# 16. 開発フレームワークに関する知識 I

## 1. 科目の概要

開発フレームワークとは何か、その基本的な概念、歴史、特徴とメリットを解説する。さらに開発フレームワークを利用して開発を行う際に必要な技術を説明し、実際の開発に役立てることができる知識やノウハウを示す。

## 2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

| 習得ポイント | 説　明 | シラバスの対応コマ |
|---|---|---|
| I-16-1. 開発フレームワークの歴史、思想と背景 | 開発フレームワークとは何かを説明し、開発フレームワークの基本理念、目的、発展の歴史、開発フレームワークが登場した背景と思想について解説する。 | 1 |
| I-16-2. 開発フレームワークの特徴とメリット | 開発フレームワークを利用した開発の特徴と、開発フレームワーク利用の効果、メリットについて解説する。また代表的な開発フレームワークを紹介し、それぞれの歴史や背景、特徴、利用上の注意点、開発フレームワークの選択方法、ライセンス形態などを説明する。 | 1,2 |
| I-16-3. Webアプリケーション構築で利用できるOSSフレームワーク | オープンソースによるWebアプリケーション開発フレームワークを紹介する。Struts、JSF、IBATIS、Hibernate、Seasar 2、Springといった代表的なフレームワークを紹介し、それぞれの開発内容の違いについて解説する。 | 3 |
| I-16-4. Tomcatの特徴とServlet/JSPによるアプリケーション開発 | Java Servlet/JSP コンテナに関する開発フレームワークの代表的なものとして、Tomcatを取り上げ、その位置づけ、特徴とServlet/JSPによるWebアプリケーション開発の具体的な手順を説明する。 | 4 |
| I-16-5. アプリケーションサーバJBossの機能と特徴 | アプリケーションサーバについて、例として代表的なアプリケーションサーバであるJBossを取り上げ、その位置づけ、特徴を解説する。またTomcatとの連携やEJBコンテナ機能、アプリケーションサーバ機能など、JBossの持つ様々な機能を紹介する。 | 4 |
| I-16-6. ツール実行を自動化するAnt | コンパイルやテスト、バージョン管理などソフトウェア開発に関わるあらゆる処理の実行を自動化するツールAntを説明する。Antの基本処理であるコンパイル、ビルドのXMLによる設定方法を示す。また、Antで自動化できるテスト、ファイル転送、リモートホスト操作、バージョン管理などを紹介する。 | 5 |
| I-16-7. Javaプロジェクト管理ツールMaven | Javaソフトウェア開発のコンパイル、ビルド、テスト、パッケージングに至る一連の作業の自動化するプロジェクト管理ツールMavenを解説する。スクリプトを記述する必要が少ないことなどAntに比べて優れた点にも触れる。 | 5 |
| I-16-8. Webアプリケーションのテスト支援ツールCactus | Javaソフトウェアのテスト支援ツールであるJUnitを紹介し、JUnitでテストできないServletやJSPに対応したCactusを説明する。テストコードの記述方法、テストの実行方法を解説する。 | 5 |
| I-16-9. 要求分析モデル(DOA、OOA、Web MVC) | 開発フレームワークによるアプリケーション開発を行う際に実施する要求分析について、DOA (データ中心アプローチ)、OOA (オブジェクト指向分析)、Web MVC (モデルービューーコントローラ)モデルといった各種のモデルの特徴と違いについて解説する。 | 6 |
| I-16-10. Webアプリケーション開発における実際の作業プロセス | Webアプリケーション開発に関して、プレゼンテーション設計、画面遷移設計、ビジネスロジック設計、データベース設計、コンポーネント間インタフェース設計といった実際の実装作業プロセスを説明する。 | 6 |

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念･知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

## 3. IT 知識体系との対応関係

「16. 開発フレームワークに関する知識Ⅰ」と IT 知識体系との対応関係は以下の通り。

| 科目名 | 基本レベル（Ⅰ） | | | | | | 応用レベル（Ⅱ） | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16. 開発フレームワークに関する知識 | <開発フレームワークとは何か> | <開発フレームワークの種類と特徴> | <オープンソースによるWebアプリケーションのフレームワーク> | <フリーのWebコンテナ／J2EEコンテナの概要> | <オープンソースの開発ツール> | <開発フレームワークによる開発プロセスの手順> | <Ruby on Railsによるアプリケーション開発> | <Strutsとは> | <MyFace（JSF）の開発モデルとは> | <データベース接続・アクセスのフレームワーク> | <DIxAOPコンテナの概要> | <Springフレームワーク> | <Seasar2> | <Tapestry> | <Strutsによるアプリケーション開発> |

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_16.pdf]

<IT 知識体系上の関連部分>

| 分野 | | 科目名 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 組織関連事項と情報システム | 1 | IT-IAS 情報保証と情報セキュリティ | IT-IAS1. 基礎的な問題 | IT-IAS2. 情報セキュリティの仕組み（対策） | IT-IAS3. 運用上の問題 | IT-IAS4. ポリシー | IT-IAS5. 攻撃 | IT-IAS6. 情報セキュリティ分野 | IT-IAS7. フォレンジック（情報証拠） | IT-IAS8. 情報の状態 | IT-IAS9. 情報セキュリティサービス | IT-IAS10. 脅威分析モデル | IT-IAS11. 脆弱性 | | |
| | 2 | IT-SP 社会的な観点とプロフェッショナルとしての課題 | IT-SP1. プロフェッショナルとしてのコミュニケーション | IT-SP2. コンピュータの歴史 | IT-SP3. コンピュータを取り巻く社会環境 | IT-SP4. チームワーク | IT-SP5. 知的財産権 | IT-SP6. コンピュータの法的問題 | IT-SP7. 組織の中のIT | IT-SP8. プロフェッショナルとしての倫理的な問題と責任 | IT-SP9. プライバシーと個人の自由 | | | | |
| 応用技術 | 3 | IT-IM 情報管理 | IT-IM1. 情報管理の概念と基礎 | IT-IM2. データベース問合わせ言語 | IT-IM3. データアーキテクチャ | IT-IM4. データモデリングとデータベース設計 | IT-IM5. データと情報の管理 | IT-IM6. データベースの応用分野 | | | | | | | |
| | 4 | IT-WS Webシステムとその技術 | IT-WS1. Web技術 | IT-WS2. 情報アーキテクチャ [16-Ⅰ-6] | IT-WS3. デジタルメディア | IT-WS4. Web開発 | IT-WS5. 脆弱性 | IT-WS6. ソーシャルソフトウェア | | | | | | | |
| ソフトウェアの方法と技術 | 5 | IT-PF プログラミング基礎 | IT-PF1. 基本データ構造 | IT-PF2. プログラミングの基本的構成要素 | IT-PF3. オブジェクト指向プログラミング | IT-PF4. アルゴリズムと問題解決 | IT-PF5. イベント駆動プログラミング | IT-PF6. 再帰 | | | | | | | |
| | 6 | IT-IPT 技術を統合するためのプログラミング | IT-IPT1. システム間通信 | IT-IPT2. データ割り当てと交換 | IT-IPT3. 統合的コーディング | IT-IPT4. スクリプティング手法 | IT-IPT5. ソフトウェアセキュリティの実現 | IT-IPT6. 種々の問題 | IT-IPT7. プログラミング言語の概要 | | | | | | |
| | 7 | CE-SWE ソフトウェア工学 | CE-SWE0. 歴史と概要 | CE-SWE1. ソフトウェアプロセス | CE-SWE2. ソフトウェアの要求と仕様 [16-Ⅰ-6] | CE-SWE3. ソフトウェアの設計 [16-Ⅰ-6] | CE-SWE4. ソフトウェアのテストと検証 | CE-SWE5. ソフトウェアの保守 | CE-SWE6. ソフトウェア開発・保守ツールと環境 | CE-SWE7. ソフトウェアプロジェクト管理 | CE-SWE8. 言語翻訳 | CE-SWE9. ソフトウェアのフォールトトレランス | CE-SWE10. ソフトウェアの構成管理 | CE-SWE11. ソフトェアの標準化 | |
| | 8 | IT-SIA システムインテグレーションとアーキテクチャ | IT-SIA1. 要求仕様 | IT-SIA2. 調達/手配 | IT-SIA3. インテグレーション | IT-SIA4. プロジェクト管理 | IT-SIA5. テストと品質保証 | IT-SIA6. 組織の特性 | IT-SIA7. アーキテクチャ | | | | | | |
| システム基盤 | 9 | IT-NET ネットワーク | IT-NET1. ネットワークの基礎 | IT-NET2. ルーティングとスイッチング | IT-NET3. 物理層 | IT-NET4. セキュリティ | IT-NET5. アプリケーション分野 | IT-NET6. ネットワーク管理 | | | | | | | |
| | 10 | CE-NWK テレコミュニケーション | CE-NWK0. 歴史と概要 | CE-NWK1. 通信ネットワークのアーキテクチャ | CE-NWK2. 通信ネットワークのプロトコル | CE-NWK3. LANとWAN | CE-NWK4. クライアントサーバコンピューティング | CE-NWK5. データのセキュリティと整合性 | CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング | CE-NWK7. データ通信 | CE-NWK8. 組込み機器向けネットワーク | CE-NWK9. 通信技術とネットワーク概要 | CE-NWK10. 性能評価 | CE-NWK11. ネットワーク管理 | CE-NWK12. 圧縮と伸張 |
| | | | CE-NWK13. クラスタシステム | CE-NWK14. インターネットアプリケーション | CE-NWK15. 次世代インターネット | CE-NWK16. 放送 | | | | | | | | | |
| | 11 | IT-PT プラットフォーム技術 | IT-PT1. オペレーティングシステム | IT-PT2. アーキテクチャと機構 | IT-PT3. コンピュータインフラストラクチャ | IT-PT4. デプロイメントソフトウェア | IT-PT5. ファームウェア | IT-PT6. ハードウェア | | | | | | | |
| | 12 | CE-OPS オペレーティングシステム | CE-OPS0. 歴史と概要 | CE-OPS1. 並行性 | CE-OPS2. スケジューリングとディスパッチ | CE-OPS3. メモリ管理 | CE-OPS4. セキュリティと保護 | CE-OPS5. ファイル管理 | CE-OPS6. リアルタイムOS | CE-OPS7. OSの概要 | CE-OPS8. 設計の原則 | CE-OPS9. デバイス管理 | CE-OPS10. システム性能評価 | | |
| コンピュータアーキテクチャとネットワーク | 13 | CE-CAO コンピュータのアーキテクチャと構成 | CE-CA00. 歴史と概要 | CE-CA01. コンピュータアーキテクチャの基礎 | CE-CA02. メモリシステムの構成とアーキテクチャ | CE-CA03. インタフェースと通信 | CE-CA04. デバイスサブシステム | CE-CA05. CPUアーキテクチャ | CE-CA06. 性能・コスト評価 | CE-CA07. 分散・並列処理 | CE-CA08. コンピュータによる計算 | CE-CA09. 性能向上 | | | |
| 複数領域にまたがるもの | 14 | IT-ITF IT基礎 | IT-ITF1. ITの一般的なテーマ | IT-ITF2. 組織の問題 | IT-ITF3. ITの歴史 | IT-ITF4. IT分野（学科）とそれに関連のある分野（学科） | IT-ITF5. 応用領域 | IT-ITF6. IT分野における数学と統計学の活用 | | | | | | | |
| | 15 | CE-ESY 組込みシステム | CE-ESY0. 歴史と概要 | CE-ESY1. 低電力コンピューティング | CE-ESY2. 高信頼性システムの設計 | CE-ESY3. 組込み用アーキテクチャ | CE-ESY4. 開発環境 | CE-ESY5. ライフサイクル | CE-ESY6. 要件分析 | CE-ESY7. 仕様定義 | CE-ESY8. 構造設計 | CE-ESY9. テスト | CE-ESY10. プロジェクト管理 | CE-ESY11. 並行設計（ハードウェア、ソフトウェア | CE-ESY12. 実装 |
| | | | CE-ESY13. リアルタイムシステム設計 | CE-ESY14. 組込みマイクロコントローラ | CE-ESY15. 組込みプログラム | CE-ESY16. 設計手法 | CE-ESY17. ツールによるサポート | CE-ESY18. ネットワーク型組込みシステム | CE-ESY19. インタフェースシステムと混合信号システム | CE-ESY20. センサ技術 | CE-ESY21. デバイスドライバ | CE-ESY22. メンテナンス | CE-ESY23. 専門システム | CE-ESY24. 信頼性とフォールトトレランス | |

## 4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識として、具体的な Java 言語の開発フレームワークに関する知識がある。ここで扱うフレームワークの実装は、一般的な開発フレームワークの考え方を踏襲したものである。

| 科目名 | 第1回 | 第2回 | 第3回 | 第4回 | 第5回 | 第6回 |
|---|---|---|---|---|---|---|
| 16.開発フレームワークに関する知識Ⅰ | (1)開発フレームワークの特徴<br>(2)開発フレームワークの特性<br>(3)どのようにフレームワークを選択し、利用する<br>(4)ライセンス形態 | (1)基本的なアプリケーション記述の例<br>(2)開発フレームワークの特徴<br>(3)フレームワークによる開発のメリット<br>(3)フレームワークによる開発のメリット | (1)Web アプリケーションとオープンソースフレームワーク<br>(2)Web アプリケーション構築で利用される代表的なオープンソースフレームワーク<br>(3)それぞれの開発内容の違い | (1)Tomcat<br>(2)JBoss | (1)Ant<br>(2)Maven<br>(3)Cactus<br>(4)Log4j<br>(5)XDoclet | (1)要件分析の方向<br>(2)実装のプロセス |

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-1. 開発フレームワークの歴史、思想と背景 | |
| 対応する<br>コースウェア | 第 1 回（開発フレームワークとは何か） | |

# I-16-1. 開発フレームワークの歴史、思想と背景

開発フレームワークとは何かを説明し、開発フレームワークの基本理念、目的、発展の歴史、開発フレームワークが登場した背景と思想について解説する。

【学習の要点】

* 開発フレームワークとは、アプリケーションソフトを開発する際に頻繁に必要とされる汎用的な機能をまとめて提供し、アプリケーションの土台として機能するソフトウェアのことである。
* 開発フレームワークは、ソフトウェア開発におけるさまざまな経験を踏まえ、開発生産性を向上させようとする試みの中で、長い時間をかけて発展してきた。

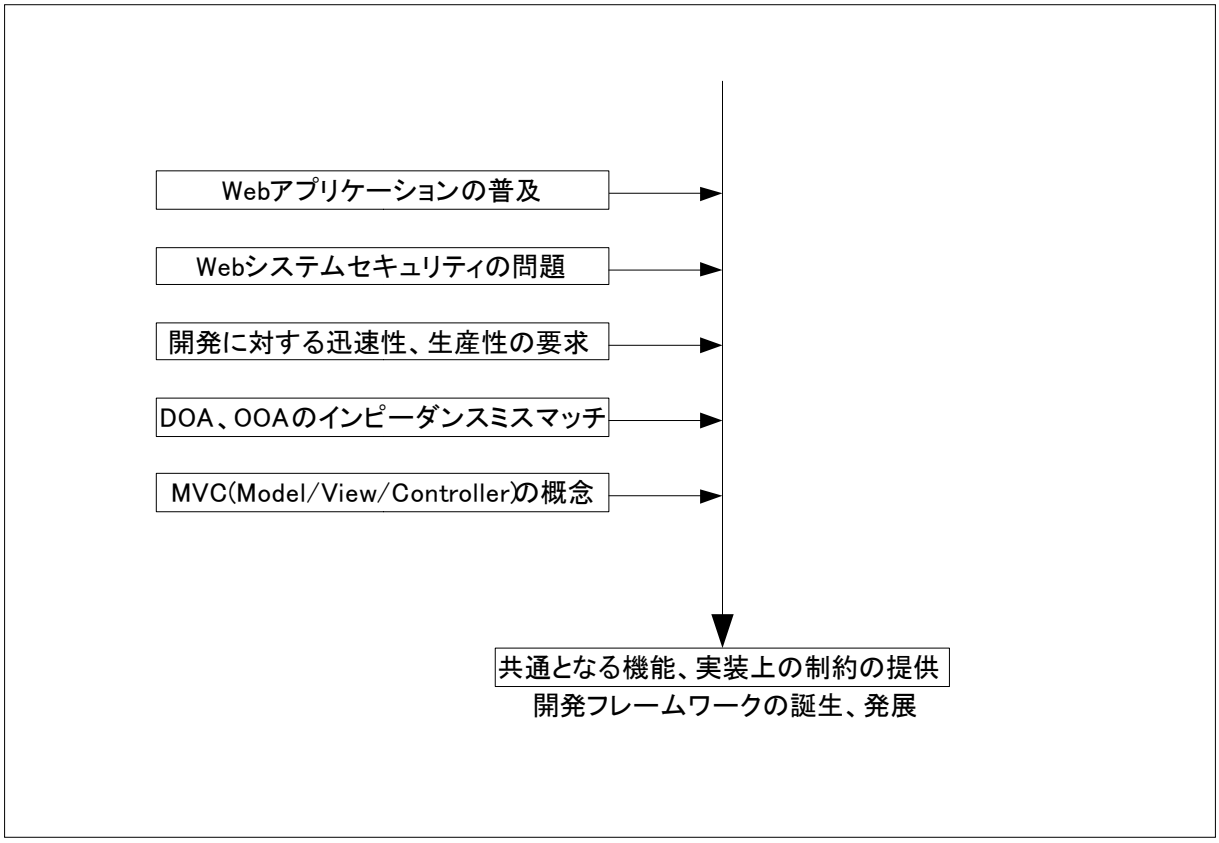


図 I-16-1.　開発フレームワークの背景

【解説】

1) 開発フレームワークとは

開発フレームワークとは、アプリケーションを開発する場合に必要となる部品や、実装上の制約などを提供するもので、ソフトウェア開発における生産性や保守性の向上を目的としたものである。アプリケーションの骨組み部分が、よく洗練された形であらかじめ提供されるため、開発担当者のスキルへ依存する部分を減らすことで、設計レベルの欠陥を最小限に止め、一定の成果物品質を維持することができる。

2) 開発フレームワークの基本構成

開発フレームワークは、フローズンスポットとホットスポットから構成される。

＊ フローズンスポット
アプリケーションによらず常に必要となる機能を有した部分であり、固有のアプリケーションを開発する際には変更を加えない部分。開発フレームワークでは、これらの機能をそのまま提供する。

＊ ホットスポット
アプリケーションによって個別に開発される部分。開発フレームワークでは、この部分を実装上の制約という形で提供する。フレームワークが規定する制約に則ってホットスポット部分を実装することにより、初めて開発フレームワークの利点を享受することができる。

フローズンスポットの占める割合がホットスポットのそれに対して大きければ大きいほど、固有のアプリケーション開発時の工数は少なくなり、より高い開発生産性を得られる。しかしそれと同時に、フレームワークによる制約が多くなるため、そのフレームワークを適用できるアプリケーションは限定され、汎用性を失ってしまう。このように、フレームワークの汎用性と、そのフレームワークを適用したことにより得られる開発生産性の向上度合いは、常にトレードオフの関係にある。

3) 開発フレームワーク発展の背景

開発フレームワークは、ソフトウェア開発において先人の知恵を再利用し、開発生産性を向上させようとする試みの中で、長い時間をかけて発展してきたものである。特に GUI アプリケーションの開発における定型処理を標準化しようとする動きの中で、1980 年に発表された統合開発環境である、Smalltalk において取り入れられた MVC(Model/View/Controller)の概念は、現在では GUI アプリケーションのみならず、Web アプリケーション開発に用いられるフレームワークにも大きな影響を与えている。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-2. 開発フレームワークの特徴とメリット | |
| 対応する<br>コースウェア | 第 1 回（開発フレームワークとは何か）<br>第 2 回（開発フレームワークの種類と特徴） | |

# I-16-2. 開発フレームワークの特徴とメリット

開発フレームワークを利用した開発の特徴と、開発フレームワーク利用の効果、メリットについて解説する。また代表的な開発フレームワークを紹介し、それぞれの歴史や背景、特徴、利用上の注意点、開発フレームワークの選択方法、ライセンス形態などを説明する。

【学習の要点】

＊ Web MVC フレームワークを導入すれば、Controller 部分を記述することなく、Model と View とを記述すれば済む。

＊ O/R マッピングフレームワークを導入すれば、オブジェクトと関係データベースとのインピーダンスミスマッチを解消できる。

＊ DI コンテナを導入すれば、オブジェクト間の依存関係の管理を DI コンテナに移譲でき、オブジェクト間の依存性を疎に保つことができる。

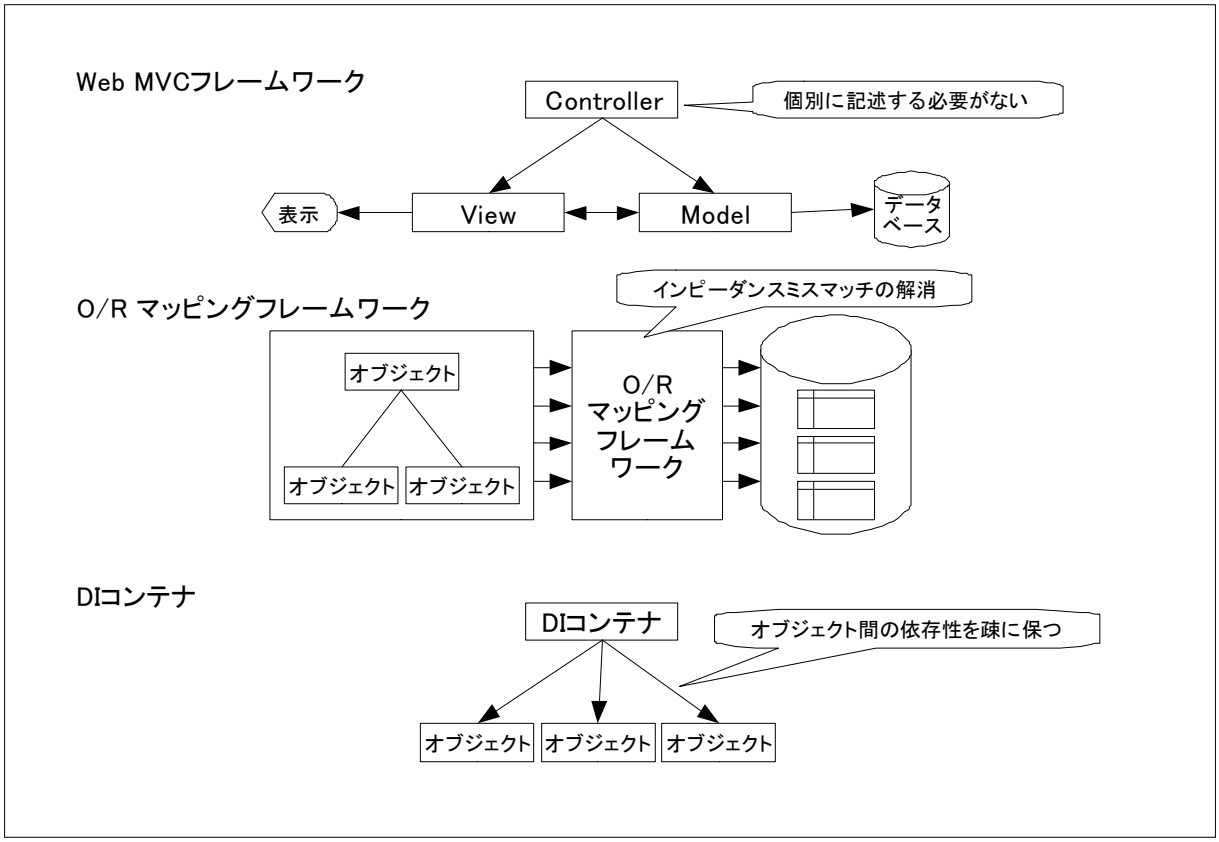


図 I-16-2.　代表的なフレームワーク

【解説】

1）開発フレームワークの種類と特徴

Web アプリケーション開発でよく用いられている開発フレームワークには、Web MVC フレームワーク、O/R マッピングフレームワーク、DI コンテナがある。

＊ Web MVC フレームワーク

Web MVC フレームワークは、MVC モデルを適用した Web アプリケーション用の開発フレームワークである。一般的な Web MVC フレームワークでは、Controller をフローズンスポットとしてフレームワーク側で提供し、残りの Model と View をホットスポットとしてアプリケーションごとに開発する。Web MVC フレームワークの代表的なものとして Struts があり、Struts が普及して以来、PHP の Zend Framework や Ruby の Ruby on Rails など、他の言語でも Web MVC フレームワークが開発され、普及していった。

＊ O/R(オブジェクト/関係)マッピングフレームワーク

O/R マッピングフレームワークは、オブジェクト指向言語におけるオブジェクトと関係データベースのデータ構造の差異を吸収し、自動変換を行う機能を有した開発フレームワークである。オブジェクト指向言語で関係データベースを扱う場合、データモデルの設計思想の違いから、オブジェクトとデータベースデータとのマッピングのために、毎回煩雑な処理を行わなければならない。このデータモデルの設計思想の違いをインピーダンスミスマッチというが、O/R マッピングフレームワークでは、このマッピングにおける定型処理をフローズンスポットとして提供し、インピーダンスミスマッチを解消する手助けをする。Java では元々、インピーダンスミスマッチを解消するための仕組みとして、EJB(Enterprise Java Beans)における Entity Bean があった。しかしこの EJB は、分散オブジェクトとしての性質も併せ持つなど、単なるオブジェクトとリレーショナルデータのマッピングツールとしては複雑過ぎ、中小規模の開発を主とする多くの開発者には受け入れられなかった。このような中、もっと簡単に O/R マッピングを実現したいという要望から、Apache Torque をはじめ、現在よく使われている Hibernate や iBATIS などの O/R マッピングフレームワークが開発され、次いで他の言語でも ActiveRecord など同様のものが開発されていった。なお、現在の EJB 最新仕様である EJB3 では、以上の教訓から、Hibernate などの O/R マッピングフレームワークの影響を大きく受けており、従来に比べ大幅に利用しやすくなっている。

＊ DI(Dependency Injection)コンテナ

DI コンテナは、オブジェクト間の依存関係を設定ファイルに記述することで、依存関係の管理を DI コンテナに移譲することができる仕組みを持つ開発フレームワークである。IoC(Inversion of Control)コンテナ、軽量コンテナ、DI フレームワークなどとも呼ばれる。他のオブジェクトを呼び出す際には抽象的な名前とインタフェースのみを利用し、どの具象クラスを用いてどのような初期化を行ったオブジェクトを用いるかは、全て DI コンテナが外部から注入してくるのに任せる。このような仕組みを利用して開発することで、他の具象クラスに依存したコードを排除し、オブジェクト間の依存性を疎に保つことができる。Java における EJB も、同様の機能を有するが、O/R マッピングフレームワークの項でも述べた通り、EJB は汎用的な反面、非常に複雑で扱いづらい仕様であった。DI コンテナの代表的な実装として Spring があるが、Spring はこの複雑すぎる EJB の代替として、個人により開発が開始され、以降オープンソースプロジェクトとして急速に成長し、普及していった。日本でも、オープンソースのプロジェクトで開発が進められている DI コンテナとして、Seasar2 がある。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-3. Web アプリケーション構築で利用できる OSS フレームワーク | |
| 対応する<br>コースウェア | 第 3 回 (オープンソースによる Web アプリケーションのフレームワーク) | |

# I-16-3. Web アプリケーション構築で利用できる OSS フレームワーク

オープンソースによる Web アプリケーション開発フレームワークを紹介する。Struts、JSF、IBATIS、Hibernate、Seasar II、Spring といった代表的なフレームワークを紹介し、それぞれの開発内容の違いについて解説する。

【学習の要点】

* Struts は Java の Web MVC フレームワークとして広く利用されている。
* Hibernate は Java の O/R マッピングフレームワークとして広く利用されている。
* IBATIS は Hibernate などと異なり SQL を直接記述できるので、柔軟なデータベースアクセスが可能である。
* Seasar2 は国産の DI コンテナで、日本語ドキュメントが非常に豊富である。

| 種類 | 名称 | 特徴 |
|---|---|---|
| Web MVC<br>フレームワーク | Struts<br>JSF (Sunの参照実装)<br>Apache MyFaces<br>ICEFaces | Servlet/JSPを利用、設定をXMLで記述<br>JSF準拠<br>JSF準拠<br>JSF準拠 |
| O/Rマッピング<br>フレームワーク | Hibernate<br>IBATIS | O/RマッピングをXMLで記述<br>SQL文を直接記述可能 |
| DIコンテナ | Seasar2<br>Spring Framework | 国産、AOPサポート、Less Configuration思想<br>AOPサポート、MVCフレームワーク内包 |

図 I-16-3.　開発フレームワークの主な OSS 実装

【解説】

1) OSS の Web MVC フレームワーク

＊ Struts(Apache Struts)　　http://struts.apache.org/
Struts は、Servlet と JSP の技術を用いて Web アプリケーションを開発する際に利用する、Web MVC フレームワークで、Apache ソフトウェア財団のトップレベルプロジェクトとして開発が進められている。Struts を利用して Web アプリケーションを開発する際には、Model 部分である Action クラス、View 部分である JSP ページと View に埋め込まれる ActionForm クラスを作成し、これらの関連を XML の設定ファイルに記述する。

＊ JSF(JavaServer Faces)　　http://java.sun.com/javaee/javaserverfaces/
JSF は、JSR(Java Specification Request)に定められた Java における Web MVC フレームワークの標準仕様である。JSF では、Web ページ表示を構成するコンポーネントを UI コンポーネントと呼び、高度に抽象化されている。また、標準で JSP カスタムタグライブラリを提供しているが、View 部分に JSP を強制するわけではなく、別のものに置き換えることも可能である。JSF の代表的な OSS 実装は以下の通りである。

- Sun の参照実装　　http://java.sun.com/javaee/javaserverfaces/download.html
- Apache MyFaces　　http://myfaces.apache.org/
- ICEFaces　　http://www.icefaces.org/main/home/index.jsp

2) OSS の O/R マッピングフレームワーク

＊ Hibernate　　http://www.hibernate.org/
Hibernate は、オブジェクトのプロパティと関係データベースのカラムとのマッピングを XML で記述しておくことで、O/R マッピングを Hibernate に移譲できる。

＊ IBATIS　　http://ibatis.apache.org/
IBATIS は Apache ソフトウェア財団のトップレベルプロジェクトとして開発が進められている。Hibernate などと異なり、SQL 文と、その SQL 文を呼び出すメソッド名、および SQL 文の実行結果として取得された結果セットのマッピング先クラス名を XML で記述し、O/R マッピングを実現する。SQL 文を直接記述できるため、柔軟なデータベースアクセスを行いつつ、ソースコードからはインピーダンスミスマッチの問題を排除できる。

3) OSS の DI コンテナ

＊ Seasar2　　http://s2container.seasar.org/
Seasar2 は、DI と AOP(アスペクト指向プログラミング)をサポートした軽量コンテナである。O/R マッピングフレームワークである S2Dao や S2JDBC や、JSF 実装の Teeda など、サブプロジェクトで連携先エンジンが開発されているほか、S2JSF や S2Struts、S2Hibernate などを使用することにより、他のエンジンとシームレスに統合できる。設計思想として Less Configuration を掲げており、大抵のアプリケーションで同様であろう部分は、極力設定をしなくても動作するように意識されている。国産であるため、日本語ドキュメントが非常に豊富である。

＊ Spring Framework　　http://www.springframework.org/
Spring Framework は、DI と AOP をサポートした軽量コンテナで、JDBC によるデータベースアクセスを抽象化するレイヤや MVC フレームワークを内包している。また Struts や Hibernate との連携機能を持つ。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-4. Tomcat の特徴と Servlet/JSP によるアプリケーション開発 | |
| 対応する<br>コースウェア | 第 4 回（フリーの Web コンテナ/J2EE コンテナの概要） | |

# I-16-4. Tomcat の特徴と Servlet/JSP によるアプリケーション開発

Java Servlet/JSP コンテナの代表的なものとして、Tomcat を取り上げ、その位置づけ、特徴と Servlet/JSP による Web アプリケーション開発の具体的な手順を説明する。

【学習の要点】

* Tomcat は Java Servlet/JSP コンテナとして、高いシェアを誇り、事実上の標準となっている。
* Tomcat の実行モードには、スタンドアロン、内部プロセス、外部プロセスの 3 つがあるが、内部プロセスにより、Apache HTTP Server と連携させる方法が主流である。

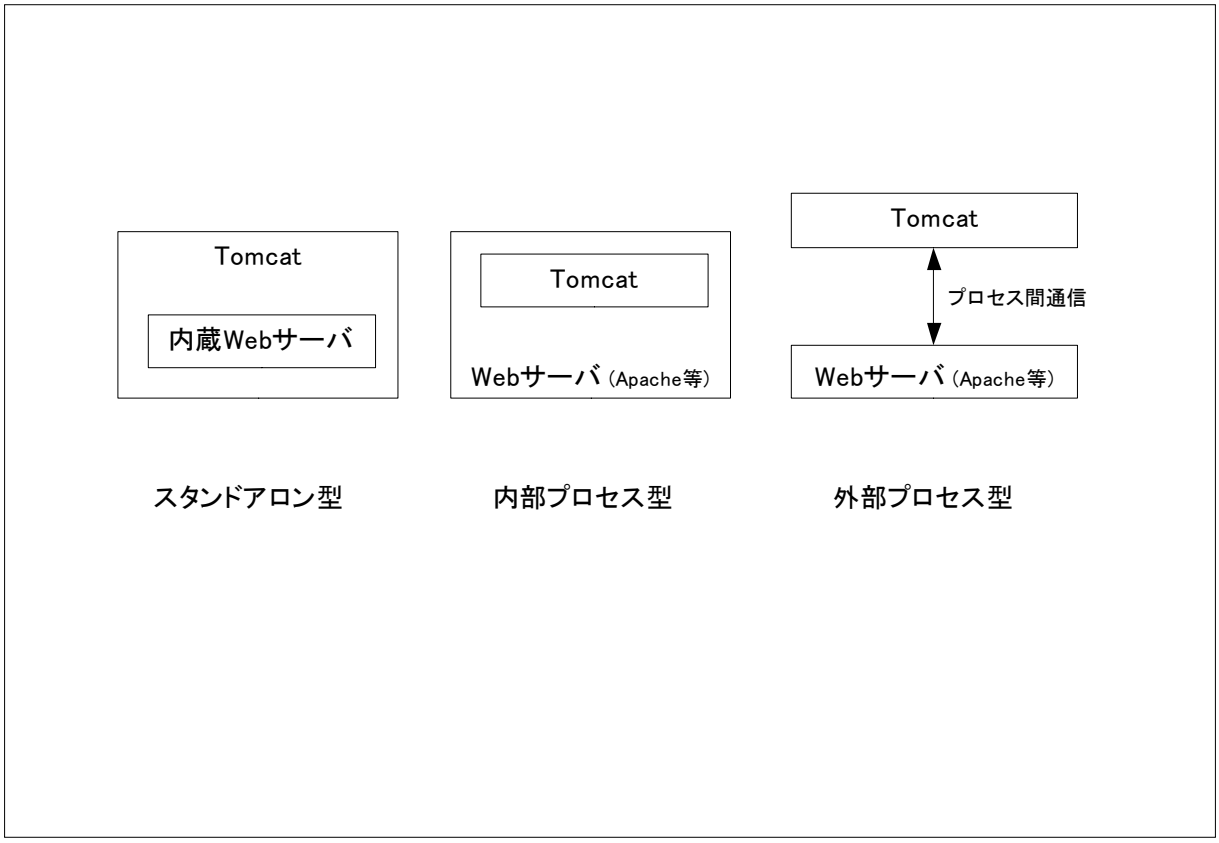


図 I-16-4.　Tomcat の実行モード

【解説】

1) Tomcat の特徴

http://tomcat.apache.org/

Tomcat (Apache Tomcat)は、Java Servlet と JSP のコンテナ(実行環境)であり、Java Servlet と JSP 技術の公式な参照実装(ソースコードレベルで参照できる実装)として使われている。ASF(Apache ソフトウェア財団)が運営するプロジェクトで開発されている OSS であり、Apache ソフトウェアライセンスに基づいてリリースされている。

2) Tomcat の位置づけ

Tomcat は実行モードによって 3 つの位置づけがある。Apache HTTP Server と連携する内部プロセス型がもっとも多く利用されている。

＊ スタンドアロン型
Tomcat 自体に Web サーバの機能が内蔵されており、Java ベースの Web アプリケーションであれば、別途 Web サーバを導入することなく利用することができる。

＊ 内部プロセス型
他の Web サーバ上で Tomcat を実行することで、その Web サーバと連携する。

＊ 外部プロセス型
他の Web サーバとは別に Tomcat を実行し、プロセス間通信によってその Web サーバと連携する。内部プロセス型に比べ、処理速度は落ちるが、安定性や拡張性の点でメリットがある。

3) Servlet/JSP による Web アプリケーション開発

Servlet/JSP による Web アプリケーションを Tomcat で動作させるには、基本的には以下のような手順で行う。

＊ アプリケーション用ディレクトリ作成
アプリケーションを配置するディレクトリを作成し、アプリケーション(JSP ファイルや class ファイル)を配置する。

＊ WEB-INF 作成
WEB-INF ディレクトリを作成し、WEB-INF ディレクトリ下に web.xml を作成する。web.xml には、Servlet/JSP のマッピング情報などを記述する。

＊ server.xml 編集
Tomcat をインストールしたディレクトリ下の conf/server.xml を編集し、アプリケーション用ディレクトリを登録する。

＊ 他の Web サーバの設定
他の Web サーバと連携する場合、その Web サーバ経由でアクセスできるよう設定する。

＊ Tomcat 再起動
Tomcat を再起動する。他の Web サーバと連携する場合、その Web サーバも再起動する。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-5. アプリケーションサーバ JBoss の機能と特徴 | |
| 対応する<br>コースウェア | 第 4 回（フリーの Web コンテナ/J2EE コンテナの概要） | |

# I-16-5. アプリケーションサーバ JBoss の機能と特徴

アプリケーションサーバについて、例として代表的なアプリケーションサーバである JBoss を取り上げ、その位置づけ、特徴を解説する。また Tomcat との連携や EJB コンテナ機能、アプリケーションサーバ機能など、JBoss の持つ様々な機能を紹介する。

【学習の要点】

＊ Java EE 規格に準拠したソフトウェアは Java アプリケーションサーバと呼ばれ、アプリケーションサーバの主流をなす。

＊ JBoss は、欧米では企業システムや政府機関システムでの導入実績もあり、他の商用アプリケーションサーバに劣らない機能と性能を実現している。

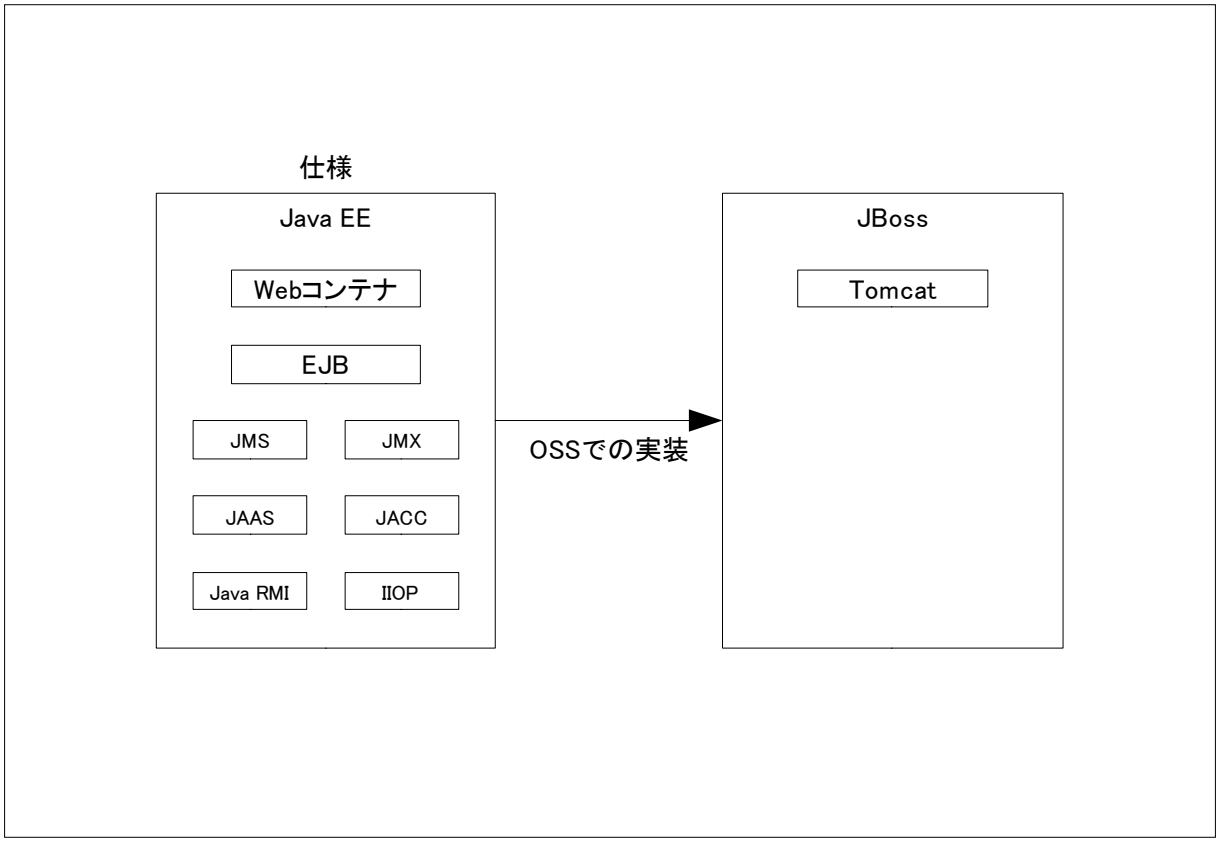


図 I-16-5.　JBoss の構成

【解説】

1) JBoss とは

JBoss (JBoss Application Server)は、OSS の Java アプリケーションサーバである。Java アプリケーションサーバとは、Java EE に準拠したソフトウェアを指す名称である。JBoss は LGPL ライセンスで提供される。

2) JBoss の位置づけと特徴

JBoss にはWebコンテナ(Tomcat)やデータベース管理システム(HSQLDB)が同梱されているので、JBoss を導入するだけで Web アプリケーションの動作環境が構築できるようになっている。また、プログラムをアップデートする際、特定のディレクトリにファイルを設置すれば、JBoss が自動でデプロイする。JBoss 本体を動作させたままプログラムのアップデートが可能なため、この機能は「ホットデプロイ」と呼ばれる。

3) Java EE (Java Platform, Enterprise Edition)

Java EE は、Java SE(Java Platform, Standard Edition)を主に企業の大規模システム向けに拡張した、Java の機能セットの仕様であり、次のような機能(抜粋)が定められている。

* Web コンテナ(Java Servlet、JSP のサポート)
* EJB(Enterprise JavaBeans)コンテナ
* JMS(Java Message Service)による非同期メッセージ通信
* JAAS(Java Authentication and Authorization Service)や JACC(Java Authorization Contract for Containers)による認証
* JMX(Java Management Extensions)によるシステム管理
* Java RMI(Remote Method Invocation)や IIOP(Internet Inter-ORB Protocol)による分散処理

4) Tomcat との連携

JBoss のパッケージには Tomcat が同梱されており、Tomcat と連携することで、Web コンテナの機能を実現している。

5) EJB コンテナ

JBoss は EJB コンテナ機能を有する。EJB とは、JavaBeans(Java アプリケーション開発における部品の作成や利用に関する規格)を企業向けに拡張した仕様である。EJB では、ネットワークシステムにおけるサーバプログラミングのための機能が追加されており、セッション処理のための Session Bean、データ保存のための Entity Bean、非同期処理などのための MDB(Message-Driven Bean)の 3 つに大別される。EJB は低レベルな処理の一切を引き受けているので、開発者はビジネスロジックだけに専念することができる。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-6. ツール実行を自動化する Ant | |
| 対応する<br>コースウェア | 第 5 回 (オープンソースの開発ツール) | |

# I-16-6. ツール実行を自動化する Ant

コンパイルやテスト、バージョン管理などソフトウェア開発に関わるあらゆる処理の実行を自動化するツール Ant を説明する。Ant の基本処理であるコンパイル、ビルドの XML による設定方法を示す。また、Ant で自動化できるテスト、ファイル転送、リモートホスト操作、バージョン管理などを紹介する。

【学習の要点】

* Ant は Java で開発されているビルドツールで、Java 版の make に相当する。
* make は OS(オペレーティングシステム)などの環境への依存性が高く、記述も比較的難解であるが、Ant はこのような make の欠点を補うものである。

```
<?xml version="1.0" encoding="UTF-8" ?>

<!-- project要素はビルドファイル内に1つ記述 -->
<!-- default属性は実行時にtargetを指定しない場合に実行するtargetを指定 -->
<project default="dist">

    <!-- target要素は一連の処理のまとまり -->
    <target name="compile">
        <!-- コンパイル ソースと出力先のディレクトリを指定-->
        <javac srcdir="src" destdir="class"/>
    </target>

    <!-- depends属性はこのtargetの前に実行するtargetを指定 -->
    <target name="dist" depends="compile">
        <!-- jarファイルの作成 ファイル名と対象ディレクトリを指定-->
        <jar jarfile="sample.jar" basedir="class"/>
    </target>
</project>
```

図 I-16-6.　ant が読み込む build.xml のサンプル

【解説】

1) Ant とは

Ant (Apache Ant)とは、Jakarta プロジェクトで開発されている、ビルド(ソースコード等から実行ファイルを作成すること)のためのツールであり、Apache ライセンスにて配布されている OSS である。統合開発環境である Eclipse には、Ant プラグインが標準装備されている。Ant は Java で開発されており、Java の動作する環境であれば Ant を利用できる。Ant は「ant」コマンドで実行され、引数を指定しない場合、カレントディレクトリにある build.xml というビルドファイル(ビルドのための設定を記述するファイル)に従って処理される。

2) Ant のビルドファイル

Ant のビルドファイルは XML で記述する。最上位の要素として、1 つの project 要素を記述する。project 要素の子要素として、1 つ以上の target 要素を記述する。一連の処理のまとまりは、target 要素単位で記述する。target 要素の子要素には、タスク(Ant で扱う処理の最小単位)を表す要素を記述する。タスクは、プラグインとして提供されているものを追加したり、Ant の API(アプリケーションプログラミングインタフェース)に従って自作したものを追加したりすることができる。主な Ant タスクは以下の通りである。

* コアタスク　　Ant コアパッケージに含まれているタスク
  - javac　　Java ソースコードをコンパイルする。
  - javadoc　　Java ソースコードからドキュメントを生成する。
  - java　　Java プログラムを実行する。
  - echo　　メッセージを出力する。
  - ant　　別のビルドファイルにあるタスクを読み込んで実行する。
* オプションタスク　　外部ライブラリを用いて機能させるタスク
  - junit　　テストフレームワーク JUnit を使って Java プログラムをテストする。
  - ftp　　FTP によるファイルのアップロード、ダウンロードを行う。
  - scp　　SCP、SFTP によるファイルのアップロード、ダウンロードを行う。
  - rexec　　リモートホストに対し rexec によるコマンド実行を行う。
  - sshexec　　リモートホストに対し SSH によるコマンド実行を行う。
  - cvs　　CVS によるバージョン管理を行う。

3) ビルドファイルの基本

ビルドファイルの例として、Ant の基本処理であるコンパイル、ビルドの記述例を図に示す。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-7. Java プロジェクト管理ツール Maven | |
| 対応する<br>コースウェア | 第 5 回（オープンソースの開発ツール） | |

# I-16-7. Java プロジェクト管理ツール Maven

Java ソフトウェア開発のコンパイル、ビルド、テスト、パッケージングに至る一連の作業の自動化するプロジェクト管理ツール Maven を解説する。スクリプトを記述する必要が少ないことなど Ant に比べて優れた点にも触れる。

【学習の要点】

* Maven は Java プロジェクト管理ツールとして、Ant の機能を包含するだけでなく、プロジェクトの作成から公開に至るまでのライフサイクル全体を管理できる。
* Maven は Ant のように XML を直接記述する必要が少なく、Ant に取って代わる可能性を秘めている。

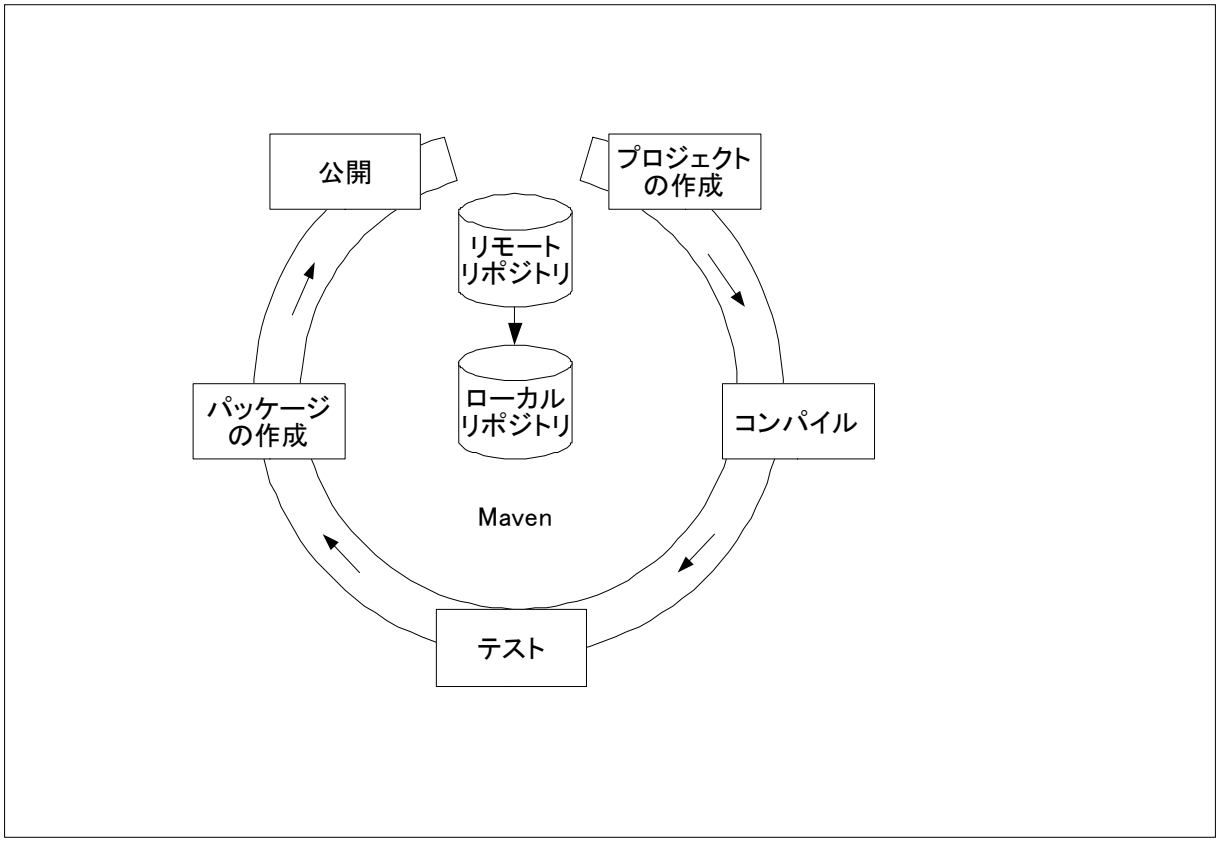


図 I-16-7.　Maven の概念

【解説】

1) Maven とは

Maven (Apache Maven)とは、Jakarta プロジェクトで開発されている、Java 用プロジェクト管理ツールであり、Apache ライセンスにて配布されている OSS である。Maven バージョン 2 はバージョン 1 から大きく変更されている。以下、バージョン 2 について解説する。

2) Maven の特徴

＊ ビルド支援

Maven は内部で Ant を使用しており、Ant のようにビルドスクリプトを一から書く必要がないようになっている。

＊ POM に基づいた動作

Java プロジェクト全体をオブジェクトとみなすモデル「プロジェクトオブジェクトモデル(POM)」に基づき、プロジェクトの作成から公開までのライフサイクル全体を管理できる。

＊ サイト公開が容易

POM に定義されている情報にアクセスして、HTML としてプロジェクトサイトを生成できる。生成される HTML には、POM に直接記述している情報のほか、以下のようなものが提供される。

- リポジトリ情報から直接生成された変更ログ
- 相互参照ソース
- ソースのメトリクス
- メーリングリスト
- 開発者リスト
- 依存関係リスト
- ユニットテストレポートとカバレッジレポート
- 記事のコレクション
- ソフトウェア開発のリファレンス
- ソフトウェア開発プロセスの文書

＊ リポジトリの共有

プロジェクトの使用者がビルドに必要な JAR ファイルをリモートリポジトリからダウンロードする機能を持つ。複数のプロジェクトに渡って JAR ファイルを再利用することができる。

＊ ディレクトリ構成の標準化

プロジェクトの推奨ディレクトリ構成があり、成果物の保守性、再利用性が高まる。

＊ 成果物の自動インストール

POM の依存関係で定義している成果物をリモートリポジトリより自動取得して、ローカルリポジトリにインストールする機能がある。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-8. Web アプリケーションのテスト支援ツール Cactus | |
| 対応する<br>コースウェア | 第 5 回 （オープンソースの開発ツール） | |

# I-16-8. Web アプリケーションのテスト支援ツール Cactus

Java ソフトウェアのテスト支援ツールである JUnit を紹介し、JUnit でテストできない Servlet や JSP に対応した Cactus を説明する。テストコードの記述方法、テストの実行方法を解説する。

【学習の要点】

* JUnit は Java プログラム用のテスト支援ツールとして広く利用されている。
* Cactus を利用することで、JUnit 単独では出来ない、Servlet やJSPのテストの自動化を行うことが出来る。

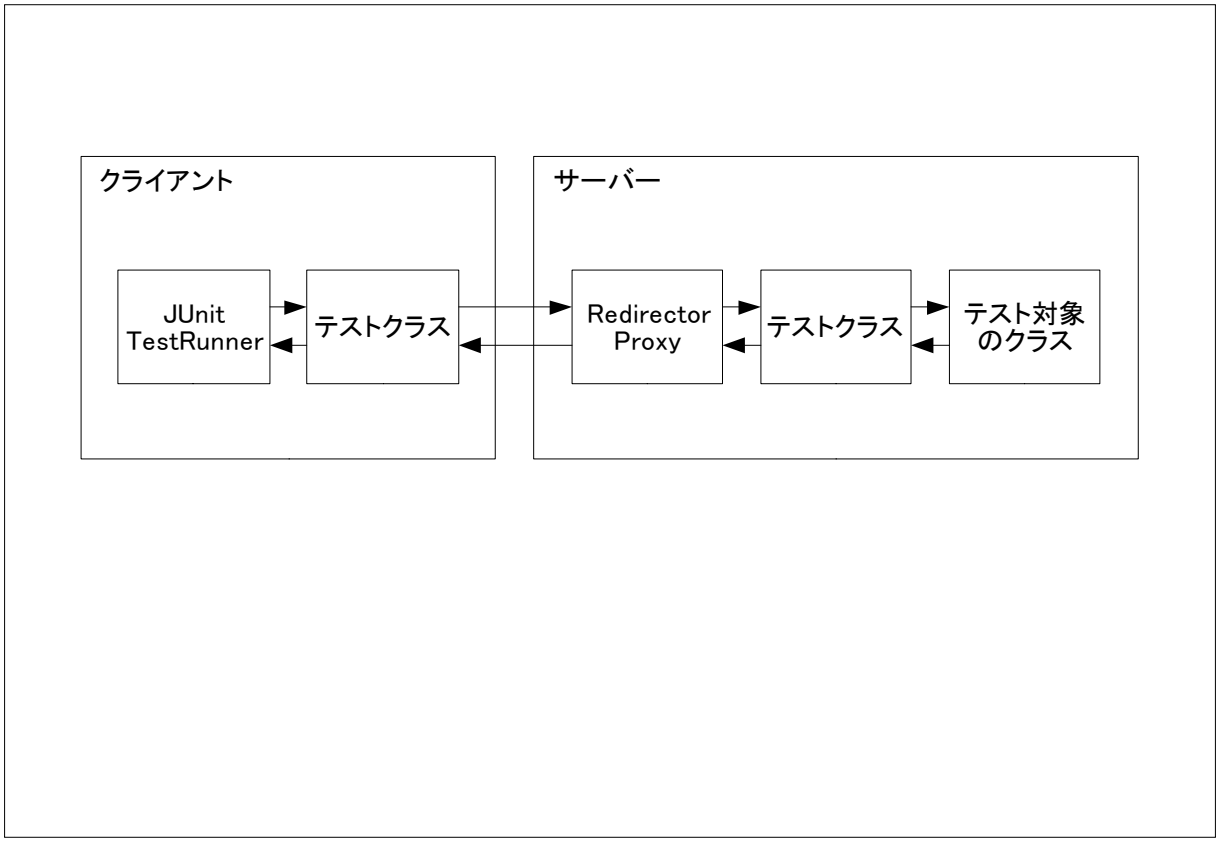


図 I-16-8.　Cactus によるテストの流れ

【解説】

1) JUnit とは

JUnit は Java プログラムをテストするためのツールである。JUnit 単独で動作するほか、Ant や統合開発環境 Eclipse のプラグインとしても利用可能である。

2) Cactus とは

Cactus とは、Jakarta プロジェクトで開発されている、サーバ側の Java プログラムをテストするためのフレームワークである。JUnit を拡張しており、JUnit 単独では出来ない、Servlet やJSPのテストが可能である。また、テストの自動化のために、Ant と連動して動作する機能を提供している。

3) Cactus の動作

Cactus では、クライアントとサーバの双方に同一のテストクラスを配置し、クライアントとサーバ間の HTTP 通信を、Redirector Proxy とよばれるオブジェクトを介して処理する。

* クライアント側では、JUnit で用意されている TestRunner クラスが、テストクラスの runTest()メソッドを呼び出す。
* runTest()が beginXXX()メソッド(XXX にはテスト毎につける名称が入る)を呼び出す。
* runTest()が Redirector Proxy に対し HTTP リクエストを発行する。
* サーバ側では、Redirector Proxy がテストクラスの testXXX()メソッドを呼び出す。
* testXXX()では、サーバ側のテスト対象となるクラスを呼び出し、JUnit の API を使用してテスト結果をチェックする。
* Redirector Proxy はクライアントに HTTP レスポンスを返す。テストが失敗した場合は、例外に関する情報をクライアント側に返す。
* 例外が発生しなければ、クライアント側で、runTest()が endXXX()メソッドを呼び出す。

4) テストコードの記述方法

* テストクラスの作成
  Cactus には以下の 3 つのテストケースクラスが用意されているので、テスト対象のコードに応じて、これらのクラスを継承してテストクラスを作成する。
  - ServletTestCase　　Servlet 用
  - JspTestCase　　JSP、カスタムタグ用
  - FilterTestCase　　フィルタ用
* テストメソッドの作成
  上で作成したクラスにおいて、テスト毎に、beginXXX()、testXXX()、endXXX()の 3 つのメソッドを記述する。beginXXX()では設定する HTTP リクエスト情報、endXXX()では取得する HTTP レスポンス情報が引数となる。

5) テストの実行方法

Cactus には Ant でテストを実行するためのビルドファイル build.xml が用意されており、「test」という名の target が記述されている。「ant test」コマンドを実行することで、テストを実行でき、結果が出力される。その他のテスト実行方法としては、統合開発環境 Eclipse に CactusRunner プラグインを導

入してテストする方法などがある。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-9. 要求分析モデル(DOA、OOA、Web MVC) | |
| 対応する<br>コースウェア | 第 6 回（開発フレームワークによる開発プロセスの手順） | |

## I-16-9. 要求分析モデル(DOA、OOA、Web MVC)

開発フレームワークによるアプリケーション開発を行う際に実施する要求分析について、DOA (データ中心アプローチ)、OOA (オブジェクト指向分析)、Web MVC (モデルービューーコントローラ)モデルといった各種のモデルの特徴と違いについて解説する。

【学習の要点】

* DOA によりデータを処理から独立させることで、プログラムに変更があった場合にデータへの影響を抑えることができる。
* OOA により各部品を分業させることで、プログラムに変更があった場合に影響する部品を最小限にすることができる。
* MVC により役割を分離させることで、プログラムに変更があった場合に他の役割への影響を抑えることができる。

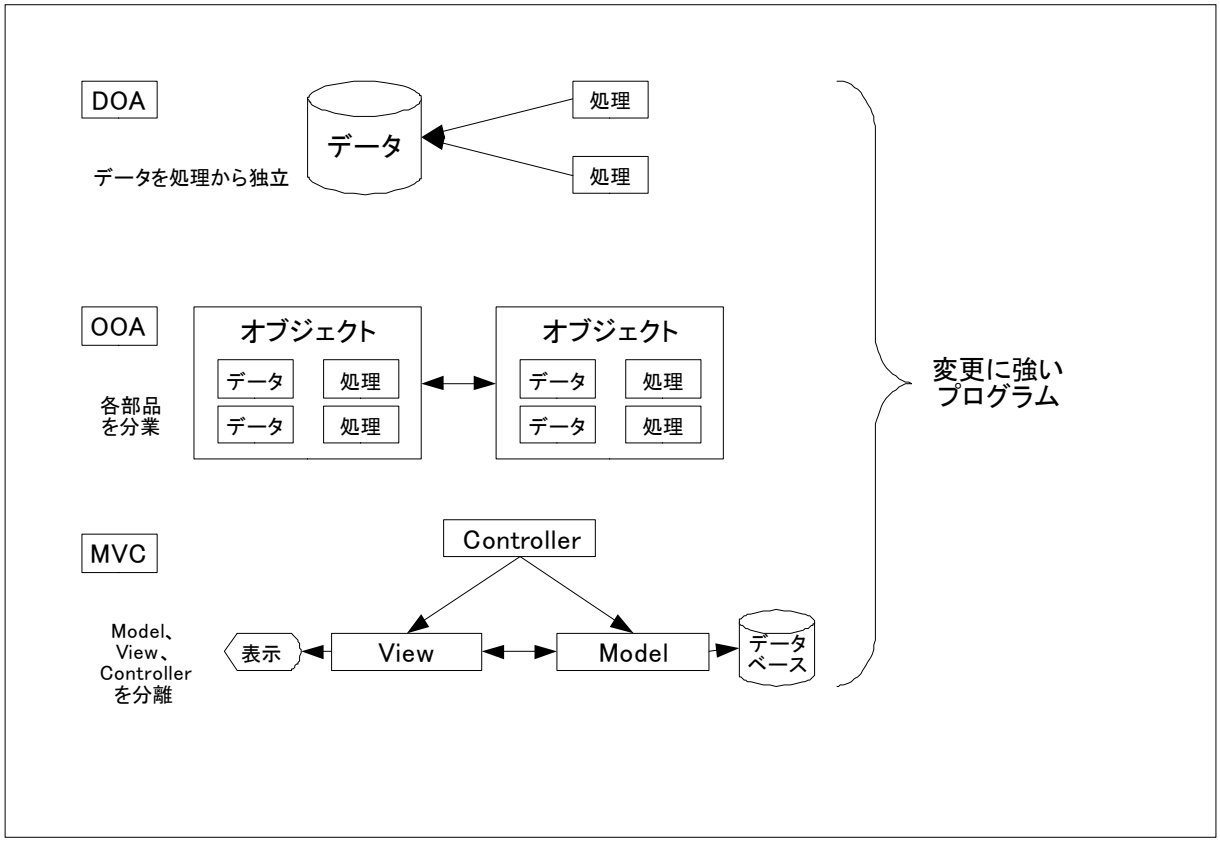


図 I-16-9.　要求分析モデルとプログラム変更への対応

【解説】

1) DOA (データ中心アプローチ)

従来は、まず処理について設計し、データは処理に付随するものとして後から考えられる、という手法が一般的であったが、この場合、同じデータが重複して多重管理になったり、処理に変更が加わる際にデータも影響を受けたりするという問題があった。DOA は業務システムにおいて関係データベースの普及とともに広まっていった手法で、「データは組織の最重要資産である」「データは処理に比べて仕様変更の頻度が低い」という考え方がもとになっている。DOA では、変化の多い処理からデータを独立させ、業務処理の流れをデータの流れを中心に捉え、個々の処理はデータに付随するものと考える。データモデリングを ERD(Entity-Relationship Diagram)で記述するという方法が一般的である。

2) OOA (オブジェクト指向分析)

OOA は、オブジェクト指向によるシステム開発の上流工程での分析手法である。オブジェクト指向では、各オブジェクトが分業しながら機能を果たすので、仕様変更の際の影響範囲が小さくて済む。DOA では、データの独立性が高くなる反面、データと関連する処理の特定が困難になるという問題点が生じる。これに対し、オブジェクト指向は処理とデータを一体化させている。OOA は組み込みシステムにおいて Java の普及とともに広まっていった手法で、人間の発想や活動に近い考え方でオブジェクトのモデリングを行い、UML(統一モデリング言語)で記述すると言う方法が一般的である。

3) Web MVC (モデルービューーコントローラ)

MVC モデルは、オブジェクト指向プログラミング言語 Smalltalk の GUI 設計に用いられた概念であり、Web アプリケーションの開発フレームワークに多く採用されるようになった。MVC では、モデル(データを保持し業務ロジックを実行する部分)、ビュー(プレゼンテーションを行う部分)、コントローラ(モデルとビューとを制御する部分)の 3 つの役割に分けてアプリケーションを開発する。役割によって独立性を高めることで、仕様変更の際の影響範囲が小さくて済む。要求分析の段階から、保持するデータと画面とを分けて考えることができる。

4) 要求分析モデルの併用

Web アプリケーションの実際としては、オブジェクト指向言語で Web MVC フレームワークを用い、データベースは関係データベースというように、上の 3 つのモデルを組み合わせるケースが多い。いずれのモデルも仕様変更に強いなどの共通の目標を持つが、アプローチの違いにより、インピーダンスミスマッチ等の問題が生じるので、要求分析レベルから各アプローチのすり合わせを行い、O/R マッピングを利用するなどして、問題の解消を行う。

| スキル区分 | OSS モデルカリキュラムの科目 | レベル |
|---|---|---|
| 開発体系分野 | 16　開発フレームワークに関する知識 I | 基本 |
| 習得ポイント | I-16-10. Web アプリケーション開発における実際の作業プロセス | |
| 対応する<br>コースウェア | 第 6 回（開発フレームワークによる開発プロセスの手順） | |

# I-16-10. Web アプリケーション開発における実際の作業プロセス

Web アプリケーション開発に関して、プレゼンテーション設計、画面遷移設計、ビジネスロジック設計、データベース設計、コンポーネント間インタフェース設計といった実際の実装作業プロセスを説明する。

【学習の要点】

＊ Web アプリケーション開発に必要な各設計項目のプロセスを把握しておくことで、漏れのない設計ができ、開発作業の効率化やソフトウェアの品質向上が期待できる。

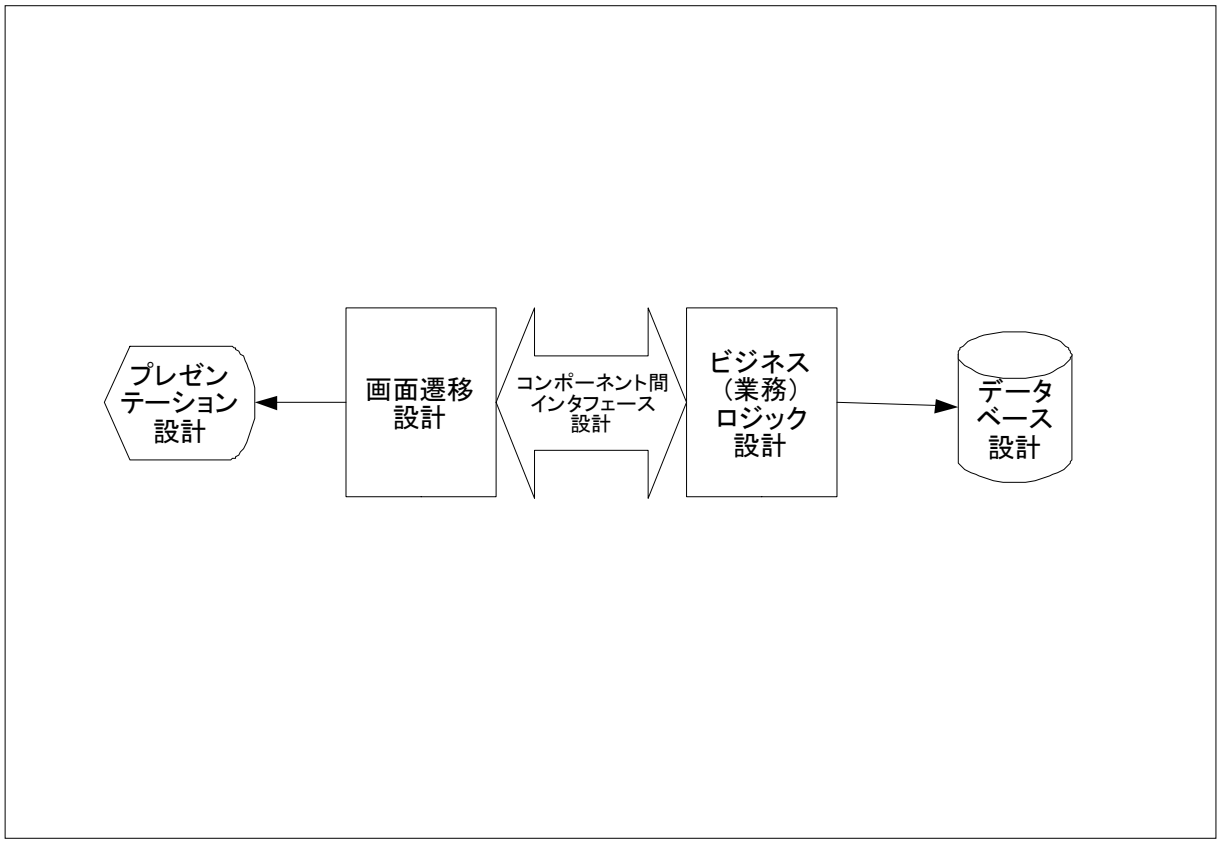


図 I-16-10.　様々な作業プロセス

【解説】

1) Web アプリケーションの設計項目

ここでは、Java などのオブジェクト指向言語と、関係データベース、O/R マッピングフレームワークを利用した場合を例として、一般的な Web アプリケーションを開発する場合の設計プロセスについて示す。設計項目には、以下のようなものが必要となる。

* データベース設計
* プレゼンテーション設計　　（個々の Web ページの表示に関する設計）
* 画面遷移設計
* ビジネス(業務)ロジック設計
* コンポーネント間インタフェース設計

2) データベース設計のプロセス

ERD に基づき、正規化を行う。また、運用上などの要件により、必要に応じて逆正規化を行う。ERD を物理設計レベルまでブレイクダウンして作成し、データベース構築の SQL が生成可能な状態にする。

3) プレゼンテーション設計のプロセス

各画面ごとに、表示項目、入力項目を洗い出し、各項目の表示内容、表示場所、HTML 要素やスタイルを決定する。入力項目については、入力可能な値の範囲を明確にする。画面設計書などを作成する。

4) 画面遷移設計のプロセス

各画面から遷移可能な画面を洗い出し、遷移するための条件を明確にする。画面遷移図などを作成する。

5) ビジネスロジック設計のプロセス

UML のクラス図を、クラスの属性やメソッド、クラス間の関係などの記述を追加した形でブレイクダウンして作成する。加えて、UML のアクティビティ図、状態図、シーケンス図などを作成する。

6) コンポーネント間インタフェース設計のプロセス

コンポーネント(プログラムの部品)同士のやり取りの規約を作成する。Web MVC フレームワークにおけるモデルとビューとのやりとりは、コンポーネント間インタフェースを必要とする典型例である。UML のコンポーネント図などを作成する。