2009－SE－163 (9)
2009／3／18

Webアプリケーション開発向けAOP機構の実装

外村慶二[†1] 成瀬龍人[†1] 塩塚大[†1]
白石卓也[†1] 鵜林尚靖[†1] 中島震[†2]

AOWPは，アクセス制御等のWeb固有の横断的関心事を容易にモジュール化することを目的としたAOP機構である．AOWPの特長は，Web固有のイベントを取り扱うポイントカット記述子と定義している点と，Webのコンセプトに即したアスペクトのインスタンス管理をサポートしている点である．本稿では，PHPを対象としたAOWPのプロトタイプ実装であるAOWP/PHPの実装について説明を行い，Webに特化したAOP機構の実現方法を明らかにする．また，AOWP/PHPの織り込み処理について，オープンソースのPukiWikiを用いた簡単な性能評価を行った．

Implementation of AOP mechanism for Web application development

Keiji Hokamura,[†1] Ryoto Naruse,[†1] Masaru Siozuka,[†1]
Takuya Shiraishi,[†1] Naoyasu Ubayashi[†1] and Shin Nakajima[†2]

We proposed AOWP for modularizing Web-specific crosscutting concerns such as access control, load balancing. AOWP provides pointcut designators which handle Web-specific events, and supports aspect instantiation mechanism which are suitable for Web concept. This paper explains implementation of AOWP/PHP, which is prototype implementation of AOWP. This paper also reports a small performance evaluation about weaving of AOWP/PHP.

1. はじめに

Webアプリケーションには，アクセス制御，ページ遷移制御，パフォーマンス・チューニング等の，Webアプリケーション固有の横断的関心事があり，それらはプログラムの保守性を低下させる要因となる為，モジュール化する事が望ましい．

横断的関心事をモジュール化する為の技術として，AOP (Aspect-oriented Programming)[6] がある．しかし，既存のAOP言語は，Webアプリケーションに固有なページ遷移イベントやページ遷移履歴等を直接取り扱う機構等を提供していない為，Webアプリケーションの横断的関心事をモジュールする上で，必ずしも適しているとは言えない．

そこで，我々は，Webアプリケーション開発向けのAOP言語として，AOWP[5),17)] を提案している．AOWPでは，HTTPリクエストに対する処理等，Web固有のイベントを取り扱う為のポイントカット記述子と，Webのコンセプトに即したアスペクトのインスタンス管理をサポートしている．また，AOWPのPHPを対象としたプロトタイプ実装として，AOWP/PHP[2)] を開発している．

本稿では，AOWP/PHPの実装における，Web固有のAOP機構の実現方法について説明する．また，オープンソースのWikiシステムであるPukiWikiを対象として，AOWP/PHPの簡単な性能評価を行う．

本稿の構成は，以下の通りである．2節で背景知識としてAOWPについて説明する．3節でAOWP/PHPの実装の概要を説明し，4節，5節で，Web固有のAOP機構の実現方法について詳しく説明する．6節では，AOWP/PHPが性能面で実用的であるかを確認する為の小規模な実験について報告する．そして，7節で今後の課題に付いて述べる．

2. AOWPの特長

Webアプリケーションを対象とした横断的関心事のモジュール化に関しては，Webサーバ上のプログラムに含まれるプログラムの振舞いに基づく横断的関心事の分離する為にAspectJ[14)] に基づくAOP言語を用いる手法[8),13)] や，複数の画面にまたがる横断的関心事の分離を目的としたフレームワークのWebJinn[7)]

†1 九州工業大学
Kyushu Institute of Technology
†2 国立情報学研究所
National Institute of Informatics

等がある．

一方，AOWP は，Web のプロトコルに基づくイベントに対して横断的になる関心事の分離を目的としている．本節では，まず AOWP について説明し，その PHP を対象としたプロトタイプの実装である AOWP/PHP について簡単に説明する．

2.1 AOWP

Web アプリケーションでは，HTTP プロトコルに基づいて，不特定多数のクライアントとデータ授受を伴うリクエスト/レスポンスの送受信を非同期に行うことで目的の処理を行っている．多くの Web アプリケーションは，目的に応じて，アプリケーション全体，もしくはユーザ単位で一連の HTTP リクエストを関連付けて処理する事で，様々なサービスを提供している．

AOWP では，一連の HTTP リクエスト処理に対して横断的関心事の分離を容易に行う事を目的としており，その為に，HTTP プロトコルが定める Web 固有のイベントを取り扱うポイントカット記述子と，Web に特化したアスペクトのインスタンス管理をサポートしている．

(1) Web 固有のポイントカット記述子

AOWP では，リクエスト URL，クライアントからサーバに送信されるフォームデータ，送信元の IP アドレス等のリクエストヘッダーの情報を元に，HTTP リクエストの受付イベントを選択する記述子を定義している．

また，ページ遷移履歴等の Web 固有のコンテキストに基づいてジョインポイントを選択する幾つかのポイントカット記述子を定義している．これらの記述子は，プログラムの振る舞いに基づくコンテキストを取り扱う cflow[4]，dflow[9]，tracecut[15]，tracematch[1] 等の ポイントカット記述子を参考に定義している．

(2) アスペクトのインスタンス管理

通常，アスペクトのインスタンス化はアスペクトの定義に従って暗黙の内に行われる．AOWP では，Web に特化したアスペクトのインスタンスの生存期間として，特定のユーザに対して 1 つのインスタンスを生成する方法と，Web アプリケーション全体で一つのインスタンスを生成する方法等をサポートしている．

2.2 PHP による実装

我々は，PHP を対象とした AOWP の実装である，AOWP/PHP を開発した．PHP を用いた理由は，スクリプト言語であることから，AspectJ が対象としている Java 等の言語と比較した時，実行時の動的織り込みの実現に適していると考えたからである．また，

図 1 ユーザ認証処理を行うアスペクト
Fig. 1 User authentication aspect

```
1  class Auth extends PerSessionAspect {
2
3    private $_loginID = null;
4
5    public function __construct() {
6      $advice = new BeforeAdvice();
7      $allReqPC = new RequestPointcut('.*');
8      $advice->setPointcut($allReqPC);
9      $advice->setAdviceBody('authChk');
10     $this->addAdvice($advice);
11   }
12
13   public function authChk($context) {
14     $id = $_POST['id'];
15     $pw = $_POST['password'];
16     if ($this->_loginID !== null)
17       return;
18     else if (LoginMgr::login($id, $pw))
19       $this->_loginID = $id;
20     else {
21       include 'login.php';
22       die();
23     }
24   }
25 }
```

PHP で実装されたオープンソースのプロダクトが多数あり，記述実験対象が豊富にある事も選定理由の 1 つである．

AOWP/PHP では，図 1 に示すように，PHP のクラスとしてアスペクトを記述する．AOWP/PHP が提供する，アスペクトの定義に利用するクラスの一覧を，表 1，表 2，表 3 に示す[*1]

図 1 の *Auth* は，Web サイト全体にユーザの識別 ID とパスワードを用いたユーザ認証処理を追加するアスペクトである．7 行目の *RequestPointcut* は，ページ遷移イベント (HTTP リクエストの受付) を選択するポイントカット記述子を表すクラスであり，コンストラクタで任意の文字列を表す '.*' をリクエスト URL として指定する事で，全てのページ遷移イベントに対してユーザ認証の処理を適用する事を表している．なお，*RequestPointcut* は，ページ遷移イベントを指定する要素として，フォームデータの値のパターン，HTTP リクエストヘッダの値のパターンについて，省略可能なコンストラクタの引き数として指定できるが，図 1 ではこれらの値を利用しない為省略している．

また，アドバイスとして，6-10 行目でポイントカットが選択するページ遷移イベントを処理する直前にユーザ認証の処理を行う *authChk* メソッドを実行す

*1 AOWP/PHP のバージョンアップに伴い 5)，17) と言語仕様が若干異なっている．

表 1 ポイントカット記述子を表すクラス (抜粋)
Table 1 Classes which represent pointcut designators

クラス名	説明
(1) プログラムの振る舞いを選択する記述子	
FuntionCall	関数呼び出しを選択する
FuntionExecution	関数の実行を選択する
GrobalVariableGet	グローバル変数の読み込みを選択する
GrobalVariableSet	スローバル変数の書き込みを選択する
ScriptExecution	スクリプトファイルの実行を選択する
(2) Web 固有のイベントを選択する記述子	
Request	ページ遷移イベントを選択する
SessionGet	セッションデータの読み込みイベントを選択する
(3) Web 固有のコンテキストを取り扱う記述子	
PageHistory	ページ遷移履歴に基づいてジョインポイントを選択する
SessionDataFlow *1	セッションのデータフローに基づいてジョインポイントを選択する

表 2 アスペクトのインスタンスの生存期間を指定する為の基底クラス (抜粋)
Table 2 Classes which specify a way of aspect instantiation

クラス名	説明
PerRequestTransactionAspect	一つの HTTP リクエストの処理に対して一つのインスタンス
PerSessionAspect	Web アプリケーションのクライアント毎に一つのインスタンス
PerAppicationAspect	Web アプリケーション全体で一つのインスタンス

る事を定義している．*Auth* アスペクトは，1 行目で Web アプリケーションのユーザ毎に一つインスタンス化されるアスペクトとして定義しており，*$_loginID* フィールドを用いてユーザ認証の結果を保持している．*authChk* メソッドでは，*$_loginID* が *null* の時 (現在のユーザが認証処理を行っていない時) に，ユーザが Web サーバに送信した *id* と *password* の値を用いて認証処理を行い，認証処理に成功したときは *$_loginID* に認証処理に使ったユーザの識別 ID を設定し，失敗したときは認証処理を行う為の Web ページを表示する処理を行っている．

*1 次期バージョンで実装予定

表 3 アドバイスの定義に用いる基底クラス
Table 3 Classes which represent advice

クラス名	説明
BeforeAdvice	選択したジョインポイントの前にアドバイスの処理を追加
AfterAdvice	選択したジョインポイントの後にアドバイスの処理を追加
AroundAdvice	選択したジョインポイントをアドバイスの処理で置き換える

3. 実装の概要

本節では，最初に AOWP/PHP の実装方針に付いて説明し，その後織り込み処理の全体像について説明する．

3.1 実装方針

ここでは，AOWP/PHP の，織り込みの実現方法と AOWP/PHP が取り扱うジョインポイントについて説明する．

織り込みの実現方法

織り込みの実現方法としては，(1) 実行時に全ての織り込み処理を行う，(2) 実行前に全てのジョインポイントにウィーバーの呼び出しのコードを追加し，実行時に織り込み処理を行う，(3) 実行前にジョインポイントの解析，ポイントカットの評価を行い，必要な箇所にアドバイスの呼び出しのコードを追加する，の 3 つの方法が考えられる．

AOWP/PHP では，(3) の織り込み方法を採用している．動的織り込みを実現する上では，(1)，(2) の手法が適しているが，AOWP/PHP では，関数呼び出し等のプログラムの振舞いに基づくジョインポイントもサポートしており，全てのジョインポイントを実行時に評価するのは性能面で負担が大きいと考え (1)，(2) の手法を採用しなかった．ただし，今後，AOWP/PHP の動的織り込みへの対応を予定しており，特定のジョインポイントに対して部分的に (1)，(2) の手法を採用する事を検討している．

ジョインポイント

AOWP/PHP では，ジョインポイントを PHP のプログラムの振舞いに基づいて定義している．例えば，メソッド呼び出し/実行等の AspectJ で定義されているジョインポイントの一部を，AOWP/PHP でもジョインポイントとして定義している．また，スクリプト言語である PHP に特徴的なものとして，関数呼び出し/実行，スクリプトファイルの読み込み/実行，グローバル変数への書き込み/読み込み等をジョインポイントととして定義している．

なお，PHP は，変数や関数の返り値等に対する型

図 2 織り込みで追加されるアドバイスの呼び出す為のコード例
Fig. 2 An example of a woven code snippet for invoking advice

```
1  require_once 'AOWP_CLS/ConfMgr.class.php';
2  ConfigurationMgr::includeAOWPClasses();
3  $aspect =
4    AspectInstanceMgr::getInstance('Auth');
5  $con = new Context(__FILE__);
6  $con->setAspect($aspect);
7  if($aspect->runtimeMatch(0, $con)){
8    $aspect->executeAdvice(0, $con);
9  }
10 AspectInstanceMgr::release($aspect);
11 /* 以降元のコードが続く */
```

指定をサポートしていない為，型情報に基づくポイントカットの定義に関して制限を設けている．例えば，PHP ではインスタンス変数の型が明示的に記載されていない為，特定クラスのインスタンスを用いたメソッド呼び出しを特定することは困難である．変数の型名を推測する為にプログラムのデータフロー解析を行う事が考えられるが，現バージョンの AOWP/PHP では，メソッド呼び出しを選択するポイントカットを定義する際，対象となるインスタンスの型名を禁止することで対応している．

Web 固有の AOP 機構の実現

AOWP/PHP では，Web 固有のイベントを取り扱うポイントカット記述子について，それらのイベントを上で述べたジョインポイントに対応付けて織り込みを行う事で実現している．また，アスペクトのインスタンス管理に関しては，Web に適した範囲でインスタンスを管理するファクトリクラスを設ける事で実現している．

3.2 織り込みの全体像

AOWP/PHP では，実行前に，ポイントカットが選択するジョインポイントに対応する PHP コードに，アドバイスの呼び出しを追加する事で織り込みを行っている．このコード変換は，Web アプリケーションを構成する全ての PHP スクリプトファイルを一旦 AST (Abstract Syntax Tree) に変換し，その AST をアスペクトの記述に基づいて変換し，最後に織り込み処理を行った AST からソースコードを生成する事で行っている．なお，アドバイスの呼び出しコードは，実行時にしか判断できないポイントカットの定義について，それを評価する条件文を伴う形で追加される．

織り込みで追加するアドバイスの呼び出しコードの例として，先に示した図 1 の *Auth* アスペクトに対応するコードを図 2 に示す．1，2 行目は，AOWP/PHP の実行に必要な PHP のスクリプトファイルを読み込む処理である．また，3，10 行目は，*AspectInstanceMgr* クラスをファクトリクラスとして利用し，*Auth* アスペクトのインスタンス化，及びインスタンスの解放の処理を行っている．5，6 行目は，織り込まれるジョインポイントに関するコンテキスト情報を作成している．そして，7 行目で，*Auth* アスペクトのポイントカットが指定する実行時のアスペクトの適用条件を評価し，その条件が満たされる時に 8 行目のアドバイスの呼び出しが実行される．なお，このコードは，*Auth* アスペクトのアドバイスは，*BeforeAdvice* として定義されている為，対象のジョインポイントに対応する PHP コードの前に追加される．

我々は，織り込みの処理系自体も，PHP を用いて実装した．なお，コード解析の処理に関しては，オープンソースの PHP ライブラリを整理している PEAR プロジェクトの PHP_Parser[11] パッケージを拡張する事で実装した．

4. Web 固有のポイントカット記述子の実現

本節では，表 1 の (2)，(3) に示す Web 固有のポイントカット記述子の実現方法について述べる．まず，(2) の Web 固有のイベントを直接選択するポイントカット記述子について説明し，続いて (3) の Web 固有のコンテキストを取り扱うポイントカット記述子について説明する．

4.1 Web 固有イベントの処理

ここでは，表 1 の (2) のポイントカット記述子の中から *RequestPointcut* を例として取り上げ，実行前の織り込み処理と，実行時の取り扱いについて順に説明する．

(1) 実行前の織り込み処理

PHP では，ユーザから送信された HTTP リクエストは，リクエスト URL で指定されるスクリプトファイルの実行により処理される．その為，織り込み処理では，*RequestPointcut* が選択する HTTP リクエストの受付イベントを，指定したリクエスト URL のパターンに合致するファイル名を持つスクリプトファイルの実行を表すジョインポイントと対応付けてコード変換を行っている．

例えば，図 1 の *Auth* アスペクトは，*RequestPointcut* のリクエスト URL パターンとして任意の文字列を表す '.*' を指定しており (7 行目)，対象の Web アプリケーションを構成する全てのスクリプトファイルに対してアドバイス呼び出しのコードが追加される．なお，*Auth* アスペクトのアドバイスは，ジョインポイントの実行前に適用される *BeforeAdvice* として定

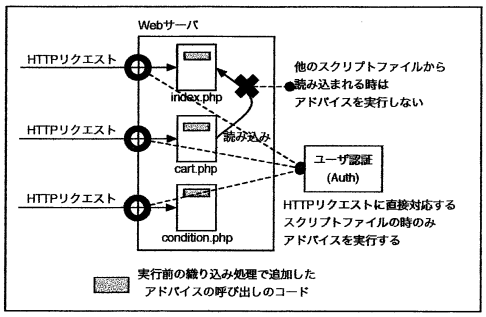

図 3 *RequestPointcut* の実行時の処理
Fig. 3 Behavior of *RequestPointcut* at runtime

義されている為，全てのスクリプトファイルの先頭に図 2 に示すコードが追加される．

(2) 実行時のポイントカットの評価

PHP では，他のスクリプトファイルを任意の位置で実行する事が出来る為，(1) の織り込み処理で見出したスクリプトファイルの実行箇所が，HTTP リクエストの受付イベントと必ず対応するとは限らない．その為，*RequestPointcut* によって織り込まれたアドバイスは，図 3 に示すように，そのスクリプトファイルの実行が HTTP リクエストに対して最初に実行されているかを実行時に評価している．また，フォームデータと HTTP リクエストのヘッダ情報が *RequestPointcut* が指定するパターンをみたすかについても，同様に実行時に評価を行う．

4.2 コンテキスト情報の取り扱い

ここでは，表 1 の (3) の中から *PageHistoryPointcut* を取り上げ，最初にそれを用いた記述例に付いて説明し，その実現方法に付いて説明する．

(1) PageHistoryPointcut を用いた記述例

図 4 に，*PageHistoryPointcut* を用いた *ConOfUse* アスペクト の記述例を示す．*ConOfUse* アスペクトは，利用規約ページを一度も閲覧していないユーザの全てのページ遷移イベントに対して，利用規約ページを表示する処理を適用する処理を実装している．

ConOfUse のポイントカットは，利用規約ページ以外へのページ遷移イベントを選択する *$otherReq* (8, 9 行目) と，利用規約ページの閲覧を行っていないときの全てのジョインポイントを選択する *$conHist* (11-14 行目) の 2 つポイントカットを組み合わせる事で定義している．

$otherReq は，8 行目で，利用規約ページと対応するリクエスト URL を表すクラス定数 *CURL* を用いて，利用規約ページへのページ遷移イベントを選択するポイントカットとして定義されている．9 行目の *Not* メソッドは，そのポイントカット記述子が取り扱う全てのジョインポイントを全体としたときの (ここでは *RequestPointcut* が取り扱う全てのページ遷移イベント)，元々のポイントカットが選択していたジョインポイントの補集合を選択するようにポイントカットに指示するものであり，ここでは，*$otherReq* を，利用規約ページ以外のページへの遷移イベントを選択するポイントカットとして定義している．

$conHist は，11 行目で，指定したページ遷移が現在アクセスしているユーザのページ遷移履歴に存在するときの全てのジョインポイントを選択する *PageHistoryPointcut* を用いて定義されている．13 行目の *addRequest* は，引き数の *RequestPointcut* が選択するページ遷移イベントを評価する履歴として指定するメソッドであり，ここでは，12 行目で定義した利用規約ページへの遷移イベントを *$conHist* で取り扱う履歴としている．また，14 行目で，*$conHist* が利用規約ページへの遷移履歴が存在しないときのジョインポイントを選択する事を指定している．

16 行目の *addAnd* メソッドは，呼び出されるインスタンスのポイントカットに対して，元々選択していたジョインポイントと，引き数で指定されたポイントカットが選択するジョインポイントの積集合を選択するように指示するものである．ここでは，*$otherReq* と *$conHist* の選択するジョインポイントの積集合をとることで，利用規約ページを遷移していないユーザの，利用規約ページ以外への全てのページ遷移イベントを選択するポイントカットを定義している．

(2) ページ遷移履歴の監視アスペクトの生成

ページ遷移イベント (HTTP リクエストの受付けイベント) は，4.1 で説明した通り，スクリプトファイルの実行のジョインポイントに対応づける事ができる．そこで，*PageHistoryPointcut* では，ページ遷移イベントに対応する全てのスクリプトファイルの実行に対して，ページ遷移履歴を管理するアドバイスを織り込む事で，ページ遷移履歴の取り扱いを実現している．

例として，図 4 の 11-13 行目のポイントカット記述に基づいて生成された，ページ遷移履歴の管理を行うアスペクトを，図 5 に示す．2 行目の *RequestObserveAspect* は，*PerSessionAspect* を継承したページ遷移履歴を管理する為のアスペクトである．1，2 行目では，このアスペクトのサブクラスとして *498fe08bb2027_GeneratedAspect* の任意の名前を持つアスペクトを，*ConOfUse* アスペクトのポイントカットを実現する為に生成しており，6，7 行目では，利用

図 4　ページ遷移履歴に基づくアクセス制御を行うアスペクト
Fig. 4　Access control aspect based on page history

```
1  class ConOfUse extends Aspect {
2
3    const CURL = '/condition\.php';
4
5    public function __construct() {
6      $advice = new BeforeAdvice();
7
8      $otherReq = new RequestPointcut(CURL);
9      $otherReq->Not();
10
11     $conHist = new PageHistoryPointcut();
12     $conReq = new RequestPointcut(CURL);
13     $conHist->addRequest($conReq);
14     $conHist->Not();
15
16     $otherReq->addAnd($conHist);
17
18     $advice->setPointcut($otherReq);
19     $advice->setAdviceBody('toCon');
20     $this->addAdvice($advice);
21   }
22
23   public function toCon($context) {
24     header('Location: ' . CURL);
25     exit();
26   }
27 }
```

図 5　織り込み時に自動生成されるページ遷移履歴を管理する為のアスペクト
Fig. 5　A automatically generated aspect for managing page transition history

```
1  class 498fe08bb2027_GeneratedAspect
2      extends RequestObserveAspect {
3
4    public function __construct() {
5      parent::__construct();
6      $this->_observeRequestArray[] =
7        new RequestPointcut("/condition\.php");
8    }
9  }
```

規約ページへの遷移を評価するページ遷移履歴の内容として定義している．

(3) 実行時のポイントカットの評価

生成されたアスペクトの名前は，図 4 の 11-13 行目の *$conHist* ポイントカットと関連付けて管理され，実行時に，*498fe08bb2027_GeneratedAspect* を用いてページ遷移履歴を評価を行い，アドバイスの実行を決定している．図 6 に，*ConOfUse* と *498fe08bb2027_GeneratedAspect* の実行時の振る舞いを示す．なお，例えば，*ConOfUse* の場合，利用規約ページ (*confition.php*) への遷移のみを監視するだけで良いように，定義されたページ遷移履歴のパターンによっては，生成する監視アスペクトのポイントカットを最適化できるが，これについては今後の課題である．

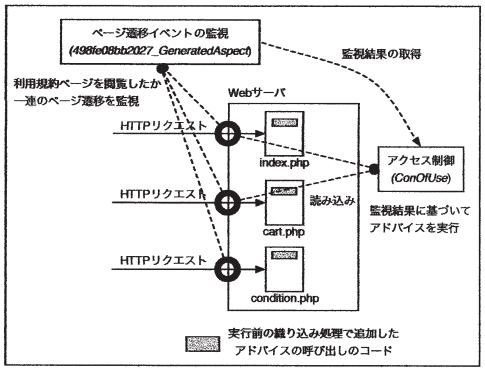

図 6　*PageHistoryPointcut* の実行時の処理
Fig. 6　Behavior of *PageHistoryPointcut* at runtime

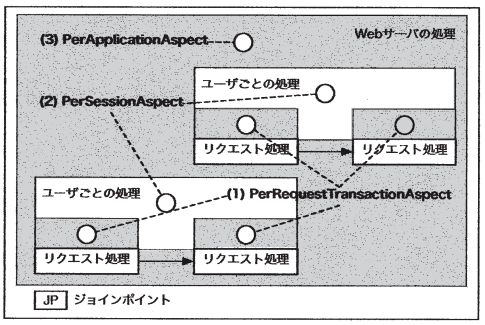

図 7　Web に適したアスペクトインスタンスの生存期間
Fig. 7　Web-specific aspect instance life cycle

5. アスペクトのインスタンス管理

AOWP/PHP では，アスペクトのファクトリクラスである *AspectInstanceMgr* で Web 固有のアスペクトのインスタンス管理を実現している．本節では，図 7 に示すインスタンス管理を例として取り上げ，これらの実現方法に付いて順に説明する．

(1) PerRequestTransactionPointAspect

PerRequestTransactionPointAspect を用いて定義したアスペクトは，PHP は，通常，1 つのリクエストに対して 1 つのプロセスが立ち上がる為，*AspectInstanceMgr* のクラス変数 (*static* 変数) の中で 1 つのインスタンスとなるように管理している．このアスペクトは，AspectJ の *issingleton* を指定したアスペクトと同様のインスタンスの生存期間を持つ．

(2) PerSessionPointAspect

PerSessionPointAspect を用いて定義したアスペクトは，HTTP で定義されるクッキーを用いて全てのユーザに固有の識別子を設定し，その識別子と関連付けて

アスペクトのインスタンスを管理することで実現している．クッキーとは，ユーザ側で一定の時間，特定のWebサーバと関連付けたデータを管理する事で，Webサーバと特定のユーザとの間で名前と値の対になったデータを共有する為のメカニズムであり，多くのWebアプリケーション用のプログラミング言語で，ユーザセッション機能を実装する為に利用されている．

(3) PerApplicationPointAspect

PerApplicationPointAspect を用いて定義したアスペクトは，1つのインスタンスを直列化しファイルに保存して，Webアプリケーション全体で共有する事により実現している．*AspectInstanceMgr* では，このアスペクトのインスタンスの直列化に関する処理と，直列化したインスタンスの読み込みに関する排他制御を行っている．

6. 性能評価

我々は，AOWP/PHP の織り込み速度に関して，オープンソースのWikiシステムであるPukiWiki[12]を対象とした性能評価を行った．織り込み処理は，PukiWiki の HTML (Web ページ) の生成に関わるPHPを対象として行い[*1]，スクリプトファイルの数は107個，全体のコード行数は約21000行であった．計測には，CPUが Intel Core 2 Duo 2.4GHz，メモリが4GBの Mac Book Pro を使用して行った．織り込み処理を適用するアスペクトについては，図1に示す *Auth* アスペクトを用いて行った．

織り込み処理に費やした時間は，約77.46秒となり，PukiWiki規模のWebアプリケーションに対しては，実用上許容できる時間で織り込み処理が行える事を確認できた．具体的には，スクリプトファイルの AST への変換に 約36秒，アドバイスの織り込みに約14秒，織り込み後の AST をソースコードに変換に約27秒 の時間を費やしている．

最も時間を消費している織り込み処理の行程は，スクリプトファイルから AST を生成する行程である．この行程に関しては，一度 AST の生成を行えば，対象のアプリケーションに変更がない限り再度行う必要がない．そこで，アスペクトを記述する際のテスト時の時間を軽減する為に，生成した AST をキャッシュしてすることで織り込み時間を短縮するようにしている．

また，上の測定結果から，さらに大規模な Web アプリケーションに対して AOWP/PHP を適用する際に，織り込み処理の時間が非常に大きくなる事が予測できる．例えば，オープンソースのブログシステムである WordPress[16] は，全体のコード行数が約 113,000 行あり，これらのアプリケーションに対して AOWP/PHP を適用する上で，織り込み処理の最適化を行う必要があると考えている．

[*1] Web 画面のデザインを指定する CSS ファイルの動的生成を行う，*.css.php* の拡張子を持つ PHP スクリプトを織り込み対象から除外した．

7. 今後の課題

今後の課題は，AOWP/PHP に動的織り込みの機能を追加する事である．現在の AOWP/PHP の実装では，3節で述べたように，実行前の織り込み処理でどのジョインポイントにどのアスペクトを適用するかを決定している為，実行時にアスペクトの追加や削除を行う事ができない．

動的織り込みに関しては，JAC[10]，GAP[3]，AOJS[18] 等がある．特に GAP は，PHP を対象とした 3.1 節で述べた (1) の 織り込み方法を採用した AOP 実装であり，PHP の処理系である Zend Engine を拡張することで，実行時にスクリプトファイルが処理系に読み込まれたタイミングで織り込み処理を行っている．

AOWP/PHP では，AspectJ に基づくジョインポイントを全て取り扱っている為，全体的に動的織り込み方法を採用するのは，性能上現実的ではないと考えている．そこで，現在，Web アプリケーションで頻繁にアスペクトの変更が行われるジョインポイントに範囲を限定した形で，動的織り込みを実現する事を検討している．

参 考 文 献

1) Allan, C., Avgustinov, P., Christensen, A.S., Hendren, L., Kuzins, S., Lhoták, O., de Moor, O., Sereni, D., Sittampalam, G. and Tibble, J.: Adding Trace Matching with Free Variables to AspectJ, *Proceedings of the 20th annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '05)*, New York, NY, USA, ACM, pp.345–364 (2005).
2) AOWP (Aspect-Oriented Web Programming): `http://sourceforge.jp/projects/aowp/`.
3) Bergmann, S. and Kniesel, G.: GAP: Generic Aspects for PHP, *Third European Workshop on Aspects in Software* (2006).
4) Douence, R. and Teboul, L.: A Pointcut Language for Control-Flow, *Proceedings of the Generative Programming and Component Engineering (GPCE '04)*, pp.95–114 (2004).
5) Hokamura, K., Ubayashi, N., Nakajima, S. and Iwai, A.: Aspect-Oriented Programming

for Web Controller Layer, *Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific*, pp.529–536 (2008).
6) Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J.-M. and Irwin, J.: Aspect-Oriented Programming, *Proceeding of European Conference on Object-Oriented Programming (ECOOP '97)*, pp.220–242 (1997).
7) Kojarski, S. and Lorenz, D.H.: Domain driven web development with WebJinn, *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA '03)*, New York, NY, USA, ACM Press, pp.53–65 (2003).
8) Lemos, O. A. L., Junqueira, D. C., Silva, M. A. G., de Mattos Fortes, R. P. and Stamey, J.: Using Aspect-oriented PHP to Implement Crosscutting Concerns in a Collaborative Web System, *Proceedings of the 24th Annual Conference on Design of Communication (SIGDOC '06)*, New York, NY, USA, pp.134–141 (2006).
9) Masuhara, H. and Kawauchi, K.: Dataflow Pointcut in Aspect-Oriented Programming, *Proceedings of the First Asian Symposium on Programming Languages and Systems (APLAS '03)*, pp.105–121 (2003).
10) Pawlak, R., Seinturier, L., Duchien, L. and Florin, G.: JAC: A Flexible Solution for Aspect-Oriented Programming in Java, pp.1–24 (2001).
11) PEAR PHP_Parser: `http://pear.php.net/package/PHP_Parser/`.
12) PukiWiki: `http://pukiwiki.sourceforge.jp/`.
13) Stamey, J., Saunders, B. and Blanchard, S.: The Aspect-Oriented Web, *Proceedings of the 23rd Annual International Conference on Design of Communication (SIGDOC '05)*, New York, NY, USA, ACM Press, pp.89–95 (2005).
14) The Eclipse Foundation: `http://www.eclipse.org/aspectj/`.
15) Walker, R. J. and Viggers, K.: Implementing Protocols via Declarative Event Patterns, *ACM SIGSOFT Software Engineering Notes*, Vol.29, No.6, pp.159–169 (2004).
16) WordPress: `http://wordpress.org/`.
17) 外村慶二, 鵜林尚靖, 中島震: AOWP: Web アプリケーション開発向け AOP 機構, SES 2008: ソフトウェアエンジニアリングシンポジウム 2008, pp.181–182 (2008).
18) 久保淳人, 水町友彦, 鷲崎弘宜, 深澤良彰, 鹿糠秀行, 小高敏裕, 杉本信秀, 永井洋一, 山本里枝子, 吉岡信和: AOJS: アスペクトを完全分離記述可能な JavaScript アスペクト指向プログラミング・フレームワーク, FOSE '08: ソフトウェア工学の基礎ワークショップ, pp.145–154 (2008).