



-The WEB2.0 Connector-

T2プロジェクト

<http://code.google.com/p/t-2/>

shot(Shinpei Ohtani)

自己紹介

- 大谷 晋平 (Shinpei Ohtani)
- blog <http://d.hatena.ne.jp/shot6/>
- twitter/wassr : shot6

本日のアジェンダ

- T2ってなんだろう？
- RIA時代の到来とそのアーキテクチャ
- T2の基本モデル
- T2のデモ
- T2の開発体制とポリシー



T2ってなんだろう？(1)

- T2とは、シンプルな部品化指向のWebフレームワーク
- Webにつなげる・つながる部分に特化
 - the Web Connectorがコンセプト
- HTTPの原則に従いやすいフレームワーク
 - キレイなURL、HTTPメソッドごとにメソッドを分ける



T2ってなんだろう？(2)

- T2がやること
 - Webにつながるところをリッチに！
 - ステートレスなフレームワーク
 - コードを小さく保つ
 - マルチビューフレームワーク
 - 保守性を考慮した機能
 - 規約よりもアノテーションを重視



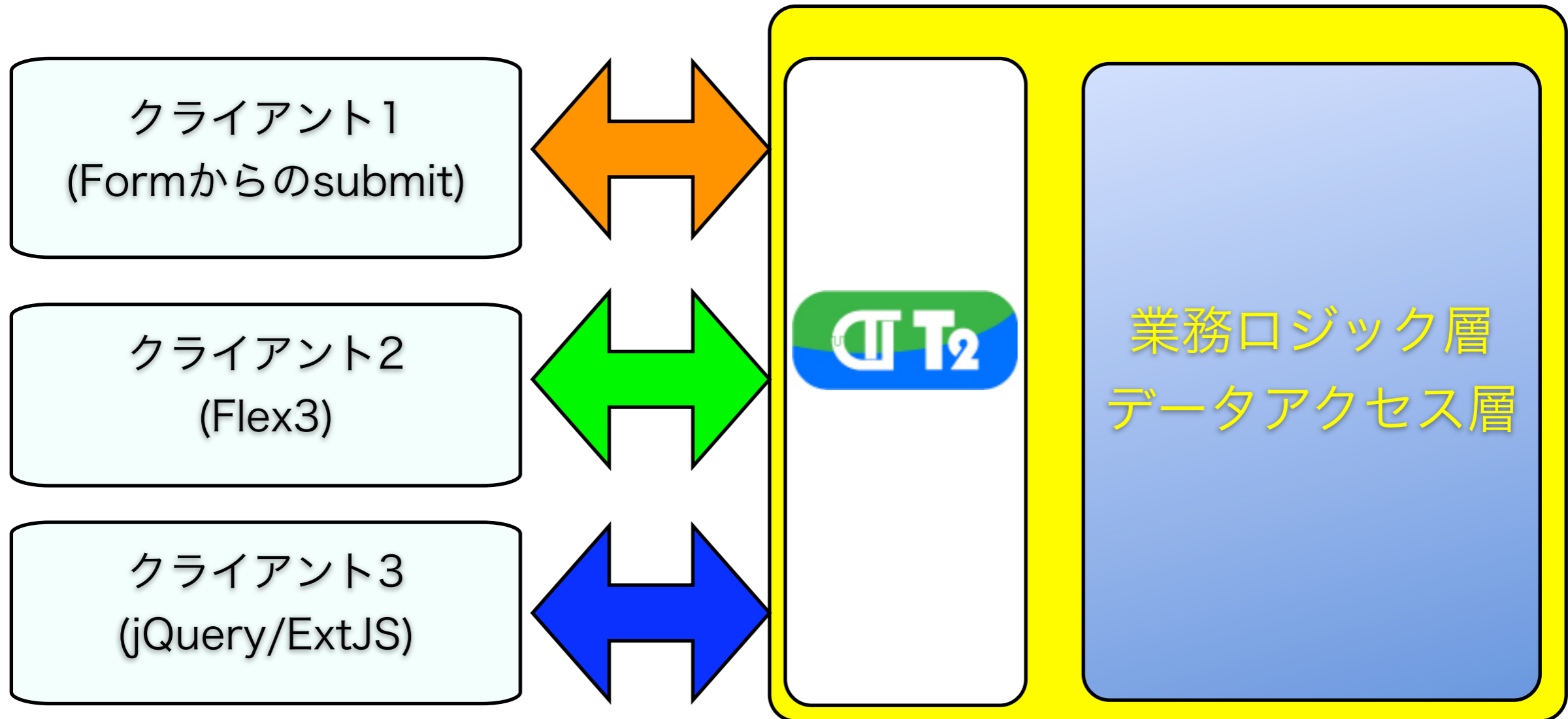
T2ってなんだろう？ (3)

- T2がやらないこと
 - バリデーション
 - 自動コンバージョン
 - 程度による. 適度なバランスが重要
 - リッチコンポーネントの開発
 - 規約ばりばりの開発スタイル

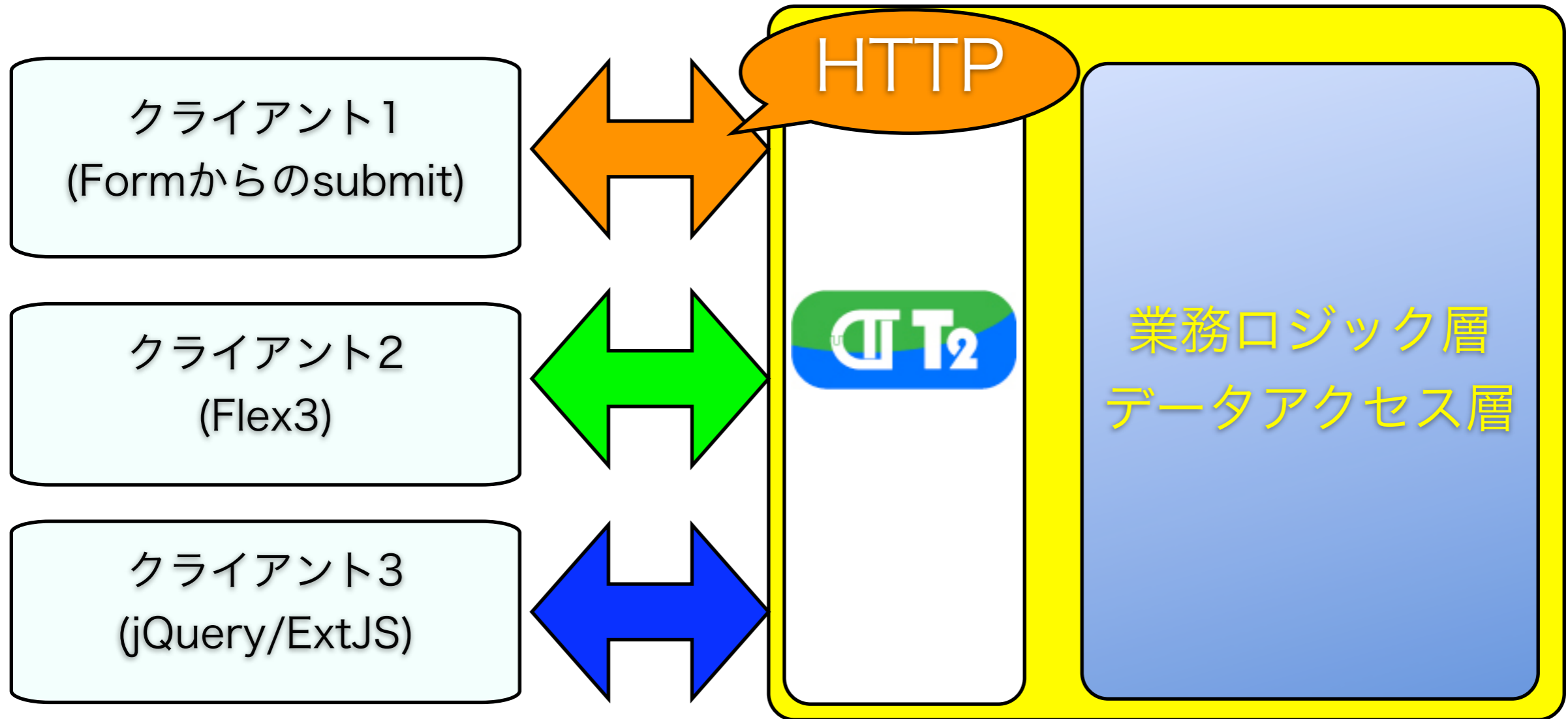
RIA時代の到来

- RIA(Rich Internet Application)時代の到来
 - jQuery/ExtJS、Flex/AIR、Silverlightなど
 - REST/JSON/POX/AMFなど多種多様なリクエストとレスポンスの組み合わせ
 - クライアントマシンのスペックの高速化
 - 時代はよりクライアントサイドのリソースをいかに有効に使うか、にシフトしている

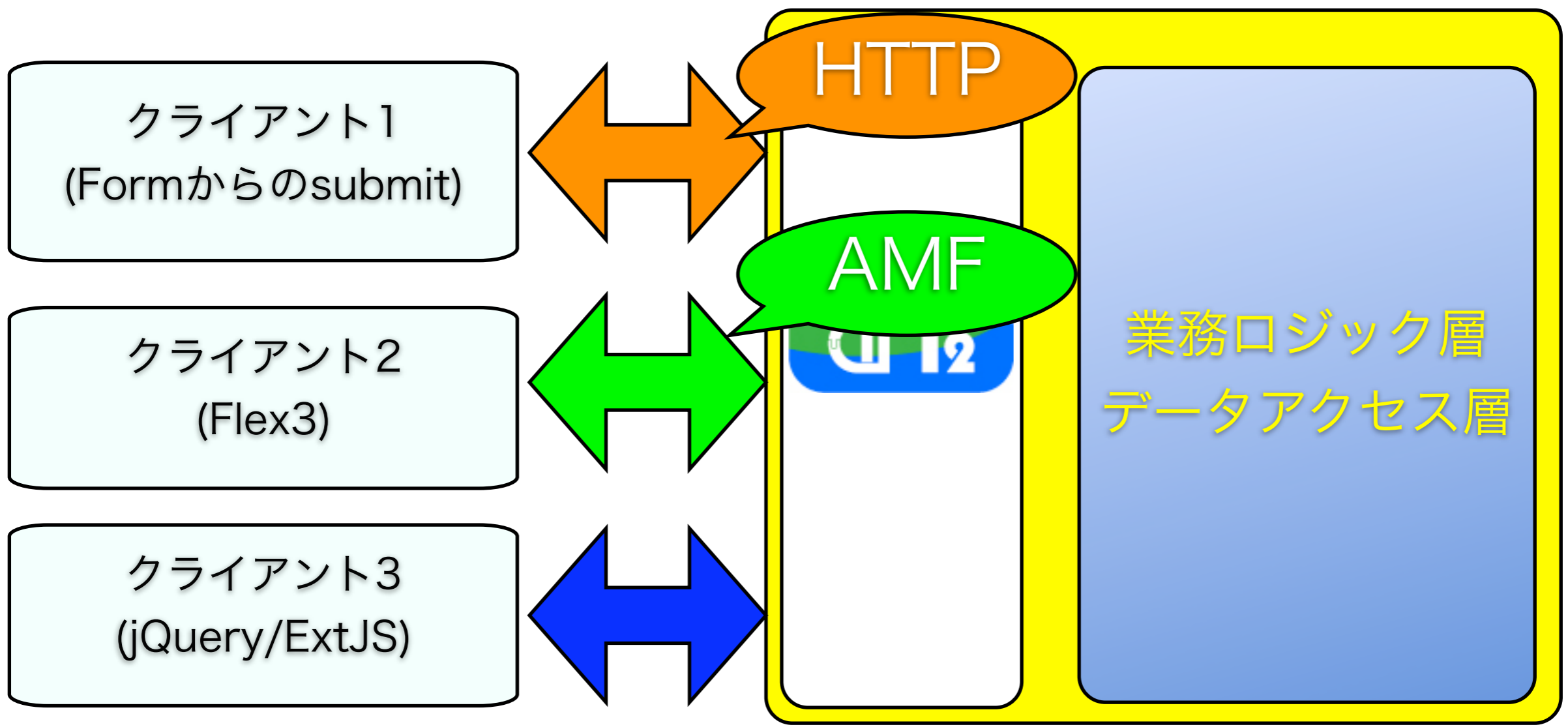
RIA時代のアーキテクチャ



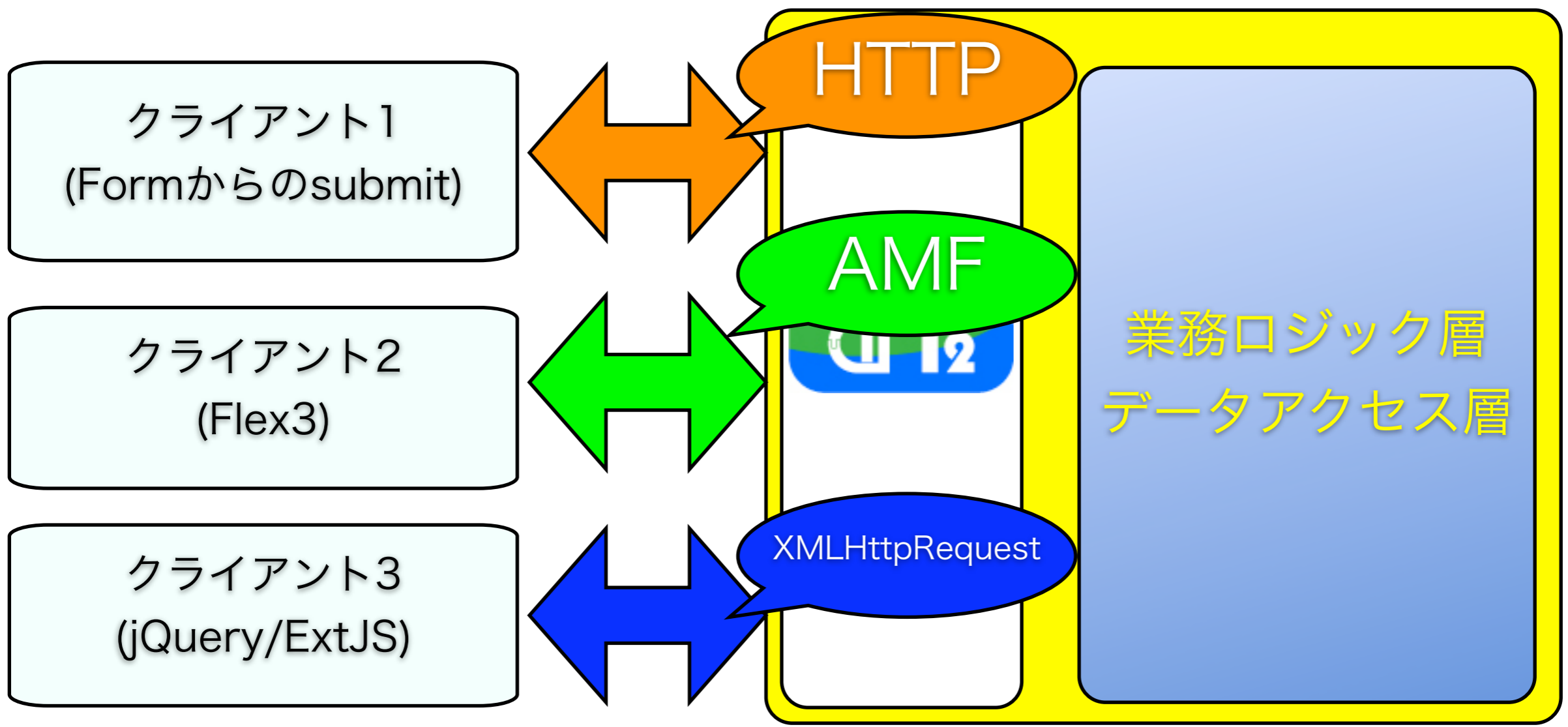
RIA時代のアーキテクチャ



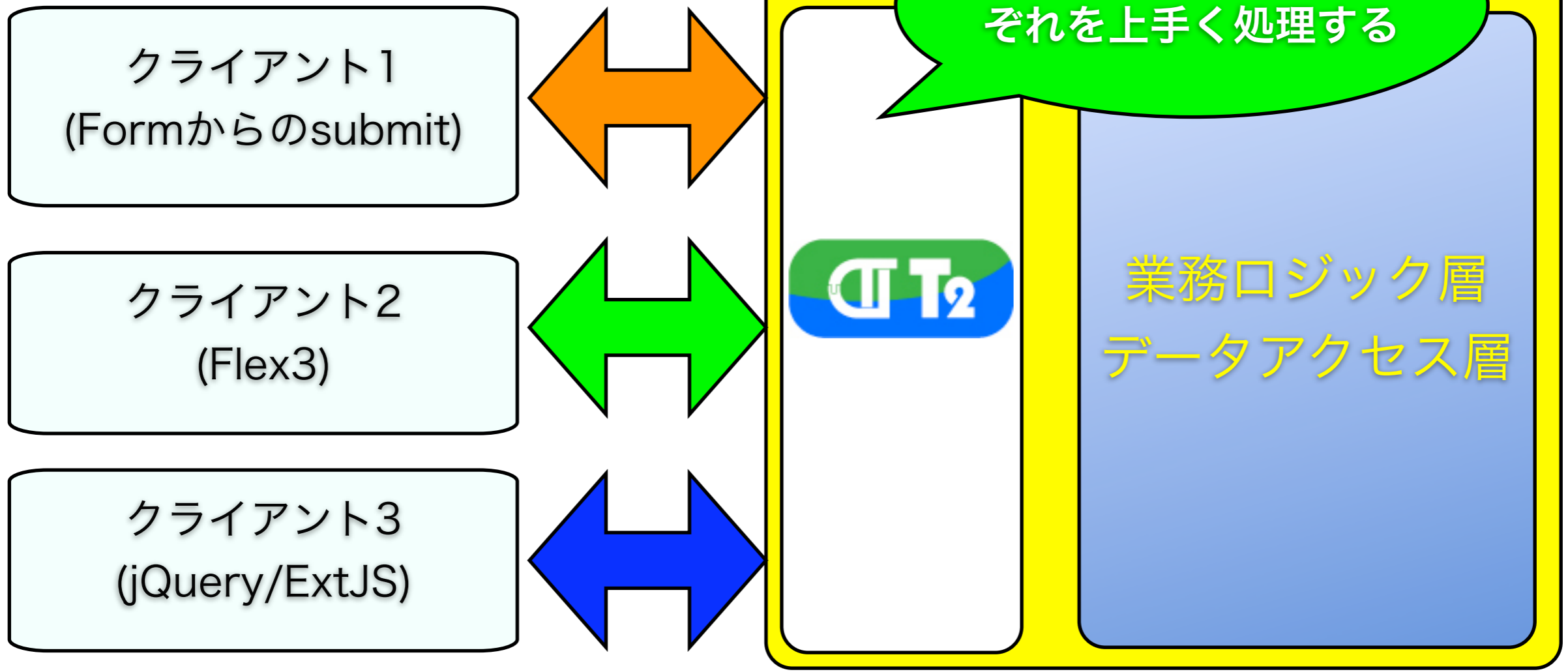
RIA時代のアーキテクチャ



RIA時代のアーキテクチャ



RIA時代のアーキテクチャ





T2とRIA時代のアーキテクチャ

- T2はRIA時代アーキテクチャをふまえたフレームワークをゴールにしている
- もちろん今のWebアプリケーションでも使えなくては行けない
- ただしそれだけでは駄目.この先求められる多種多様なリクエストをさばける能力が必要.



T2の基本モデル

- URLとPOJOをマッピングする
- POJOのメソッドにつけられたアノテーションから条件にマッチするメソッドを特定
- 引数の解決
- メソッドを呼び出し、実行
- 実行結果であるNavigationを受け取り、遷移先とその遷移方法を決定する



T2の基本モデルその1

URLとPOJOをマッピングする

```
http://yoursite.com/t2-samples/hello
```

```
public class Hello {
```

```
.....
```



T2の基本モデルその1

URLとPOJOをマッピングする

```
http://yoursite.com/t2-samples/hello
```

```
@Page("hello")  
public class Hello {  
  
    .....
```

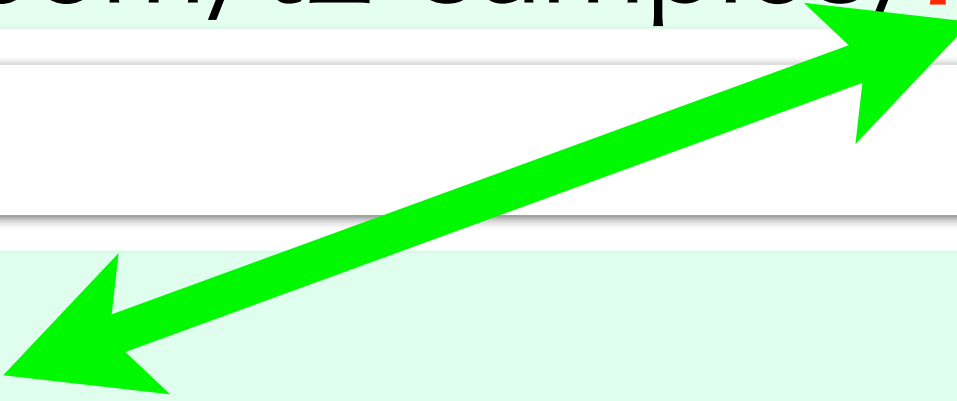



T2の基本モデルその1

URLとPOJOをマッピングする

```
http://yoursite.com/t2-samples/hello
```

```
@Page("hello")  
public class Hello {  
  
    .....  
}
```





T2の基本モデルその2-1

メソッドアノテーションでメソッドをマッピング

```
@Page("hello")
public class Hello {
    @GET
    @ActionPath
    public Navigation request(
        HttpServletRequest request) {
        .....
    }
}
```

T2の基本モデルその2-1

メソッドアノテーションでメソッドをマッピング

```
@Page("hello")
public class Hello {
    @GET
    @ActionPath
    public Navigation request(
        HttpServletRequest request) {
        .....
    }
}
```

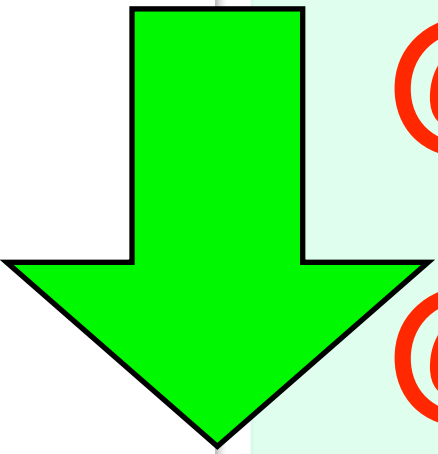
順番も重要！

@GET

@ActionPath

T2の基本モデルその2-2

メソッドアノテーションでメソッドをマッピング



@GET 受け付けるHTTPメソッド

@ActionPath より詳細な制約

メソッド部分

T2の基本モデルその2-3 ろうと (funnel) モデル

URLとPOJOをマッピング

使うHTTPメソッドは？

それ以外の制約は？

該当メソッドを実行

T2の基本モデルその2-3 ろうと (funnel) モデル

URLとPOJOをマッピング
使うHTTPメソッドは？
それ以外の制約は？

自然に決まる

該当メソッドを実行

T2の基本モデルその2-4

アノテーションの種類

- HTTPメソッドアノテーション
 - 文字通りHTTPのメソッドごと
 - @POST
 - @GET
 - @PUT
 - @DELETE

T2の基本モデルその2-5

アノテーションの種類

- **@ActionPath**

- リクエストされたURLに対して制約をあたえる

```
@Page("hello")
```

```
public class HelloPage {
```

```
    /**
```

```
     * http://yourdomain/t2-samples/hello/requestで呼ばれる.
```

```
    */
```

```
@GET
```

```
@ActionPath
```

```
public Navigation request(HttpServletRequest request) {
```




T2の基本モデルその2-5

アノテーションの種類

- **@RequestParam**

- submitされたリクエストパラメータに対して制約をあたえる

```
@RequestScope
```

```
@Page("add")
```

```
public class AddPage {
```

```
    @POST
```

```
    @RequestParam
```

```
    public Navigation add(WebContext context) {
```

T2の基本モデルその2-6

アノテーションの種類

- **@RequestParam** つづき

- つまり、サブミットされたボタンごとにメソッドを実行可能

```
<input type="submit" name="add"
      value="同一ページ" />
```

@RequestParam

```
public Navigation add(WebContext context) {
```

T2の基本モデルその2-7

アノテーションの種類

- **@Default**

- Pageクラス内のどのメソッドにもマッチしない場合動く

@Default

```
public Navigation index() {  
    return Forward.to("/jsp/download.jsp")  
}
```

T2の基本モデルその2-8

アノテーションの種類

- **@Ajax**

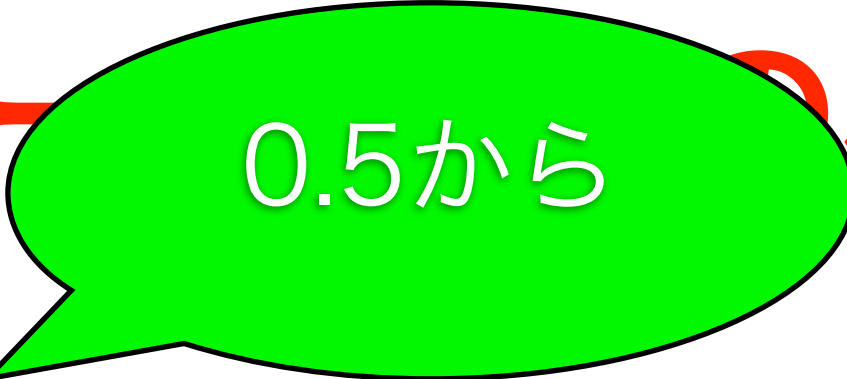
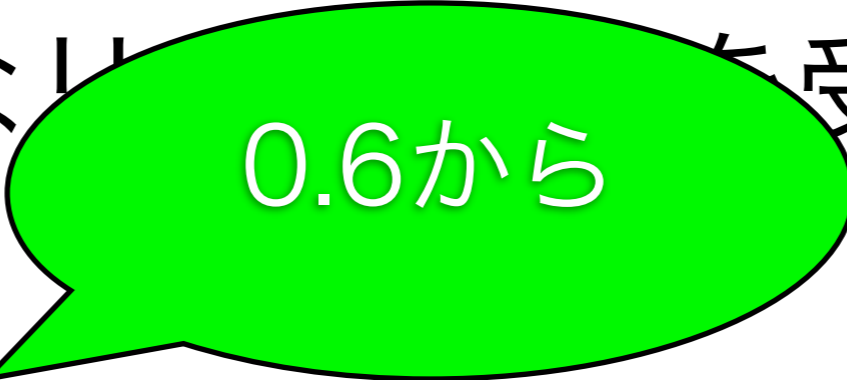
- Ajaxなリクエストを受け取る

- **@Amf**

- Flex/AIRからのAMFなリクエストを受け取る

T2の基本モデルその2-8

アノテーションの種類

- @Ajax 
- Ajaxなりきりを受け取る 
- @Amf
- Flex/AIRからのAMFなりきリクエストを受け取る



T2の基本モデルその3-1

引数の解決1

- 暗黙的にインジェクト可能なもの
 - HttpServletRequest/HttpServletResponse
 - HttpSession
 - ServletContext
 - WebContext(Webであつかう情報の集まり. T2独自)
 - Cookie

T2の基本モデルその3-2

引数の解決2

- 引数アノテーションで情報を補ってインジェクト可能なもの
 - **@RequestParam** : リクエストパラメータを受け取る
 - **@SessionAttr** : Sessionの値を受け取る
 - **@Upload** : Uploadされたファイルを受け取る

T2の基本モデルその3-2

引数の解決2

- 引数アノテーションで情報を補ってインジェクタ可能なもの
 - **@RequestParam** :
クォータを受け取る
 - **@SessionAttr** : Ses
 - **@Upload** : Uploadクォ
取る

0.4から



T2の基本モデルその3-2

引数の解決3

- 引数アノテーションで情報を補ってインジェクト可能なもの
 - **@Var** : RESTなURLの一部を受けとる
 - **@Index** : Foreachの行番号を受け取る

T2の基本モデルその3-2

引数の解決3

- 引数アノテーションで情報を補ってインジェクタ可能なもの

- **@Var** : REST
- **@Index** : For

0.4から

T2の基本モデルその4

メソッドの戻り値

- T2で呼び出すメソッドの戻り値はNavigationインタフェースの実装クラス
 - Forward : フォワードする
 - Redirect : リダイレクトする
 - Direct : レスポンスに直接出力
 - JSON : JSON形式で出力



T2の基本モデル

@RequestScope

@Page("hello")

```
public class HelloPage {
```

```
    /**
```

```
     * http://yourdomain/t2-samples/hello/requestで呼ばれる.
```

```
     */
```

@GET

@ActionPath

```
public Navigation request(HttpServletRequest request) {
```

```
    System.out.println("request.getContextPath() : "
```

```
        + request.getContextPath());
```

```
    request.setAttribute("greet", helloService.hello()
```

```
        + " from request().");
```

```
    return Forward.to("/jsp/hello.jsp");
```

```
}
```



T2のデモ1



with



いまの最新版のサンプルです



T2のデモ2



with



もう少し後のバージョンの
サンプルです



T2のデモ3



with 

RESTでのSilverlightとの
連携サンプルです



T2のサブプロジェクトと 依存関係1

- Commons : 共通ユーティリティ/メタクラス
- Lucy : シンプルなDIコンテナ
- プラグイン
 - S2Adapter : S2との連携機能
 - SpringAdapter : Springとの連携機能
 - GuiceAdapter : Guiceとの連携機能



T2のサブプロジェクトと 依存関係2

- T2自体はCommonsにのみ依存
- T2はDIコンテナ非依存型なので、お好きなのをどうぞ
- もちろん別にDIコンテナ使わなくてもかまいません。

ちょっとだけLucyの紹介1

- 小さなIoCコンテナ
- Configuration by Annotationが基本概念
- 型によるインジェクションが好きです♪
- JavaBeansにあわせるとか面倒です



ちょっとだけLucyの紹介2

```
import commons.annotation.composite.PrototypeScope;  
import lucy.annotation.core.Inject;
```

@PrototypeScope

```
public class Client {
```

```
    protected Greeting greeting;
```

```
    public String execute() {  
        return "hello " + greeting.greet();  
    }  
}
```

@Inject//セッターじゃなくてもメソッドならインジェクト可能

```
    public void setGreeting(Greeting greeting) {  
        this.greeting = greeting;  
    }  
}
```

```
}
```

T2の開発体制とポリシー

- T2の開発ではポリシーに沿っているかが最優先
 - 立ち戻るべき原点。
 - 人間の意見は状況によってかわる。
- フレームワークが輝くのではなく、その上のアプリケーションが輝かなくては意味が無い。
- 最初にあげた、やること・やらないことをはっきりとしよう！
- 楽しく、長くやる！！

T2の開発体制とポリシー

- 開発体制テーマは「**緩やかな進化**」
 - バグ修正は迅速に。
 - 機能追加は緩やかに。
 - リリースの前にサンプルアプリを。
- 後方互換性の維持
- 継続性の確保



T2の機能提供について

- フレームワークのジレンマ
 - 機能追加による肥大化
 - 後方互換性の維持の難易度があがる
- T2では
 - 提供する機能は削りに削る
 - 二度目はうまく行く戦略
 - **厳選されたミニマルセットの提供**



T2の機能提供について

- フレームワークのジレンマ

<http://d.hatena.ne.jp/shot6/20080627#1214519810>

- 後方互換性の維持の難易度があがる
- T2では
 - 提供する機能は削りに削る
 - 二度目はうまく行く戦略
 - **厳選されたミニマルセットの提供**



T2の機能提供について

- フレームワークのジレンマ

<http://d.hatena.ne.jp/shot6/20080627#1214519810>

<http://d.hatena.ne.jp/shot6/20080630#1214800235>

- T2では
 - 提供する機能は削りに削る
 - 二度目はうまく行く戦略
 - **厳選されたミニマルセット**の提供



T2のリリースプラン

- 現在の最新版は0.3
- 0.4 : 通常のWebアプリの機能強化
- 0.5 : REST/Ajax機能の強化
- 0.6 : Flex3/AIRとの連携強化
- 0.7 : Silverlight2との連携強化
- 0.8 : RSS/AtomPub関連予定



T2プロジェクト参考

URL

- プロジェクトサイト
 - <http://t-2.googlecode.com/>
- ドキュメント
 - http://t-2.googlecode.com/files/T2_UserGuide_Japanese_0.3.pdf
- メーリングリスト
 - <http://groups.google.com/group/t2-users>



T2プロジェクト参考

URL

- コミッタ
 - shot6 : <http://d.hatena.ne.jp/shot6>
 - yone098 : <http://d.hatena.ne.jp/yone098>
 - skirnir : <http://d.hatena.ne.jp/skirnir>
 - c9katayama : <http://d.hatena.ne.jp/c9katayama>
- はてなキーワード
 - T2Frameworkで拾ってます

ご清聴ありがとうございました

- the WEB connector -