HỌC VIỆN KỸ THUẬT QUÂN SỰ

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÙI THỊ MAI HƯƠNG**

**KHÓA 12**

**HỆ: KỸ SƯ TIN HỌC**

**BÁO CÁO**

**THỰC TẬP CÔNG NGHIỆP**

NGÀNH CÔNG NGHỆ THÔNG TIN

MÃ SỐ: 01.00

**XÂY DỰNG VÀ BẢO TRÌ HỆ THỐNG**

**DIRECTV DIGITAL LOCKER WEB SERVICE (ASDL)**

***Cán bộ hướng dẫn:*** Bùi Hữu Phúc

Hà Nội, 05/2017

**Lời nói đầu**

Ngày nay, công nghệ thông tin đã và đang được áp dụng trong hầu hết các lĩnh vực đời sống. Theo đó, các công ty và doanh nghiệp về lĩnh vực này ngày một nhiều. Học phần Thực tập Công nghiệp giúp sinh viên tiếp xúc với môi trường làm việc thực tế, quy trình sản xuất và kịp thời nắm bắt các công nghệ được các doanh nghiệp áp dụng trong vận hành. Qua đó, sinh viên có được cái nhìn cụ thể hơn và định hình đúng hướng đi cho bản thân.

FPT Software là công ty phần mềm trong lĩnh vực tích hợp hệ thống, phân phối và bán lẻ, dịch vụ công nghệ thông tin, xuất khẩu gia công phần mềm và bán lẻ sản phẩm CNTT. Qua 2 tháng tham gia thực tập tại FPT software, em đã được làm quen với môi trường làm việc chuyên nghiệp, học hỏi thêm nhiều kiến thức mới và tiếp xúc với dự án thực tế Hệ thống DIRECTV digital locker Web service (ASDL). Em chọn đó làm đề tài báo cáo Thực tập Công nghiệp.

Em xin chân thành cảm ơn thầy Nguyễn Văn Giang cùng các thầy cô giáo khoa Công nghệ thông tin trường Học viện Kỹ thuật Quân sự đã hướng dẫn và tận tình giúp đỡ em trong quá trình học tập tại trường và trong thời gian thực tập này.

Em xin chân thành cảm ơn Ban Giám đốc cùng toàn thể nhân viên FPT, đặc biệt là bộ phận CME.CCA đã tạo điều kiện giúp đỡ em hoàn thành tốt kỳ thực tập của mình.

Với kinh nghiệm non nớt, cùng với kiến thức còn hạn chế, tuy đã cố gắng nỗ lực hết mình song báo cáo đề tài của em vẫn không tránh khỏi những thiếu sót. Em rất mong nhận được những đóng góp của Quý thầy cô và các bạn để rút kinh nghiệm và hoàn thiện hơn.

*Em xin chân thành cảm ơn!*

**Phụ lục**

[**I.** **Tên đề tài** 4](#_Toc483212734)

[**II.** **Nơi thực tập** 4](#_Toc483212735)

[**III.** **Nội dung thực tập** 4](#_Toc483212736)

[**1.** **Mục đích của quá trình thực tập** 4](#_Toc483212737)

[**2.** **Các nội dung đã tham gia** 4](#_Toc483212738)

[**IV.** **Kết quả đạt được** 5](#_Toc483212739)

[**1.** **Nghiên cứu đề tài, thu thập tài liệu** 5](#_Toc483212740)

[**2.** **Báo cáo chi tiết nội dung đã tiến hành trong quá trình thực tập** 5](#_Toc483212741)

[**2.1** **Tìm hiểu công nghệ Web service (REST và SOAP)** 5](#_Toc483212742)

[**2.2** **Tìm hiểu business logic của một API authorize trong hệ thống ASDL, tìm bộ data tối thiểu và vẽ flowchart.** 8](#_Toc483212743)

[**2.3** **Xây dựng API authorize kích hoạt hệ thống ASDL và optimize code.** 11](#_Toc483212744)

[**2.4** **Unit test sử dụng JUnit trong java.** 11](#_Toc483212745)

[**3.** **Xây dựng chương trình** 13](#_Toc483212746)

[**3.1** **Xây dựng service sử dụng REST** 13](#_Toc483212747)

[**3.2** **Xây dựng service sử dụng SOAP** 17](#_Toc483212748)

[**3.3** **Xây dựng API authorize** 18](#_Toc483212749)

[**3.4** **Viết unit test sử dụng JUnit trong java** 23](#_Toc483212750)

[**4.** **Những khó khăn trong quá trình thực tập** 25](#_Toc483212751)

[**5.** **Kiến nghị** 25](#_Toc483212752)

1. **Tên đề tài**

Đề tài: Xây dựng và bảo trì Hệ thống DIRECTV Digital Locker Web service (ASDL).

1. **Nơi thực tập**

Đơn vị thực tập: Công ty TNHH Phần mềm FPT – FPT software

Bộ phận thực tập: CME.CCA

Địa chỉ: Số 17 Duy Tân, Cầu Giấy, Hà Nội

Website: www.fpt-software.com

1. **Nội dung thực tập**
2. **Mục đích của quá trình thực tập**

Thực tập công nghiệp là cơ hội giúp sinh viên tiếp xúc với môi trường làm việc thực tế, quy trình sản xuất và kịp thời nắm bắt các công nghệ được các doanh nghiệp trong ngành Công nghệ thông tin áp dụng trong vận hành. Áp dụng tư tưởng đó, mục đích của quá trình thực tập tại FPT software bao gồm:

* Về kiến thức:
  + Tìm hiều công nghệ và nghiệp vụ của hệ thống DIRECTV Digital Locker Web service.
  + Hiểu và biết cách vận dụng các dịch vụ web: REST và SOAP
  + Lập trình bằng ngôn ngữ Java thành thạo.
* Về kỹ năng:
  + Làm quen với môi trường làm việc chuyên nghiệp
  + Rèn luyện kỹ năng làm việc nhóm, kỹ năng báo cáo
  + Học hỏi kinh nghiệm
  + Tham gia các buổi Daily meeting

1. **Các nội dung đã tham gia**

* Tìm hiểu công nghệ Web service (REST và SOAP)
* Tìm hiểu business logic của một API authorize trong hệ thống ASDL, tìm bộ data tối thiểu và vẽ flowchart.
* Xây dựng API authorize kích hoạt hệ thống ASDL.
* Optimize code
* Unit test sử dụng JUnit trong java.

1. **Kết quả đạt được**
2. **Nghiên cứu đề tài, thu thập tài liệu**

DIRECTV là một trong những nhà cung cấp dịch vụ giải trí truyền hình kỹ thuật số hàng đầu thế giới cung cấp trải nghiệm video chất lượng cao thông qua công nghệ hiện đại, lập trình chưa từng có và dịch vụ khách hàng hàng đầu cho hơn 37 triệu khách hàng ở Hoa Kỳ và Mỹ Latinh.

DIRECTV Digital Locker Web service (ASDL) là một ứng dụng Java Enterprise chịu trách nhiệm quản lý các Locker Entry thuộc về quyền lợi kỹ thuật số của khách hàng. Trong đó, mỗi Locker Entry được gắn với tài khoản với thông tin về ủy quyền của tài khoản cũng như các thiết bị được phép với chính sách nhất định. ASDL có thể truy cập bằng cách sử dụng công nghệ web REST và HTTPS. ASDL nằm trong Head-End (HE) của DIRECTV ở cả CRBC BSO và POP. Các chức năng chính của ASDL là:

* Provisioning: Cung cấp và cho phép các hệ thống HE khác lưu trữ các ủy quyền theo một tài khoản cụ thể.
* Authorizing: Ủy quyền - Cho phép khách hàng truy cập và vận dụng các quyền của khách hàng đó.

Trong khuôn khổ đề tài, em sẽ nghiên cứu chức năng Authorizing.

Trình điều khiển hàng đầu cho chức năng này là đáp ứng yêu cầu của dự án Download and Go (DNG)/ Electronic Sell-Through. Một trong những thiết kế của dự án DNG là thiết kế hệ thống có thể được tận dụng qua nhiều dự án, các mục locker (quyền lợi) được lưu trữ trong tủ số kỹ thuật số của khách hàng sẽ lưu trữ ID Policy thay vì ID CAMC đã sử dụng trước đây (các dịch vụ và trả cho mỗi lần xem). Sử dụng ID Policy cho phép ASDL lưu trữ các Locker Entry thay cho nhiều quyền lợi khác nhau.

1. **Báo cáo chi tiết nội dung đã tiến hành trong quá trình thực tập**
   1. **Tìm hiểu công nghệ Web service (REST và SOAP)**

Web service – Dịch vụ web là một hệ thống phần mềm được thiết kế để hỗ trợ khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau thông qua mạng internet, giao diện chung và sự gắn kết của nó được mô tả bằng XML. Dịch vụ web là tài nguyên phần mềm có thể xác định bằng địa chỉ URL, thực hiện các chức năng và đưa ra các thông tin người dùng yêu cầu. Một dịch vụ web được tạo nên bằng cách lấy các chức năng và đóng gói chúng sao cho các ứng dụng khác dễ dàng nhìn thấy và có thể truy cập đến những dịch vụ mà nó thực hiện, đồng thời có thể yêu cầu thông tin từ dịch vụ web khác. Nó bao gồm các modul độc lập cho hoạt động của khách hàng và doanh nghiệp, bản thân nó được thực thi trên server.

Web service cung cấp khả năng tương tác giữa các ứng dụng trên các máy tính khác nhau thông qua mạng internet. Web service cho phép client và server tương tác được với nhau ngay cả trong những môi trường khác nhau, dữ liệu trả về thường dưới dạng xml hoặc json.

Hai giải pháp web service chính để giao tiếp với các ứng dụng web là: REST (Representational State Transfer) và SOAP (Simple Object Access Protocol)

1. REST:

REST web service sử dụng HTTP để truyền nhận dữ liệu trực tiếp thông qua URI

Ví dụ: http://localhost:8080/ListPeople/rest/people

REST là một kiểu cấu trúc, không phải một giao thức – protocol, cho phép thao tác dữ liệu dưới các phương thức được hỗ trợ trong HTTP: GET, POST, PUT, DELETE để quy định rõ cách thao tác dữ liệu trên server.

|  |  |  |
| --- | --- | --- |
| Method | Annotation | Mô tả |
| Get | @GET | Truy xuất đến tài nguyên máy chủ |
| Post | @POST | Tạo một tài nguyên trên máy chủ |
| Put | @PUT | Thay đổi trạng thái tài nguyên hoặc cập nhật |
| Delete | @DELETE | Hủy bỏ một tài nguyên |

* Cơ chế hoạt động:
  + - Client gửi request thông qua HTTP
    - Web service dựa trên request để đáp ứng yêu cầu và kết quả trả về
    - Response được trả về theo đúng định dạng yêu cầu của client
    - Client xử lý kết quả response để đáp ứng yêu cầu của mình
* Định nghĩa 2 loại resource: root và sub resource thông qua @PATH. Trong đó sub resource được truy cập thông qua root
  + - Root resource thường là khai báo ở đầu class.
    - Sub resource được khai báo trên các method, các method được truy cập thông qua root theo đường dẫn trên server.
* Cú pháp @PATH: @PATH(“resource/{param1}/…/{paramN}”)
* Để truy xuất dữ liệu từ HTTP request và đưa vào service – resource xử lý:
  + - @PathParam(): Lấy giá trị từ URI trên đường dẫn từ sau dấu /
    - @QueryParam(): Lấy query parameter từ request, lấy giá trị từ URI từ sau dấu ? và &
    - @FormParam(): Lấy query parameter nhưng được truyền từ form parameters.
* Để xác định loại dữ liệu được xử lý trên server hay lấy về từ server:
  + @Produces():
    - Xác định định dạng dữ liệu khi phương thức xử lý và trả về
    - Định dạng quy định: text/plain, text/xml, application/xml hay text/json
    - Áp dụng cho các phương thức @GET, @POST, @PUT
  + @Consumes():
    - Xác định loại dữ liệu được gửi từ client đến server cho xử lý trên server.
    - Các định sạng hỗ trợ: text/plain, text/xml, application/x-www-form-urlencoded
    - Áp dụng cho đầy đủ 4 phương thức của RESTful

1. SOAP:

SOAP là giao thức giao tiếp có cấu trúc như XML, cho phép ứng dụng trao đổi thông tin qua HTTP.

* Khai báo @WebService ở đầu các class chứa các method web
* Khai báo @WebMethod ở đầu các web method.

Sử dụng WSDL để mô tả các dịch vụ web theo cấu trúc của XML, bao gồm các thông tin: tên dịch vụ, giao thức và kiểu mã hóa, thông tin về tham số, kiểu dữ liệu…

Khi client muốn kết nối tới các dịch vụ web có thể đọc WSDL để xác định những chức năng có sẵn trên server. Sau đó, client sử dụng SOAP để lấy ra chức năng chính xác có trong WSDL.

1. So sánh REST và SOAP

|  |  |  |
| --- | --- | --- |
| STT | SOAP | REST |
| 1 | SOAP là 1 giao thức | REST là một cách kiến trúc |
| 2 | SOAP cung cấp giao diện dịch vụ cho các thành phần bên ngoài sử dụng | REST sử dụng địa chỉ URI để cung cấp dịch vụ |
| 3 | Java API: jax-ws | Java API: jax-rs |
| 4 | SOAP định nghĩa các chuẩn và quy tắc chặt chẽ | Không định nghĩa nhiều chuẩn như SOAP |
| 5 | Sử dụng băng thông và nhiều tài nguyên hơn REST | Sử dụng băng thông và tài nguyên ít hơn REST |
| 6 | Bảo mật, tin cậy | Ít tin cậy hơn |
| 7 | SOAP chỉ hỗ trợ định dạng dữ liệu XML | REST hỗ trợ các định dạng khác nhau: text, html, xml, json |
| 8 | Ít được sử dụng hơn REST | Được ưa chuộng hơn SOAP |
| 9 | Hỗ trợ 2 giao thức HTTP và SMTP | REST gắn với giao thức HTTP |
| 10 | Được thiết kế dùng trong tính toán phân tán | Không sử dụng trong tính toán phân tán, REST hữu ích cho các thiết bị có giới hạn (như điện thoại di động..) |

* 1. **Tìm hiểu business logic của một API authorize trong hệ thống ASDL, tìm bộ data tối thiểu và vẽ flowchart.**
* Business logic:

Authorizing: là một trong những chức năng chính của hệ thống ASDL cung cấp các chính sách cho phép khách hàng truy cập và vận dụng các quyền của khách hàng.

Locker Entry là tập các thông tin gắn với tài khoản khách hàng về các chính sách cũng như các thiết bị được phép với chính sách nhất định.

* Các trường sử dụng:

Card ID: ID của loại thiết bị khách hàng sử dụng.

Account ID: Id tài khoản khách hàng.

Business Unit: mã đơn vị sử dụng, có thể là US hoặc PR

Authorization Start: thời điểm bắt đầu gửi yêu cầu.

Authorization End: thời điểm kết thúc

CAMC service: dịch vụ được cung cấp

Policy ID: id chính sách

Event code: tập các sự kiện

UPDA, HCAS là các hệ thống bên ngoài lưu trữ các Policy và Cap – chính sách và thiết bị được cung cấp.

* Chức năng authorize xử lý các logic:

Chuẩn hóa Locker Entry để chuyển đổi các thông số đầu vào được cung cấp thành các giá trị cho các thông số thực sự cần thiết cho Digital Locker để tồn tại.

* + Nếu Locker Entry không cung cấp Subscriber hoặc Card ID ta sẽ chuyển đổi thành Business Unit và Account ID.
  + Nếu không cung cấp Authorization Start, sẽ lấy giá trị current timestamp- thời gian hiện tại.
  + Nếu không cung cấp, thiết lập Authorization End = Authorization Start + configured delay.
  + Nếu không cung cấp Policy ID, chuyển đổi từ CAMC service hoặc Event Code. Gọi sang hệ thống UPDA – Policy Service tra cứu Policy tương ứng.

Nếu LockerID hoặc PolicyID được cung cấp, tiến hành lưu trữ Locker Entry.

* Thiết đặt lại các giá trị ở các trường được cung cấp thay vì các giá trị hiện có.

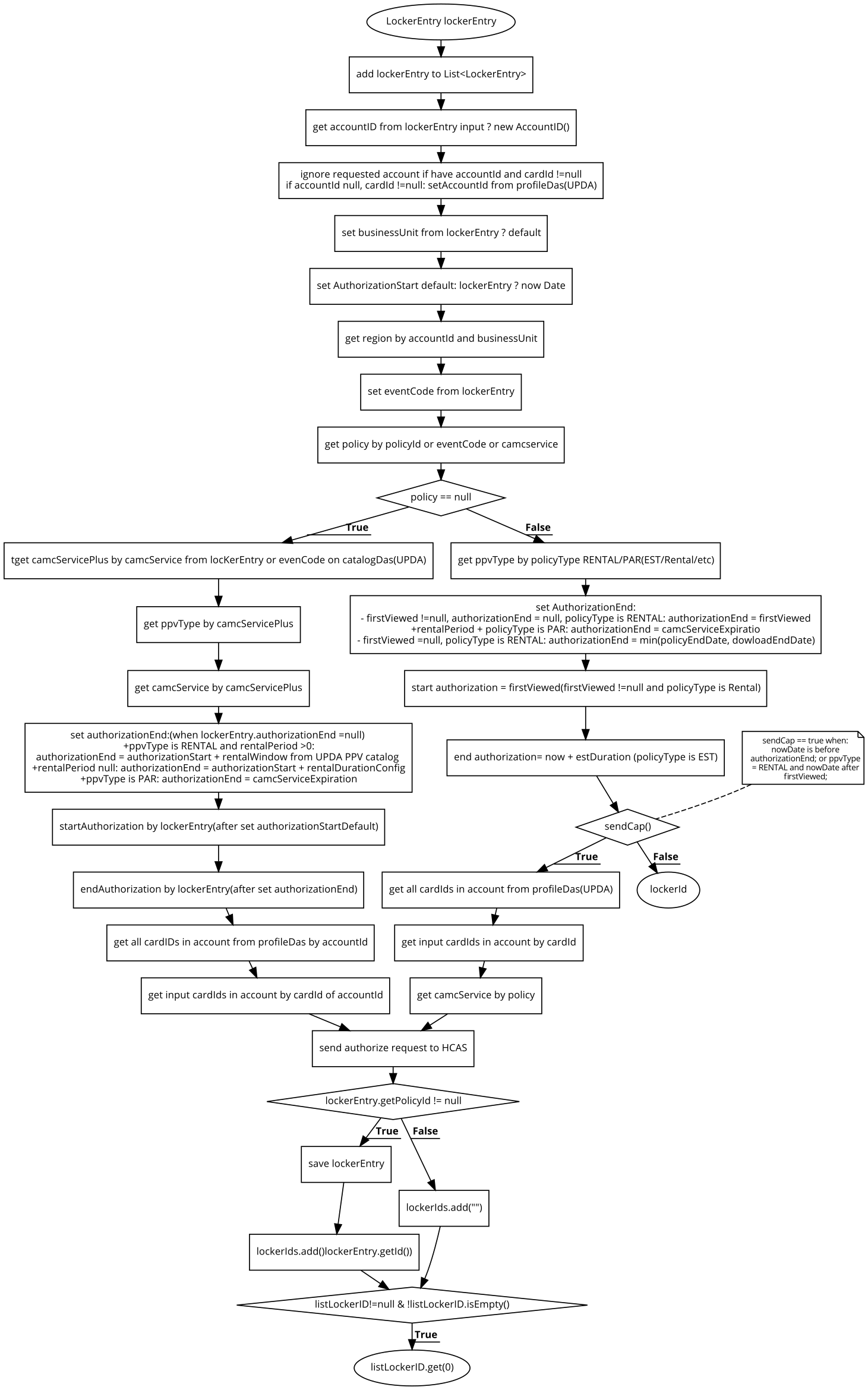
Nhận chính sách từ Policy Service.

Nếu chính sách là RENTAL (thuê), gửi yêu cầu ủy quyền đến HCAS cho dịch vụ CAMC có thể mua được.

Transform Digital Locker, chuyển Locker kỹ thuật số đến UPDA Locker.

Cập nhật kho lưu trữ UPDA.

* Bộ data tối thiểu bao gồm: AccountID, LockerID, PolicyID.
* Flowchart:



* 1. **Xây dựng API authorize kích hoạt hệ thống ASDL và optimize code.**

Sau khi phân tích logic API authorize, thực hiện xây dựng API authorize kích hoạt cho hệ thống ASDL với logic như sau:

Input: Locker Entry với các thông số AccountID, LockerID, BusinessUnit, PolicyID được cung cấp.

Output: LockerID của lockerEntry truyền vào, sau khi lưu vào hệ thống UPDA.

Logic: Xây dựng cho trường hợp policy!= null

* Lấy policy theo policyId được cung cấp
* Lấy PpvType theo policy
* Lấy lockerEntry từ LockerEntry Service trong hệ thống UPDA theo accountId, lockerId được cung cấp
  + Gọi LockerEntry Service bằng SOAP theo URL:

<http://upda-osb-pi.dtveng.net:80/upda/LockerService?wsdl>

Port: LockerServicePortBindingQSService

* Reconcile: cập nhật thông tin lockerEntry truyền vào theo LockerEntry trong UPDA, những thuộc tính không được cung cấp ban đầu sẽ được cập nhật theo lockerEntry trong UPDA.
* Thiết lập lại authorization End
  + ﬁrstViewed !=null, authorizationEnd = null, policyType is RENTAL:

authorizationEnd = ﬁrstViewed

+rentalPeriod + policyType is PAR:

authorizationEnd = camcServiceExpiration

* + ﬁrstViewed =null, policyType is RENTAL:

authorizationEnd = min(policyEndDate, dowloadEndDate)

* Lấy authorization end được thiết đặt trên Cap

Nếu policyType = “EST”, authorizationEnd= currentDate+24 giờ

* Kiểm tra sendCap và lưu locker entry vào UPDA
  + sendCap = true khi thời điểm authorizationEnd ở sau thời điểm hiện tại hoặc ppvType = RENTAL và nowDate after firstViewed
  + Lưu lockerEntry và trả về kết quả qua ResponseStatus và NameValuePair
* Lấy lockerId từ locker entry vừa lưu.
  1. **Unit test sử dụng JUnit trong java.**
* Tìm hiểu unit test – kiểm thử đơn vị

Một Unit là một thành phần phần mềm nhỏ nhất mà ta có thể kiểm tra được. Theo định nghĩa này, các hàm (function), thủ tục (Proceduce), lớp (Class), hoặc các phương thức (Method) đều có thể được xem là Unit.

Unit được chọn để kiểm tra thường có kích thước nhỏ và chức năng hoạt động đơn giản, không khó khăn gì trong việc tổ chức, kiểm tra, ghi nhận và phân tích kết quả kiểm tra. Nếu phát hiện lỗi, việc xác định nguyên nhân và khắc phục cũng tương đối dễ dàng vì chỉ khoanh vùng trong một đơn thể Unit đang kiểm tra. Thời gian tốn cho Unit Test sẽ được đền bù bằng việc tiết kiệm rất nhiều thời gian và chi phí cho việc kiểm tra và sửa lỗi ở các mức kiểm tra sau đó.

Unit test thường do lập trình viên thực hiện. Mục đích của Unit Test là bảo đảm thông tin được xử lý và xuất khỏi Unit là chính xác, trong mối tương quan với dữ liệu nhập và chức năng của Unit. Điều này thường đòi hỏi tất cả các nhánh bên trong Unit đều phải được kiểm tra để phát hiện nhánh phát sinh lỗi. Một nhánh thường là một chuỗi các lệnh được thực thi trong một Unit, ví dụ: chuỗi các lệnh sau điều kiện If và nằm giữa then… else là một nhánh. Thực tế việc lựa chọn các nhánh để đơn giản hóa việc kiểm tra và quét hết Unit đòi hỏi phải có kỹ thuật, đôi khi phải dùng thuật toán để chọn lựa. Cũng như các mức kiểm tra khác, Unit Test cũng đòi hỏi chuẩn bị trước các tình huống (test case) hoặc kịch bản (script), trong đó chỉ định rõ dữ liệu vào, các bước thực hiện và dữ liệu mong chờ sẽ xuất ra.

* JUnit trong java

JUnit là một “Framework kiểm thử hồi quy” được sử dụng bởi các lập trình viên để thực hiện kiểm thử “đơn vị” trong java và đẩy nhanh tốc độ lập trình và nâng cao tốc độ lập trình cũng như chất lượng code, Framework JUnit có thể dễ dàng tích hợp với một số thành phần như: eclipse, maven, ant.

Trong nhiều trường hợp method được kiểm thử gọi đến những phương thức của interface nào đó, mà interface này chưa được cài đặt (implementing) bởi bất cứ lớp đối tượng nào. Khi đó, trong mã lệnh unit test cần giả lập (mocking) các hàm của interface được sử dụng. Để giả lập chức năng của các hàm trong interface, người ta thường sử dụng một số thư viện hỗ trợ kiểm thử như PowerMockito, JMock,… để giả lập một đối tượng cài đặt interface, giả lập lời gọi hàm đến đối tượng giả này và chỉ định kết quả trả lại cho lời gọi hàm giả lập đó.

Các hàm thường sử dụng trong PowerMockito:

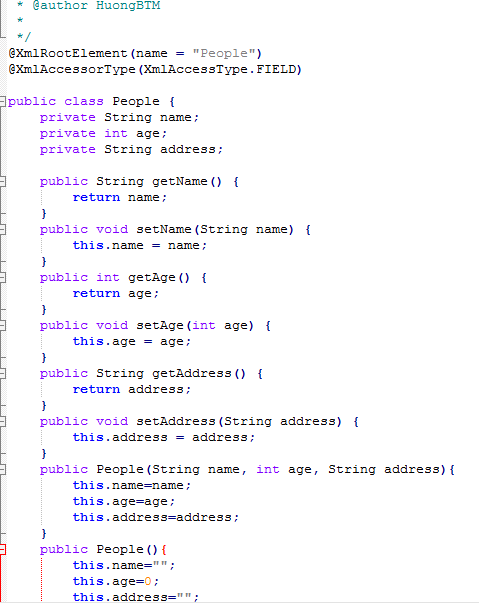
* PowerMockito.mock(class): giả lập đối tượng của lớp
* PowerMockito.when(T methodCall): dùng để giả lập một lời gọi hàm nào đó được sử dụng bên trong method đang được kiểm thử. Thường đi kèm với .thenReturn(), .thenAnswer() để chỉ định kết quả trả lại.

Để kiểm tra các kết quả được mong đợi, ta sử dụng các phương thức Assert():

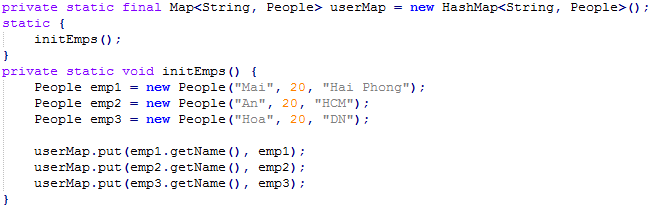
* assertEquals(): so sánh 2 giá trị để kiểm tra bằng nhau, test được chấp nhận nếu các giá trị bằng nhau
* assertFalse(): đánh giá biểu thử luận lý, test được chấp nhận nếu biểu thức sai
* assertNotNull(): so sánh tham chiếu của một đối tượng với null, test được chấp nhận nếu tham chiếu đối tượng khác null
* assertNull():so sánh tham chiếu của một đối tượng với null, test được chấp nhận nếu tham chiếu đối tượng là null
* assertTrue(): đánh giá một biểu thức luận lý, test được chấp nhận nếu biểu thức đúng
* fail(): phương thức này làm cho test hiện hành thất bại, phương thức này thường được sử dụng khi xử lý ngoại lệ

1. **Xây dựng chương trình**
   1. **Xây dựng service sử dụng REST**

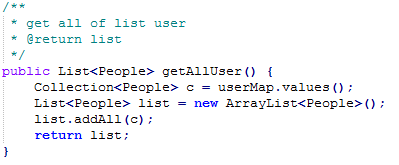
* Xây dựng class People, 3 thuộc tính: name, age, people

****

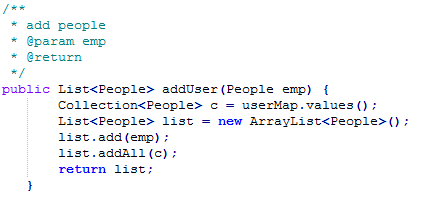
* Xây dựng controller xử lý các logic:
  + Khởi tạo 1 Map các people:

****

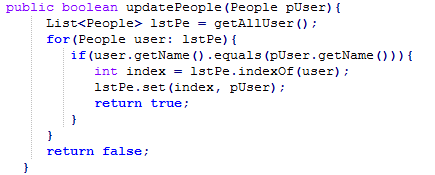
* + Xây dựng hàm lấy thông tin tất cả People

****

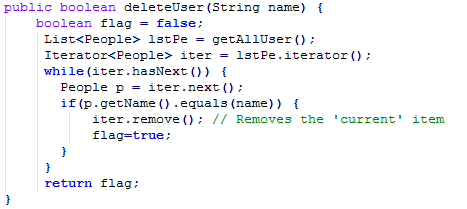
* + Xây dựng hàm thêm 1 people

****

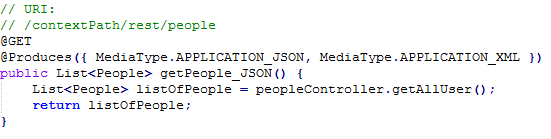
* + Xây dựng hàm cập nhật 1 people

****

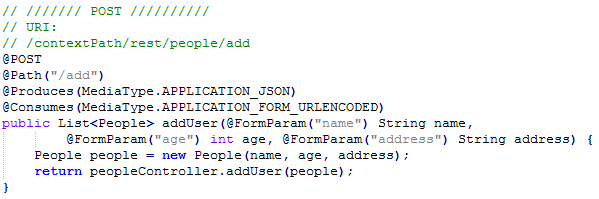
* + Xây dựng hàm xóa 1 people

****

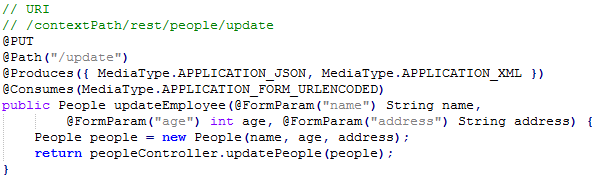
* Xây dựng service cho các method GET, POST, PUT, DELETE
  + GET: lấy tất cả people trong Map, kết quả trả về dạng XML hoặc JSON



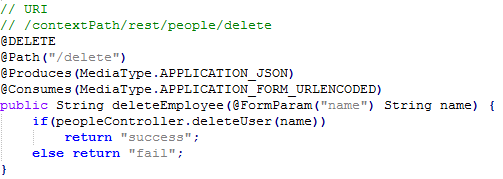
* + POST: thêm mới 1 people vào Map, input dạng FormParam, output là list danh sách các people, kết quả trả về dạng JSON



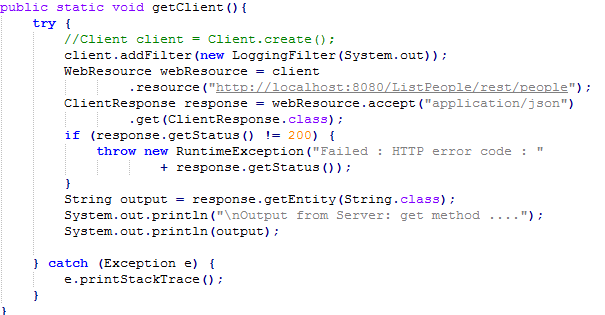
* + PUT: update thông tin people trong Map, input dạng FormParam, kết quả trả về dạng JSON hoặc XML



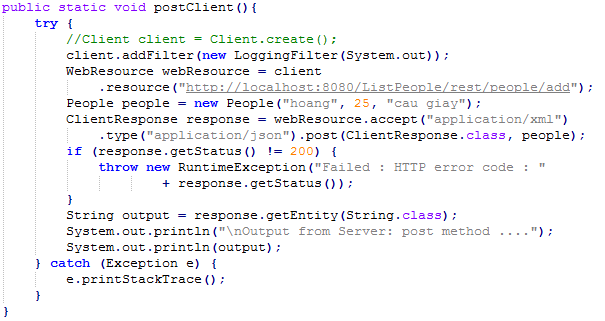
* + DELETE: xóa 1 people khỏi Map.



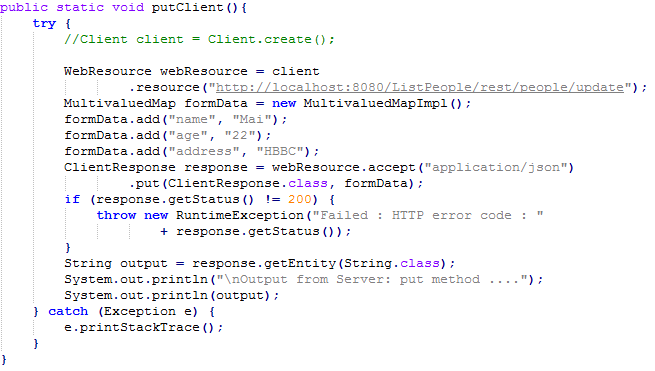
* Xây dựng client gọi đến các REST service:
  + Gọi GET method:



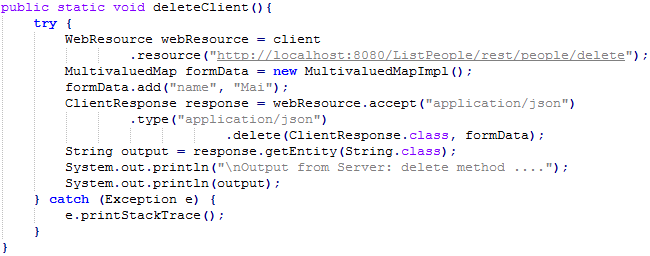
* + Gọi POST method:



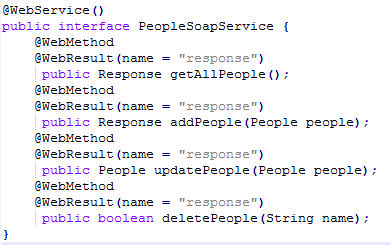
* + Gọi PUT method:



* + Gọi DELETE method:



* 1. **Xây dựng service sử dụng SOAP**
* Xây dựng interface định nghĩa các webMethod

****

* Xây dựng client: import file WSDL để kết nối tới các dịch vụ web.

public class SoapPeopleClient {

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

URL wsdlURL = new URL("http://localhost:8080/PeopleSoap/SoapService?wsdl");

QName qName = new QName("http://impl.service/","PeopleSoapServiceImplService");

Service service = Service.create(wsdlURL, qName);

People p1 = new People("Huong",22,"HBBC");

PeopleSoapService ps = service.getPort(PeopleSoapService.class);

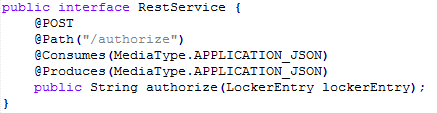
System.out.println("delete people status="+ps.getAllPeople());

System.out.println("add people status="+ps.adddUser(p1));

}

}

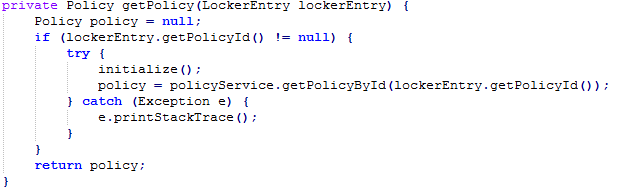
* 1. **Xây dựng API authorize**
* Xây dựng interface định nghĩa service

****

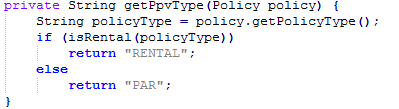
* Khởi tạo policyService và lockerDas vì các đối tượng này được gọi từ các service bên ngoài



* Lấy policy theo policyId được cung cấp



* Lấy PpvType theo policy

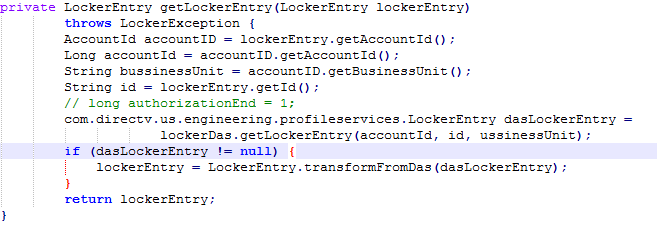


* Lấy lockerEntry từ LockerEntry Service trong hệ thống UPDA theo accountId, lockerId được cung cấp

Gọi LockerEntry Service bằng SOAP theo URL:

<http://upda-osb-pi.dtveng.net:80/upda/LockerService?wsdl>

Port: LockerServicePortBindingQSService



* Reconcile: cập nhật thông tin lockerEntry truyền vào theo LockerEntry trong UPDA, những thuộc tính không được cung cấp ban đầu sẽ được cập nhật theo lockerEntry trong UPDA.



* Thiết lập lại authorization End
  + ﬁrstViewed !=null, authorizationEnd = null, policyType is RENTAL:

authorizationEnd = ﬁrstViewed

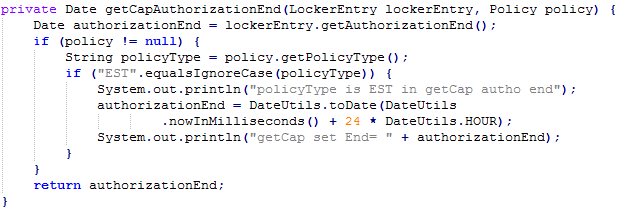
+rentalPeriod + policyType is PAR: authorizationEnd = camcServiceExpiration

* + ﬁrstViewed =null, policyType is RENTAL:

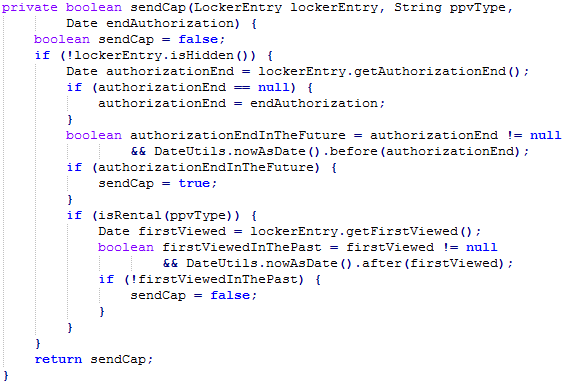
authorizationEnd = min(policyEndDate, dowloadEndDate)

* Lấy authorization end được thiết đặt trên Cap

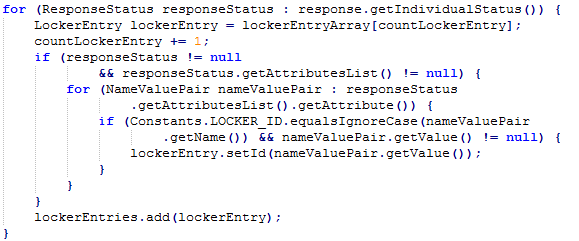
Nếu policyType = “EST”, authorizationEnd= currentDate+24 giờ



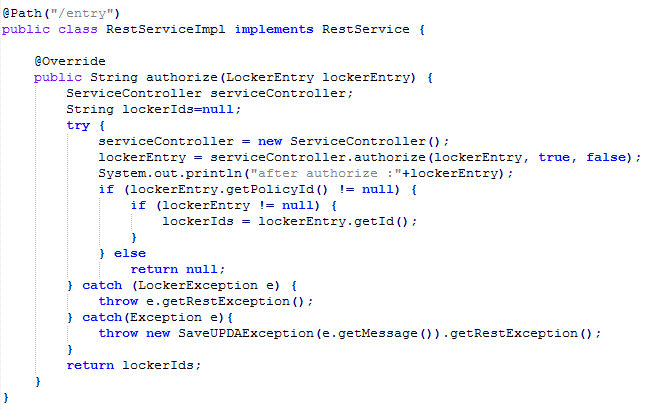
* Kiểm tra sendCap và lưu locker entry vào UPDA
  + sendCap = true khi thời điểm authorizationEnd ở sau thời điểm hiện tại hoặc ppvType = RENTAL và nowDate after firstViewed



* Lưu lockerEntry và trả về kết quả qua ResponseStatus và NameValuePair

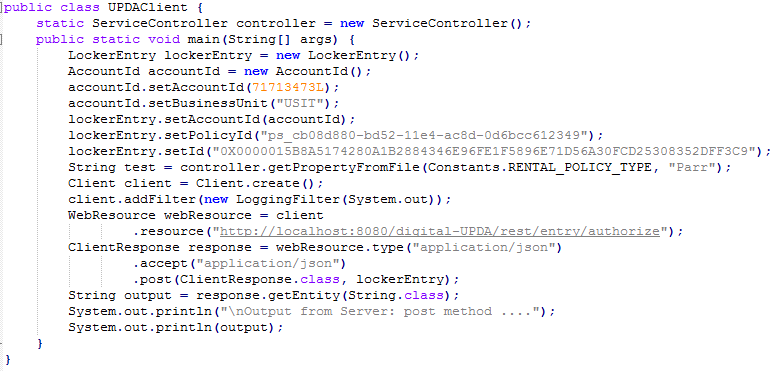


* Lấy lockerId từ locker entry vừa lưu.

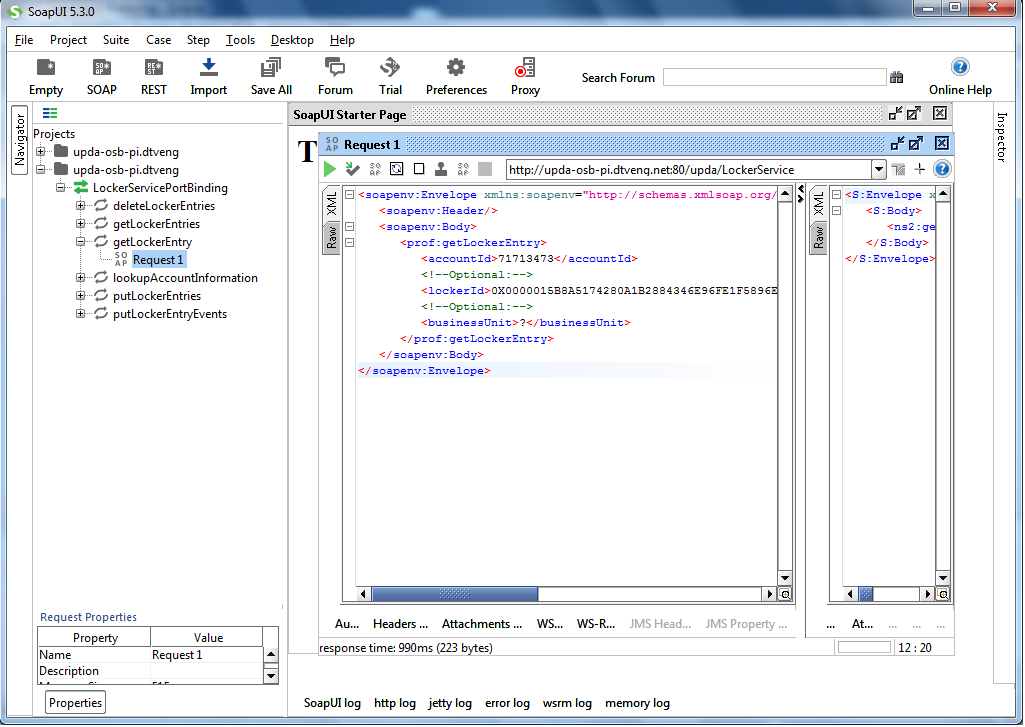


* Để gọi đến API authorize, ta gọi theo URL:

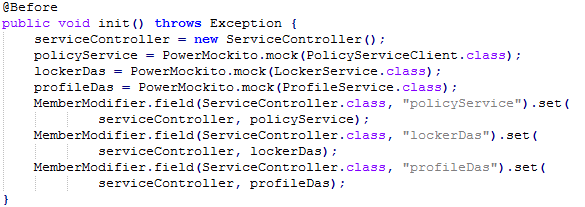
http://localhost:8080/digital-UPDA/rest/entry/authorize



* Sử dụng SOAPUI để test các service

****

* 1. **Viết unit test sử dụng JUnit trong java**
* Do trong API authorize có gọi đến các service bên ngoài, mà service đó chưa được cài đặt trong hệ thống, vì vậy cần giả lập các service này:

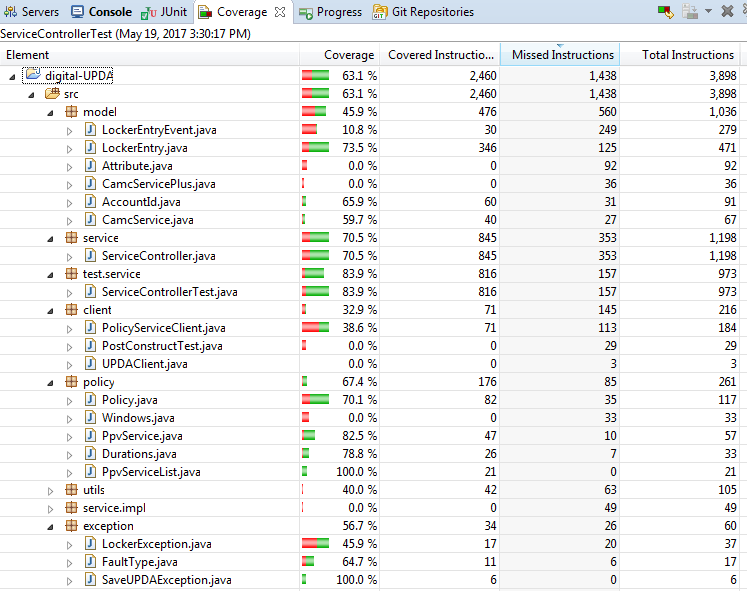


* Xây dựng TestMethod, trường hợp success: hàm test trả về đúng kết quả mong muốn, đi qua tất cả các case theo kịch bản:





* Sử dụng EclEmma theo dõi flow và các kết quả test:

****

1. **Những khó khăn trong quá trình thực tập**

* ASDL là hệ thống lớn, đã được phát triển trong thời gian dài, đồng thời gọi đến nhiều serervice bên ngoài, vì vậy khó khăn trong việc hiểu và theo sát hệ thống trong thời gian ngắn.

1. **Kiến nghị**

Thực tập công nghiệp có ý nghĩa thiết thực đối với sinh viên, em rất mong nhà trường tăng thêm thời gian thực tập và tạo nhiều cơ hội hơn cho sinh viên để phát triển kỹ năng và rèn luyện kiến thức.

Kết thúc thời gian thực tập theo lịch nhà trường, em rất mong được tiếp tục ở lại đơn vị thực tập học hỏi thêm nhiều kiến thức thực tế hơn nữa.

Em xin chân thành cảm ơn./.