16. 開発フレームワークに関する知識 II

1. 科目の概要

実際の開発フレームワークについてその構成と特徴を解説し、それらの開発フレームワークを利用する方法と、OR マッピングや DIxAOP コンテナなど、開発フレームワークで利用される基本的な概念について説明する。

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
II-16-1. Light Weight Languageによるフレームワーク	Light Weight Languageによる開発フレームワーク例として、RubyのRuby on Rails (RoR)やPHPでWebアプリケーションフレームワークとして用意されているEthna、PerlのWebフレームワークCatalystなどがある。これらの基本的な構成と特徴、Light Weight Languageによる開発フレームワークを利用したアプリケーション開発について説明する。	7
II-16-2. Strutsを利用した開発	JavaによるWebアプリケーションフレームワークであるStrutsの基本的な構成と特徴、Strutsを利用したアプリケーション開発について説明する。ユーザからのリクエストを処理部に渡すコントローラサーブレット、JavaBeansプロパティの設定機能、国際化支援などStrutsの様々な機能を解説する。	8
II-16-3. MyFacesを利用した開発	Webアプリケーションのユーザインタフェースを作成するためのフレームワークであるJava Server Faces (JSF)を紹介し、オープンソースのJSF実装であるMyFacesの基本的な構成と特徴、MyFacesを利用したアプリケーション開発の具体例について説明する。	9
II-16-10. Tapestryを利用した開発	Webアプリケーション開発において、Webデザイナの作業とプログラマの作業を分離し、並行して開発を進めることを実現するためのフレームワークであるTapestryの基本的な仕組みと基本的な構成、特徴について解説する。	14
II-16-4. ORマッピングの仕組み	オブジェクト指向アプリケーションのデータとして取り扱うオブジェクトと、リレーショナルデータベースに格納されるレコードとの対応を取るための仕組みであるORマッピングについて、基本的な考え方と処理の流れ、使い方などについて解説する。	10
II-16-5. ORマッピングツール(Hibernate)	SQLを意識せずに利用することができるORマッピングツールであるHibernateの概要について述べる。Hibernateの基本的な構成と特徴、マッピングの仕様、Hibernateを利用したデータベースの操作方法などについて説明する。	10
II-16-6.ORマッピングツール(iBATIS)	SQL文をマッピングファイルに記述して利用する形式のORマッピングツールであるiBATISの概要について述べる。iBATISの基本的な構成と特徴、マッピングの仕様、iBATISを利用したデータベースの操作方法などについて説明する。	10
II-16-7. DIxAOPコンテナの仕組み	Dependency Injection (DI)とAspect Oriented Programming (AOP)の考え方を整理し、新しいWebアプリケーションの基盤を成すDIxAOPコンテナの仕組みと基本的な構成、特徴について解説する。また個々の実装例の概要を紹介する。	11
II-16-8. DIxAOPコンテナ(Springフレームワーク)	DIxAOPコンテナ実装のひとつであるSpring Framework (Springフレームワーク)の基本的な仕組みと基本的な構成、特徴について解説する。またStrutsやJSF、iBATIS、Hibernateなど他のフレームワークとの連携や開発ツールなど、実際の利用に有効なトピックを紹介する。	12
II-16-9. DIxAOPコンテナ(Seaser2)	DIxAOPコンテナ実装のひとつであるSeasar2の基本的な仕組みと基本的な構成、特徴について解説する。またStrutsやJSF、Hibernateなど他のフレームワークとの連携や開発ツールなど、実際の利用に有効なトピックを紹介する。	13

【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Ⅱ」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スキル	<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 [16-6]	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. データ割り当てと変換	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 [16-6]	CE-SWE3. ソフトウェアの設計 [16-6]	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. ソフトウェアの標準化	
	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	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 Ruby 言語、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、OR マッピング・DI・AOP など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7回	第8回	第9回	第10回	第11回	第12回	第13回	第14回	第15回
	(1)Ruby on Rails の概要	(1)Struts の概要	(1)MyFace の概要	(1)OR	(1)DIxAOP コンテナの機能	(1)Spring の構成と特徴、メリット	(1)Seasar2 の構成と特徴、メリット	(1)Tapestry の構成と特徴、メリット	(1)Web アプリケーションの作成手順
	(2)Ruby on Rails 開発の内容	(2)Struts を利用するメリットとデメリット	(2)MyFace を利用するメリットとデメリット	(2)Hibernate の概要	(2)実装フレームワーク	(2)DIxAOP 以外の主な機能	(2)DIxAOP 以外の主な機能	(2)主な機能と開発スタイル	(2)Struts で提供されるコンポーネント
16.開発フレームワークに関する知識Ⅱ	(3)開発環境	(3)Struts の動作環境	(3)MyFace の動作環境	(3)iBATIS の概要		(3)開発ツール	(3)開発ツール		(3)Struts で提供されているタグ
		(4)Struts の発展：Shale				(4)ライセンス	(4)ライセンス		(4)Struts の便利な機能を利用 (5)他のプロダクトとの連携 (6)実用アプリケーションの作成

（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	応用
習得ポイント	II -16-1. Ruby on Rails(RoR)の仕組み	
対応するコースウェア	第 11 回 (Ruby on Rails)	

II -16-1. Light Weight Language によるフレームワーク

Light Weight Language による開発フレームワーク例として、Ruby の Ruby on Rails (RoR)や PHP の Ethna、Perl の Catalyst などがある。これらの基本的な構成と特徴、Light Weight Language による開発フレームワークを利用したアプリケーション開発について説明する。

【学習の要点】

* Ruby on Rails(RoR)は、Ruby 言語を利用する Web アプリケーション用のフレームワークである。MVC を採用しており、ソースコードや設定ファイルの記述量を削減することで、動作するアプリケーションを素早く開発することに主眼を置いたフレームワークである。
* Ethna は、PHP 言語を利用する Web アプリケーション用のフレームワークである。Struts の構造を参考にした MVC アーキテクチャを採用し、少ない記述量でアプリケーションの作成が可能である。入力値のフィルタリングや検証の仕組みが充実しているという特徴がある。
* Catalyst は RoR 等に影響を受けて開発された、Perl 言語を利用する Web アプリケーションフレームワークである。RoR と同様 MVC を採用しており、Model や View には既存の Perl モジュールを用いている。

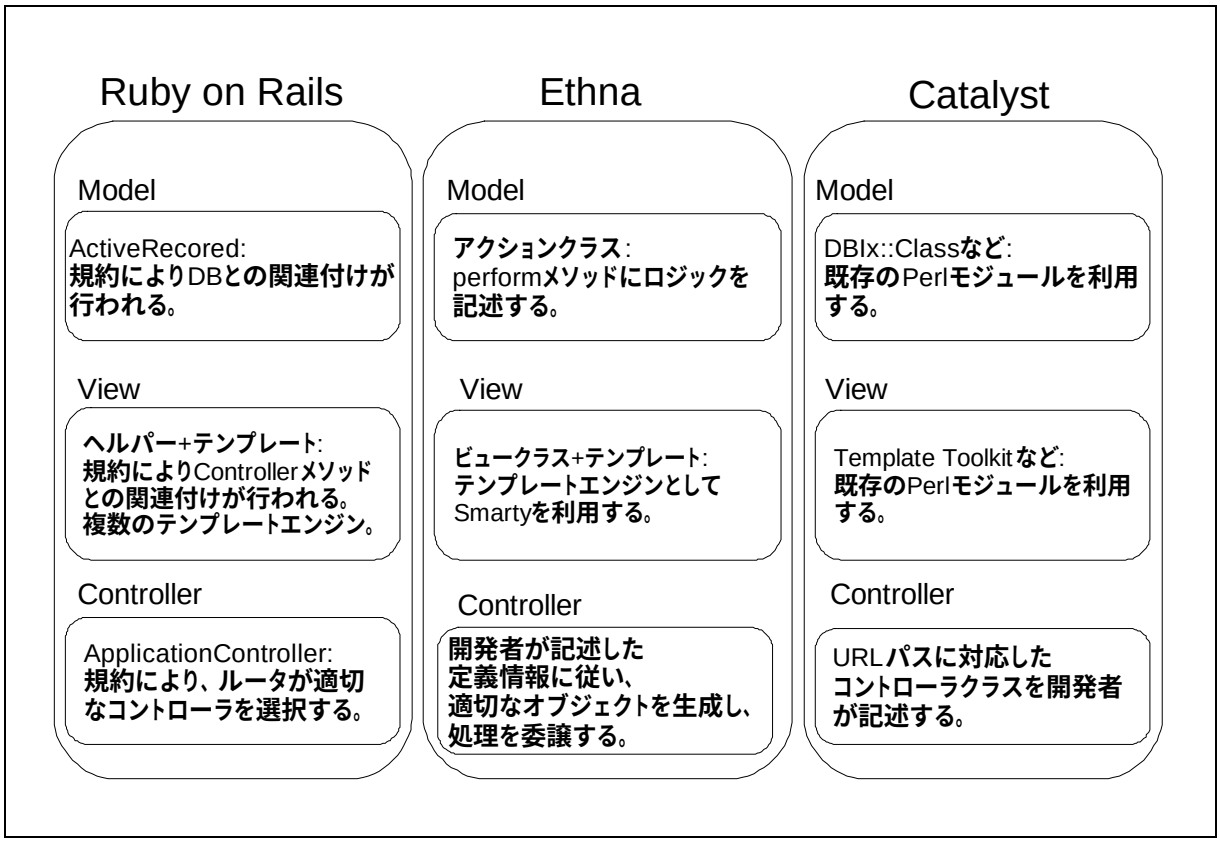

図 II-16-1.　RoR、Ethna、Catalyst の MVC

【解説】

1) Light Weight Language によるフレームワーク

Light Weight Language は、コンパイルの必要がなく動作確認が容易であることや、テキスト処理を得意とする言語が多いことから、Web の世界で広く利用されてきた。RoR はこれらの機能に加えて、Light Weight Lanugage が動的に拡張可能である点を利用し、設定ファイルやソースコードを大幅に減らすことに成功した。RoR の成功をきっかけとして、Light Weight Language によるフレームワークへの注目が高まっている。

2) Ruby on Rails(RoR)

RoR についての詳細は「II-15-1. Ruby on Rails (RoR)の仕組み」を参照のこと。Ruby on Rails(RoR)で全面的に採用された「設定より規約:CoC（Convention over Configration）」、および「DRY(Don't Repeat Yourself)」の方針は、以降のフレームワークに大きな影響を与えた。また、既存のフレームワークにおいても、ソースコードや設定ファイルの削減を目指した改良を行うものが増えている。

3) Ethna

Ethna は PHP 言語を利用した国産の Web アプリケーションフレームワークであり、「似たようなコードを書かなくてよい」ことを目標として開発が進められている。

* Struts の構造を参考にした MVC アーキテクチャを提供する。
* RoR のような、プロジェクト、アクション、ビューの雛形の生成機能を持つ。
* フォーム入力を伴う Web アプリケーション開発で必要となる、フォーム値の検証機能や、フィルタリング(適切な形式への変換)のための機能が充実している。
* フィルタチェインの仕組みを利用することで、既存のアプリケーションコードを変更せずに、リクエストに対する前処理、およびレスポンスに対する後処理を追加することができる。
* Ethna_AppObject により、O/R マッピング機能を利用することができる。
* Ethna は PEAR (The PHP Extension and Application Repository)を利用してインストールすることができる。

4) Catalyst

Catalyst は、Perl 言語を利用した Web アプリケーションフレームワークであり、RoR や Spring に影響を受けている。

* MVC アーキテクチャを採用しており、Model、View に関しては既存の Perl モジュールの中から要件に合うものを選択して利用することが想定されている。
* DRY に加え、Perl の特徴である「There Is More Than One Way To Do It」も同様に重視されており、フレームワークで提供される機能を他のモジュールで置き換えることが容易である。
* Perl は CGI を利用した動的ページの作成言語として Web 分野で利用されてきたことから、Web アプリケーション開発に必要な機能のほとんどを既存のモジュールを利用することで実現することができる。Perl の公開モジュールは CPAN(Comprehensive Perl Archive Network)を通してインストールすることが可能であり、Catalyst はこれらの資産を容易に利用可能であるという強みがある。
* Catalyst そのものも、CPAN を利用してインストールすることができる。

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	応用
習得ポイント	II -16-2. Struts を利用した開発	
対応する コースウェア	第 8 回 (Struts とは)	

II -16-2. Struts を利用した開発

Java による Web アプリケーションフレームワークである Struts の基本的な構成と特徴、Struts を利用したアプリケーション開発について説明する。ユーザからのリクエストを処理部に渡すコントローラサーブレット、JavaBeans プロパティの設定機能、国際化支援など Struts の様々な機能を解説する。

【学習の要点】

* Struts は Java 言語を利用する Web アプリケーション用のフレームワークであり、MVC を実現する Web アプリケーションフレームワークの中で最も広く利用されているものの一つである。
* Struts では Model は Action クラスの継承クラス、View は JSP、Controller は ActionServlet によって実現される。Struts では制御情報を XML の設定ファイルに記述するため、個々のビジネスロジックを行うソースコードと、それらの関連についての情報が分離される。
* ActionForm クラスを継承したクラスを用意し、設定ファイルに登録することで、JSP のフォームの内容を Action クラスへ渡すことができる。
* メッセージリソースを利用することで、アプリケーションの国際化を実現できる。

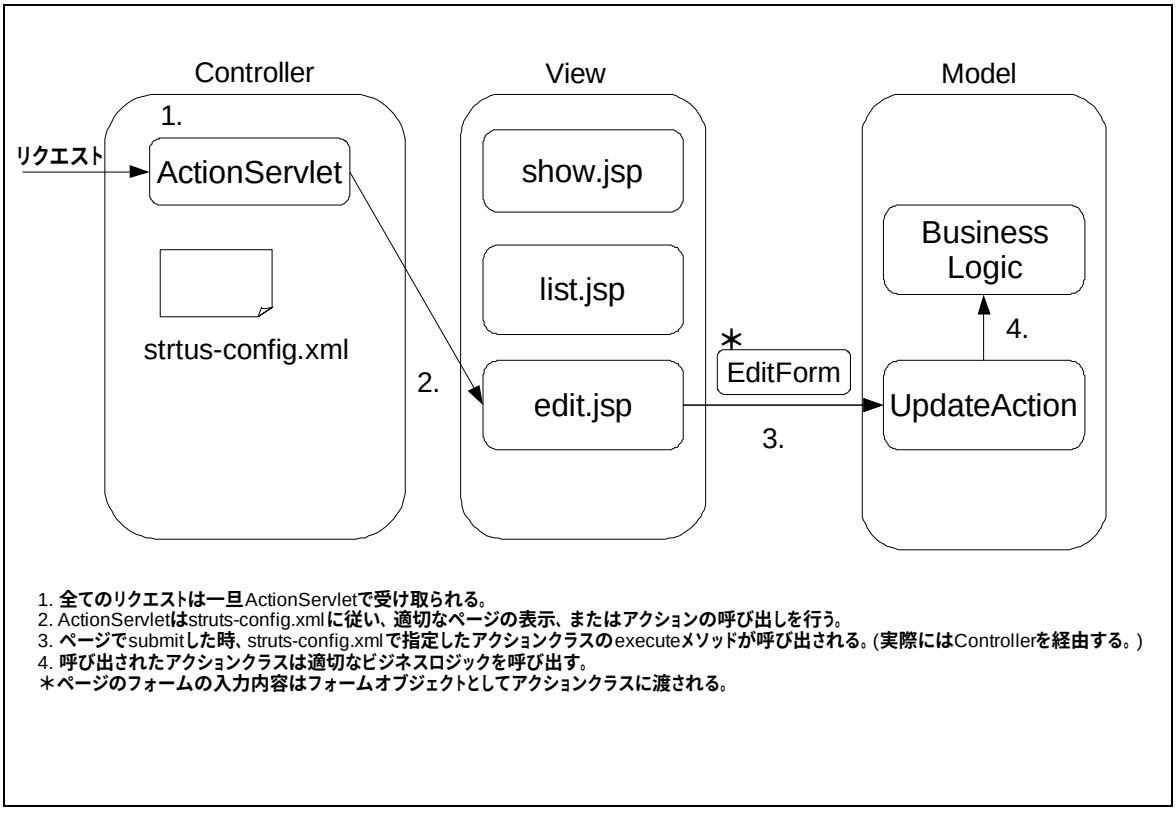

図 II-16-2.　Struts における MVC

【解説】

1) Struts

Struts は Java EE アプリケーションの開発を支援する Web アプリケーションフレームワークであり、Servlet API を利用した MVC アーキテクチャを提供する。ソフトウェアの 3 層アーキテクチャのうち、主にプレゼンテーション層をカバーする。全てのリクエストは一旦 ActionServlet によって受け付けられた後、適切なオブジェクトに委譲される。

2) Struts の MVC

* Model
  - ビジネスロジックは、org.apache.struts.action.Action クラスを継承したクラス(アクションクラス)の execute メソッドを開始点として処理される。返値で処理後の遷移情報を指定する。
  - View からの入力値は、org.apache.struts.action.ActionForm クラスを継承したクラス(フォームオブジェクト)を経由してアクションクラスに渡される。
  - アクションクラス内で、ビジネスロジック処理、およびデータベースアクセス処理を行うことも可能であるが、EJB コンテナや、O/R マッピングライブラリなど、他のフレームワークを組み合わせるなどして、多層アーキテクチャとする構成も広く用いられている。
* View
  - Struts では、View を実現するための技術として JSP を利用する。
  - Strtus から JSP 用のタグライブラリが提供されている。フォームを特定のアクションクラスと結びつける場合や、入力フィールドをフォームオブジェクトの属性に関連付ける場合などは、このタグライブラリを用いて記述する。
* Controller
  - Struts の Controller は、ActionServlet として提供され、開発者は設定ファイル struts-config.xml に制御情報を記述する。
  - struts-config.xml に記述する情報の例としては、フォームオブジェクトや、アクションクラスに関する情報、アクションの返値に応じた画面遷移に関する情報などがあげられる。

3) Struts アプリケーションの国際化

Struts ではアプリケーションを多言語に対応させるために、メッセージリソースと呼ばれる機能を利用することができる。

* メッセージリソース機能は、リソースファイルを用意することで使用できる。
* リソースファイルは、「キー=メッセージ」形式の properties ファイルとして記述する。

例) message.complete=完了しました

* リソースファイルは言語毎に用意し、ファイル名により、どの国、および言語に対応するファイルであるかを区別する。
* リソースファイルは、struts-config.xml ファイルに、message-resources タグを用いて登録する。
* メッセージリソース機能を利用するアプリケーションは、Java コード、および JSP 内で、メッセージを直接記述する代わりに、リソースファイルで指定したキーを用いる。

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	応用
習得ポイント	II -16-3. MyFaces を利用した開発	
対応するコースウェア	第 9 回 (MyFace(JSF)の開発)	

II -16-3. MyFaces を利用した開発

Web アプリケーションのユーザインタフェースを作成するためのフレームワークである Java Server Faces (JSF)を紹介し、オープンソースの JSF 実装である MyFaces の基本的な構成と特徴について説明する。

【学習の要点】

* JavaServer Faces は Web アプリケーションの MVC のうち、主に View と Controller の機能を提供するフレームワークであり、Java Community Process により標準化されている。
* MyFaces プロジェクトでは、JSF 準拠の実装を提供する MyFaces Core の他にも様々なサブプロジェクトが存在し、拡張機能が提供されている。
* JavaServer Faces では View の記述方法として、タグライブラリを用いた JSP によって記述する方法や、Facelets を利用して記述する方法がある。
* JavaServer Faces では View からの入力情報の処理を POJO (Plain Old Java Object) に記述し、Managed Bean として登録して利用する。

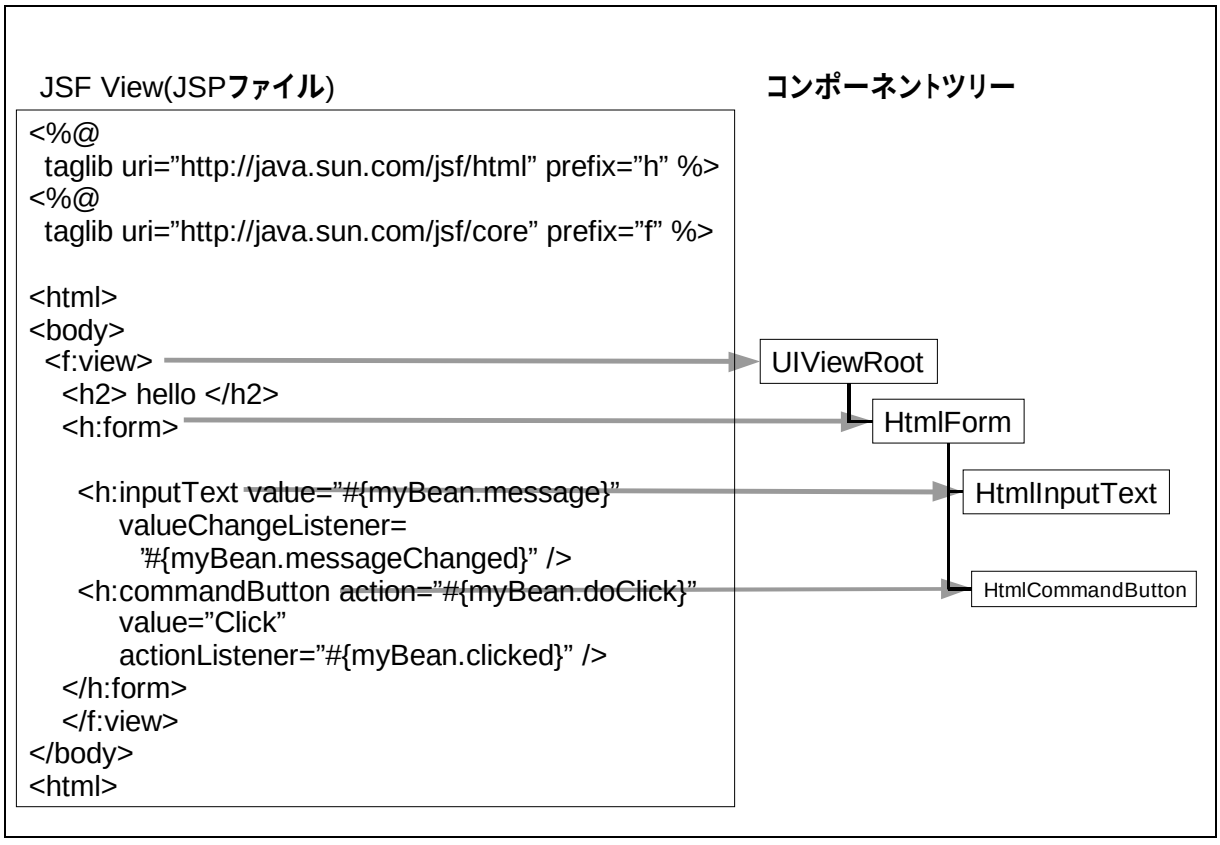

図 II-16-3.　JSF におけるコンポーネントツリー

【解説】

1) JSF(JavaServer Faces)

JSF は Java EE アプリケーションの開発を支援する Web アプリケーションフレームワーク、および仕様である。

* JSF の仕様は、Java Community Process により、Java EE の標準仕様として策定されている。
* JSF はコンポーネント指向のフレームワークであり、ページを構成する部品をコンポーネント単位で管理している。また、JSF は標準仕様であるため、サードパーティによるコンポーネントの開発も行われており、開発にあたって利用することができる。
* JSF は、View に対応したコンポーネントのツリー構造を Java のオブジェクト構造として構築し、保存する。このことにより、コンポーネント毎の値の変化の有無といった情報を知ることが可能になり、デスクトップアプリケーションのようなイベントベースのプログラミングが可能になる。

2) JSF の MVC

* Model
  - JSF では、フォームデータの格納とビジネスロジックの実行を行う Java オブジェクトとして、POJO (Plain Old Java Object) を用いることができる。この POJO を Managed Bean と呼び、設定ファイル faces-config.xml で managed-bean ディレクティブを用いて登録する。
  - Struts と同様に、多層アーキテクチャとする場合も多い。
* View
  - JSF の View は JSP で記述することが一般的であるが、JSP 以外の技術を利用することも可能である。例えば、JSF 用の View 技術として新たに開発された Facelets や、Mozilla プロジェクトで開発された View 技術である XUL などを JSF の View を実現するための手段として利用することができる。
  - JSP で View を記述する場合、JSF から提供されるカスタムタグライブラリを利用する。カスタムタグで記述されたツリー構造に基づいて内部的なコンポーネントツリーが構築される。
* Controller
  - JSF の Controller として FacesServlet が提供されている。
  - 制御情報は設定ファイル faces-config.xml に記述する。faces-config.xml に記述する情報の例としては、Managed Bean の登録情報や、画面遷移に関する情報があげられる。

3) MyFaces

MyFaces は、JSF の実装の一つであり、Apache Software Foundation のプロジェクトとして開発が行われている。JSF 仕様に準拠した実装を提供する MyFaces Core の他に様々なサブプロジェクトが存在する。以下に例をあげる。

* Portlet Bridge - JSR301 として策定された Portlet Bridge 仕様の実装。
* Tomahawk - MyFaces プロジェクトの開発者により開発されたコンポーネント集。
* Trinidad - Oracle 社から寄贈されたコードをベースに開発されているコンポーネント集。
* Tobago - Atanion GmbH 社から寄贈されたコードをベースに開発されているコンポーネント集。
* Orchestra - 様々なスコープにおける永続セッションを管理するフレームワーク。

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	応用
習得ポイント	II -16-4. OR マッピングの仕組み	
対応する コースウェア	第 10 回 (データベース接続･アクセスのフレームワーク)	

II -16-4. OR マッピングの仕組み

オブジェクト指向アプリケーションのデータとして取り扱うオブジェクトと、リレーショナルデータベースに格納されるレコードとの対応を取るための仕組みである OR マッピングについて、基本的な考え方やオブジェクトとレコードの対応関係、OSS による OR マッピングの実装例について解説する。

【学習の要点】

* オブジェクト指向言語を用いてリレーショナルデータベースを利用するアプリケーションを開発した場合、オブジェクトとデータベースのデータ構造の違いが問題となる場合がある。
* OR マッピング(Object Relational Mapping)はオブジェクトとリレーショナルデータベースの相互変換を行う仕組みである。
* OR マッピングではクラスがテーブル、インスタンスが行、インスタンスの属性が列に対応する。

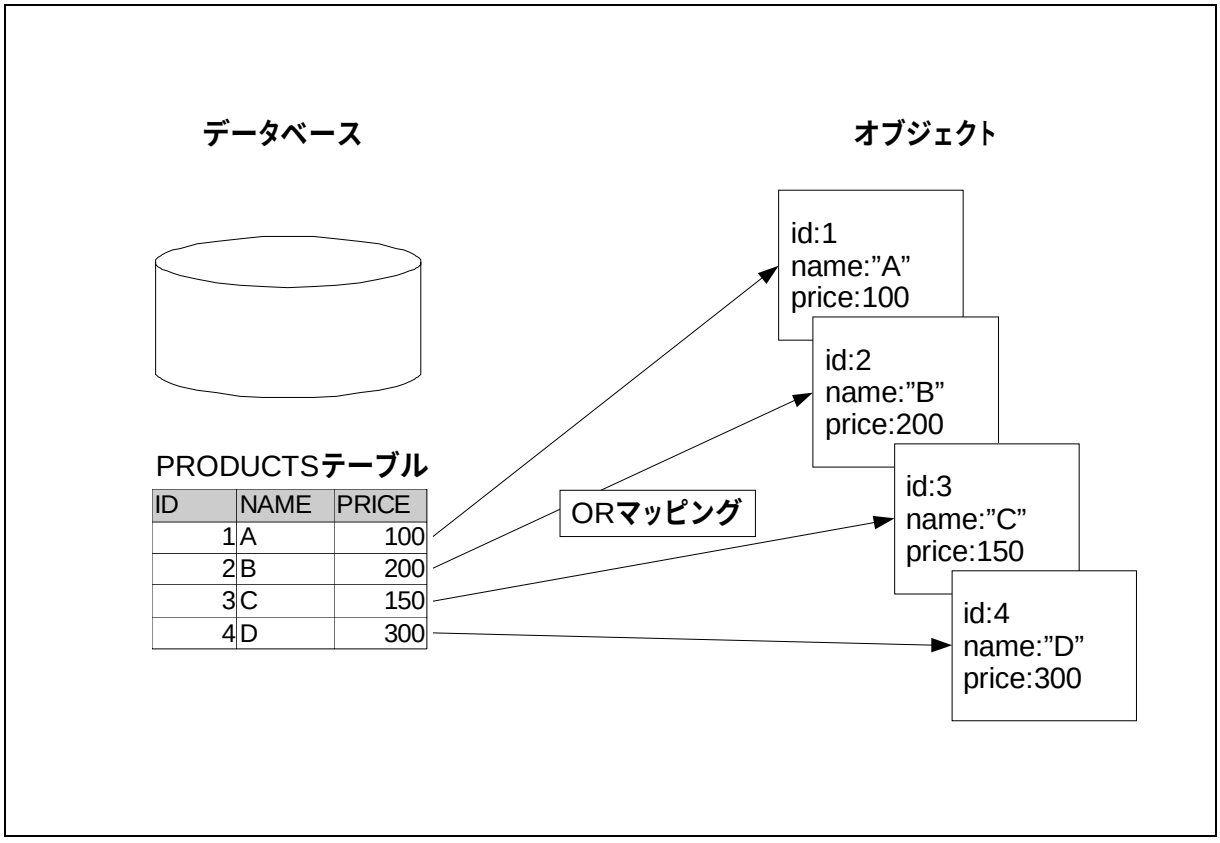

図 II-16-4.　OR マッピング

【解説】

1) オブジェクト指向とリレーショナルデータベース

オブジェクト指向言語を用いてリレーショナルデータベースを利用するアプリケーションを開発した場合、データ構造がアプリケーション内とデータベース内に二重に存在することになる。これらのデータ構造の間に互換性がない（インピーダンスミスマッチ）ため、データ構造間の変換処理が必要となる。また、変換処理のほかにも、アプリケーション内のデータとデータベース内のデータの同期をとる処理、データの一貫性を確保するための仕組みなど、煩雑なプログラムが必要となる場合が多く、開発者の負担となりやすい。

2) OR マッピング

OR マッピングとは、オブジェクト指向言語におけるデータ構造と、リレーショナルデータベースのデータ構造の違いを吸収し、相互変換を可能とする仕組みである。OSS を含め様々な実装が存在し、無償で利用可能なものも多い。これらを利用することで開発者の負担を削減することができる。

3) OR マッピングの基本

OR マッピングの利用方法として、多くのライブラリ/フレームワークに共通する要素を以下にあげる。

* オブジェクト指向のクラスは、データベースのテーブルに対応する。
* インスタンスは、データベースの行に対応する。
* インスタンス属性は、データベースの列に対応する。
* データの検索結果はインスタンスのセットとして返却される。
* データの更新は、検索によって得られたインスタンスの属性値を変更することによって行う。

4) OR マッピングの実装

OR マッピングを実現するライブラリ/フレームワークの例を以下にあげる。

* JPA(Java Persistent API)
  EJB3 の仕様の一部であり、EJB3 に準拠したコンテナで利用することができる。J2SE 5.0 仕様で導入されたアノテーション（メタデータとして注釈を記入すること）を利用してクラスとテーブルを関連付ける。
* Hibernate
  Java で利用可能な OR マッピングフレームワークのうち、最も広く利用されているものの一つ。Hibernate 本来の API のほかに JPA に準拠したインタフェースも提供する。
* iBatis
  検索に利用する SQL を設定ファイル内に明示する、という特徴を持つ OR マッピングフレームワークであり Java から利用できる。SQL の知識を生かした性能チューニングが行いやすい。
* ActiveRecord
  Ruby on Rails に含まれている OR マッピングライブラリ。クラスとテーブルの関連付けを規約によって行うため、設定ファイルやソースコードの記述量を大幅に削減できる。

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	応用
習得ポイント	II -16-5. Hibernate の構成と特徴	
対応する コースウェア	第 10 回 (データベース接続･アクセスのフレームワーク)	

II -16-5. Hibernate の構成と特徴

SQL を意識せずに利用することができる OR マッピングツールである Hibernate の概要について述べる。Hibernate の基本的な構成と特徴、マッピングの仕様、Hibernate を利用したデータベースの操作方法などについて説明する。

【学習の要点】

* Hibernate は最も広く使われている OR マッピングライブラリの一つであり、Java EE 標準の OR マッピング API である JPA(Java Persistence API)をサポートしている。
* Hibernate ではマッピングファイルによって POJO とテーブルが関連付けられる。
* Hibernate では検索方法として、従来の SQL のほかに、SQL に似た HQL(Hibernate Query Language)を使って検索条件を指定する方法と、Criteria オブジェクトを利用し、Java オブジェクトの組み合わせによって検索条件を指定する方法が用意されている。

```
import org.hibernate.*;
import org.hibernate.cfg.*;

public class HibernateTest {

   public static void main(String[] args){
       Session session = getSessionFactory().getCurrentSession();

       session.beginTransaction();

       Product product = new Product();
       product.setName("product name");
       product.setPrice(100);

       session.save(product);

       session.getTransaction().commit();
   }

    public static SessionFactory getSessionFactory() {
       try {
          sessionFactory = new Configuration().configure().buildSessionFactory();
       } catch (Throwable ex) {
          System.err.println("Initial SessionFactory creation failed." + ex);
          throw new ExceptionInInitializerError(ex);
       }
       return sessionFactory;
    }

}
```

session.beginTransaction:
トランザクションの開始

session.save(product):
新たな行を挿入

commit():
コミット

図 II-16-5.　Hibernate を利用したプログラムの例

【解説】

1) Hibernate

Hibernate は最も広く使われている OR マッピングフレームワークの一つである。通常の Java アプリケーションで利用できるほか、Java EE 環境でも利用されている。Hibernate プロジェクトで開発されているモジュールの一部を以下にあげる。

* Hibernate Core
  Hibernate 本来の API と、XML によるマッピングの仕組みを提供する。
* Hibernate Annotations
  アノテーションの機能を利用してマッピングを行う仕組みを提供する。アノテーションの仕様は JPA に準拠している。
* Hibernate EntityManager
  JPA に準拠した EntityManager API を提供する。
* Hibernate Validator
  アノテーションによるデータのバリデーション機能を提供する。
* Hibernate Search
  OSS の全文検索ツールである Lucene と連携した全文検索機能を提供する。
* Hibernate Tools
  Hibernate を利用した開発を支援する Eclipse のプラグインおよび Ant タスクを提供する。

2) Hibernate におけるデータベース操作

* トランザクションの開始は、org.hibernate.Session オブジェクトの beginTransaction メソッドで行う。
* コミットは、org.hibernate.Transaction オブジェクトの commit メソッドで行う。Transaction オブジェクトは Session オブジェクトの getTransaction メソッドにより取得できる。
* 新たな行の追加は、マッピングファイルで登録したオブジェクトを生成し、Session オブジェクトの save メソッドの引数に指定する。データベースへの反映はフラッシュ時(コミット時、Session オブジェクトの flush メソッド呼び出し時など)に行われる。
* データの検索は、SQL、HQL(Hibernate Query Language)、Criteria オブジェクトを利用する。
  - HQL は SQL に似たクエリ言語である。検索対象(from 句の対象)としてクラス名を指定する。Session オブジェクトの createQuery メソッドに HQL 文を渡すことで、org.hibernate.Query オブジェクトを取得する。実行結果は Query オブジェクトの結果取得用メソッドにより取得する。
  - Criteria オブジェクトを利用することで、オブジェクトの組み合わせにより検索条件を指定することができる。Session オブジェクトの createCriteria メソッドで org.hibernate.Criteria オブジェクトを取得し、Criteria オブジェクトの add メソッドで検索条件を表す Criterion オブジェクトを加える。実行結果は Criteria オブジェクトの結果取得用メソッドにより取得する。
* 既存の行の更新は、検索により取得したオブジェクトの属性値を変更することで行う。データベースへの反映はフラッシュ時に行われる。
* 既存の行の削除は、取得したオブジェクトを Session オブジェクトの delete メソッドの引数として指定することにより行う。データベースへの反映はフラッシュ時に行われる。

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	応用
習得ポイント	II -16-6. iBATIS の構成と特徴	
対応するコースウェア	第 10 回 (データベース接続･アクセスのフレームワーク)	

II -16-6. iBATIS の構成と特徴

SQL 文をマッピングファイルに記述して利用する形式の OR マッピングツールである iBATIS の概要について述べる。iBATIS の基本的な構成と特徴、マッピングの仕様、iBATIS を利用したデータベースの操作方法などについて説明する。

【学習の要点】

* iBATIS はマッピングファイルに SQL 文を直接記述するという特徴を持つ OR マッピングライブラリである。
* iBATIS の利点として、パフォーマンスチューニングが行いやすい点や、既存のデータベース構造に柔軟に対応可能な点があげられる。

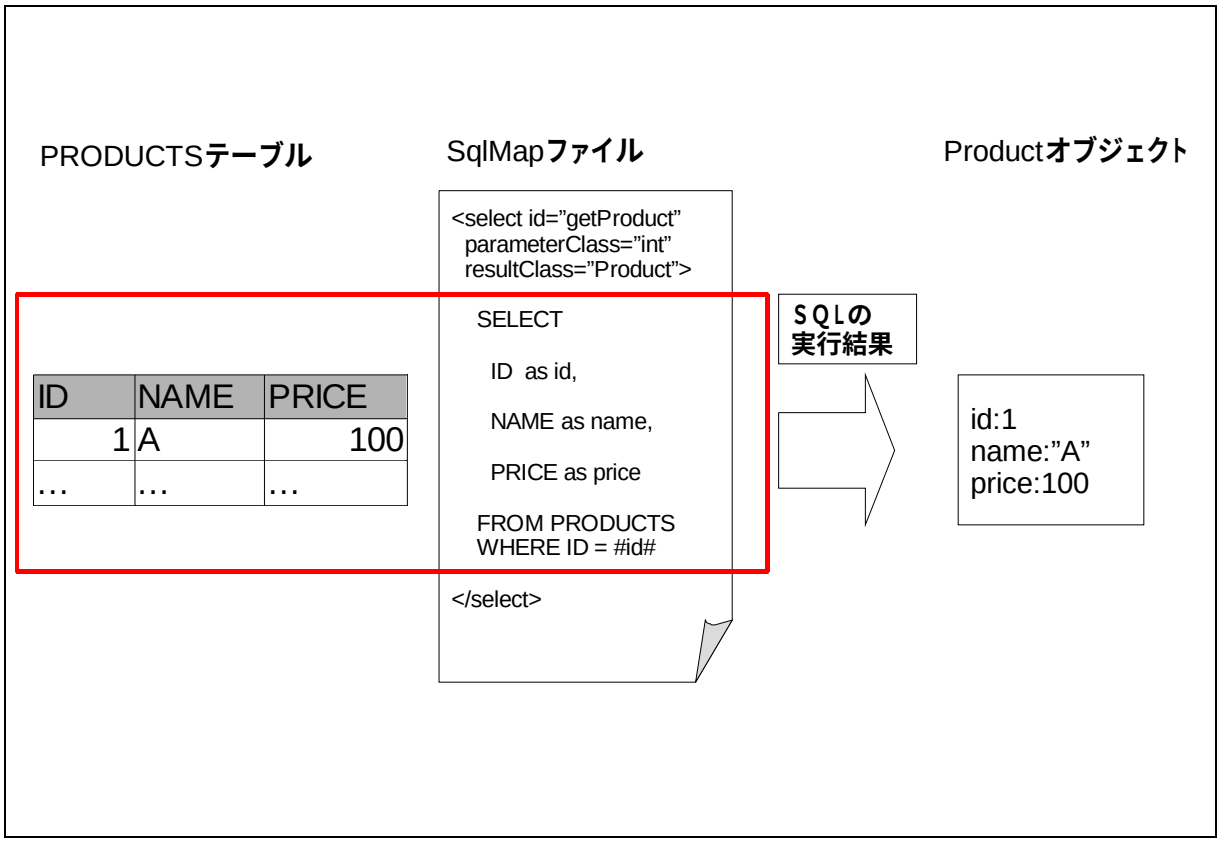

図 II-16-6.　iBATIS による SQL とクラスのマッピング

【解説】

1) iBATIS

一般的な OR マッピングライブラリ/フレームワークが、オブジェクトとデータベースレコードとの間でマッピングを行うのに対し、iBATIS にはオブジェクトと SQL の実行結果との間でマッピングを行うという特徴がある。SQL 文は SqlMap と呼ばれる設定ファイル内に記述される。一般的な OR マッピングライブラリと比較したときの iBATIS の利点を以下にあげる。

* SQL の工夫によるパフォーマンスチューニングが行いやすい。SQL が隠蔽されている OR マッピングライブラリの場合、アプリケーション上で行った操作に対応してデータベースに発行される SQL 文が非効率である場合がある。iBATIS の場合には開発者が効率を意識した SQL 文を指定することができる。
* データベースのスキーマとオブジェクトの構造が一致していない場合でも柔軟に対応することができる。例えば、複数のテーブルを結合した検索結果を単一のクラスのオブジェクトにマッピングする、といった処理を容易に行うことができる。

2) iBATIS の構成

iBATIS に含まれるコンポーネントを以下にあげる。

* iBATIS Data Mapper(SQL Maps)
  iBATIS の主要な機能である、オブジェクトと SQL の実行結果を関連付ける機能を提供する。
* iBATIS Data Access Objects
  iBATIS を利用する際、データアクセス部分の実装を隠蔽するための中間レイヤとして、DAO(Data Access Object)を導入することを容易にする。

3) iBATIS におけるマッピング

iBATIS のマッピング情報は、SqlMap ファイルに XML で指定する。SqlMap ファイル内で利用されるタグの例を以下にあげる。

* <select> - 検索を行う SQL 文を指定する。
* <insert> - 行の挿入を行う SQL 文を指定する。
* <update> - 既存の行の更新を行う SQL 文を指定する。
* <delete> - 既存の行の削除を行う SQL 文を指定する。

各タグには id 属性が必須であり、アプリケーション側から呼び出す際の識別子として利用される。

4) iBATIS におけるデータベース操作

iBATIS におけるデータベース操作は、com.ibatis.sqlmap.client.SqlMapClient インタフェースを通して行う。API の一部を以下にあげる。

* queryForList – 検索を実行し、結果を List として受け取る。
* insert – オブジェクトの挿入を行う。
* update – 既存の行の更新を行う。
* delete – 既存の行の削除を行う。

上記のメソッドは、いずれも第一引数に SqlMap ファイルで記述した id を指定する。

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	応用
習得ポイント	II -16-7. DIxAOP コンテナの仕組み	
対応する コースウェア	第 11 回 (DIxAOP コンテナの概要)	

II -16-7. DIxAOP コンテナの仕組み

Dependency Injection (DI)と Aspect Oriented Programming (AOP)の概念と利点を整理し、新しい Web アプリケーションの基盤を成す DIxAOP コンテナの仕組みと基本的な構成、特徴について解説する。

【学習の要点】

* Dependency Injection(DI)はオブジェクト間の依存関係を実行時に自動的に解決する仕組みである。
* Aspect Oriented Programming(AOP)は、ログの出力や例外処理のような、アプリケーション中の複数の箇所で横断的に利用される共通処理を分離することで、プログラムのモジュール性を高めようとするプログラミング方法である。
* DI と AOP の実行環境を提供するフレームワークとして Seasar2 や Spring などがある。

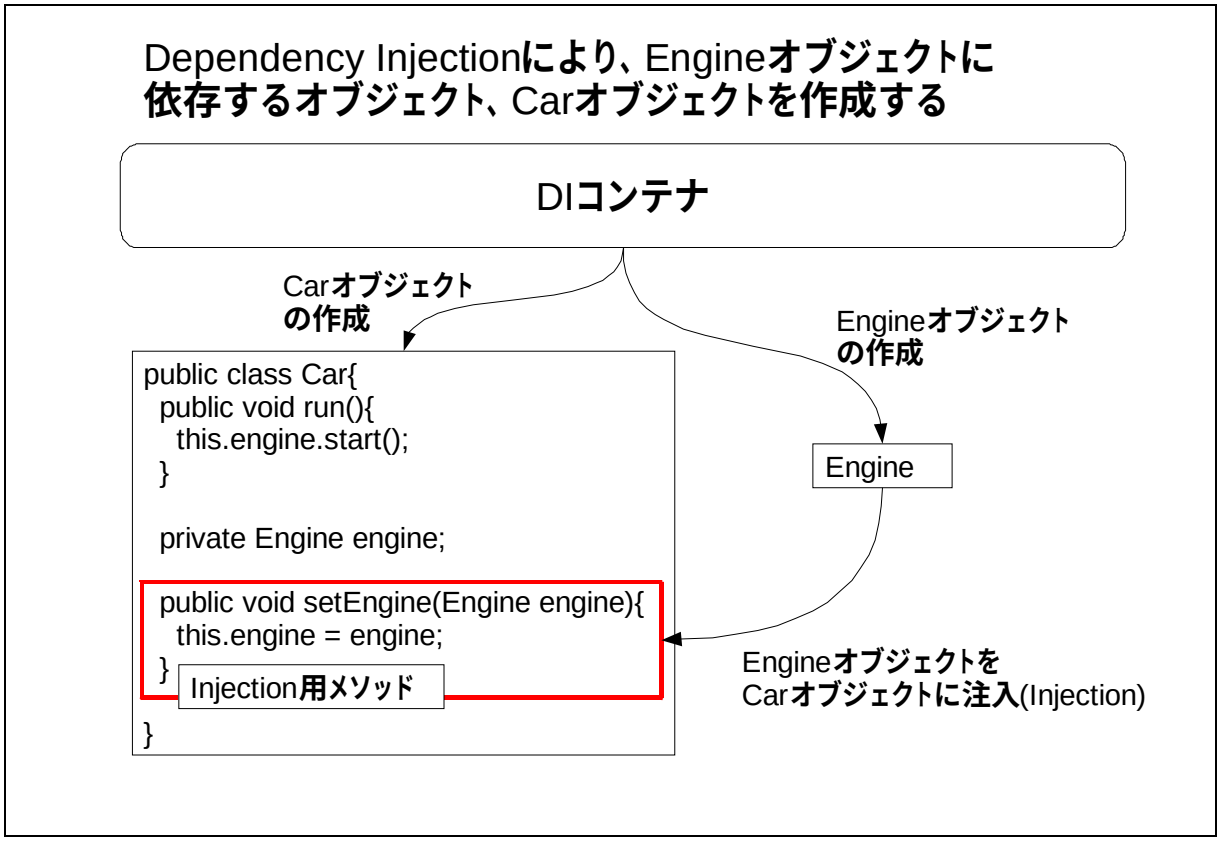

図 II-16-7.　Dependency Injection

【解説】

1) DI(Dependency Injection)

DIとは、プログラムのモジュール間の依存関係を実行時に解決する仕組みである。依存関係とは、例えばあるオブジェクトAが別のオブジェクトBを利用している状態を指す。この場合、オブジェクトAがオブジェクトBを生成した上で利用する方法が考えられる。この方法の欠点は、オブジェクトAがオブジェクトBに関する情報を持っていなければならないことである。オブジェクトAがオブジェクトBの代わりに別なオブジェクトを利用する必要が生じた場合、オブジェクトAのソースコードを書き換えなければならない。DI を利用した場合、依存するオブジェクトの生成手続きは実行環境によって自動的に行われる。そのため、オブジェクト同士の依存関係をソースコード中に記述する必要が無い。このことにより、ソフトウェアのコンポーネント化が促進され、保守のしやすい構造を実現できる。また、JNDI (Java Naming and Directory Interface) を利用した場合のような、煩雑な生成手続きを省略することが可能になる。

2) DI の実現

DI の実現方法として、オブジェクトの依存関係を設定ファイル、またはアノテーションに記述しておく方法が利用されている。DI コンテナは設定ファイル、およびアノテーションに基づいてオブジェクトの生成と注入を行う。Seasar2 のように、コンテナが自動的に依存関係の解決を行う実装も存在する。注入の方式として、setter メソッドを利用する場合や、コンストラクタを利用する場合などがある。

3) AOP(Aspect Oriented Programming)

AOPとは、プログラム中で横断的に利用される機能をアスペクトとして分離する手法である。AOPを利用することで、問題領域に特化したロジック部分を変更することなく、プログラムの複数の部分で利用される機能の修正、追加を行うことが容易になる。アスペクトとして扱うことが可能な機能の例としては、ログの出力や例外処理などがあげられる。

4) AOP の実現

AOP の実現方法として、アスペクトとなる処理を行うメソッドを実装したクラスを用意し、設定ファイル、またはアノテーションで実行条件に関する設定を行う方法が利用されている。AOP コンテナは設定ファイル、およびアノテーションに基づいて、アスペクトを適用する。AOP を利用する上では、以下のような概念を理解する必要がある。

* Advice
  アスペクトが行う処理と、実行するタイミングを表す。タイミングとは、例えば後述する Pointcut としてあるメソッドが指定された場合に、処理が実行されるのはメソッドの呼び出し前か、呼び出し後か、といった情報である。
* Joinpoint
  Adviceを適用可能な場所を表す。一般的なJoinpointとしては、メソッド呼び出しなどがあげられる。
* Pointcut
  Joinpoint のうち、実際に Advice を適用する場所を表す。具体的なメソッド名で指定する場合や、正規表現によって複数のクラス、メソッドなどを指定する場合がある。
* Aspect
  Advice と Pointcut を合わせたものを Aspect と呼ぶ。

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	応用
習得ポイント	II -16-8. Spring フレームワークの利用	
対応する コースウェア	第 12 回 (Spring フレームワーク)	

II -16-8. Spring フレームワークの利用

DIxAOP コンテナ実装のひとつである Spring Framework (Spring フレームワーク)の基本的な仕組みと基本的な構成、特徴について解説する。また Struts や Hibernate など他のフレームワークとの連携や開発ツールなど、実際の利用に有効なトピックを紹介する。

【学習の要点】

* Spring は DI と AOP の実行環境を提供する他、MVC のサポートや、トランザクション管理、およびデータベース接続の抽象化機能などを備え、Web アプリケーションの全ての層をカバーするフレームワークである。
* Spring の DI は他のフレームワークと併用することが可能であり、Controller として Struts を利用する、OR マッピングライブラリとして Hibernate を利用する、といった使い方が可能である。
* Spring の開発に利用できるツールとして、Spring IDE が Eclipse のプラグインとして提供されている。

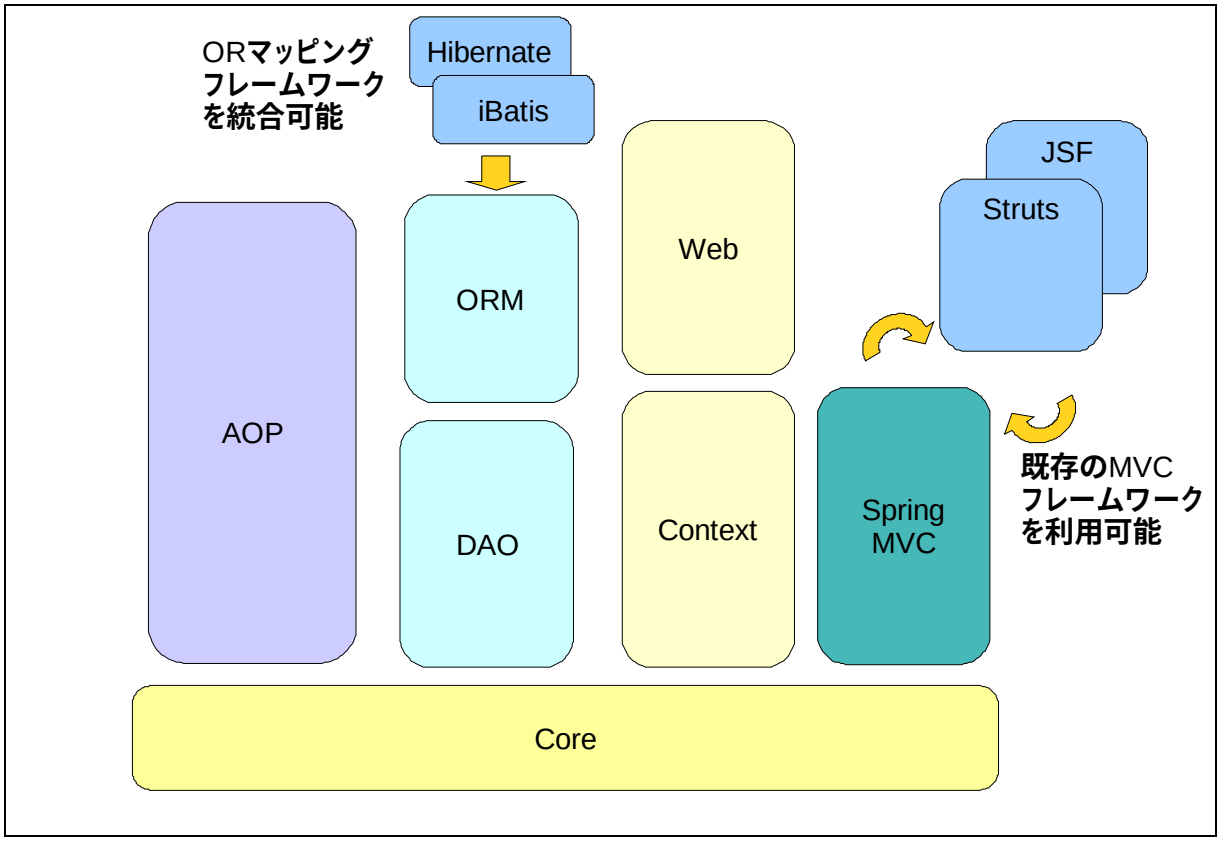

図 II-16-8.　Spring の構造

【解説】

1) Spring の構成

Spring を構成するモジュールのうち、主なものを以下にあげる。

* Core container
  DIを始めとした全ての機能の基礎となる、BeanFactoryを提供する。BeanFactoryの基本的な機能は、XML ファイルなどの定義に従って、オブジェクトを適切に組み立てることである。
* Application context module
  BeanFactory を拡張し、国際化メッセージ機能や、アプリケーションのライフサイクルにおけるイベント処理機能、バリデーション機能などを提供する。また、JNDI アクセス機能や、EJBとの統合機能など、エンタープライズ分野向けの機能を提供する。
* AOP module
  AOP 環境を提供する。AOP Alliance インタフェースに基づいたアスペクトの構築機能と、AspectJ(AOP が可能となるように拡張した Java)のサポートを提供する。
* JDBC abstraction and the DAO module
  JDBC を利用したプログラミングで発生する煩雑なコードを削減し、簡潔な DAO(Data Access Object)の構築を支援する。また、AOP を利用したトランザクション管理機能を提供する。
* Object-relational mapping integration module
  JDBC abstraction and the DAO module をベースとして、Hibernate や iBatis といった OR マッピングフレームワーク向けの DAO の構築を支援する。
* Spring MVC framework
  Spring 自身が提供する、MVC アーキテクチャによるプレゼンテーション層を構築するためのフレームワーク。

2) Spring と他のフレームワークの連携

* Struts との連携
  Struts との連携用のプラグインが用意されており、連携にあたっては struts-config.xml に登録しておく必要がある。連携方法として以下の方法がある。
  - アクションクラス内で、ApplicationContext オブジェクトを経由して、Spring 管理下の Bean を取得する。アクションクラスで DI 機能を利用することはできない。
  - リクエストが Spring 管理化のアクションクラスに委譲されるように、Struts の processorClass として DelegatingRequestProcessor を指定する。アクションクラスは通常の Bean と同様に、Spring のコンテキストに登録する。この場合、アクションクラスで DI 機能を利用できる。
* Hibernate との連携
  Hibernate との連携にあたっては、抽象化レイヤとして HibernateTemplate を利用することができる。HibernateTemplate を利用することで、セッションのオープンとクローズといった手続きを Spring にまかせることができる。HibernateTemplate の設定情報として、SessionFactory の生成に必要な情報(DataSource クラス、Hibernate マッピングファイルなど)を Spring コンテキストで指定する。

3) Spring アプリケーションの開発ツール

Spring を利用したアプリケーションの開発を支援するツールとして Eclipse のプラグイン「Spring IDE」が公開されている。

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	応用
習得ポイント	II -16-9. Seasar2 の構成と特徴	
対応する コースウェア	第 13 回 (Seasar2)	

II -16-9. Seasar2 の構成と特徴

DIxAOP コンテナ実装のひとつである Seasar2 の基本的な仕組みと基本的な構成、特徴について解説する。また Struts や JSF、Hibernate など他のフレームワークとの連携や開発ツールなど、実際の利用に有効なﾄﾋﾟｯｸを紹介する。

【学習の要点】

* Seasar2 コンテナは、DI と AOP の実行環境を提供するほか、ソースコードを修正した際、再デプロイ、およびアプﾘｹｰｼｮﾝサーバの再起動なしに修正を認識する HOT deploy 機能を備える。
* Seasar プロジェクﾄ内には、Seasar2 コンテナが提供する DI/AOP/HOT deploy 機能を、Struts や JSF など既存のフレームワークと共に利用することを目的とした様々なサブプロジェクﾄが存在する。
* Seasar2 の開発に利用できるツールとして、利用するフレームワークに応じた Eclipse のプラグインが提供されている。

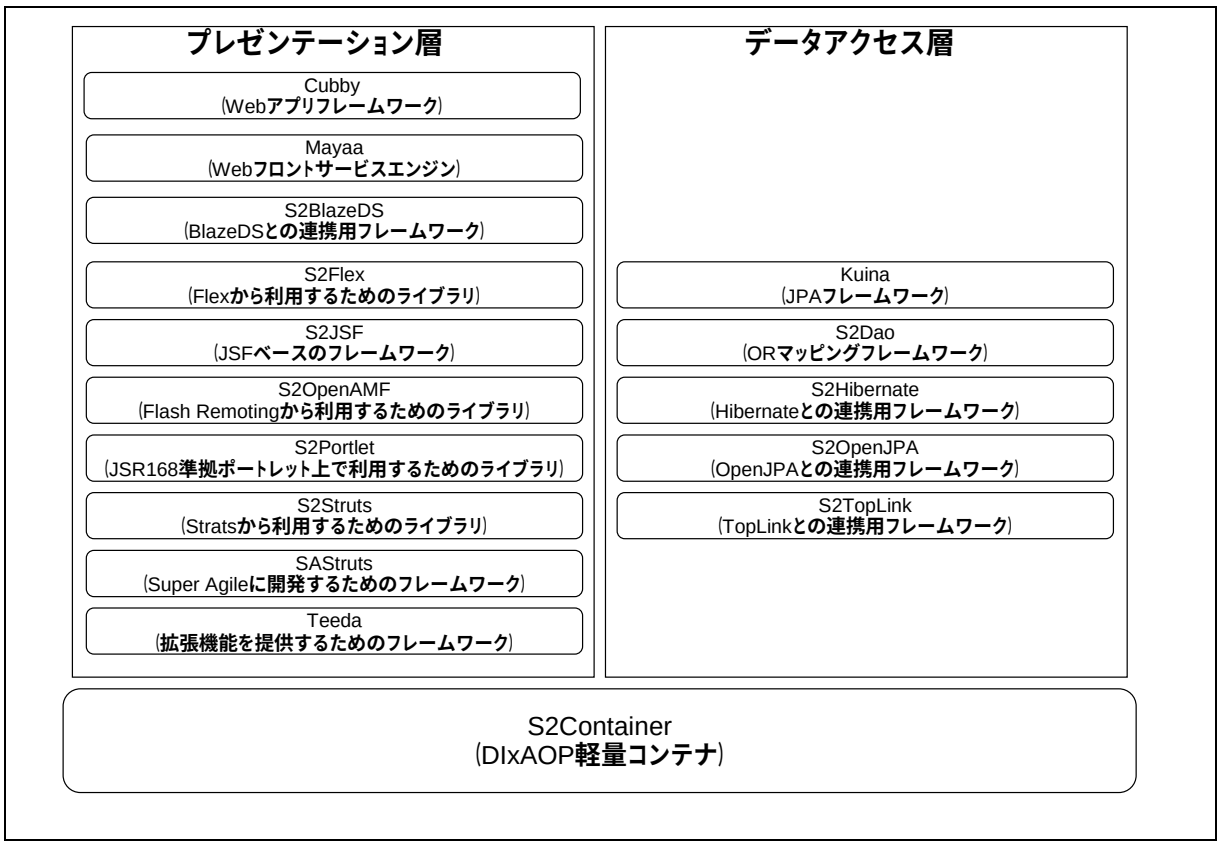

図 II-16-9. Seasar2 プロジェクト

【解説】

1) Seasar2 の設定

Seasar2 は Spring と同様、DI と AOP の環境を提供するコンテナ/フレームワークである。Spring では従来、設定の多くを XML ファイルで行っていたために、設定ファイルの記述が膨大なものとなり、開発者の負担となる場合があった。Spring2.5 では、アノテーションを利用することで XML ファイルによる設定量を減らすことが可能になっているものの、コンテナの動作を開発者が明示する、という方針は変わっていない。Seasar2 では Spring 以上に設定量の削減に力が入れられており、コンテナが自動的に設定を行う機能が充実している。以下に例をあげる。

* 自動バインディング
  DI を利用する際に、依存関係を明示していない場合でも、インタフェースやクラス、属性名などによってコンテナが自動的に依存関係を解決する。
* コンポーネントの自動登録
  ファイルシステム、および jar ファイルからクラスを検索し、Seasar2 のコンポーネントとして自動的に登録する。コンポーネントに対する名前付けも自動的に行うことができる。
* アスペクトの自動登録
  アスペクトの適用先となるクラスのクラス名のパターンや、実装インタフェースを指定することで、複数のクラスに対し、自動的にアスペクトを登録する。

2) HOT deploy

Seasar2 には HOT deploy と呼ばれる、コンテナに登録されたクラスを自動的にロードし直す仕組みが存在する。HOT deploy 機能により、ソースコードを修正した際に、アプリケーションサーバの再起動や、プログラムの再デプロイを行うことなく、修正結果を確認することが可能になる。

3) Seasar2 と他のフレームワークの連携

Seasar プロジェクトには、Seasar2 で既存のフレームワークを利用することを目的とした様々なサブプロジェクトが存在する。以下に例をあげる。

* Teeda
  JSF に DI と AOP の機能をベースとした拡張機能を提供し、より開発しやすい環境を提供する。View には JSP の代わりに HTML を利用する。
* SAStruts(Super Agile Struts)
  Struts をベースとした、POJO と DI を中心とするフレームワークを提供する。アクションクラスやアクションフォームには POJO を利用する。アクションクラス内で必要なオブジェクトはアノテーションによる DI で取得する。
* S2Hibernate-JPA
  Seasar2、Hibernate3、Hibernate Annotations、Hibernate EntityManager を連携させ、Seasar2 のアプリケーション内で JPA を利用することを容易にするフレームワーク。

4) Seasar2 アプリケーションの開発ツール

* Kijimuna
  Eclipse プラグイン。Seasar2 の設定ファイルである Dicon ファイルの編集を支援する。
* Dolteng
  Teeda 向けの Eclipse プラグイン。ソースコードの自動生成など、様々な機能を提供する。

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	応用
習得ポイント	II -16-10. Tapestry の構成と特徴	
対応する コースウェア	第 14 回（Tapestry）	

II -16-10. Tapestry の構成と特徴

Web アプリケーション開発において、Web デザイナの作業とプログラマの作業を分離し、並行して開発を進めることを実現するためのフレームワークである Tapestry の基本的な仕組みと基本的な構成、特徴について解説する。

【学習の要点】

* Tapestry は Web アプリケーションの MVC のうち、主に View と Controller の機能を提供するフレームワークであり、Struts や JavaServer faces と競合する技術である。
* Tapestry では、HTML タグによって View テンプレートを記述する。ページのレイアウトを実行環境に依存せずに確認、変更することができるため、Web デザイナの作業とプログラマの作業を分離することができる。

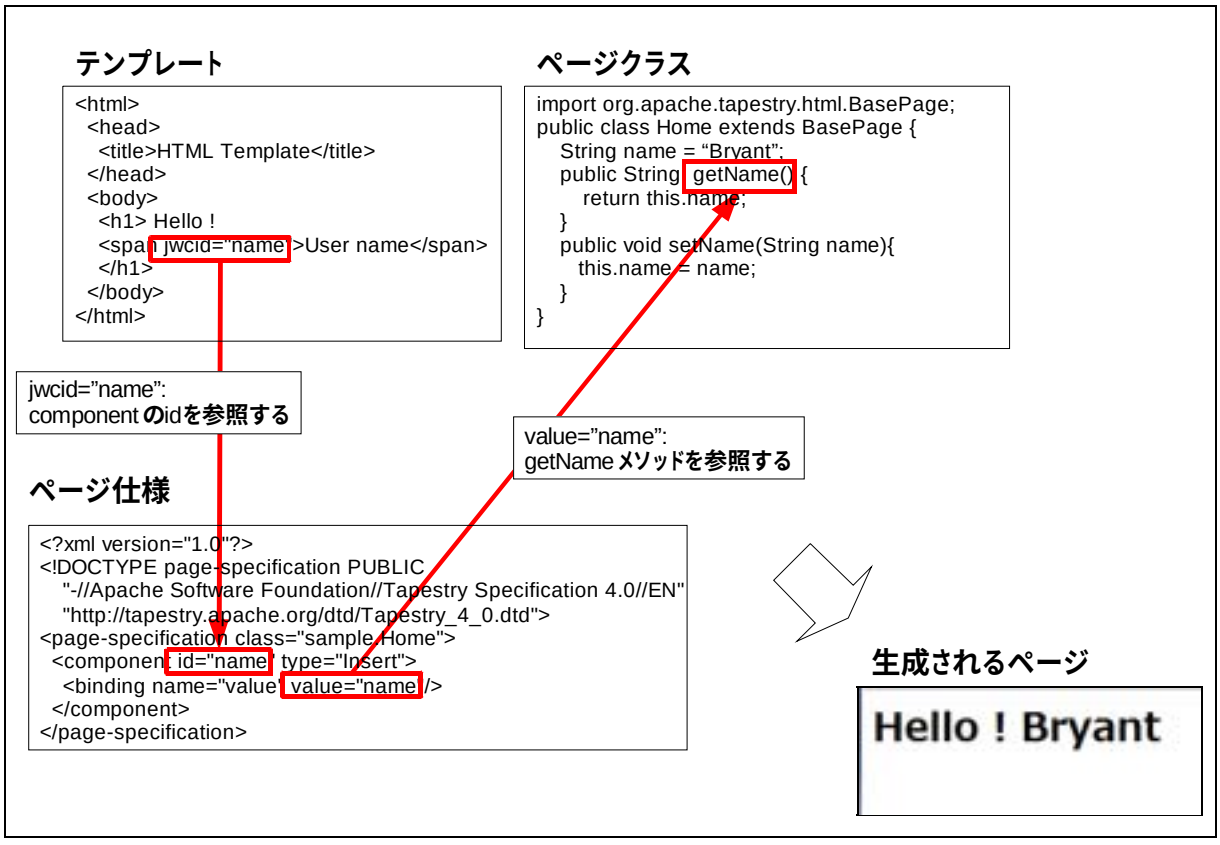

図 II-16-10.　Tapestry のページの構成要素

【解説】

1) Tapestry

Tapestry は、Struts や JSF と同様 MVC アーキテクチャを採用した Web アプリケーションの開発を支援するフレームワークである。以下のような特徴をもつ。

* HTML タグによる View テンプレート
  Tapestry では、View テンプレートを HTML タグによって記述する。JSP のように、独自のタグや記法を用いて動的要素を記述した場合、通常のブラウザでレイアウトを確認したり、Web オーサリングツールを利用してレイアウトを修正することは困難となってしまう。Tapestry は独自タグを用いないため、View に動的な要素を追加したあとでも、ブラウザや Web オーサリングツールによるレイアウトの確認、および修正を行うことが可能である。この性質により、Web デザイナの作業とプログラマの作業の分離を容易にすることができる。
* コンポーネント中心の制御
  Tapestry はコンポーネントを中心に制御を行う。Tapestry におけるコンポーネントとは、ページ、およびページを構成する部品であり、それぞれに id が割り当てられる。Tapestry では、例えば特定のメソッドの呼び出しや、ページの遷移はコンポーネントを通して行うことが可能であり、URL を意識する必要はない。

2) Tapestry アプリケーションの構造

Tapestry アプリケーションは、1 つ以上のページが集まって構成される。ページは Tapestry におけるコンポーネントの一種であり、以下の要素から構成される。

* テンプレート
  ページのビューを定義する HTML ファイル。タグ内の jwcid 属性でコンポーネントを参照することにより、動的なビューを実現する。
* ページクラス
  ページに特有なロジックを実装するための Java クラス。org.apache.tapestry.html.BasePage クラスを継承したクラスとして実装する。
* ページ仕様
  ページ内に存在するコンポーネントの宣言を行い、ページクラスとテンプレートの関連付けを行う設定ファイル。

3) Tapestry における MVC

* Model
  ビジネスロジック、およびデータベースへのアクセス処理については、ページクラスのメソッド、またはページクラスのメソッドから呼び出されるモジュールに記述する。他のフレームワークを組み合わせるなどして、多層アーキテクチャとする場合も多い。
* View
  HTML によるテンプレートファイルにより、View を定義する。
* Controller
  ページ仕様ファイルの記述、またはページクラスのメソッドによりページ間の制御を行う。