他システムとの接続方法

インターシステムズジャパン株式会社

InterSystems®
Health | Business | Government

他システム（技術）との連携の必要性

インターシステムズのテクノロジーだけでアプリケーション構築の全てを提供できない

- フロントエンドツールの不足

既存システム（サービス）との連携

主な連携手段

リレーショナルインタフェース

- ODBC
- JDBC
- ADO.NET（ADO.NET Managed Provider）
- SQLゲートウェイ

主な連携手段

オブジェクトインタフェース

- Java
  - Java eXTreme
  - Hibernate
- .NET
  - .NET eXTreme
  - .NET Entity Framework
- Node.js
- C++
- Objective-C
- Perl
- Python

主な連携手段

その他

- コールイン
- コールアウト・ゲートウェイ
- .NETゲートウェイ
- Javaゲートウェイ（Ensemble、HealthShareのみ）
- ウェブサービス(SOAP)
- HTTP
  - REST
  - JSON
  - Web Socket

- TCP/IP ソケット通信
- モバイルでは、REST/JSON+HTTP(S)が主流

接続に際しての考慮点

インターシステムズ・テクノロジーの利点を生かす

- ビジネスロジックとデータベースエンジンの最適化
  - ビジネスロジックはできるだけサーバー側で記述する
  - データ通信（量、回数）は最低限に
  - JSON形式は複雑なデータ構造を効率良く交換可能
- SQL、オブジェクトインタフェース双方にメソッド呼出し手段あり
  - JDBC,ODBC　Call文（クラスメソッド、sqlproc属性あり）
- 時代の流れはステートフルよりステートレスへ
- それでも他言語でビジネスロジックを書きたい場合
  - eXTreme(Java,.NET)
  - Hibernate
  - .Net Entity Framework
  - Node.js

eXTreme

通信処理を最適化した高速インタフェース

- eXTreme Event Persistence (XEP)
- 動的オブジェクトAPI（Javaのみ）
- グローバルAPI
- TCP/IPを使用

ステートフル　VS　ステートレス

モバイルの特性

- 消費電力
- 無線LANの不安定性
- 通信の帯域
- スケーラビリティ

ステートレスな通信が望ましい

HTTP　+　JSON

RESTとは

- **ロイフィールディングが提唱したウェブアプリケーションのアーキテクチャ上のスタイル**
  - “表現上の状態の転送をよいウェブアプリケーションの振舞のイメージを喚起することと想定する: リンクを選択すること（状態遷移）でアプリケーションが進行中のウェブページ（仮想状態マシン）の次のページに移動し（アプリケーションの次の状態を表現しながら）、ユーザーに橋渡しされ、ユーザーの利用に合わせて表現される ”

さらにいいことには

"RESTは、コンポーネントのやりとりのスケーラビリティ、インターフェースの一般性、コンポーネントの導入非依存性、中間コンポーネント間のやりとりの遅延の削減、セキュリティの強制、レガシーシステムのカプセル化を強調している"

- Webopedia

REST

RESTは標準でもプロトコルでもなくて、アーキテクチャー上のスタイル

RESTは、既存のウェブ標準であるHTTP、URL、XML、JSONなどを使う

RESTはリソース指向
リソースまたは情報の断片をURIで指定し、サーバー/クライアント間の双方向に渡される

RESTの原則

一定のインタフェース: 簡潔にアーキテクチャーに紐づけない、その結果各部分は独立に進化する

ステートレス: クライアントのコンテキストは要求間でサーバーに保存しない
リクエストをサービスするために必要な情報はすべて毎回送る

キャッシュ可能: よく管理された部分的および完全なキャッシングがいくつかのクライアント/サーバー間のやりとりを削る
スケーラビリティと性能を改善する

RESTfulウェブサービス

RESTfulウェブサービスというのは、HTTPとRESTの原則を使って実装したウェブAPI

URIのようなディレクトリ構造で識別するリソースの集合

(https://www.googleapis.com/calendar/v3/calendars/*GlobalSummit*/events)

操作は、明示的にHTTPメソッドを基礎とする(GET, POST, PUT, DELETE)

情報は、インターネットのメディアタイプ、通常はJSONに基づき転送
他のタイプにはXML,HTML, CSV (テキスト)が含まれる

CRUD操作

REST操作はhhtpプロトコルメソッドで定義されている４つのタイプに集約される:

REST	HTTP	
Create	Post	POST https://api.twitter.com/1.1/statuses/retweet/241259202004267009.json
Read	Get	GET https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=twitterapi&count=2
Update	Put	PUT https://www.googleapis.com/calendar/v3/calendars/calendarId/events/eventId
Delete	Delete	DELETE https://www.googleapis.com/calendar/v3/calendars/*calendarId*/events/*eventId*

REST優位性

REST

- 簡潔性 (使用、保守、テストが簡単)
- 表現のたくさんな選択肢がある(JSON, CSV, HTML, XML)
- 人間が可読できる結果
- 性能
  - スケーラブルアーキテクチャ
  - 軽量要求と軽量応答
  - より簡単な応答の解析
  - 帯域の削減 (キャッシング、条件付GETなど)
  - JSON表現を使うとクライアントに適している

REST優位性

Soap要求

```
<?xmp version="1.0"?>
<soap:Envelope xmlns:soap=
   soap:encodingStyle=                                    >
<soap:Body ord=                                    ">
   <ord:GetOrderDetails>
      <ord:OrderNumber>12345</ord:OrderNumber>
   </ord:GetOrderDetails>
</soap:Body>
</soap:Envelope>
```

REST要求

http://www.igroup.com/order?ordernum=12345

 |

URL / URI

RESTインターフェースは URL/URI で定義する

URI – Uniform Resource Identifier

- ネットワーク上の特定のリソースを識別
- 例: http://www.igroup.com/order

URL – Uniform Resource Locator

- ネットワーク上のリソースの特定の表現方法のアクセスを提供
- http://www.igroup.com/order?ordernum=12345 または
- http://www.igroup.com/order/ordernum/12345

セキュリティ

セキュリティはインターフェース開発者にゆだねられる

- RESTには予め定義済メソッドはない

ウェブアプリケーションとして既に利用可能なものをおおいに利用すべし

- SSL/TLS（https:）
- OpenId Authorization (Oauth)
- Hash-based Message Authentication Code (HMAC)

Cachéでの実装

2014.1に新クラス - %CSP.Rest

SMP上でディスパッチクラスを登録
RESTアプリケーションベースURLとマッチングする

- システム>セキュリティ管理>ウェブアプケーション>ウェブアプリケーションの編集
- 新規ウェブアプリケーション
- /csp/samples/globalsummit
- Dispatch Class: Rest.Broker

UrlMap Xdataブロックを使ってリクエストをＨＴＴＰ操作とターゲットのクラスメソッドに引き渡す

- XData UrlMap {
  <Routes>
  <Route Url="/employee/html/list" Method="GET" Call="Rest.HTML:GetAllEmployees"/>
  </Routes>}

REST vs. SOAP

REST	SOAP
１つのスタイル	標準
“適切な” RESTとしては、トランスポートにはHTTP/HTTPSが必須	普通はトランスポートはHTTP/HTTPSだがほかのものでもよい
応答データは通常XMLやJSON形式で転送される 平均的にはJSONのほうが軽い（SOAPヘッダーのオーバーヘッドがない）	応答データはXML形式で転送される

REST vs. SOAP (続き)

REST	SOAP
要求はURI形式で転送 •SOAPに比較してかなり軽い •長さに制限あり •入力フォームフィールドを簡単に使用可能	要求はXML形式で転送
メソッドとURIを解析するとその意図がわかる	意図を理解するにはメッセージペイロードを解析しなければならない
	WS* イニシアティブが圧縮やセキュリティのような課題の改善に取り組む

REST vs. SOAP (続き)

REST	SOAP
JavaScriptから呼出し簡単	JavaScriptはSOAPを呼び出すことは可能だが、難しく洗練されたやりかたではない
JSONが返ってくると、非常に強力	JavaScriptのXML解析は遅くて方法がブラウザ毎に異なる

REST/JSONは以下のようなケースに最適…

限られた帯域とリソース

- 開発者定義の構造の柔軟性
- どのブラウザも利用可能

完全にステートレスな操作

- 例えば、ステートレスなCRUD操作

キャッシング状況

- RESTアプローチは情報がキャッシュできるときに非常にうまく動作する

主要な開発環境でサポート

- Java（Android含む）
- .NET
- iOS

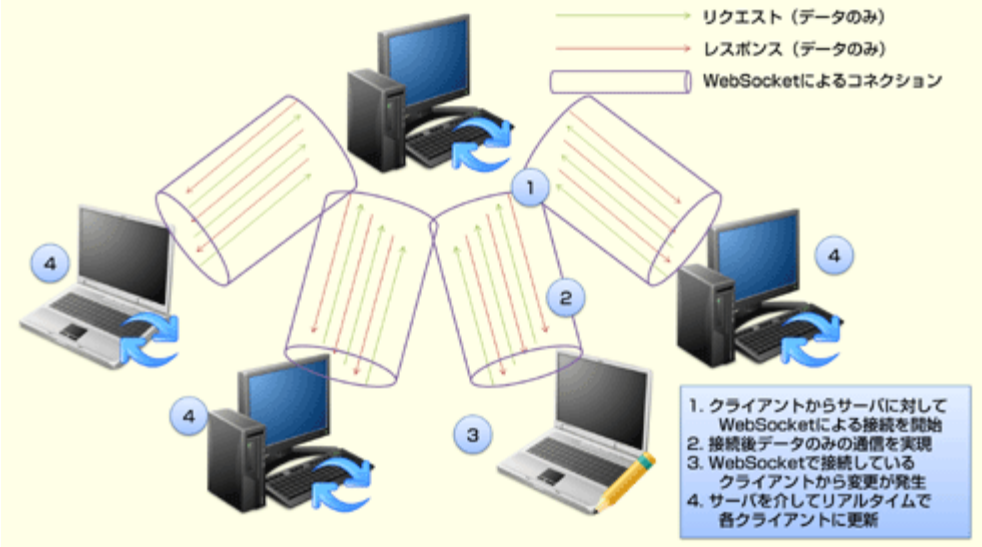
JSON vs. XML

JSON	XML
データ構造	データ構造
検証の仕組みなし	XSD
名前空間（ネームスペース）なし	名前空間（ネームスペース）あり (複数使用可)
解析は高速、特にJavascript eval()を使うと	解析にはXpathなどを使ったXMLドキュメント解析が必要

WebSocket

サーバとクライアント間は一度でも接続が確立すると、明示的に切断しない限り通信手順を意識することなくデータのやり取りをソケット通信で実施できる

WebSocketで接続が確立しているサーバとすべてのクライアントは同じデータを共有し、リアルタイムで送受信できる



他システムとの接続方法

インターシステムズジャパン株式会社