CSI5380: Systems & Architectures

For

Electronic Commerce

**Project Part 1: CD Store –Documentation**

**Submit by;**

**Team”Kill – 9”**

|  |  |  |
| --- | --- | --- |
| No. | Team Members | Student ID |
| 1 | Michele Belanger-Petrucci (Team Leader) | 566220 |
| 2 | Kaifan Zhu |  |
| 3 | Yuan Liu |  |
| 4 | Jiaren Suo |  |
| 5 | Cynthia Torkuma Ikongo |  |

Contents

[1 Introduction 3](#_Toc434193256)

[2 Objectives 3](#_Toc434193257)

[3 Design Analysis 4](#_Toc434193258)

[3.1 Use Case Specification 4](#_Toc434193259)

[3.2 MVC framework 7](#_Toc434193260)

[3.3 N-Tier Architecture 7](#_Toc434193261)

[3.4 Deployment Diagram 10](#_Toc434193262)

[4 Logical Design 11](#_Toc434193263)

[4.1 Package Design and Diagram 11](#_Toc434193264)

[4.2 Database Design and Schema 12](#_Toc434193265)

[5 Sequence Diagrams 13](#_Toc434193266)

[5.1 ProductCatalogService Webservice 15](#_Toc434193267)

[5.2 ProcessOrderService WebService 19](#_Toc434193268)

[5.3 LoginAction Controller 23](#_Toc434193269)

[5.4 ProductDetailAction Controller 23](#_Toc434193270)

[5.5 CategoryAction Controller 24](#_Toc434193271)

[5.6 ProductDetailAction Controller 24](#_Toc434193272)

[5.7 ProductListAction Controller 25](#_Toc434193273)

[5.8 CheckOutAction Controller 26](#_Toc434193274)

[5.9 PurchaseAction Controller 27](#_Toc434193275)

[5.10 RegistrationAction Controller 28](#_Toc434193276)

[6 Screenshots 29](#_Toc434193277)

[7 Testing Framework 29](#_Toc434193278)

[8 Individual Contribution towards the project 29](#_Toc434193279)

## 1 Introduction

This project is about a simple E-commerce based website that allows users to browse and order CDs though catalog. This document is a generic Design Document, which provides guidance and template material step by step to show how to accomplish the whole system. Details on building and deploying the environment are provided in the included README.txt file.

## 2 Objectives

The purpose for this project is for customers to browse the list of available CDs in the music category of their choice.. The list of CDs and information about them is obtained from a relational database. Users can add CDs to a Shopping Cart, update the quantity of each CD in their cart, remove CDs from their cart, and view a summary of their order before they enter their credit card information and submit the order. The web application is secured using industry standard SSL (Secure Socket Layer) technology. In order for the user to proceed to checkout, he/she must either register with the site if he/she is a new user or if they are returning users, they must login with proper credentials.

## 3 Design Analysis

### 3.1 Use Case Specification

Register or login

Browse CDs

Add to shopping cart

Check out

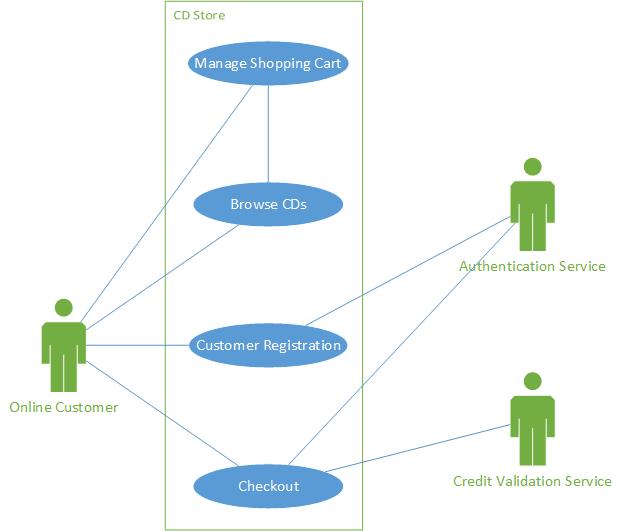


Figure : CD Online Store Use Case Diagram

* **Client register and login**
* The new customer clicks the registration icon on the top of the Home page or Item detail page.
* The Account Information page is presented to the customer.
* The new customer enters the following details in the Account Information page: Username, Password, Confirm Password, First Name, Last Name, E-mail address, shipping address and billing address.
* The customer clicks on Submit and the system validates all the customer information.
* For returning customers, they are presented with a screen prompting for username and password. Once entered, the customer clicks the “Login” button. If the login information provided passes validation, the customer is now logged into the CD store.
* **Browsing products.**
* Customer opens a browser and enters the online CD store URL.
* Customer enters search criteria in the text box next to search icon and clicks on search icon.
* The CDs matching the search criteria are then displayed on the Home page. For each CD listed, and the following information is displayed:
  1. Name of CD (displayed as a hyperlink)
  2. Name of Artist
  3. Price
* Customer can click the hyperlinks in the search results to display details of the selected CD. The following information is displayed on the Item detail page:
  1. CD Title
  2. CD Category
  3. Description of the CD
  4. Picture (if available)
  5. Price per CD
* Customer can add CDs to the Shopping Cart by clicking the ‘add to cart’ button on the Item detail page.
* **Mange shopping cart**
* The user clicks on the Shopping Cart icon on the top right hand corner of the Home page or Item Detail Page.
* The Shopping Cart page is displayed with the items that have been selected. The following details are displayed for each Item in the Shopping Cart:

1. The name of the CD displayed as a hyperlink
2. The quantity of each Shopping Cart Item to be purchased
3. The unit price of each Shopping Cart Item

* The Shopping Cart page also displays the total cost of the items in the Shopping Cart.
* The customer can modify the quantity desired for each Item and the total cost displayed on the Shopping Cart is automatically updated.
* The customer can remove items from the Shopping Cart by clicking the “Remove” button next to the item no longer required. Once the button is clicked, the item is removed and the total cost is updated.
* The customer can proceed to purchasing the items in the Shopping Cart by clicking “Checkout”.
* **Check out**
* Customer clicks “Checkout” on the Shopping Cart page.
* Customer is then present with the Checkout page. The following information must be entered:
  1. First Name
  2. Last Name
  3. Street
  4. City
  5. Province (drop down list)
  6. Postal Code (A1A1A1 mask)
  7. Country (drop down list)
  8. Phone number (XXX XXX-XXXX mask)
  9. Name on credit card
  10. Expiration date (MM/YYYY mask)
  11. CCV (3 digit number)
* Once all information is entered, the customer clicks “Place Order”.
* The system validates the information as follows:
  1. All fields are populated
  2. Field level formatting rules have been respected
  3. Credit card validation service is called to ensure the credit card information is valid.
* If all information passes the validation step, the customer is presented with the Order Detail page. This page acts as the final step in the ordering process. The details of the order being placed is displayed. The information that is presented to the customer is the following:
  1. Order Number
  2. Order Status
  3. First Name
  4. Last Name
  5. Street
  6. City
  7. Province
  8. Postal Code
  9. Country
  10. Phone number
  11. Total cost of the order
* If the order is correct, the customer clicks the “Confirm Order” button in order to place the order.
* If the order is successfully processed, the Order Confirmation page is presented to the customer. This page contains the following information:
  1. Order Number
  2. Order Status
  3. First Name
  4. Last Name
  5. Street
  6. City
  7. Province
  8. Postal Code
  9. Country
  10. Phone Number
  11. Name on the credit card
  12. Credit card number (all numbers are represented by X’s except the last 4 digits)
  13. Total cost of the order
* An email is then sent to the customer with the information contained on the Order Confirmation page.

### 3.2 MVC framework

The design pattern that was adhered to in the CD store was the Model-View-Controller pattern. The figure below depicts the pattern and the responsibilities of each component. The CD store was implemented in a way that the View always calls the Controller(s) in order to access the Model.

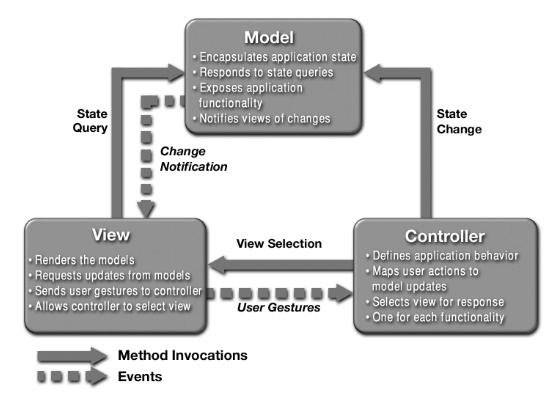
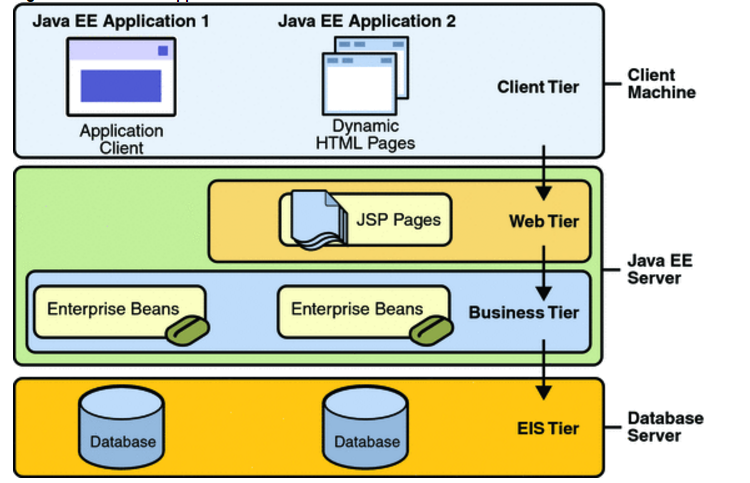


Figure 2 The MVC Pattern

### 3.3 N-Tier Architecture

This system consisted in four tiers (the diagram is shown below):

* Client tier components run on the client machine.
* Web tier components run on the server.
* Business tier components run on the server.
* Enterprise Information System tier software runs on the EIS server.



Product Catalog

Web Service

Order processing

Web Service

Servlet

Figure : Java EE applications divided into the tiers

* Client Tier:

At the client we have a web browser as the client. By using a web browser as the client allows the entire application to be located on the server.  Also the client can use any web browser from any location provided that they can connect to the application server.  The client tier can connecting to server through HTTP to the application server and gets a JSP page in return.

* Web Tier

At the web tier, it consists of JSP pages and a Session Controller Servlet served by Apache Tomcat. Servlets can process HTTP requests coming from client and construct responses by returning a certain JSP page. In the web tier, the server can communicate with business tier (ProductCatalogService and OrderProcessingService) using SOAP Web Service protocol.

* Business Tier

At business tier, which is logic that solves or meets the needs of a particular business domain such handled by Web Service using SOAP. The business tier can connect with EIS tier using DB Agent.

* EIS Tier

The EIS tier stores critical data required by an enterprise to run its business operations. Web-enabled enterprise applications have the ability to access and manage EIS stored data. And EIS tier technologies include Java Database connectivity API. We use a MySQL database for storing data and all SQL statements are contained in stored procedures.

### 3.4 Deployment Diagram

Hibernate

Connecting Pool

BROWER

**CLIENT**

WEB SERVER

Servlet

JSP

WEB SERVICE

Order Processing WS Client

Product Catalog WS Client

Order Processing WS

WSDL

Product Catalog WS

WSDL

DB Agent

DATABASE SERVER

Database

Figure : Deployment Diagram

The Web Services can be deployed on a separate application server since communication takes place over HTTP (SOAP). In addition, the application server can be configured to connect to a separate machine running the Database Server. While the architecture supports such a wide deployment, it is possible to run the client web browser, servlet container, web services containers and CD Store database on the same machine.

## 4 Logical Design

## 4.1 Package Design and Diagram

The packages in our system consists:

* Servlet package.
* JSPs package.
* Web Service package: ProductCatalogService and OrderProcessingService
* WS Client package: ProductCatalogService and OrderProcessingService
* Database Agent: Hibernate.dao package.
* Session. Bean package and util package

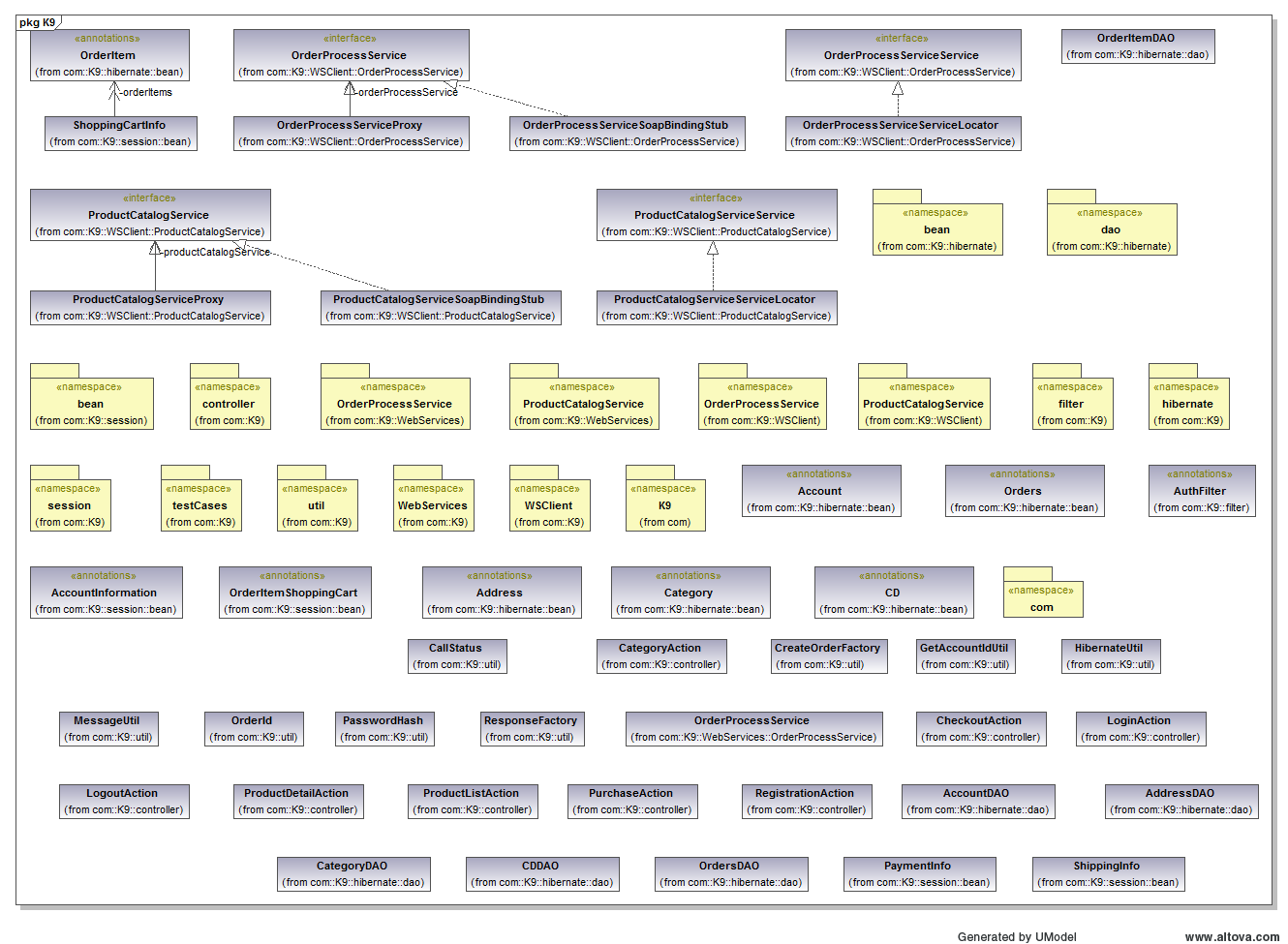


Figure : Content of K9 for all subpackages

Http response

JDBC

Http request

Web Tier

Jsp View Package

J2EE Server

Servlet Package

Web Service Package

Business Tier

Product Catalog Service

Order Process Service

Product Catalog Service

Order Process Service

WS Client Package

Hibernate.dao

Session. Bean

Util

Client Tier



Enterprise Info DB Tier



Figure : High level of packages diagram

## 4.2 Database Design and Schema

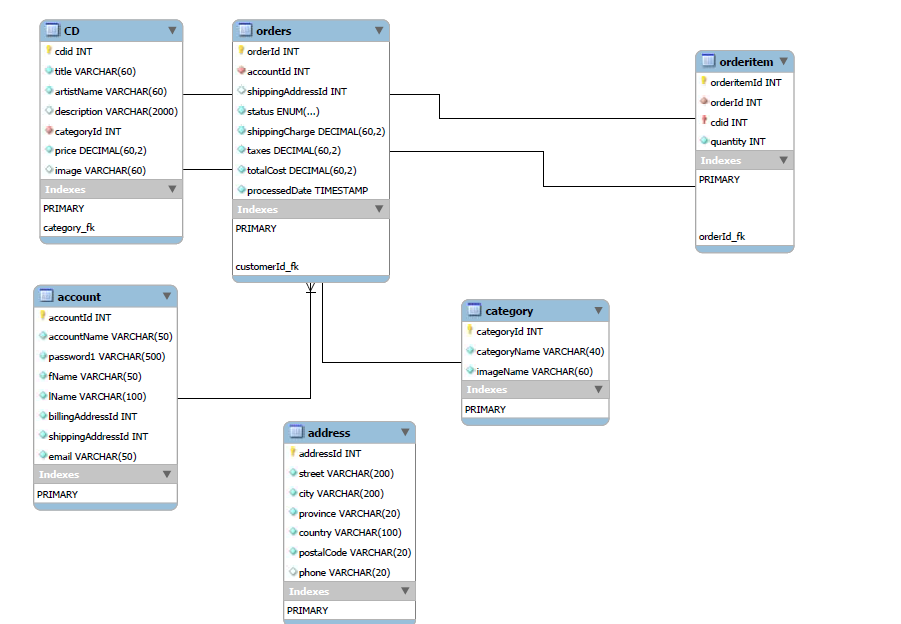


Figure 7：Database Schema

## 5 Sequence Diagrams

The Figure 6 below depicts the typical interaction between the View, Controller and Model. This is the design pattern that was adhered to in the CD project.

The views are jsp pages that interact with the Model through calls to the controllers. Each jsp page will call a specific controller in order to display the desired information. The table 1 outlines which jsp pages call which servlet.

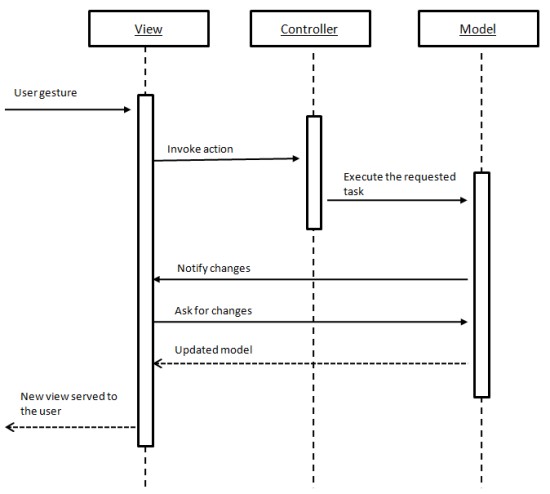


Figure 8 Generic Sequence Diagram for a typical MVC Application

TODO – add table to map jsp to controller.

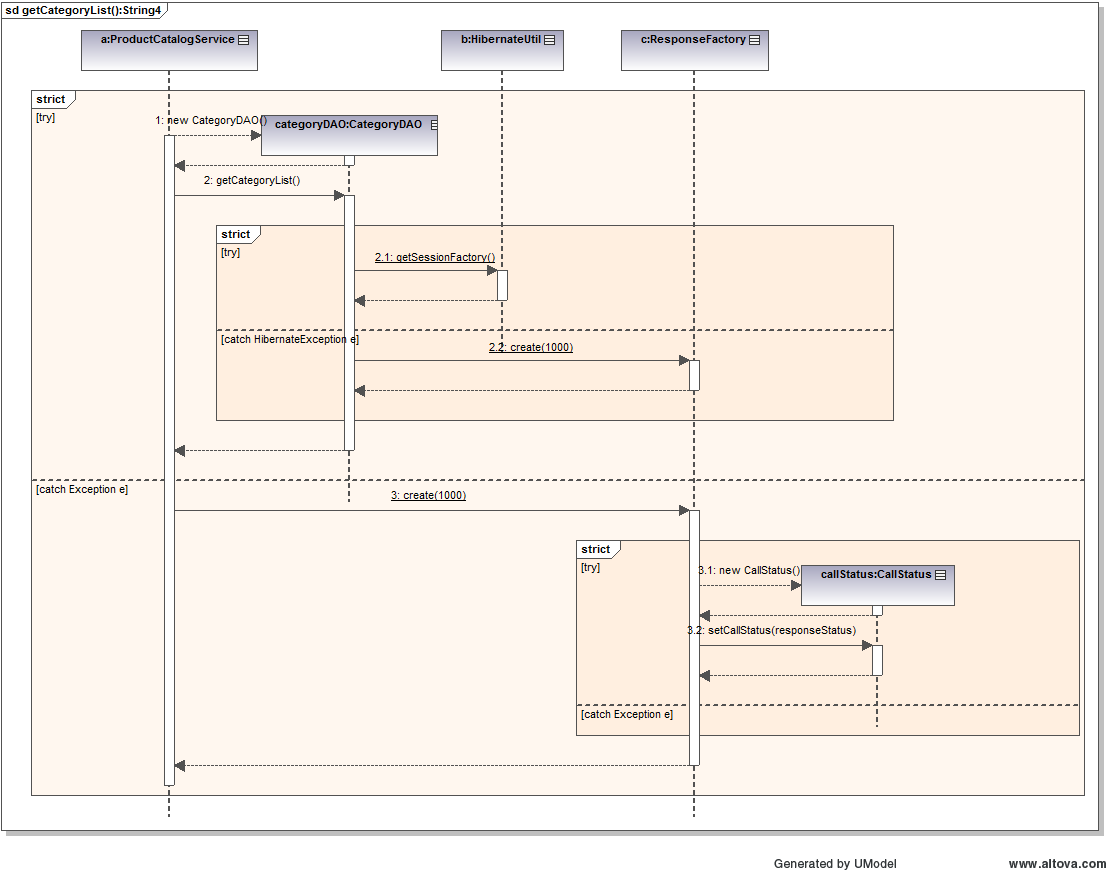
|  |  |
| --- | --- |
| JSP name | Controller Name |
| cart | addtoCartAction |
| category | categoryAction |
| finish | logoutAction |
| index |  |
| info | ProductDetailAction |
| payment |  |
| register |  |
| shipping |  |
| signin | loginAction |

Table 1 Mapping of JSP Calls to Servlets

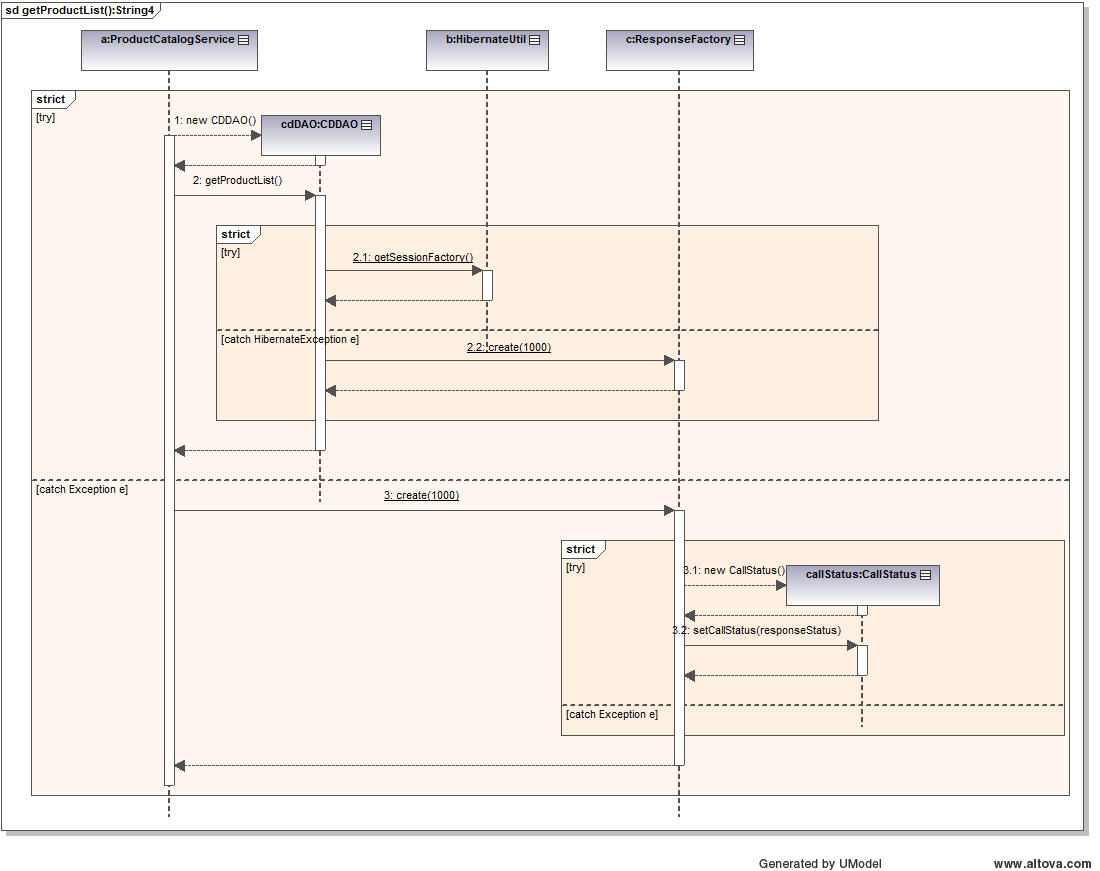
The controllers are responsible for interacting with the Model. All Model-related functionality is exposed through two webservices. Client stubs were created so that the controllers could use them to call the webservices as if the webservices resided locally. The following sequence diagrams depict the flow of each method call contained within the controllers and the two webservices. The calls to the database are handled through hibernate calls so the specifics of the interactions between the webservices and the databases are abstracted out. They are not explicitly depicted since the hibernate framework does the work. No programming was required, only configuration. There is a need to zoom into the pages in order to see the details of the sequence diagrams.

### 5.1 ProductCatalogService Webservice

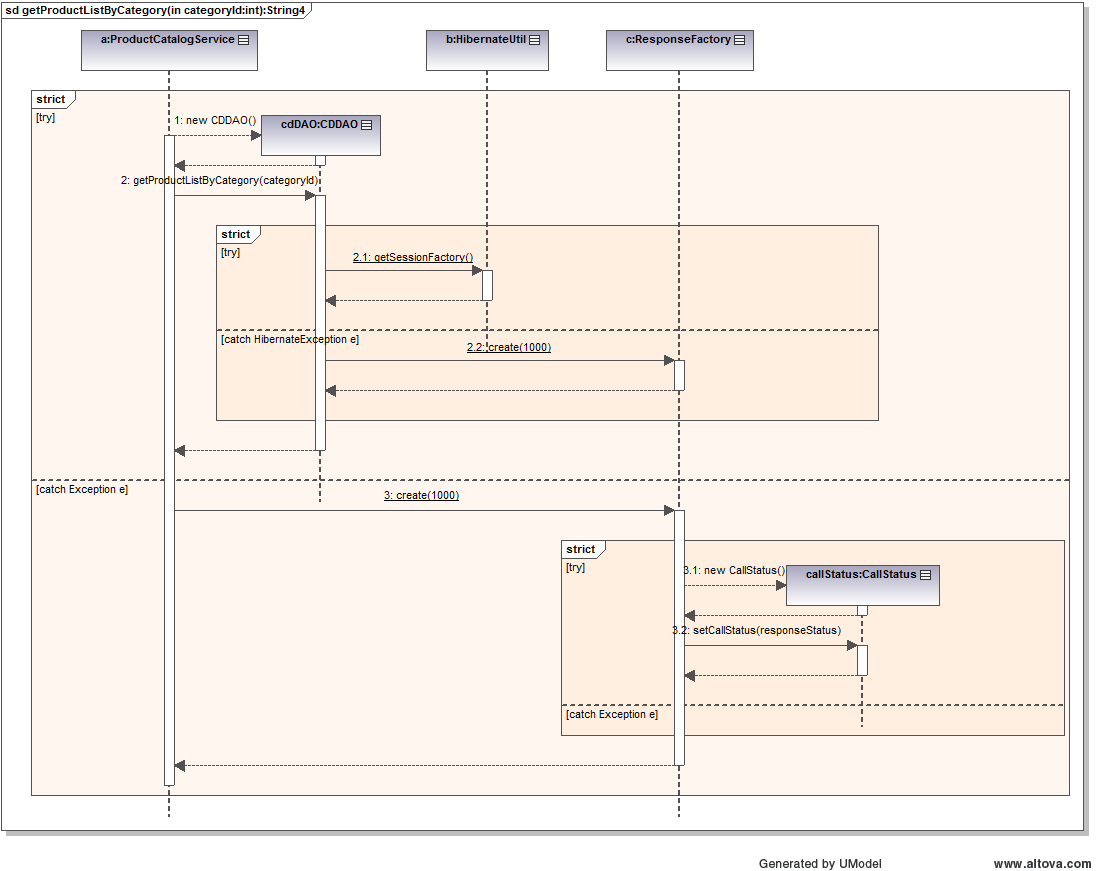
#### 5.1.1 getCategoryList Sequence Diagram



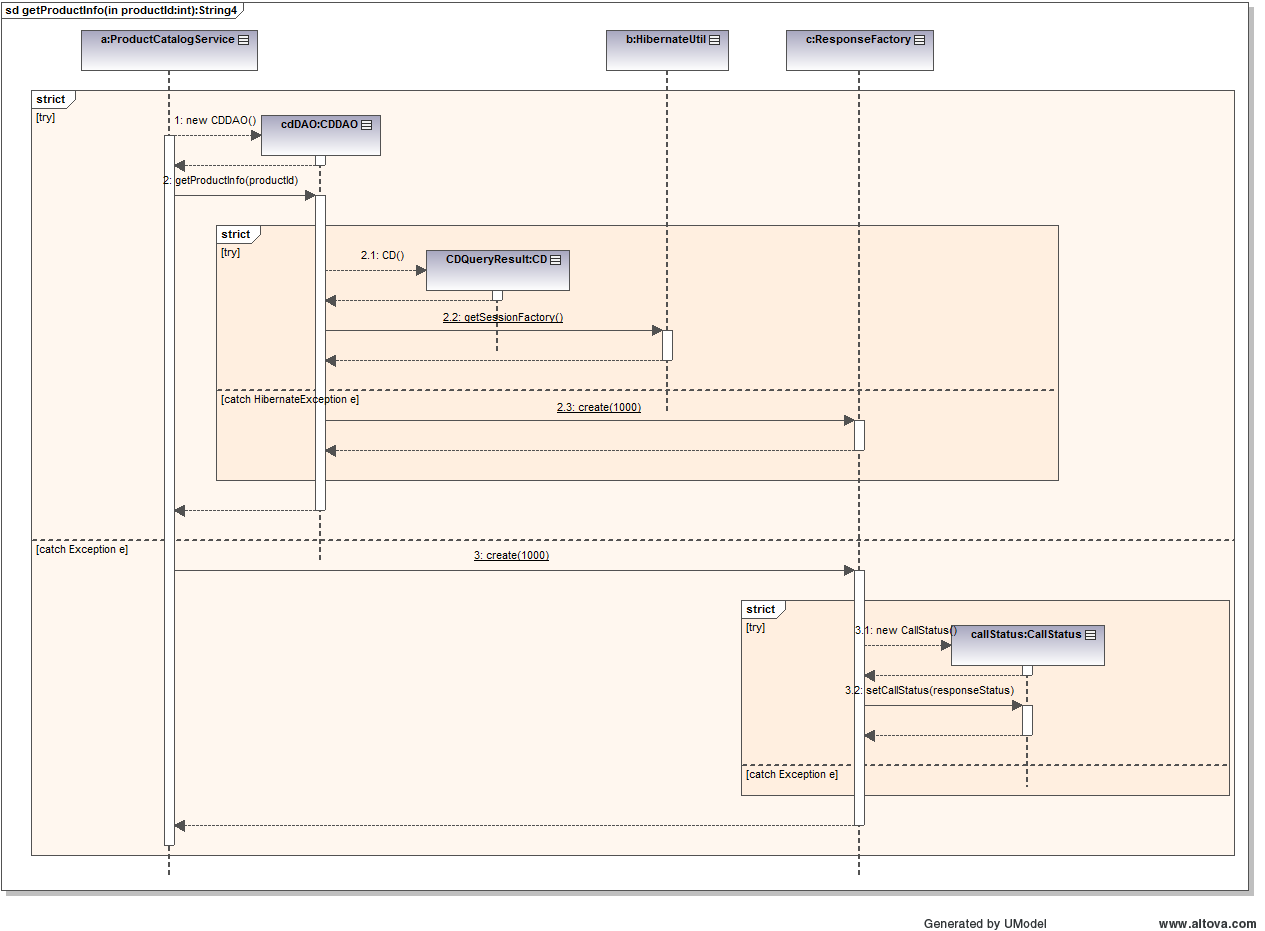
#### 5.1.2 getProductList Sequence Diagram



#### 5.1.3 getProductListByCategory Sequence Diagram

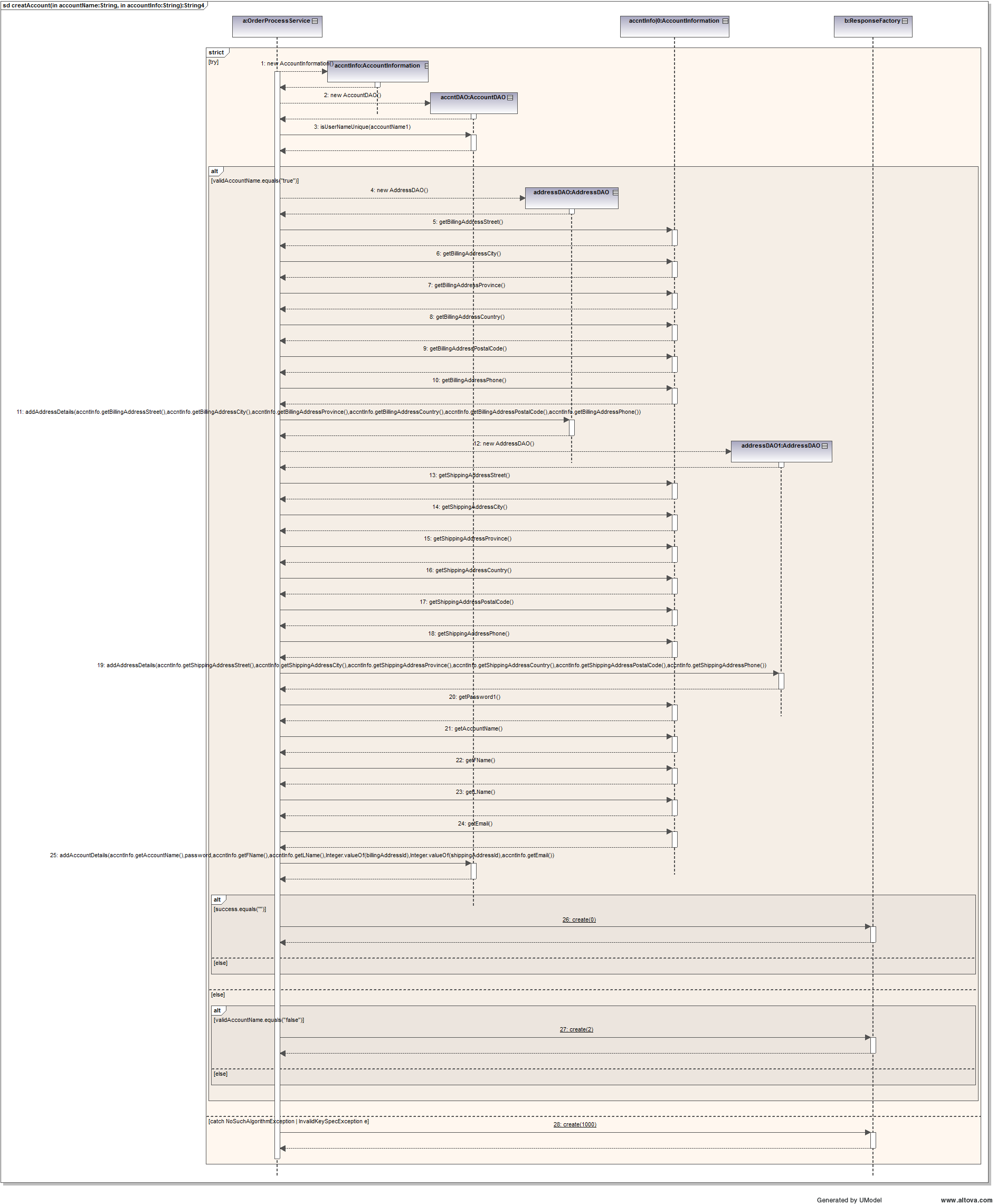


#### 5.1.4 getProductInfo Sequence Diagram

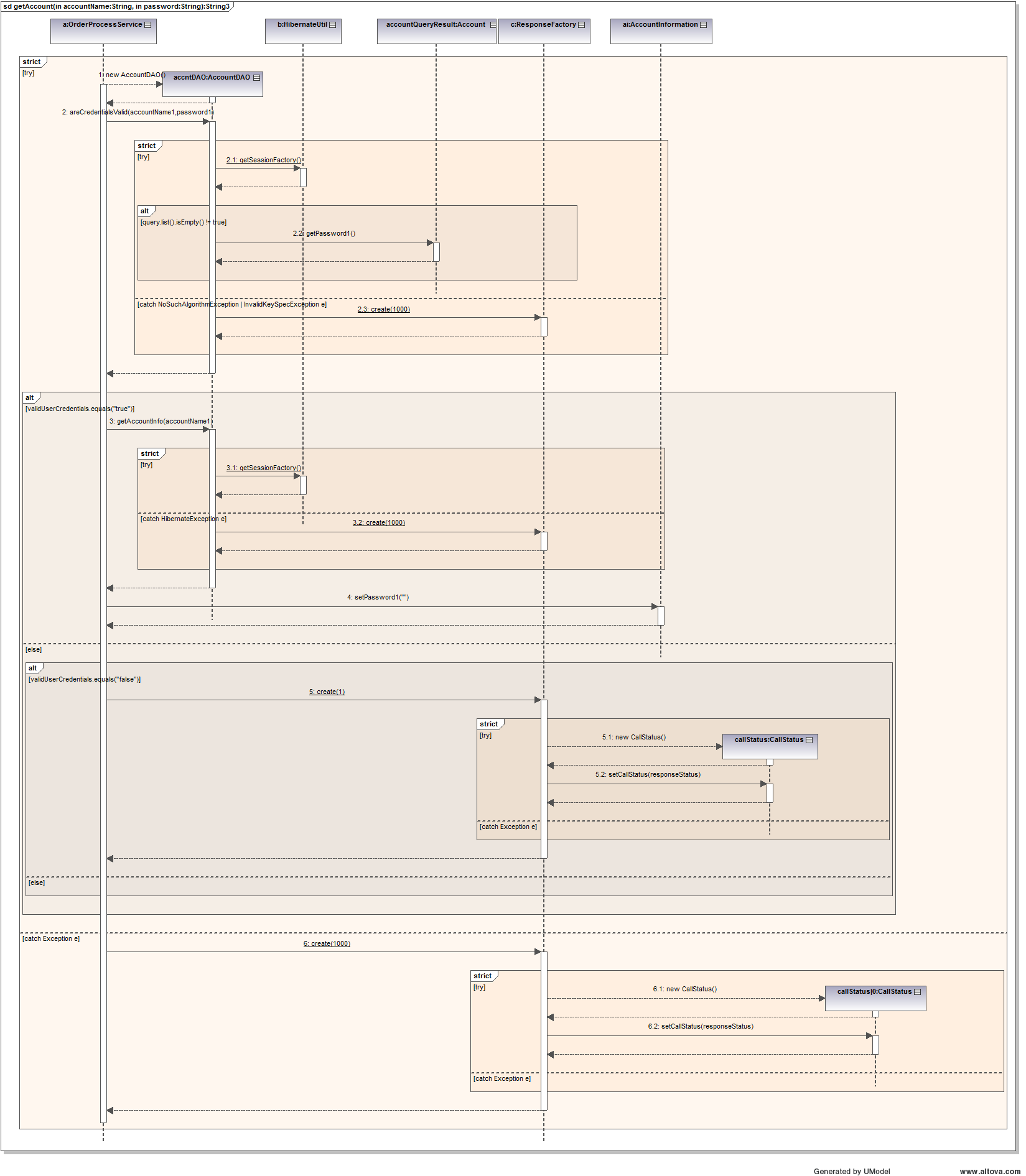


### 5.2 ProcessOrderService WebService

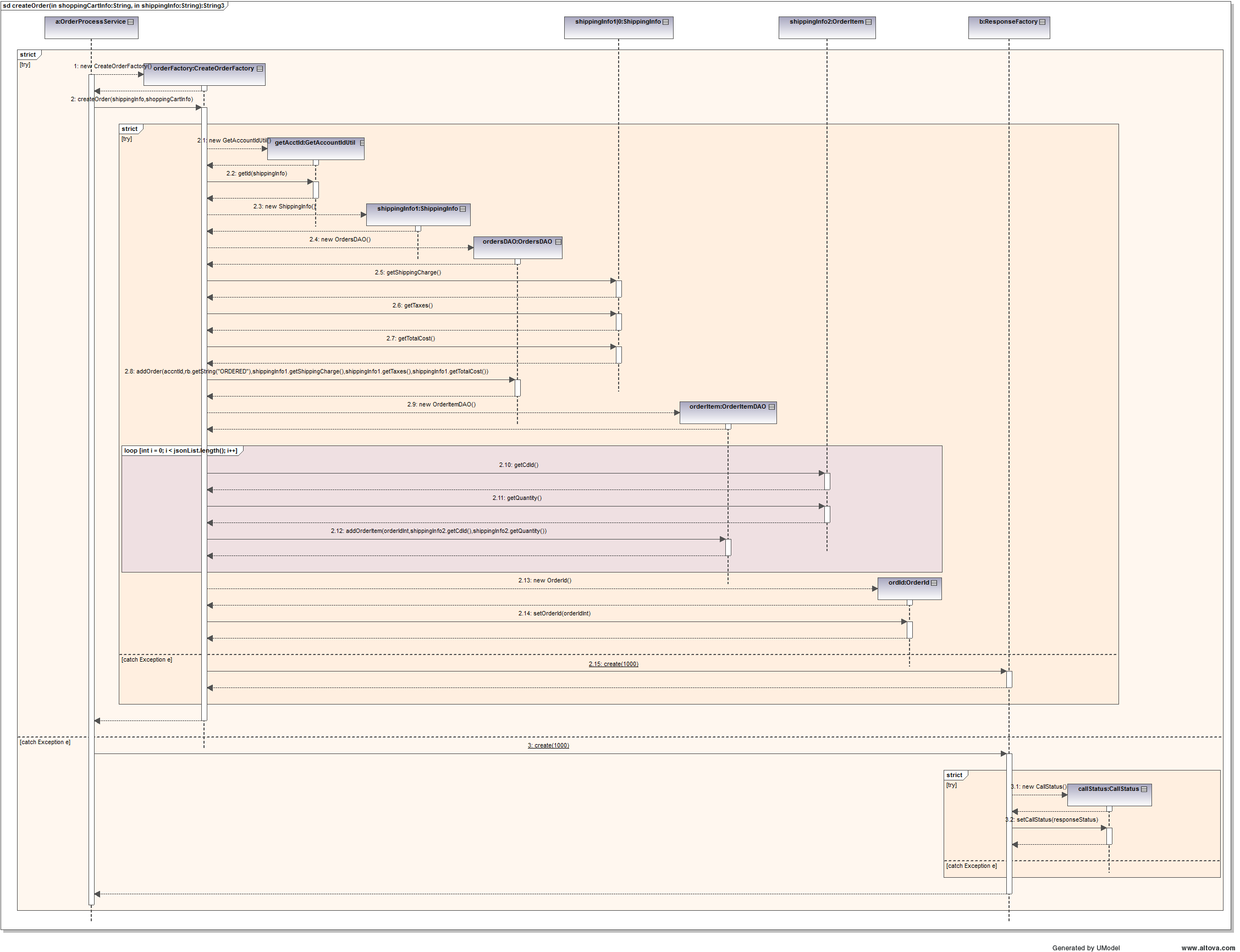
#### 5.2.1 createAccount Sequence Diagram



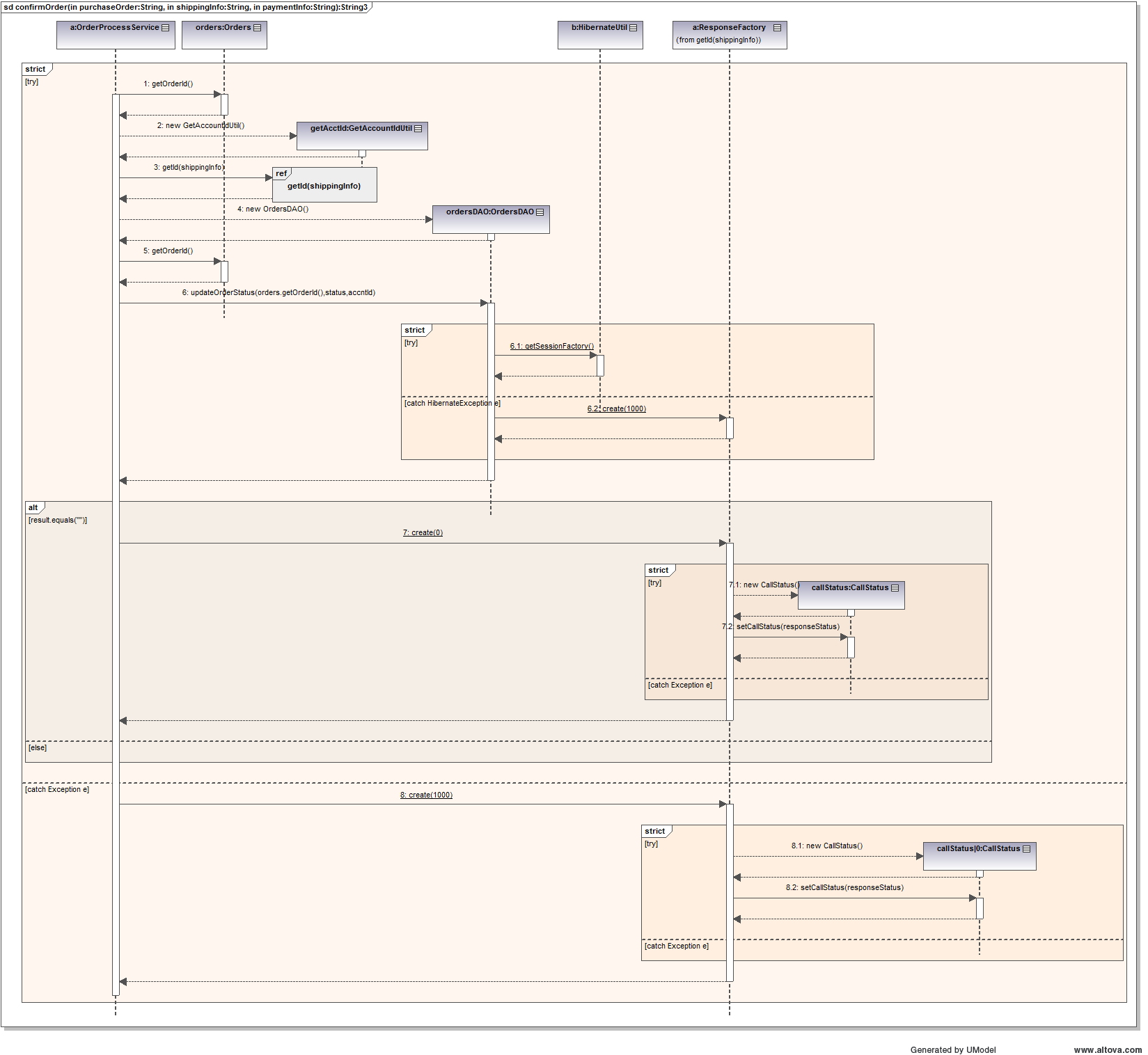
#### 5.2.2 getAccount Sequence Diagram



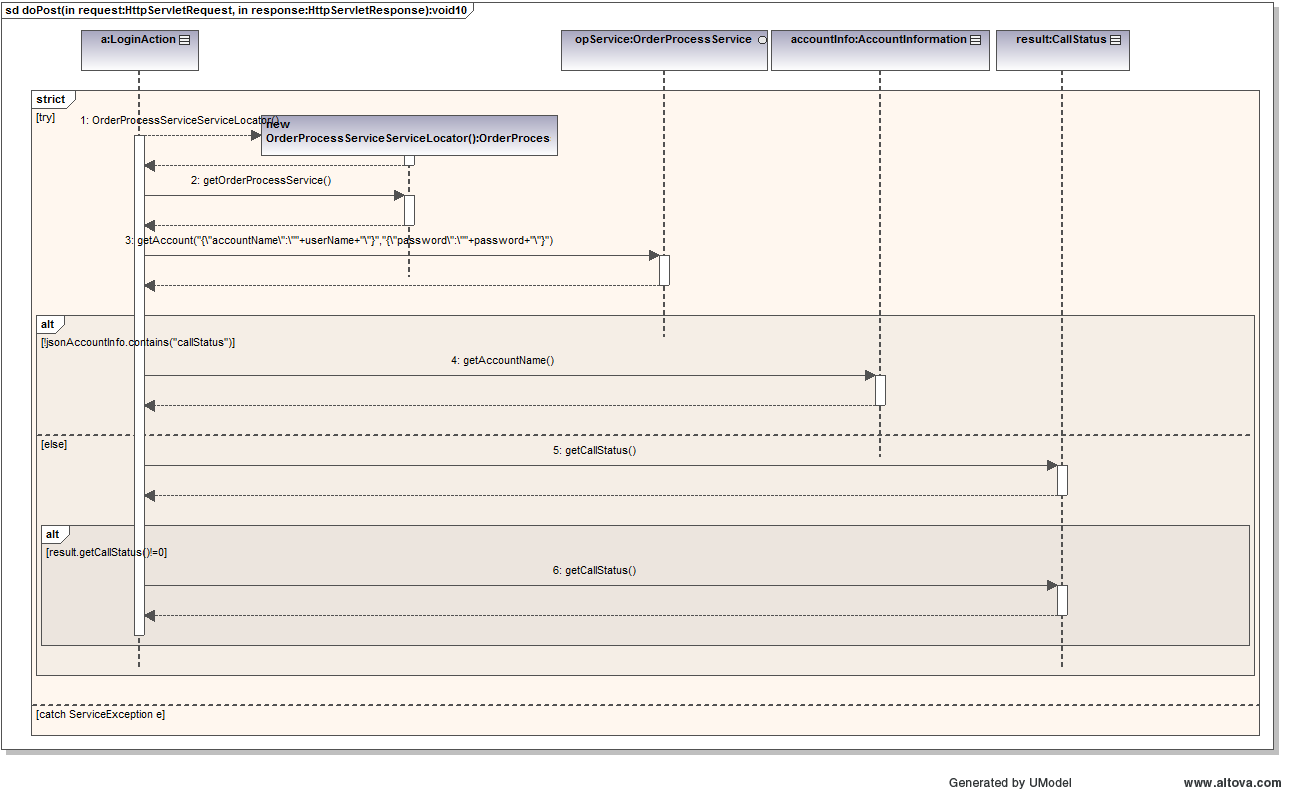
#### 5.2.3 createOrder Sequence Diagram



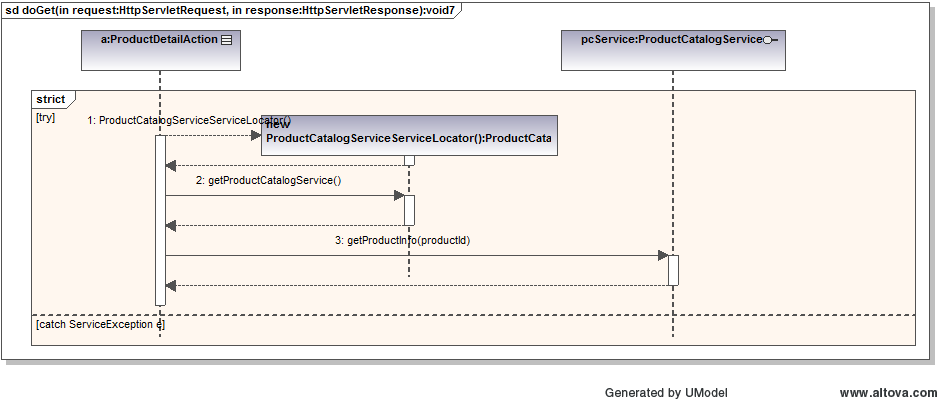
#### 5.2.4 confirmOrder Sequence Diagram



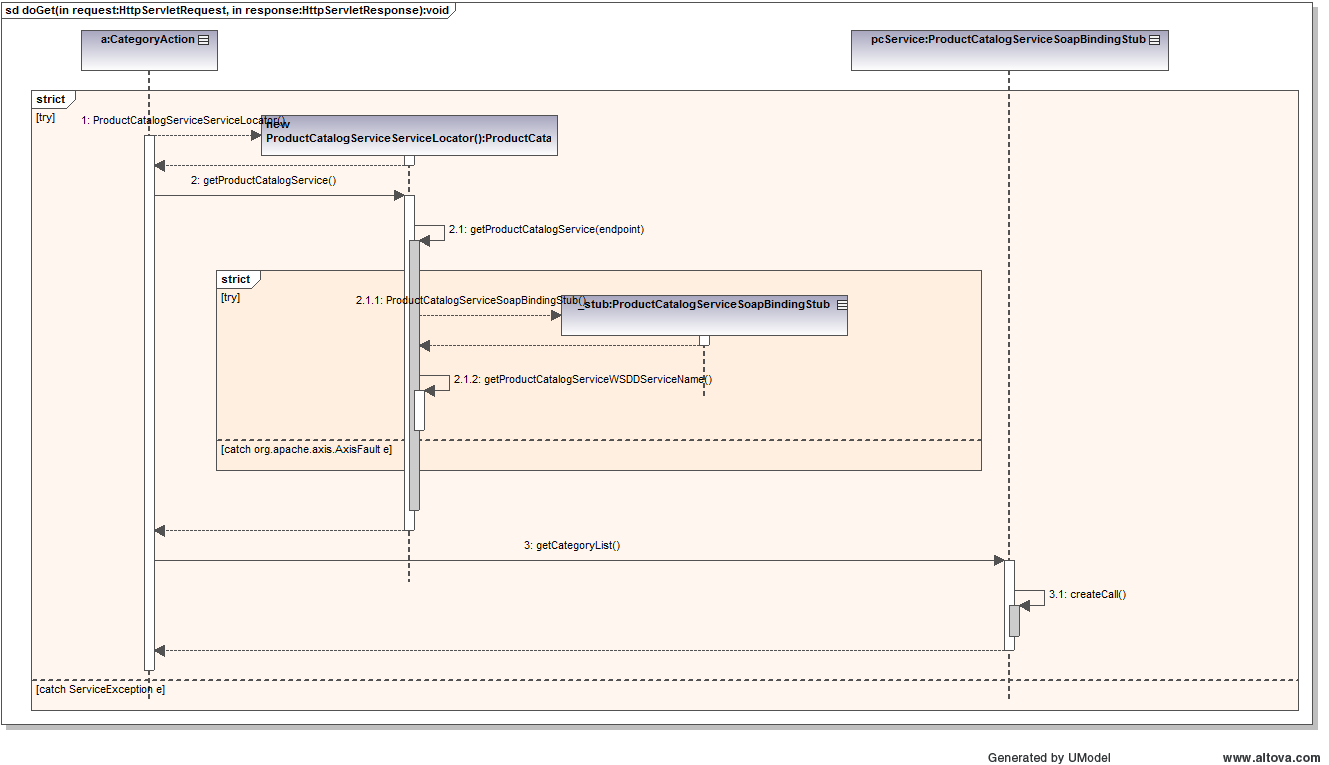
### 5.3 LoginAction Controller



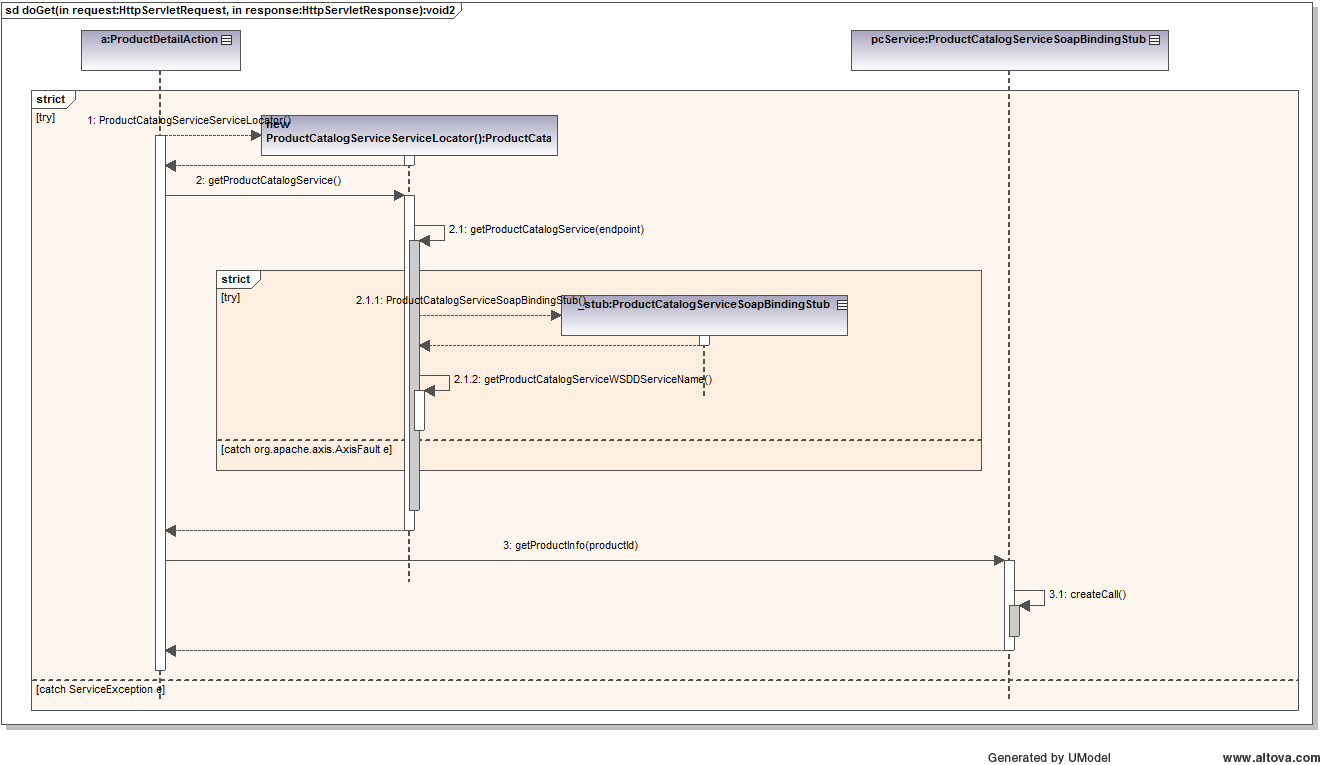
### 5.4 ProductDetailAction Controller



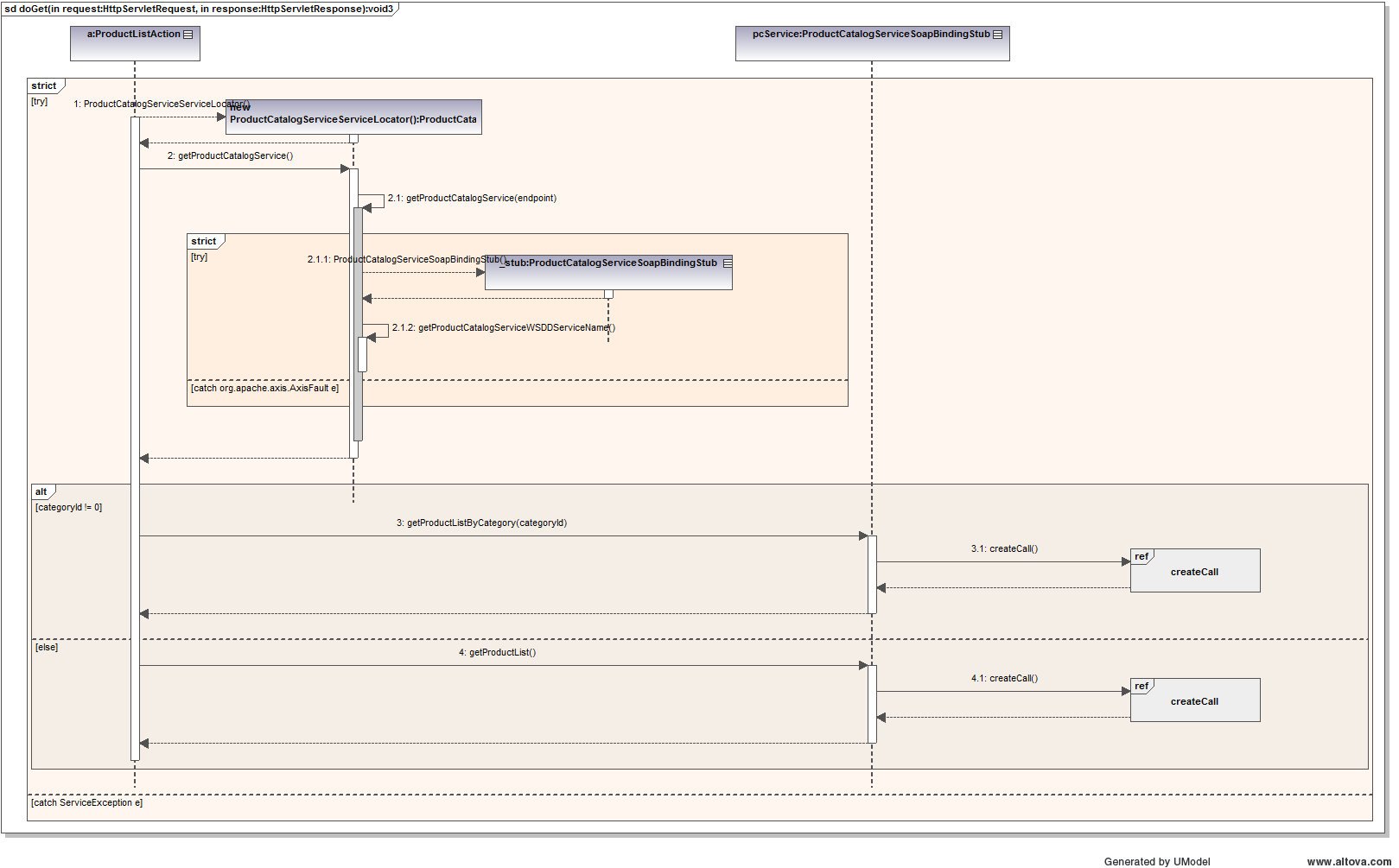
### 5.5 CategoryAction Controller



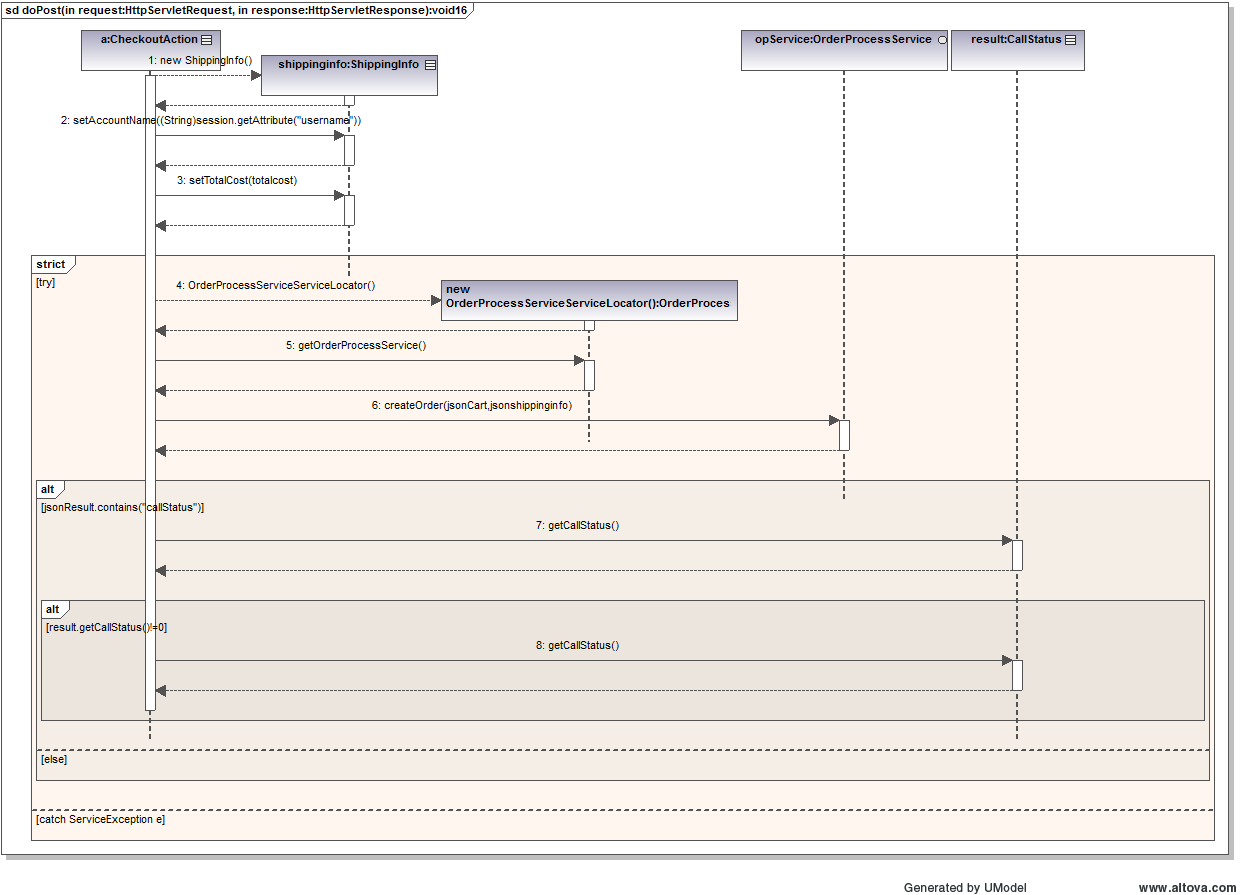
### 5.6 ProductDetailAction Controller



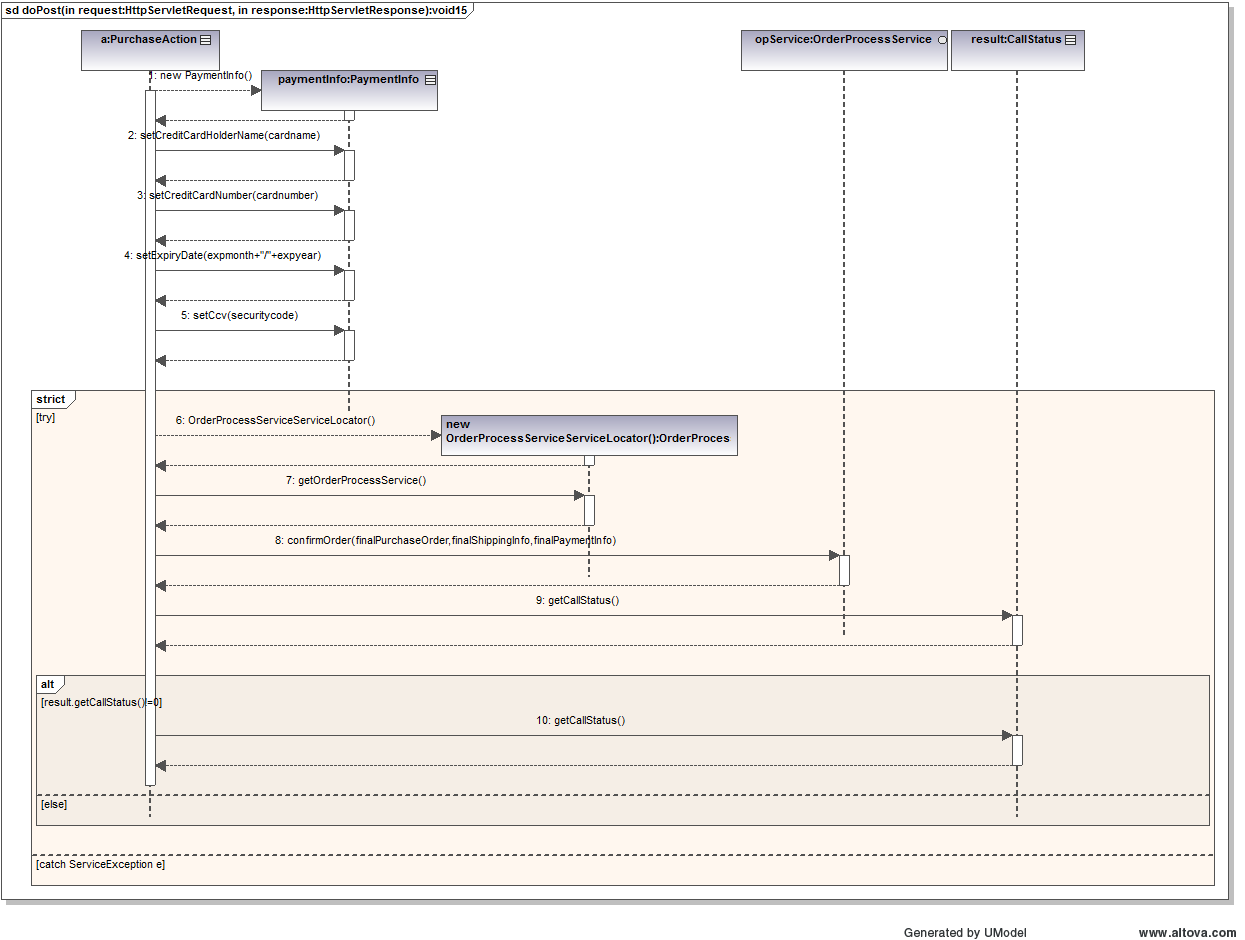
### 5.7 ProductListAction Controller



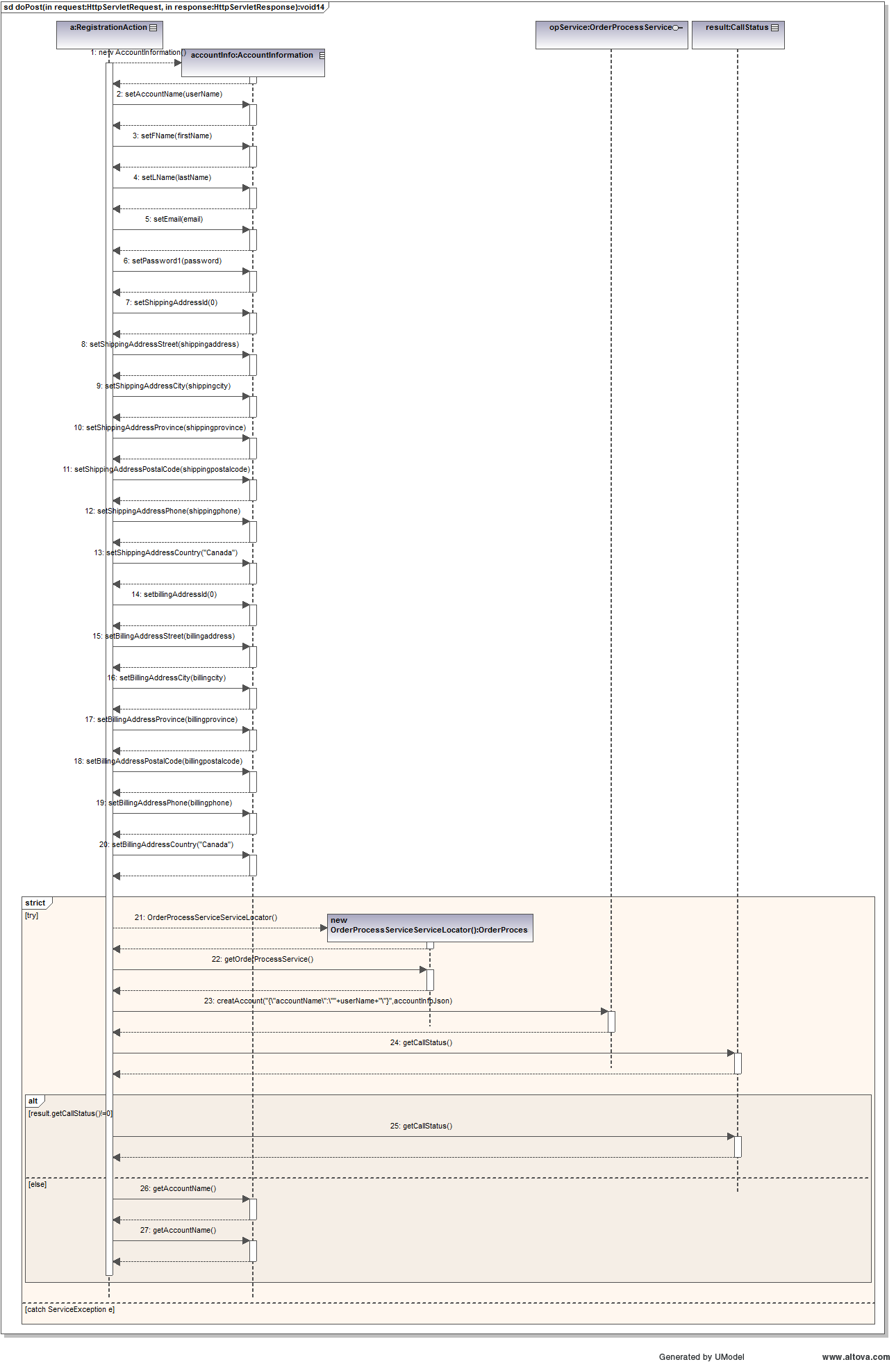
### 5.8 CheckOutAction Controller



### 5.9 PurchaseAction Controller



### 5.10 RegistrationAction Controller



## 6 Screenshots

## 7 Testing Framework

For this project, the Model was tested using Junit and SOAPUI. Included with the project and documentation is a spreadsheet which outlines the various Junit tests that were performed along with the expected outcomes, the actual outcomes and text files which contain the outputs for the test scripts. The spreadsheet also documents the tests that were performed using SOAPUI.

## 8 Individual Contribution towards the project

|  |  |
| --- | --- |
| **Team Member** | **Member's contribution** |
| Michele Belanger-Petrucci | Model design and implementation, documentation, testing framework, database design and implementation |
| Kaifan Zhu | Controller implementation, architecture design |
| Yuan Liu | HTML, javascript, jquery |
| Jiaren Suo | HTML, bootstrap, documentation |
| Cynthia Torkuma Ikongo | Documentation |