2013年度　卒業研究
学内Webサービスの利用環境改善の検討

指導教員　渡辺　恭人

学籍番号　1040037

氏名　藤木公揮

提出日：2013年12月16日

目次

1.　背景・目的

1-1　背景

1-1-1　ログイン

1-2　目的

2.　現状と問題点

2-1　現状

2-1-1　WWW 学生サービス

2-1-2　CUCWebMail

2-1-3　Wellness System

2-1-4　ログインの移管について

3.　解決方法

3-1　解決へのアプローチ

3-2　シングルサインオン（SSO）とは

3-3　SSO の背景

3-4　シングルサインオンの方式

4.　関連研究

4-1　OpenAM

4-1-1 OSS Tech オープンソース・ソリューション・テクノロジ株式会社

4-2　既存の類似サービス

4-3　シングルサインオンに伴う危険性

4-4　学術認証フェデレーション

5.　システムの調査

5-1　OpenAM の試験導入

5-1-1　OpenAM Server のインストール

5-1-2　OpenAM Server の設定

5-1-3　ユーザー設定

5-1-4　Web エージェントの作成

5-1-5　Web サーバーの設定

5-1-6　Windows と glassfish による OpenAM 導入方法

5-1-7　OpenAM と OAuth

5-2　動作の確認

5-3　動作の評価

5-4　考察

6.　まとめと今後の課題

6-1　まとめ

6-2　今後の課題

7.　参考文献

8.　謝辞

1.　背景・目的

1-1　背景

現在（2013年）インターネットの普及や発展によって、生活、社会、のさまざまな場面において情報化が行われている。我が大学にも情報化が進み実装されたオンライン履修システムがある。もはや大学の履修登録ですらオンライン化されている。そのようなオンラインサービスのなかで情報が混乱しないようにとられた政策の一つであり、私たちがインターネットで様々なサービスを受ける際に行う「ログイン」という行為がある。オンライン履修システムなど学内Webサービスもその行為を必要としているもののひとつである。その際、自分だけが知っているパスワードと ID などの認証情報入力し、サービスを提供する側がクライアントであるユーザーを認証する。ログインを通してのサービスの提供は今日のWebサービス上で当たり前の行為となっており、本学の学内Webサービスも例外ではない。オンライン履修登録にもWebサービスを利用しており、ログインがそこでも必要となっている。

1-1-1 ログイン

参考文献【1】

> ログインとは、コンピュータやインターネット上の様々なサービスを利用する際に、予め登録しておいたアカウント情報を用いて個々人のデータにアクセスする認証行為のことです。例えば、オンラインゲームではユーザー名とパスワードを入力することにより自分で作成したキャラクターデータにログインすることができます。そもそも、ログとは「log = 記録」という意味で、「log in」で記録の中、すなわち「記録を利用できる状態にする」という事なのです。その他にもログオン・サインインという言葉がありますが、どれも意味は同じです。

ログインを通してサービスを受けることによるメリットは多く、ログインを通し個人認識させることでユーザー個々人に対するサービスを提供できる。逆にログインをせずに受けられるサービスには個人に対するサービスが提供できない。たとえば動画配信サイト Youtube にはログインをせずとも動画を視聴することが可能となっているが、アカウントを作成しログインをすることによって自分のお気に入りの動画を記憶

させておき、好きな時にお気に入りの動画をすばやく見ることができたり、動画を配信しているユーザーを記憶していち早くそのユーザーが動画を投稿したことを知ることができたりするためログインを通して得るメリットは大きいといえる。しかし、ログインにはメリットだけでなくデメリットも存在する。個人を認証するために必要なログインは ID やパスワードを入力することで成立し、急いでいる時は認証情報の入力の労力などがある。また ID やパスワードを忘れてしまったり、盗まれてしまうこともあり、管理者側はパスワードを忘れたユーザーへの再発行やパスワードのヒントを提示するなど、認証のためのユーザーへのサービスが新たに必要となってくる。

1-2　目的

本研究ではログインにおける労力を極力軽減させる方法について検討・提案を行うことでよりよい学内 Web サービスの利便性を向上させ、本研究で提案される手法を適用することで、より快適な学内 Web サービスの利用環境を将来の入学する学生や在学中の学生に提供することを目指す。

各 Web サービス全てにより少ないログイン回数で利用できるサービスの提案をするとともに実際に稼働・運用・実験をし、サービスとして成り立つのかどうか実現可能性を検討し、より利便性の高い Web サービスの利用方法を目指す。また、既存のシステムだけで本当に満足なのかを考え、本学に導入すべきシステムについて検討し提案を行う。

2.　現状と問題点

2-1　現状

千葉商科大学には6000人を超える学生が一人ひとり別々のアカウントを持っている。各学生が本学の講義の履修登録、ウェルネスの履修予約、学籍アドレス（学内で学生は端末利用時及び､Webサービス利用時に学籍番号に基づいたアカウントを使用している。これを本論文では学籍アドレスと呼ぶ）によるメール確認等は学校が提供しているさまざまなWebサービスによって構築されている。各Webサービスは入学してから卒業をするまでの4年間（留学する学生は4年以上）という長期間にわたって利用する。各サービスはWeb上で行えるという利点によっていつでも6000人のどの学生でも自宅や学校から遠く離れた場所からどこでも利用できる。しかし、各Webサービスには本人を認証するためのログインが各Webサービスに必要となる。自身の学籍ID（学籍IDとは学籍番号を元に生成される7桁の文字列（例：b040000）のことである）と自身で設定したパスワードを入力するログインは各サービスに行わなくてはいけない。パスワードも半角英数字、一部の記号込みで構成することを強制されており、非常に入力に時間や労力がかかる。

2-1-1WWW学生サービス

千葉商科大学の学生が授業の履修を登録するためのWebサービス。年に2回、学期の始めに履修を登録するために使うだけでなく、就職支援のサービスもあるため履修を登録する機会が少なくなった4年次でも利用するので全学部全学年が共通で使う。本学の教員はログインを行うと所属ゼミの学生の情報などを見ることができ、それによって学生にアドバイスなどができるため、本学の学生だけでない多くのユーザーが利用していると言える。

WWW学生サービスは学籍ID、パスワード入力後にENTERキーを押してもログインすることができずキーボードからいちいち手を放してマウスで[ログイン]と書かれたボタンにマウスを運ばなくてはいけない。またこの履修登録サービスはログインしてからしばらく操作しない時間があると勝手にログアウトする仕様になっており、ゆっくりと様々な作業を行いながらの登録作業がしにくくなっている。(図2-1-1-5参照)

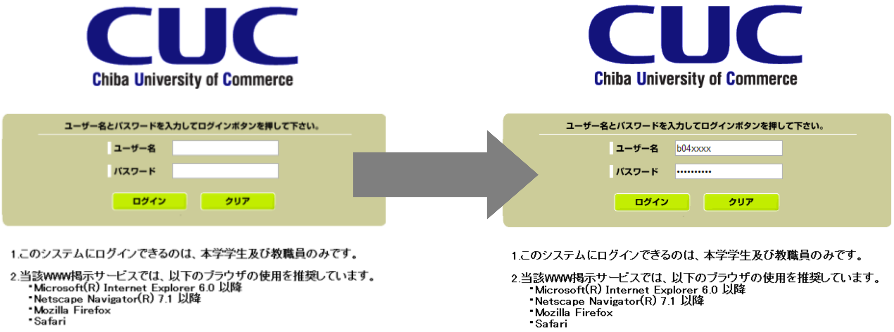

図 2-1-1-1: WWW 学生サービス（オンライン履修登録）画面のうちのひとつ。ログインのための情報を入力するページ

以下は様々な方法でログインを試みた場合に起こるエラー画面である。(図 2-1-1-2 ～図 2-1-1-4 参照)

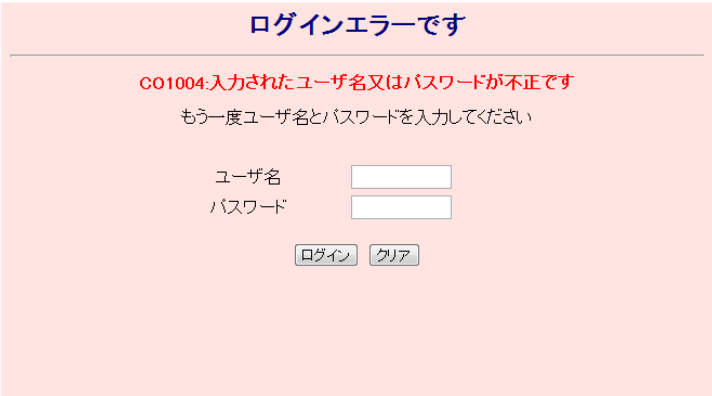

図 2-1-1-2: 学籍 ID とパスワードを入力しない状態でログインを試みた場合

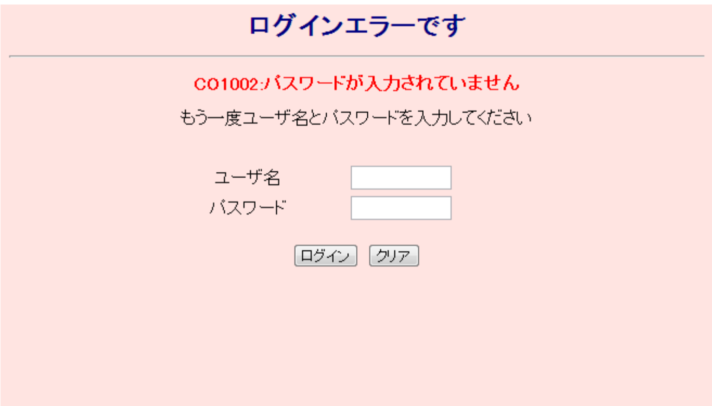

図 2-1-1-3: 学籍 ID のみ入力した場合

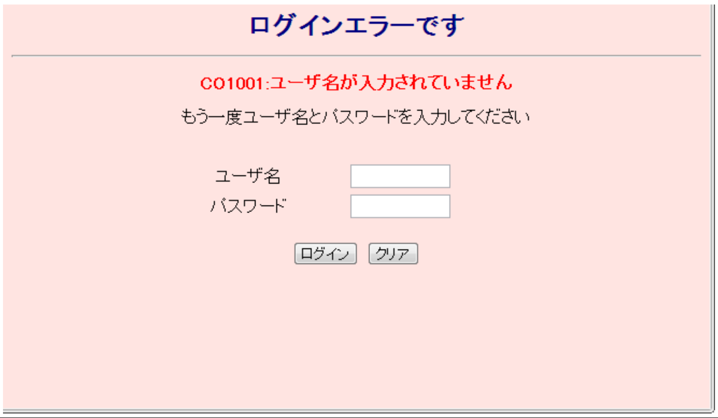

図 2-1-1-4: パスワードのみ入力した場合

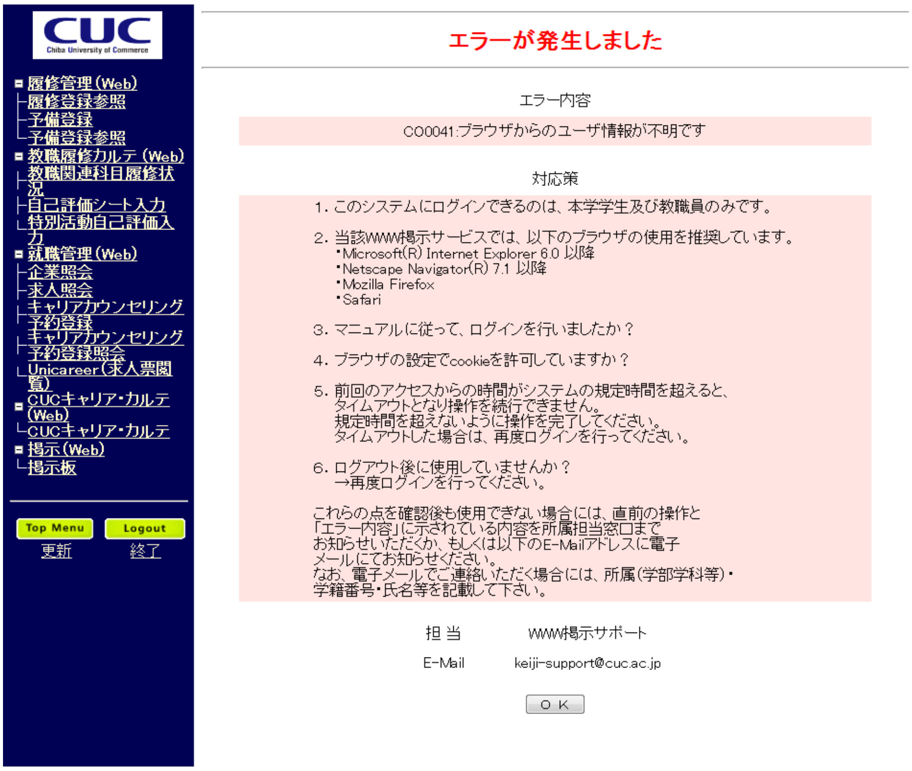

図 2-1-1-5: WWW 学生サービス利用時に長時間操作をしなかった場合自動的にログアウト処理をされる場合

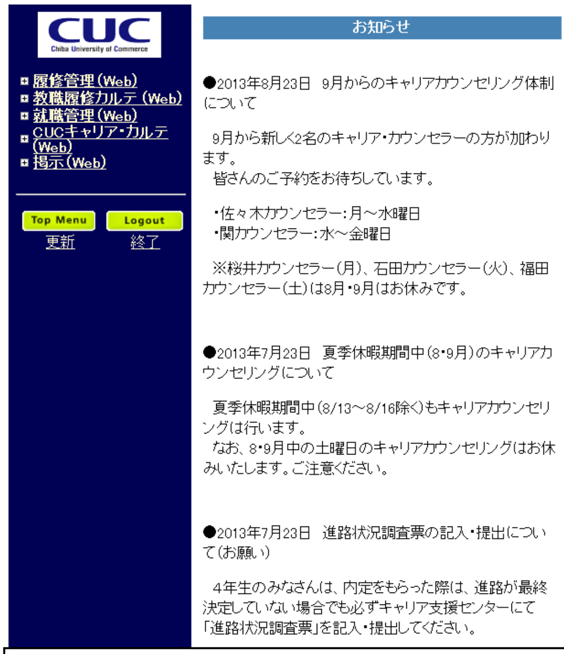

図 2-1-1-6: WWW 学生サービスにログインできた場合

2-1-2　CUCWebMail

学籍メールアドレスによるメールの着信・受信状況を確認できる。

学籍番号という本学の各学生に割り振られる 7 ケタの英数字からなる番号がある。その学籍番号に続いて@cuc.ac.jp を加えたアドレスが学籍メールアドレスとなっており、学内での連絡はもちろん。職員からの連絡やゼミ内でのメーリングリストなど自身の持っている携帯電話またはスマートフォンなどのメールアドレスを使うことなく新しいメールアドレスとして学内でのスムーズな連絡手段となっている。また、学生だけでなく教員や職員も同じドメインからなるメールアドレスを持っており、授業外での連絡や相談などが可能となっている。(図 2-1-2-1～図 2-1-2-3 参照)

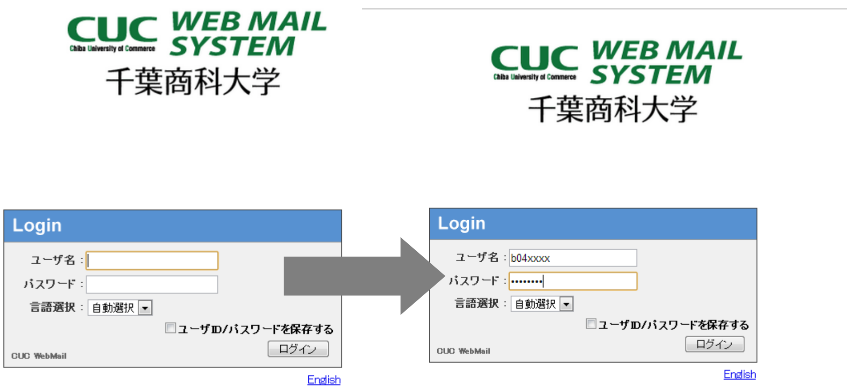

図 2-1-2-1: CUCWebMail (学籍メールアドレスによる) 画面のうちのひとつであり、ログインのための情報を入力するページ

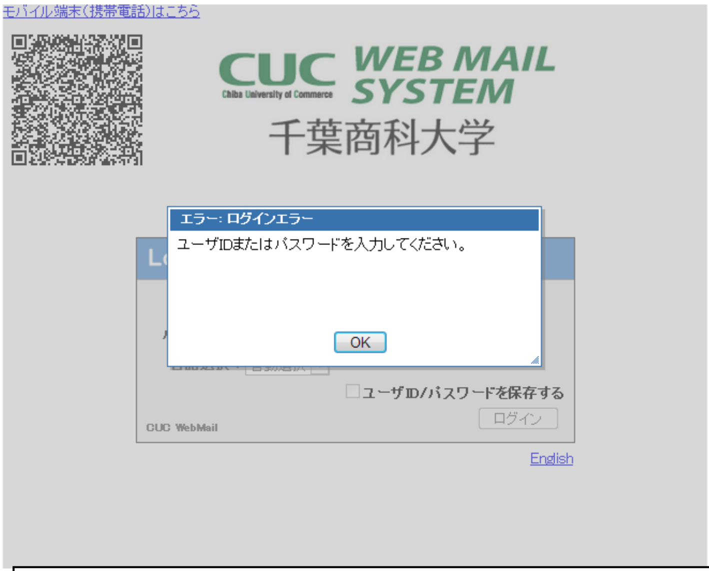

図 2-1-2-2: CUCWebMail にログイン情報を入力する際にログインに必要な情報が不足していた場合に表示される画面

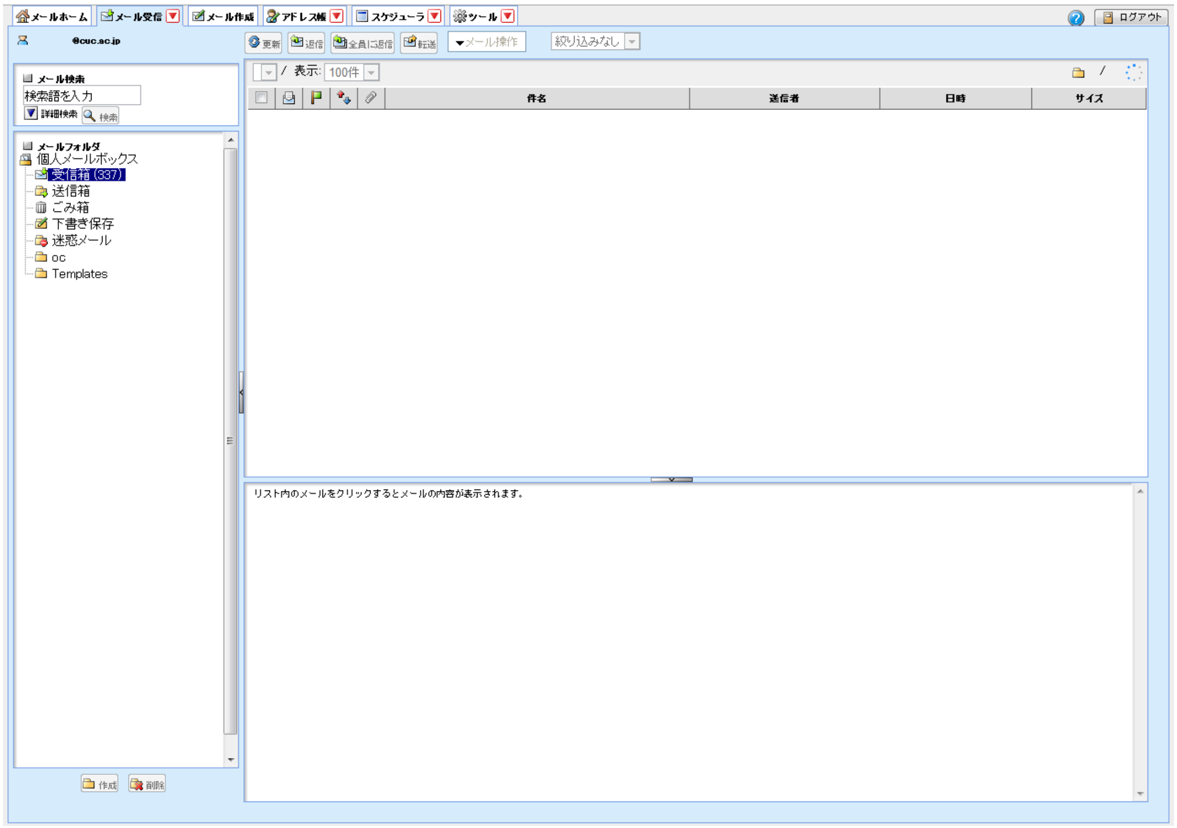

図 2-1-2-3: CUCWebMail にログインできた場合

2-1-3 CUC　Wellness System

千葉商科大学の政策情報学部には Wellness というシステムによって体育を受講している。インターネット上でログインをし、自分の都合のよい日程に合わせて毎週履修

登録をするシステムとなっており、都合が悪くなればキャンセルなどができる点から利用がしやすいと言える。また卒業のための単位数を見積もって、1 年間で集中して受講をし、4 年間を使ってゆっくり受講できるなど自由度が高い。またインターネットから履修登録をできるので急な予定の変更にも柔軟に対応できるといえる。（図 2-1-3-1 ～図 2-1-3-3 参照）

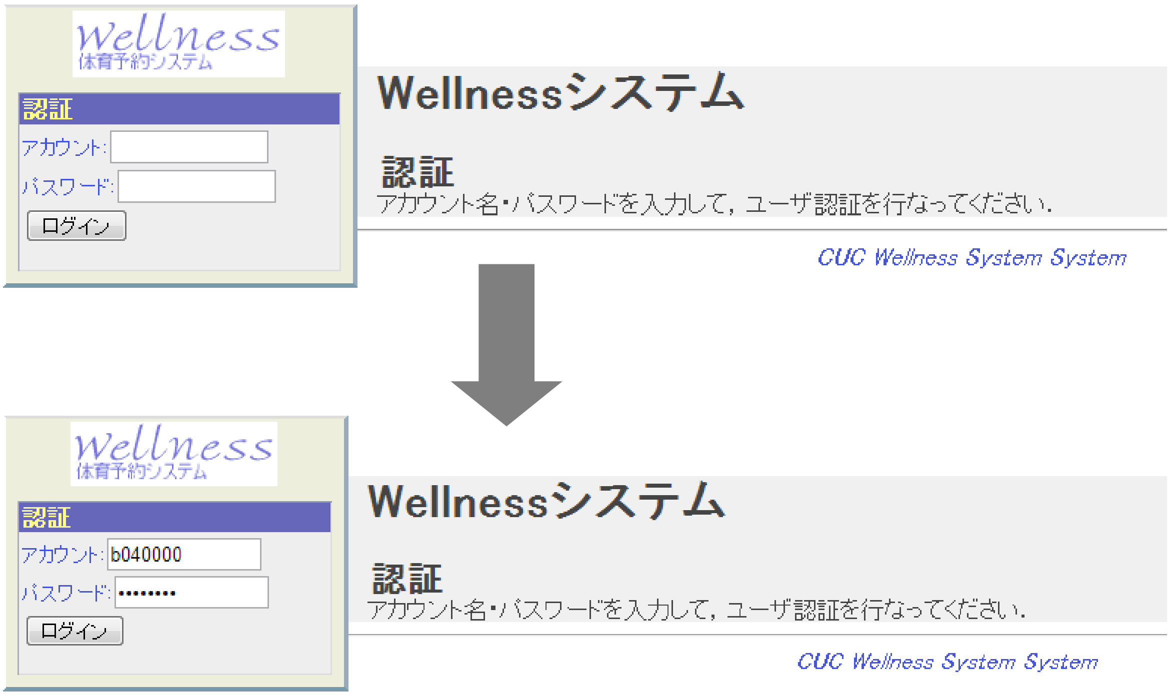

図 2-1-3-1: CUC Wellness System 画面のうちのひとつであり、ログインのための情報を入力するページ

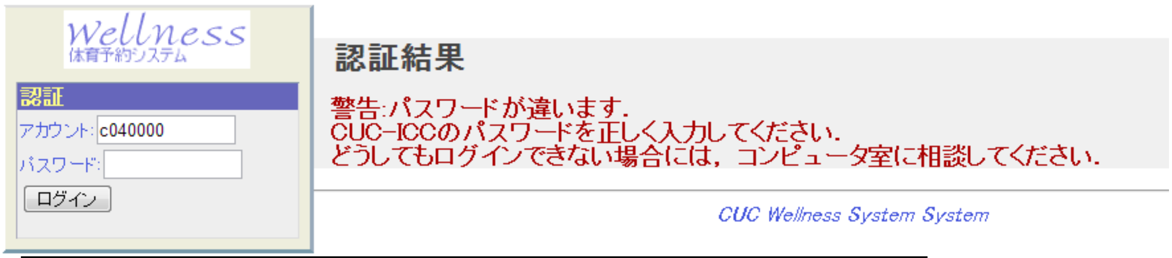

図 2-1-3-2: 認証情報を誤って入力してしまった場合

図 2-1-3-3: CUC Wellness System にログインできた場合

2-1-4 ログインの移管について

また大学における学生向け Web サービスはログインした状態での移管ができず、各 Web サービスをまたいでの利用をする場合はその時利用しているサービスとはまた別に新しいページを開き再度ログインを行う必要がある。たとえば「履修登録をするために WWW 学生サービス（履修登録サービス）のページを開いたとする。ログインをしてから履修を登録していく段階でウェルネスシステム（体育限定の履修登録サービス）との時間が重なっていないかどうかを確認する。ログインをへて自身で登録した体育の教科を確認する。メールを確認したくなった場合 CUC Webmail システムを新たに開く。ログイン状態を移管できないのでまたしても再度同じ内容の学籍 ID とパスワードを入力する。そしてメール・体育の履修を確認後 WWW 学生サービスのページに戻るとオートログアウトされており、WWW 学生サービスにもう一度ログインすることとなった。」という例が実際に著者におきており、何度も同じ認証情報を入力しなくてはいけないことになった。（図 2-1-4-1 参照）

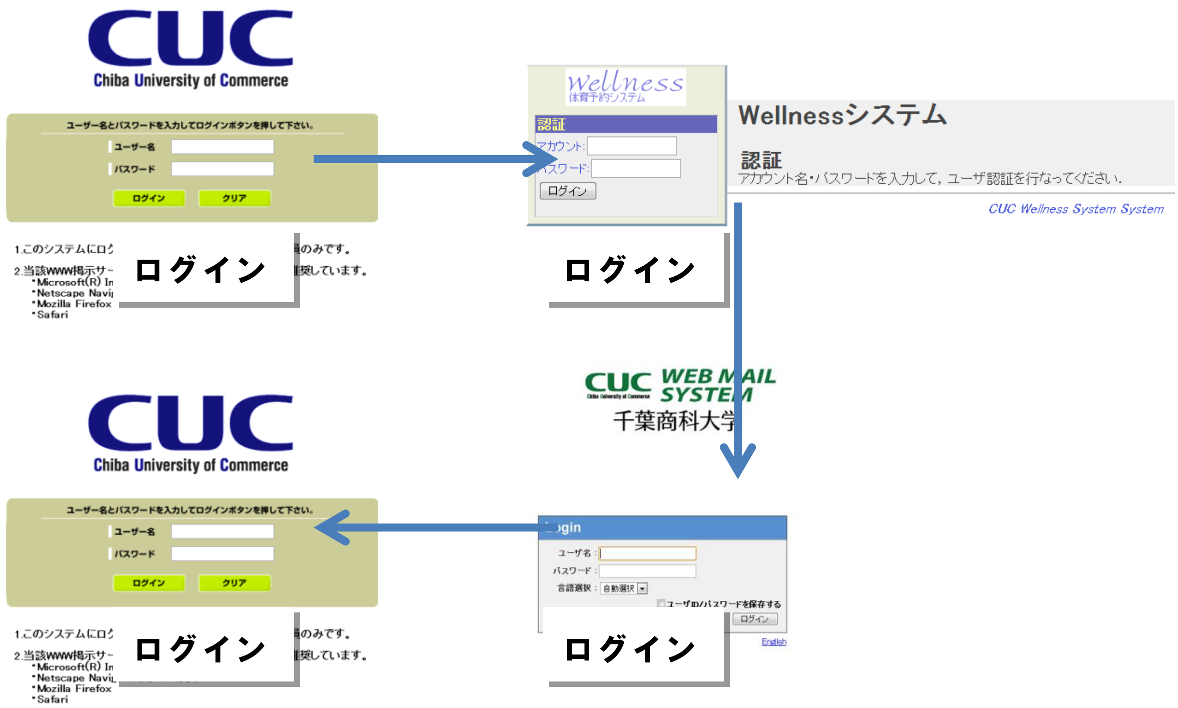

図 2-1-4-1: 各 Web サービスのログイン状況が移管できず、何度もログインを繰り返し、WWW 学生サービスが長時間操作をしなかったため自動的にログアウト処理されてしまった例

3.　解決方法

3-1　解決へのアプローチ

現在、学内に存在するすべての Web サービスに対して毎回ログインしなくてはいけない。この状況を解決するために一度入力した認証情報を一時的にサーバーに保存し、再度ログインする際にサーバーから認証情報を引き出すことのできるサービス、シングルサインオン（SSO）を導入することで各 Web サービスのログインの労力を軽減することができる。（図 3-1-1 参照）

［期待される効果］

・ログイン時の認証情報入力の労力を減らすことができる

・ワンステップ進んだ学内 Web サービスによってより快適に Web サービスを利用できる

・実質的な時間短縮につながる

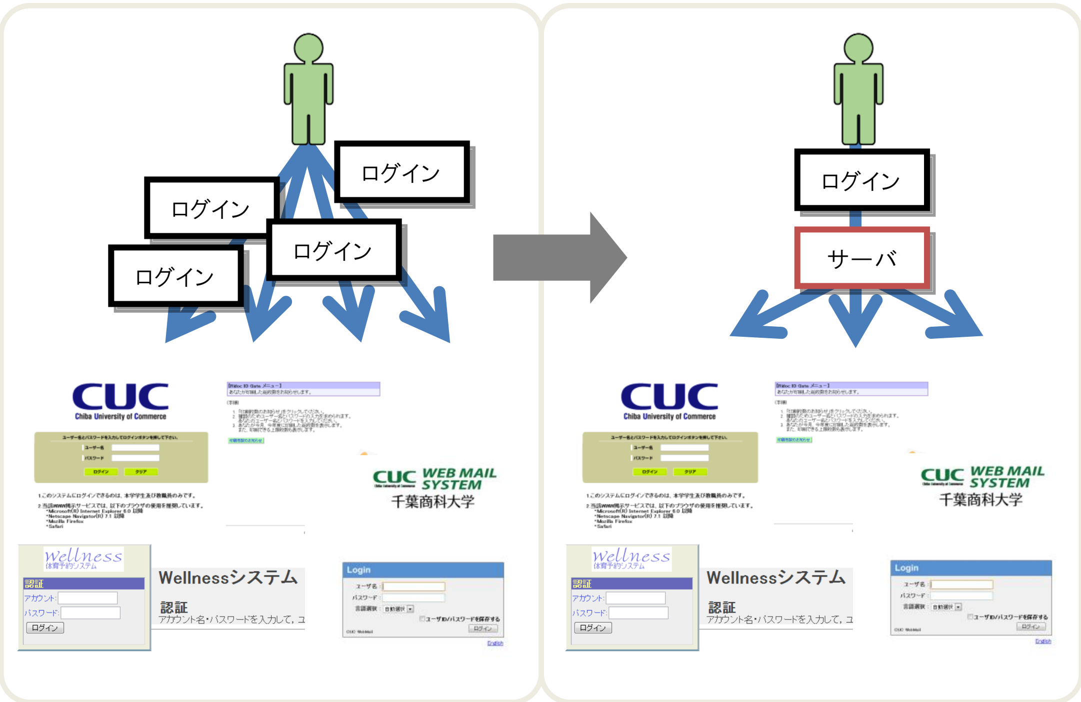

図 3-1-1: 従来の方法ではなくシングルサインオンになった場合、管理者がサーバーを設置するだけで利用者はただログインの労力が省けるだけで済む

3-2　シングルサインオン（SSO）とは

参考文献【2】､【8】

SSO（Single Sign-On）とは、一度で複数のシステムが利用可能になるログイン（およびそのような方式）を意味します。例えば、グループウェアにログインした後に、CRM（主に情報システムを用いて顧客の属性や接触履歴を記録・管理し、それぞれの顧客に応じたきめ細かい対応を行うことで長期的な良好な関係を築き、顧客満足度を向上させる取り組み。また、そのために利用される情報システム）にアクセスすると、通常はログイン画面が表示されますが、SSO の場合はログイン画面を経由せずに、そのままシステムを利用することができます。

一度 Web サービスにログインする際に ID やパスワードなどの本人を確認する認証情報を専用のサーバーに一時的に記憶させ、再度ログインする際に認証情報入力欄に自動的に情報を専用サーバーから出力するサービスのことである。

3-3　SSO の背景

参考文献【2】

また、ID とパスワードが増えるにつれて、システム管理者の作業負担も増加します。パスワードを忘れたユーザに対する再発行や、アカウントの使用状況の把握、利用停止などシステムごとに実施しなければなりません。このような状況で、パスワード忘れ対策のために導入されたパスワードリマインダ（パスワードを忘れてしまった場合に、あらかじめ利用者が設定しておいた特定の質問に答えることにより、本人確認ができたとみなされてパスワードが表示される機能です。第三者から推測されやすい質問と答えを設定すると、パスワードを割り出され、メールの内容を盗み見られたり、システム内のアカウントを悪用されたりします。）機能が悪用されたり、ユーザがパスワードを PC に付箋で貼り付けるようになったり、セキュリティ上の問題になるとも多々あります。

これらの課題を解決する方法として SSO が注目されるようになり、それを実現するための技術やソフトウェアが次々と生まれました。技術の標準化やソフトウェアの高機能化などにより、SSO のソリューションの範囲は年々拡大し、現在はインターネット

上のサービスとの連携や認証の強化、きめ細かなアクセス制御なども可能になっています。

一度のログイン操作さえ完了すれば、複数の Web アプリケーションにログイン操作することなくログインすることが可能である。インターネットやパーソナルコンピュータ、タブレット、スマートフォンの普及によって Web サービスを利用することが日常的になっている今日で認証のための ID やパスワードをサーバーで管理し、自動でログインを補助してくれるシングルサインオンへの期待は大きいといえる。

3-4　シングルサインオンの方式

参考文献【7】

SSO の構成

SSO を実現するシステムは、一般的にリバースプロキシ型とエージェント型に分類されます。この分類に従えば OpenSSO はエージェント型です。しかし一般的なエージェント型から受ける印象とは少し違いエージェントに相当するモジュールが policy agent として提供されているので(apache のモジュールや tomcat のフィルタ)、対応済みの Web サーバーやアプリケーションサーバであれば SSO 対象 Web アプリにエージェントのコードを組み込む必要はありません。policy agent をモジュールとして組み込んだ apache をリバースプロキシにすれば、リバースプロキシ型として OpenSSO を動かせます

認証のための方式は以下の 3 つになるが今回 OpenAM で行う方式はエージェント方式に当てはまる。一重にシングルサインオンと呼んでもその方式にはいくつか種類があり、以下の 3 つの方法がある。(図 3-4-1～図 3-4-3 参照)

参考文献【10】

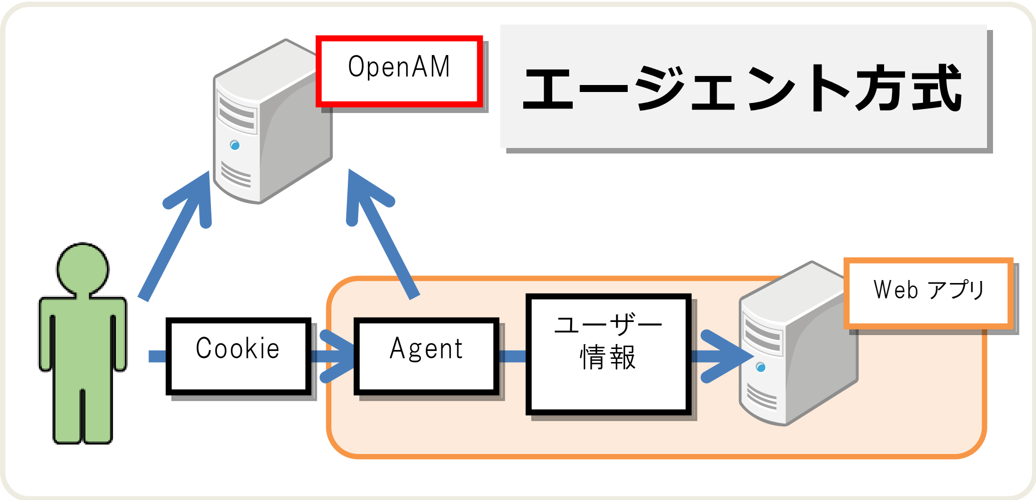

図 3-4-1: 認証のための方式の一つエージェント方式

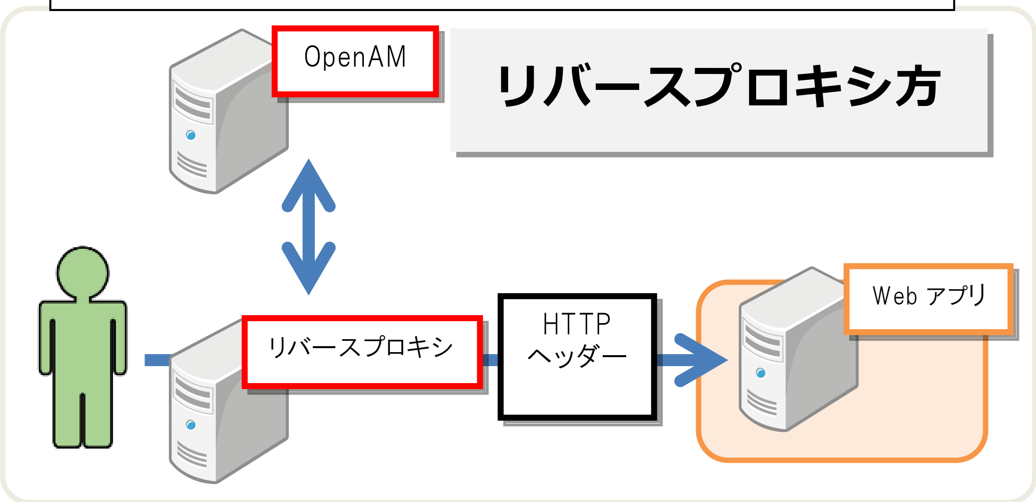

図 3-4-2: 認証のための方式の一つリバースプロキシ方式

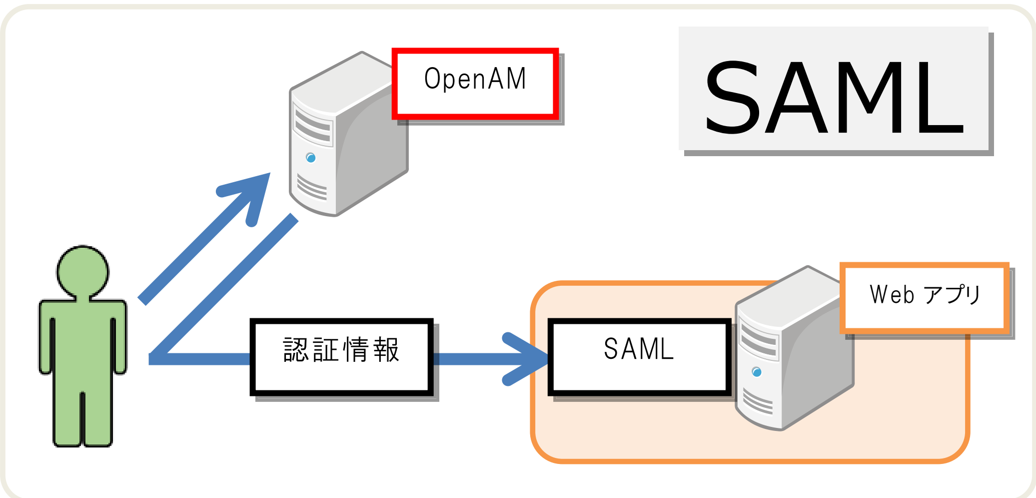

図 3-4-3: 認証のための方式の一つ SAML

4　関連研究

4-1　OpenAM

OpenAM とは ForgeRock 社の出しているソフトウェアであり Web アプリケーションにおけるシングルサインオンを実現するためのプラットフォームである。シングルサインオンによって一時的に認証情報を記憶することができる。現在（2013 年 11 月 1 日現在）無料で公開されているので誰でも利用できる。実装言語は Java(エージェントは C など)で実装されている。IC カード認証や生体認証などの豊富な認証方法があり、Google Apps などとも連携ができるので利便性が高いソフトウェアであるといえる。以下が OpenAM の基本情報となる。（表 4-1-1～表 4-1-2 参照）

参考文献【2】

表 4-1-1: OpenAM の基本情報

機能	SSO、認証、認可、アカウント連携
サポートする OS	マルチプラットフォーム
対応言語	マルチランゲージ（日本語にも対応）
実装言語	主に Java（エージェントと呼ばれるモジュールの一部は、C や C++で実装されている）
ライセンス	CDDL
Web サイト	http://www.forgerock.com/openam.html

表 4-1-2: OpenAM が提供する SSO の代表的な方式

SSO の方式	概要	対象アプリケーション
認証プロトコルを利用した方式	SAML や OAuth などの認証プロトコルを利用した方式。SSO 対象のアプリケーションが認証プロトコルに対応している必要がある。	Google Apps や Salesforce CRM などのクラウドサービス

エージェントを利用した方式	エージェント（OpenAM Policy Agent）をSSO対象アプリケーションのサーバ上にインストールし、リクエストを監視する方式。エージェントはログインリクエストをインターセプトし、OpenAMに認証要求する。	WEBサーバやJava EEコンテナ上で動作するアプリケーション
代理認証方式	通常はユーザーが行うログイン画面でIDとパスワードを入力する作業を、サーバーが代行する方式。既存のアプリケーションの改修をせずにSSOできる。リバースプロキシサーバに全アプリケーションのリクエストを集中させる方式や、全アプリケーションへのリンクを持つ共通ポータルサイトを経由させる方式などがある。前者の場合、性能面に考慮が必要。ForgeRock社では、OpenAMと連携し代理認証を実現するためのOSSであるOpenIGを提供している。	一般的なForm認証をするWEBアプリケーション

4-1-1 OSS Tech オープンソース・ソリューション・テクノロジ株式会社

OpenAMによるシングルサインオンの世界は個人の枠に収まらず、OpenAMを利用したビジネスも存在する。OSS Tech オープンソース・ソリューション・テクノロジ株式会社はOpenAMのシングルサインオンを使って製品としてパッケージ化をして提供を行っている会社である。

2011年7月25日にオープンソース・ソリューション・テクノロジ株式会社は北見工業大学でOpenAMを利用したシングルサインオンを導入した事例発表を行った。北見工業大学情報は以下の通りである。（図4-1-1-1～図4-1-1-2、表4-1-1-1参照）
参考文献【10】、【21】

図 4-1-1-1: OSS Tech オープンソース・ソリューション・テクノロジ株式会社の公式ページ

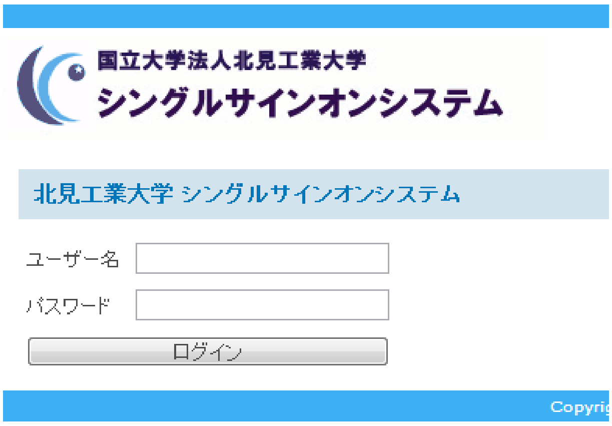

図 4-1-1-2: 北見工業大学 シングルサインオンシステム

表 4-1-1-1: 北見大学情報

北見工業大学	
生徒数	約 2600 名
職員数	約 260 名

実際に大学という大規模環境で OpenAM を実装している例としては生徒数、職員数合わせて約 3000 人という人数は十分といえる。北見大学における IT システムについて北見大学情報処理センター升井洋志准教授は以下のように述べた。

『10 数年前までは、大学の IT といえば計算機とメール、Web だけでした。もちろん、その前は大型計算機のアカウントのみ、 という時代もありましたが。そういう時代では、アカウントが各所に点在することも無く、ある意味「統制のとれた」状態だったと言えます。しかし、現在では、学生は教務関連、メール、演習室端末等、教職員はそれに加えて業務関連やグループウェア、と様々なシステムが存在しています。「紙とペン」で出来ることを「サーバーとクライアント」に移設するのが IT というのであれば、現在の大学の状況は十分に IT 化されていると言えるでしょう。しかし、各システム間でのユーザー情報やデータ・コンテンツが共有されないでいる状態では、いたずらにユーザーつまり学生および教職員の手間を増やすことにつながりかねません。

北見工業大学では、学内事務の効率化や業務のスリム化を目的に、さまざまなシステムや Web アプリケーションが導入されています。 人事、経理、教務など、業務ごとに利用されているそれらのシステムを結びつけ、セキュリティを高めた上で、よりシームレスに使いやすくするために、新しく投入されたのが OpenAM です。

『導入前の課題としては、本学でも、様々なシステムが独立に導入されその認証システムも個別であったために、ユーザーが覚えなくてはいけない ID・パスワードの組がサーバーの数だけ存在していました。ID・パスワードがバラバラなことも問題の一つでしたが、なによりも学内にどのようなシステムが存在し、そのアカウントがどういう範囲のユーザーに配布されているかを統括的に管理している組織が無いことが、導入に際しての最大の問題点でした。そこで、まずは各部署に対してサーバーの調査を行い、アカウントについて把握することから始めました。その結果をもとに、ユーザーにとって重要で、かつシングルサインオンが適用可能なシステムを選定し、システムの構築を進めました。』

OpenAM の利用のされ方として近年は､クラウドコンピューティングや SaaS のアプリケーションで複数の認証を統合する際においても、利用が増えています｡『まだまだ学内には今回のシングルサインに未対応のシステムがありますし、学認などの学外のシステム（電子ジャーナルやクラウドサービス等）への対応についてやるべきことは多数あります』

この OpenAM の導入を通してシステムごとに必要だった ID とパスワードの入力やシステムごとの ID、パスワードを記憶しておく必要がなく、現在のサーバーのアカウントを SSO のアカウントとすることで比較的スムーズに SSO に移行できていた。北見大学の学生や職員からも「いちいちパスワードを引っ張り出さなくてよくなった」と好評だったようだ。

4-2　既存の類似サービス

アプリケーションによってシングルサインオンを実現させている例として以下のサービスがある。

・Login Code（ログインコード）

LoginCode から Web ページを開くとあらかじめ登録しておいた ID とパスワードとログインページの URL と[Tab]キーで入力フォームまでたどり着く回数が即座に入力される。ベーシック認証によるログインに対応していない。ベーシック認証と知らずに登録してしまうとパスワード欄に入るべき認証情報が関係のない欄に記述され、「******」という伏字がなくなってしまう場合がある。(図 4-2-1 参照)

参考文献【5】

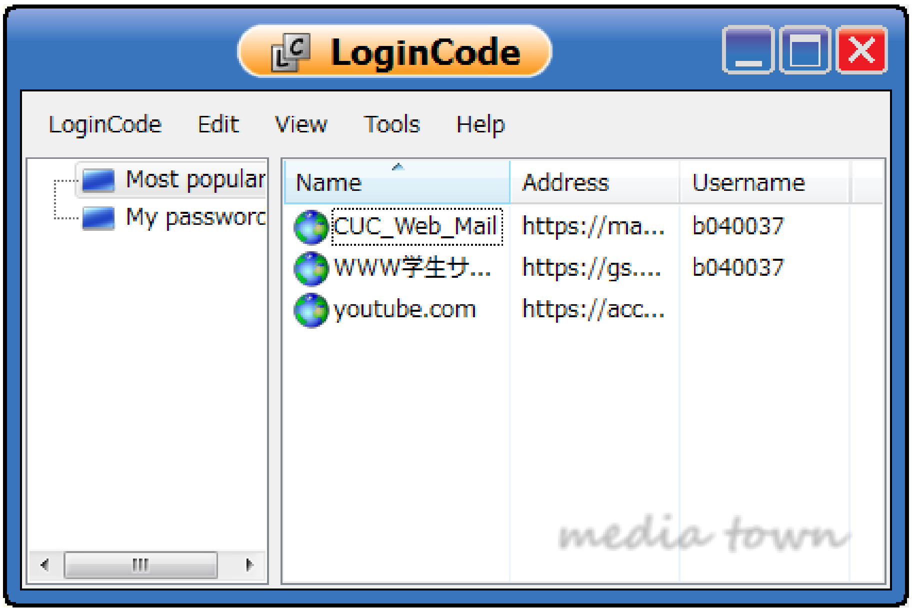

図 4-2-1:LoginCode に CUCWebMail、WWW 学生サービス、Youtube のログイン情報を記録させている図

・RoboForm（ロボフォーム）

ブラウザに常駐し、登録したページを開くと自動的に ID とパスワードを入力する。10 件までの登録をすることができるが、10 件を超えると有料の会員登録が必要となる。CUC の学内 Web サービスのみに視点を置いた場合は 10 件以内におさまるが、SNS などのその他 Web サービスのデータを併用したログインにおける労力を軽減することを考えると難しい。（図 4-2-2 参照）

参考文献【6】

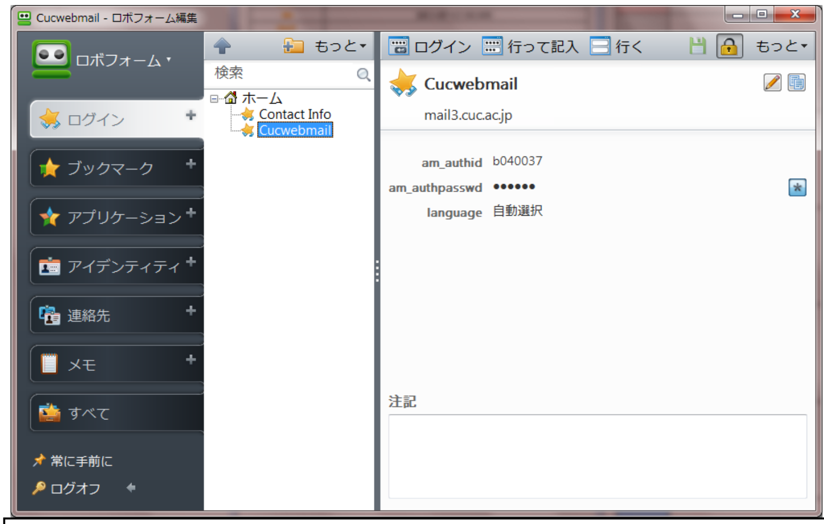

図 4-2-2:RoboForm に CUCWebMail の認証に利用する ID やパスワードを記憶させている図

4-3　シングルサインオンに伴う危険性

シングルサインオンはログインの労力が省け、効率的にも思えるがその分多くの危険をはらんでいる。主にあげられる危険な問題として以下の事例がある。

・認証情報を忘れやすくなる

ログインを繰り返し、認証情報の入力を何度も繰り返すことによって認証情報を入力することを体で覚える機会が減るので認証情報を忘れやすくなる可能性がある。シングルサインオンサーバーが急きょ使用不能な場面に対して認証情報を忘れてしまった場合認証情報を呼び出すためのサービスや作業が必要になり通常のログインに比べてより多くの労力がかかってしまう。

・セキュリティの問題

パスワードとは自分の頭の中だけで記憶し、誰にも知られずにいることがもっとも強固なパスワードとも言える。そのパスワードをメモ帳や付箋などに残す、誰かに教える、知られてしまうことでそのパスワードのセキュリティの硬さは一気に崩壊する。そんなパスワードを一時的とはいえどサーバーに保存されてしまうこと、自分以外のどこかに出てしまうことは大変危険であり、そのサーバーを狙ったハッキングによってパスワードが流出してしまう危険がある。管理者が何らかの形で認証情報を保存しておく際に管理責任の所在があいまいになってしまった場合セキュリティが疎かになってしまうと情報漏えいの危険性もある。

サービスを受けるために必要な情報を悪用しようとする危険な人々から守る必要がある。そのために必要な安全性（認証情報をサーバーに保存しておく際にデータが盗まれない工夫や盗まれても解読に時間のかかる複雑な暗号化など）が求められる一方でより快適なサービスを提供するための利便性も求められている。本節では安全でかつ利便性のあるシングルサインオンを目指す必要があるといえる

4-4　学術認証フェデレーション

学内でのWebサービスによって快適で利便性のある生活は本学に限った話ではない。この章ではでは本研究に似た目的を持った団体の紹介をする。(図 4-4-1 参照)

参考文献【9】

全国の大学等と NII が連携して、「学術認証フェデレーション（愛称：GakuNin)」の構築・運用を平成 21 年度から本格的に開始しました。

学術認証フェデレーションとは、学術 e-リソースを利用する大学，学術 e-リソースを提供する機関・出版社等から構成された連合体のことです。各機関はフェデレーションが定めた規程（ポリシー）を信頼しあうことで，相互に認証連携を実現することが可能となります。

認証連携を実現することができれば，学内でのシングルサインオン（一つの ID・パスワードであらゆるシステムが利用可能であること）を実現することが可能になるとともに，他大学や商用のサービスにおいても，1 つのパスワードを利用し，かつ ID・パスワードの再入力を行わずに利用できる環境を実現することができます。例えば，他大学の無線 LAN をいつも大学で使用している ID とパスワードで利用することができ，かつ自大学が契約している電子ジャーナルへシームレスにアクセスすることも可能となります。

技術的には、米国 Internet2 が開発した Shibboleth（注 1）を使用して実現していきます。Shibboleth は海外の多くの国で既に運用されていますが、日本語対応、運営組織の構築、規程の作成等、技術面・運用面で決めていくべきことが多数あります。特に、個人情報保護に関する法律は日本の法律に適合させる必要がある等、日本独自で開発すべきものが多数あります。

この実現のため、平成 20 年度に実証実験を行い、平成 21 年度から試行運用を実施し、この成果をもとにして平成 22 年度より本格運用を開始しました。学認では、本格運用への移行を機に、利用者によるフェデレーションへと発展させることを念頭に、学認タスクフォースを発足いたしました。以前よりフェデレーションの活動にご理解、ご協力頂いている機関の基盤センターあるいは図書館の方々に、利用者の代表の立場でご参加頂き、学認の活動に関する様々な検討を行っています。

図 4-4-1：学術認証フェデレーションの公式ページ

シングルサインオンを利用し学内の Web サービスをよりよく快適にしようとする団体であり OpenAM やアプリケーションでのシングルサインオンではなく Shibboleth という米国に 2000 年に発足した SAML 認証を標準仕様とした認証のためのオープンソースを利用している。平成 20 年にダミーデータによる実証の実験検証をおこない、平成 21 年には実データでの運用を開始し試験運用を行った。平成 22 年本格運用として実際に実データを利用し、運用を行っている。

5　システムの調査

5-1　OpenAM の試験導入

実際に OpenAM を使ったサーバーを設置し、シングルサインオンサービスを個人が提供できるのか、またそのサーバーを学内の環境でも利用できるのかを調査する。（図 5-1-1～図 5-1-2 参照）

参考文献【3】、【10】、【11】、【13】、

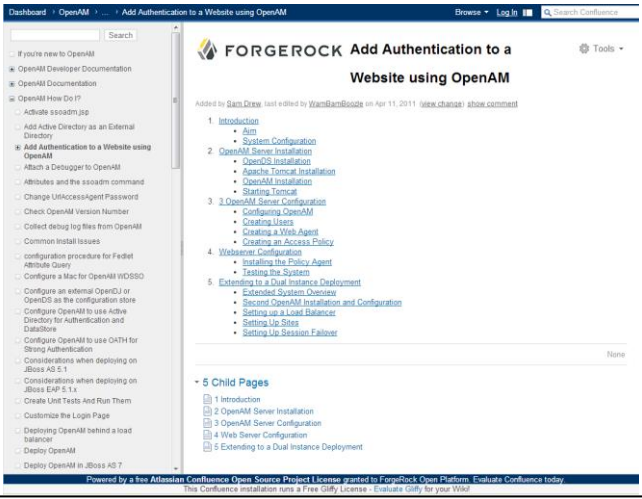

図 5-1-1: ForgeRock 社公式の OpenAM の導入方法

図 5-1-2:SIOS”OSS よろず”ブログ出張所

オープンソースソフトウェアを多く紹介するブログ

5-1-1 OpenAM Server のインストール

Ubuntu と Tomcat による OpenAM 導入方法

必要なもの

・OpenAM（openam バージョン 10）

・Tomcat

（図 5-1-1-1 参照）

参考文献【3】、【11】、【13】。

1)コマンドプロンプトの起動

2)http://forgerock.org/opendj.html から OpenDJ をダウンロードする

3)unzip　OpenDJ-2.5.0-Xpress1.zip で解凍

4)$ /path/to/OpenDJ-2.5.0-Xpress1/setup --cli 管理者権限で setup を起動インストールを開始する

5)

What would you like to use as the initial root user DN for the Directory Server? [cn=Directory Manager]:

cn=Directory Manage

ディレクトリマネージャー（サーバーで認証情報を補完する人）名の設定

6)

Please provide the password to use for the initial root user:

自分の ROOT ユーザーのためのパスワード

Please re-enter the password for confirmation:

確認のためのパスワード入力をする

7)

Provide the fully-qualified directory server host name that will be used when generating self-signed certificates for LDAP SSL/StartTLS, the administration connector, and replication [opendj.example.com]:

fujiki.hage.org

URL となる文字を打ち込む

8)

On which port would you like the Directory Server to accept connections from LDAP clients? [389]:

1389

ディレクトリサーバが LDAP クライアントからの（今回の SSO サーバーの利用者に対して）接続を受け入れるポート番号を指定する

9)

On which port would you like the Administration Connector to accept connections? [4444]:

4444

OpenDJ の管理者コントロールパネルへの接続をする際に受け付けるポート番号を指定する

10)

Do you want to create base DNs in the server? (yes / no) [yes]:

yes

ディレクトリ検索を行うために、ベース DN を作成するかどうかを選択する

11)

Provide the base DN for the directory data:

dc=fujiki.hage,dc=org

今回は、ディレクトリ階層構造でトップレベルのドメインが example.com となるため、ベース DN には dc=fujiki.hage, dc=org を指定する。ベースとなるエントリを作成するので、「ベースエントリのみを作成する (dc=fujiki.hage, dc=org)」を選択する

12)

Options for populating the database:

1) Only create the base entry

2) Leave the database empty

3) Import data from an LDIF file

4) Load automatically-generated sample data

Enter choice [1]:

1

データーベースを移入させるためのオプション

ベースエントリのみ作成するため、1 番を選択する

13)

Do you want to enable SSL? (yes / no) [no]:

no

SSL を使用しますか？

SSL は使用しないので、デフォルトの no を指定する

14)

Do you want to enable Start TLS? (yes / no) [no]:

no

TLS を使用しますか？

TLS は使用しないので、デフォルトの no を指定する

15)

Do you want to start the server when the configuration is completed? (yes / no) [yes]:

yes

設定完了後 OpenDJ サーバーを起動するため、yes を指定する。

```
Setup Summary
=============
LDAP Listener Port:            1389
Administration Connector Port: 4444
LDAP Secure Access:            disabled
Root User DN:                  cn=Directory Manager
Directory Data:                Create New Base DN dc=fujiki.hage,dc=org.

Start Server when the configuration is completed
```

16)OpenDJが起動する

17)

```
What would you like to do?

    1)  Set up the server with the parameters above
    2)  Provide the setup parameters again
    3)  Print equivalent non-interactive command-line
    4)  Cancel and exit

Enter choice [1]:
1
```

今までの設定でサーバーを起動する

18) ps -ef | grep opends　このコマンドでOpenDJの起動を確認できます。

19)サーバー上でjavaを動かすためのソフト「Tom cat」のインストール

http://tomcat.apache.org/download-70.cgi からTom catをダウンロードする

20)unzip apache-tomcat-7.0.30.zipで解凍

```
21) cd apache-tomcat-7.0.30/bin
    chmod +x *.sh

22)OpenAMをインストール
http://forgerock.org/openam.html

23) cp openam_10.0.0.war  apache-tomcat-3.0.30/webapps/openam.war

24)ApathTomcatの起動
export JAVA_OPTS="-Xmx1024m -XX:MaxPermSize=256m"

25) ./apache-tomcat-7.0.30/bin/startup.sh

26)
Using CATALINA_BASE:   /opt/apache-tomcat-7.0.30
Using CATALINA_HOME:   /opt/apache-tomcat-7.0.30
Using CATALINA_TMPDIR: /opt/apache-tomcat-7.0.30/temp
Using JRE_HOME:        /usr
Using
CLASSPATH:        /opt/apache-tomcat-7.0.30/bin/bootstrap.jar:/opt/apache-tom
cat-7.0.30/bin/tomcat-juli.jar

27)
ブラウザからhttp://openam.fujiki.hage.org:8080/openamにアクセス
```

図 5-1-1-1:インストールが成功した場合ブラウザにて上記の画面が出る

5-1-2 OpenAM Server の設定

OpenAM の初期インストール時にデフォルトで設定されている自動設定モードで設定をしてオリジナルのサーバーURL を使用していた場合、Cookie ドメインとの整合性が取れず設定画面が進行しなかったためカスタム設定を推奨する。以下はカスタム設定の設定方法である。(図 5-1-2-1～図 5-1-2-9 参照)

1）デフォルトユーザーのパスワードを設定する

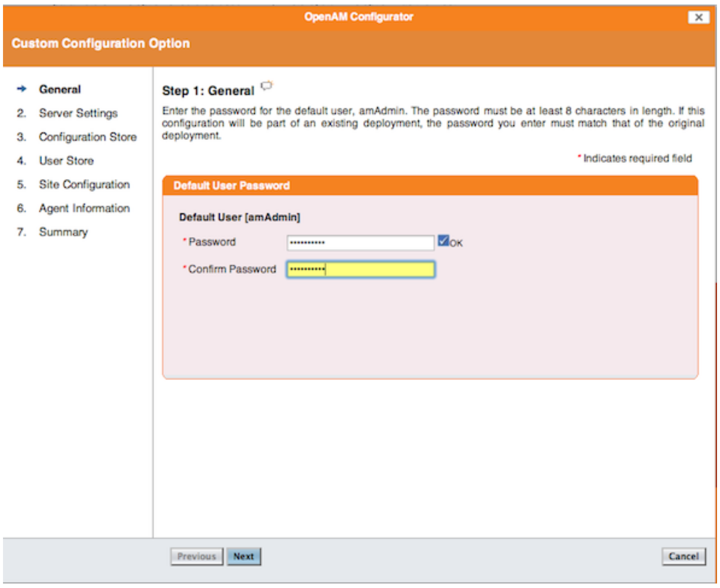

図 5-1-2-1:OpenAM Server パスワード設定画面

2） 使用言語（プラットフォームローカル）の欄に日本語であれば ja_JP と記入

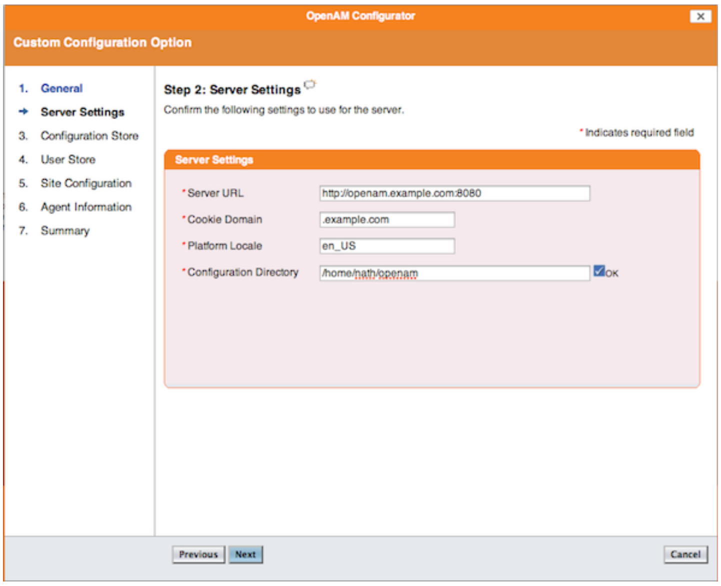

図 5-1-2-2:OpenAM Server 設定画面

3)　ポートは AdminPort を先ほど設定した 4444 に

RootSuffix : dc=fujiki, dc=hage, dc=org とする

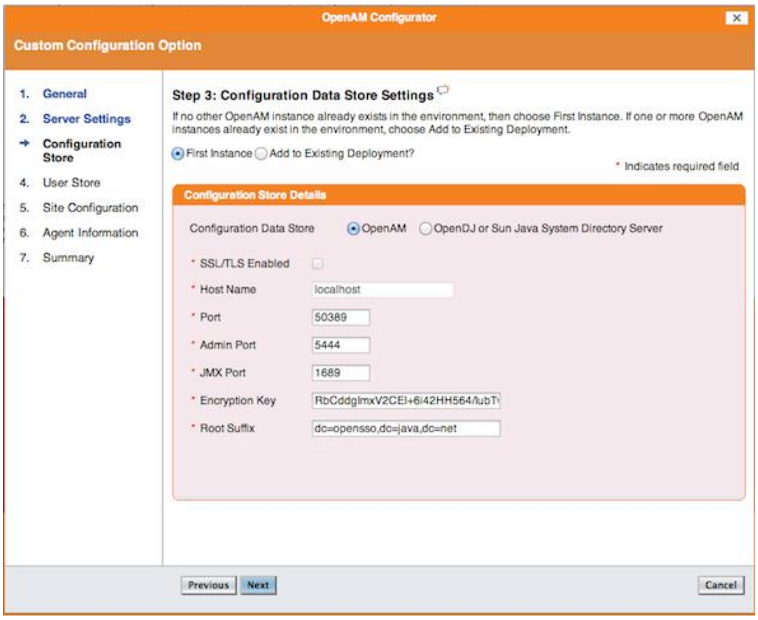

図 5-1-2-3:OpenAM Server 設定画面

4)

Directory Name : fujiki.hage.org

Port : 1389

Root Suffix : dc=fujiki, dc=hage, dc=org

Login ID : cn=Directory Manager

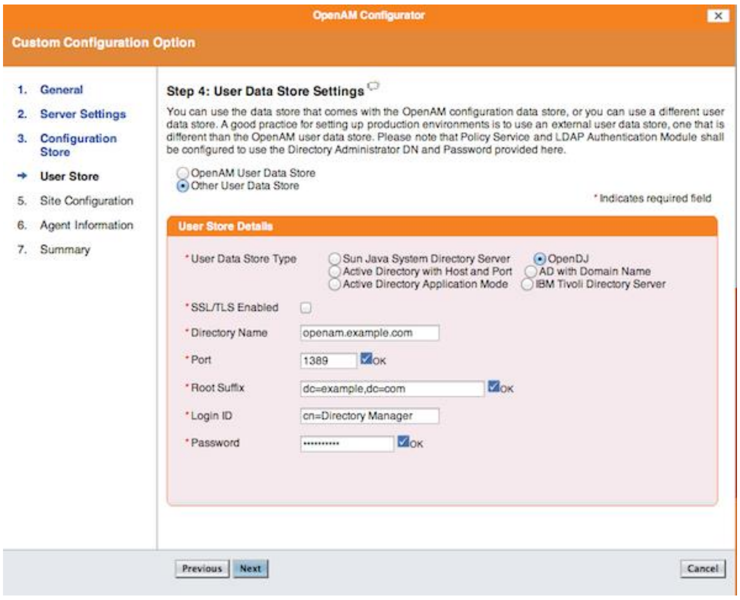

図 5-1-2-4:OpenAM Server 設定画面

5)サイトの構成について初期設定のまま

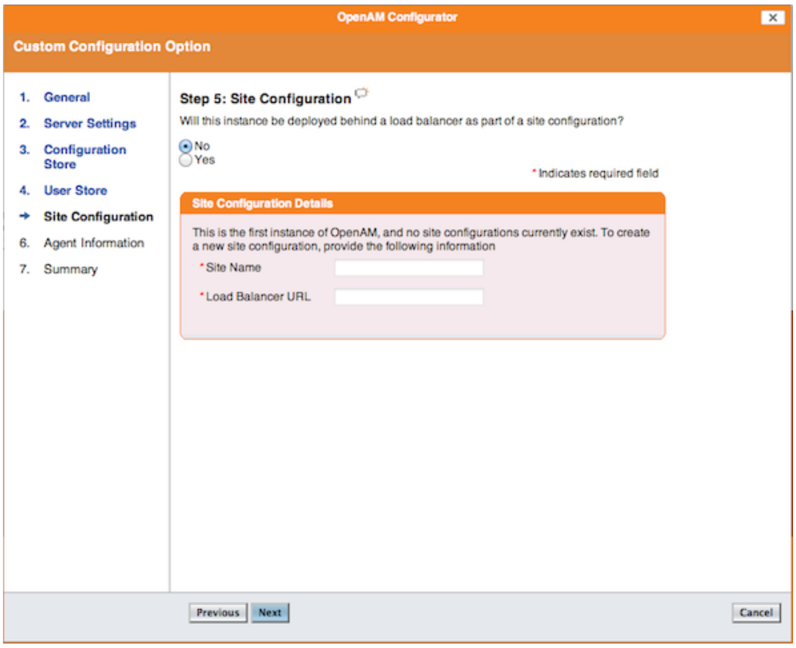

図 5-1-2-5:OpenAM Server 設定画面

6) デフォルトポリシーエージェントのパスワードを設定する

この章の最初に設定したパスワードではないものを設定しなくてはいけない。

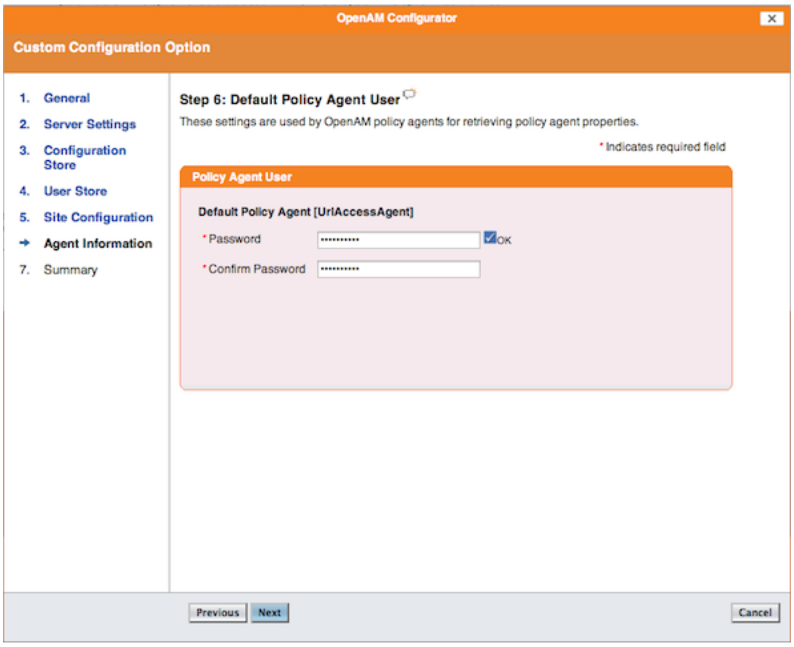

図 5-1-2-6:OpenAM Server 設定画面

7)以下の内容で構成の作成が正しければ CreateConfiguration を押す

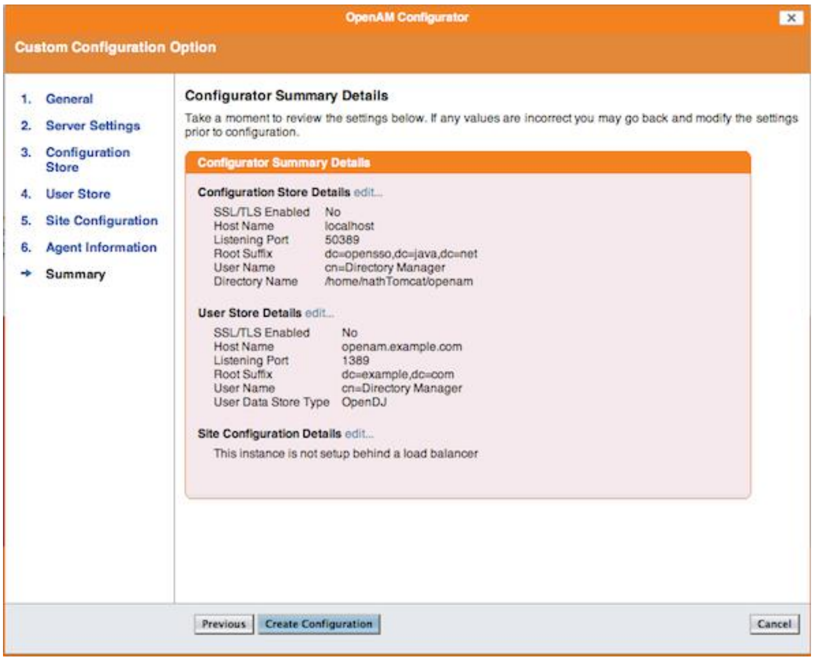

図 5-1-2-7:OpenAM Server 設定画面

8)サーバーの設定が完了するまで待つ

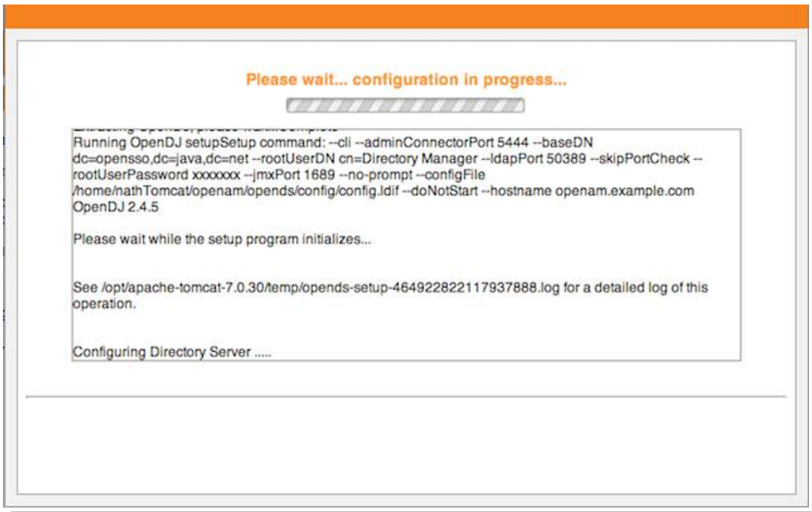

図 5-1-2-8:OpenAM Server 設定画面

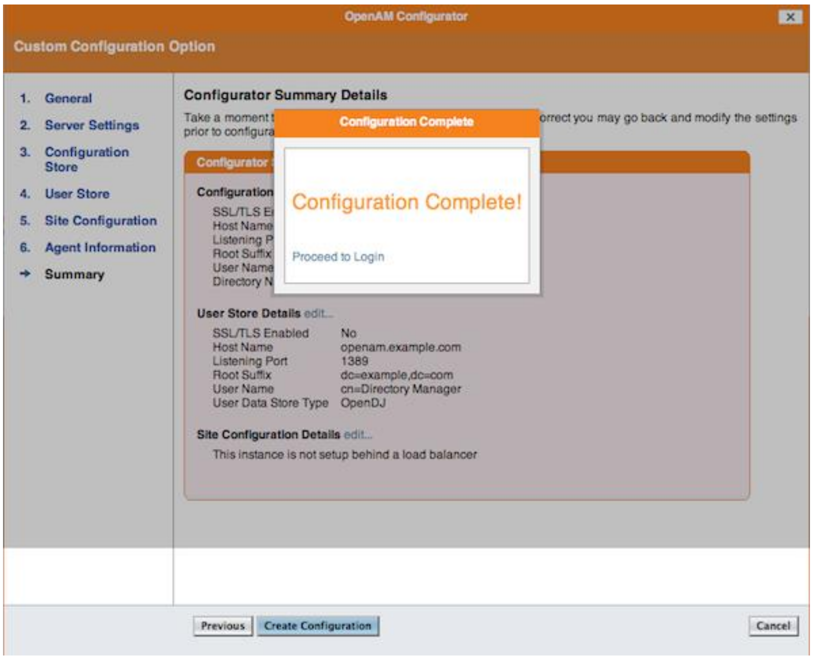

図 5-1-2-9:OpenAM Server 設定画面

5-1-3　ユーザーの設定

OpenAM を利用するユーザーを登録・設定する（図 5-1-3-1～図 5-1-3-5 参照）

1）

ID: amAdmin

パスワード:*****

図 5-1-3-1:OpenAM 設定画面

2) ログイン後このようなホーム画面になる

図 5-1-3-2:OpenAM 設定画面

3) アクセス制御タブ→（最上位のレルム）

図 5-1-3-3:OpenAM 設定画面

4) 対象タブ→新規

ここでユーザーの登録を行う

図 5-1-3-4:OpenAM 設定画面

5) ユーザーの情報を登録していく

表 5-1-3-1:OpenAM Server 設定画面

ID	user001	user002	user003	user004
氏	User	User	User	User
名	One	Two	Three	Four
氏名	User One	User Two	User Three	UserFour
パスワード	firstuser	seconduser	thirduser	fouthuser
パスワードの再入力	firstuser	seconduser	thirduser	fouthuser
ステータス	アクティブ	アクティブ	アクティブ	アクティブ

図 5-1-3-5:OpenAM 設定画面

5-1-4　Web エージェントの作成

（図 5-1-4-1～図 5-1-4-7 参照）

1) アクセス制御→（最上位のレルム）

図 5-1-4-1:OpenAM 設定画面

2) エージェント→Web→新規

図 5-1-4-2:OpenAM 設定画面

3)以下の内容を記入していく

名前：webagent

パスワード：cangetinwa

設定：Centralized

サーバーURL：http://fujiki.hage.org:8080/openam

エージェント URL：http:// fujiki.hage.org:80

図 5-1-4-3:OpenAM 設定画面

4)アクセスポリシーの作成

アクセス制御→(最上位のレルム)→ポリシータブ→新規ポリシー

名前 ： URLPolicy

説明： http://fujiki.hage.org.com/

アクティブ:有効

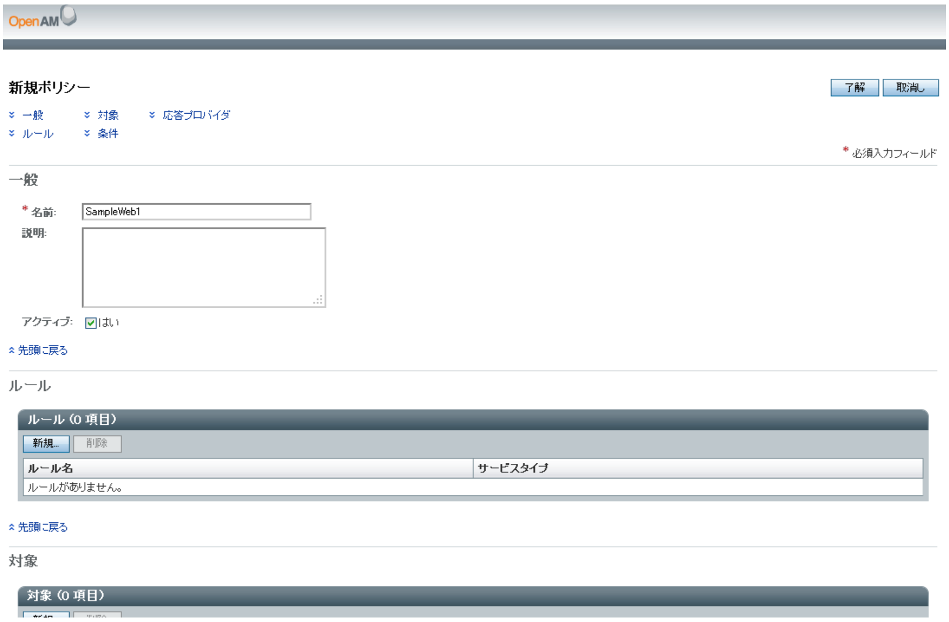

図 5-1-4-4:OpenAM 設定画面

5)ルールの設定

サービスタイプ ： URL ポリシーエージェント(リソース名あり)

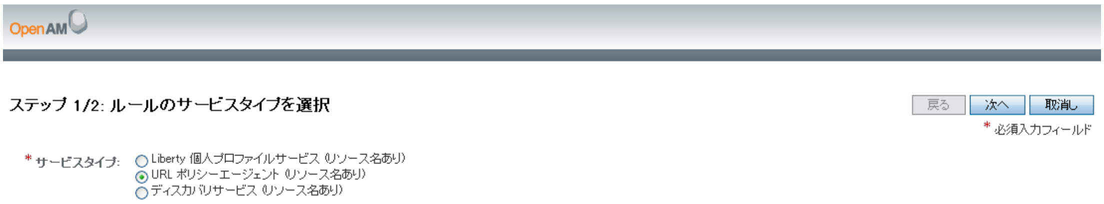

図 5-1-4-5:OpenAM 設定画面

6)

名前：Sample1

リソース名：http://fujiki.hage.org:8080/SampleWeb1/Sample1

アクション：GET と POST を許可

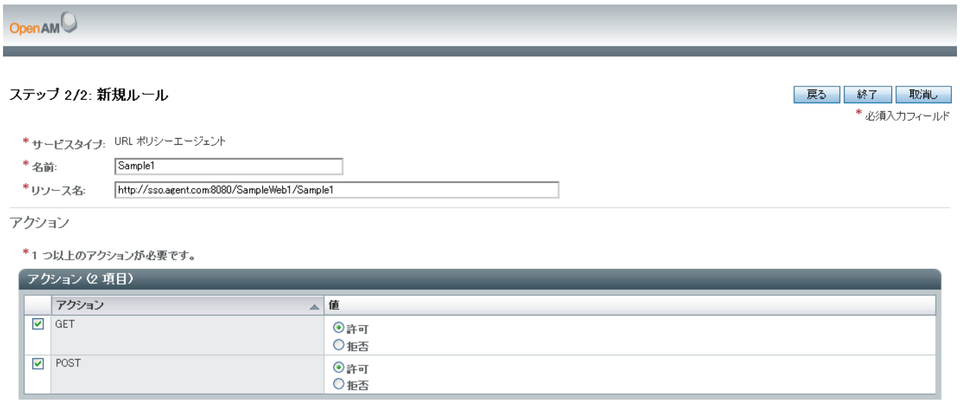

図 5-1-4-6:OpenAM 設定画面

7)user001,user002 を許可するルールを追加する

図 5-1-4-7:OpenAM 設定画面

5-1-5　Web サーバーの設定

1)ポリシーエージェントのインストール

http://forgerock.org/openam.html から apache_v22_Linux_64_agent_3.zip をダウンロード

2) unzip apache_v22_Linux_64_agent_3.zip

3)一度、管理者権限で apache2 を止める

sudo　/etc/init.d/apache2 stop

4) sudo vi /var/tmp/passwd

```
5) locate httpd.conf

6) cd web_agents/apache22_agent/bin

7)エージェントアドミンのインストール
sudo ./agentadmin --install

  Enter the Apache Server Config Directory Path [/opt/apache22/conf]:

  [Directory containing httpd.conf]

  Enter the URL where the OpenSSO server is running.
  Please include the deployment URI also as shown below:
  (http://opensso.sample.com:58080/opensso)
   >http://fujiki.hage.org:8080/openam

  Enter the Agent URL as shown below: (http://agent1.sample.com:1234)
  >http://fujiki.hage.org:80

   Enter the Agent profile name
   >webagent

  Enter the path to a file that contains the password to be used for identifying
the Agent.
  >/var/tmp/passwd

  8) 管理者権限で止めた apach2 を動かす
  sudo /etc/init.d/apache2 start
```

5-1-5　Winows と glassfish による OpenAM 導入方法

必要なもの

・OpenAM（openam バージョン 10）

・GlassFish（glassfish-installer-v2.1.1-b31g-windows）

・JDK(jdk-6u45-windows-x64)

あらかじめ Java のバージョンを 6 にしておくこと

（図 5-1-5-1～図 5-1-5-9 参照）

参考文献【14】、【15】、【16】

1)Glassfish を任意のフォルダに配置。今回は C:¥gf にファイルを置いた。

2)コマンドプロンプトを起動し C:¥gf に移動後に以下のコマンドを実行

```
C:¥gf>java -Xmx256m -jar glassfish-installer-v2.1.1-b31g-windows.jar
```

3)起動する License を下までスクロールし、Accept を選択

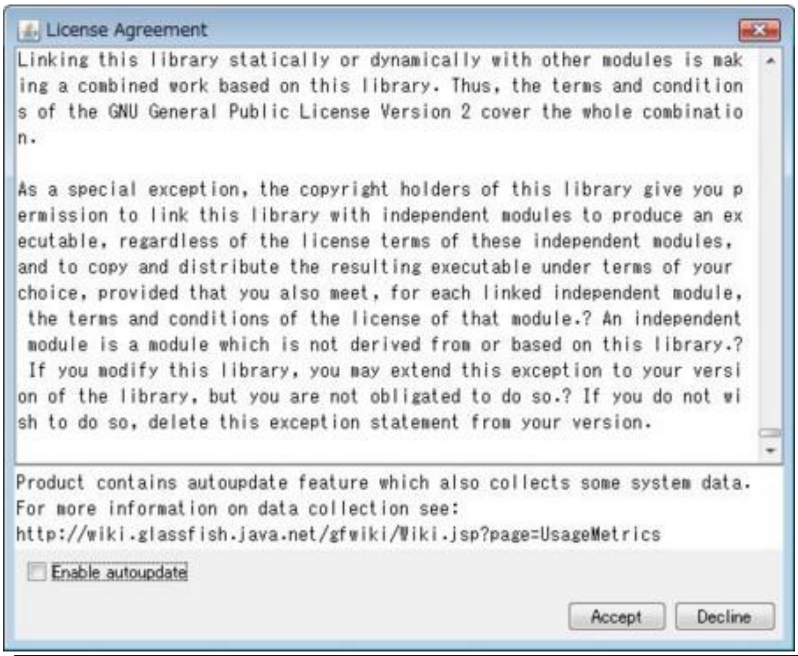

図 5-1-5-1:GlassFish のインストール

4)自動的にファイルの解凍が始まる。Installation compiete と出れば終了

5)Java の環境変数を設定する

スタート→コンピュータを右クリック→プロパティ→システムの詳細設定

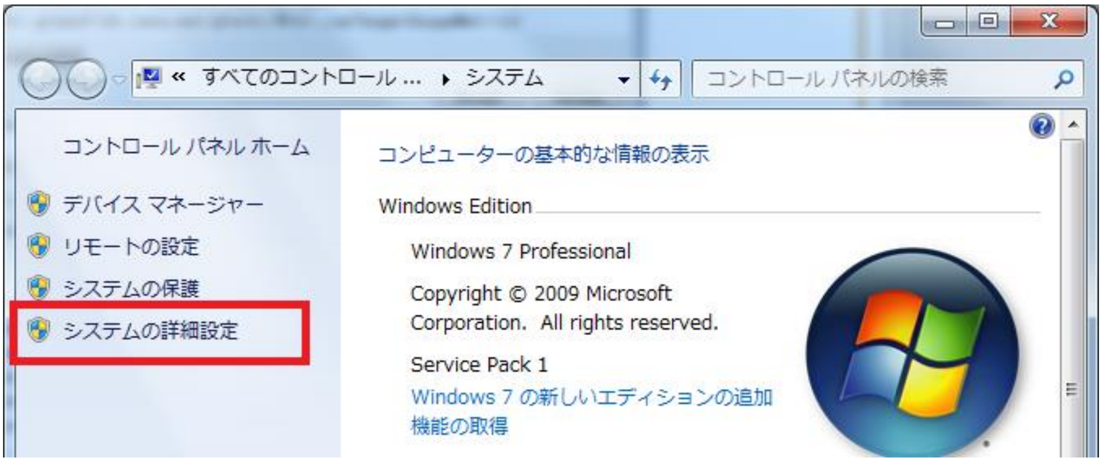

図 5-1-5-2:環境変数の設定

6)詳細設定タブ→環境変数

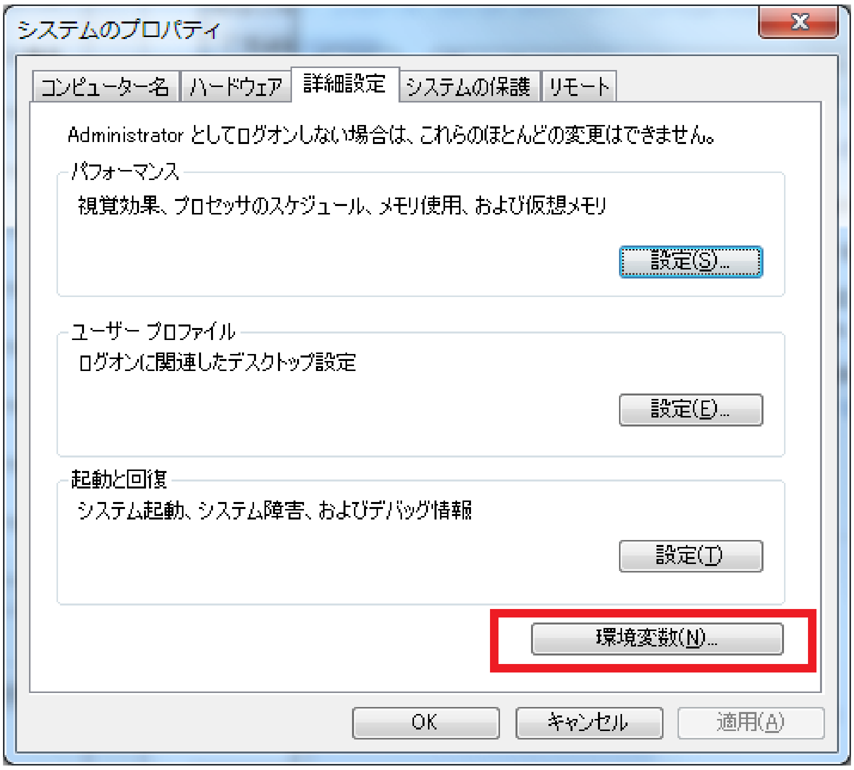

図 5-1-5-3:環境変数の設定

7)システム環境変数の Path を選択し編集

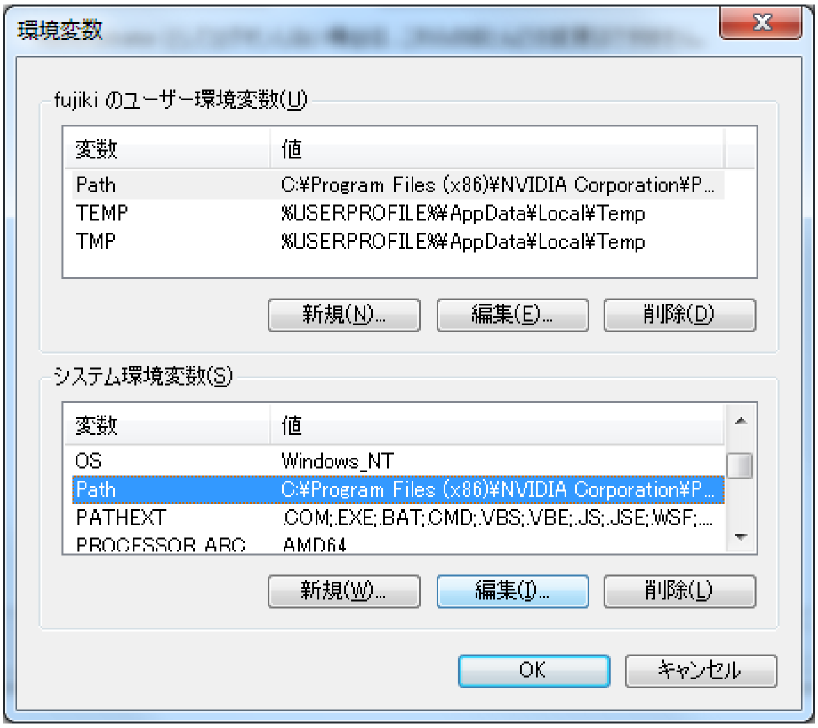

図 5-1-5-4:環境変数の設定

8) 「変数値」の欄にあらかじめ入力がされているのが Path の値である。最初から入力されている値を消さないように末尾に C:¥Program Files¥Java¥jdk1.6.0_45¥bin を追加(JDK をインストールしたディレクトリに合わせて設定する値は変更)する。

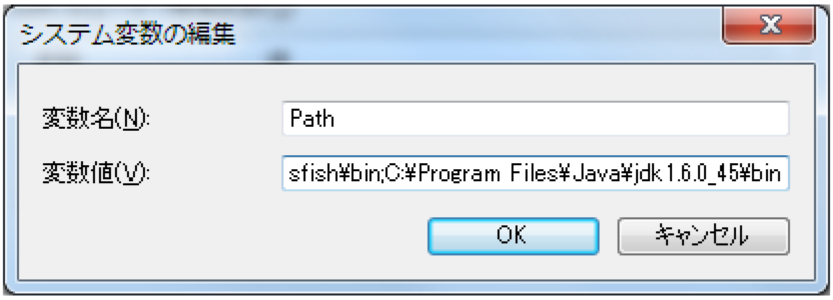

図 5-1-5-5:環境変数の設定

9)システム環境変数から新規

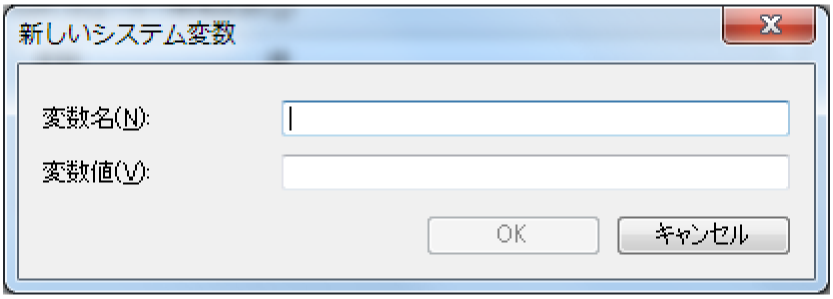

図 5-1-5-6:環境変数の設定

10)

変数名に JAVA_HOME

変数値に JDK をインストールしたディレクトリに書く

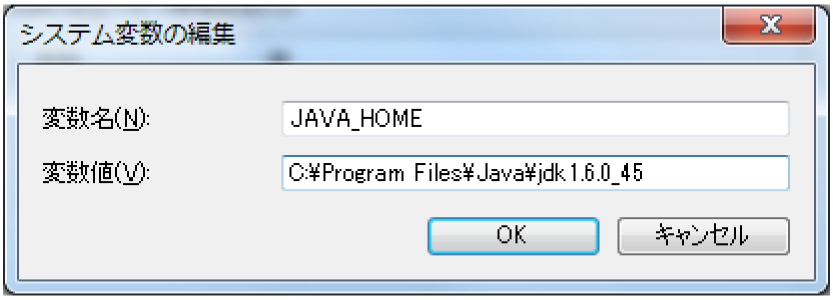

図 5-1-5-7:環境変数の設定

11)コマンドプロンプトを起動しなおし、

Set JAVA_HOME で JAVA_HOME の設定の確認ができる

12)ant ファイルの実行

C:¥gf>echo %JAVA_HOME%

C:¥jdk1.6.0_45

C:¥gf>cd glassfish

C:¥gf¥glassfish>lib¥ant¥bin¥ant -f setup.xml

13)ant 実行結果が BUILDSUCCESSFUL となればインストール完了

14)Glassfish のドメインを作成する

```
C:¥gf¥glassfish¥bin>asadmin  create-domain  --adminport 24848 --instancep
ort 28080  --savemasterpassword=true --savelogin=true openiddomain
```

15)質問に答えていく

```
Please enter the admin user name>admin
Please enter the admin password>
Please enter the admin password again>
Please enter the master password [Enter to accept the default]:>
Please enter the master password again [Enter to accept the default]:>
```

16) 成功すると以下のようなメッセージが出る

```
Using port 24848 for Admin.
Using port 28080 for HTTP Instance.
Default port 7676 for JMS is in use. Using 58525
Default port 3700 for IIOP is in use. Using 58526
Default port 8181 for HTTP_SSL is in use. Using 58527
Using default port 3820 for IIOP_SSL.
Using default port 3920 for IIOP_MUTUALAUTH.
Default port 8686 for JMX_ADMIN is in use. Using 58528
Domain being created with profile:developer, as specified by variable
AS_ADMIN_P
ROFILE in configuration file.
------ Using Profile [developer] to create the domain ------
XML processing for profile: Base document
[C:¥glassfish¥glassfish¥lib¥install¥te
mplates¥default-domain.xml.template]. Profile name [developer]. Processing
prope
rty [domain.xml.style-sheets].

The file in given locale [ja_JP] at:
```

```
[C:¥glassfish¥glassfish¥lib¥install¥templat
  es¥locales¥ja_JP¥index.html] could not be found. Using default (en_US)
index.htm
  l instead.
  Security Store uses: JKS
  Domain openiddomain created.
  Admin login information for host [localhost] and port [24848] is being overwritt
  en with credentials provided. This is because the --savelogin option was used
du
  ring create-domain command.
  Login information relevant to admin user name [admin] for this domain [openiddom
  ain] stored at [C:¥Users¥yasu¥Downloads¥.asadminpass] successfully.
  Make sure that this file remains protected. Information stored in this file
will
   be used by asadmin commands to manage this domain.
```

17)作成したドメインを起動する

```
C:¥gf¥glassfish¥bin>asadmin  start-domain openiddomain
```

18)成功すると以下のようなメッセージが出る

```
  Starting Domain openiddomain, please wait.
  Default Log location is
C:¥glassfish¥glassfish¥domains¥openiddomain¥logs¥server.
  log.
  Redirecting output to
C:/glassfish/glassfish/domains/openiddomain/logs/server.lo
  g
  Domain openiddomain is ready to receive client requests. Additional services
are
   being started in background.
```

Domain [openiddomain] is running [Sun GlassFish Enterprise Server v2.1.1 ((v2.1 Patch06)(9.1_02 Patch12)) (build b31g-fcs)] with its configuration and logs at:

[C:¥glassfish¥glassfish¥domains].

Admin Console is available at [http://localhost:24848].

Use the same port [24848] for "asadmin" commands.

User web applications are available at these URLs:

[http://localhost:28080 https://localhost:58527].

Following web-contexts are available:

[/web1 /__wstx-services].

Standard JMX Clients (like JConsole) can connect to JMXServiceURL:

[service:jmx:rmi:///jndi/rmi://yasu-PC:58528/jmxrmi] for domain management purpo

ses.

Domain listens on at least following ports for connections:

[28080 58527 24848 58526 3820 3920 58528].

Domain does not support application server clusters and other standalone instanc

es.

19) 「http://localhost:28080/」にアクセスできることを確認する。

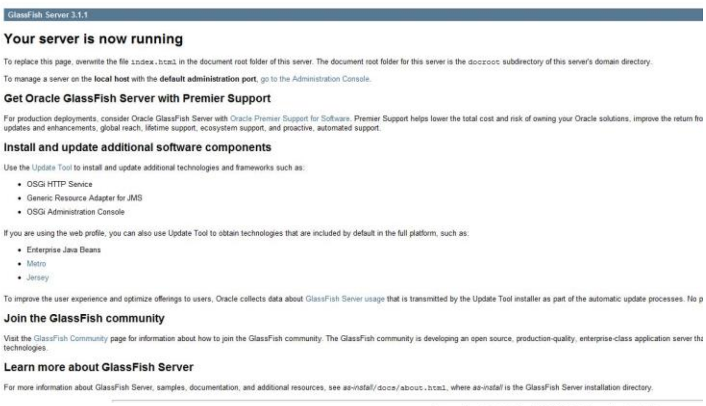

図 5-1-5-8:GlassFish の起動確認

20)openam ファイルを「openam.war」に改名し、以下に配置する。

C:¥gf¥glassfish¥domains¥openiddomain¥autodeploy

21) http://localhost:28080/openam/config/options.htm にアクセスする。

図 5-1-5-9:OpenAM のホーム画面

5-1-6　OpenAM と OAuth

（図 5-1-6-1～図 5-1-6-7、表 5-1-6-1 参照）

OpenAM10.0 が提供する Consumer 機能を使用

facebook の認証方式 OAuth と OpenAM を連携する。Facebook のアカウントを所持していない場合、作成をすることになるが、作成直後のアカウントを使用することができない仕様になっている。

http://www.osstech.co.jp/_media/techinfo/opensso/openam_oauth_setting.pdf

1) https://developers.facebook.com/apps にアクセス

2)右上の「新しいアプリを作成」をクリック

3)アプリ名やカテゴリは仕様に深く関わらない。

今回はアプリ名を OpenAM、カテゴリを教育とする。

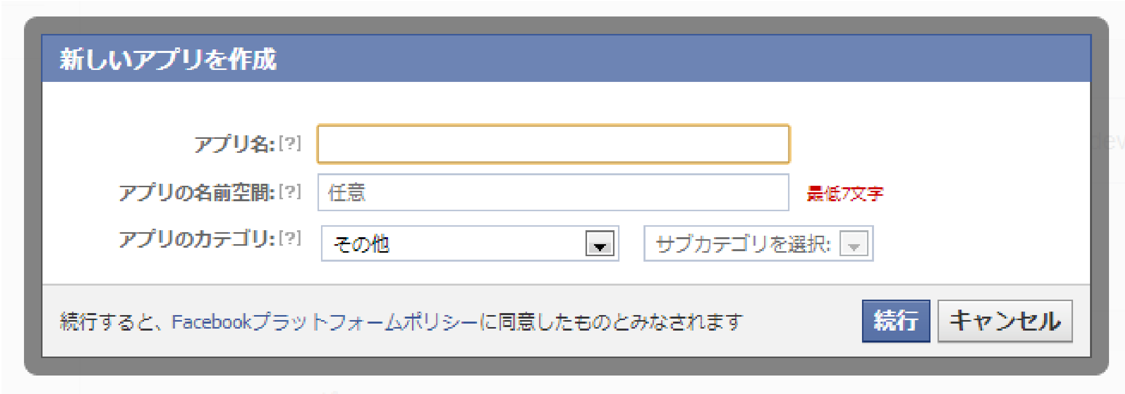

図 5-1-6-1:facebook Developer でのアプリ作成画面

4)設定にワンタイムパスワードを要求される

図 5-1-6-2:facebook Developer からのワンタイムパスワード

5)作成時に生成される APP キーと APP シークレットをこの後の手順で使用する。

図 5-1-6-3:facebook Developer でのアプリ作成画面

6)App ドメインとサイト URL を記入

App ドメインには OpenAM サーバーの FQDN を記入

サイト URL には OpenAM の URL を記入する。

図 5-1-6-4:facebook Developer でのアプリ作成画面

7)OpenAM の認証画面を開く

アクセス制御タブ→（対象のレルム）→認証タブ

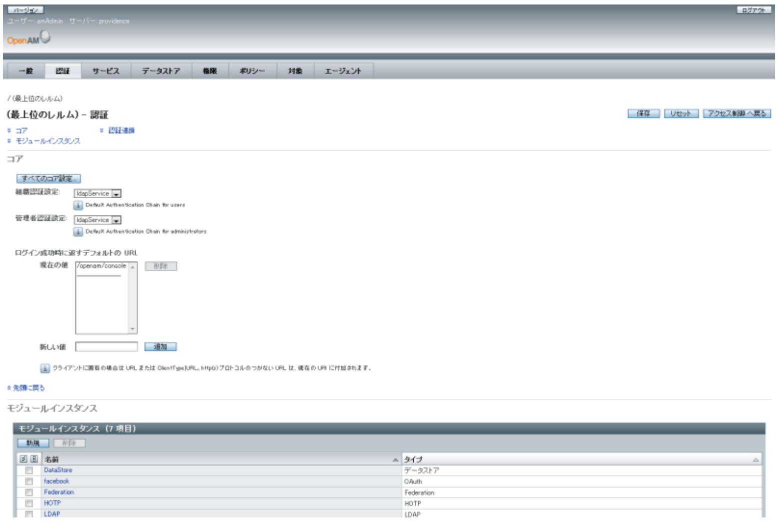

図 5-1-6-5:OpenAM 設定画面

8)モジュールインスタンスから新規を選択

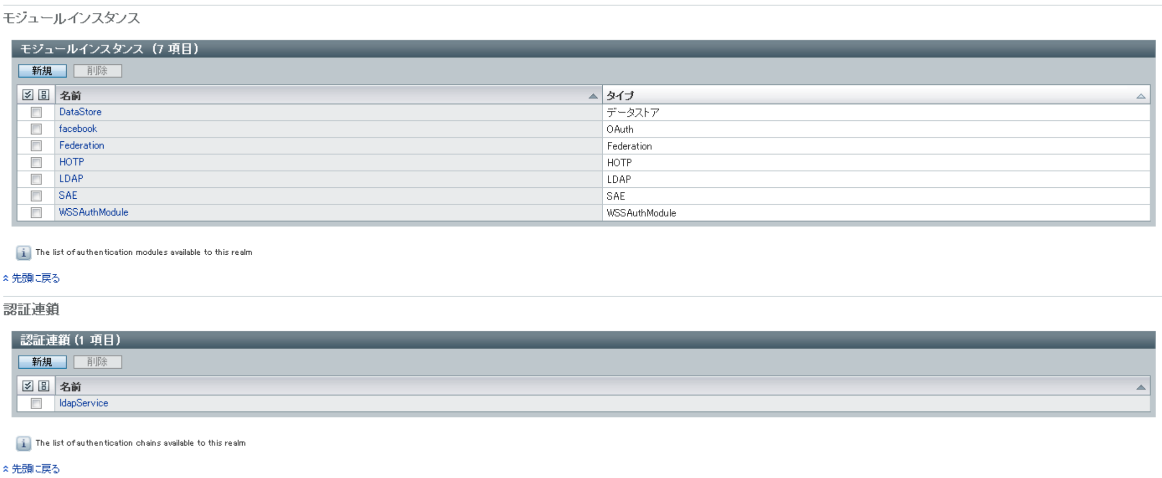

図 5-1-6-6:OpenAM　設定画面

9)名前を Facebook　とし、OAuth 認証モジュールの作成は終了

10)必要事項を記入していく

以下表に必要事項を記載する。

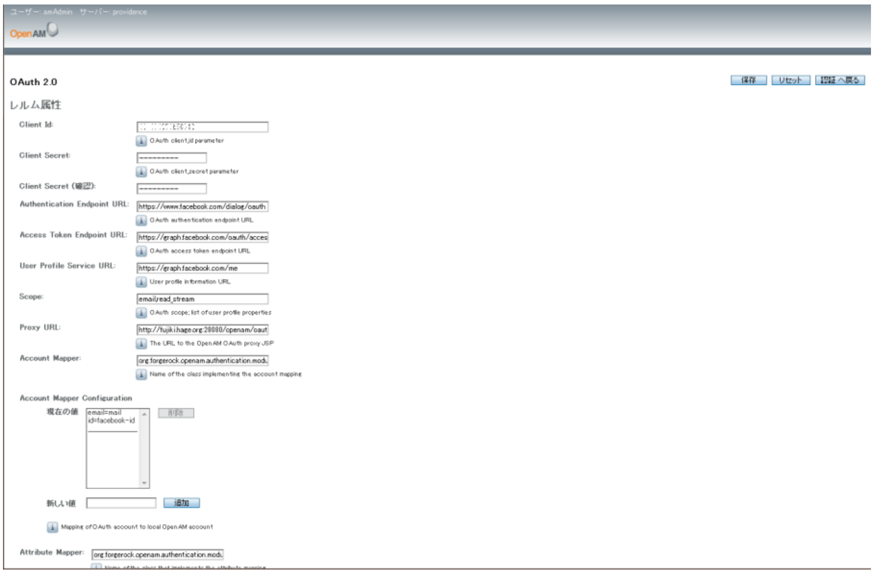

図 5-1-6-7:OpenAM 設定画面

表 5-1-6-1:OpenAM 設定画面

パラメーター名	設定値
ClientID	「4)」で登録した Facebook アプリケーションの App ID/App Key を入力する
Client Secret	「4)」で登録した Facebook アプリケーションの App シークレットを入力する
Authentication Endpoint URL	https://www.facebook.com/dialog/oauth (デフォルト)
Access Token Endpoint URL	https://graph.facebook.com/oauth/access_token (デフォルト)
Userprofile Service URL	https://grapth.facebook.com/me (デフォルト)
Scope	email,read-stream (デフォルト)
Proxy URL	https://fujiki.hage.org:28080/openam/oauth2c/OAuthProxy.jsp
Account Mapper	(デフォルト)

Account Mapper Configuration	email=mail
Attribute Mapper	(デフォルト)
Attribute Mapper Configuration	last_name=sn email=mail first_name=givenname name=cn
Save attribute in the session	有効
Email attribute in OAuth2 Response	email
Create account if it does not exit	無効
Prompt for password setting and activation code	無効
Map to anonymous user	無効
Anoymous User	anonymous(デフォルト)
Oath2.0 Provider logout service	http://www.facebook.com/logout.php (デフォルト)
Logout opions	Prompt (デフォルト)

Facebook とのアカウントの連携は、メールアドレスを利用して実施するため、作成したアカウントには、facebook 側のメールアドレスを必ず設定する必要がある。

5-2　動作の確認

1)認証モジュールに facebook を指定して OpenAM にアクセスする「https://[openam サーバーの FQDN]:[ポート番号]/openam/UI/Login?module=facebook」

2)示される facebook のログイン画面でメールアドレス及びパスワードを入力

図 5-2-1:facebook との連携

3)ニュースフィードへのアクセスを求める同意画面が表示されるため、許可をする

4) OpenAM のパスワード確認画面が表示されるので入力し、Submit を選択する

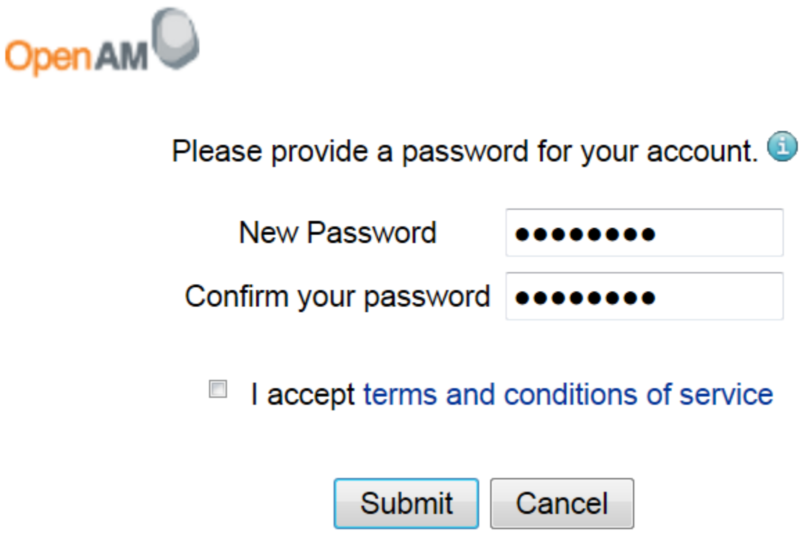

図 5-2-2:facebook との連携

5）最初のアクセスのみ表示され、２回目以降のアクセスでは、アカウントが連携されているため、画面は表示されない

図 5-2-3:OpenAM 設定画面

5-3　動作の評価

日常生活で学内の Web サービスを利用する際に重要となる点は「安全性」と「利便性」である。一度のログインで認証情報をサーバーに記憶させ、暗号化をしてサーバーに保存。次にログインを行った際にサーバーから自動的に認証情報を出力させる機能を兼ね備えている SSO が便利だ。しかし数多くの認証方式に対応している OpenAM を使った認証方式のうち今回試験導入したのは OAuth2.0 のみであった。学内に存在する各 Web サービスの認証方法までは調べきることができなかったが豊富な認証方式に対応している OpenAM であれば学内の Web サービスの SSO 化も不可能ではない。

SSO を実現させている既存のアプリケーションの例として RoboForm（ロボフォーム）がある。RoboForm は無償で利用する際には ID とパスワードが 10 件までしか登録できず、個人向けの利用には向いていなかったが大学という環境での利用で有料版を購入することができれば 10 件までだったログイン登録も無制限になる。

しかし、RoboForm を購入し、学内の全 PC に RoboForm をインストールする労力や RoboForm の使い方を一から学生、職員、教員に教えるための手段を考えるとアプリケーションでの SSO の実現は難しいといえる。利用するユーザー側の目線から見てアプリケーションへの登録の労力や時間をなくした上で認証方式を OpenAM に適応させるだけで SSO を可能にする OpenAM は優れているといえる。（図 5-3-1、表 5-3-1 参照）

参考文献【18】

図 5-3-1:OpenAM が対応している認証方式一覧

表 5-3-1:RoboForm 無料版と有料版の違い

	無料版	**有料版**
「ログイン帳」の数	10 個まで	無制限
法人	×	○
ユーザー数	なし	無制限

5-4　考察

本研究ではログインにおける労力を極力軽減させる方法について検討・提案を行うことでよりよい学内 Web サービスの利便性を向上させ、本研究で提案される手法を適用することで、より快適な学内 Web サービスの利用環境を将来の入学する学生や在学中の学生に提供することを目指した。目標としていた学内の Web サービス環境を改善することはできなかったが、OpenAM の導入や実際に OpenAM が提供する OAuth2.0 認証を使った facebook の SSO の実現をすることができた。

本研究で学内の Web サービスを利用した SSO システムの実現ができていれば、学内の様々な Web サービスを一度の ID、パスワード入力だけで認証させることが可能にな

る。さらに、各 Web サービスのサーバーに登録されている情報を SSO アカウントに移行できれば、学内に存在する全 Web サービスが一種の ID とパスワードだけでログインすることができる。これらの機能を適用し、より良い学内 Web サービスの SSO システムを目指し取り組むべきである。

6　まとめと今後の課題

6-1　まとめ

本研究の目的はログインにおける労力を極力軽減させる方法について検討・提案を行うことでよりよい学内 Web サービスの利便性を向上させること、本研究で提案される手法を適用し将来の入学する学生や在学中の学生に提供することの 2 つである。実際に OpenAM を利用した SSO を実現したが本研究では OpenAM の Consumer 機能である OAuth2.0 を利用した facebook の SSO のみとなった。Ubuntu や CentOS などざまざまな OS を使用し、各ソフトウェアのバージョンの違いが本研究で問題として大きくあげられた。

6-2　今後の課題

今後の課題についてこれまでの考察に基づき述べる。目標としていた学内の Web サービスを利用した SSO システムの実現をすべて研究することができなかった。したがって今回できなかった機能の完成を目指す。学内の環境から OpenAM を利用すること、学内の Web サービスに SSO システムを導入することである。特に OpenAM のインストールや設定については OpenAM を起動させるための Tomcat や JDK、glassfish の各バージョンの相性が使用する OS によって異なってしまう点は今後研究を進めていく上で注意するべき点といえる。そして、目標としていた学内の Web サービスを利用した SSO システムの実現することができなかったので、今後より良い SSO システムになるよう努力する。

7　参考文献

【1】とはサーチ

http://www.toha-search.com/it/login.htm

2013 年 11 月 30 日

【2】CodeZine コードジン

http://codezine.jp/

2013 年 11 月 8 日

【3】ForgeRock

http://forgerock.com/

2013 年 9 月 26 日

【4】Tomcat

http://tomcat.apache.org/

2013 年 10 月 18 日

【5】LoginCode

http://logincode.com/content/home.php

2013 年 7 月 12 日

【6】RoboForm

http://www.roboform.com/jp

2013 年 6 月 7 日

【7】ありえるえりあ

http://dev.ariel-networks.com/wp/

2013 年 11 月 30 日

【8】IT 用語辞典 e-Words

http://e-words.jp/w/CRM.html
2013 年 11 月 8 日

【9】学認 GakuNin
http://www.gakunin.jp/
2013 年 9 月 26 日

【10】OSS Tech オープンソース・ソリューション・テクノロジ株式会社
https://www.osstech.co.jp/
2013 年 11 月 29 日

【11】SIOS "OSS よろず" ブログ出張所/小川氏
http://sios-oss.blogspot.jp/2012/09/openam-opendj.html
2013 年 10 月 25 日

【12】Diary
http://spiral.world.coocan.jp/diary/
2013 年 11 月 15 日

【13】LAJ 技術ブログ/taka 氏
http://www.la-j.com/tech-blog/?cat=103
2013 年 11 月 29 日

【14】メモーる/山川氏
http://memoyasu.blogspot.jp/
2013 年 11 月 29 日

【15】JavaDrive/Tatsuo Ikura 氏
http://www.javadrive.jp
2013 年 12 月 12 日

【16】Shinya’s Daily Report/しんや氏

http://d.hatena.ne.jp/absj31/

2013年12月11日

【17】Facebook Developers

https://developers.facebook.com/

2013年12月11日

【18】RoboFormの使い方

http://roboform.siyo.org/

2013年12月13日

【19】Indira Thangasamy「Openam」Packt Publishing

【20】鶴長 鎮一「サーバ構築の実際がわかる Apache[実践]運用/管理」技術評論社

【21】北見工業大学 シングルサインオンシステム

https://kit-sso.cc.kitami-it.ac.jp/portal/login/login.jsp

2013年12月13日

8　謝辞

本研究を行うにあたり、常に丁寧でわかりやすい指導をしてくださった指導教員の渡辺恭人准教授に大変感謝しております。論文の執筆から研究のための資材の提供に至るまでご丁寧に語移動していただきました。結果卒業論文を完成させることができたと感じており感謝の念でいっぱいです。　渡辺ゼミでは、プログラムの基礎やその難しさ、そして身近にある「あったらいいな」を大切にする精神を知ることができました。作れなくても形にできなくても動かしたい、直したい、もっとよくしたいという思いが大切なのだと先生から学び、本研究をしたいと思いました。

最後に、私の卒業研究に関わって下さった方全員にもう一度感謝を述べさせて頂き謝辞とさせていただきます。ありがとうございました。