Javaアプリケーション
障害解析支援ツール
Excat for Java Version 2.0

2009年10月9日

株式会社アイ・プライド

Excat for Java

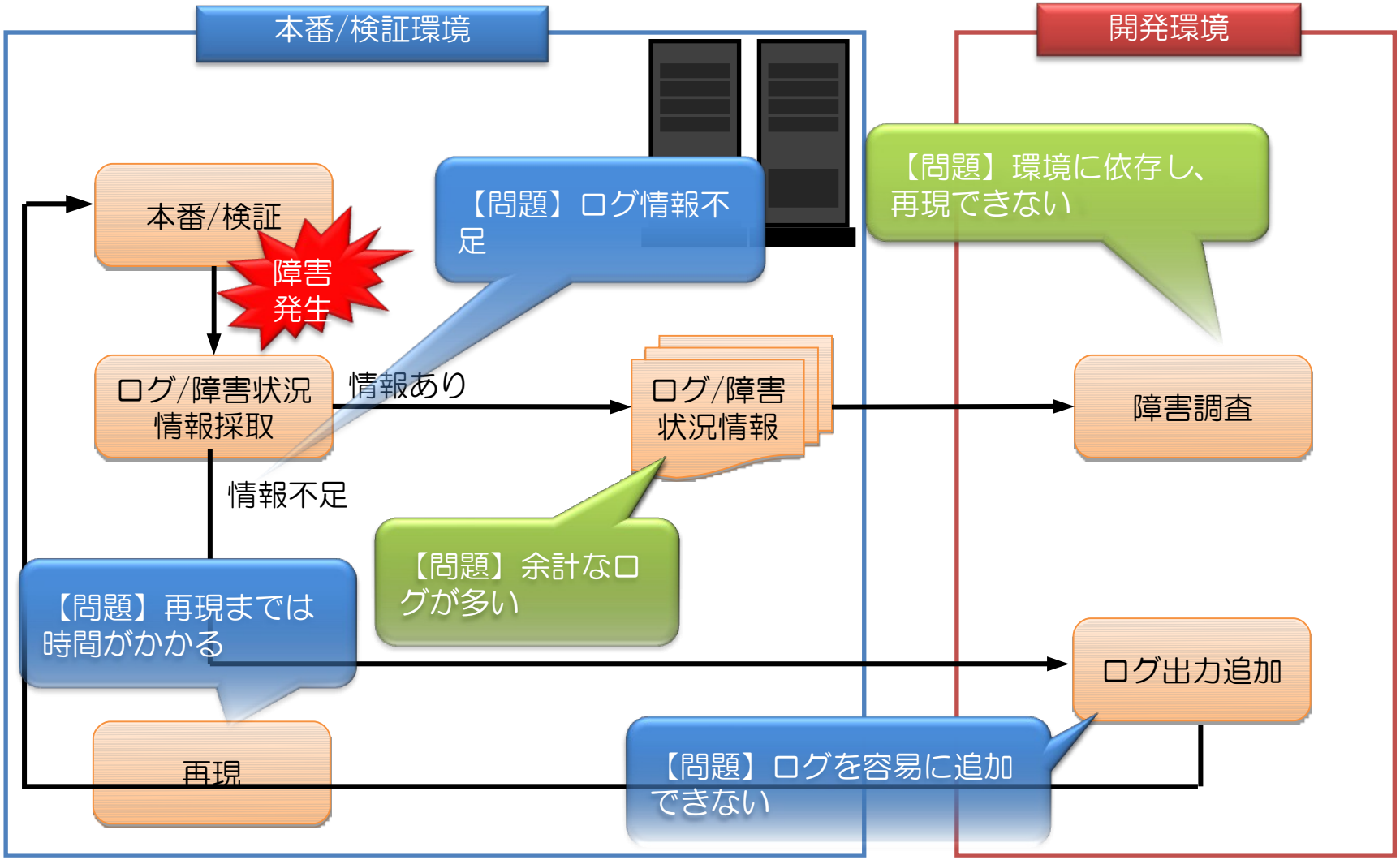
Javaアプリケーション障害の原因解析を支援する
障害情報取得・表示ツール

障害時のアプリケーション等の実行状態を
タイムリーに取得し、可視化させる！

結合・総合テストにおける障害調査の効率
をアップさせる！

本番環境での障害調査をいち早く可能にす
る！

障害対応の現状



障害対応時の問題

ログ情報不足

・ログ設計の不備
・パフォーマンスを考慮したログレベル制限

ログを容易に追加できない

・本番/検証環境などに容易にログを追加できない

環境依存

・データ依存
・設定依存
・システム構成の差異
・発生タイミング

余計なログが多い

障害の再現困難

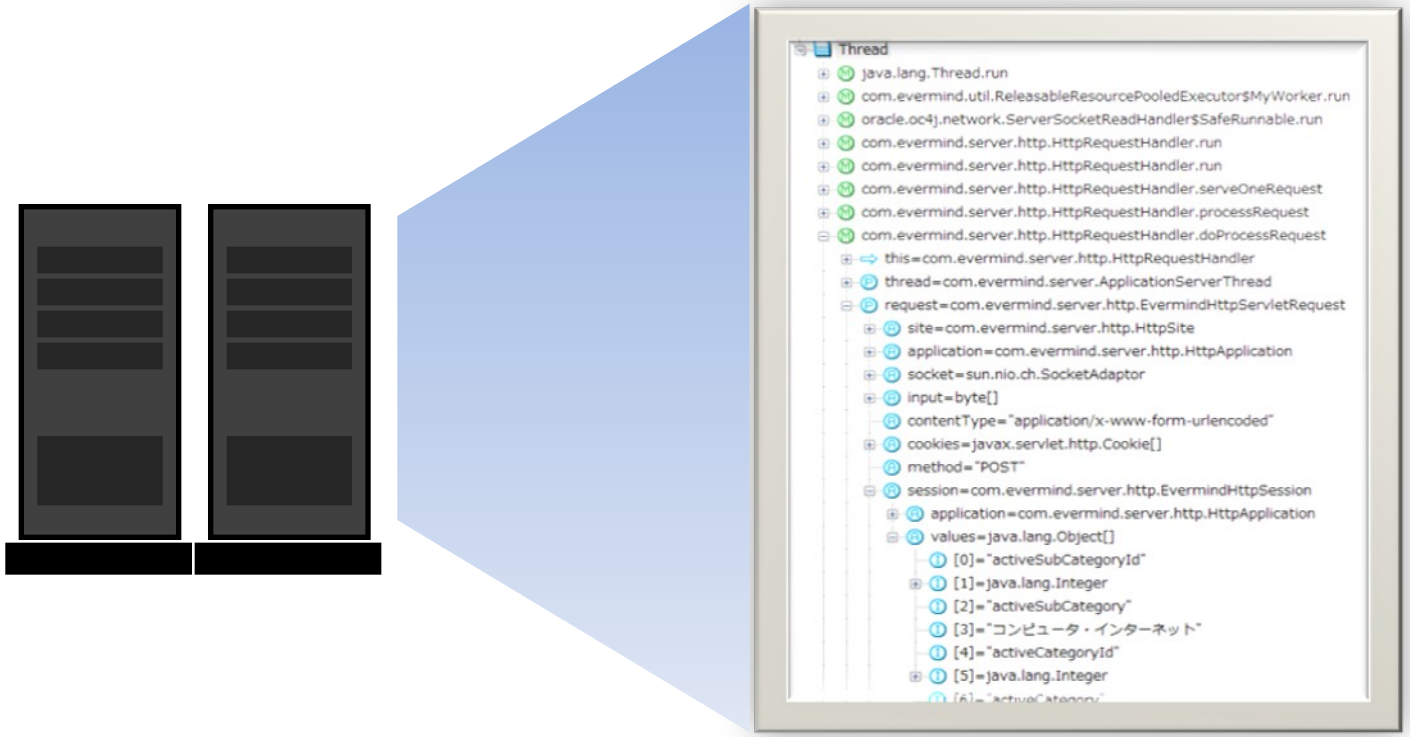
ログの特定困難

サーバーからタイムリーに適確な情報取得ができない

障害原因を速やかに解析することは困難

アイ・プライドのご提案

障害発生時に適確な情報をタイムリーに取得することによって、障害解析の効率をアップさせる。



Excat for Javaの概要

情報取得のタイミング

- 指定された例外が発生した時
- 指定されたメソッドが呼ばれた時
- 外部シグナル（Windowsの場合、[Ctrl]＋[Break]を押す）

取得できる情報

- スタック情報
- メソッドのパラメータ
- メソッドのローカル変数
- メソッドのthisオブジェクト
- 指定された他のオブジェクト

その他

- 情報取得のタイミングで指定したメールアドレスへ通知

Excat for Javaの特長

障害発生時のアプリケーションの状態をスナップショットとしてまるごと保存

・ログ情報不足を解決します
・障害の情報のみを保存するので余計なログからも解放されます

スナップショットの取得設定をリアルタイムに変更可能

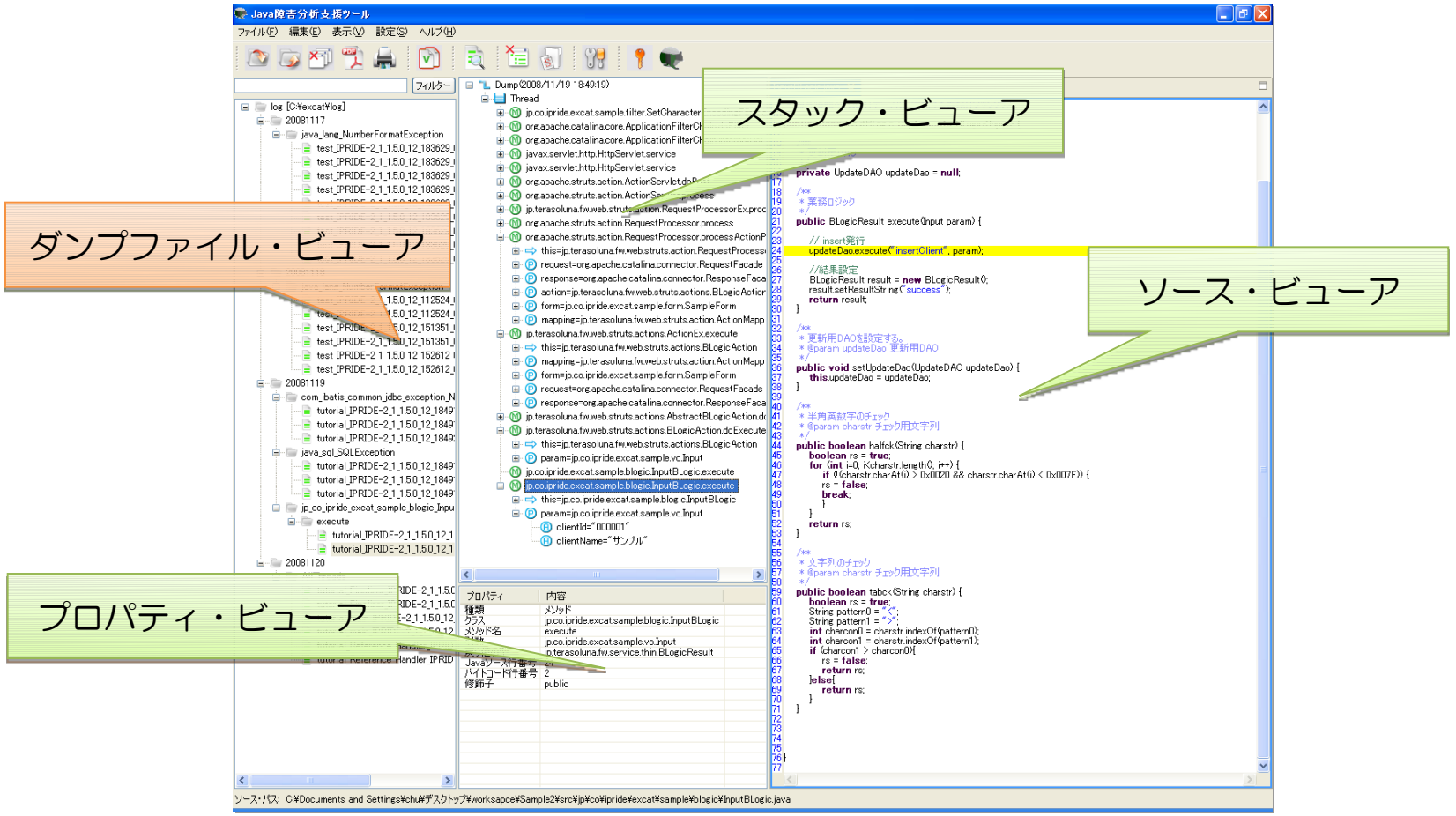
・ログ出力のためのコード追加が不要です
・変更を適用するのにAP等の再起動も必要ありません

本番/検証環境から適確な情報取得

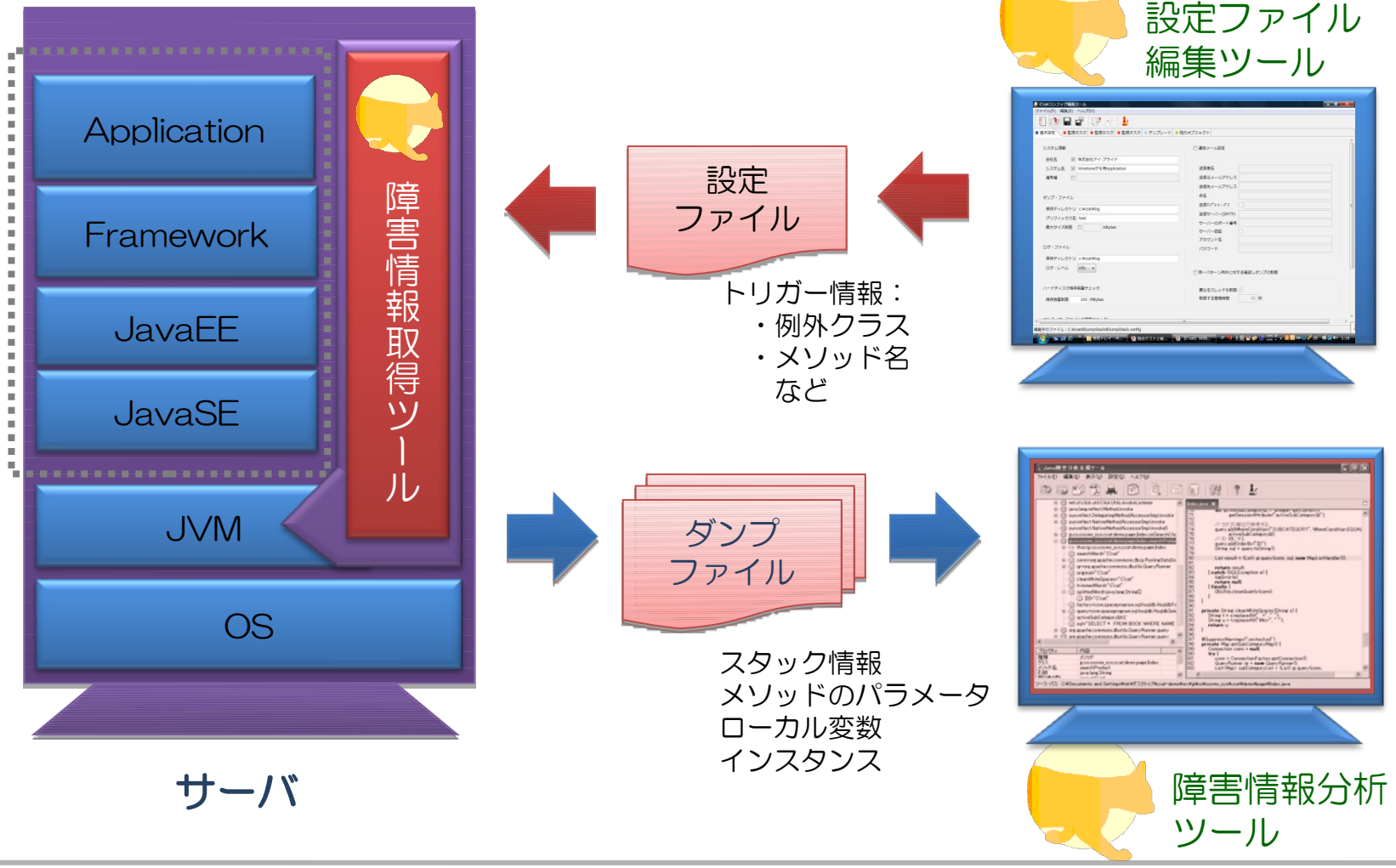
・環境に依存して発生した障害に対して開発環境での再現を待たずとも、原因特定の可能性が高まります。

取得情報の表示

- 実行の流れに沿った表示
- 対応ソースコードの表示



Excat for Javaの構成と動作イメージ



適用環境

- 障害情報取得ツール

以下のプラットフォームで稼動可能です。

- Windowsサーバ（CPU: x86 32/64bit)
- Linux（CPU: x86 32/64bit)
- Solarisサーバ（CPU: SPARC, x86)
- ＨＰ-UXサーバ（ＣＰＵ：Itanium 2）
- IBM AIXサーバ（ＣＰＵ：POWER）

以下のWebアプリケーションサーバでの稼働を検証しました。

- Cosminexus
- Interstage
- Weblogic
- Websphere
- Oracle Application Server
- Adobe ColdFusion
- Tomcat
- JBoss

- 設定ファイル編集ツール＆障害情報分析ツール

以下のプラットフォームで稼働可能です。

- Windows、Linux

デモストレーション

デモ事例

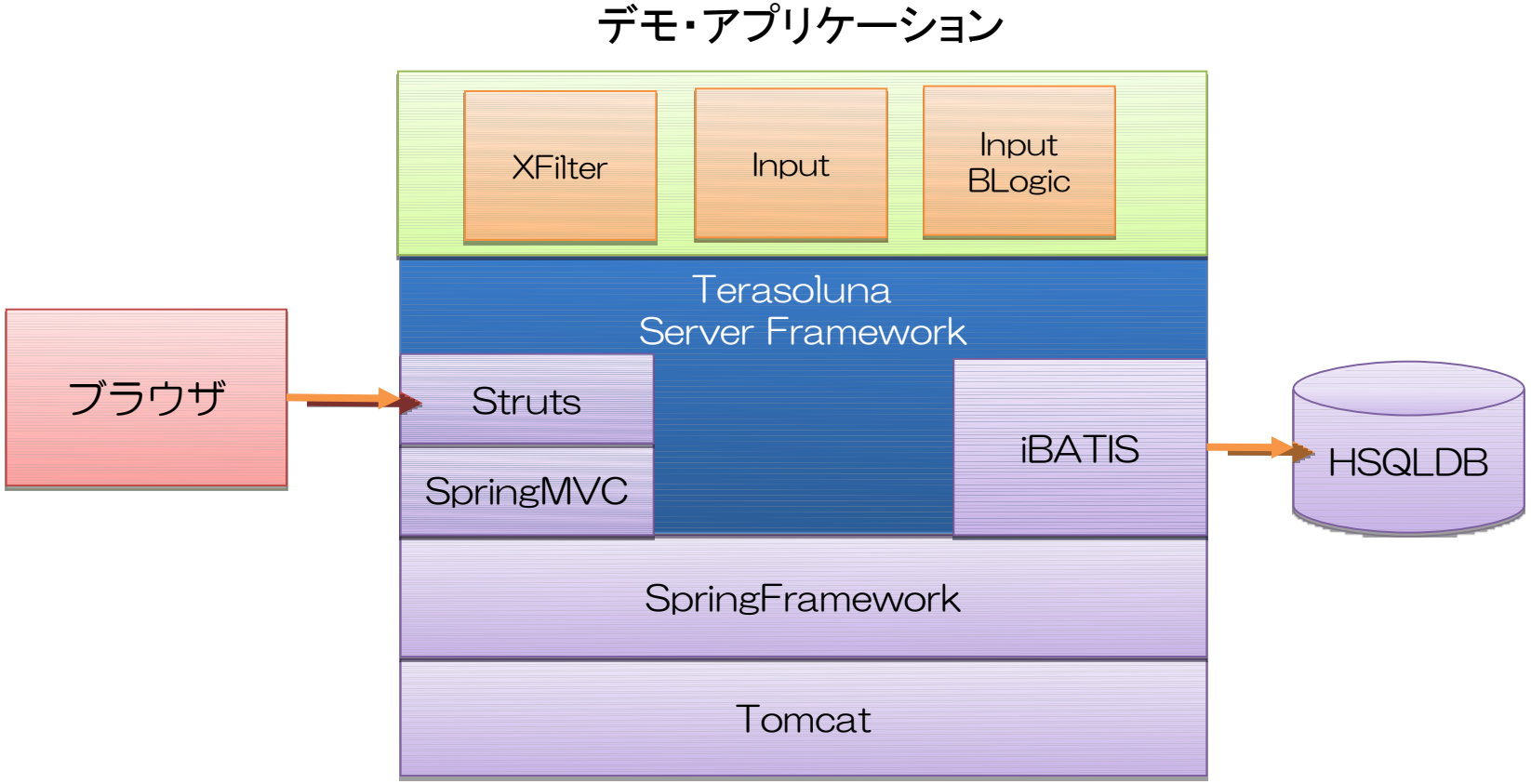
①システム構成
- J2EEサーバ：Tomcat
- フレームワーク：Terasoluna、SpringFramework、Struts、iBATIS等
- DB：HSQLDB
- AP：Webシステム

②障害現象：
- 顧客を登録したところ：エラー・ページが表示されました。
- ログ情報から、一意制約に違反したために例外SQLExceptionが発生したということが読み取れるが、原因がわからない。

③Excatの設定
- 方法１：このような障害を起こすExceptionを常時監視。
- 方法２：業務処理を呼出すメソッドを監視。

④Excatによる解析結果
- 一意制約に違反したデータと、データ・フローがわかった。

デモ・システムの構成


デモ・アプリケーション
XFilter
Input
Input
BLogic
Terasoluna
Server Framework
ブラウザ
Struts
SpringMVC
iBATIS
HSQLDB
SpringFramework
Tomcat

Excat for Javaの適用事例

事例１：原因不明の異常終了

１．システム構成
・J2EEサーバ：Tomcat
・DB：PostgreSql
・AP：Webシステム
２．障害事象
ある原因不明な異常終了。

３．Excat for Javaの設定
Exceptionを監視（Tomcat対応の設定ファイルのテンプレートを利用）。

４．障害再現
Excat for Javaによって以下の情報を取得。
①SQLExceptionが発生したスレッドの情報。

５．Excatによる解析結果
該当SQLExceptionをキャッチしてからの処理にバグがある。

６．原因と分析
該当バグによって後続の処理では想定していない問題が発生し、
処理は異常終了となった。

事例２：原因不明のロールバック

１．システム構成
- J2EEサーバ：Weblogic
- DB：PostgreSql
- AP：基幹システム

２．障害事象
- DBのロールバックが時々、発生。
- ロールバックの発生と同時にjava.sql.SQLExceptionと例外XXXExceptionが発生。

３．Excat for Javaの設定
- トランザクションのロールバック・メソッドを監視。
- XXXExceptionを監視。

４．障害再現

Excat for Javaによって以下の情報を取得。

①ロールバック・メソッドを実行しているスレッドの情報（スレッドA）

②同時にXXXExceptionが発生したスレッドの情報（スレッドB）

事例２：原因不明のロールバック（続き）

５．Excatによる解析結果
①スレッドAから、以下の結果が得られた。
・タイムアウトの設定時間は４時間である。
・ロールバックの原因はトランザクションのタイムアウトである。
②スレッドBから、以下の結果がえられた。
・JMSで送信されてきたCSVファイルをレコード毎に処理している。
・トランザクションはCSVファイル毎にコミットしている。
・処理しているCSVファイル内のレコード件数は４５００件であり、
このうち、４０００件は処理していたが、未完了である。
・この処理はXXXExceptionによって中断されている。
・ XXXExceptionはタイムアウトのSQLExceptionによって発生している。

６．原因と対策
①ロールバックの原因はトランザクションのタイムアウトである。
②対策としては、CSVファイル内のレコードの最大件数を制限すること、
若しくは、トランザクションのタイムアウト時間を延ばすことである。

事例３：メソッド呼び出し構造調査

１．システム構成
・J2EEサーバ： なし
・フレームワーク：不明
・DB：PostgreSQL
・AP：バッチ処理のパッケージ

２．障害事象
・APの処理時間が異常に長い。
・他のプロファイラツールから、Aメソッドの実行時間が長いことと、Bメソッドの実行回数が多いことが判明。
・しかし、該当パッケージの構造は複雑でわからないため、構造上の改善ができない。

３．Excat for Javaの設定
・Aメソッド、Bメソッドを監視。

４．再現
①Aメソッドに対して呼び出し経路をすべて取得。
②Bメソッドに対する呼び出し経路をすべて取得。

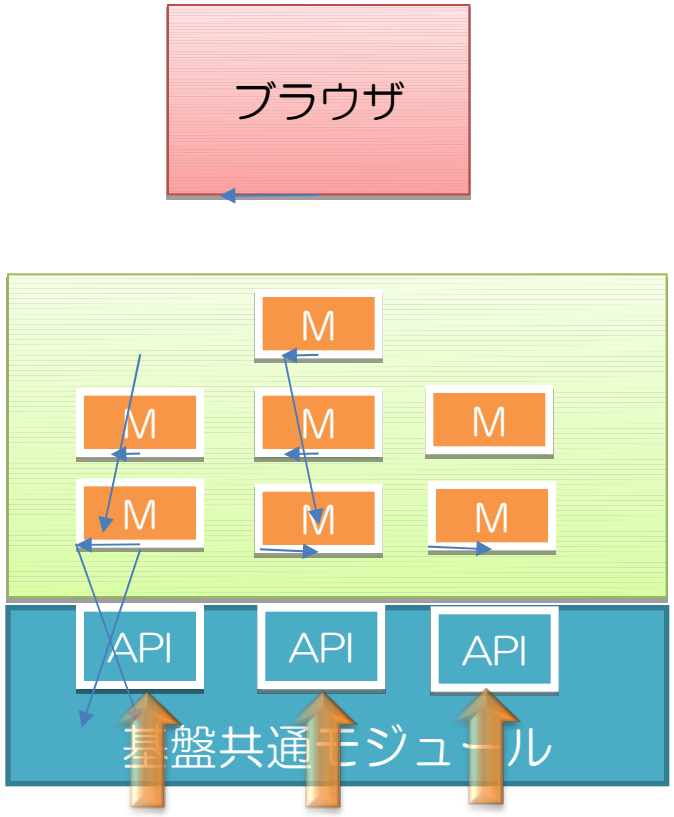
５．Excatによる解析
AメソッドとBメソッドの呼び出し経路とデータフローが判明。

事例４：結合テストの適用

１．システム構成
- J2EEサーバ： Tomcat
- フレームワーク：Struts
- 基盤：基盤共通モジュール
- AP：某業務システム

２．アプリケーションの構造
- 業務APは基盤共通モジュールのAPIを呼び出し。
- 基盤共通モジュールは他のAPを呼び出して処理した結果を業務APに返す。
- 業務APは基盤共通モジュールから得られた結果をブラウザに表示。

３．Excat for Javaの適用概要
- 基盤共通モジュールのAPIを監視。
- 障害発生時に、Excatで得られたデータを使って、該当APIを呼び出しの制御フローとデータ・フローを確認。



Excat監視

事例５：エラー情報不足

１．システム構成
・J2EEサーバ： 不明
・フレームワーク：不明
・DB：不明
・AP：業務システム

２．障害事象
・log4j.errorによるエラー・ログが発生。
・情報不足のため、原因特定ができない。

３．Excat for Javaの設定
・ log4j.error()のメソッドを監視。

４．再現
エラーが発生した箇所の実行経路とデータを取得。

５．Excatによる解析
エラーが発生した箇所までの実行経路とデータフローが判明。

有難うございました