平成 26 年度 公立はこだて未来大学卒業論文

コード生成を用いたフレームワーク向け Web アプリケーション開発支援ツールの作成

京谷 和明

情報アーキテクチャ学科　　1011129

指導教員　　伊藤 恵

提出日 2015 年 1 月 30 日

Making a Support Tool of Web Application Development for Framework Using Code Generation

by

Kazuaki Kyoya

BA Thesis at Future University Hakodate, 2015

Advisor: Assoc. Prof. Kei Ito

Department of Media Architecture
Future University Hakodate
January 30, 2015

Abstract– Recent years, Web application development is more needed the short delivery time and high quality. Therefore, various frameworks are used in Web application development. But if developers use such framework, they often write similar code, so development working with framework is inefficient. In addition, definition and rules of frameworks are different. So learning period about the framework is often lengthy. And, if engineers who do not know about the framework use the framework, development processes to touch up and modify may will be longer. On the other hand, Web application development use various auto code generation. But for using code generation tools, developers must deeply understand about the model for the tools, and they must write details in the model, and learning period about the model is often lengthy. So this study made a semi-automatic code generating tool that the conforming CakePHP architecture. The input models of this tool are ER model and screen transition that diagrams are typical created in Web application development. I applied this tool to a management system of appointment of interview. As a result, 80% source codes including redundant parts were generated. And, this tool can flexibly generate more amount of source code than other auto code generation tools for CakePHP. By this tool, developers rapidly develop Web application and the generated codes are high quality based on the framework. So, this tool is thought as a utility support tool of Web application development, although this tool has several research tasks. In addition, this tool may be able to solve various other research tasks by develop this tool. So, I will solve these research tasks, and I want to improve performance of this tool.

Keywords: Web Application Development, Code Generation Tool, Framework, Model, CakePHP

概 要: 近年，Web アプリケーション開発では，より一層の短納期化，高品質化が求められている．そのため，Web アプリケーション開発には様々なフレームワークが用いられていることが多い．しかし，フレームワークを用いる場合，類似したコードを記述することも多く，開発作業が非効率となる．その上フレームワークごとに定義や作法が異なるため，そのことを学習するのに時間がかかってしまい，その定義や作法を知らないまま開発を進めてしまうと後々加筆修正を行う際に余計に時間がかかってしまうことがある．一方で，効率化のため自動生成ツールを用いる場面も増えてきている．しかし，自動生成ツールを用いる場合は，生成元のモデルについて深く理解し，細部まで記述しなければならないことが多く，その自動生成ツールを扱うための学習期間が長くなってしまうことがある．そこで本研究では，Web アプリケーションの作成を行う際に，一般的に作成され，理解のしやすいモデルである画面遷移図と ER 図より，CakePHP のアーキテクチャに則ったコードを半自動で生成することのできるツールの作成を行った．作成したツールを面談予約管理システムに適用したところ，システム全体の 8 割程度を生成することができ，冗長な部分を生成することができた．CakePHP の機能を自動生成できる他のツールと比較しても，本ツールのほうが生成率が高く，柔軟に自動生成を行うことができていた．そのため，機敏に開発を行うことができ，フレームワークによる質の良いコードが生成できたと思われる．開発支援ツールとして有用性のあるツールを作成することができたが，いくつか課題が残っていることも判明した．また，本ツールを発展させることにより，様々な他の課題も解決できるのではないかと思われる．そのため，今後はその課題を解決し，ツールの性能を向上させていきたい．

キーワード: Web アプリケーション開発, 自動生成ツール, フレームワーク, モデル, CakePHP

目 次

第1章　序論

Web アプリケーション開発では，より一層の短納期化，高品質化が求められている．この目的を達成するために，様々な Web アプリケーションのフレームワークや，ソースコードの自動生成ツールが導入されている．しかし，このフレームワークや自動生成ツールには様々な問題を抱えており，導入がうまく行えていない事例が多々ある．本研究ではそのことに着目し，課題を上げ，その課題を解決していくことにした．

1.1　背景

Web アプリケーションの開発では様々なフレームワークが用いられることが多い．フレームワークとはソフトウェアを開発する際の枠組みとなるものであり，頻繁に作成される機能やコードをまとめて提供することで，システム開発の支援をしている．Web アプリケーションのフレームワークとして，例えば Ruby のフレームワークである Ruby on Rails，PHP のフレームワークである CakePHP などが挙げられる．Web アプリケーションの開発にフレームワークを用いるメリットとしては一般的に開発工数の削減と品質の均質化などが挙げられる．デメリットとしては，フレームワークごとに定義や作法が異なり，そのフレームワークを扱うのに学習期間が長くなってしまい，学習したとしても他のフレームワークでは扱うことができないことがある．また，そのフレームワークの作法を知らないまま開発を進めてしまうと，後々デバッグやテストを行う際に余計な時間がかかってしまうことが多い，といったことが挙げられる．

また，Web アプリケーションの開発には自動生成ツールを導入する場面が増えてきている．自動生成の例としては，各クラスの名前や属性や関数などを図で表すクラス図やオブジェクト間の処理の流れを図で表すシーケンス図といった，統一モデリング言語（UML）や，設計書から直接自動生成を行う場合がある．或いは独自規格を作成し，それに則って画面レイアウトや機能を決定し，そこからソースコードを自動で生成する製品も存在する．自動生成ツールを導入するメリットとしては，開発工数の削減と品質の均質化，設計したドキュメントや図を最大限に活用できるといったことが挙げられる．同様に，デメリットとしてはモデルから自動生成を行う場合は設計の段階で非常に細かいところまで機能の洗い出しをして設計しなければならないことが多く，独自規格の自動生成ツールを扱う場合は学習期間が長くなってしまうことが多いことである．

1.2 対象とする領域

ここではWebアプリケーション開発プロセスにおける，設計と実装に関する部分について研究する．設計工程ではモデリングを行い，実装ではフレームワークを導入する．設計段階で作成したモデルファイルに自動生成ツールを導入することで，一部のソースコードの生成を行い，実装段階における作業を減らすことを行う．

1.3 研究目標

本研究では，学習期間があまり長くならず，設計も簡単に行うことができ，さらにコメントも充実させた雛形を生成することで機敏に開発を行うことのできるツールの作成を行うことを目的とする．

今回はソースコードの自動生成に関する研究目標でよく挙げられる，自動生成率を高めることで，ソフトウェアの完全な自動生成を目指すものではなく，設計段階で必要とするモデルを簡単なものにし，そのモデルに追記する情報を少なくすることで，開発工数を確実に減らすことのできるツールの作成を目指す．なお，今回はレイアウトの自動生成に関する部分は考慮せず，機能的な部分の自動生成を行えるようにする．

第2章　関連研究

2.1　フレームワーク

本研究で扱う題材として，フレームワークがある．フレームワークは「同じ目的のアプリケーションなら，同じような書き方をする」という考えに基づいて,「同じような書き方」のところはすでに実働可能な構造や形式で提供され，使用者は自分の固有の目的に合わせて一部だけを書き換えたり書き足したりすればよい [1]，という考え方に基づいたアプリケーションの作成方式である．フレームワークを導入することにより，複雑な部分，冗長な部分はフレームワークが行ってくれ，自分は目的に合わせて一部だけに集中して実装を行うことができる．

2.1.1　Ruby on Rails

本研究で扱うフレームワークである CakePHP が，大いな影響を受けたのが Ruby on Rails である．Ruby on Rails の登場以前は，Web アプリケーションの開発というものは，データベースやサーバの複雑な知識といった，様々な知識を持たないと作成が難しいものであった．また，データベースの連携も自分で行わないといけないため，コードの中に SQL を書かなければならないことが多く，コードが肥大化し，非常に可読性が低いものとなることが多かった．2004 年の夏頃に登場した Ruby on Rails はこれらのことを解決すべく作成されたものであり，データベースとの連携を自動で行い，MVC アーキテクチャに則ってコーディングを行うことで，Web アプリケーション開発の着手が容易になり，コードの可読性も向上した．Rails の揺るぎない基盤として，Don't Repeat Yourself(繰り返しを避けよ) と Convention over Configuration(設定より規約)[2] がある．Rails のアプリケーションは 1 箇所にそのことを記述すれば，大抵のことは MVC アーキテクチャが案内してくれ，そこから先に進むことで，機能が実現できる．また，Rails の規約に従ってコーディングを行うことで，Rails が適切なデフォルトの動作を用意し，そこからすぐに目的の機能が実現できる．CakePHP はこの Rails の流れを汲んだものであり，思想や設計手法などの影響を大いに受けている．

2.1.2　CakePHP

本研究では Web アプリケーション開発フレームワークである，CakePHP 用のソースコードを生成する．CakePHP は PHP で記述されたオープンソースソフトウェアのフレームワークであり，公式サイト [3] からパッケージファイルをダウンロードすることで利用可能である．ダウンロードした圧縮ファイルを展開すると，図 2.1 のようなディレクトリ

構造となっている．

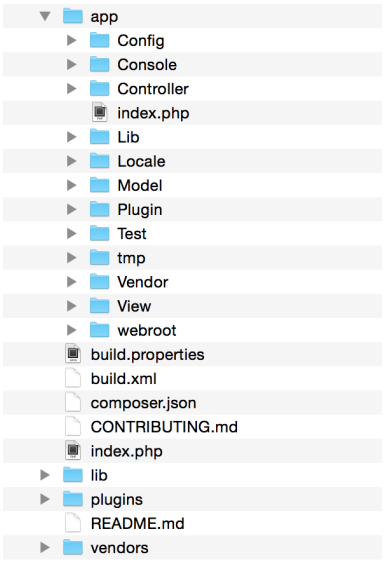

図 2.1: CakePHP のディレクトリ構造

CakePHP は他のライブラリをインストールする必要はなく，展開したフォルダを Web サーバ上にコピーするだけで，使えるようになる [4]．CakePHP には，様々なヘルパーや，コンポーネントといったものが用意されており，それらを利用することで，PHP のコードだけでなく，HTML や JavaScript といった言語のサポートも行い，スムーズなコーディングが行えるようになっている．また，有志が作成した CakePHP のコードをプラグインとして配布することで，他の人がそれを入手し，ソースコードを流用できるような仕組みにもなっている．さらに，CakePHP は多くのリレーショナルデータベース管理システムと連結することができ，MVC アーキテクチャに準拠している．MVC とは Model-View-Controller の頭文字をとったものであり，図 2.1 の app フォルダの直下にそれぞれのディレクトリが存在していることがわかる．図 2.2 は，CakePHP における MVC アーキテクチャの基本構造である．

Controller にシステムの機能の内部処理を記述し，主に View と Model にデータの受け渡しを行う．View では画面への出力法を記述し，PHP だけではなく，HTML や JavaScript, CSS といった言語も記述可能である．Model ではデータベース周りのことを記述し，データベースとの連携を行う．このように MVC アーキテクチャに則ってコーディングを行うことで，それぞれの処理を独立させ，コードの均質化を図ることができ，タスクの振り分けやコードリーディングがしやすくなる．

しかし，MVC アーキテクチャに反して内部処理を Model，View に記述してもフレームワーク側が柔軟に対応してしまい，仕様通りの挙動ができてしまうことがある．そのため，プログラマによって記述する場所が異なり，特に納期が短い中でシステム開発を行うと，コードに統一性が出てこなくなってしまうことが多い．

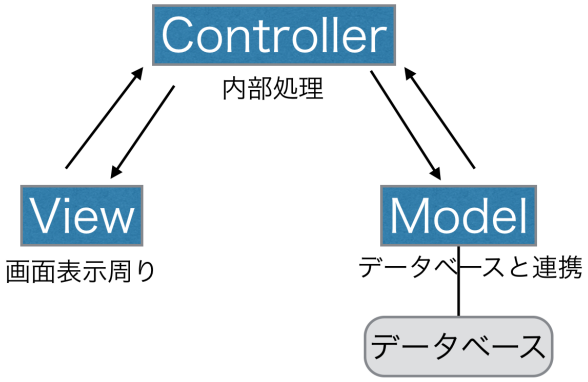

図 2.2: CakePHP における MVC アーキテクチャの基本構造

2.2 モデルからのコード生成

2.2.1 モデル駆動型アーキテクチャ

本研究の先行的な考え方として，2001 年に Object Management Group が発表した，モデル駆動型アーキテクチャ(MDA)[5] というものがある．モデル駆動型アーキテクチャとは，モデルはソフトウェアの開発と保守を行うための優れた基盤であるとし，UML といったモデリング言語を発展させるために考えられたものであり，さまざまな抽象化レベルでモデルを作成し，それらのモデルを繋ぎあわせて実装するという考え方である．このモデル駆動型アーキテクチャは，実装技術に依存しない標準仕様である PIM(Platform Independent Model:プラットフォーム独立モデル) を策定し，そこから各プラットフォームに依存した形式である PSM(Platform Specific Model:プラットフォーム依存モデル) に変換することで，様々な実装技術に対応でき，月日が経過して実装技術が変化しても開発資産の経年劣化を防ぐことができる [6].

2.2.2 Executable UML

本研究の先行研究としては，Sally Shlaer，Stephen J. Mellor らが提唱した Executable UML(xUML)[7] というものがある．Executable とは「実行可能な」という意味を表しており，UML を拡張することで，確実に実装のできるコードを記述できるモデルを作成できるようすることである．この Executable UML を導入することで，1 つの問題領域の完全なモデルを構築できるようになるとのことである．

これらモデル駆動型アーキテクチャや Executable UML の考え方を導入することにより，モデルで動作の確認を行うことができ，そのままソースコードの生成も行うことができると言われている．しかし，モデル駆動型アーキテクチャは元となる UML といったモデルが完全に標準化を行うことができず，近年，この考えは落ち着いている．Executable UML は Mellor により，2011 年に標準化されているが，まだまだ普及しているとは言い難い．

2.2.3 UML を入力とするソースコード自動生成ツールの開発

河村，浅見らの研究 [8] では Web アプリケーション開発における自動生成技術の適用に対して，クラス図とシーケンス図からフレームワーク用のコードを自動生成するツールの作成に取り組んでいる．この研究で作成されたツールを適用することで，9 割以上のソースコードの生成に成功している．しかし，クラス図とシーケンス図はフレームワークを扱う場合，ある程度そのフレームワークを使ってのシステム開発の経験が無いとどう作成すればよいのかわからないことがある．

2.2.4 モデル変換に基づくエンドユーザ主導の Web アプリケーション開発技法

八木，中所らの研究 [9] ではエンドユーザが主導して，Web アプリケーションを開発するために 3 つの自動生成ツールを作成している．3 つのツールを順に扱っていくことで，Web アプリケーションモデルから実際に稼働する Web アプリケーションへの自動生成を行えるとのことである．その 3 つのツールを適用することでシステム開発経験の浅いエンドユーザでも直感的に開発が行えるようにしている．この研究ではこのツールを使って汎用プログラミング言語を用いるよりも，容易に Web アプリケーションを作成することに成功している．

2.3 コード生成ツール

以下に関連する製品や，ソフトウェアを示す．

2.3.1 Web Performer

本研究の関連製品としては，キヤノンソフトウェア株式会社が作成した,「Web Performer」[10] が存在する．この製品は GUI 操作で要件定義情報から業務用の Web アプリケーションを 100 %自動生成できるツールであるとのことである．しかし，こちらは 1ヶ月弱の学習期間が必要であり，また製品の値段が 300 万円以上するとのことなので，小さなシステム開発会社にとっては敷居が高くなってしまう．

2.3.2 その他関連するツール

その他の関連するツールとしては，株式会社ジャスミンソフトが販売している「Wagby」[11]，南米ウルグアイの Artech 社が開発した「GeneXus」[12]，オープンソースソフトウェアである「blanco Framework」[13] 等が挙げられる．

いずれのツールも自動生成率は高く，Web アプリケーションの開発に適用することで機敏にシステム開発を進めることができると思われる．しかし，設計段階で必要となるモデルを細かく作成しなければならなく，学習期間も長くなってしまうツールが多く，これらのツールの導入に手間取ってしまうかもしれない．

第3章　アプローチ

本研究では，既存のツールと比較して，生成できるソースコードの量は少ないが，必要とするモデルを簡単にし，少ない手順で，CakePHP のアーキテクチャにしっかりと則った形でコードの自動生成を行うことで，コードに統一性を出し，導入をしやすくするツールの作成を目指す．本研究で扱うモデルは画面遷移図と ER 図である．これらのモデルを入力として，CakePHP のコードを生成する．

なお，今回はレイアウトに関する部分は考慮せず，主にシステムの機能的な部分の自動生成を行う．

3.1　ツール構成図

ツールは Java を使って作成する．画面遷移図と ER 図は UML を中心としたモデリングを行うことのできるツール astah*[14] を使って作成する．そして astah*の XML 出力機能を利用して XML ファイルを生成し，その XML ファイルを解析することで作成した画面遷移図と ER 図の XML ファイルから文字列を抽出し，抽出した文字列からディレクトリやファイルを生成し，そのファイルに CakePHP 用のコードを記述する．図 3.1 に本ツールの構成図を示す．

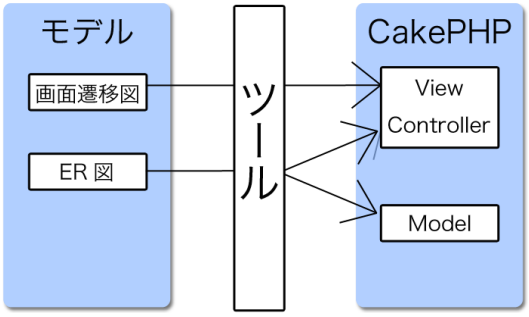

図 3.1: ツールの構成図

3.2　画面遷移図と ER 図

本研究ではモデルとして画面遷移図と ER 図を扱う．

画面遷移図とは画面の設計を行う際に作成されるものであり，各画面の機能や画面の名前，ボタン押したときや処理を行った際にどの画面に遷移するかが記述されている．

一方，ER 図は和名では実体関連図と呼ばれている．システムのデータを図に表すことで，データベースを抽象的に示したものである．本研究で扱うデータベースは第 3 正規形であり，ER 図は IDEF1X 表記のものを取り扱っていく．また，テーブル名や外部キーは実際にデータベースに登録する際の名称をそのまま ER 図に記述する．この名称は CakePHP の命名規約に従って，テーブル名は単数形で記述をし，外部キーは,「元となるテーブル名の単数形」＋「_id」と記述するものとする．

画面遷移図と ER 図はいずれも Web アプリケーションを開発する際に作成されることが多く，理解もしやすい．また，フレームワークを扱った経験が無くともある程度のデータベースとモデリングの知識があれば，作成が可能である．

本研究ではこれらのモデリングツールとして，UML を中心としたモデリングを行うことのできる astah*を扱う．画面遷移図は astah*のステートマシン図作成ツールを使い，ER 図は astah*の ER 図作成ツールを用いる．この astah*で作成した画面遷移図と ER 図のファイルを解析することでコードの自動生成を試みる．

3.3 生成手順

本研究では上記で述べたとおり，画面遷移図と ER 図から CakePHP のソースコードを半自動で生成する．図 3.2 は第 4 章のコード生成実験で扱う面談予約管理システムの ER 図の一部である．

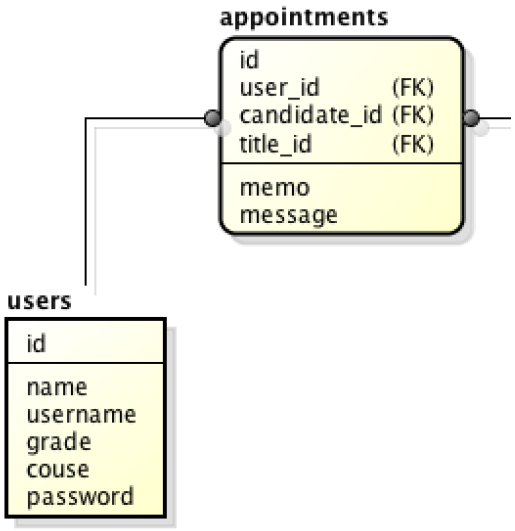

図 3.2: ER 図の一部

図 3.2 では各テーブルの名前とカラム名が記述されており，users テーブルと appointments テーブルが 1 対 0 以上の関係で表されている．主として Model 部分のコード生成のために，このような ER 図の各テーブル名やリレーションシップを参照する．Model に対するバリデーションに関しては，そのカラムが not null かどうか判別し，空の内容を許可するかしないかを生成する．バリデーションとは，例えばデータの追加を行う際に，入力

フォームにデータを入力するが，その入力されたものがありえないもの，ルールに従わないものだった場合に弾いてエラーメッセージを表示できるようにするものである．

続いては，画面遷移図より Controller と View を生成していく．図 3.3 に第 4 章のコード生成実験で扱う面談予約管理システムの画面遷移図の一部示す．

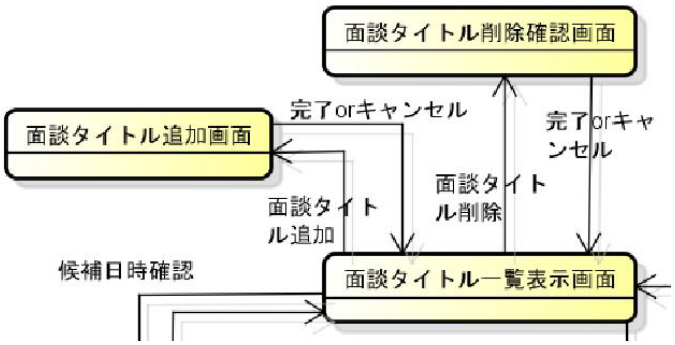

図 3.3: 画面遷移図の一部

図 3.3 では面談タイトル一覧表示画面で面談タイトル追加ボタンを押すと面談タイトル追加画面へと遷移することが示されている．図 3.3 で示されているように，このままでは各画面が ER 図のどのテーブルに値するのかが不明確なため，各画面のプロパティビューに予めテーブル名を記述しておく．図 3.4 にプロパティビューの例を示しておく．

図 3.4: プロパティビューの例

画面遷移図からは，各画面の名前や処理を行った際の画面の遷移先を参照する．

3.4 生成されるもの

上記で述べた生成手順を踏むことで，ER 図のテーブル名からは，各 Model のファイルが生成される．そのファイルにクラス名やメソッド名を記述し，バリデーションやアソシエーション等が記述される．アソシエーションとは，リレーショナルデータベースにおけるリレーションシップにあたるものであり，どの Model と何対何で繋がっているかを示すものである．

また，画面遷移図からは各 View，Controller のファイルが生成され，そのファイルの中に各機能の処理や画面を表示するためのコードが記述される．具体的な例としては，図 3.5 のように，画面の名前に「追加」とあったら View に追加画面を表示するための add ファイルを生成し，そこに入力フォームを表示して，ボタンが押されたら POST 送信を行うように記述する．また，Controller にも add メソッドを記述し，View から POST 送信された内容を，対応するテーブルにデータを追加する処理を記述する．

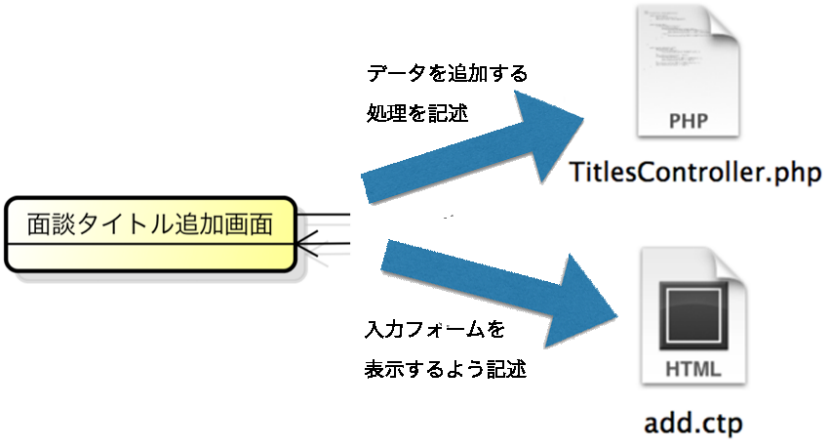

図 3.5: 追加機能の生成の例

また，ER 図からも情報を読み込み，カラム数やデータの型の判別を行う．このように ER 図からも情報を読み込むことで，入力フォームをいくつ生成すればよいのか，どのような入力フォームを生成すればよいのかを判別する．具体的にはデータの型が DATE，あるいは DATETIME だった場合は，図 3.8 のように，入力フォームはプルダウン形式で年月日を選択できるように自動生成を行う．データの型がそれ以外だった場合は，テキスト入力フォームを生成し，そこに記述を行えるようにする．また，図 3.7 のように，外部キーが存在した場合は，元となる Model と何対何の関係かを判別し，1 対 1，あるいは多対 1 だった場合はラジオボタンで選択できるようにし，1 対多，多対多の場合はチェックボックス形式で複数選択できるように自動生成を行う．　追加機能以外にも，画面遷移図に「削除」とあった場合，Controller に delete メソッドを記述し，テーブルから該当データを削除する機能を搭載する．他にも画面の名前に「一覧」と記述してあった場合はテーブルの全てのデータを一覧で表示する,「編集」と記述してあった場合は該当レコードを編集できる機能を搭載する，といったように，画面遷移図から文字列を読み取ることで，各機能を搭載する．様々なシステム開発で作成された画面遷移図を参照していくことで，どの文字

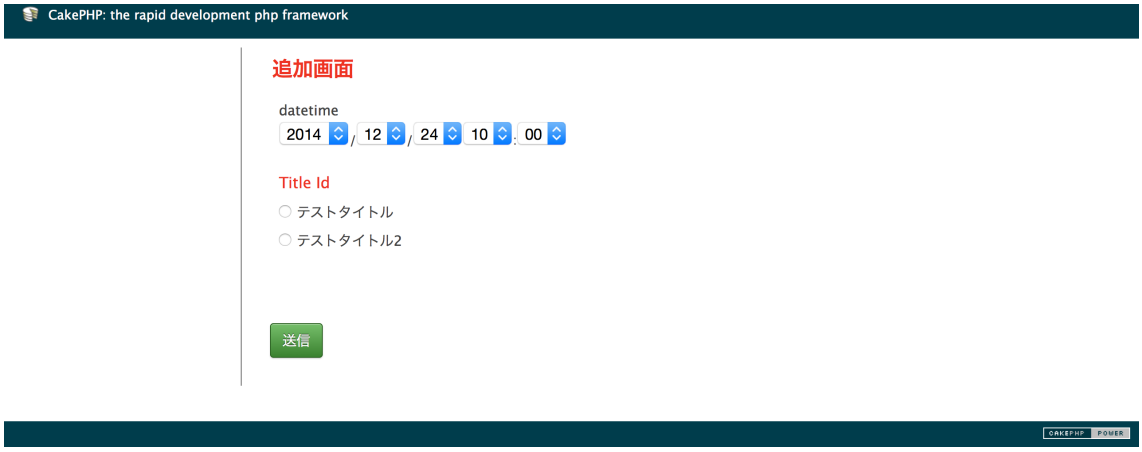

図 3.6: フォーム生成の例

列を抽出したらどの機能のソースコードを記述するのかを決定していく．現時点では Web アプリケーションの基本的な機能である「追加」,「削除」,「編集」,「一覧表示」,「履歴表示」,「詳細表示」,「ログイン」といった機能の自動生成を行う．

ソースコードを生成する際に，コメントも生成する．具体的には，例えば CakePHP の各命名規則に従ってソースコードが生成されたときは，そのすぐ下にコメントとして，その命名規則についての説明が軽く生成される．また，各種ヘルパーやコンポーネントを使った場合は，そのヘルパーやコンポーネント名を生成する．こうすることで，そのコードが何を表しているのかがわからない場合に，その名前ですぐに調べることができる．ヘルパーやコンポーネントはプログラマが加筆修正を行う可能性が高いと思われる場所には，そのことのコメントも生成する．他にも，Controller が Model や View に値を受け取ったり渡したりするコードが生成された場合は，そのことをコメントで示す．

コメントの種類としては，CakePHP 独自のコードの説明，Web アプリケーションの基本的な説明，プログラマが加筆修正を行いそうな部分にそのことを示す説明，条件分岐についての説明の 4 種類に分けられる．CakePHP 独自のコードの説明としては，HtmlHelper や，SessionComponent 等を使った時に，それらを使用したということを示したり，Model からのデータの取得や，View に値をセットすることを示したりを生成する．Web アプリケーションの基本的な説明としては，画面遷移することを示したり，POST 送信や GET 送信をすることを示したりを生成する．プログラマが加筆修正を行いそうな部分にそのことを示す説明としては，コードの記述例を生成し，その下にどう加筆修正をすればよいのかについての説明を生成する．条件分岐の説明としては，主に if 文で何を条件として分岐するかについてが生成される．実際に生成されるコメントについては，第 4 章で述べていく．

3.5 本ツールのソースコード

付録その 1 に，Java で記述された本ツールのソースコードを掲載しておく．表 3.1 にあるのは本ツールのソースコードリストである．

表 3.1: 本ツールのソースコードリスト

ソースコード 1	CreateCode.java
ソースコード 2	LoadER.java
ソースコード 3	LoadScreen.java
ソースコード 4	CreateDirectory.java
ソースコード 5	CreateFile.java
ソースコード 6	EditChar.java
ソースコード 7	WriteController.java
ソースコード 8	WriteView.java
ソースコード 9	WriteModel.java

CreateCode は本ツールのメインクラスであり，ここで他のクラスの呼び出しを行っている．LoadER は，ER 図の XML 読み込み用クラスであり，ここで ER 図を読み込み，配列へと格納し，読み込んだデータをメインクラスへと送っている．LoadScreen は画面遷移図の XML 読み込み用クラスであり，ER 図と同様に，配列へと格納してメインクラスへとデータを送っている．CreateDirectory と CreateView は，それぞれフォルダとファイル生成用クラスであり，メインクラスから受け取った名前でフォルダやファイルの生成を行っている．EditChar は文字列操作用クラスであり，文字列を受け取ったら最初の文字を大文字にしたり，文字列の一部分を抽出したりを行って，文字列を返す．WriteControler，WriteView，WriteModel はそれぞれ CakePHP の Model，View，Controller のソースコードを生成するクラスであり，ここでメインクラスからデータを受け取り，CreateFile で生成されたファイルにソースコードの記述を行う．本ツールのクラス図は，図 3.7 に示しておく．

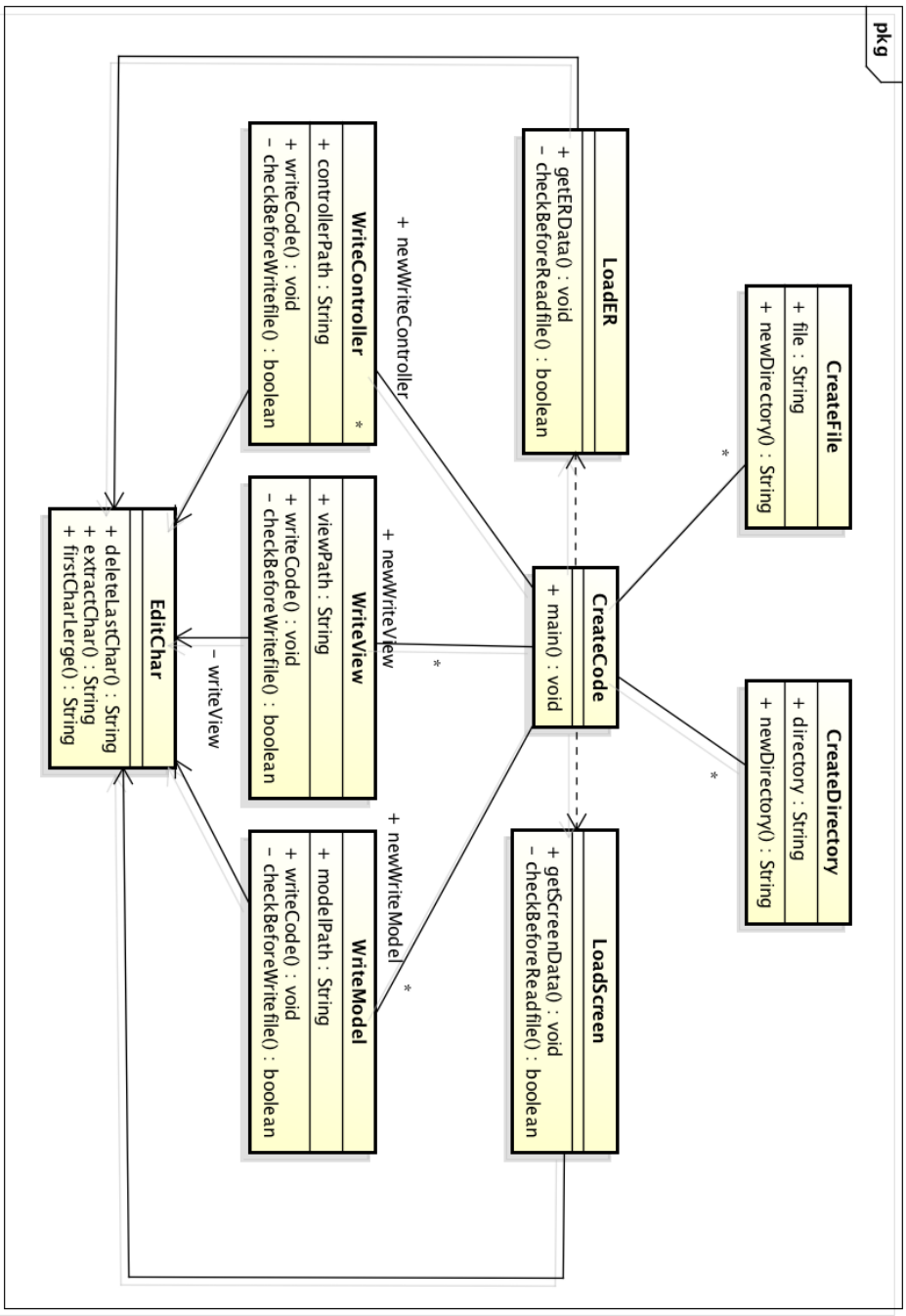

図 3.7: 本ツールのクラス図

第4章　コード生成実験

本研究では本ツールの有用性を示すために面談予約管理システムの作成を行う．

4.1　実験題材

面談予約管理システムとは，学生が教員に面談を予約するときに利用するものであり，教員は面談候補日の登録，削除，編集，一覧表示機能，面談タイトルの登録，削除，一覧表示機能，面談情報の登録，削除，編集，一覧表示，全削除機能といった機能を利用することができる．学生はこの機能の一部を利用することができ，面談情報の登録，削除，編集機能を利用することができる．他にも，ログイン機能や過去に行われた面談の一覧表示といった機能もある．

この面談予約管理システムの画面遷移図は，図 4.1 に示されている．

図 4.1 の画面遷移図には画面数が 17 個存在している．認証画面でログインをすることで，各種画面の機能を利用することができ，各一覧表示画面から,「追加」,「削除」,「編集」といった画面に遷移できることがわかる．

面談予約管理システムの ER 図は図 5.1 で示されている．

図 5.1 の ER 図ではテーブル数は 4 個存在し，各テーブル間の対応関係や，カラム名が記述されている．appointments テーブルと，users テーブルが多対 1 の関係で示されており，appointments テーブルに users テーブルの外部キーとして，user_id というカラムが存在していることがわかる．appointments テーブルは他にも candidates テーブルと多対 1 の関係があり，canditates テーブルと 1 対多の関係がある titles テーブルの外部キーも持っていることがわかる．

4.2　実験の手順と制約

本研究では画面遷移図と ER 図を作成し，そこから本ツールを適用して，CakePHP のソースコードを生成することでこの面談予約管理システムの作成を試みる．そして，複数の学生たちによって本ツールを使わずに作成された同様の機能を持つ面談予約管理システムのソースコードと比較することで，本ツールの有用性を確かめる．なお，この学生たちはプログラミングの経験はあるが，CakePHP を使ってのシステム開発は初めての人たちである．また，レイアウトに関する部分は考慮しないものとする．

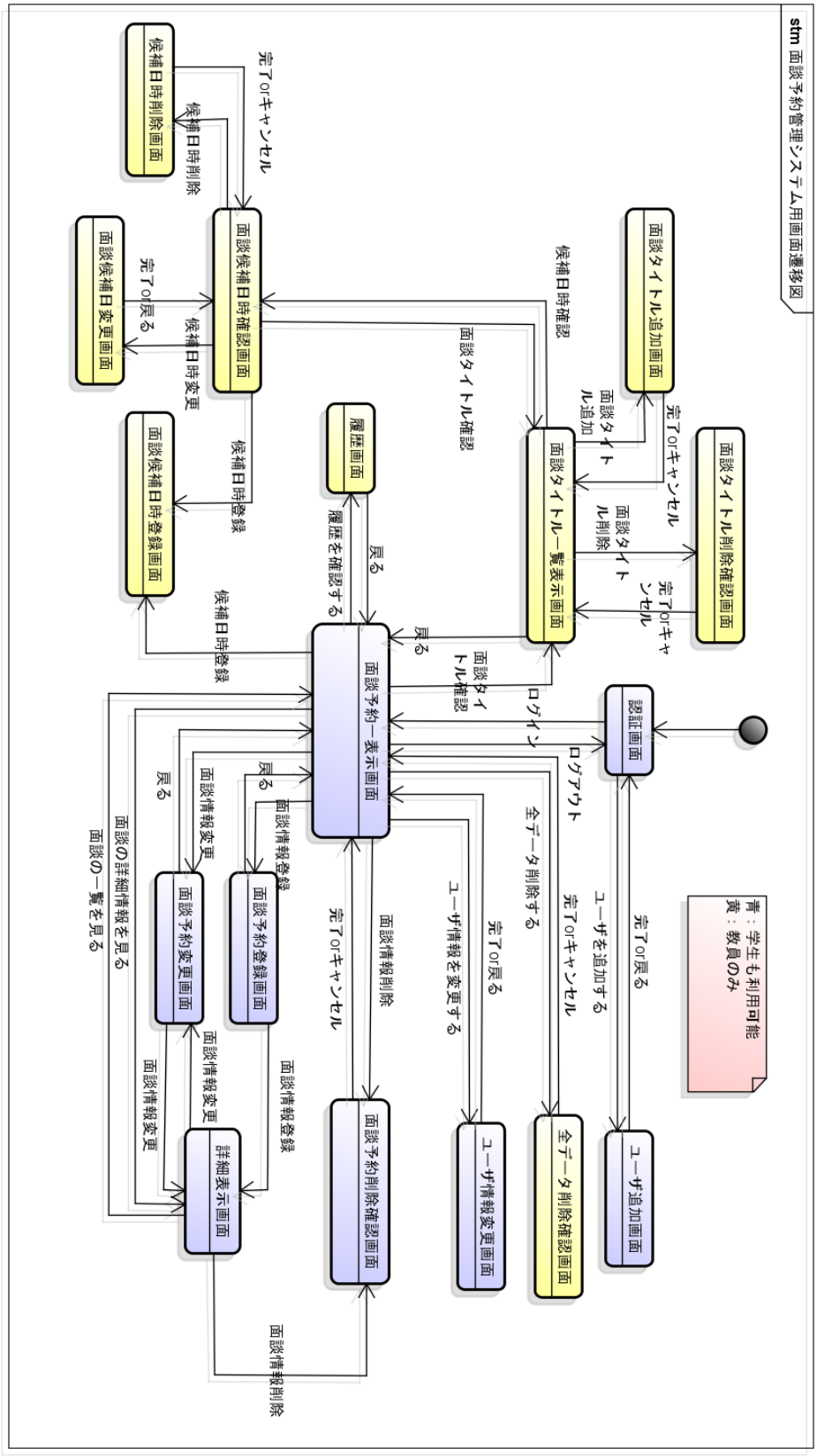

図 4.1: 面談予約管理システム用画面遷移図

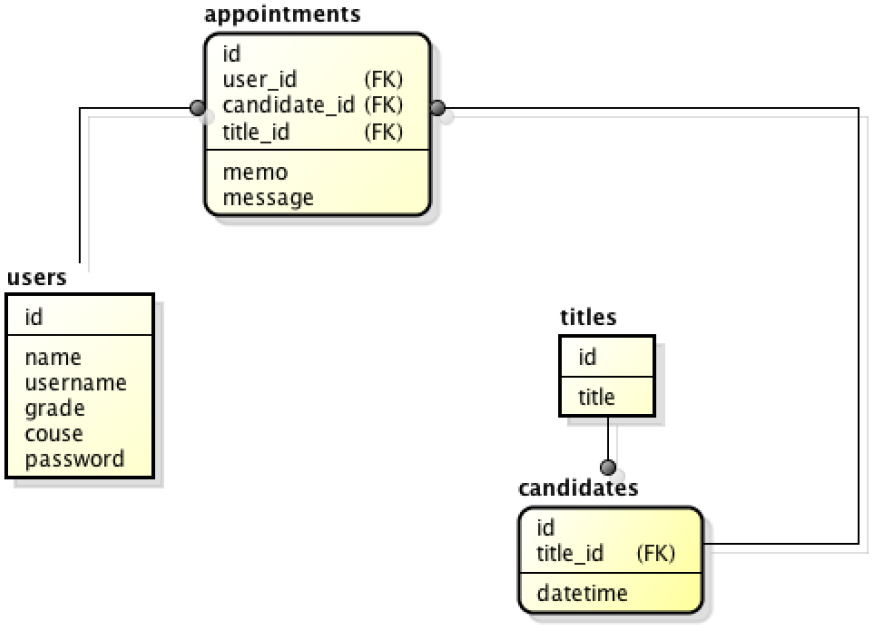

図 4.2: 面談予約管理システム用 ER 図

4.3 実験結果

面談予約管理システムの画面遷移図と ER 図に本ツールを適用した結果，表 4.1 に示されているファイルができた．

学生たちが作成したファイルと同じものが全て生成されていた．Model と Controller は ER 図のテーブル数と対応し，それぞれ 4 つのファイルが生成された．View は画面遷移図の画面数と対応し，17 個のファイルが生成された．生成されたファイルには，例えば 4.1，4.2 のようなソースコードが記述されていた．

ソースコード 4.1: add.ctp

```
1  <div class="candidates␣form">
2  <?php echo $this->Form->create('Candidate'); ?>
3  <?php //Form ヘルパーを使用 ?>
4          <fieldset>
5                  <legend>追加画面</legend>
6          <?php
7                  echo $this->Form->input('datetime',
8                          array(
9                                  'label' => 'datetime',
10                                 'dateFormat' => 'YMD',
11                                 'timeFormat' => '24',
12                                 'monthNames' => false,
13                                 'separator' => '/',
14                         ));
```

表 4.1: 生成されたファイル一覧

Model	Controller	View
Appointment.php	AppointmentsController.php	add.ctp delete_all.ctp delete.ctp edit.ctp history.ctp index.ctp view.ctp
Candidate.php	CandidatesController.php	add.ctp delete.ctp edit.ctp index.ctp
Title.php	TitlesController.php	add.ctp delete.ctp index.ctp
User.php	UsersController.php	add.ctp edit.ctp login.ctp

```
15                  echo $this->Form->input('title_id',
16                      array(
17                          'type' => 'radio',
18                          'value' => 1,
19                          'options' => $radio1,
20                      ));
21                  //入力フォームを表示
22          ?>
23          </fieldset>
24  <?php echo $this->Form->end('送信'); ?>
25  <?php //フォームの内容を POST 送信する ?>
26  </div>
27
28  <div class="actions">
29          <h3>画面</h3>
30          <ul>
31                  <li><?php echo $this->Html->link('index', array('action' => '
                        index')); ?></li>
32          </ul>
33  </div>
```

ソースコード 4.1 の add.ctp は面談候補日の追加を行うときに表示される Candidates の View ファイルである．CakePHP に用意されている Form ヘルパーを利用し，入力フォームの生成を行っている．datetime というカラムは，型が DATETIME になっているため，入力フォームはプルダウン形式で年月日を選択できるように生成されている．title_id は外部キーであり，candidates テーブルが titles テーブルと多対 1 の関係があるため，ラジオボタン形式で面談タイトルを選択できるように生成されている．

ソースコード 4.2: AppointmentsController.php

```
1  <?php
2  class AppointmentsController extends AppController {
3          public $name = 'Appointments';
4
5          public $uses = array(
6                  'Appointment'
7                  , 'User'
8                  , 'Candidate'
9                  , 'Title'
10         );
11         //使用するモデル名を列挙
12
13         public $components = array('Auth');
14         //AuthComponent を使用
15
16         public function beforeFilter(){
17                 $this->Auth->authError='ログインが必要です';
18                 $this->Auth->allow('add');
19                 //ここで認証なしでも表示できる画面 (メソッド名)を記述
20         }
21
22         public function add() {
23                 $radio1 = $this->User->find('list', array('fields' => array('id'
                       , 'name')));
24                 //ラジオボタン用に User の値を取得
25                 $this->set('radio1', $radio1);
26                 //ビューに値をセット
27                 $radio2 = $this->Candidate->find('list', array('fields' => array
                       ('id', 'datetime')));
28                 //ラジオボタン用に Candidate の値を取得
29                 $this->set('radio2', $radio2);
30                 //ビューに値をセット
31                 $radio3 = $this->Title->find('list', array('fields' => array('id'
                       , 'title')));
32                 //ラジオボタン用に Title の値を取得
33                 $this->set('radio3', $radio3);
34                 //ビューに値をセット
35                 if ($this->request->is('post')) {
36                 //POST 送信されたら
37                         $this->Appointment->create();
38                         if ($this->Appointment->save($this->request->data)) {
39                         //送信されたデータがテーブルに保存されたら
40                                 $this->Session->setFlash('保存しました。');
41                                 //SessionComponent を使用
42                                 $this->redirect(array('action' => 'index'));
43                                 //index に遷移する
44                         } else {
45                                 $this->Session->setFlash('保存できませんでした。');
46                                 //SessionComponent を使用
47                         }
48                 }
49         }
50
51         public function index() {
52                 $this->Appointment->recursive = 0;
53                 $this->set('appointments', $this->paginate());
54                 //Pagination を利用
55                 $appointments= $this->Appointment->find('all');
```

```
56              //find メソッド（テーブルの中身を全て取得する）
57              $this->set('appointments', $appointments);
58              //ビューに値をセット
59      }
60
61      public function delete_all() {
62              if($this->request->is('post','delete_all')) {
63              //POST 送信かつ delete_all されたら
64                      if ($this->Appointment->query('TRUNCATE␣appointments;'
                            )) {
65                      //テーブルの全レコードを削除
66                              $this->Session->setFlash('
                                    appointments テーブルの全レコードを削除しました。
                                    ');
67                      //SessionComponent を使用
68                              $this->redirect(array('action' => 'index'));
69                              //index に遷移する
70                      }
71                      $this->Session->setFlash('全削除できませんでした。。');
72                      //SessionComponent を使用
73                      $this->redirect(array('action' => 'index'));
74                      //index に遷移する
75              }
76      }
77
78      public function edit($id = null) {
79              $radio1 = $this->User->find('list', array('fields' => array('id'
                    , 'name')));
80              //ラジオボタン用に User の値を取得
81              $this->set('radio1', $radio1);
82              //ビューに値をセット
83              $radio2 = $this->Candidate->find('list', array('fields' => array
                    ('id', 'datetime')));
84              //ラジオボタン用に Candidate の値を取得
85              $this->set('radio2', $radio2);
86              //ビューに値をセット
87              $radio3 = $this->Title->find('list', array('fields' => array('id'
                    , 'title')));
88              //ラジオボタン用に Title の値を取得
89              $this->set('radio3', $radio3);
90              //ビューに値をセット
91              $this->Appointment->id = $id;
92              if (!$this->Appointment->exists()) {
93              //その id が存在しなかった場合
94                      throw new NotFoundException('不正な id です。');
95              }
96              if ($this->request->is('post') || $this->request->is('put')) {
97              //post、あるいは put が実行されたら
98                      if ($this->Appointment->save($this->request->data)) {
99                      //送信されたデータがテーブルに保存されたら
100                             $this->Session->setFlash('変更しました。');
101                             //SessionComponent を使用
102                             $this->redirect(array('action' => 'index'));
103                             //index に遷移する
104                     }
105                     else {
106                             $this->Session->setFlash('変更できませんでした。');
107                             //SessionComponent を使用
108                     }
109             }
110             else {
```

```
111                         $options = array('conditions' => array('Appointment.' .
                                $this->Appointment->primaryKey => $id));
112                         $this->request->data = $this->Appointment->find('first
                                ', $options);
113                 }
114         }
115
116         public function view($id = null) {
117                 if (!$this->Appointment->exists($id)) {
118                 //その id が存在しなかった場合
119                         throw new NotFoundException('不正な id です。');
120                 }
121                 $options = array('conditions' => array('Appointment.' . $this->
                        Appointment->primaryKey => $id));
122                 $this->set('appointments', $this->Appointment->find('first',
                        $options));
123                 //ビューに値をセット
124         }
125
126         public function delete($id = null) {
127                 $this->Appointment->id = $id;
128                 if (!$this->Appointment->exists()) {
129                 //その id が存在しなかった場合
130                         throw new NotFoundException('不正な id です。');
131                 }
132                 $this->request->onlyAllow('post', 'delete');
133                 //post 送信かつ delete のときのみ実行を許可する
134                 if ($this->Appointment->delete()) {
135                         //データが削除できたら
136                         $this->Session->setFlash('削除しました。');
137                         //SessionComponent を使用
138                         $this->redirect(array('action' => 'index'));
139                         //index に遷移する
140                 }
141                 else {
142                         $this->Session->setFlash('削除できませんでした。');
143                         //SessionComponent を使用
144                         $this->redirect(array('action' => 'index'));
145                         //index に遷移する
146                 }
147         }
148
149         public function history() {
150                 $this->Appointment->recursive = 0;
151                 $this->set('appointments', $this->paginate());
152                 //Pagination を利用
153                 $appointments= $this->Appointment->find('all');
154                 //find メソッド（テーブルの中身を全て取得する）
155                 $this->set('appointments', $appointments);
156                 //ビューに値をセット
157         }
158
159 }
160 ?>
```

ソースコード 4.2 は，面談情報の追加や一覧表示，全削除，編集，詳細，削除，履歴表示機能の内部処理を行っている AppointmentsController である．最初に使用する Model 名が列挙され，ログイン周りの機能のことが生成されている．各メソッドでは View や Model と連携し，各 Model や View に値を渡したり，受け取ったりをして，データのやりとりを

行えるように生成されていることがわかる．生成されたコメントに関しては，CakePHP 独自のコードについての説明は，AuthComponent や SessionComponent を使用したことや，モデルから値を取得し，ビューに値をセットするといったことが生成されていることがわかる．なお，MVC アーキテクチャの View と，詳細画面についての view との混乱を避けるために，Model，View，Controller はそれぞれモデル，ビュー，コントローラーと全て片仮名表記で統一している．Web アプリケーションの基本的な説明はラジオボタンや POST 送信といった用語や，画面遷移が行われるといったコメントが生成されている．プログラマが加筆修正を行いそうな部分にそのことを示す説明は，AuthComponent を利用するときに，19 行目に「ここに認証なしでも表示できる画面 (メソッド名) を記述」といったようなことが書かれている．条件分岐についての説明は，if 文で条件分岐を行う条件についてが説明されている．

学生たちが作成したものと比較すると，画面遷移図を参照していたため，全てのファイルの自動生成が行えていた．ソースコードの行数に関しては，学生たちは 1043 行，本ツールでは 970 行記述されていた．そのうちコメントは，学生たちが 40 行，本ツールでは 145 行存在していた．

第5章　評価

評価では，学生たちが記述したコードや bake，Scaffolding との比較を行い，本ツールの有用性を確かめる．

5.1　学生たちが記述したコードとの比較

学生たちが記述したソースコードと比較すると，生成できたソースコードは 80 %程であり，フレームワークを使用する際にありがちな冗長な部分を生成することができていた．具体的には，Controlller では，各クラスやメソッド名の記述や，データベースからのデータの取得や，View や Model へのデータの受け渡しや，POST 送信の処理や，データを追加，変更，削除を行った際に，成功した場合の処理と失敗した場合の処理の条件分岐といったことが生成されていた．View や Model でも，View では，HTML の各タグや，入力フォームの生成，データの表示，Model では，各 Model 名や Model の連結，バリデーションといった，複数箇所に渡って記載されることの多い部分が全て自動生成されていた．そのため，コーディング作業は大幅に効率化できると思われる．また，変数の宣言の仕方やデータの取得の仕方等に統一性が生まれており，CakePHP に関するコメントも充実していたので，可読性が高く，容易に改変ができ，CakePHP を初めて扱う人でもソースコードが理解しやすいと思われる．他にも学生たちが記述したソースコードは，インデントがうまく整えられておらず，アルゴリズムが微妙に異なる部分もあったが，本ツールで作成されたソースコードはインデントが全て統一されており，一定の規則でコードが記述されていたのでコードの均質化が行われていた．他にも MVC それぞれに独立した処理を記述することができたため，ここからタスクの振り分けを行いたい場合でも，振り分けがしやすくなると思われる．

今回は機能面では全ての機能を生成することができたが，完全に仕様通りの自動生成を行えたわけではなかった．ログイン機能においては，本来なら教員がログインしたときと，学生がログインしたときでは，表示できる画面が異なるが，教員がログインしたときのみの機能しか生成されなかった．面談情報の一覧表示機能では，本来なら今日以降の面談情報のみを表示する仕様となっているが，テーブルの中身を全て参照してしまい，過去に行われた面談情報も全て表示されてしまっていた．面談情報の追加機能では最初に面談タイトルを選択すると，その面談タイトルに紐付けされた日付が表示され，面談情報を登録する際にその日付を選択する形式となっているが，その機能を生成することができず，全ての日付が表示され，選択できるような形で生成されてしまっていた．

他にも一部のバリデーションが生成されず，例えばユーザの追加を行う際に，パスワードを入力するが，そのパスワードの長さは 8 文字以上でなければ弾く仕様となっていたが，そのバリデーションを自動生成することができなかった．

実装が不十分だった部分もあり，認証画面でログインが成功した場合，面談予約一覧表示画面へと遷移するが，その遷移がうまく行えていなかった．このことに関しては，実装を進めることで対応可能だと思われる．

5.2 bake，Scaffoldingとの比較

CakePHPに予め用意されているbakeとScaffoldingで生成されるソースコードとの比較も行った．bakeとは，CakePHPの基本的な機能をコマンドライン上で生成できるツールである．bakeを起動すると，どのデータベースを扱うのか，どのModel，View，Controllerを生成するのか等を1つ1つ尋ねられるので，それに1つ1つ答えていくことでコードの自動生成が可能である．図5.1にbakeを実行しているときの画面を示す．

```
You are *required* to use the date.timezone setting or the date_default_timezone
_set() function. In case you used any of those methods and you are still getting
 this warning, you most likely misspelled the timezone identifier. We selected t
he timezone 'UTC' for now, but please set date.timezone to select your timezone.
 in /Applications/XAMPP/xamppfiles/htdocs/seminar/lib/Cake/Cache/CacheEngine.php
 on line 60

Welcome to CakePHP v2.4.10 Console
---------------------------------------------------------------
App : app
Path: /Applications/XAMPP/xamppfiles/htdocs/seminar/app/
---------------------------------------------------------------
Interactive Bake Shell
---------------------------------------------------------------
[D]atabase Configuration
[M]odel
[V]iew
[C]ontroller
[P]roject
[F]ixture
[T]est case
[Q]uit
What would you like to Bake? (D/M/V/C/P/F/T/Q)
>
```

図 5.1: bakeの実行画面

bakeで生成できる機能は,「追加」,「削除」,「編集」,「一覧表示」,「詳細表示」の5つの機能である．Scaffoldingはデータベースを用意し，コントローラーに$scaffoldと記述するだけで自動的に「追加」,「削除」,「編集」,「一覧表示」,「詳細表示」の5つの機能が利用できるようになるものである．

bakeやScaffoldingとの比較は表5.1で示す．

Modelの自動生成に関しては，本ツールではER図を参照しているため，それに応じた形で自動生成を行うが，bakeは決め打ちでしか生成できなかった．ViewとControllerも同様に，本ツールでは画面遷移図を参照していたため，それに応じた形で自動的に機能の生成を行うが，bakeでは決め打ちでしか生成できなかった．

生成されるソースコードに関しては，本ツールがbakeより多かった．機能面で見ると，bakeは面談予約管理システムの一部の機能しか作成することができず，ログイン機能や履歴表示機能，全削除機能といった機能を生成することができなかった．生成されるバリデーションに関しては，bakeは全てのカラムに対し，バリデーションをコメントとして記述し，必要に応じてそのコメントをプログラマが修正することで，バリデーションを行える形式になっていた．コメントで記述されたバリデーションはエラーメッセージの記述の仕方やallowEmptyかどうか，requiredかどうか等が記述されていた．コマンドライン上

表 5.1: 本ツールと bake と Scaffolding と学生のコードの比較

	本ツール	bake	Scaffolding	学生
Model	ER図から	決め打ち	生成不可	
View Controller	画面遷移図から	決め打ち	生成不可	
記述されたソースコード	全体の約80%	全体の約50%	0%	全体の100%
コメント	あり	あり	なし	少しあり
実施容易性	容易	やや難あり	容易	
インデント、アルゴリズム	統一	統一		統一されていない部分が多い

で 1 つのカラムに対し，1 回 1 回バリデーションを設定することもできるが，実際にコーディングを行う作業より，若干速くなる程度のものであると思われる．本ツールでは，ER 図を参照していたため，空の値を許さない場合はコメントアウトせずに notEmpty が記述され，型が date や datetime だった場合，ありえない日時を弾くようにバリデーションが記述されていた．そのため，ER 図を参照している本ツールのほうが，柔軟にバリデーションの自動生成を行うことができていた．バリデーションだけでなく，Model 同士の連結も bake では，1 つ 1 つの Model に対し，どの Model と繋がっているのかを答えていなければならなかった．本ツールでは ER 図のリレーションシップを参照しているため，すぐに自動生成を行える．これらの機能やバリデーションといったものは，まだ一部のものしか自動生成を行えるように実装を行っていないため，今後様々なシステムの画面遷移図や ER 図やソースコードを参照し，ツールの性能を向上させていくことで，本ツールのほうがより柔軟に，多機能の自動生成を行えるようになっていくと思われる．なお，Scaffolding は機能が使えるようになるだけであり，ソースコードの生成を自動で行うものではないので，ここからシステム開発を行うことはあまりないと思われる．

コメントに関しては，本ツールと bake では両方ともコメントの生成を行うが，中身は大きく異なることが判明した．bake で生成されたコメントは上記で述べた Model のバリデーション以外では，Controller にソースコード 5.1 のようなものが生成された．ソースコード 5.1 は bake で生成された AppointmentsController の edit メソッドである．

ソースコード 5.1: AppointmentsController.php の一部

```
1  /**
2   * edit method
3   *
4   * @throws NotFoundException
5   * @param string $id
6   * @return void
7   */
8          public function edit($id = null) {
```

```
 9                  if (!$this->Appointment->exists($id)) {
10                          throw new NotFoundException(__('Invalid appointment'));
11                  }
12                  if ($this->request->is(array('post', 'put'))) {
13                          if ($this->Appointment->save($this->request->data)) {
14                                  $this->Session->setFlash(__('The appointment has
                                        been saved.'));
15                                  return $this->redirect(array('action' => 'index'
                                        ));
16                          } else {
17                                  $this->Session->setFlash(__('The appointment
                                        could not be saved. Please, try again.'));
18                          }
19                  } else {
20                          $options = array('conditions' => array('Appointment.' .
                                $this->Appointment->primaryKey => $id));
21                          $this->request->data = $this->Appointment->find('first
                                ', $options);
22                  }
23                  $users = $this->Appointment->User->find('list');
24                  $candidates = $this->Appointment->Candidate->find('list');
25                  $titles = $this->Appointment->Title->find('list');
26                  $this->set(compact('users', 'candidates', 'titles'));
27          }
```

4 行目に記述されている NotFoundException は 404 Not Found エラーを表示させるという意味であり，ソースコード 5.1 のコメントには id が見つからなかった場合は 404 エラーを出し，id のパラメータは string 型であり，返り値はないといったことが，edit メソッドの上部に記述されていた．一方，本ツールで生成されたコメントは第 4 章のソースコード 4.2 で示されている通り，各行ごとにそのコードが何を示しているのかが記述されていた．

実施容易性に関しては，本ツールは画面遷移図の各画面に対応するテーブル名を記述するだけですぐにコードの自動生成が行えるが，bake はコマンドラインの設定を行い，どの部分を生成するのかを答えていかなければならない．bake ではオプションで 5 つの機能を一括で生成することも可能だが，そのようなことをすると必要のない機能まで生成されてしまうことがある．本ツールでは画面遷移図に応じて必要な機能のみを生成し，不要な機能の生成は行わない．

以上のことから，本ツールは他の自動生成ツールや学生が記述したコードと比べると，有用性が高いと思われる．bake や Scaffolding 等と比較すると，ER 図や画面遷移図を作らなければならない手間がかかるが，これらのモデルは Web アプリケーションの作成を行う際にはほぼ必須なものである．単独で小さな Web アプリケーションの開発を行う際は bake や Scaffolding のほうが有用性が高いかもしれないが，システムの規模が大きくなるほど可視化のためや意識共有のためにモデルが作成されるため，そこから簡単にコード生成を行ってくれる本ツールの有用性が高くなっていくと思われる．

第6章　考察

コード生成実験の結果を踏まえた考察を，本章で述べる．

6.1　入力モデルとコード生成率

今回は機能的な面で見ると，面談予約管理システム全体の自動生成が行えた．しかし，簡単な処理でも自動生成できなかった部分が多々あり，そういったものは画面遷移図やER図からでは読み取れないものであった．今回は，作成するモデルを簡単なものにすることで，雛形を生成し，機敏に開発できるように支援を行うツールであったため，これらの細かな処理はプログラマに記述してもらう生成方針であったが，完全に自動生成を行えるようにしたい場合は他のモデルも参照するか，画面遷移図とER図により多くの情報を付加する必要がある．例えば，画面遷移図にユーザ権といった情報を付加し，マスターユーザは全ての画面にアクセス可，その他のユーザは許可されている画面のみにアクセス可，といったようにすれば，今回自動生成を行えなかった教員と学生のログイン周りの機能を自動生成できるのではないかと思われる．

しかし，そのようにより多くの情報を付加する，あるいは入力とするモデルを増やすとなると，最初に述べた自動生成ツールの問題点に繋がってしまうため，このような細かな処理も生成できるようにしたい場合は，どういった形で自動生成を行うのかをよく検討していく必要がある．

6.2　コメントの生成

今回はCakePHPやWebアプリケーションの開発経験が浅い人が，ソースコードを理解しやすく，簡単に学べるような方針で，コメントも生成したが，同じようなコメントが複数箇所に出てきてしまう場合があることが判明した．例えば，Controllerでは，コンポーネントを使用する際に，そのコンポーネント名を生成するが，今回の生成実験ではSessionComponentを使用したという内容が全体で22ヶ所も生成されていた．さらに，CakePHPを既に何度も扱ったことのある人にとっては至る所にあるコメントが煩わしく感じてしまう可能性もある．よって，1，2回説明したコメントは表示させないようにし，習熟度によって生成するコメントの量を調整できるようにする等，コメントについてもよく検討していく必要性が出てきた．

6.3 ユーザインタフェースの生成

今回はレイアウトに関する部分は考慮せず，機能的な部分のみの自動生成を行ったが，ワイヤーフレーム等の Web ページの画面設計図を入力に加えれば，ユーザインタフェース周りの自動生成も行えそうである．ワイヤーフレームとは，Web ページの大まかなコンテンツやレイアウトを示した構成図のことであり，基本的には色や細かい装飾を考えるものではなく，何をどの位置には配置するのかといったレイアウト周りの設計を行うために作成されるものである．このワイヤーフレームを作成することのできるツールは様々なものが存在しており，その中には作成したワイヤーフレームを HTML で出力できるものも存在しているため，この HTML ファイルを解析し，入力として加えることで，レイアウト周りの自動生成も行えるのではないかと思われる．現状の画面遷移図と ER 図だけでも自動生成を行えるようにし，それに加えて画面設計図からも自動生成を行えるようにすれば，レイアウトに関する実装作業も大幅に短縮できるのではないかと思われる．

あるいは本ツールに様々なスタイルを用意しておき，追加，削除，編集などの画面それぞれにどのスタイルを適用するかを選んでもらうことで自動生成を行えるようにする方法も考えられる．今回開発した機能周りの自動生成のように，スタイルシートの記述例やコメントを生成することで，画面周りの開発支援も行えるのではないかと思われる．

6.4 他の MVC フレームワークへの対応

今回は CakePHP を題材としたが，他の MVC アーキテクチャを利用したフレームワークでも，画面遷移図と ER 図からソースコードの自動生成が行えるのではないかと思われる．今回開発したツールは表 3.1 のように，クラス分けがされており，WriteModel.java，WriteView.java，WriteController.java それぞれに CakePHP の Model，View，Controller のソースコードを埋め込んでいるため，埋め込むソースコードを各 MVC フレームワークのコードに書き換えることで，CakePHP 以外にも様々な PHP，Ruby，Java といった言語の MVC フレームワークへの対応が行えそうである．

他にも現在，Titanium Studio や，Apache Cordova(PhoneGap) といった，JavaScript や HTML5 を使って iOS や Android の両アプリを作成できるハイブリッドアプリが複数登場してきている．それらには MVC フレームワークが用意されているものもあり，上記で述べたように本研究で行ったことと同様の手順でこれらのソースコードが生成できれば，スマートフォンアプリにも対応できるのではないかと思われる．

第7章　まとめと今後の展望

7.1　まとめ

本研究では，Web アプリケーション開発事業で扱われているフレームワークや自動生成ツールが抱えている問題を解決するために，画面遷移図と ER 図から CakePHP のソースコードを自動で生成してくれるツールを開発した．本ツールを面談予約管理システムの画面遷移図と ER 図に適用することで，面談予約管理システム全体の 8 割程度の自動生成を行うことができた．フレームワークを使用する際にありがちな冗長な部分を生成できたが，いくつか課題があることも判明した．

7.2　今後の展望

今後の展望としては，CakePHP フレームワークを使用している他の Web アプリケーションに対して本ツールを適用し，生成できるソースコードを増やしていき，様々な機能を自動生成できるようにしていきたい．また，考察で出てきた課題を解決し，実システム開発で本ツールを適用することで，どの程度機敏に開発を行えるのかの検証を行いたい．さらに，他の MVC アーキテクチャを利用したフレームワークでも同様なことが行えないか，どんな画面設計図を入力とすればレイアウト周りの自動生成も容易に行えるのかの検証もしていきたい．

謝辞

本研究に対して様々な指導や的確なアドバイスをしてくださった，伊藤恵先生に深くお礼を申し上げます．また，ゼミの中で様々な助言をしてくださった，研究室のメンバー，合同ゼミで多くの指摘をしてくださった，奥野先生並びにその研究生達，南部先生並びにその研究生達，田柳先生並びにその研究生達にも心から感謝致します．また，本研究の査読を行ってくださった，日本ソフトウェア科学会の査読者の皆さんにも深くお礼を申し上げます．

参考文献

[1] 清水美樹, はじめての Ruby on Rails2, 2008, 工学社

[2] Sam Ruby, Dave Thomas, David Heinemeier Hansson, et al, 前田修吾, Rails によるアジャイル Web アプリケーション開発, 2009, オーム社

[3] Cake Software Foundation, CakePHP, http://cakephp.org/, 参照 (2015-1-20)

[4] 掌田津耶乃, CakePHP2.1 による Web アプリケーション開発, 2012, 株式会社 秀和システム

[5] スティーブ J. メラー, ケンドール スコット, アクセル ウール, ディルク ヴァイセ, 二上貴夫, 長瀬嘉秀, MDA Distilled:Principles of Model-Driven Architecture, MDA のエッセンス, 2004, 翔泳社

[6] 吉田裕之, 基礎からわかる MDA, 2006, 日経 BP 社

[7] スティーブ J. メラー, マーク J. バルサー, 二上 貴夫, 長瀬嘉秀, Executable UML A Foundation for Model-Driven Architecture MDA モデル駆動型アーキテクチャの基礎, 2003, 翔泳社

[8] 河村美嗣, 浅見可津志, UML を入力とするソースコード自動生成ツールの開発, 全国大会講演論文集 第 72 回平成 22 年 (1), 一般社団法人情報処理学会, 2010, p1-337-p1-338, http://ci.nii.ac.jp/naid/110008105498, 参照 (2015-1-20)

[9] 八木紀幸, 中所武司, モデル変換に基づくエンドユーザ主導の Web アプリケーション開発技法, 情報処理学会研究報告. ソフトウェア工学研究会報告, 一般社団法人情報処理学会, 2009, no.31, p81-p86, http://ci.nii.ac.jp/naid/110007333794, 参照 (2015-1-20)

[10] キヤノンソフトウェア株式会社, Web Performer, http://www.canon-soft.co.jp/product/web_performer/, 参照 (2015-1-20)

[11] 株式会社ジャスミンソフト, Wagby, http://wagby.com/, 参照 (2014-12-16)

[12] ジェネクサス・ジャパン株式会社, GeneXus, http://www.genexus.com/japan/genexus-japan?ja, 参照 (2014-12-16)

[13] 伊賀敏樹, blanco Framework, http://www.igapyon.jp/blanco/blanco.ja.html, 参照 (2014-12-16)

[14] 株式会社チェンジビジョン, astah*, `http://astah.change-vision.com/ja/index.html`, 参照 (2015-1-20)

付録1:本ツールで開発した自動生成ツールのソースコード

ソースコード 1: CreateCode.java

```
1  package cakephpgenerator;
2
3
4  //メインクラス
5  public class CreateCode {
6  public static void main(String args[]){
7  String result;//結果通知用変数
8  String erdata[] = new String[50];//ER 図からのデータの格納用配列
9  String stdata[] = new String[50];//画面遷移図からのデータの格納用配列
10 String path = "/Applications/XAMPP/xamppfiles/htdocs/seminar";
11
12 LoadER.getERData(erdata);//ER 図の読み込み
13 LoadScreen.getScreenData(stdata);//画面遷移図の読み込み
14
15 for(int j=0;;j++) {
16 if(erdata[j]==null) break;
17 String[] table = erdata[j].split(",",0);//「,」で区切って配列として格納
18 //モデルのファイルの作成
19 String modelname = table[0].substring(0, table[0].length()−1);//文字列の最後の
       1文字を削除
20 CreateFile newModelFile = new CreateFile();
21 newModelFile.file = path + "/app/Model/" + modelname + ".php";
22 result = newModelFile.newFile(modelname + ".php");
23 System.out.println(result);
24 //外部キーを持っている時
25 String FK[] = new String[50];
26 int k = 0;
27 for(int i=0;i<table.length;i++){
28 if (table[i].indexOf("_id") != −1) {//もし外部キーがあったら
29 FK[k] = table[i];
30 k++;
31 }
32 }
33 k = 0;
34 //外部キーが持たれているとき
35 String FK2[] = new String[50];
36 String lowermodelname = modelname.toLowerCase();
37 for(int i=0;;i++){
38 if(erdata[i]==null) break;
39 String[] table2 = erdata[i].split(",",0);//「,」で区切って配列として格納
40 for(int l=1;l<table2.length;l++){
41 if(table2[l].indexOf(lowermodelname) != −1) {
42 String[] table3 = table2[l].split("&",0);
43 if(table3.length==3){
44 FK2[k] = table2[0] + "&" + table2[l];
45 k++;
46 }
```

```
47 }
48 }
49 }
50 //モデルへの書き込み
51 WriteModel newtable = new WriteModel();
52 newtable.modelPath = path + "/app/Model/" + modelname + ".php";
53 FK = deleteNotNullForArray(FK);
54 FK2 = deleteNotNullForArray(FK2);
55 newtable.writeCode(modelname,FK,FK2,stdata,erdata);
56 }
57
58 //ビューとコントローラーの作成
59 erdata = deleteNotNullForArray(erdata);
60 for(int j=0;;j++) {
61 if(stdata[j]==null) break;
62 String[] table = stdata[j].split(",",0);
63 //ビューのフォルダ生成
64 CreateDirectory newViewFolder = new CreateDirectory();
65 newViewFolder.directory = path + "/app/View/" + table[0];
66 result = newViewFolder.newDirectory(table[0] + "フォルダ");
67 System.out.println(result);
68 //コントローラーのファイル生成
69 CreateFile newControllerFile = new CreateFile();
70 newControllerFile.file = path + "/app/Controller/" + table[0] + "Controller.php";
71 result = newControllerFile.newFile(table[0] + "Controller.php");
72 System.out.println(result);
73 //コントローラーへの書き込み
74 for(int k = 0;;k++){
75 if(erdata[k]==null) {
76 System.out.println("画面遷移図のテーブル名が
      ER 図のテーブル名と一致するものがありませんでした␣画面遷移図のテーブル名␣="
      + table[0]);
77 break;
78 }
79 String[] ertable = erdata[k].split(",",0);
80 if(ertable[0].equals(table[0])){
81 WriteController newWriteController = new WriteController();
82 newWriteController.controllerPath = path + "/app/Controller/" + table[0] + "
      Controller.php";
83 //    String[] erdatas = deleteNotNullForArray(erdata);
84 //    String erdatass = deleteNotNull(erdata[k]);
85 newWriteController.writeCode(stdata[j], erdata[k], erdata, stdata);
86 break;
87 }
88 }
89 for(int k=1;k<table.length;k++){
90 //ビューのファイルの作成
91 CreateFile newViewFile = new CreateFile();
92 newViewFile.file = path + "/app/View/" + table[0] + "/" + table[k] + ".ctp";
93 result = newViewFile.newFile(table[k] + ".ctp");
94 System.out.println(result);
95 //ビューへの書き込み
96 WriteView newwriteview = new WriteView();
97 newwriteview.viewPath = path + "/app/View/" + table[0] + "/" + table[k] + ".ctp";
98 String meth = table[k];//メソッド名
99 for(int l=0;;l++){
100 if(erdata[l]==null) {
101 System.out.println("画面遷移図のテーブル名が
      ER 図のテーブル名と一致するものがありませんでした␣画面遷移図のテーブル名␣="
      + table[0]);
102 break;
```

```
103 }
104 String[] tabledata = erdata[l].split(",",0);
105 if(tabledata[0].equals(table[0])) {
106 //      String[] erdatas = deleteNotNullForArray(erdata);
107 //      String[] tabledatass = deleteNotNullForArray(tabledata);
108 newwriteview.writeCode(meth,tabledata,erdata,stdata);
109 break;
110 }
111 }
112 }
113 }
114 }
115
116 public static String deleteNotNull(String data){
117 return data.replaceAll("@notnull", "");
118 }
119
120 public static String[] deleteNotNullForArray(String[] data){
121 for(int i = 0;;i++){
122 if(data[i]==null) break;
123 data[i] = deleteNotNull(data[i]);
124 }
125 return data;
126 }
127 }
```

ソースコード 2: LoadER.java

```
1  package cakephpgenerator;
2
3  import java.io.File;
4  import java.io.FileReader;
5  import java.io.BufferedReader;
6  import java.io.FileNotFoundException;
7  import java.io.IOException;
8
9  //ER 図読み込み用クラス
10 public class LoadER{
11 String loadm;
12
13 public static void getERData(String array[]){
14 //String array[] = new String[5000];//デバッグ用
15 String type_ids[] = new String[5000];//データの型の id と名前を格納する配列
16 String rship_ids[] = new String[5000];//多重度の id と上限、下限を格納する配列
17 String notnull_ids[] = new String[500];//not_null の id を格納する変数
18 int count = 0;
19 try{
20 File file = new File("/Users/b1011129/資料/sample/面談予約管理システム用ER 図 2.
       xml");
21
22 if (checkBeforeReadfile(file)){
23 BufferedReader br = new BufferedReader(new FileReader(file));
24 BufferedReader br2 = new BufferedReader(new FileReader(file));//データの型取得用
25 BufferedReader br3 = new BufferedReader(new FileReader(file));//多重度の取得用
26 BufferedReader br4 = new BufferedReader(new FileReader(file));
27
28 String str;
29 int i = 0; //配列用変数
30 int j = 0;
31
```

```
32
33 rship_ids = getXMLData(br3, count);
34 count++;
35 br3.close();
36 type_ids = getXMLData(br2, count);
37 br2.close();
38 count++;
39 notnull_ids = getXMLData(br4, count);
40 br4.close();
41 while((str = br.readLine()) != null){//1行ずつ確認していく
42 if(str.indexOf("<JUDE:EREntityPresentation") != -1) break;
43 else if(str.indexOf("<JUDE:EREntity") != -1) {//テーブル名取得
44 str = EditChar.extractChar(str, "name=\"",␣"\"");
45 array[i] = EditChar.firstCharLerge(str);
46 i++;
47 }
48 else if(str.indexOf("<JUDE:ERAttribute.referencedPrimaryKey>") != -1) {
49 //str = "PK";
50 }
51 else if(str.indexOf("<JUDE:ERAttribute.referencedRelationship>") != -1) {
52 //str = "";
53 }
54 else if(str.indexOf("<JUDE:ERAttribute.referencedForeignKeys>") != -1) {
55 //str = ".FK";
56 //array[i-1] += str;
57 }
58 else if(str.indexOf("<JUDE:ERAttribute␣xmi.idref") != -1) {
59 //str = "";
60 }
61 else if(str.indexOf("<JUDE:ERAttribute") != -1) {//カラム名を取得
62 array[i-1] += "," + EditChar.extractChar(str, "name=\"",␣"\"");
63 }
64 else if(str.indexOf("<UML:Classifier␣xmi.idref=") != -1) {//各カラムに対し、型を
       検索し、一致するものがあったら名前だけを入れる
65 str = EditChar.extractChar(str, "xmi.idref=\"",␣"\"");
66 for(int k = 0;;k++){
67 if(type_ids[k]==null) break;
68 if(type_ids[k].indexOf(str) != -1){
69 String[] name = type_ids[k].split(",",0);
70 array[i-1] += "&" + name[1];
71 }
72 }
73 }
74 else if(str.indexOf("<UML:Constraint␣xmi.idref=") != -1) {//各カラムに対し、
       notnull の id が存在するか確認する
75 str = EditChar.extractChar(str, "xmi.idref=\"",␣"\"");
76 for(int k = 0;;k++){
77 if(notnull_ids[k]==null) break;
78 if(notnull_ids[k].indexOf(str) != -1){
79 array[i-1] += "@notnull";
80 }
81 }
82 }
83 else if(str.indexOf("<UML:ERAttribute␣xmi.idref=") != -1) {//外部キーに対し、多重
       度を格納する
84 str = EditChar.extractChar(str, "xmi.idref=\"",␣"\"");
85 for(int k = 0;;k++){
86 if(rship_ids[k]==null) break;
87 if(rship_ids[k].indexOf(str) != -1){
88 String[] name = rship_ids[k].split("%",0);
```

```
89  if(name[1].equals("1")&&name[2].equals("1")&&name[3].equals("1")&&name[4].equals(
        "1")) array[i-1] += "&" + "one_to_one";
90  else if(name[1].equals("0")&&name[2].equals("1")&&name[3].equals("0")&&name[4].
        equals("1")) array[i-1] += "&" + "one_to_one";
91  else if(name[1].equals("1")&&name[2].equals("1")&&name[3].equals("0")&&name[4].
        equals("1")) array[i-1] += "&" + "one_to_one";
92  else if(name[1].equals("0")&&name[2].equals("1")&&name[3].equals("1")&&name[4].
        equals("1")) array[i-1] += "&" + "one_to_one";
93  else if(name[1].equals("1")&&name[2].equals("1")&&name[3].equals("1")&&name[4].
        equals("-1")) array[i-1] += "&" + "many_to_one";
94  else if(name[1].equals("0")&&name[2].equals("1")&&name[3].equals("0")&&name[4].
        equals("-1")) array[i-1] += "&" + "many_to_one";
95  else if(name[1].equals("1")&&name[2].equals("1")&&name[3].equals("0")&&name[4].
        equals("-1")) array[i-1] += "&" + "many_to_one";
96  else if(name[1].equals("0")&&name[2].equals("1")&&name[3].equals("1")&&name[4].
        equals("-1")) array[i-1] += "&" + "many_to_one";
97  else if(name[1].equals("1")&&name[2].equals("-1")&&name[3].equals("1")&&name[4].
        equals("1")) array[i-1] += "&" + "one_to_many";
98  else if(name[1].equals("0")&&name[2].equals("-1")&&name[3].equals("0")&&name[4].
        equals("1")) array[i-1] += "&" + "one_to_many";
99  else if(name[1].equals("1")&&name[2].equals("-1")&&name[3].equals("0")&&name[4].
        equals("1")) array[i-1] += "&" + "one_to_many";
100 else if(name[1].equals("0")&&name[2].equals("-1")&&name[3].equals("1")&&name[4].
        equals("1")) array[i-1] += "&" + "one_to_many";
101 else if(name[1].equals("1")&&name[2].equals("-1")&&name[3].equals("1")&&name[4].
        equals("-1")) array[i-1] += "&" + "many_to_many";
102 else if(name[1].equals("0")&&name[2].equals("-1")&&name[3].equals("0")&&name[4].
        equals("-1")) array[i-1] += "&" + "many_to_many";
103 else if(name[1].equals("1")&&name[2].equals("-1")&&name[3].equals("0")&&name[4].
        equals("-1")) array[i-1] += "&" + "many_to_many";
104 else if(name[1].equals("0")&&name[2].equals("-1")&&name[3].equals("1")&&name[4].
        equals("-1")) array[i-1] += "&" + "many_to_many";
105 }
106 }
107 }
108 }
109
110 System.out.println("ERdata------------------------------");
111 for(j=0;;j++){//デバッグ用、全体
112 if(array[j]==null) break;
113 System.out.println(array[j]);
114 }
115
116 //    for(j=0;;j++){//デバッグ用、型
117 //     if(type_ids[j]==null) break;
118 //     System.out.println(type_ids[j]);
119 //    }
120
121 //    for(j=0;;j++){//デバッグ用、多重度
122 //    if(rship_ids[j]==null) break;
123 //    System.out.println(rship_ids[j]);
124 //    }
125
126 //    for(j=0;;j++){//デバッグ用、notnull
127 //    if(notnull_ids[j]==null) break;
128 //    System.out.println(notnull_ids[j]);
129 //    }
130 System.out.println("-----------------------------------");
131 br.close();
132 }else{
133 System.out.println("ファイルが見つからないか開けません");
```

```
134 }
135 }catch(FileNotFoundException e){
136 System.out.println(e);
137 }catch(IOException e){
138 System.out.println(e);
139 }
140 }
141
142 public static String[] getXMLData(BufferedReader br, int count){
143 String array[] = new String[5000];
144 try{
145 //String rship_id;
146 String str;
147 //String rship_ids[] = new String[5000];//多重度のid と上限、下限を格納する配列
148 String upper;//上限
149 String lower;//下限
150 String type_name;//型の名前格納用変数
151 String notNullName;
152 int i = 0;
153 while((str = br.readLine()) != null){//1行ずつ確認していく
154 switch(count){
155 case 0://多重度の id と上限、下限を取得
156 if(str.indexOf("<JUDE:ERRelationship␣xmi.id=") != -1){//id を取得
157 str = EditChar.extractChar(str,"xmi.id=\"","\"");
158 array[i] = str;
159 //rship_ids[i] = rship_id;
160 i++;
161 }
162 if(str.indexOf("<UML:MultiplicityRange␣xmi.id=") != -1){//上限と下限を取得
163 lower = EditChar.extractChar(str,"lower=\"","\"");
164 upper = EditChar.extractChar(str,"upper=\"","\"");
165 array[i-1] += "%" + lower + "%" + upper;
166 //rship_ids[i-1] += "%" + lower + "%" + upper;
167 }
168 case 1:
169 if(str.indexOf("<UML:Class␣xmi.id=") != -1){//全てのデータの型の
        id と名前を探し、配列へ格納していく
170 type_name = EditChar.extractChar(str, "name=\"",␣"\"");
171 str = EditChar.extractChar(str, "xmi.id=\"",␣"\"");
172 array[i] = str + "," + type_name;
173 i++;
174 }
175 case 2:
176 if(str.indexOf("<UML:Constraint␣xmi.id=") != -1){//NOT NULL かどうかを取得
177 notNullName = EditChar.extractChar(str, "name=\"",␣"\"");
178 if(notNullName.equals("NOT+NULL")){
179 array[i] = EditChar.extractChar(str, "xmi.id=\"",␣"\"␣name=");
180 i++;
181 }
182
183 }
184 }
185 }
186 }catch(FileNotFoundException e){
187 System.out.println(e);
188 }catch(IOException e){
189 System.out.println(e);
190 }
191 return array;
192 }
193
194 //ファイルを開く前の確認用メソッド
```

```
195 private static boolean checkBeforeReadfile(File file){
196 if (file.exists()){
197 if (file.isFile() && file.canRead()){
198 return true;
199 }
200 }
201
202 return false;
203 }
204 }
```

ソースコード 3: LoadScreen.java

```
1  package cakephpgenerator;
2
3  import java.io.File;
4  import java.io.FileReader;
5  import java.io.BufferedReader;
6  import java.io.FileNotFoundException;
7  import java.io.IOException;
8  import java.net.URLDecoder;
9
10 //画面遷移図読み込み用クラス
11 public class LoadScreen{
12 String loads;
13
14 static void getScreenData(String table[]){
15 try{
16 File file = new File("/Users/b1011129/資料/sample/面談予約管理システム用画面遷移
       図.xml");
17
18 if (checkBeforeReadfile(file)){
19 BufferedReader br = new BufferedReader(new FileReader(file));
20
21 String str;
22 String array[] = new String[50];//デバッグ用
23 //String[] table = new String[50];
24 String sort2[] = new String[50];//ソート用配列
25 int i = 0; //配列用変数
26 int name;
27 while((str = br.readLine()) != null){//1行ずつ確認していく
28 if(str.indexOf("</JUDE:StateChartDiagram>") != −1) break;//画面の名前を取得
29 else if(str.indexOf("<UML:CompositeState␣xmi.id=") != −1) {//画面遷移図名取得
30 name = str.indexOf("name=\"");
31 name␣+=␣"name=\"".length();
32 str = str.substring(name);//前の余計な文字列を削除
33 name = str.indexOf("\"");
34 str␣=␣str.substring(0,name);//後ろの余計な文字列を削除
35 str␣=␣URLDecoder.decode(str,"utf−8");
36 if(str.indexOf("認証")␣!=␣-1||str.indexOf("ログイン")␣!=␣-1)␣str␣=␣"login";
37 else␣if(str.indexOf("追加")␣!=␣-1||str.indexOf("登録
       ")␣!=␣-1||str.indexOf("保存")␣!=␣-1)␣str␣=␣"add";
38 else␣if(str.indexOf("編集")␣!=␣-1||str.indexOf("変更")␣!=␣-1)␣str␣=␣"edit";
39 else␣if(str.indexOf("削除")␣!=␣-1)␣{
40 if(str.indexOf("全")␣!=␣-1)␣str␣=␣"delete_all";
41 else␣str␣=␣"delete";
42 }
43 else␣if(str.indexOf("一覧")␣!=␣-1||str.indexOf("確認
       ")␣!=␣-1)␣str␣=␣"index";
44 else␣if(str.indexOf("詳細")␣!=␣-1)␣str␣=␣"view";
```

```
45 else␣if(str.indexOf("履歴")␣!=␣-1)␣str␣=␣"history";
46 else␣str␣=␣"none";
47 array[i]␣=␣str;
48 }
49 else␣if(str.indexOf("<UML:ModelElement.definition xmi.value=")␣!=␣-1)␣{//プ
       ロパティビューの要素を取得
50 name␣=␣str.indexOf("xmi.value=\"");
51 name += "xmi.value=\"".length();
52 str␣=␣str.substring(name);//前の余計な文字列を削除
53 name␣=␣str.indexOf("\"");
54 str = str.substring(0,name);//後ろの余計な文字列を削除
55 str = EditChar.firstCharLerge(str);
56 array[i] = str + "," + array[i];
57 i++;
58 }
59 }
60 //ソート (1つの配列の一番左にテーブル名､以降はカンマで区切り各メソッドが入る)
61 i=0;
62 while(array[i]!=null){
63 String[] sort = array[i].split(",",0);
64 for(int j=0;;j++){
65 if(table[j]!=null&&table[j].indexOf(",") != −1)
66 sort2 = table[j].split(",",0);
67 if(table[j]==null) {
68 table[j] = sort[0] + "," + sort[1];
69 break;
70 }
71 if(sort2[0].equals(sort[0])) {
72 table[j] += "," + sort[1];
73 break;
74 }
75 }
76 i++;
77 }
78
79 i=0;
80 System.out.println("STdata------------------------------");
81 while(table[i]!=null){//デバッグ用
82 System.out.println(table[i]);
83 i++;
84 }
85 System.out.println("-----------------------------------");
86 br.close();
87 }else{
88 System.out.println("ファイルが見つからないか開けません");
89 }
90 }catch(FileNotFoundException e){
91 System.out.println(e);
92 }catch(IOException e){
93 System.out.println(e);
94 }
95 }
96
97 //ファイルを開く前の確認用メソッド
98 private static boolean checkBeforeReadfile(File file){
99 if (file.exists()){
100 if (file.isFile() && file.canRead()){
101 return true;
102 }
103 }
104
105 return false;
```

```
106 }
107 }
```

ソースコード 4: CreateDirectory .java

```
1  package cakephpgenerator;
2
3  import java.io.File;
4
5  //ディレクトリ生成用クラス
6  public class CreateDirectory {
7  String directory;
8  String newDirectory(String name) {
9  File newdirectory = new File(directory);
10 if(newdirectory.mkdir()) return name + "を生成しました";
11 else return name + "は既に存在しています";
12 }
13 }
```

ソースコード 5: CreateFile.java

```
1  package cakephpgenerator;
2
3  import java.io.File;
4  import java.io.IOException;
5
6  //ファイル生成用クラス
7  public class CreateFile {
8  String file;
9  String newFile(String name) {
10 File newfile = new File(file);
11
12 try{
13 if(newfile.createNewFile()) return name + "を生成しました";
14 else return name + "は既に存在しています";
15 }catch(IOException e) {
16 System.out.println(e);
17 return name + "が生成できませんでした";
18 }
19 }
20 }
```

ソースコード 6: EditChar.java

```
1  package cakephpgenerator;
2
3  //文字列操作用クラス
4  public class EditChar {
5
6  //文字列の最後の 1文字を消去するメソッド
7  public static String deleteLastChar(String text){
8  return text.substring(0, text.length()−1);
9  }
10
11 //文字列の一部を抽出するメソッド
12 public static String extractChar(String text, String fronttext, String latertext){
```

```
13 int index;
14 index = text.indexOf(fronttext);
15 index += fronttext.length();
16 text = text.substring(index);//前の文字列を消去
17 index = text.indexOf(latertext);
18 text = text.substring(0,index);//後ろの文字列を消去
19 return text;
20 }
21
22 //文字列の最初の文字だけを大文字にするメソッド
23 public static String firstCharLerge(String text){
24 String firsttext = text.substring(0,1);//1文字目を取得
25 firsttext = firsttext.toUpperCase();//大文字に
26 text = text.substring(2−1);//2文字目から最後の文字まで (1文字目を消去)
27 text = firsttext + text;
28 return text;
29 }
30 }
```

ソースコード 7: WriteController .java

```
1  package cakephpgenerator;
2
3  import java.io.File;
4  import java.io.FileWriter;
5  import java.io.BufferedWriter;
6  import java.io.PrintWriter;
7  import java.io.IOException;
8
9  //コントローラー書き込み用クラス
10 class WriteController {
11 String controllerPath;
12
13 void writeCode(String stdate, String tablename, String[] erdata, String[] stdatas) {
14 String[] array = stdate.split(",",0);
15 String modelname = array[0].substring(0, array[0].length()−1);//モデル名
16 String lcname = array[0].toLowerCase();//小文字に変換
17 String[] tabledata = tablename.split(",",0);
18 int checkbox_count;
19 int radio_count;
20 int foreign_flag = 0;
21 int login_flag = 0;
22 int index;
23 try {
24 File file = new File(controllerPath);
25
26 if (checkBeforeWritefile(file)) {
27 PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(file)));
28
29 pw.println("<?php");
30 pw.println("class␣" + array[0] + "Controller␣extends␣AppController␣{");
31 pw.println("\tpublic␣$name␣=␣\'" + array[0] + "\';\n");
32 for(int i=1;i<tabledata.length;i++){
33 if(tabledata[i]==null) break;
34 String[] colum = tabledata[i].split("&",0);
35 if(colum.length==3){
36 if(foreign_flag == 0){
37 pw.println("\tpublic␣$uses␣=␣array(");
38 pw.println("\t\t'" + modelname + "\'␣");
39 foreign_flag = 1;
```

```
40 }
41 index = colum[0].indexOf("_id");
42 String columname = colum[0].substring(0,index);//後ろの余計な文字列を削除
43 String foreignmodelname = EditChar.firstCharLerge(columname);//外部キー元のモデ
       ル名
44 if(foreign_flag == 0) pw.println("\t\t\'" + foreignmodelname + "\'");
45 else pw.println("\t\t,␣\'" + foreignmodelname + "\'");
46 }
47 }
48 if(foreign_flag==1) {
49 pw.println("\t);");
50 pw.println("\t//使用するモデル名を列挙\n");
51 }
52
53 for(int i=0;;i++){//ログイン周りの記述
54 if(stdatas[i]==null) break;
55 String[] tablemethods = stdatas[i].split(",", 0);
56 for(int j=1;tablemethods.length>j;j++){
57 if(tablemethods[j].equals("login")){
58 pw.println("\tpublic␣$components␣=␣array('Session',␣'Auth');\n");
59 pw.println("\tpublic␣function␣beforeFilter(){");
60 pw.println("\t\t$this->Auth->authError='ログインが必要です';");
61 pw.println("\t\t//$this->Auth->allow('add');");
62 pw.println("\t\t//ここで認証なしでも表示できる画面(メソッド名)を記述");
63 pw.println("\t}\n");
64 if(tablemethods[0].equals(modelname + "s")){//もし
       login メソッドが記述されているテーブル名だったら
65 login_flag = 1;
66 }
67 break;
68 }
69 }
70 }
71
72 for(int i=1;i<array.length; i++){
73 if(array[i].equals("add")) {//追加機能
74 checkbox_count = 1;
75 radio_count = 1;
76 pw.println("\tpublic␣function␣add()␣{");
77 for(int j=1;j<tabledata.length;j++) {
78 if(tabledata[j]==null) break;
79 String[] colum = tabledata[j].split("&",0);
80 if(colum.length==3){//外部キーが存在する場合
81 index = colum[0].indexOf("_id");
82 String columname = colum[0].substring(0,index);//後ろの余計な文字列を削除
83 String foreignmodelname = EditChar.firstCharLerge(columname);//外部キー元のモデ
       ル名
84 String foreigncolum;
85 for(int k=0;;k++){
86 String[] table = erdata[k].split(",",0);
87 if(table[0].equals(foreignmodelname + "s")){
88         for(int l=1;;l++){
89             if(table[l].indexOf("id")==-1){//主キー、外部キーでなかった場合
90                 String[] columnames = table[l].split("&",0);
91                 foreigncolum = columnames[0];
92                 break;
93             }
94         }
95         break;
96 }
97 }
```

```
98  if(colum[2].equals("one_to_many")||colum[2].equals("many_to_many")){//1対多、多対多
        の場合
99  pw.println("\t\t$select" + checkbox_count + "␣=␣$this->" + foreignmodelname +
        "->find('list',␣array('fields'␣=>␣array('id',␣'" + foreigncolum + "'));
        ");
100 pw.println("\t\t$rhis->set->('select" + checkbox_count + "',␣$select" +
        checkbox_count + ")");
101 checkbox_count++;
102 }
103 else {
104 pw.println("\t\t$radio" + radio_count + "␣=␣$this->" + foreignmodelname + "->
        find('list',␣array('fields'␣=>␣array('id',␣'" + foreigncolum + "'));");
105 pw.println("\t\t$this->set('radio" + radio_count + "',␣$radio" + radio_count +
        ");");
106 radio_count++;
107 }
108 }
109 }
110 pw.println("\t\tif␣($this->request->is('post'))␣{");
111 pw.println("\t\t//POST 送信されたら");
112 pw.println("\t\t\t$this->" + modelname + "->create();");
113 pw.println("\t\t\tif␣($this->" + modelname + "->save($this->request->data))
        ␣{");
114 pw.println("\t\t\t//送信されたデータがテーブルに保存されたら");
115 pw.println("\t\t\t\t$this->Session->setFlash('保存しました。');");
116 pw.println("\t\t\t\t//SessionComponent を使用");
117 if(login_flag==1) pw.println("\t\t\t\t$this->Auth->login();");
118 pw.println("\t\t\t\t$this->redirect(array('action'␣=>␣'index'));");
119 pw.println("\t\t\t\t//index に遷移する");
120 pw.println("\t\t\t}␣else␣{");
121 pw.println("\t\t\t\t$this->Session->setFlash('保存できませんでした。');");
122 pw.println("\t\t\t\t//SessionComponent を使用");
123 pw.println("\t\t\t}");
124 pw.println("\t\t}");
125 pw.println("\t}\n");
126 }
127
128 else if(array[i].equals("index")) {//一覧表示機能
129 pw.println("\tpublic␣function␣index()␣{");
130 pw.println("\t\t$this->" + modelname + "->recursive␣=␣0;");
131 pw.println("\t\t$this->set('" + lcname + "',␣$this->paginate());");
132 pw.println("\t\t//Pagination を利用");
133 pw.println("\t\t$" + lcname + "=␣$this->" + modelname + "->find('all');");
134 pw.println("\t\t//find メソッド（テーブルの中身を全て取得する）");
135 pw.println("\t\t$this->set('" + lcname + "',␣$" + lcname + ");");
136 pw.println("\t\t//ビューに値をセット");
137 pw.println("\t}\n");
138 }
139 else if(array[i].equals("delete")) {//削除機能
140 pw.println("\tpublic␣function␣delete($id␣=␣null)␣{");
141 pw.println("\t\t$this->" + modelname + "->id␣=␣$id;");
142 pw.println("\t\tif␣(!$this->" + modelname + "->exists())␣{");
143 pw.println("\t\t//そのid が存在しなかった場合");
144 pw.println("\t\t\tthrow␣new␣NotFoundException('不正なid です。');");
145 pw.println("\t\t}");
146 pw.println("\t\t$this->request->onlyAllow('post',␣'delete');");
147 pw.println("\t\t//post 送信かつ delete のときのみ実行を許可する");
148 pw.println("\t\tif␣($this->" + modelname + "->delete())␣{");
149 pw.println("\t\t\t//データが削除できたら");
150 pw.println("\t\t\t$this->Session->setFlash('削除しました。');");
151 pw.println("\t\t\t//SessionComponent を使用");
```

```
152 pw.println("\t\t\t$this->redirect(array('action' => 'index'));");
153 pw.println("\t\t\t//index に遷移する");
154 pw.println("\t\t}");
155 pw.println("\t\telse {");
156 pw.println("\t\t\t$this->Session->setFlash('削除できませんでした。');");
157 pw.println("\t\t\t//SessionComponent を使用");
158 pw.println("\t\t\t$this->redirect(array('action' => 'index'));");
159 pw.println("\t\t\t//index に遷移する");
160 pw.println("\t\t}");
161 pw.println("\t}\n");
162 }
163 else if(array[i].equals("edit")) {//編集機能
164 checkbox_count = 1;
165 radio_count = 1;
166 pw.println("\tpublic function edit($id = null) {");
167 for(int j=1;j<tabledata.length;j++) {
168 if(tabledata[j]==null) break;
169 String[] colum = tabledata[j].split("&",0);
170 if(colum.length==3){
171 index = colum[0].indexOf("_id");
172 String columname = colum[0].substring(0,index);//後ろの余計な文字列を削除
173 String foreignmodelname = EditChar.firstCharLerge(columname);//外部キー元のモデル名
174 String foreigncolum;
175 for(int k=0;;k++){
176 String[] table = erdata[k].split(",",0);
177 if(table[0].equals(foreignmodelname + "s")){
178         for(int l=1;;l++){
179                 if(table[l].indexOf("id")==-1){//主キー、外部キーでなかった場合
180                         String[] columnames = table[l].split("&",0);
181                         foreigncolum = columnames[0];
182                         break;
183                 }
184         }
185         break;
186 }
187 }
188 if(colum[2].equals("one_to_many")||colum[2].equals("many_to_many")){//1対多、多対多の場合
189 pw.println("\t\t$select" + checkbox_count + " = $this->" + foreignmodelname + "->find('list', array('fields' => array('id', '" + foreigncolum + "')));");
190 pw.println("\t\t$rhis->set->('select" + checkbox_count + "', $select" + checkbox_count + ")");
191 checkbox_count++;
192 }
193 else {
194 pw.println("\t\t$radio" + radio_count + " = $this->" + foreignmodelname + "->find('list', array('fields' => array('id', '" + foreigncolum + "')));");
195 pw.println("\t\t$this->set('radio" + radio_count + "', $radio" + radio_count + ");");
196 radio_count++;
197 }
198 }
199 }
200 pw.println("\t\t$this->" + modelname + "->id = $id;");
201 pw.println("\t\tif (!$this->" + modelname + "->exists()) {");
202 pw.println("\t\t//そのid が存在しなかった場合");
203 pw.println("\t\t\tthrow new NotFoundException('不正なid です。');");
204 pw.println("\t\t}");
```

```
205 pw.println("\t\tif␣($this->request->is('post')␣||␣$this->request->is('put
        '))␣{");
206 pw.println("\t\t//post、あるいは put が実行されたら");
207 pw.println("\t\t\tif␣($this->" + modelname + "->save($this->request->data))
        ␣{");
208 pw.println("\t\t\t//送信されたデータがテーブルに保存されたら");
209 pw.println("\t\t\t\t$this->Session->setFlash('変更しました。');");
210 pw.println("\t\t\t\t//SessionComponent を使用");
211 if(login_flag==1) pw.println("\t\t\t\t$this->Auth->login();");
212 pw.println("\t\t\t\t$this->redirect(array('action'␣=>␣'index'));");
213 pw.println("\t\t\t\t//index に遷移する");
214 pw.println("\t\t\t}");
215 pw.println("\t\t\telse␣{");
216 pw.println("\t\t\t\t$this->Session->setFlash('変更できませんでした。');");
217 pw.println("\t\t\t\t//SessionComponent を使用");
218 pw.println("\t\t\t}");
219 pw.println("\t\t}");
220 pw.println("\t\telse␣{");
221 pw.println("\t\t\t$options␣=␣array('conditions'␣=>␣array('" + modelname + "
        .'␣.␣$this->" + modelname + "->primaryKey␣=>␣$id));");
222 pw.println("\t\t\t$this->request->data␣=␣$this->" + modelname + "->find('
        first',␣$options);");
223 pw.println("\t\t}");
224 pw.println("\t}\n");
225 }
226 else if(array[i].equals("view")) {//詳細表示機能
227 pw.println("\tpublic␣function␣view($id␣=␣null)␣{");
228 pw.println("\t\tif␣(!$this->" + modelname + "->exists($id))␣{");
229 pw.println("\t\t//そのid が存在しなかった場合");
230 pw.println("\t\t\tthrow␣new␣NotFoundException('不正なid です。');");
231 pw.println("\t\t}");
232 pw.println("\t\t$options␣=␣array('conditions'␣=>␣array('" + modelname + ".'
        ␣.␣$this->" + modelname + "->primaryKey␣=>␣$id));");
233 pw.println("\t\t$this->set('" + lcname + "',␣$this->" + modelname + "->find
        ('first',␣$options));");
234 pw.println("\t\t//ビューに値をセット");
235 pw.println("\t}\n");
236 }
237 else if(array[i].equals("history")) {//履歴表示機能
238 pw.println("\tpublic␣function␣history()␣{");
239 pw.println("\t\t$this->" + modelname + "->recursive␣=␣0;");
240 pw.println("\t\t$this->set('" + lcname + "',␣$this->paginate());");
241 pw.println("\t\t//Pagination を利用");
242 pw.println("\t\t$" + lcname + "=␣$this->" + modelname + "->find('all');");
243 pw.println("\t\t//find メソッド（テーブルの中身を全て取得する）");
244 pw.println("\t\t$this->set('" + lcname + "',␣$" + lcname + ");");
245 pw.println("\t\t//ビューに値をセット");
246 pw.println("\t}\n");
247 }
248 else if(array[i].equals("delete_all")) {//全削除機能
249 pw.println("\tpublic␣function␣delete_all()␣{");
250 pw.println("\t\tif($this->request->is('post','delete_all'))␣{");
251 pw.println("\t\t//POST 送信かつ delete_all されたら");
252 pw.println("\t\t\tif␣($this->" + modelname + "->query('TRUNCATE␣" + lcname +
        ";'))␣{");
253 pw.println("\t\t\t//テーブルの全レコードを削除");
254 pw.println("\t\t\t\t$this->Session->setFlash('" + lcname + "テーブルの全レコー
        ドを削除しました。');");
255 pw.println("\t\t\t\t$this->redirect(array('action'␣=>␣'index'));");
256 pw.println("\t\t\t}");
257 pw.println("\t\t\t$this->Session->setFlash('全削除できませんでした。。');");
```

```
258 pw.println("\t\t\t$this->redirect(array('action' => 'index'));");
259 pw.println("\t\t}");
260 pw.println("\t}\n");
261 }
262 else if(array[i].equals("login")) {//ログイン機能
263 pw.println("\tpublic function login() {");
264 pw.println("\t\tif($this->request->is('post')) {");
265 pw.println("\t\t//ポスト送信されたら");
266 pw.println("\t\t\tif($this->Auth->login()){");
267 pw.println("\t\t\t\t$this->redirect(array('action' => 'index'));");
268 pw.println("\t\t\t}");
269 pw.println("\t\t\telse $this->Session->setFlash('ログインに失敗しました。');"
        );
270 pw.println("\t\t}");
271 pw.println("\t}\n");
272 }
273 else {
274 pw.println("\tpublic function " + array[i] + "() {");
275 pw.println("\t\t");
276 pw.println("\t}\n");
277 }
278 }
279 pw.println("}");
280 pw.println("?>");
281
282 pw.close();
283 System.out.println(array[0] + "Controller.php にコードを記述しました");
284 } else {
285 System.out.println(array[0] + "Controller.php にコードを記述できませんでした");
286 }
287 } catch(IOException e) {
288 System.out.println(e);
289 }
290 }
291
292 private static boolean checkBeforeWritefile(File file){
293 if (file.exists()) {
294 if (file.isFile() && file.canWrite()) {
295 return true;
296 }
297 }
298
299 return false;
300 }
301 }
```

ソースコード 8: WriteView.java

```
1  package cakephpgenerator;
2
3  import java.io.File;
4  import java.io.FileWriter;
5  import java.io.BufferedWriter;
6  import java.io.PrintWriter;
7  import java.io.IOException;
8
9  //ビューファイル書き込み用クラス
10 public class WriteView {
11 String viewPath;
12
```

```
13 void writeCode(String name, String[] tabledata, String[] erdata, String[] stdata) {//メ
       ソッド名、テーブル名、ER データ全部、画面遷移図全部
14 String modelname = tabledata[0].substring(0, tabledata[0].length()−1);//モデル名
15 String lcname = tabledata[0].toLowerCase();//小文字に変換
16 String lcnames = lcname.substring(0, lcname.length()−1);
17 int checkbox_count = 1;
18 int radio_count = 1;
19 //int date_count = 0;
20 //String[] columname = null;
21
22 try{
23 File file = new File(viewPath);
24
25 if(checkBeforeWritefile(file)) {
26 PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(file)));
27
28 if(name.equals("add")) {
29 pw.println("<div␣class=\""+␣lcname␣+␣" form\">");
30 pw.println("<?php␣echo␣$this->Form->create('" + modelname + "');␣?>");
31 pw.println("<?php␣//Form ヘルパーを使用␣?>");
32 pw.println("\t<fieldset>");
33 pw.println("\t\t<legend>追加画面</legend>");
34 pw.println("\t<?php");
35 for(int j=1;j<tabledata.length;j++) {
36 if(tabledata[j]==null) break;
37 String[] colum = tabledata[j].split("&",0);
38 if(colum[1].equals("DATE")){//型が date だったら
39 pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣");
40 pw.println("\t\t\tarray(");
41 pw.println("\t\t\t\t'label'␣=>␣'"+ colum[0] +"',");
42 pw.println("\t\t\t\t'dateFormat'␣=>␣'YMD',");
43 pw.println("\t\t\t\t'monthNames'␣=>␣false,");
44 pw.println("\t\t\t\t'separator'␣=>␣'/',");
45 pw.println("\t\t\t));");
46 }
47 else if(colum[1].equals("DATETIME")){//型が datetime だったら
48 pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣");
49 pw.println("\t\t\tarray(");
50 pw.println("\t\t\t\t'label'␣=>␣'"+ colum[0] +"',");
51 pw.println("\t\t\t\t'dateFormat'␣=>␣'YMD',");
52 pw.println("\t\t\t\t'timeFormat'␣=>␣'24',");
53 pw.println("\t\t\t\t'monthNames'␣=>␣false,");
54 pw.println("\t\t\t\t'separator'␣=>␣'/',");
55 pw.println("\t\t\t));");
56 }
57 else if(colum.length==3){//外部キーが存在した場合
58 if(colum[2].equals("one_to_many")||colum[2].equals("many_to_many")){//1対多、多対多
       の場合
59 pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣");
60 pw.println("\t\t\tarray(");
61 pw.println("\t\t\t\t'type'␣=>␣'select',");
62 pw.println("\t\t\t\t'multiple'=>␣'checkbox',");
63 pw.println("\t\t\t\t'options'␣=>␣$checkbox" + checkbox_count + ",");
64 pw.println("\t\t\t\t'label'␣=>␣'"+ colum[0] + "',");
65 pw.println("\t\t\t));");
66 checkbox_count++;
67 }
68 else {
69 pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣");
70 pw.println("\t\t\tarray(");
71 pw.println("\t\t\t\t'type'␣=>␣'radio',");
```

```
72  pw.println("\t\t\t\t'value'␣=>␣1,");
73  pw.println("\t\t\t\t'options'␣=>␣$radio" + radio_count + ",");
74  pw.println("\t\t\t));");
75  radio_count++;
76  }
77  }
78  else if(colum[0].equals("id")){
79
80  }
81  else pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣array('label'
        ␣=>␣'" + colum[0] + "'));");
82  }
83  pw.println("\t\t//入力フォームを表示");
84  pw.println("\t?>");
85  pw.println("\t</fieldset>");
86  pw.println("<?php␣echo␣$this->Form->end('送信');␣?>");
87  pw.println("<?php␣//フォームの内容をPOST 送信する␣?>");
88  pw.println("</div>\n");
89  pw.println("<div␣class=\"actions\">");
90  pw.println("\t<h3>画面</h3>");
91  pw.println("\t<ul>");
92  pw.println("\t\t<li><?php␣echo␣$this->Html->link('index',␣array('action'␣=>
        ␣'index'));␣?></li>");
93  pw.println("\t</ul>");
94  pw.println("</div>");
95  }
96  if(name.equals("index")) {
97  pw.println("<div␣class=\""+␣lcname␣+␣" index\">");
98  pw.println("\t<legend>一覧画面</legend>");
99  pw.println("\t<table␣cellpadding=\"0\"␣cellspacing=\"0\">");
100 pw.println("\t<tr>");
101 for(int j=1;j<tabledata.length;j++){
102 if(tabledata[j]==null) break;
103 String[] colum = tabledata[j].split("&",0);
104 if(colum[0].indexOf("_id") != −1){//外部キーが存在したら
105 for(int k=0;;k++){
106 if(erdata[k]==null) break;
107 String[] foreigntabledata = erdata[k].split(",",0);
108 String foreignmodelname = foreigntabledata[0].substring(0, foreigntabledata[0].length
        ()−1);//モデル
        名
109 String lcforeignmodelname = foreignmodelname.toLowerCase();//小文字に変換
110 if(colum[0].indexOf(lcforeignmodelname) != −1){
111 for(int l=1;l<foreigntabledata.length;l++){
112 String[] foreigncolum = foreigntabledata[l].split("&",0);
113 if(foreigncolum[0].equals("id")){
114
115 }
116 else if(foreigncolum[0].indexOf("_id") != −1){
117
118 }
119 else{
120 pw.println("\t\t<th><?php␣echo␣$this->Paginator->sort('" + foreigncolum[0] + "
        ',␣'" + foreigncolum[0] + "');␣?></th>");
121 }
122 }
123 }
124 }
125 }
126 else if(colum[0].equals("id")){
127
```

```
128 }
129 else{
130 pw.println("\t\t<th><?php␣echo␣$this->Paginator->sort('" + colum[0] + "',␣'"
        + colum[0] + "');␣?></th>");
131 }
132 }
133 pw.println("\t\t<th><?php␣?>");
134 pw.println("\t\t<th␣class=\"actions\">操作</th>");
135 pw.println("\t</tr>\n");
136 pw.println("\t<?php␣foreach␣($" + lcname + "␣as␣$" + lcnames + "):␣?>");
137 pw.println("\t<?php␣//foreach 文（繰り返し）?>");
138 pw.println("\t<tr>");
139 for(int j=1;j<tabledata.length;j++) {
140 if(tabledata[j]==null) break;
141 String[] colum = tabledata[j].split("&",0);
142 if(colum[0].indexOf("_id") != −1){//外部キーが存在したら
143 for(int k=0;;k++){
144 if(erdata[k]==null) break;
145 String[] foreigntabledata = erdata[k].split(",",0);
146 String foreignmodelname = foreigntabledata[0].substring(0, foreigntabledata[0].length
        ()−1);//モデル
        名
147 String lcforeignmodelname = foreignmodelname.toLowerCase();//小文字に変換
148 if(colum[0].indexOf(lcforeignmodelname) != −1){
149 for(int l=1;l<foreigntabledata.length;l++){
150 String[] foreigncolum = foreigntabledata[l].split("&",0);
151 if(foreigncolum[1].equals("DATE")){
152 pw.println("\t\t<td><?php␣echo␣h(date('Y/m/d',␣strtotime($" + lcnames + "['"
        + foreignmodelname + "'][' + foreigncolum[0] + "']));␣?> </td>");
153 }
154 else if(foreigncolum[1].equals("DATETIME")){
155 pw.println("\t\t<td><?php␣echo␣h(date('Y/m/d␣H:i',␣strtotime($" + lcnames +
        "['" + foreignmodelname + "'][' + foreigncolum[0] + "']));␣?> </td>
        ");
156 }
157 else if(foreigncolum[0].equals("id")){
158
159 }
160 else if(foreigncolum[0].indexOf("_id") != −1){
161
162 }
163 else{
164 pw.println("\t\t<td><?php␣echo␣h($" + lcnames + "['" + foreignmodelname + "
        '][' + foreigncolum[0] + "']);␣?> </td>");
165 }
166 }
167 }
168 }
169 }
170 else if(colum[0].equals("id")){
171
172 }
173 else {
174 pw.println("\t\t<td><?php␣echo($" + lcnames + "['" + modelname + "'][' +
        colum[0] + "']);␣?> </td>");
175 }
176 }
177 pw.println("\t\t<td␣class=\"actions\">");
178 for(int i=0;;i++){
179 if(stdata[i]==null) break;
180 String[] st = stdata[i].split(",",0);
```

```
181 if(modelname.equals(EditChar.deleteLastChar(st[0]))){
182 for(int j=1;j<st.length;j++){
183 if(st[j].equals("edit")){
184 pw.println("\t\t\t<?php␣echo␣$this->Html->link('編集',␣array('action'␣=>␣'
        edit',␣$" + lcnames + "['" + modelname + "']['id']));␣?>");
185 pw.println("\t\t\t<?php␣//Html ヘルパーを利用␣?>");
186 }
187 else if(st[j].equals("view")){
188 pw.println("\t\t\t<?php␣echo␣$this->Html->link('詳細',␣array('action'␣=>␣'
        view',␣$" + lcnames + "['" + modelname + "']['id']));␣?>");
189 pw.println("\t\t\t<?php␣//Html ヘルパーを利用␣?>");
190 }
191 else if(st[j].equals("delete")){
192 pw.println("\t\t\t<?php␣echo␣$this->Form->postLink('削除',␣array('action'␣=>
        ␣'delete',␣$" + lcnames + "['" + modelname + "']['id']),␣null,␣'本当に
        削除しますか？');␣?>");
193 pw.println("\t\t\t<?php␣//Form ヘルパーを利用、削除確認画面が表示される␣?>");
194 }
195 }
196 }
197 }
198 pw.println("\t\t</td>");
199 pw.println("\t</tr>");
200 pw.println("\t<?php␣endforeach;␣?>");
201 pw.println("\t<?php␣//foreach 文終わり␣?>");
202 pw.println("\t</table>\n");
203 pw.println("\t<div␣class=\"paging\">");
204 pw.println("\t<?php");
205 pw.println("\t\techo␣$this->Paginator->prev('<␣'␣.␣'前のページ',␣array(),␣
        null,␣array('class'␣=>␣'prev␣disabled'));");
206 pw.println("\t\techo␣$this->Paginator->numbers(array('separator'␣=>␣''));");
207 pw.println("\t\techo␣$this->Paginator->next('次のページ'␣.␣'␣>',␣array(),␣
        null,␣array('class'␣=>␣'next␣disabled'));");
208 pw.println("\t?>");
209 pw.println("\t</div>");
210 pw.println("</div>\n");
211 pw.println("<div␣class=\"actions\">");
212 pw.println("\t<h3>画面</h3>");
213 pw.println("\t<ul>");
214 for(int i=0;;i++){
215 if(stdata[i]==null) break;
216 String[] st = stdata[i].split(",",0);
217 if(modelname.equals(EditChar.deleteLastChar(st[0]))){
218 for(int j=1;j<st.length;j++){
219 if(st[j].equals("index")||st[j].equals("view")||st[j].equals("edit")||st[j].equals("delete"
        )){
220
221 }else {
222 pw.println("\t\t<li><?php␣echo␣$this->Html->link('" + st[j] + "',␣array('
        action'␣=>␣'" + st[j] + "'));␣?></li>");
223 }
224 }
225 }else{
226 pw.println("\t\t<li><?php␣echo␣$this->Html->link('" + st[0] + "/index',␣array
        ('controller'␣=>␣'" + st[0] + "',␣'action'␣=>␣'index'));␣?></li>");
227 }
228 }
229 pw.println("\t</ul>");
230 pw.println("</div>");
231 }
232 if(name.equals("edit")) {
```

```
233 pw.println("<div␣class=\""+␣lcname␣+␣" form\">");
234 pw.println("<?php␣echo␣$this->Form->create('" + modelname + "');␣?>");
235 pw.println("<?php␣//Form ヘルパーを使用␣?>");
236 pw.println("\t<fieldset>");
237 pw.println("\t\t<legend>編集画面</legend>");
238 pw.println("\t<?php");
239 for(int j=1;j<tabledata.length;j++) {
240 if(tabledata[j]==null) break;
241 String[] colum = tabledata[j].split("&",0);
242 if(colum[1].equals("DATE")){//型が date だったら
243 pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣");
244 pw.println("\t\t\tarray(");
245 pw.println("\t\t\t\t'label'␣=>␣'"+ colum[0] +"',");
246 pw.println("\t\t\t\t'dateFormat'␣=>␣'YMD',");
247 pw.println("\t\t\t\t'monthNames'␣=>␣false,");
248 pw.println("\t\t\t\t'separator'␣=>␣'/',");
249 pw.println("\t\t\t));");
250 }
251 else if(colum[1].equals("DATETIME")){//型が datetime だったら
252 pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣");
253 pw.println("\t\t\tarray(");
254 pw.println("\t\t\t\t'label'␣=>␣'"+ colum[0] +"',");
255 pw.println("\t\t\t\t'dateFormat'␣=>␣'YMD',");
256 pw.println("\t\t\t\t'timeFormat'␣=>␣'24',");
257 pw.println("\t\t\t\t'monthNames'␣=>␣false,");
258 pw.println("\t\t\t\t'separator'␣=>␣'/',");
259 pw.println("\t\t\t));");
260 }
261 else if(colum.length==3){
262 if(colum[2].equals("one_to_many")||colum[2].equals("many_to_many")){//1対多、多対多
      の場合
263 pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣");
264 pw.println("\t\t\tarray(");
265 pw.println("\t\t\t\t'type'␣=>␣'select',");
266 pw.println("\t\t\t\t'multiple'=>␣'checkbox',");
267 pw.println("\t\t\t\t'options'␣=>␣$checkbox" + checkbox_count + ",");
268 pw.println("\t\t\t\t'label'␣=>␣'"+ colum[0] + "',");
269 pw.println("\t\t\t));");
270 checkbox_count++;
271 }
272 else if(colum[0].equals("id")){
273
274 }
275 else {
276 pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣");
277 pw.println("\t\t\tarray(");
278 pw.println("\t\t\t\t'type'␣=>␣'radio',");
279 pw.println("\t\t\t\t'value'␣=>␣1,");
280 pw.println("\t\t\t\t'options'␣=>␣$radio" + radio_count + ",");
281 pw.println("\t\t\t));");
282 radio_count++;
283 }
284 }
285 else pw.println("\t\techo␣$this->Form->input('" + colum[0] + "',␣array('label'
      ␣=>␣'" + colum[0] + "'));");
286 }
287 pw.println("\t\t//入力フォームを表示");
288 pw.println("\t?>");
289 pw.println("\t</fieldset>");
290 pw.println("<?php␣echo␣$this->Form->end('送信');␣?>");
291 pw.println("<?php␣//フォームの内容をPOST 送信する␣?>");
```

```
292 pw.println("</div>\n");
293 pw.println("<div␣class=\"actions\">");
294 pw.println("\t<h3>画面</h3>");
295 pw.println("\t<ul>");
296 pw.println("\t\t<li><?php␣echo␣$this->Html->link('index',␣array('action'␣=>
        ␣'index'));␣?></li>");
297 pw.println("\t</ul>");
298 pw.println("</div>");
299 }
300 if(name.equals("history")) {
301 pw.println("<div␣class=\""+␣lcname␣+␣" history\">");
302 pw.println("\t<legend>履歴画面</legend>");
303 pw.println("\t<table␣cellpadding=\"0\"␣cellspacing=\"0\">");
304 pw.println("\t<tr>");
305 for(int j=1;j<tabledata.length;j++){
306 if(tabledata[j]==null) break;
307 String[] colum = tabledata[j].split("&",0);
308 if(colum[0].indexOf("_id") != −1){//外部キーが存在したら
309 for(int k=0;;k++){
310 if(erdata[k]==null) break;
311 String[] foreigntabledata = erdata[k].split(",",0);
312 String foreignmodelname = foreigntabledata[0].substring(0, foreigntabledata[0].length
        ()−1);//モデル
        名
313 String lcforeignmodelname = foreignmodelname.toLowerCase();//小文字に変換
314 if(colum[0].indexOf(lcforeignmodelname) != −1){
315 for(int l=1;l<foreigntabledata.length;l++){
316 String[] foreigncolum = foreigntabledata[l].split("&",0);
317 if(foreigncolum[0].equals("id")){
318
319 }
320 else if(foreigncolum[0].indexOf("_id") != −1){
321
322 }
323 else{
324 pw.println("\t\t<th><?php␣echo␣$this->Paginator->sort('" + foreigncolum[0] + "
        ',␣'" + foreigncolum[0] + "');␣?></th>");
325 }
326 }
327 }
328 }
329 }
330 else if(colum[0].equals("id")){
331
332 }
333 else{
334 pw.println("\t\t<th><?php␣echo␣$this->Paginator->sort('" + colum[0] + "',␣'"
        + colum[0] + "');␣?></th>");
335 }
336 }
337 pw.println("\t\t<th><?php␣?>");
338 pw.println("\t\t<th␣class=\"actions\">操作</th>");
339 pw.println("\t</tr>\n");
340 pw.println("\t<?php␣foreach␣($" + lcname + "␣as␣$" + lcnames + "):␣?>");
341 pw.println("\t<?php␣//foreach 文（繰り返し）?>");
342 pw.println("\t<tr>");
343 //     if(colum[1].equals("DATE")||colum[1].equals("DATETIME")){
344 //      if(date_count==0){
345 //       pw.println("\t<?php if (date(\"Y/m/d\")<=date(\"Y/m/d\", strtotime($" +
        lcname + "['" + modelname + "'][" + colum[0] + "'])))): ?>");
346 //       date_count++;
```

```
347 //     }
348 //   }
349 for(int j=1;j<tabledata.length;j++) {
350 if(tabledata[j]==null) break;
351 String[] colum = tabledata[j].split("&",0);
352 if(colum[0].indexOf("_id") != −1){//外部キーが存在したら
353 for(int k=0;;k++){
354 if(erdata[k]==null) break;
355 String[] foreigntabledata = erdata[k].split(",",0);
356 String foreignmodelname = foreigntabledata[0].substring(0, foreigntabledata[0].length
        ()−1);//モデル
        名
357 String lcforeignmodelname = foreignmodelname.toLowerCase();//小文字に変換
358 if(colum[0].indexOf(lcforeignmodelname) != −1){
359 for(int l=1;l<foreigntabledata.length;l++){
360 String[] foreigncolum = foreigntabledata[l].split("&",0);
361 if(foreigncolum[1].equals("DATE")){
362 pw.println("\t\t<td><?php␣echo␣h(date('Y/m/d',␣strtotime($" + lcnames + "['"
        + foreignmodelname + "'][' + foreigncolum[0] + "'])));␣?> </td>");
363 }
364 else if(foreigncolum[1].equals("DATETIME")){
365 pw.println("\t\t<td><?php␣echo␣h(date('Y/m/d␣H:i',␣strtotime($" + lcnames +
        "['" + foreignmodelname + "'][' + foreigncolum[0] + "'])));␣?> </td>
        ");
366 }
367 else if(foreigncolum[0].equals("id")){
368
369 }
370 else if(foreigncolum[0].indexOf("_id") != −1){
371
372 }
373 else{
374 pw.println("\t\t<td><?php␣echo␣h($" + lcnames + "['" + foreignmodelname + "
        '][' + foreigncolum[0] + "']);␣?> </td>");
375 }
376 }
377 }
378 }
379 }
380 else if(colum[0].equals("id")){
381
382 }
383 else {
384 pw.println("\t\t<td><?php␣echo($" + lcnames + "['" + modelname + "'][' +
        colum[0] + "']);␣?> </td>");
385 }
386 }
387 pw.println("\t\t<td␣class=\"actions\">");
388 pw.println("\t\t\t<?php␣echo␣$this->Html->link('編集',␣array('action'␣=>␣'
        edit',␣$" + lcnames + "['" + modelname + "'][' id']));␣?>");
389 pw.println("\t\t\t<?php␣//Html ヘルパーを利用␣?>");
390 pw.println("\t\t\t<?php␣echo␣$this->Form->postLink('削除',␣array('action'␣=>
        ␣'delete',␣$" + lcnames + "['" + modelname + "'][' id']),␣null,␣'本当に
        削除しますか？');␣?>");
391 pw.println("\t\t\t<?php␣//Form ヘルパーを利用、削除確認画面が表示される␣?>");
392 pw.println("\t\t</td>");
393 pw.println("\t</tr>");
394 pw.println("\t<?php␣endforeach;␣?>");
395 pw.println("\t<?php␣//foreach 文終わり␣?>");
396 pw.println("\t</table>\n");
397 pw.println("\t<div␣class=\"paging\">");
```

```
398 pw.println("\t<?php");
399 pw.println("\t\techo␣$this->Paginator->prev('<␣'␣.␣'前のページ',␣array(),␣
        null,␣array('class'␣=>␣'prev␣disabled'));");
400 pw.println("\t\techo␣$this->Paginator->numbers(array('separator'␣=>␣''));");
401 pw.println("\t\techo␣$this->Paginator->next('次のページ'␣.␣'␣>',␣array(),␣
        null,␣array('class'␣=>␣'next␣disabled'));");
402 pw.println("\t?>");
403 pw.println("\t</div>");
404 pw.println("</div>\n");
405 pw.println("<div␣class=\"actions\">");
406 pw.println("\t<h3>画面</h3>");
407 pw.println("\t<ul>");
408 pw.println("\t\t<li><?php␣echo␣$this->Html->link('index',␣array('action'␣=>
        ␣'index'));␣?></li>");
409 pw.println("\t</ul>");
410 pw.println("</div>\n");
411 }
412 if(name.equals("view")) {
413 pw.println("<div␣class=\""+␣lcname␣+␣" view\">");
414 pw.println("<h2>詳細</h2>");
415 pw.println("\t<dl>");
416 for(int j=1;j<tabledata.length;j++) {
417 if(tabledata[j]==null) break;
418 String[] colum = tabledata[j].split("&",0);
419 if(colum[0].indexOf("_id") != −1){//外部キーが存在したら
420 for(int k=0;;k++){
421 if(erdata[k]==null) break;
422 String[] foreigntabledata = erdata[k].split(",",0);
423 String foreignmodelname = foreigntabledata[0].substring(0, foreigntabledata[0].length
        ()−1);//モデル
        名
424 String lcforeignmodelname = foreignmodelname.toLowerCase();//小文字に変換
425 if(colum[0].indexOf(lcforeignmodelname) != −1){
426 for(int l=1;l<foreigntabledata.length;l++){
427 String[] foreigncolum = foreigntabledata[l].split("&",0);
428 if(foreigncolum[1].equals("DATE")){
429 pw.println("\t\t<dt>" + foreigncolum[0] + "</dt>");
430 pw.println("\t\t<dd>");
431 pw.println("\t\t\t<?php␣echo␣h(date('Y/m/d',␣strtotime($" + lcname + "['" +
        foreignmodelname + "']['" + foreigncolum[0] + "'])));␣?>");
432 pw.println("\t\t</dd>");
433 }
434 else if(foreigncolum[1].equals("DATETIME")){
435 pw.println("\t\t<dt>" + foreigncolum[0] + "</dt>");
436 pw.println("\t\t<dd>");
437 pw.println("\t\t\t<?php␣echo␣h(date('Y/m/d␣H:i',␣strtotime($" + lcname + "['
        " + foreignmodelname + "']['" + foreigncolum[0] + "'])));␣?>");
438 pw.println("\t\t</dd>");
439 }
440 else if(foreigncolum[0].equals("id")){
441
442 }
443 else if(foreigncolum[0].indexOf("_id") != −1){
444
445 }
446 else{
447 pw.println("\t\t<dt>" + foreigncolum[0] + "</dt>");
448 pw.println("\t\t<dd>");
449 pw.println("\t\t\t<?php␣echo␣h($" + lcname + "['" + foreignmodelname + "'][']
        + foreigncolum[0] + "']);␣?>");
450 pw.println("\t\t</dd>");
```

```
451 }
452 }
453 }
454 }
455 }
456 else if(colum[0].equals("id")){
457
458 }
459 else {
460 if(colum[1].equals("DATE")){
461 pw.println("\t\t<dt>" + colum[0] + "</dt>");
462 pw.println("\t\t<dd>");
463 pw.println("\t\t\t<?php␣echo␣h(date('Y/m/d',␣strtotime($" + lcname + "['" +
        modelname + "'][" + colum[0] + "'])));␣?>");
464 pw.println("\t\t</dd>");
465 }
466 else if(colum[1].equals("DATETIME")){
467 pw.println("\t\t<dt>" + colum[0] + "</dt>");
468 pw.println("\t\t<dd>");
469 pw.println("\t\t\t<?php␣echo␣h(date('Y/m/d␣H:i',␣strtotime($" + lcname + "['
        " + modelname + "'][" + colum[0] + "'])));␣?>");
470 pw.println("\t\t</dd>");
471 }
472 else{
473 pw.println("\t\t<dt>" + colum[0] + "</dt>");
474 pw.println("\t\t<dd>");
475 pw.println("\t\t\t<?php␣echo␣h($" + lcname + "['" + modelname + "'][" +
        colum[0] + "']);␣?>");
476 pw.println("\t\t</dd>");
477 }
478 }
479 }
480 pw.println("\t</dl>");
481 pw.println("</div>\n");
482 pw.println("<div␣class=\"actions\">");
483 pw.println("\t<h3>画面</h3>");
484 pw.println("\t<ul>");
485 pw.println("\t\t<li><?php␣echo␣$this->Html->link('index',␣array('action'␣=>
        ␣'index'));␣?></li>");
486 pw.println("\t</ul>");
487 pw.println("</div>");
488 }
489 if(name.equals("delete_all")){
490 pw.println("<div␣class=\""␣+␣lcname␣+␣" form\">");
491 pw.println("<?php␣echo␣$this->Form->create('" + modelname + "');␣?>");
492 pw.println("\t<fieldset>");
493 pw.println("\t\t<legend>" + lcname + "テーブルの中身を全削除しますか？
        </legend>");
494 pw.println("\t</fieldset>");
495 pw.println("<?php␣echo␣$this->Form->submit('はい');␣?>");
496 pw.println("</div>\n");
497 pw.println("<div␣class=\"actions\">");
498 pw.println("\t<h3>画面</h3>");
499 pw.println("\t<ul>");
500 pw.println("\t\t<li><?php␣echo␣$this->Html->link('index',␣array('action'␣=>
        ␣'index'));␣?></li>");
501 pw.println("\t</ul>");
502 pw.println("</div>");
503 }
504 if(name.equals("login")){
505 pw.println("<div␣class=\""␣+␣lcname␣+␣" form\">");
```

```
506 pw.println("<?php␣echo␣$this->Form->create('" + modelname + "');␣?>");
507 pw.println("<?php␣//Form ヘルパーを利用?>");
508 pw.println("\t<fieldset>");
509 pw.println("\t\t<legend>認証画面</legend>");
510 pw.println("\t<?php");
511 //    for(int i=0;;i++){
512 //     if(erdata[i]==null) break;
513 //     String[] tablecolums = erdata[i].split(",", 0);
514 //     if(tablecolums[0].equals(modelname + "s")){
515 //      for(int j=1;;j++){
516 //       if(tablecolums[j].indexOf("id") == -1){
517 //        columname = tablecolums[j].split("&", 0);
518 //        break;
519 //       }
520 //      }
521 //     }
522 //    }
523 pw.println("\t\techo␣$this->Form->input('username',␣array('label'␣=>␣'
        username'));");
524 pw.println("\t\techo␣$this->Form->input('password',␣array('label'␣=>␣'
        password'));");
525 pw.println("\t\t//入力フォームを表示");
526 pw.println("\t?>");
527 pw.println("\t</fieldset>");
528 pw.println("<?php␣echo␣$this->Form->end('送信');␣?>");
529 pw.println("<?php␣//フォームの内容をPOST 送信する␣?>");
530 pw.println("</div>\n");
531 pw.println("<div␣class=\"actions\">");
532 pw.println("\t<h3>画面</h3>");
533 pw.println("\t<ul>");
534 pw.println("\t\t<li><?php␣echo␣$this->Html->link('index',␣array('action'␣=>
        ␣'index'));␣?></li>");
535 pw.println("\t</ul>");
536 pw.println("</div>");
537 }
538
539
540 //   for(int j=0;;j++){//デバッグ用
541 //    if(tabledata[j]==null) break;
542 //    pw.println(tabledata[j]+"\n");
543 //   }
544
545 pw.close();
546 System.out.println(name + ".ctp にコードを記述しました");
547 }else {
548 System.out.println(name + ".ctp にコードを記述できませんでした");
549 }
550 }catch(IOException e) {
551 System.out.println(e);
552 }
553 }
554
555 private static boolean checkBeforeWritefile(File file) {
556 if(file.exists()) {
557 if (file.isFile() && file.canWrite()) {
558 return true;
559 }
560 }
561
562 return false;
563 }
```

```
564 }
```

ソースコード 9: WriteModel .java

```
1  package cakephpgenerator;
2
3  import java.io.BufferedWriter;
4  import java.io.File;
5  import java.io.FileWriter;
6  import java.io.IOException;
7  import java.io.PrintWriter;
8
9  //モデル書き込み用クラス
10 public class WriteModel {
11 String modelPath;
12
13 void writeCode(String table, String[] array, String[] array2, String[] stdata, String[]
       erdata) {
14 String FK[] = new String[50];
15 String FK2[] = new String[50];
16 String modelname;
17 String validateData = null;
18 int flag = 0;
19 int flag_validate=0;
20
21 try {
22 File file = new File(modelPath);
23
24 if (checkBeforeWritefile(file)) {
25 PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(file)));
26
27 pw.println("<?php");
28 pw.println("class␣" + table + "␣extends␣AppModel␣{");
29 pw.println("\tpublic␣$name␣=␣'" + table + "';");
30 if(array[0]!=null) {//外部キーを持っているなら
31 for(int i=0;;i++){
32 if(array[i]==null) break;
33 FK = array[i].split("&",0);//外部キーを「&」で区切って配列として格納
34 int index = FK[0].indexOf("_id");
35 modelname = FK[0].substring(0,index);//「_id」を削除
36 modelname = EditChar.firstCharLerge(modelname);
37 if(FK[2].equals("many_to_one")||FK[2].equals("one_to_one")||FK[2].equals("
       many_to_many")){//多対
       1、あるいは 1対 1、あるいは多対多のとき
38 if(flag==0){
39 pw.println("\tpublic␣$belongsTo␣=␣array(");
40 pw.println("\t\t'"+ modelname +"'␣=>␣array(");
41 flag = 1;
42 } else pw.println("\t\t,'"+ modelname +"'␣=>␣array(");
43 pw.println("\t\t\t'className'␣=>␣'"+ modelname +"',");
44 pw.println("\t\t\t'foreignKey'␣=>␣'"+ FK[0] +"'");
45 pw.println("\t\t)");
46 }
47 }
48 if(flag==1){
49 pw.println("\t);\n");
50 }
51 flag = 0;
52 }
53 if(array2[0]!=null) {//外部キーが持たれているなら
```

```
54 for(int i=0;;i++){
55 if(array2[i]==null) break;
56 FK2 = array2[i].split("&",0);//外部キーを「&」で区切って配列として格納
57 modelname = FK2[0].substring(0, FK2[0].length()−1);
58 if(FK2[3].equals("many_to_one")){//1対多のとき
59 if(flag==0){
60 pw.println("\tpublic␣$hasMany␣=␣array(");
61 pw.println("\t\t'"+ modelname +"'␣=>␣array(");
62 flag = 1;
63 } else pw.println("\t\t,'"+ modelname +"'␣=>␣array(");
64 pw.println("\t\t\t'className'␣=>␣'"+ modelname +"',");
65 pw.println("\t\t\t'foreignKey'␣=>␣'"+ FK2[1] +"'");
66 pw.println("\t\t)");
67 }
68 }
69 if(flag==1){
70 pw.println("\t);\n");
71 }
72 flag = 0;
73 for(int i=0;;i++){
74 if(array2[i]==null) break;
75 FK2 = array2[i].split("&",0);//外部キーを「&」で区切って配列として格納
76 modelname = FK2[0].substring(0, FK2[0].length()−1);
77 if(FK2[3].equals("one_to_one")){//1対 1のとき
78 if(flag==0){
79 pw.println("\tpublic␣$hasOne␣=␣array(");
80 pw.println("\t\t'"+ modelname +"'␣=>␣array(");
81 flag = 1;
82 } else pw.println("\t\t,'"+ modelname +"'␣=>␣array(");
83 pw.println("\t\t\t'className'␣=>␣'"+ modelname +"',");
84 pw.println("\t\t\t'foreignKey'␣=>␣'"+ FK2[1] +"'");
85 pw.println("\t\t)");
86 }
87 }
88 if(flag==1){
89 pw.println("\t);\n");
90 }
91 flag = 0;
92 for(int i=0;;i++){
93 if(array2[i]==null) break;
94 FK2 = array2[i].split("&",0);//外部キーを「&」で区切って配列として格納
95 modelname = FK2[0].substring(0, FK2[0].length()−1);
96 if(FK2[3].equals("many_to_many")){//多対多のとき
97 if(flag==0){
98 pw.println("\tpublic␣$hasAndBelongsToMany␣=␣array(");
99 pw.println("\t\t'"+ modelname +"'␣=>␣array(");
100 flag = 1;
101 } else pw.println("\t\t,'"+ modelname +"'␣=>␣array(");
102 pw.println("\t\t\t'className'␣=>␣'"+ modelname +"',");
103 pw.println("\t\t\t'foreignKey'␣=>␣'"+ FK2[1] +"'");
104 pw.println("\t\t)");
105 }
106 }
107 if(flag==1){
108 pw.println("\t);\n");
109 }
110 flag = 0;
111 }
112
113 //バリデーション周りの記述
114 for(int i=0;;i++){
```

```
115 if(erdata[i]==null) break;
116 String[] name = erdata[i].split(",", 0);
117 if(table.equals(EditChar.deleteLastChar(name[0]))){
118 for(int j=0;j<name.length;j++){
119 if(name[j].indexOf("@notnull") != -1||name[j].indexOf("DATETIME") != -1||name[j].
        indexOf("DATE") != -1){
120 if(!name[j].startsWith("id")&&name[j].indexOf("_id")==-1){
121 validateData = erdata[i];
122 flag_validate = 1;
123 }
124 }
125 }
126 }
127 }
128 if(flag_validate==1){
129 String[] splitData = validateData.split(",", 0);
130 pw.println("\tpublic␣$validate␣=␣array(");
131 for(int i=0;i<splitData.length;i++){
132 if(!splitData[i].startsWith("id")&&splitData[i].indexOf("_id")==-1){
133 if(splitData[i].indexOf("DATETIME")!=-1||splitData[i].indexOf("DATE")!=-1){
134 String[] splitand = splitData[i].split("&", 0);
135 String[] splitat = splitand[0].split("@", 0);
136 if(flag_validate == 1){
137 pw.println("\t\t'" + splitat[0] + "'␣=>␣'date'");
138 flag_validate = 2;
139 } else pw.println("\t\t,'" + splitat[0] + "'␣=>␣'date'");
140 }
141 else if(splitData[i].indexOf("@notnull")!=-1){
142 String[] splitand = splitData[i].split("&", 0);
143 String[] splitat = splitand[0].split("@", 0);
144 if(flag_validate == 1){
145 pw.println("\t\t'" + splitat[0] + "'␣=>␣'notEmpty'");
146 flag_validate = 2;
147 } else pw.println("\t\t,'" + splitat[0] + "'␣=>␣'notEmpty'");
148 }
149 }
150 }
151 pw.println("\t);");
152 }
153
154 //ここでログイン機能周りのことを記述 (ハッシュ化を行う)
155 for(int i=0;;i++){
156 if(stdata[i]==null) break;
157 String[] tabledata = stdata[i].split(",", 0);
158 if(tabledata[0].equals(table + "s")){
159 for(int j=1;tabledata.length>j;j++){
160 if(tabledata[j].equals("login")){
161 pw.println("\tpublic␣function␣beforeSave($options␣=␣array())␣{");
162 pw.println("\t\t$this->data['" + table + "']['password']␣=␣AuthComponent::
        password($this->data['" + table + "']['password']);");
163 pw.println("\t\treturn␣true;");
164 pw.println("\t}");
165 }
166 }
167 }
168 }
169
170 pw.println("}");
171 pw.println("?>");
172
173 pw.close();
```

```
174 System.out.println(table + ".php にコードを記述しました");
175 } else {
176 System.out.println(table + ".php にコードを記述できませんでした");
177 }
178 } catch(IOException e) {
179 System.out.println(e);
180 }
181 }
182
183 private static boolean checkBeforeWritefile(File file){
184 if (file.exists()) {
185 if (file.isFile() && file.canWrite()) {
186 return true;
187 }
188 }
189
190 return false;
191 }
192 }
```

図 目 次

表 目 次