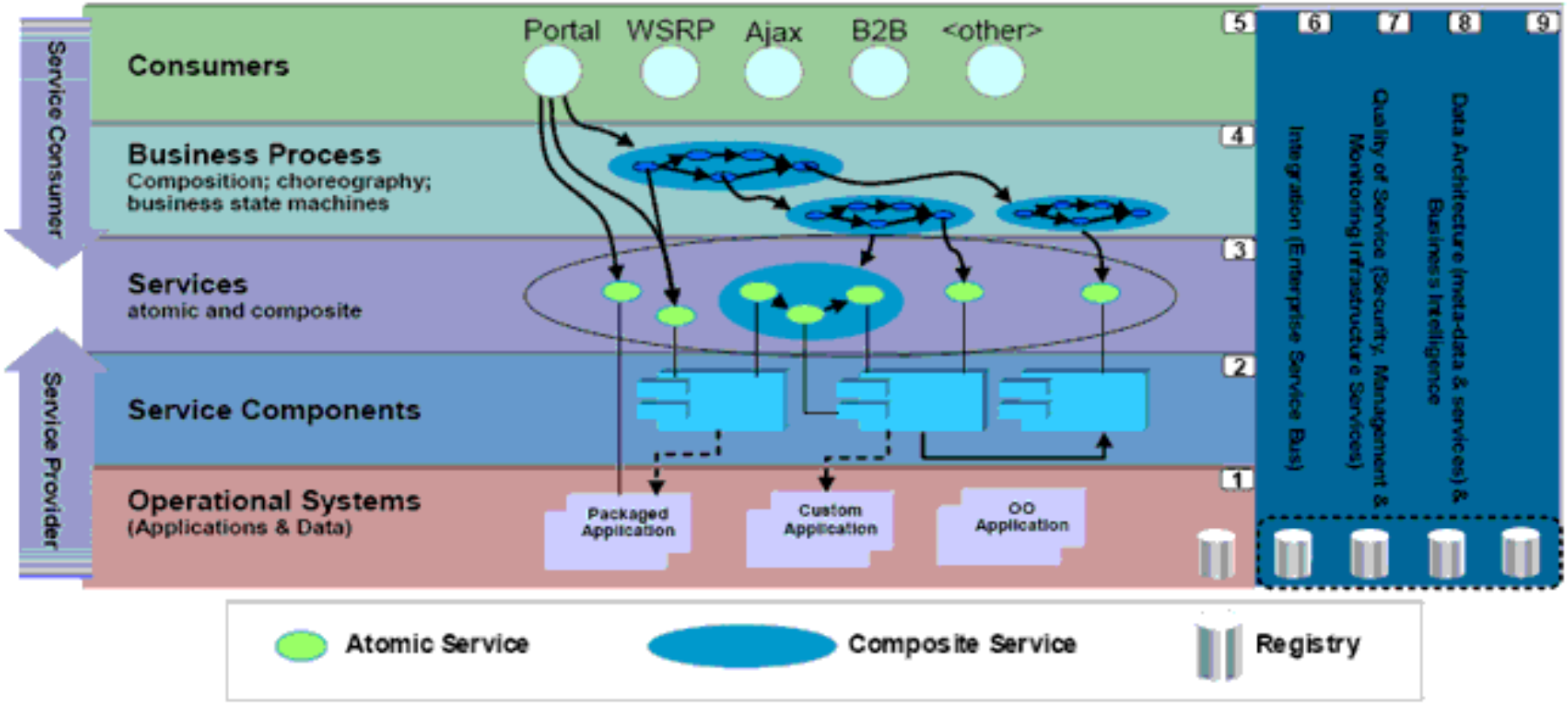


Importing BPMN xml to Activiti BPM

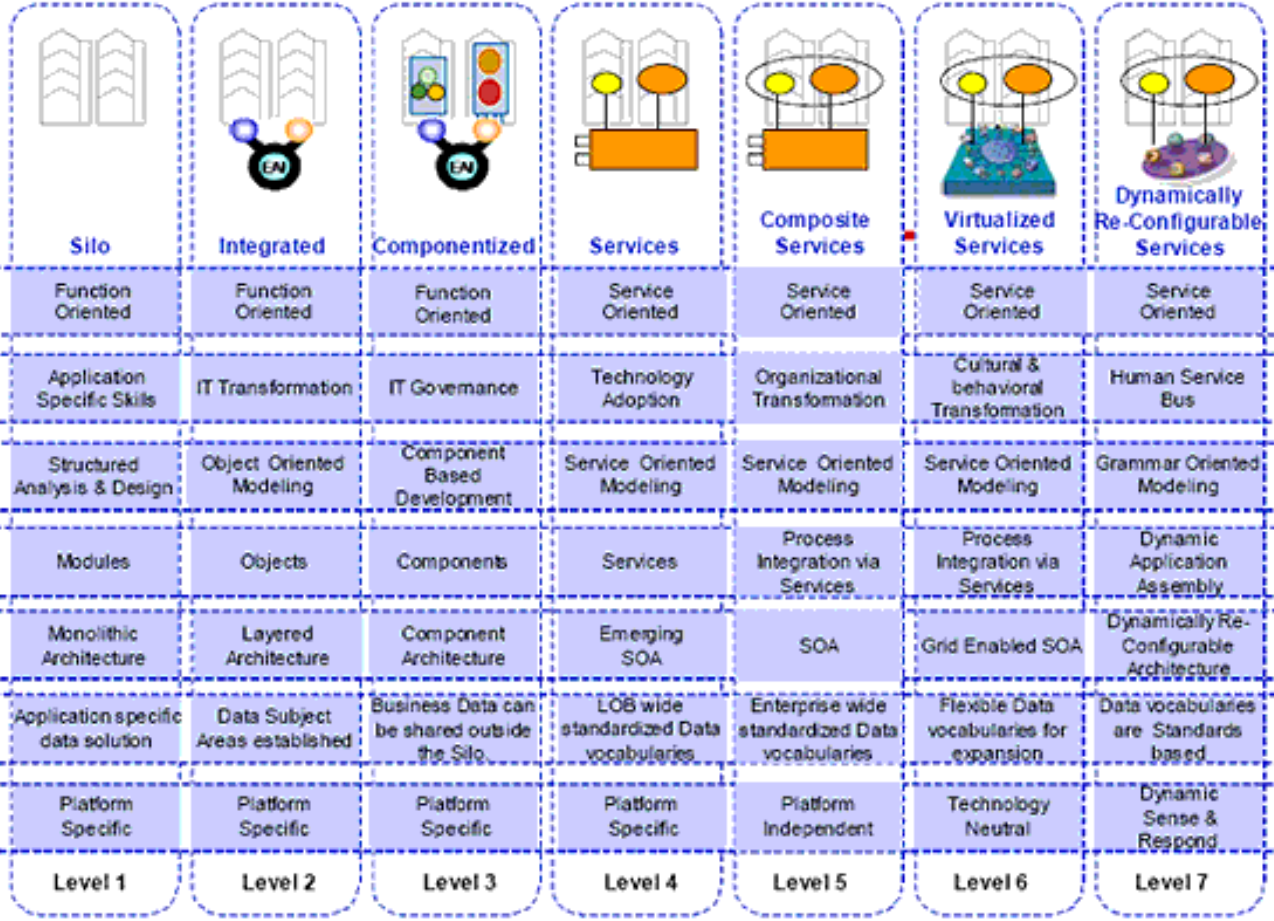
- Activiti BPM is an another open source BPMS supporting BPMN 2.0
- Processes > Model workspace > Import
- Select the BPMN xml file



Service Oriented Architecture Applied



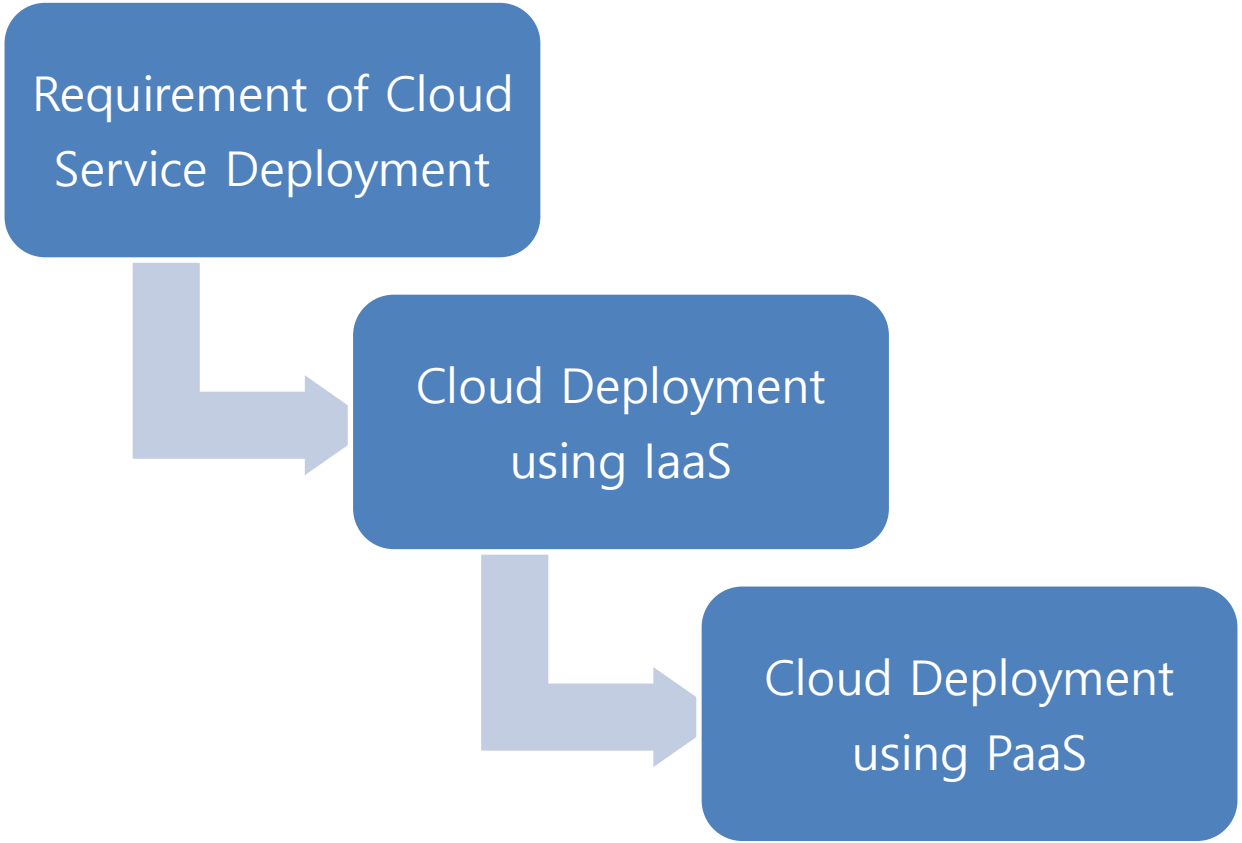
SOA Maturity Levels



	Silo	Integrated	Componentized	Services	Composite Services	Virtualized Services	Dynamically Re-Configurable Services
Business View	Function Oriented	Function Oriented	Function Oriented	Service Oriented	Service Oriented	Service Oriented	Service Oriented
Organization	Application Specific Skills	IT Transformation	IT Governance	Technology Adoption	Organizational Transformation	Cultural & behavioral Transformation	Human Service Bus
Methods	Structured Analysis & Design	Object Oriented Modeling	Component Based Development	Service Oriented Modeling	Service Oriented Modeling	Service Oriented Modeling	Grammar Oriented Modeling
Applications	Modules	Objects	Components	Services	Process Integration via Services	Process Integration via Services	Dynamic Application Assembly
Architecture	Monolithic Architecture	Layered Architecture	Component Architecture	Emerging SOA	SOA	Grid Enabled SOA	Dynamically Re-Configurable Architecture
Information	Application specific data solution	Data Subject Areas established	Business Data can be shared outside the Silo	LOB wide standardized Data vocabularies	Enterprise wide standardized Data vocabularies	Flexible Data vocabularies for expansion	Data vocabularies are Standards based
Infrastructure	Platform Specific	Platform Specific	Platform Specific	Platform Specific	Platform Independent	Technology Neutral	Dynamic Sense & Respond
	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7

Day-3: Cloud Deployment

Overview


Requirement of Cloud Service Deployment
Cloud Deployment using IaaS
Cloud Deployment using PaaS

Design factors on developing cloud applications

1. Don't code your application directly to a specific topology
2. Do not assume the local file system is permanent
3. Don't keep session state in your application
4. Don't log to the file system
5. Don't assume any specific infrastructure dependency
6. Don't use infrastructure APIs from within your application
7. Don't use obscure protocols
8. Don't rely on OS-specific features
9. Don't manually install your application

개발자가 할일.

2000 TODO

1. 아이디어 창출
2. 예산확보
3. 하드웨어 구입요청
4. 하드웨어 취득
5. 하드웨어 공간 확보 및 정돈
6. OS 설치
7. OS 패치 및 패키지 설치
8. 계정 생성
9. 프레임워크 디플로이
10. 미들웨어 디플로이
11. 테스팅 툴 디플로이
12. 테스팅 툴 테스트
13. 코딩
14. 멀티태넌시 처리
15. 서버 환경 설정(필요한 시 추가구매)
16. 생산완료
17. 런칭
18. 서버 추가구매 요청
19. 승인 응답 대기
20. 새로운 서버 디플로이 etc.

2010 TODO

1. 아이디어 창출
2. 예산확보
3. VM 요청
4. 프레임워크 디플로이
5. 미들웨어 디플로이
6. 테스팅 툴 디플로이
7. 테스팅 툴 테스트
8. 코딩
9. 멀티태넌시 처리
10. VM서버 환경 설정
11. 생산완료
12. 런칭
13. VM서버 추가구매 요청
14. 승인 응답 대기
15. 새로운 VM서버에 디플로이 etc.

By IaaS

2013 TODO

1. 아이디어 창출
2. 예산확보
3. 코딩
4. 테스트
5. 런칭

By PaaS

PaaS

A PaaS provides development services on cloud.

"Instance-on" Middleware and Platform Services

PaaS가 해주길 기대하는 것들

* 출처: IBM

PaaS의 유형들

- Application PaaS: OSS서비스에 집중
  e.g. 클라우드 파운드리, 오픈시프트

- Business PaaS: BSS on OSS (OSS+BSS)
  e.g. 포스닷컴, IBM SmartCloud, GAE, OCE

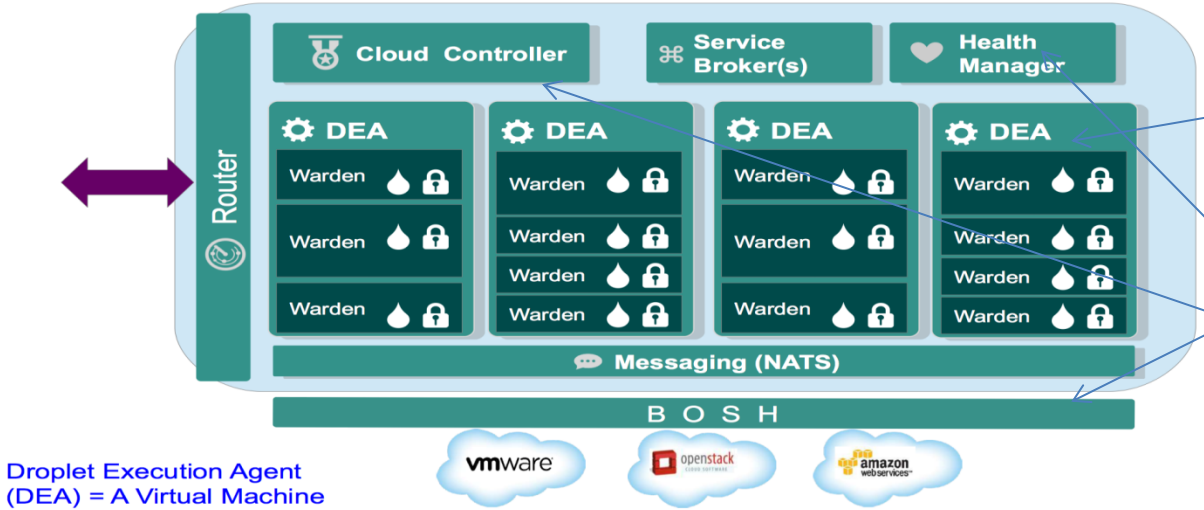
- Monetization PaaS: BSS + 소비자분석
  e.g. Zuora, Aria Systems

클라우드 파운드리

- 바이두
- Heroku
- IBM Bluemix
- Intalio

- 실습: run.pivotal.io

CF Features & Architecture



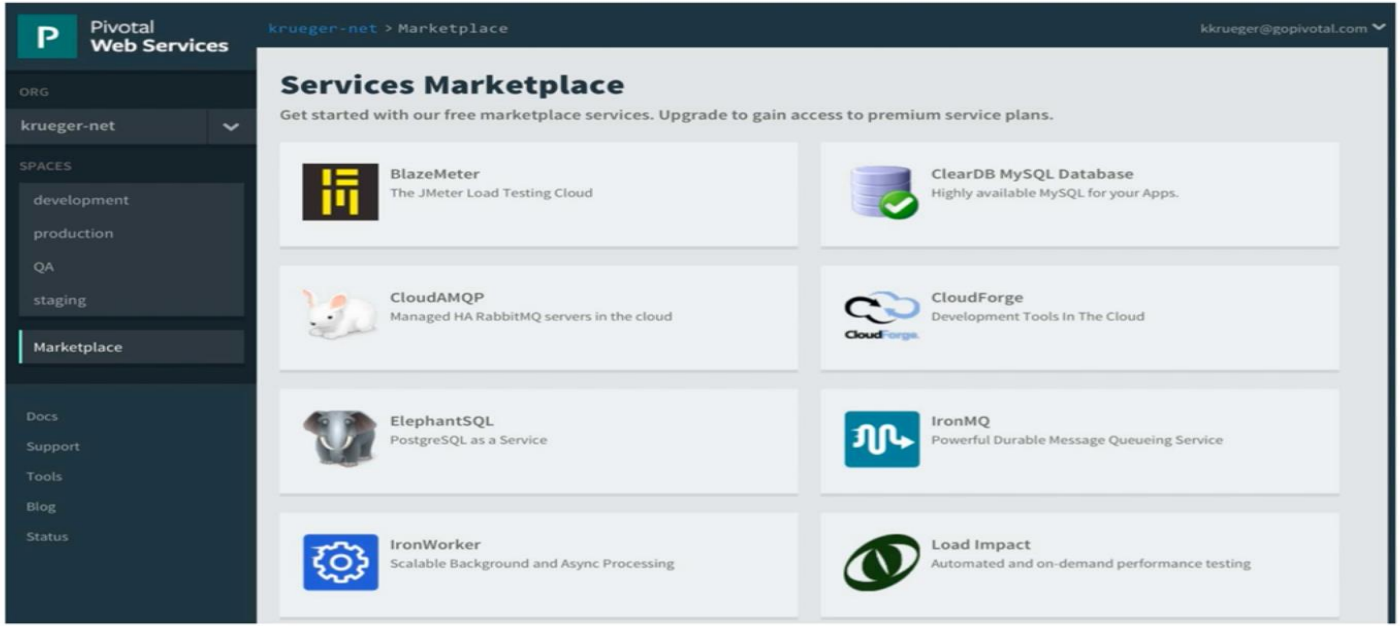
Droplet Execution Agent
(DEA) = A Virtual Machine

FEATURES

- Application Development
  - Development Spaces
  - Log aggregation
- Application Hosting
  - Infra hiding
- Business Continuity
  - Scale in/out

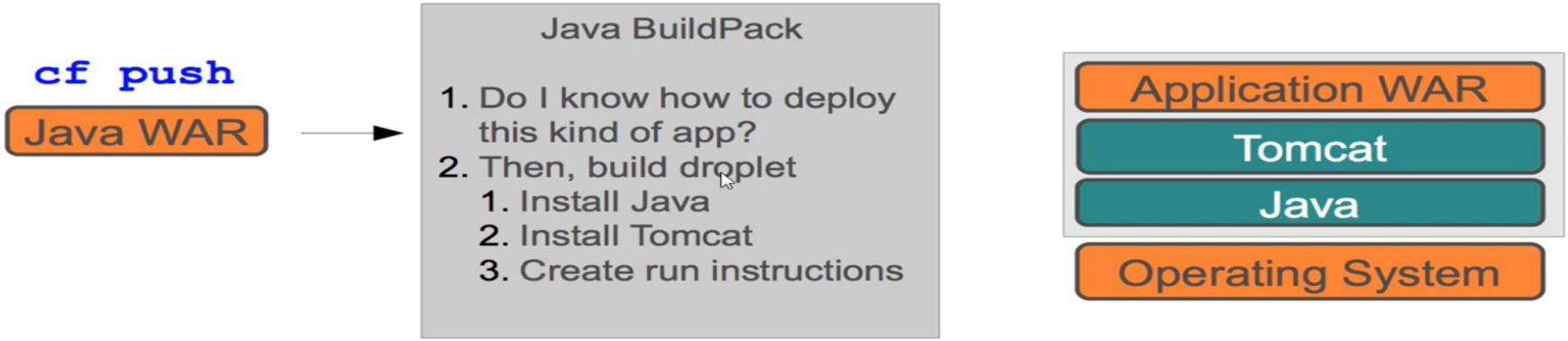
Service Brokerage

Pivotal Web Services (PWS)
Marketplace Home Page in Console



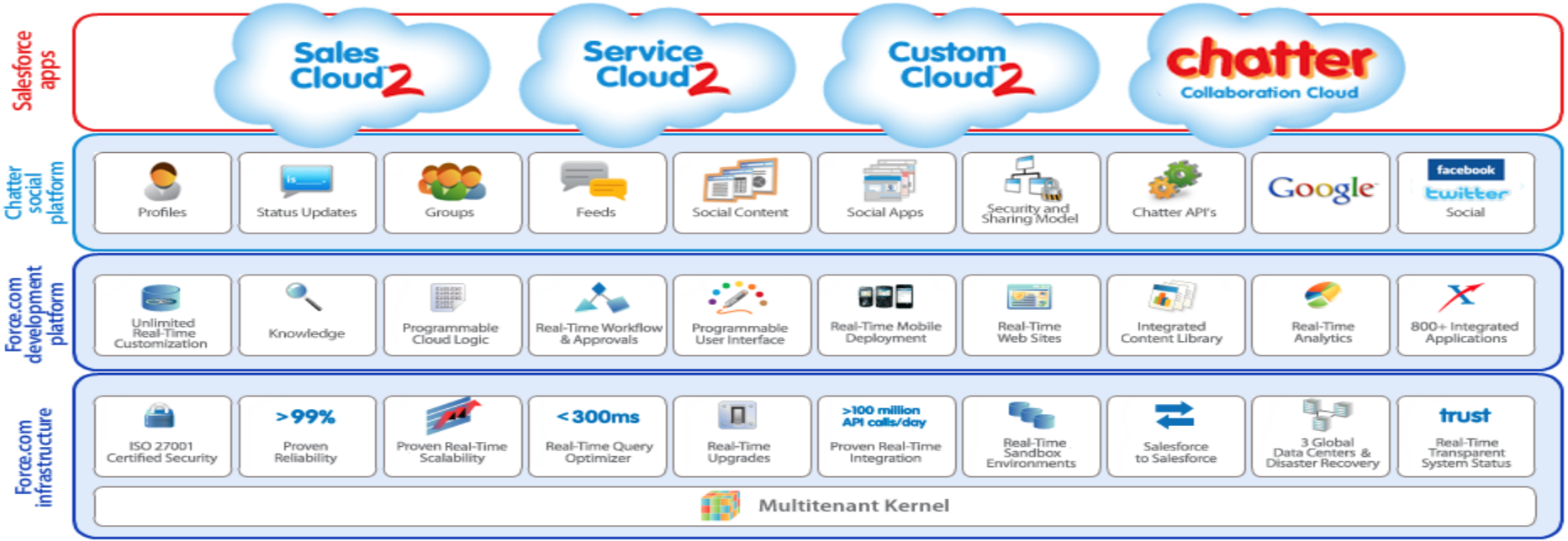
Application Packaging

- **Buildpacks** define how assemble a droplet to run a specific kind of application
- Example:



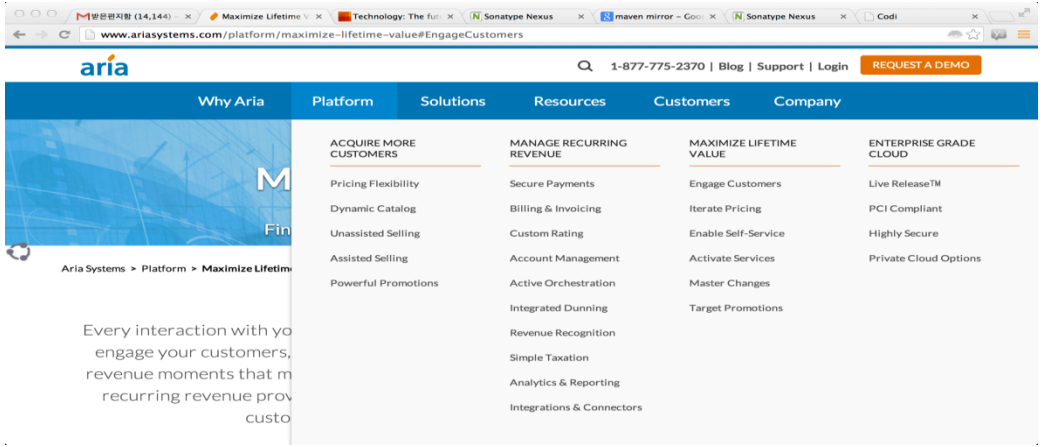
- The Buildpack "builds" the "droplet" to run an app.
  - Called *staging* the application

A BPaaS: Force.com

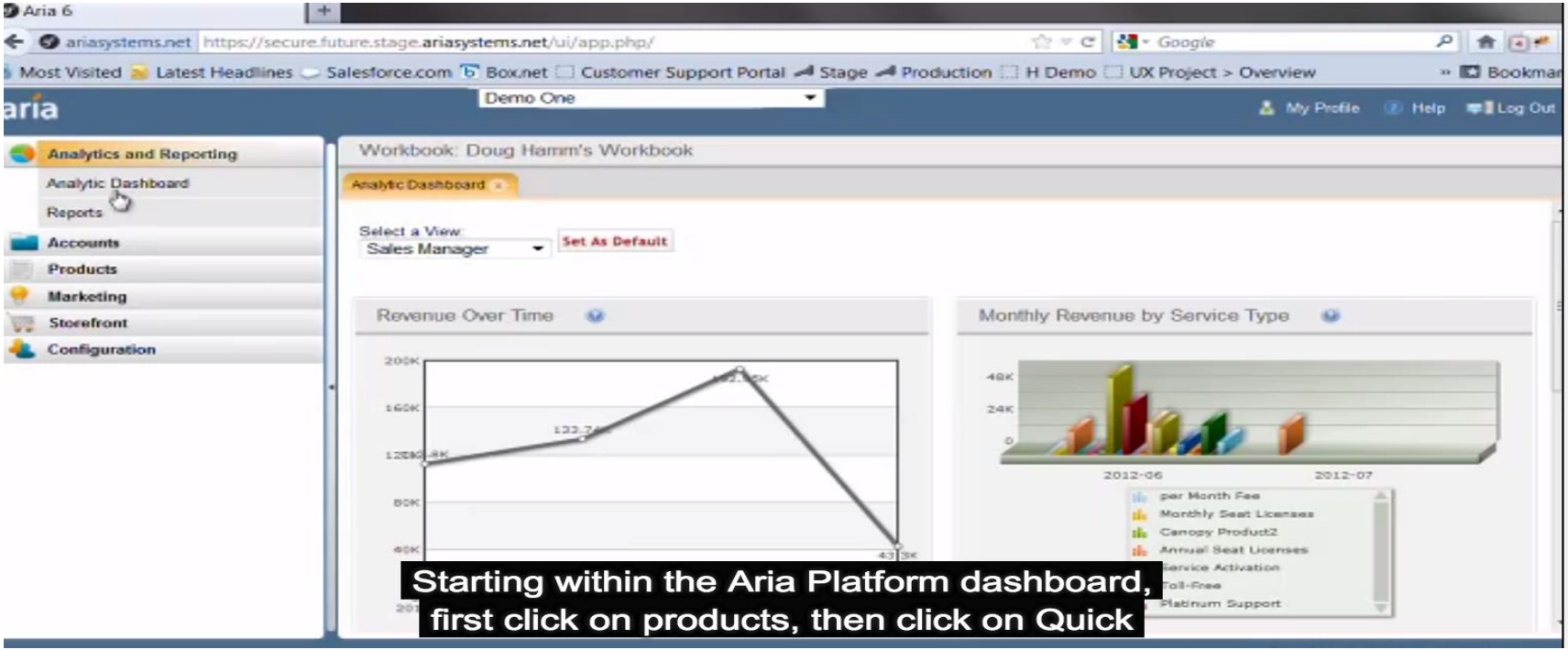


Aria Systems :
full-cycle monetization platform

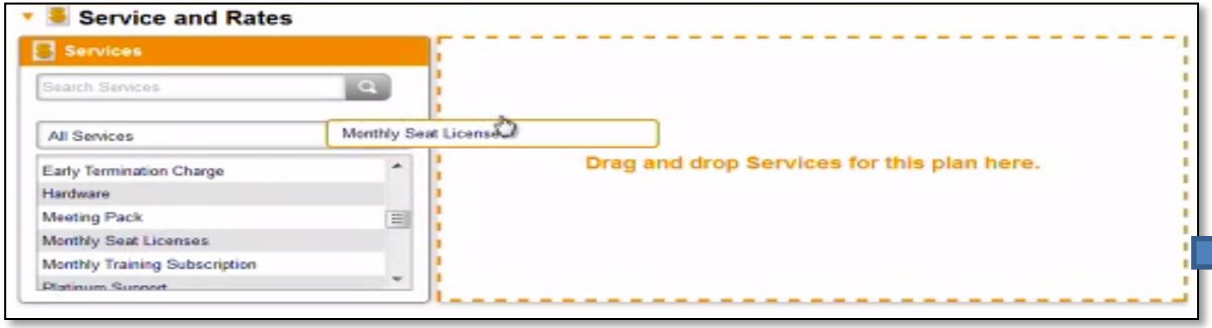
- Customer acquisition
- Service activation
- Usage tracking/rating
- Invoicing (including Calculation, Presentment, Collection, Remittance, Reconciliation)
- Customer management
- Reporting

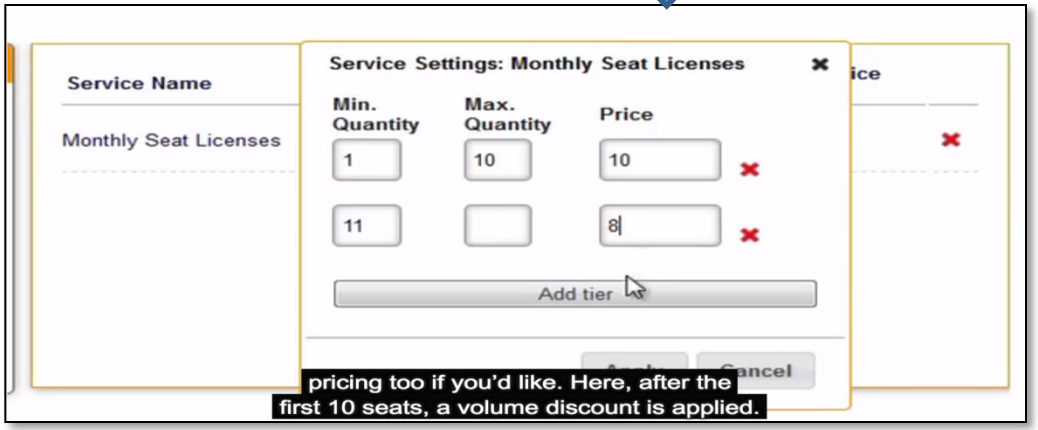


Aria Solution



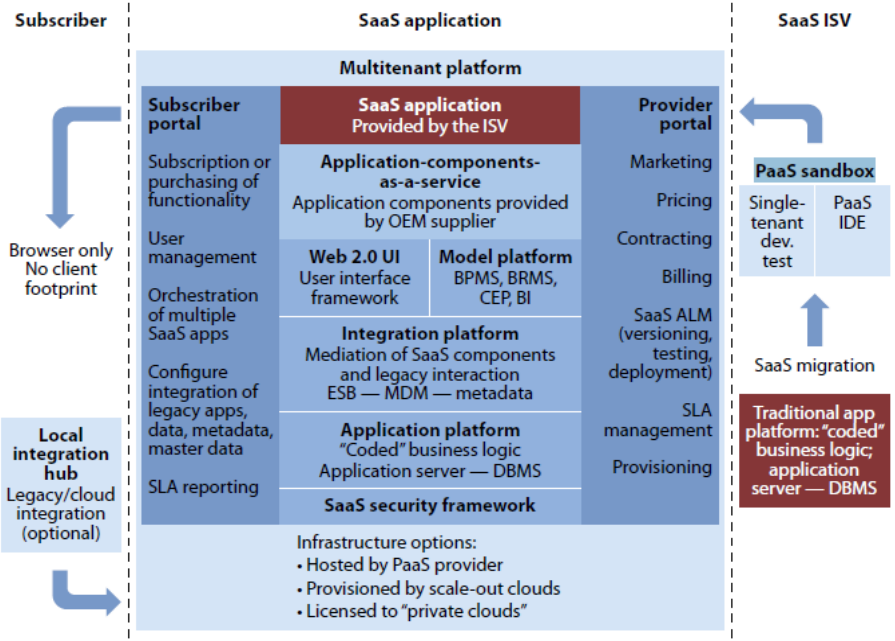
Aria: a plan has services and rates





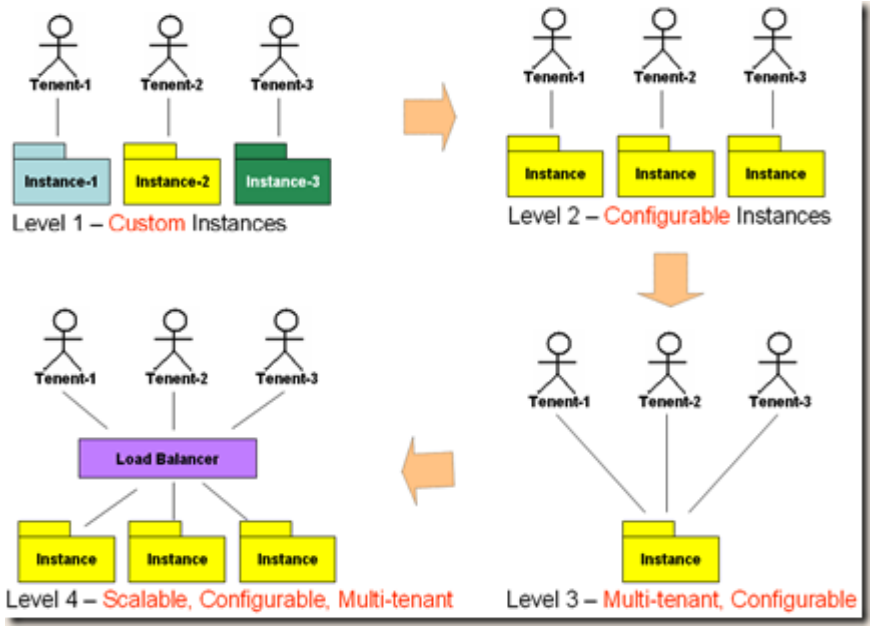
PaaS: has modularity issues

- Multiple Services



Forrester's PaaS Reference Model

- Mature SaaS Level



SaaS Maturity Level

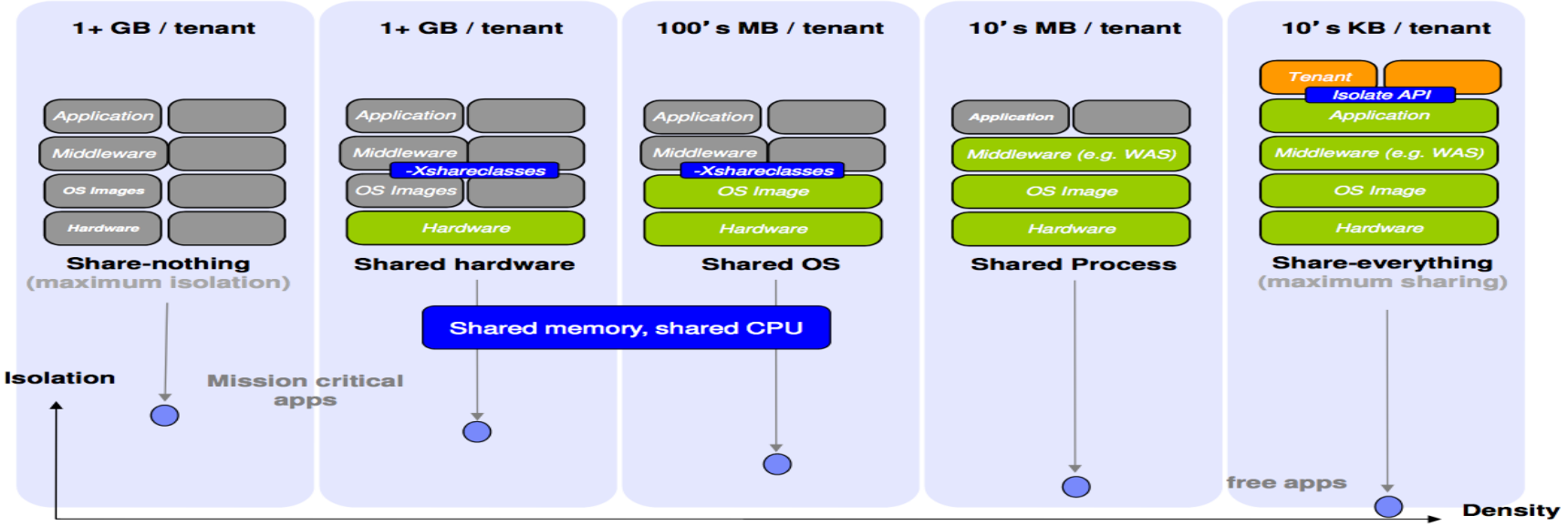
OSGi for PaaS

- More modular, so share more (at library level)
- More dynamic, so easily automated

- Provides standard J2EE framework including JAX-RS, JPA and Spring with whiteboard pattern.

Sharing Level

- Hypervisor-based vs. Container-based vs. OSGi-based PaaS
- In OSGi-based PaaS, multi-tenants can share at library



Share less, More easy & Secure !

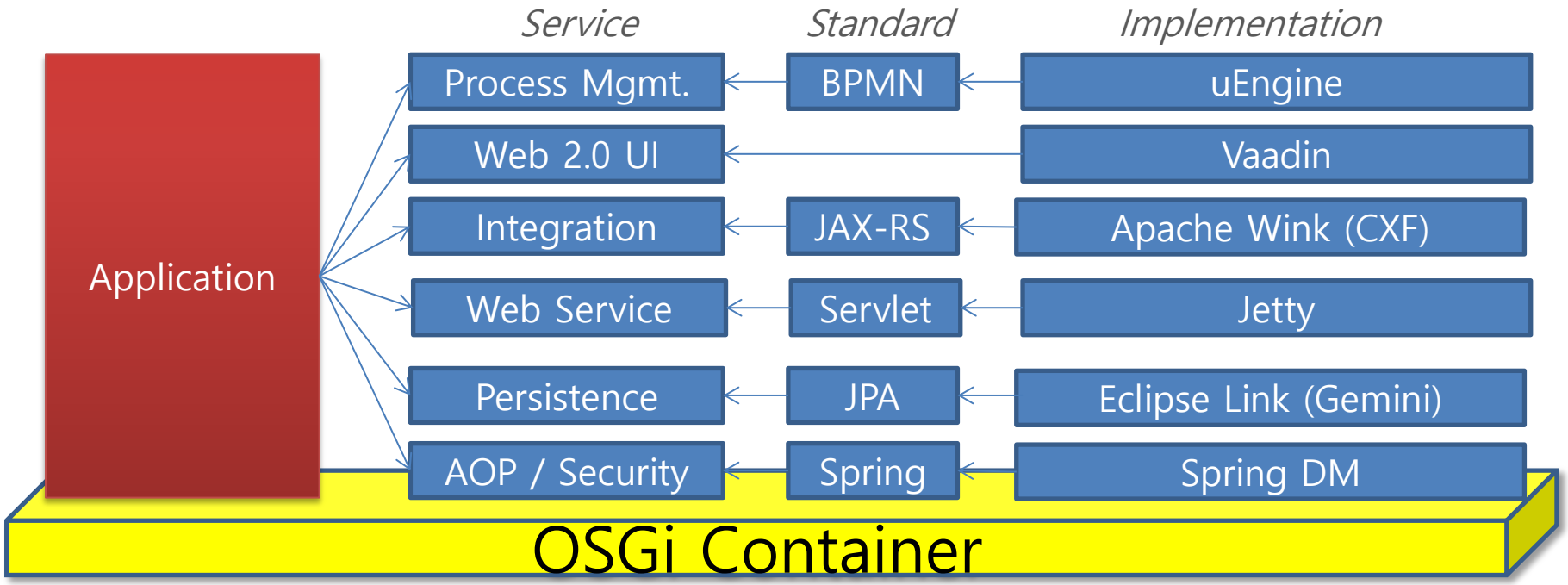
Share more, More cheap offering, More Co mpetitive in the market !

OSGi based PaaS approaches

- Apache Ace and Amdatu
- Eclipse Gyrex
- Vaadin Arvue Master
- WSO2's Stratos
- OCE Garuda
  ← Initiative from Korea!

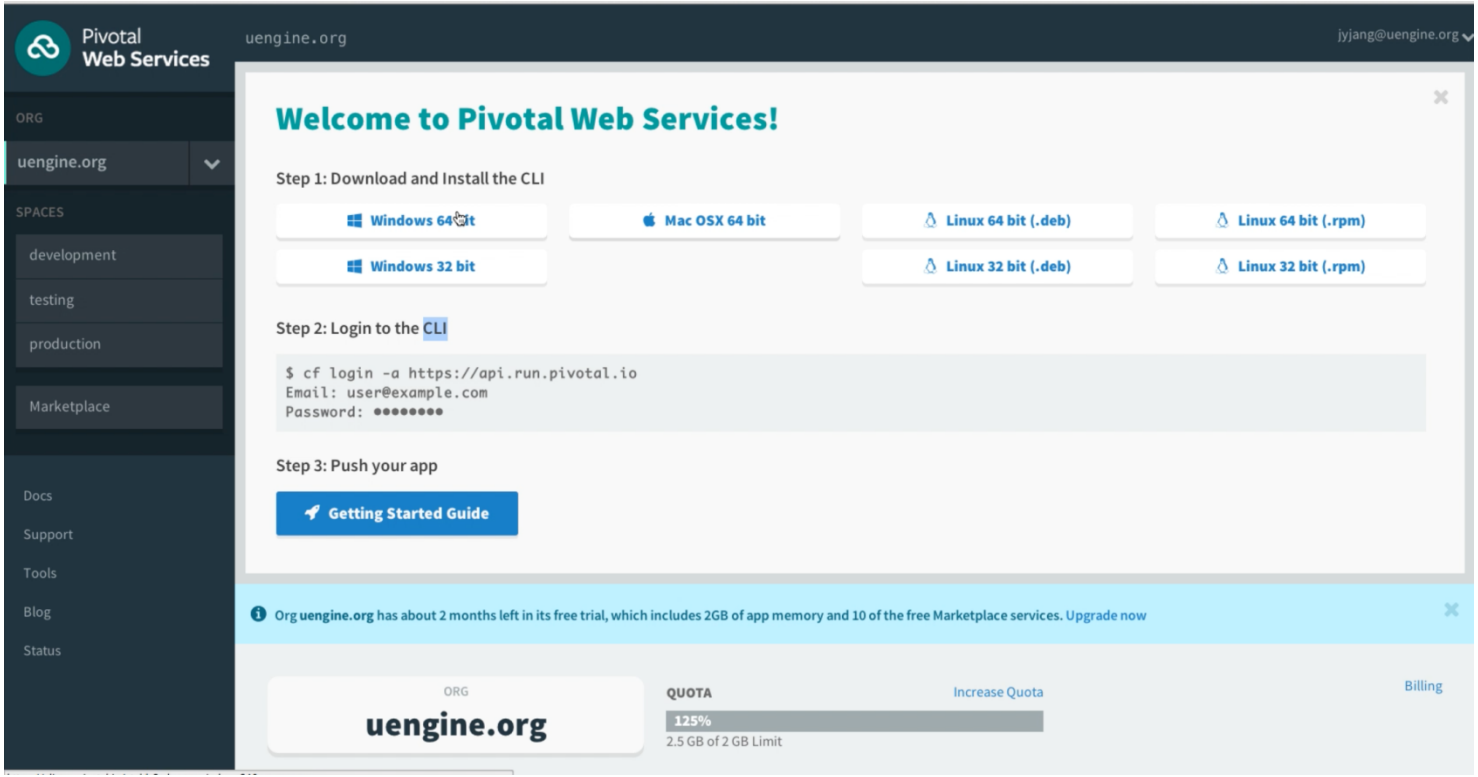
Exposing Platform Services using OSGi and RESTful APIs

e.g. OCE garuda CaaS (Component As A Service) example



Lab Time: Prerequisites

- CF Command Line Interface



Login and Select Space

```
sh-3.2# cf login -a api.run.pivotal.io -u jyjang@uengine.org
API endpoint: api.run.pivotal.io

Password>
Authenticating...
OK

Targeted org uengine.org

Select a space (or press enter to skip):
1. development
2. testing
3. production

Space> 3
Targeted space production



API endpoint:   https://api.run.pivotal.io (API version: 2.18.0)
User:           jyjang@uengine.org
Org:            uengine.org
Space:          production
```

Deploy application

```
sh-3.2# cf push spring-music -i 1 -m 512M -n spring-music-jinyoung-77 -p /java/CloudFoundryStudentFiles-master/spring-music/pre-built/spring-music
.war
Creating app spring-music in org uengine.org / space production as jyjang@uengine.org...
OK

Using route spring-music-jinyoung-77.cfapps.io
Binding spring-music-jinyoung-77.cfapps.io to spring-music...
FAILED
The route spring-music-jinyoung-77.cfapps.io is already in use.
TIP: Change the hostname with -n HOSTNAME or use --random-route to generate a new route and then push again.
sh-3.2# cf push spring-music -i 1 -m 512M -n spring-music-jinyoung-78 -p /java/CloudFoundryStudentFiles-master/spring-music/pre-built/spring-music
.war
Updating app spring-music in org uengine.org / space production as jyjang@uengine.org...
OK

Creating route spring-music-jinyoung-78.cfapps.io...
OK

Binding spring-music-jinyoung-78.cfapps.io to spring-music...
OK

Uploading spring-music...
Uploading app files from: /java/CloudFoundryStudentFiles-master/spring-music/pre-built/spring-music.war
Uploading 574.8K, 95 files
Done uploading
OK

Starting app spring-music in org uengine.org / space production as jyjang@uengine.org...
-----> Downloaded app package (21M)
-----> Java Buildpack Version: v2.5 | https://github.com/cloudfoundry/java-buildpack.git#840500e
-----> Downloading Open Jdk JRE 1.8.0_25 from https://download.run.pivotal.io/openjdk/lucid/x86_64/openjdk-1.8.0_25.tar.gz (1.6s)
       Expanding Open Jdk JRE to .java-buildpack/open_jdk_jre (1.1s)
-----> Downloading Spring Auto Reconfiguration 1.5.0_RELEASE from https://download.run.pivotal.io/auto-reconfiguration/auto-reconfiguration-1.5.0_
RELEASE.jar (0.2s)
-----> Downloading Tomcat Instance 8.0.15 from https://download.run.pivotal.io/tomcat/tomcat-8.0.15.tar.gz (0.2s)
       Expanding Tomcat to .java-buildpack/tomcat (0.1s)
-----> Downloading Tomcat Lifecycle Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-lifecycle-support/tomcat-lifecycle-support-2
.4.0_RELEASE.jar (0.0s)
-----> Downloading Tomcat Logging Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-logging-support/tomcat-logging-support-2.4.0_R
ELEASE.jar (0.0s)
-----> Downloading Tomcat Access Logging Support 2.4.0_RELEASE from https://download.run.pivotal.io/tomcat-access-logging-support/tomcat-access-lo
gging-support-2.4.0_RELEASE.jar (0.0s)

-----> Uploading droplet (67M)

0 of 1 instances running, 1 starting
```

Injecting Service Configuration to my apps

VCAP SERVICES Property

```
VCAP_SERVICES=
{
  cleardb-n/a: [
    {
      name: "cleardb-1",
      label: "cleardb-n/a",
      plan: "spark",
      credentials: {
        name: "ad_c6f4446532610ab",
        hostname: "us-cdbr-east-03.cleardb.com",
        port: "3306",
        username: "b5d435f40dd2b2",
        password: "ebfc00ac",
        uri: "mysql://b5d435f40dd2b2:ebfc00ac@us-cdbr-east-
                        03.cleardb.com:3306/ad_c6f4446532610ab",
        jdbcUrl: "jdbc:mysql://b5d435f40dd2b2:ebfc00ac@us-
                 cdbr-east-03.cleardb.com:3306/ad_c6f4446532610ab"
      }
  ...
```

ClearDB is the MySQL instance offered through App Direct

1. Get Env var 'VCAP_SERVICES'
2. JSON Parsing
3. Insert properties

```
datasource.driverClassName="com.mysql.jdbc.Driver"
datasource.username="pivotal"
datasource.password="pivotal"
datasource.url="jdbc:mysql://myDbHost:3306/pivotaldb"
```

Java

Getting Mysql service from CF marketplace

Conclusion

Lessons Learned

- POJO based frameworks are key-enablers to make domain-driven software development.
  - JPA
  - Metaworks
  - JAX-RS
  - Spring
- SOA based Application Design
  - Modularity is enhanced
  - Dynamically reconfigurable
- Model-based applications are also easy to be deployed to IaaS and PaaS.
  - Since application is more modular, applications are easy to migrated, ported and provisioned to diverse cloud environment.
  - Also easy to be scaled out and in.