TERASOLUNA フレームワークの概要

1. TERASOLUNA フレームワークとは 2
2. 特徴 4
2.1. 設定ファイルによる開発 4
2.2. トランザクション管理 4
2.3. 拡張されたアクション 4
2.4. 拡張性 4
2.5. ユースケーススコープ 4
2.6. セッション自動削除 4
2.7. 認証・アクセス制御 5
3. ベースとするフレームワークの採用理由 6
3.1. Web アプリケーション用フレームワークに Struts を採用した理由 6
3.2. DI/AOP コンテナに Spring を採用した理由 6
3.3. O/R マッピングツールに iBATIS を採用した理由 6
4. 仕組み 8
4.1. Struts と Spring の連携 8
4.2. セッション管理 8
5. メリットとデメリット 9
5.1. メリット 9
5.2. デメリット 9
6. まとめ 10

1. TERASOLUNA フレームワークとは

TERASOLUNA は、NTT データが提供する「開発プロセス」「開発環境」「サポート」を一体としたトータルソリューションです。
TERASOLUNA の全体像は以下のようになります。

表 1 TERASOLUNA の全体像

		概要	プロダクト	入手方法など
開発プロセス		オープン系システムの開発手順	開発プロセス (BOX) など	非売品
開発環境	フレームワーク	各プラットフォームに対応したフレームワーク	TERASOLUNA フレームワーク	Apache License 2.0
	開発支援ツール	フレームワークと開発プロセスをサポートする統合開発環境 ソースコード・設定ファイルを自動作成する機能を持つ	TERASOLUNA IDE	非売品
サポート		導入支援		

2007 年、TERASOLUNA の開発フレームワークである TERASOLUNA フレームワークがオープンソース化されました。
日本を代表とする SIer である NTT データが自社開発フレームワークをオープンソース化するというニュースに非常に驚きを感じました。

TERASOLUNA フレームワークは以下のプロダクトによって構成されています。

表 2 TERASOLUNA フレームワークのプロダクト一覧

プラットフォーム	サーバーフレームワーク	クライアントフレームワーク	バッチフレームワーク
Java	TERASOLUNA Server Framework for Java (Web版)		TERASOLUNA Batch Framework for Java
	TERASOLUNA Server Framework for Java (Rich版)		
.Net	TERASOLUNA Server Framework for .NET	TERASOLUNA Client Framework for .NET	
Ajax		TERASOLUNA Client Framework for AJAX	

Java プラットフォームのプロダクトの詳細については以下のとおりになります。

表 3 Java プラットフォームのプロダクト詳細

		サーバー フレームワーク		バッチ フレームワーク
		TERASOLUNA Server Framework for Java (Web版)	TERASOLUNA Server Framework for Java (Rich版)	TERASOLUNA Batch Framework for Java
動作確認環境	対応JDK	J2SE5.0	J2SE5.0	J2SE5.0
	対応アプリケーションサーバ	Tomcat5.5	Tomcat5.5	
		WebLogicServer 9.2J	WebLogic Server10.0 WebLogic Server 9.2J WebLogic Express 9.2J	WebLogicServer 9.2J
		WebSphere6.1	WebSphere6.1	
		Cosminexus7.6	Cosminexus7.6	
		Interstage Application Server9.1.0	Interstage Application Server9.1.0	
	対応データベース	Oracle10g(10.2.0)	Oracle10g(10.2.0)	Oracle10g(10.2.0)
		Oracle 9i (9.2.0)	Oracle 9i (9.2.0)	Oracle 9i (9.2.0)
		PostgreSQL8.3 PostgreSQL8.2	PostgreSQL8.3 PostgreSQL8.2	PostgreSQL8.3 PostgreSQL8.2
ベースとするオープンソースフレームワーク	Spring Framework	2.5.1	2.5.1	2.0.6
	iBATIS	2.3.0.677	2.3.0.677	2.3.0.677
	Struts	1.2.9		
	Spring MVC		2.5.1	

以下、TERASOLUNA Server Framework for Java（Web 版）を対象とします。

2. 特徴

2.1. 設定ファイルによる開発

Struts、Spring および iBATIS を用いることにより、設定ファイルベースでアプリケーションを管理することができます。これにより、仕様変更に強いアプリケーションを実現することができます。

2.2. トランザクション管理

従来の JDBC では DB 接続やトランザクション処理を複雑に記述していましたが、TERASOLUNA フレームワークを使用することにより、自動化したり簡潔に記述することができるため、コミットミス、ロールバックミスなどを防ぐことができます。

2.3. 拡張されたアクション

TERASOLUNA フレームワークによって拡張されたエンタープライズアプリケーションで使用する標準的な機能を実現する以下のようなアクションクラスを利用することにより、新たに個々のアクションクラスを実装する必要がありません。

- 指定されたセッションを削除するアクション
- 帳票の一時保存用などに使用できるディレクトリを作成するアクション
- ビジネスロジックを実行するアクション
- ログオフ処理を行うアクション

2.4. 拡張性

ベースにしているフレームワークの拡張性を損なうことなく拡張しているため、独自に機能を拡張することが容易です。

2.5. ユースケーススコープ

ベースとしている Struts のスコープとしてはページ、リクエスト、セッション、アプリケーションの４つがありますが、TERASOLUNA フレームワークではそれを独自に拡張し、より業務に適したユースケース単位でのセッション管理を実現することができます。

2.6. セッション自動削除

セッションの削除漏れに伴うメモリ枯渇のリスクを下げ、より安全に Web アプリケーションが運用できます。

2.7. 認証・アクセス制御

ログイン情報の管理機能が実装されているため、認証や権限処理が容易になります。

3. ベースとするフレームワークの採用理由

3.1. Web アプリケーション用フレームワークに Struts を採用した理由

すでにデファクトスタンダードになっており、ノウハウを持つ開発者が豊富なため、開発者を確保することが容易です。
設定ファイルベースの開発スタイルのため、品質の統一が図れます。

3.2. DI/AOP コンテナに Spring を採用した理由

DI を用いることで変更容易性、テスト容易性、再利用性が高まります。
代表的な DI コンテナとして Spring と Seasar2 があります。Spring が採用された理由は、Spring の方が海外でより普及しているため、オフショア開発を行った場合に現地の開発者を確保することが容易です。

3.3. O/R マッピングツールに iBATIS を採用した理由

マーチン・ファウラー氏の「エンタープライズアプリケーションアーキテクチャパターン」（翔泳社）によると、データアクセスのパターンには以下の４つのパターンがあります。

- テーブルデータゲートウェイ
- 行データゲートウェイ
- アクティブレコード
- データマッパー

主要な O/R マッピングツールとして、iBATIS と Hibernate が有名であり、以下の特徴があります。

表４　iBATIS と Hibernate の比較

	記述	アーキテクチャパターン	メリット	デメリット
iBATIS	SQLに近い記述方法	アクティブレコードパターン	・シンプルな機能のため学習コストが低い ・SQLのノウハウを活用できる	ORマッピングとしてJavaBeanとRDMSとのインピーダンスミスマッチの解決方法としてはあまりオブジェクト指向的ではない
Hibernate	・SQLを抽象化した記述方法 ・DBを意識させない記述方法	データマッパーパターン	・セッションによる高速な処理ができる ・DBを意識させない開発が可能	高機能な半面学習コストが高く誤って使用するとバグの原因になる

iBATIS を採用した理由は、SQL をベースにしているため、学習コストが低く、既存の SQL のノウハウを利用することが可能であるため、開発者を確保することが容易です。

4. 仕組み

4.1. Struts と Spring の連携

Spring の機能を使用して Struts との連携を実現しています。
実現方法には表 5 の 4 つのクラスのいずれかを利用する方法があります。

表 5　Struts と Spring の連携方法

連携を実現するクラス	メリット	デメリット
ActionSupport	Actionに関する設定をSpring定義ファイルに記述する必要がないため、設定ファイルの記述量が少ない。	Actionに直接呼び出すクラスを記述するため、DI/AOPをActionに適用できない。 そのため、単体テストが実施し難く 保守性・仕様変更性が低い。
AutowiringRequestProcessor	struts-configの記述方法が同じで、Spring定義ファイルの記述量も少ない。	ActionにAOPを適用できない。 RequestProcessorを拡張して実装しているため、RequestProcessorの拡張しにくい。
DelegatingRequestProcessor	SpringにActionを管理することができるため、DI/AOPをActionに適用できる。 そのため、単体テストが実施しやすく 保守性・仕様変更性が高い。	リクエストに対するActionの管理がStrtus設定ファイルとSpring設定ファイルに分散されるため、記述量が多い。 RequestProcessorを拡張して実装しているため、RequestProcessorの拡張しにくい。
DelegatingActionProxy	SpringにActionを管理することができるため、DI/AOPをActionに適用できる。 そのため、単体テストが実施しやすく 保守性・仕様変更性が高い。	リクエストに対するActionの管理がStrtus設定ファイルとSpring設定ファイルに分散されるため、記述量が多い。 記述量はもっとも多い。

TERASOLUNA フレームワークでは、DelegatingRequestProcessor クラスを用いて連携しています。
DelegatingRequestProcessor クラスを用いた理由は、DelegatingRequestProcessor クラスと DelegatingActionProxy クラスだけが Spirng の DI と AOP の両方の機能を Action に適用でき、その中で DelegatingRequestProcessor クラスがより設定ファイルの記述量が少ないため採用されたと思われます。

4.2. セッション管理

特徴であげたセッション自動削除とユースケーススコープを実現するために ActionForm を管理する RequestProcessor クラス（実際には Spring と連携させているため DelegatingRequestProcessor クラス）を拡張して『_』が先頭に付加されるアクションフォームを自動的に削除する機能を実現しています。また、この機能を使用することによって、ユースケーススコープでのセッション管理が可能になります。

5. メリットとデメリット

5.1. メリット

- 車輪の再発明が不要

  一般に Struts、Spring、iBATIS の各フレームワークを連携して Web アプリケーション開発を行う場合、その各フレームワークを連携させた後に、ログイン処理やセッション管理などの機能を追加してアプリケーション基盤を構築します。TERASOLUNA フレームワークでは、そういった各機能が用意されているため、スピーディまた安全にアプリケーション開発を行うことができます。

- 開発者の確保

  デファクトスタンダードのフレームワークを採用していることから、熟練した開発者をたやすく集めることができます。

5.2. デメリット

- 開発支援ツール

  設計書からソース・設定ファイルを自動生成できる開発支援ツール TERASOLUNA IDE が一般に公開されていないことが残念です。

6. まとめ

TERASOLUNA フレームワークは、NTT データのエンタープライズアプリケーション開発におけるノウハウを結実し、すぐにエンタープライズアプリケーションを開発できる完成度を備えたフレームワークとして、一般に無料で公開されたことは非常に価値があります。

また、日本語の詳細なドキュメントがそろっているため、学習コストが低く経験が浅い開発者でも高い効率で開発できるフレームワークです。

「5.2 デメリット」でも取り上げましたが、より開発効率を上げることができる TERASOLUNA IDE が公開されることを切に願います。

開発部
荒川　正義