特 別 研 究 報 告 書

題 目

メールアーカイブの有効利用を促進する外部連携用APIの設計と実装

指導教員

乃 村 能 成

報 告 者

北添 稚菜

岡山大学工学部 情報工学科

平成 23 年 2 月 14 日 提出

要約

メーリングリスト (以下 ML) は，同好の不特定多数を結ぶツールとして発展してきた．このため，特定の話題についての議論や情報収集は ML によって行われることが多かった．しかし，現在では，このような議論や情報収集は Wiki や掲示板で行われるのが一般的となっている．一方で，現在 ML は，小さな部署や特定のプロジェクトにおいてコミュニケーション手段や文書の共有スペースとしてさかんに利用されている．つまり，ML が大規模で不特定多数向けの利用から，小規模で特定のグループ向けの利用へと変化している．

そこで，小規模で特定のグループ向けの利用を支援するため，グループワークを支援するメーリングリスト機構が提案された．この ML 機構はグループワークを支援に特化した機能として，Web コンテンツとのマッシュアップ機能を実現するためのフレームワークを提供する．このフレームワークを利用し，外部連携用 API を設計することによって，従来 ML 内で閉じていた情報と他サービスとの連携が可能となる．組織で使用されている Web コンテンツと連携を取ることによって，ML からの一方的なやり取りではなく，双方向での情報交換を実現する．この情報交換が実現することにより，Web コンテンツを使用した組織内での情報共有が促進され，組織全体の仕事効率が上昇すると考えられる．

本論文では，ML 内で閉じてしまっている ML やアーカイブ情報を外部から利用するための連携用 API を設計した．外部連携用 API は，多くの Web サービスと連携が行えること，ML メンバ以外の情報も利用できることを方針とした．

まず，多くの Web サービスと連携を行うために，通信プロトコルとデータ形式は現在多くの Web サービスで利用されている REST と XML に決定した．次に，ML メンバ以外の情報として，ML に関する情報，ML メンバの詳細情報，ML アーカイブ情報，メール情報を選択し，操作対象によって許される操作，許されない操作を決定した．

また，この外部連携用 API は所属する研究室内で運用しているグループウェアとの連携が可能である．そして，一部の機能を実装した．

目次

図 目 次

表 目 次

第 1 章

はじめに

メーリングリスト (以下，ML) は，複数の人に同時に電子メールを配送する機構である．このため，同好の不特定多数を結ぶツールとして発展してきた．例えば，Linux に関する特定の話題についての議論や情報収集をするため，同好者による ML を利用するといったことが一般的であった．しかし，現在では，このような議論や情報収集は Wiki や掲示板で行われることが多くなっている．

一方で，現在 ML は，小さな部署や特定のプロジェクトにおいてコミュニケーション手段や文書の共有スペースとしてさかんに利用されている．つまり，ML が大規模で不特定多数向けの利用から，小規模で特定のグループ向けの利用へと変化している．

そこで，小規模で特定のグループでの利用を支援するため，グループワークを支援するメーリングリスト機構が提案された [1]．この提案機構は Web コンテンツとのマッシュアップ機能を提供する．この機能はメンバ管理に対して，それらを外部から利用するためのフレームワークを提供する．このフレームワークを利用し，外部連携用 API を設計することによって，従来 ML 内で閉じていた情報と他サービスとの連携が可能となる．組織で使用されている Web コンテンツと連携を取ることによって，ML からの一方的なやり取りではなく，双方向での情報交換を実現する．例えば，打合せの日時の情報を議事録として送信されたメール情報から取得し，組織で利用しているスケジューラに登録という連携が可能となる．

本論文では，ML 内で閉じてしまっている ML やアーカイブ情報を外部から利用するための外部連携用 API について検討する．まず，グループワークを支援する ML 機構の特徴を示し，問題点について述べる．次に，API への要求をまとめることで，API の設計方針として使用する通信プロトコルとデータ形式を定める．そして，設計方針に基づいた外部連携用 API の設計について述べる．

第 2 章

グループワークを支援するML機構

2.1 特徴

小規模特定のグループ向けの ML 機構が，Web サービスとのマッシュアップを支援する ML 機構として提案された．この ML 機構は，以下の 3 つの機能を提供する．

(1) アーカイブのカスタマイズ機能

アーカイブされているメール情報を加工することによる他 Web サービスとの連携を考える．また，こうした加工機能をメンバが自由に追加，削除することでアーカイブのカスタマイズが可能となる．

(2) 他 Web サービスの利用支援機能

メール偏重ユーザを Web サービスへスムーズに誘導するために，メール本文にそのメールと関連する Web サービスの情報を挿入する．例えば，Web サービスへのリンクをメール本文末尾に付加する．このとき，ユーザはリンクを辿るだけで Web サービスを利用出来るため，ユーザを負荷無く Web サービスへ誘導できると考えられる．

(3) Web コンテンツとのマッシュアップ機能

メンバ管理やアーカイブの各機能に対して，それらを外部から利用するための API を提供する．これにより従来 ML 内に閉じていた情報と他サービスとの連携が可能となる．これは，ML からの一方的なやり取りではなく，双方向での情報交換を実現する．

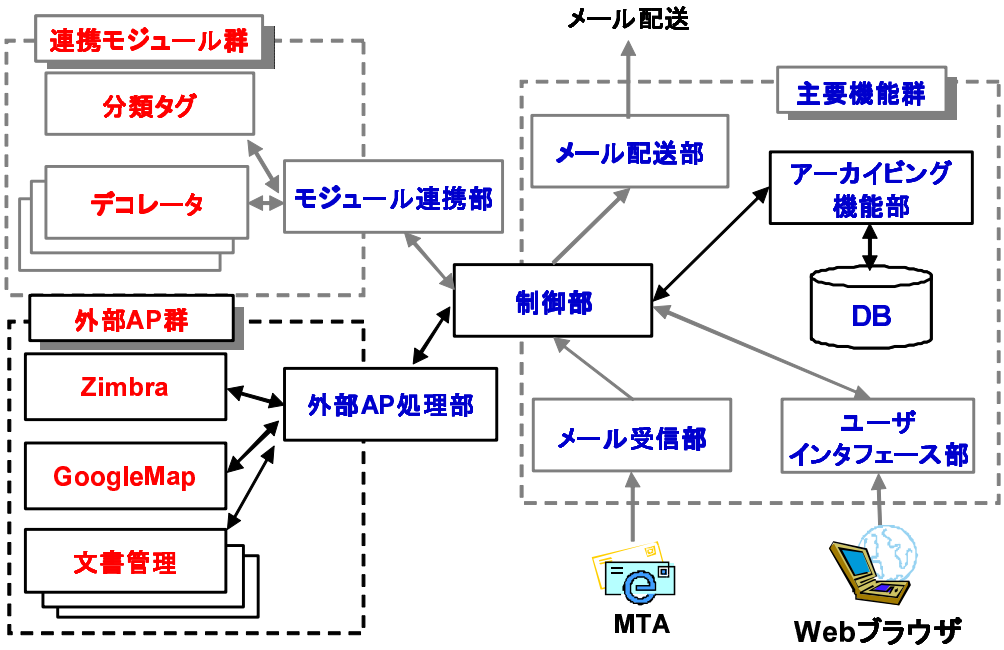

図　2.1　ML 機構のシステム構成

2.2　全体のシステム構成

2.1 節で ML 機構が提供している機能について述べた．この特徴に基づいて設計されたグループワークを支援する ML 機構のシステム構成を図 2.1 に示し，特に外部 AP 群について以下に説明する．

外部 AP 群は，以下の 2 つから構成される．

(1) 外部の AP と情報のやり取りを行う外部 AP 処理部

外部 AP 処理部は，外部 AP と制御部との情報交換，あるいは制御部から外部 AP へのアクセスを行う際の窓口となる．

(2) メール情報をもとにマッシュアップを行う外部の AP

外部 AP が取得する情報は，メンバの情報，アーカイブされているメール，および添付ファイルである．

DB にはメンバ情報やアーカイブ情報などが保存されている．

2.3 問題点

2.1 節で述べた特徴のうち，(3) 他 Web コンテンツとのマッシュアップ機能には現在 API の仕様設計が不十分であるという問題が存在する．その問題点の詳細を以下に 3 つ示す．

(1) 通信プロトコルとデータ形式が決定していない

通信プロトコルとデータ形式が決定しておらず，API の使用方法や返却される情報の型が不明である．ML 内で閉じていたメンバ情報やアーカイブの情報を外部の AP に提供するためには，API の使用方法を明確に決定する必要がある．

(2) 操作対象の選別と分類がされていない

ML 機構には，プロトタイプとして ML メンバ情報を取得する機能は実装されているが，異なる操作対象を明確に区別するための書式について考察がされていない．操作対象として何が必要か，またどのように分類するか考察する必要がある．

(3) 対象毎に許される操作と形式に関する考察がされていない

対象毎に許される操作に関する考察がされていない．外部から API を使用する際，どのような操作が必要となるか，また各対象に対して許される操作に関する考察が必要となる．

第 3 章

外部連携用 API

3.1　API への要求

2.1 節でグループワークを支援する ML 機構の特徴，2.3 節でその ML 機構の問題点について述べた．そこから，外部連携用 API が備えるべき機能として，次の 2 つが考えられる．

(1) 多くの Web サービスと連携を行いたい

ML 内だけで閉じていたメンバの情報やアーカイブの情報を外部の Web サービスから利用する連携が考えられる．この Web サービスは ML 機構が選択した Web サービスだけではなく，組織内で利用されている様々なサービスを想定する．こうすることにより，API を使う側はサービスを限定されることなく，自由に他サービスとのマッシュアップを行うことができる．

(2) ML メンバ以外の情報も取得したい

ML 機構は，ML メンバ情報だけではなく，メール情報など様々な情報を保持している．このため，ML メンバ情報以外の，アーカイブ内のメール情報なども外部から取得，操作を行いたい．そのためには，操作対象を明確に区別するための書式や操作対象毎に許される操作について考察が必要となる．

これらの要求を満たすことにより，例えばメール文中にあるスケジュール情報を抽出し，組織で利用しているスケジューラに登録するといった連携が可能になる．

3.2 通信プロトコルの検討

WebAPI で公開する際の主な通信プロトコルとして REST と SOAP がある．以降の項で REST と SOAP の特徴について述べ，3.4 節にて比較を行う．

3.2.1 REST

REST とは，Representational State Transfer の略で，HTTP[2] を使って通信を行う手法である．特徴は以下の通りである．

(1) ある URL にアクセスすると決められた形式で記述されたメッセージが返ってくる

(2) Web ブラウザに URL を入力すれば動作確認できる

(3) リソースをリンクで接続することで 1 つのアプリケーションを構成する

(4) リソースに適用できる HTTP メソッドは常に固定である

また，REST において重要な HTTP メソッドは次の 4 つである．

(1) GET

GET は指定した URI の情報の取得を意味する．最も使用頻度の高いメソッドで，Web ページの取得，画像の取得，映像の取得など，ブラウザを利用しているときは常に数多くの GET を発行している．

(2) POST

POST は子リソースの作成や既存リソースへのデータの追加を意味する．また，GET よりも大量のデータを送信して URI の情報を取得することができる．POST は GET に次いで利用頻度の高いメソッドであり，様々な用途に使われている．

(3) PUT

PUT はリソースの作成やリソースの更新を意味する．POST でリソースを作成する場合はクライアントがリソースの URI を指定することはできないが，PUT でリソースを作成する場合はクライアントがリソースの URI を決定することができる．

表 3.1 CRUD と HTTP メソッドの対応

CRUD 名	意味	メソッド
Create	作成	POST/PUT
Read	読み込み	GET
Update	更新	PUT
Delete	削除	DELETE

(4) DELETE

DELETE はリソースの削除を意味する．DELETE のレスポンスはボディを持たない．そのため，レスポンスのステータスコードには，`200 OK`，`202 Accepted` 以外にもボディがないという意味の `204 No Content` が使われる場合もある．

これらのメソッドは,「CRUD」という性質を満たすため，代表的なメソッドと言える．CRUD とは，Create(作成)，Read(読み込み)，Update(更新)，Delete(削除) というデータ操作の基本となる 4 つの処理のことである．CRUD と HTTP メソッドは表 3.1 のように対応する．

REST は HTTP を使って通信を行う手法のため，数多くの Web サービスで使われている．Twitter[3] や Yahoo![4]，Amazon.com[5] が提供する API は REST で提供されている．

3.2.2 SOAP

SOAP とは，SOAP メッセージという XML によってメッセージ交換をおこなう方法である．元々Simple Object Access Protocol の略だったが，目的がオブジェクトにアクセスするだけではなくなってしまったため，何の略でもなく単に SOAP であると改定された．特徴は以下の通りである．

(1) XML でありさえすれば何を書いても良く，自由に拡張することができる

(2) REST に比べると高機能である

図 3.1 に SOAP の構造を簡単に示している．一番外側に ⟨Envelope⟩，その下に ⟨Header⟩ と ⟨Body⟩ を記述する．⟨Header⟩ はオプショナルで，⟨Body⟩ は主要な情報を格納している部分である.「必ずルート要素は ⟨Envelope⟩」「その下は ⟨Header⟩ と ⟨Body⟩」(⟨Header⟩ は無くて

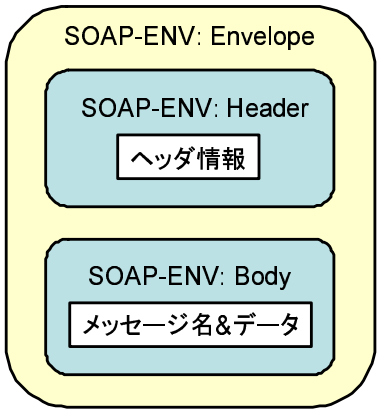

図 3.1 SOAP の構造

も良い）とすることで，システムで取り扱い易くした．これは『定型の封筒に入れれば配達員は中身を知る必要が無く，中身の便箋に何を使っても何を書いても良い』のと似ている．

SOAP はメッセージ転送の方法だけを定めた仕様のため，実際にシステムを構築する際には SOAP の上にサービスごとのプロトコルを定義しなければならない．これらを各開発グループばらばらに定義するのでは標準化競争が起こってしまうため,「WS-*」と呼ばれる周辺仕様群が次々に提案された．しかし同じような仕様が複数乱立してしまったため，結局標準化競争を引き起こしてしまい，また WS-*の仕様書群が膨大になり実装が困難になった [6]．従って，以下の問題が SOAP にはある．

(1) 仕様が複雑で実装が難しい

(2) 相互運用性が低い

3.3 データ形式の検討

WebAPI で公開する際の主なデータ形式として XML と JSON がある．以降の項で XML と JSON の詳細について述べ，3.4 節にて比較を行う．

3.3.1 XML

XML とは，文書やデータの意味や構造を記述するためのマークアップ言語の一つである．マークアップ言語とは,「タグ」と呼ばれる特定の文字列で地の文に情報の意味や構造，装飾

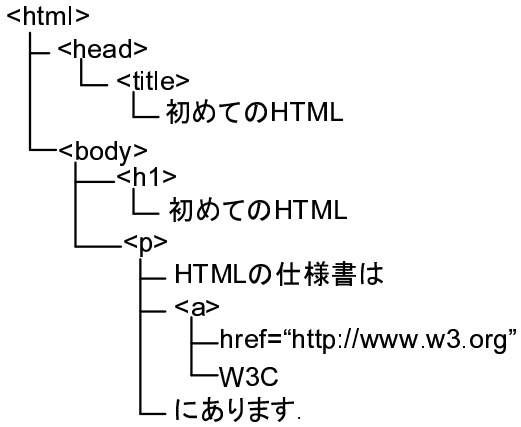

図 3.2 XML の木構造

などを記述している言語のことである．XML はユーザが独自のタグを指定できることから，マークアップ言語を作成するためのメタ言語とも言われる．

以下に示す XHTML の例を用いて説明をしていく．

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
   <head><title>初めての HTML</title></haed>
   <body>
     <h1>初めての HTML</h1>
     <p>HTML の仕様書は<a href="http://www.w3.org">W3C</a>にあります．</p>
   </body>
</html>
```

XML 文書は木構造として表現できる．先の XHTML 文書を木構造として図示すると図 3.2 のようになる．

XML は要素 (Element) で文書の構造を表現する．要素は開始タグ (Start Tag)，終了タグ (End Tag) から成る．タグには要素名が入り，開始タグは<要素名>，終了タグは</要素名>と記述する．図 3.3 に XML の構成要素を示す．

要素は属性 (Attribute) を複数持つことができる．属性は属性名と属性値の組で，開始タグの中に 属性名=”属性値” という形式で記述する．属性名，属性値ともに文字列である．また，属性には以下の特徴がある．

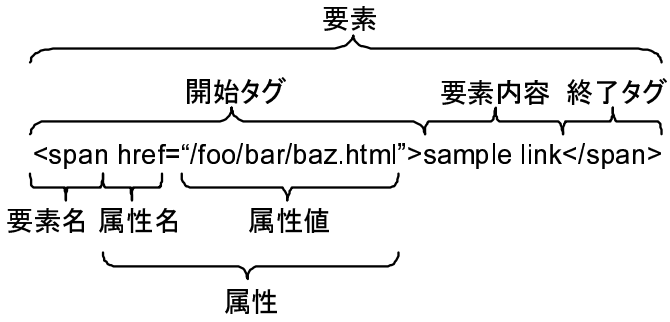

図 3.3 XML の構成要素

(1) 開始タグは属性を複数持つことができる

(2) 同じ名前の属性は 1 つだけしか記述できない

(3) 属性は入れ子にはできない

(4) 開始タグの中での属性の順番には意味が無い

XML 文書の先頭には XML 宣言 (XML Declaration) を書く.

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
...
</html>
```

XML 宣言では，XML のバージョンや文字エンコーディング方式を指定する．XML のバージョンには 1.0 または 1.1 があるが，一般には 1.0 を用いる.

3.3.2 JSON

JSON とは JavaScript Object Notation の略で，RFC 4627[7] が規定するデータ記述言語である．JavaScript の記法でデータを記述できる点が最大の特徴である．記法は JavaScript だが，そのシンプルさから多くの言語がライブラリを用意しているため，プログラミング言語間でデータを受け渡すことができる.

Web サービスでは，次の利点から JSON が Ajax 通信におけるデータフォーマットとして活用されている.

(1) ブラウザが JavaScript を実行できるので相性が良い

(2) XML と比べてデータ表現の冗長性が低い

JSON に組み込みで用意されているデータ型には次の 6 つがある．

(1) オブジェクト

オブジェクトは名前と値の集合である．名前と値の組をオブジェクトの「メンバ」と呼ぶ．

(2) 配列

配列は順序を持った値の集合である．ゼロ個以上の値を持つことができる．

(3) 文字列

JSON の文字列は必ず二重引用符 (") で囲む．また，バックスラッシュ(\) や改行といったコントロール文字は，特殊なエスケープ表記を持っている．

(4) 数値

JSON の数値には整数と浮動小数点数の両方が含まれる．数値の表記は 10 進表記に限る．

(5) ブーリアン

値に真か偽をとるブーリアン型はリテラルで用意されている．「true」と「false」のように必ずすべてを小文字で書く必要がある．

(6) null

null 値もリテラルで用意されている．必ず「null」と小文字で書く必要がある．

3.4 通信プロトコルとデータ形式の決定

3.1 節で述べた API への要求を元に，以下の 3 つの理由から通信プロトコルは REST，データ形式は XML を用いることに決定した．

(1) 多くの Web サービスが REST，XML を利用している

Twitter や Yahoo! API を代表に，多くの Web サービスが REST，XML を利用している．また，SOAP などを基盤とする Web サービスの標準化を行ってきた団体が活動を終了したということもあり，REST ベースの API が急速に充実してきている [8]．

(2) REST は開発が行いやすく，利用しやすい

ML 機構のユーザインタフェースは Web ベースである．このため，HTTP を用いて簡単に情報を授受できる REST とは相性が良い．

(3) XML は自由にタグが作成できるマークアップ言語である

自由にタグが作成できるため，統一的な記法を用いながら独自の意味や構造を持った言語を作成することができる．ソフトウェア間の通信・情報交換に用いるデータ形式の定義にも使用されている．JSON はマークアップ言語ではないため，独自の意味や構造を持った言語を作成することが難しい．

第 4 章

設計方針

4.1 操作対象

API の操作対象となる ML のメンバ情報やアーカイブ情報は DB に保存されている．この DB が持つ情報の関係を図 4.1 に示す．図中において 1 対多は 1-*，多対多は*-*となる．
この構成において，API で操作可能な対象をクラスとして以下に定める．

(1) ML サーバクラス

ML もメンバ情報だけではなく ML に関する情報を保持している．ML サーバクラスは複数の ML 情報を総括している．

(2) ML クラス

ML クラスは ML に関する情報を保持している．また，ML に属しているメンバのユーザ情報の一部を保持している．この情報をもとに，同じく DB に保存されたユーザ情報にアクセスすることができる．

(3) メンバクラス

ML に属しているユーザをメンバと呼ぶ．メンバクラスはメンバ名や属している ML の情報を保持している．

(4) ML アーカイブクラス

ML アーカイブクラスとは ML クラスに関連付けられているメール情報を総括して定義したクラスである．

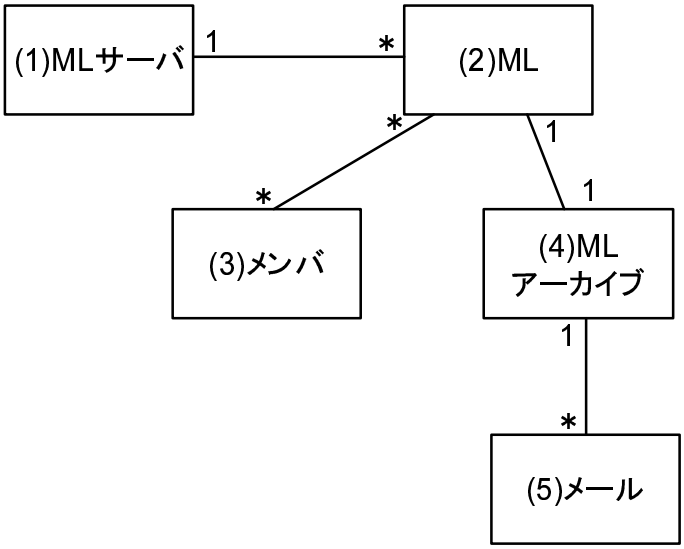

図　4.1　操作対象の関係図

(5) メールクラス

メールクラスは ML に送信されたメールの情報を保持している．

4.2　操作

以下に操作の種類を 4 つ示している．方針は REST の CRUD に基づいている．

(1) Read

操作対象自身の情報や操作対象から見て 1 対多の関係にあるものの中で条件に一致するもの一覧を取得する．

(2) Create

操作対象自身を新規追加する．

(3) Update

操作対象自身を更新する．

(4) Delete

操作対象自身を削除する．

表 4.1 可能な操作一覧

	ML サーバ	ML	メンバ	ML アーカイブ	メール
Read	○	○	○	○	○
Create	×	○	×	×	×
Update	×	○	×	×	×
Delete	×	○	×	×	×

また，操作対象毎に可能な操作を表 4.1 にまとめている．行の項目が操作対象，列の項目が操作である．表中で，許される操作が○，許されない操作が×と定めている．例えば，操作対象が ML アーカイブの場合，許される操作は Read のみである．操作対象によって許されない操作を定めた理由は以下の通りである．

(1) ML サーバ

ML サーバは ML に関する情報を保持しているが，ML サーバ自身の情報は外部から操作するべきではない．よって，新規追加，更新，削除の操作は必要ではない．

(2) メンバ

この API は ML や ML アーカイブに関する情報を取得や操作するものである．よって，メンバの新規追加や更新，削除の操作は行うべきではない．

(3) ML アーカイブ

ML アーカイブは ML と 1 対 1 の関係なため，ML アーカイブだけを新規追加や削除するという操作は考えられない．また，ML アーカイブ自身を更新をするという動作自体存在しない．

(4) メール

メールとは本来更新という行為は行われない．また，ML アーカイブの持つメール情報を外部から操作するべきではない．よって，メール情報の新規作成，更新，削除という動作は必要ではない．

また，上記の操作のうち，Create，Update，Delete の操作を全ユーザが行うことができるようにすると，セキュリティ上の問題が発生する．このため，これらの操作は管理者権限を持つユーザのみが行うことができるように設定する．

4.3　取り扱う情報

表 4.2 に各操作対象が持っている全情報を示す．この情報の中で，取扱いの欄に×が付いている情報は本 API で取り扱わない情報である．

一部の情報を取り扱わないことにした理由は以下の通りである．

(1) メール情報

- to

  メール情報を参照する場合，ML を指定して ML アーカイブにアクセスするため，受信者は自明である．

- cc

  Cc された受信者のメールアドレスを全て渡してしまうとセキュリティ面で問題となる．

- depth

  この情報はメールスレッドを作成するためのものである．API には必要ない．

- mailinglist_id

  ML を指定して ML アーカイブにアクセスするため，どの ML に送られたメールであるかは自明である．

(2) ML

- id

  ML を区別可能であるため，id の情報は必要ない．また，内部の管理情報である id を外部に出すのはふさわしくない．

4.4　取り扱うデータの型

本 API で取り扱うデータは以下の 5 つの型に定めた．これはシステムの扱うクラスと対応している．

(1) MailingLists 型 (ML サーバ)

MailingLists 型とは ML の一覧を表す型である．

(2) MailngList 型 (ML)

MailingList 型とは ML に関する情報を表す型である．ML の名前や属するプロジェクト，属しているメンバ等の情報が含まれている．プロジェクトとは，ML を利用しているグループのことである．メンバの情報を含むのはセキュリティ面で問題となる可能性があるが，今回のシステムでは ML に属しているメンバは互いに既知であるため，取り扱う情報に含めている．

(3) Member 型 (メンバ)

Member 型とは ML を利用しているメンバの ML に関する情報を表す型である．メンバ名と送信したメール数や属している ML の一覧の情報を含めている．

(4) Mail_info 型 (ML アーカイブ)

Mail_info 型とはメール情報の一部を表す型である．メールの一覧を取得したい場合，メールの本文を全て参照していたのでは取得する情報が膨大になってしまう．そこで，件名や送信日時などの最低限の情報だけを取得したいときにこの型を使用する．

(5) Mail 型 (メール)

Mail 型とはメール情報を全て含めている型である．添付ファイルや本文を含んでいるため，取得するメール情報が膨大になることが考えられる．このため，まず Mail_info 型で一覧を取得してから Message-ID を指定して取得したり，検索を行い件数を絞ってから取得する場合にこの Mail 型を使用する．

表 4.3 に各型の詳細を示している．また，存在する ML の数など ML やメールの統計情報を追加している．

表 4.2 各クラスの持つ情報一覧

操作対象	情報の名前	型	説明	取扱い
ML サーバ	mailinglists	-	ML の一覧	○
	node_name	string	ML の名前	○
ML	id	integer	ML の id	×
	node_name	string	ML の名前	○
	title	string	タイトル	○
	member	string	ML に属しているメンバ	○
	project	string	プロジェクト名	○
	description	string	ML の説明	○
メンバ	logname	string	メンバ名	○
	mail_cnt	integer	受信したメール数	○
	mailinglists	-	属している ML の一覧	○
	node_name	string	ML のノードネーム	○
ML アーカイブメール	to	string	受信者のメールアドレス	×
	from	string	送信者のメールアドレス	○
	subject	string	件名	○
	cc	string	Cc された受信者のメールアドレス	×
	msgid	string	Message-ID	○
	date	datetime	送信日時	○
	depth	integer	スレッドの深さ	×
	in_reply_to	string	返信元メールの Message-ID	○
	attachment	-	添付ファイル	○
	mailinglist_id	integer	メーリングリスト ID	×
	body	string	本文	○
	name	string	添付ファイルの名前	○
	content.type	string	添付ファイルの種類	○
	content.filename	string	添付ファイル元の名前	○
	content.path	string	添付ファイルへのパス	○

表 4.3 各データ型の詳細一覧

データ型	フィールド	型	説明
MailngLists 型	mailinglists	-	ML の一覧
	node_name	string	ML の名前
	ml_cnt	integer	存在する ML の数
MailingList 型	mailinglist	-	各個別レスポンスを含む
	node_name	string	ML の名前
	title	string	タイトル (例：GN グループの ML)
	member	string[]	name のリスト
	name	string	メンバのログイン名
	project	string	プロジェクト (例：GN)
	description	string	ML の説明
Member 型	name	string	メンバ名
	mail_cnt	integer	送信したメール数
	mailinglists	string[]	node_name のリスト
	node_name	string	ML の名前
Mail_info 型	Mail	-	各個別レスポンスを含む
	from	string	送信者のメールアドレス
	subject	string	件名
	msgid	string	Message-ID
	date	datetime	送信日時
Mail 型	Mail	-	各個別レスポンスを含む
	to	string	受信者のメールアドレス
	from	string	送信者のメールアドレス
	subect	string	件名
	msgid	string	Message-ID
	date	datetime	送信日時
	in_reply_to	string	返信元メールの Message-ID
	attachment	string[]	添付ファイルに関するリスト
	name	string	ファイル名
	content.type	string	ファイルの種類
	content.original_filename	string	元のファイル名
	content.path	string	ファイルのパス
	body	string	本文

第 5 章

システム構成

5.1 URLの基本構造

API の URL の基本構造を図 5.1 に示す.

本 API は，http://に続けてホスト名，操作対象，識別子の順に/で区切られて構成されている．その後に?記号で区切られてパラメータが続いている．それぞれの構成要素について以下に説明する.

(1) ホスト名

ホスト名とは DNS(Domain Name System) で名前が解決できるドメイン名か IP アドレスある.

(2) 操作対象

操作対象とは 4.1 節で示した 5 つの対象 (ML サーバ，ML，メンバ，ML アーカイブ，メール) を表すものである.

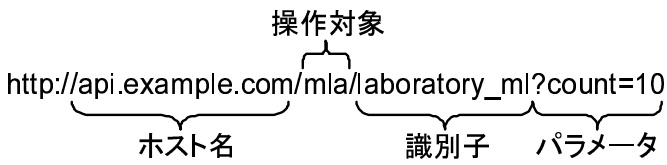

図 5.1 URL の基本構造

(3) 識別子

識別子とはMLサーバやML，MLアーカイブ，メールを操作対象とした場合はMLの名前，メンバを操作対象とした場合はメンバ名，といったように情報を取得する対象を区別するためのものである．

(4) パラメータ

パラメータは名前=値形式で指定できる．複数続ける際は&記号で区切り，別のパラメータを指定する．パラメータの種類については5.2節で示している．

URLに操作名を含めるべきではないため，含めていない．各操作を行うにはリクエストの際適したHTTPメソッド (GET，POST，PUT，DELETE) を指定する．

基本構造に照らして図5.1に示したURLを解釈すると,「MLアーカイブを操作対象とし，laboratory_mlというMLにアクセスする」URLである．このURLにGETメソッドを指定してリクエストを送るとメールを10件取得できる．

5.2 APIの操作一覧

表5.1に，操作対象，操作，パラメータ，使用するHTTPメソッドの一覧を操作対象毎にまとめている．取り扱うデータ型は操作対象と対応しているため，表には記載していない．

5.3 ステータスコード

存在しないAPIを実行しようとしたり，存在しないユーザを引数で指定してAPIを実行しようとすると，404 Not Foundが返る．認証が通っていない場合は，401 Unauthorizedが返る．返却するステータスコードの一覧を表5.2に示している．エラーの場合は，エラーであることを示すXMLレスポンスを返す．

表 5.1 操作対象毎の操作一覧

操作対象	操作	パラメータ	値	説明	メソッド
ML サーバ (mls)	Read	query	string	検索クエリ	GET
ML (ml)	Read	-	-	-	GET
	Create	title	string	タイトル	PUT
		project	string	プロジェクト	
		description	string	ML の説明	
	Update	title	string	タイトル	PUT
		project	string	プロジェクト	
		descriotion	string	ML の説明	
	Delete	-	-	-	DELETE
メンバ (member)	Read	-	-	-	GET
ML アーカイブ (mla)	Read	count	integer	取得するメール数	GET
		query	string	検索クエリ	
		logname	string	送信者のメンバ名	
		date	datetime	送信日時	
メール (mail)	Read	count	integer	取得するメール数	GET
		msgid	string	取得するメールの Message-ID	

表 5.2 返却するステータスコード

Code	状態	説明
200	OK	成功
400	Bad Request	無効な要求
401	Unauthorized	認証失敗
404	Not Found	指定されたリソースが存在しない

第 6 章

APIの使用例

6.1　実装環境

APIのプロトタイプとして，MLアーカイブからメール情報を取得する機能の実装を行った．実装を行った環境を表6.1に示す．

表　6.1　実装環境

OS	Debian 2.6.32-5-686
使用言語	Ruby 1.8.7
Webアプリケーションフレームワーク	Ruby on Rails 3.0.0

6.2　使用例

5章を基に,「議事録という文字列を検索してメール情報を取得するAPI」の使用方法について述べる．

まず，議事録という文字列を検索してメール情報を取得する場合，APIのURLは以下のようになる．複数のメールを検索するため，操作対象はMLアーカイブとなる．識別子はlaboratory_mlという名前のMLである．

```
http://api.example.com/mla/laboratory_ml?query=議事録
```

```
GET /mla/laboratory_ml?query=議事録 HTTP/1.1
Host: api.example.com
```

図 6.1 リクエストの例

```
HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<Mail_info>
<Mail>
  <from>taro@example.com</from>
  <subject>第 xx 回打合せ議事録</subject>
  <msgid>
    <20110208.131209.2152308882914694694.taro@example.com>
  </msgid>
  <date>2011-02-08 22:12:09 +0900</date>
</Mail>
<Mail>
...
</Mail>
...
</Mail_info>
```

図 6.2 レスポンスの例

このURLに図6.1のようなリクエストを送信する．すると，図6.2のようなレスポンスが返却される． このレスポンスから議事録という文字列が含まれたメールの一覧が取得できる．また，本文を参照したい際は，操作対象をメールとしMessage-IDを指定してAPIを利用すれば本文を取得できる．

第 7 章

おわりに

本論文では，ML 内で閉じてしまっている ML やアーカイブ情報を外部から利用するための外部連携用 API を設計した．まず，グループワークを支援する ML 機構の特徴，構成を示し，ML 機構の特徴の 1 つである，他 Web サービスとの連携用 API の問題点について述べた．次に，問題点を解消するために API への要求をまとめた．外部連携用 API を用いると，ML 機構が持っている ML やアーカイブの情報を外部から利用することができ，他 Web サービスとの連携が可能となる．多くの Web サービスと連携することを想定し，通信プロトコルには REST，データ形式には XML を選択し，API を設計した．また，ML メンバ情報だけではなく，ML 情報やアーカイブ情報も外部から利用できる設計となっている．

外部連携用 API は現在，所属する研究室内で研究・開発・実験的に運用されているグループウェアである LastNote との連携を行っており，一部の機能を Ruby on Rails [9] を用いて実装した．

今後の課題は，API を使用する際の認証方式についての考察，および残された機能の実装である．

謝辞

本研究を進めるにあたり，懇切丁寧なご指導をしていただきました乃村能成准教授に心より感謝の意を表します．また，研究活動において，数々のご指導やご助言を与えていただいた谷口秀夫教授，山内利宏准教授，後藤佑介助教に心から感謝申し上げます．

また，日頃の研究活動において，お世話になりました研究室の皆様に感謝いたします．

最後に，本研究を行うにあたり，経済的，精神的な支えとなった家族に感謝いたします．

参考文献

[1] 藤原 啓輔, 乃村 能成, 谷口 秀夫, “Web サービスとのマッシュアップを支援する メーリングリスト機構の提案, ” 情報処理学会研究報告, GN, グループワークとネットワークサービス研究会報告, vol.2009-GN-73, No.32(2009)

[2] R. Fielding, “Hypertext Transfer Protocol – HTTP/1.1,” IETF RFC2616, Jun 1999.

[3] Matt Harris, “Twitter API Documentation, ” http://apiwiki.twitter.com/w/page/22554679/Twitter-API-Documentation , Aug 2010.

[4] Yahoo! Inc, “Yahoo! Developer Network, ” http://developer.yahoo.com/

[5] Amazon Web Services LLC or its affiliates, “Amazon Web Services, ” http://aws.amazon.com/jp/

[6] 山本陽平, “Web を支える技術 HTTP, URI, HTML, そして REST, ” 株式会社技術評論社, May 2010.

[7] D. Crockford, “The application/json Media Type for JavaScript Object Notation (JSON) ,” IETF RFC4627, Jul 2006.

[8] Ruth Cassidy, “WS-I Completes Web Services Interoperability Standards Work,” http://www.ws-i.org/docs/press/pr_101110.pdf , Nov 2010.

[9] David Heinemeier Hansson, “Ruby on Rails,” http://rubyonrails.org/