T2 in Action

～サンプルから学ぶT2～

T2 Project

yone098, skirnir

自己紹介

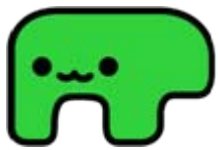
- 名前
  - 米林　正明
- ID
  - yone098
- 所属
  - T2 Project http://t2framework.org/
  - 株式会社Abby 代表取締役社長

自己紹介

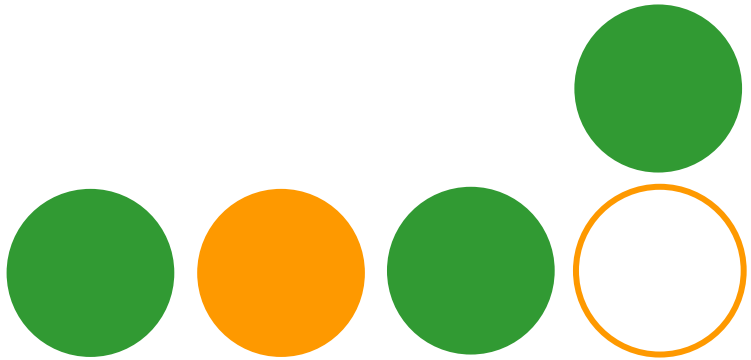
- 名前
  - 横田　健彦
- ID
  - skirnir
- 所属
  - T2 Project http://t2framework.org/
    - Ymir Projectもやってます
  - 株式会社アークシステム

Agenda

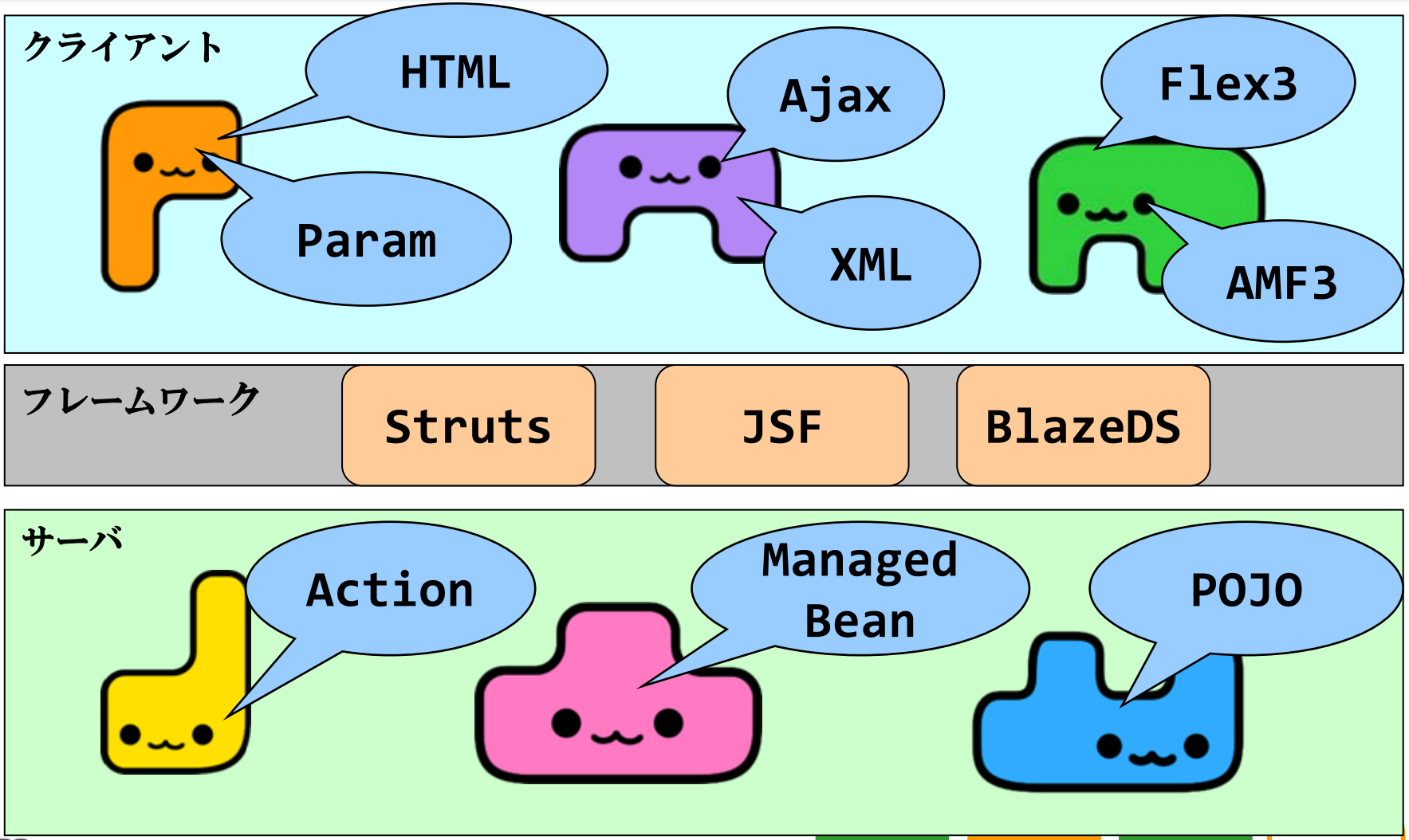
- T2の概要
  - T2の仕組み
  - T2のスタイル
- Viliで始める簡単T2
  - demo
- 新機能AMF
  - demo



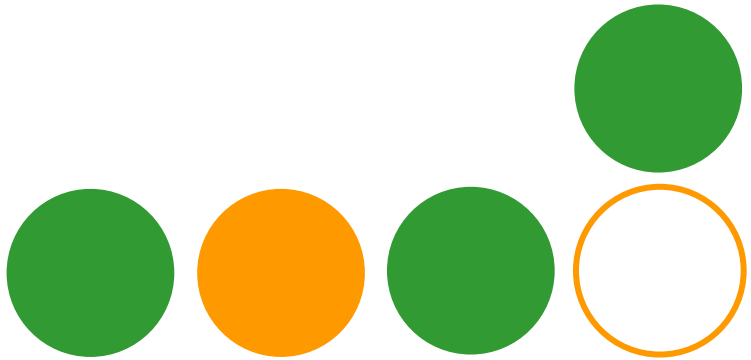
T2の概要



Webにおける環境の進化



Webにおける環境の進化



T2のモチベーション

色々なリクエストをまとめて上手に捌かなあかんやろ。

T2 ～The WEB Connector～

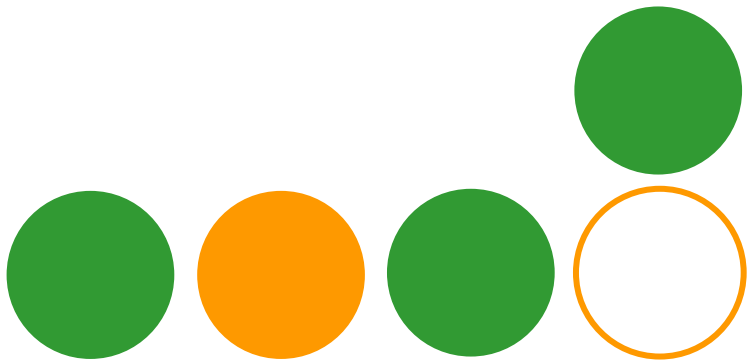
- T2とは
  - シンプルなWebフレームワーク
    - フィルタ指向
    - アノテーションベース
    - 色々なリクエストをさっくり上手に捌く
  - テーマ「つなぐ・つながる」
    - ユーザと開発者を
    - 実案件と楽しさを
    - あらゆるクライアントとサーバアプリを
  - クライアントとユーザーコードの仲介役
    - 仲介役に徹するポリシー
    - ひとつのことを上手にやる

T2 ～The WEB Connector～

- T2とは
  - クライアント側は自由に選択可能
    - フォームのサブミット
    - Ajaxリクエスト/REST
    - Flex/AIR/Silverlightなどのリッチクライアント
    - Webサービス
  - 統一したプログラミングモデルの提供
    - クライアントがどうであっても、サーバ側は常に同じように作れる。分業重要。
    - 一貫したスキルセットで開発可能

T2 ～The WEB Connector～

- つなぐ
- つながる



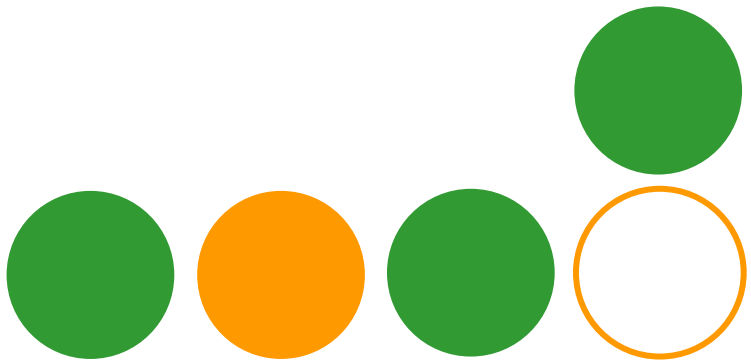
T2 ～The WEB Connector～

クライアント

T2

サーバ

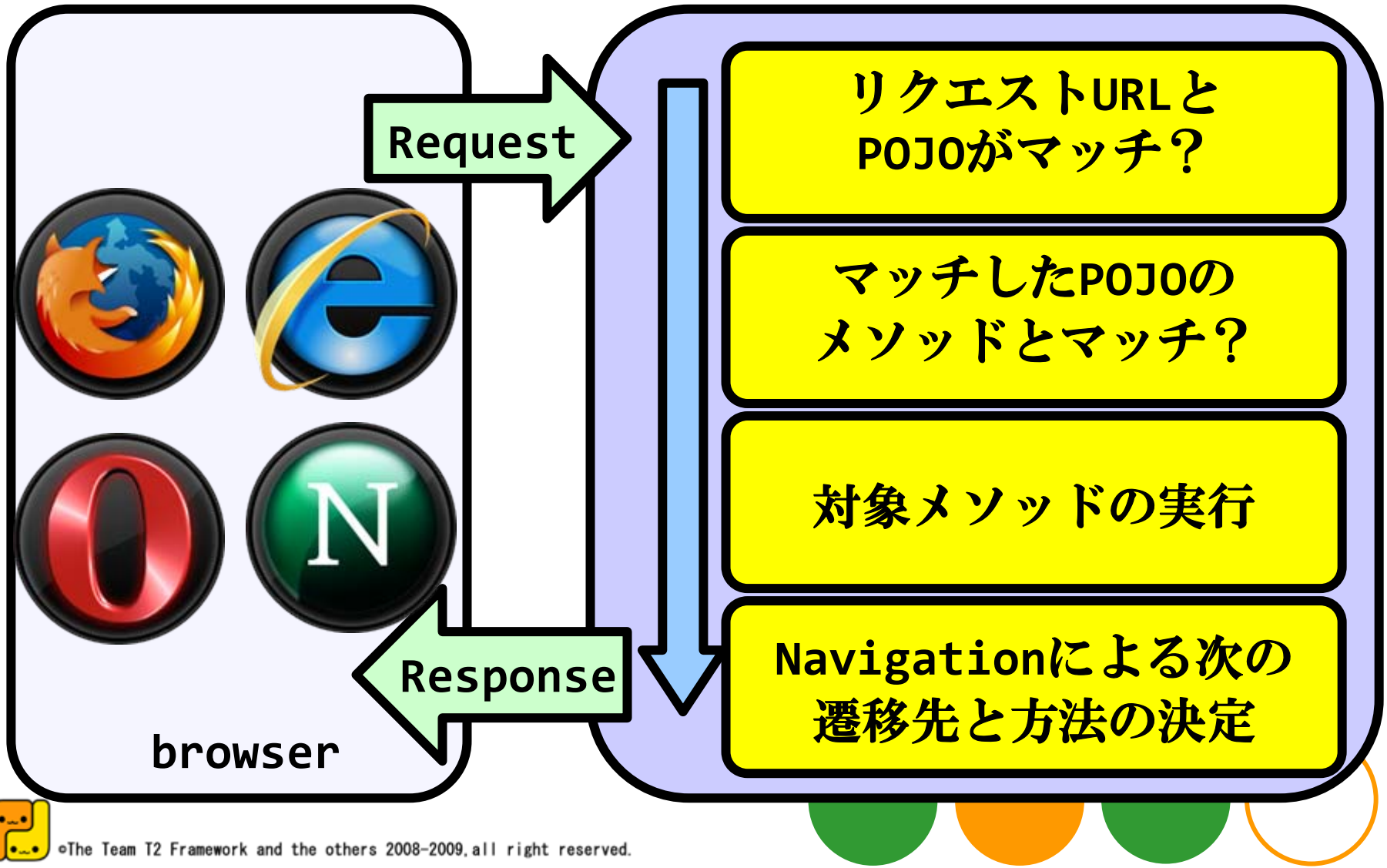
●T2の仕組み



T2 ～基本的な仕組み～

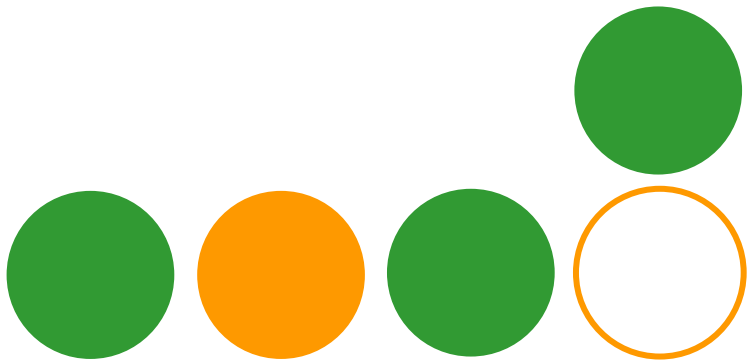
- フィルタベースのWebフレームワーク
  - URLとJavaオブジェクト(POJO)をマッチング
    - アノテーションでマッチング
  - マッチしたPOJOから適切なメソッドを選択
  - メソッドを実行
    - 行き先と遷移方法を指定するNavigationインタフェースの実装を返す
  - T2がそれに従って遷移する
    - テンプレートエンジンは任意で指定
    - JSP，ZPT，Mayaa,HTML/XHTMLなんでも可能

T2 ～基本的な仕組み～



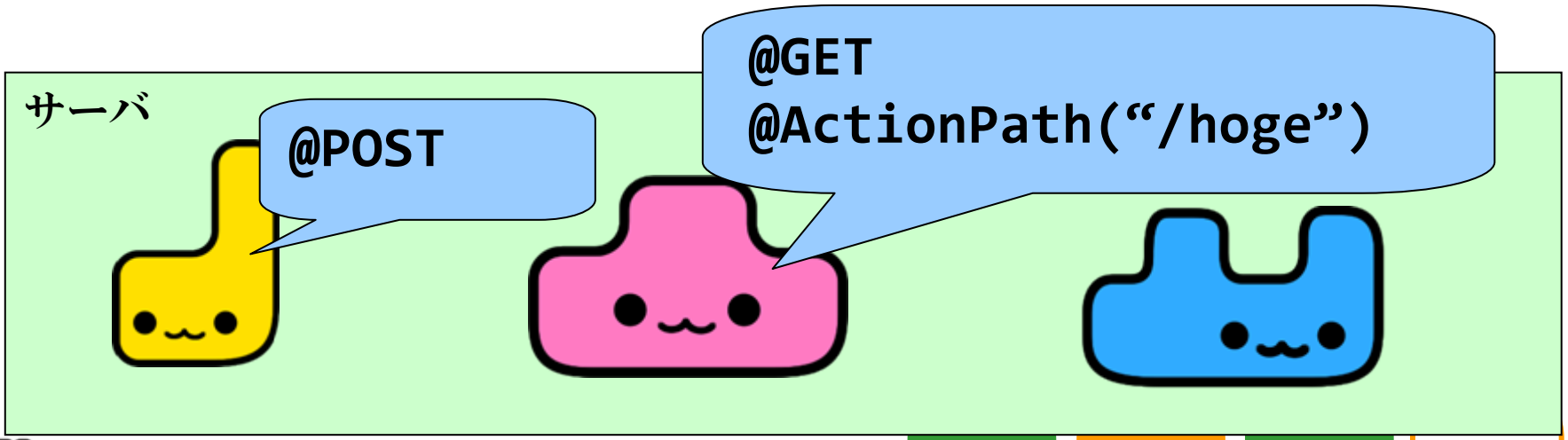
T2 -The WEB Connector-

●T2のスタイル



T2のスタイル

- アノテーションドリブン
  - リクエストを受けるPOJOにアノテーション付与
  - 受け口の分かりやすさ重視
  - アノテーションを適切に利用。規約は最小限。

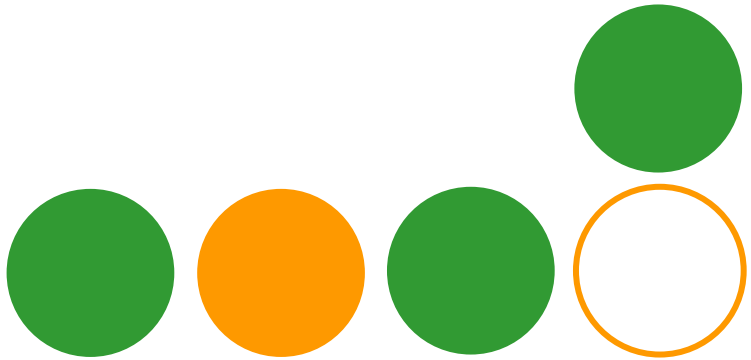


T2のスタイル

```
@Page(“/add”)  // /addというURLでこのPageが動く
public class AddPage {

    @GET //HTTPのGETメソッドのとき、メソッド実行
    public Navigation add(WebContext context) {
        return Forward.to("/pages/add.jsp");
    }

    @Amf  //FlexやAIRからのリクエストのとき、メソッド実行
    public Navigation handleAmf (HogeDto dto) {
        return AmfResponse.to(“ok”);
    }

}
```

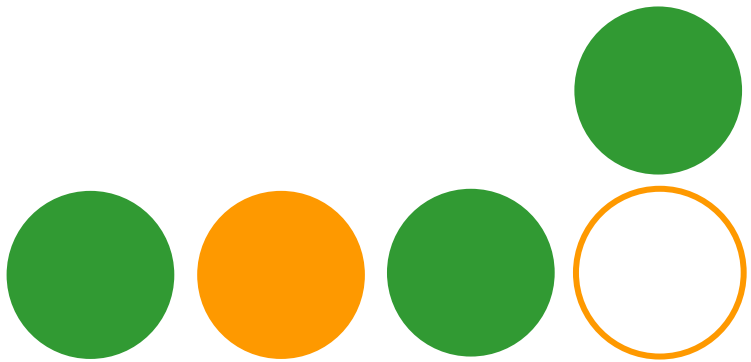
T2のスタイル

- T2のメソッドアノテーション



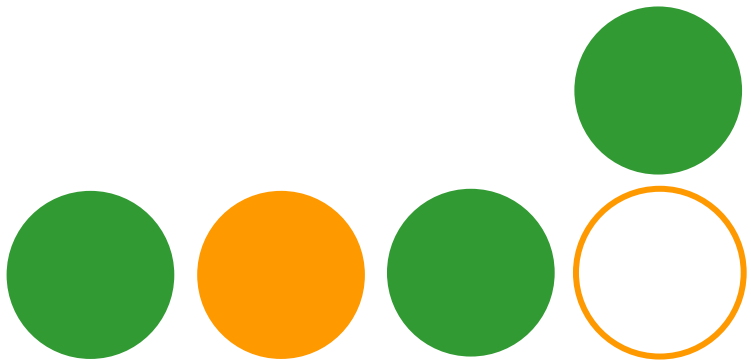
T2のメソッドアノテーション

- @GET,@POST
- @ActionPath
- @ActionParam
  - リクエストパラメータのkey
  - key-value指定も可能
- @Ajax
- @Amf
- @Default



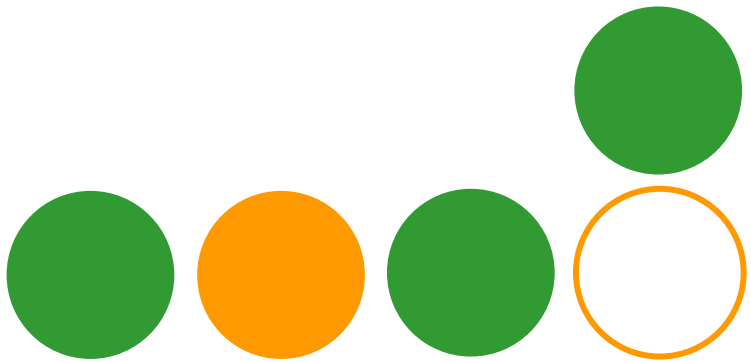
T2のスタイル

- T2のメソッド引数特定方法



T2のスタイル

- 引数のアノテーションを見て、適切なコンポーネントをインジェクト
  - @RequestParam
  - @RequestHeader
  - @SessionAttr
  - @Upload
  - @Form
  - @Index
  - @Var

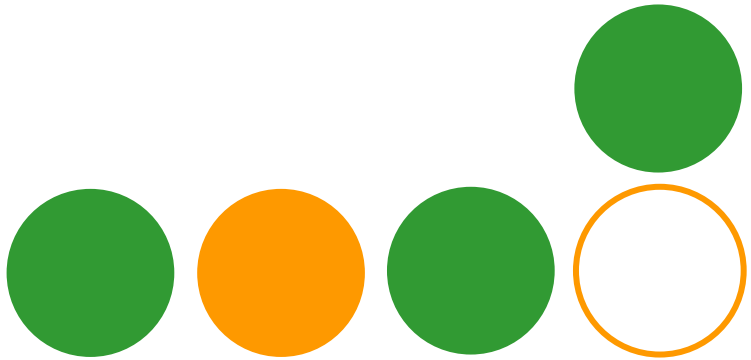


T2のスタイル

- 引数の型を見て、インジェクト
  - HttpServletRequest, HttpServletResponse
  - HttpSession
  - ServletContext
  - Cookie/Cookie[]
  - WebContext
  - Request, Response
  - UploadFile
  - ErrorInfo

T2のスタイル

- PHPとつながる

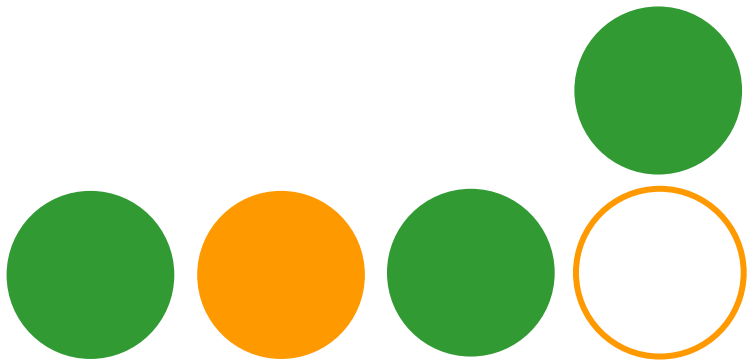


T2のスタイル

- PHPライクなアクセス
  - $_GET(“arg1”)
  - $_POST(“arg2”)
  - $_SESSION(“name”)
  - $_FILES(“sample”)

T2のスタイル

- T2のポリシー



T2のポリシー

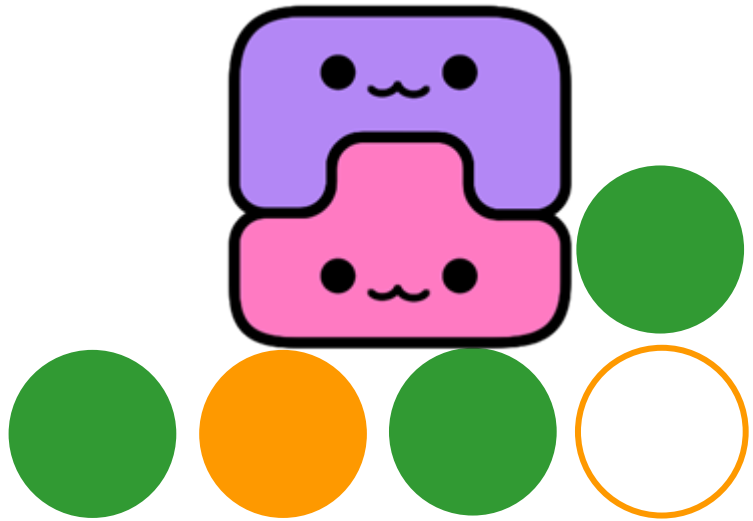
- 基本はステートレス
  - 非常に薄いフレームワークを維持する
  - プラグイン機能があるのでステートフルにも可能
  - 拡張部分で一部置き換えも可能

T2のポリシー

- 特定のDIコンテナに依存しない
  - Seasar2、Spring、Guiceで利用可能
  - 自前のシンプルDIコンテナLucyでも可能
  - アダプタ部分を作成することで他にも対応可能

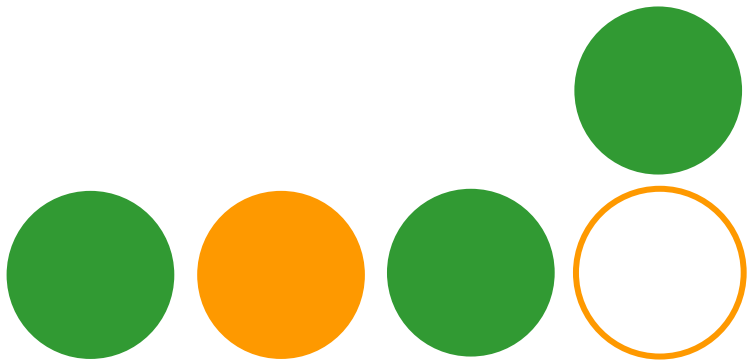
T2のまとめ

- ユーザーが比較的簡単に拡張可能
  - コアはシンプルに。
  - エクステンションで便利機能を提供
  - プロジェクトごとに必要な機能を作るのも簡単
- ユーザ自身が**制御可能だと自信をもてる事**がとても重要。
  - そのため徹底的にシンプルにする
  - やりすぎない
  - キレイなコード
  - ドキュメントで内部構造を補足



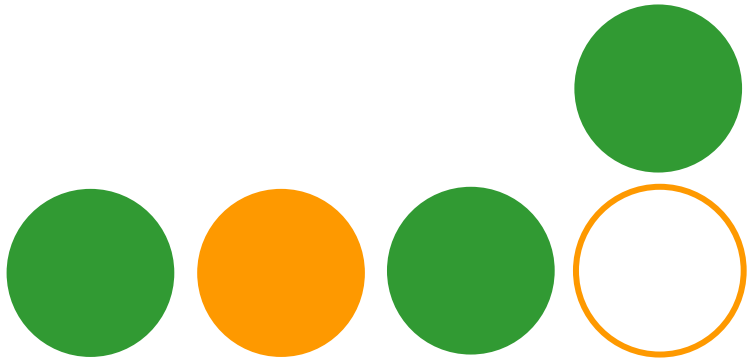
T2のまとめ

- T2のまとめ

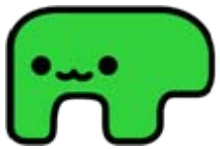


T2のまとめ

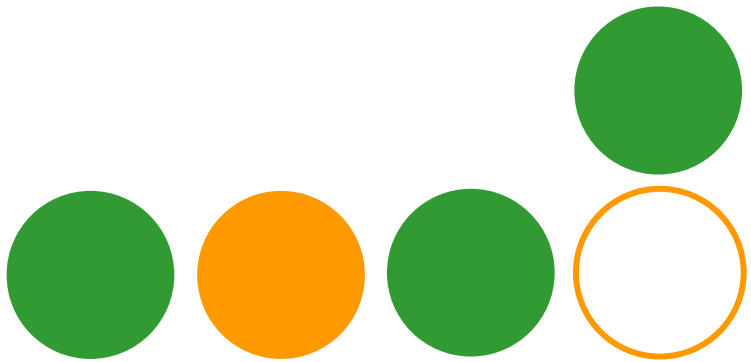
- つなぐ
- つながる



T2 -The WEB Connector-



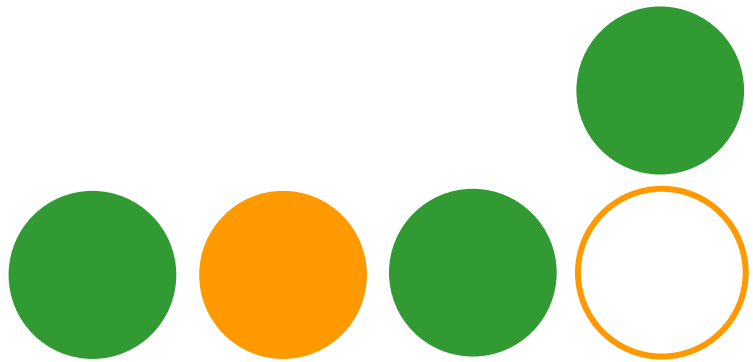
Viliで始める かんたんT2



T2 ～The WEB Connector～

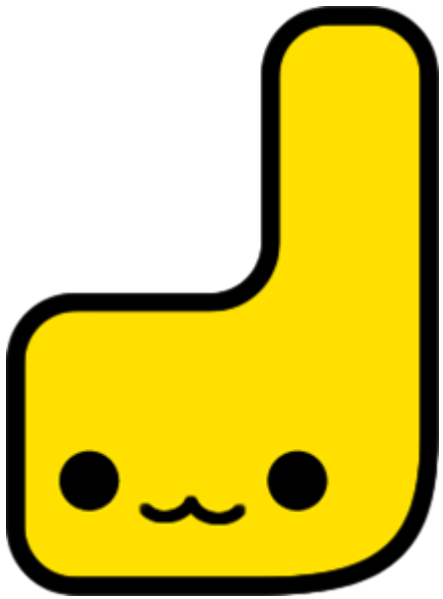
- T2でアプリケーションを作ろう！
  - EclipseでT2アプリケーションプロジェクト作成

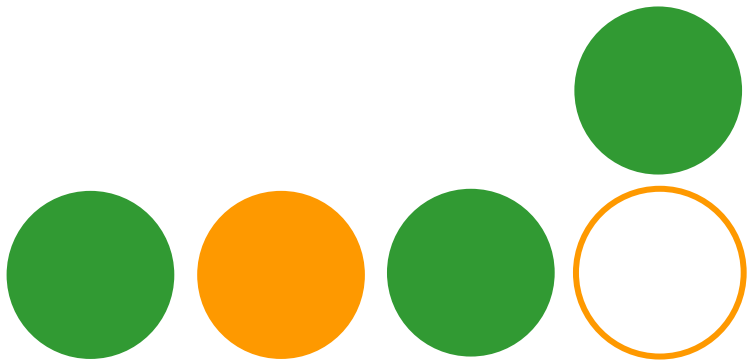
T2 ～The WEB Connector～



T2 ～The WEB Connector～

Vili（ヴィリ）を
使えば簡単だよ





T2 ～The WEB Connector～

Ymir

プロジェクト作成支援Eclipseプラグイン Vili（ヴィリ）はSeasarプロジェクトで開発されているWebフレームワーク「Ymir（ユミル）」の開発支援プラグインからプロジェクト作成支援機能を分離したものでT2プロジェクトで開発されているんだよ。

T2 ～The WEB Connector～

Viliではプロジェクトの雛形（スケルトン）をMavenのアーティファクトとして管理しているので、プロジェクトの雛形をMavenリポジトリ上に公開することで、Viliで作成できるプロジェクトを誰でも簡単に増やすことができるんだ。

また、雛形を展開する時にリソースに値を埋め込んだり、任意の処理を実行することもできたりと、雛形の表現力が高いのも大きな特徴だよ。

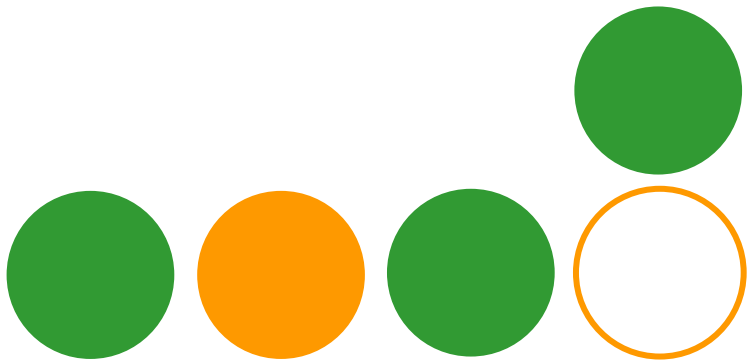
T2 ～The WEB Connector～

もう一つViliの大きな特徴として、プロジェクトに追加するためのプログラム部品（フラグメント）を定義することができるというものがあるんだ。

例えばT2でAMF連携を行なうための「T2 AMFフラグメント」を定義しておいて、Viliで作成したT2アプリケーション開発プロジェクトにAMF連携機能を後から追加する、みたいなことが簡単にできるんだよ。

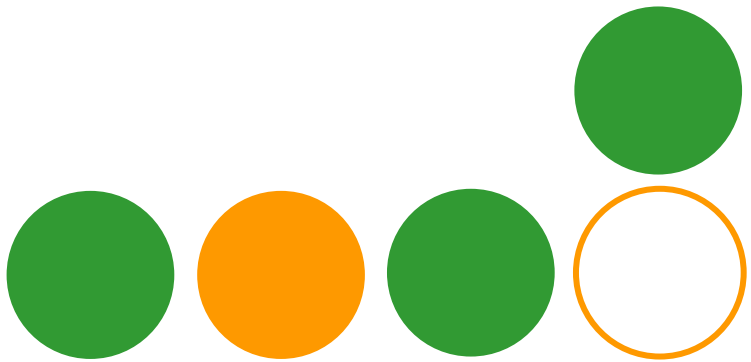
Ymirについては既に、JSON連携フラグメント、AMF連携フラグメント、UIテスト支援機能（Yonex）フラグメントのように、様々なフラグメントが公開されているんだけど、T2プロジェクトでもT2用のアプリケーションプロジェクトのために、順次便利なフラグメントを公開していきたいと思っているんだよ。

T2 ～The WEB Connector～

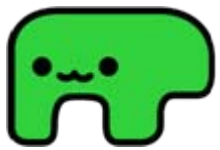


T2 -The WEB Connector-

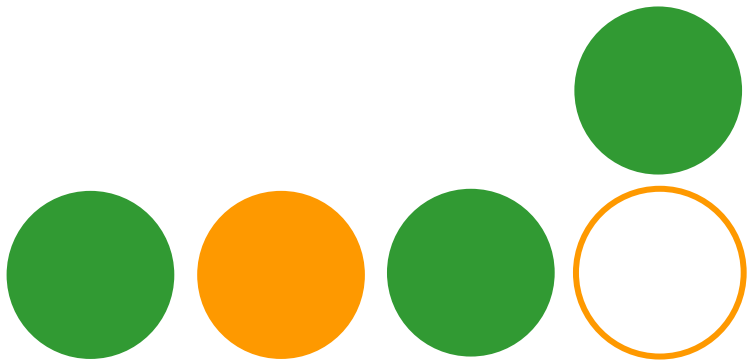
- というわけでデモ



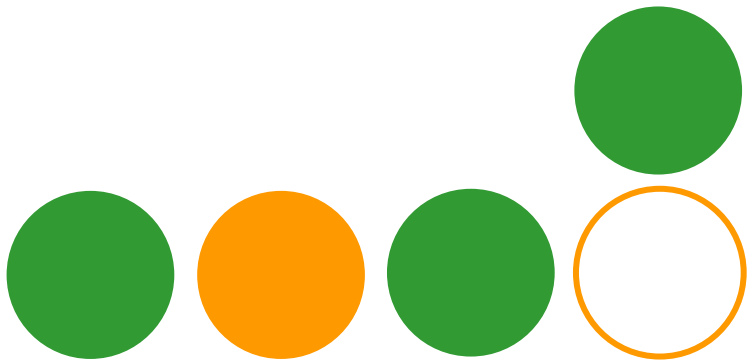
T2 ～The WEB Connector～



新機能AMF

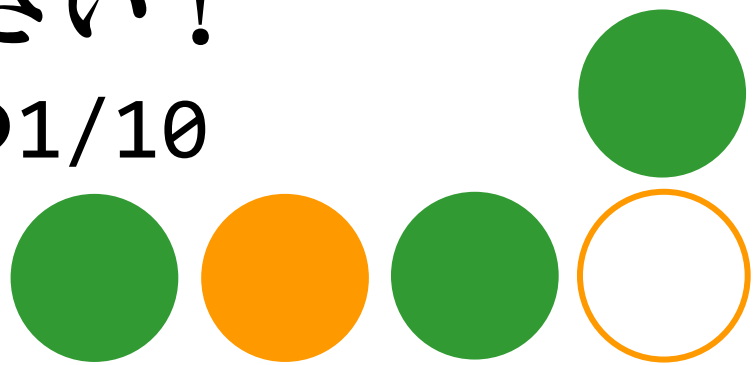


●そもそもAMFとは？



T2 ～AMF～

- AMF
  - ActionMessageFormat
  - Flash/Flexのオブジェクトシリアライズ方法
  - 通信フォーマットに利用可能
  - バイナリ形式で小さい！
    - JSONの1/4、XMLの1/10



T2 ～新機能AMF～

●例えば整数

```
(数値hex)                    :(binary)
0x00000000 - 0x0000007F : 0xxxxxxx
0x00000080 - 0x00003FFF : 1xxxxxxx 0xxxxxxx
0x00004000 - 0x001FFFFF : 1xxxxxxx 1xxxxxxx 0xxxxxxx
0x00200000 - 0x3FFFFFFF : 1xxxxxxx 1xxxxxxx 1xxxxxxx xxxxxxxx
0x40000000 - 0xFFFFFFFF : レンジ例外が発生
```

0-127までの値なら、1byteに変換！

●例えば文字列

文字列の参照テーブルがあるので、同じ文字列なら参照で済ます

→サイズを小さくする仕掛けがされてます

T2 ～AMF～

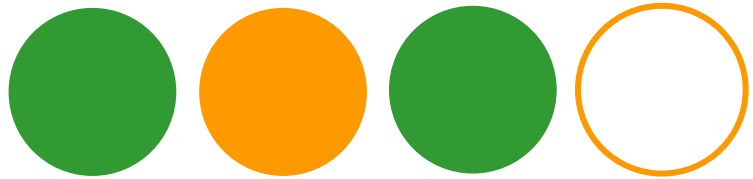
- AMFが利用できる型の一覧

undefined Type	null Type
false Type	true Type
integer Type	double Type
String Type	XMLDocument Type
Date Type	Array Type
XML Type	ByteArray Type

Object Type(型ありObject,形無しオブジェク)

T2 ～AMF～

- AMF0とAMF3
  - AMF3:ActionScript3から利用可能（FlashPlayer9以降）
  - Flex2/3ではAMF3を利用
- format仕様はOpen
  - http://download.macromedia.com/pub/labs/amf/amf3_spec_121207.pdf

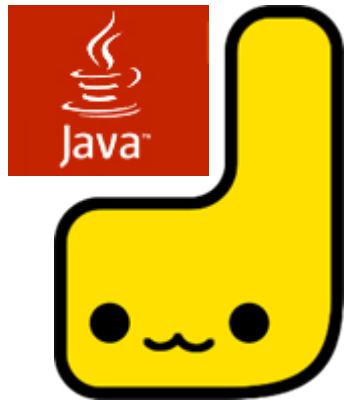


T2 ～AMF～

●T2での利用イメージ

クライアント

サーバ



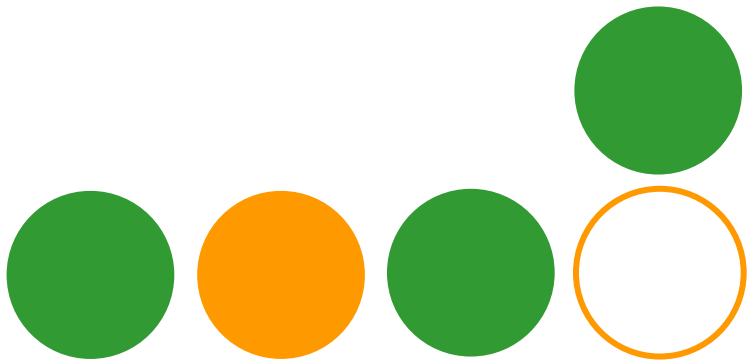
ASオブジェクト

HTTP/HTTPS
で転送

AMF3でシリアライズ

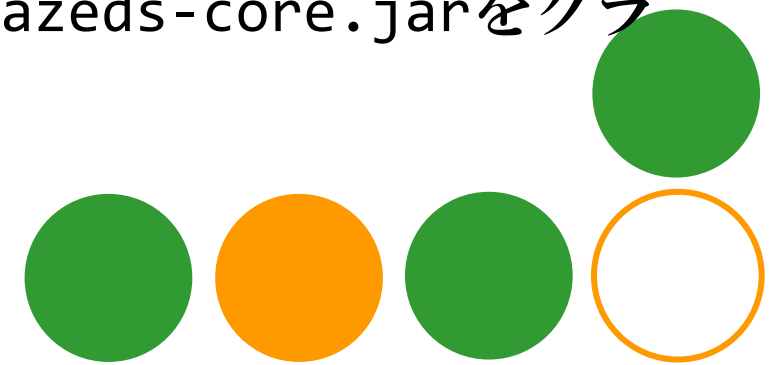
AMF3でJavaオブジェクトに
デシリアライズ

●T2での実装



T2 ～AMF～

- AMFのシリアライザ
  - 以下のAMF実装が利用可能
  - S2Flex2をベースにしたAMF実装
    - すばらしいAMF実装（コミッターの方に感謝！）
    - T2に移植
      - 依存ライブラリなしでAMFの利用が可能
  - BlazeDSのAMF実装
    - blazeds-common.jarとblazeds-core.jarをクラスパスに含める。



T2 ～AMF～

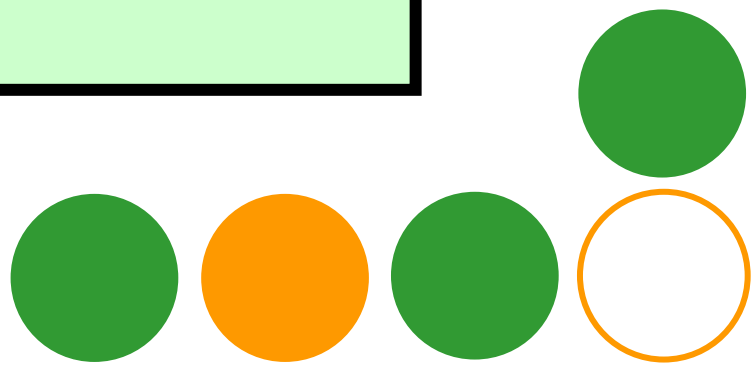
AmfContext

BLAZEDS

BlazeDSAmfContext
-serialize
-deserialize

T2InternalAmfContext
-serialize
-deserialize

blazeds-common.jar
blazeds-core.jar

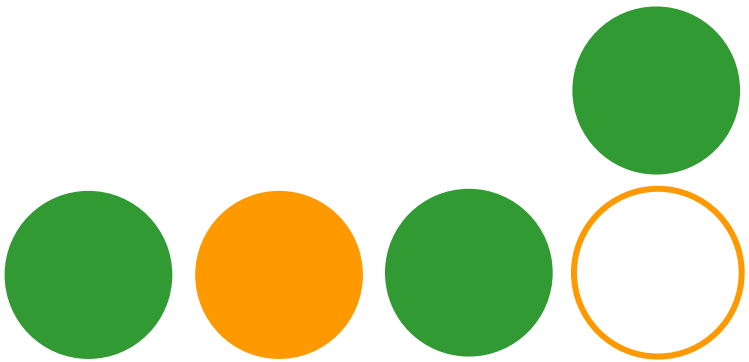


T2 ～AMF～

- なんでAMF実装が2個あるの？

	T2Internal	BlazeDS
メリット	依存jar不要 型変換が柔軟	Adobe謹製
デメリット	not Adobe謹製	依存jarが必要 型変換の変更がしにくい

●使い方



T2 ～AMF～

- 使い方
  - まずは「destination」を決めよう
    - destination=呼び出すPageクラスを示すあて先
  - つぎに呼び出すメソッドを決めよう
    - 引数と戻り値も決めよう

あて先は「unicef」
メソッドは「bokin」
引数はint
戻り値はString
でいこう！

T2 ～AMF～

- 決めたdistinationとメソッドでサーバ側を作成

```
@Page(“unicef”)  //destination
public class UnicefPage {

    @Amf  //メソッドにつける @Amf(“bokin”)でも可
    public Navigation bokin(int bokingaku){

        Unicef.sendToAfrica(bokingaku);

        //AmfResponseに戻り値を入れる
        return AmfResponse.to(“Thank you!”);
    }
}
```

T2 ～AMF～

- クライアント側
  - まずは結果を受け取るハンドラを作成

```
//通信成功時に呼ばれる
public function handleResult(e:ResultEvent,
                             token:Object=null):void{
    var message:String = String(e.result);
    Alert.show(message);//”Thank you”と表示
}
//通信失敗時に呼ばれる
public function handleFault(e:FaultEvent,token:Object=null):void{
    log.error(ObjectUtil.toString(e));
}
```

T2 ～AMF～

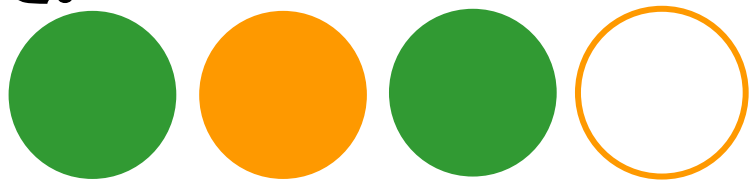
- 最後に呼び出し部分

```
//呼び出し用RemoteObject作成
var unicefPage:RemoteObject = new RemoteObject("unicef");
//接続設定（お決まり）
var endPoint:String =
        URLUtil.getFullURL(Application.application.url,“t2.amf");
var channel:Channel = URLUtil.isHttpsURL(endPoint)?
      new SecureAMFChannel(null,endPoint):
      new AMFChannel(null,endPoint);
var channelSet:ChannelSet = new ChannelSet();
channelSet.addChannel(channel);
unicefPage.channelSet = channelSet;
//サーバ呼び出し
var token:AsyncToken = unicefPage.bokin(1000000);
token.addResponder(
         new AsyncResponder(handleResult,handleFault));
```

- もし送受信にデータクラスを使う場合は

```
//aliasで、このデータクラスのデータを入れるJava側のクラスを指定
[RemoteClass(alias=“unicef.java.UnicefDto")]
public class UniceDto
{
  public var bokingaku:int;
  public var message:String;
}
```

- Java側に、クライアントのデータクラスと同じ構造のクラスを用意し、メタデータで指定。



T2 ～AMF～

- まとめると

RemoteObject作成
unicefPage.bokin(10000)

destination:unicef
メソッド：bokin
引数：1

@Page(“unicef”)
UnicefPage
@Amf
String bokin(int money)

呼び出し成功：
ResultEvent
result:”Thank you!”

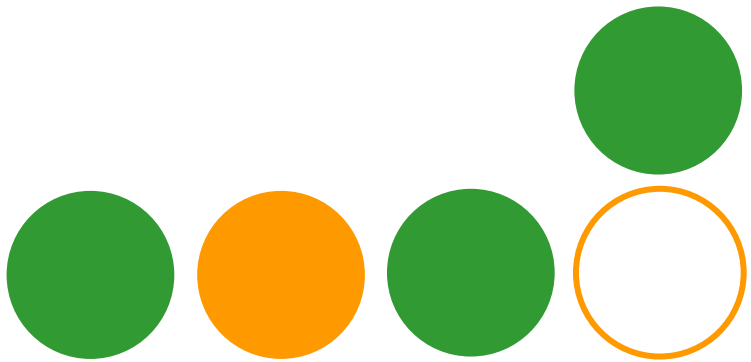
handleResult
メソッド

呼び出し失敗：
FaultEvent
message:エラー内容

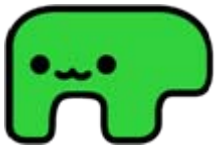
handleFault
メソッド

T2 ～AMF～

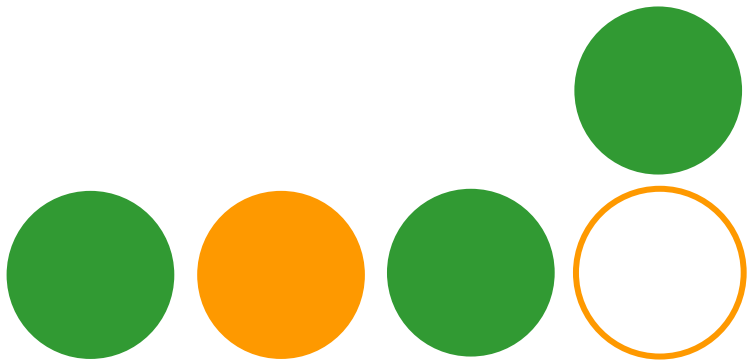
- demo



T2 ～まとめ～



まとめ



T2ロードマップ

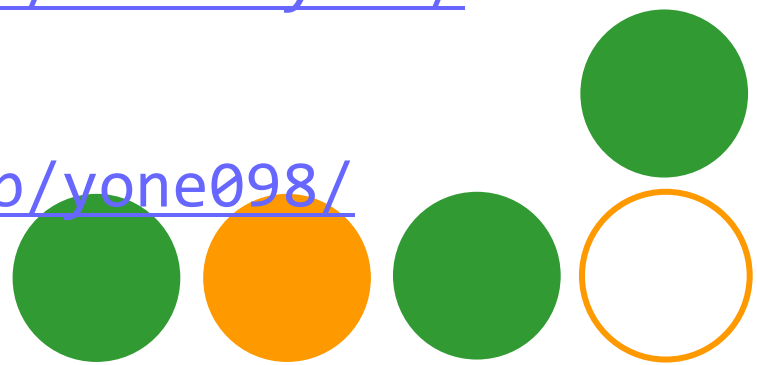
- T2は0.6-gaを開発中
  - リリース間に合わずすいませんm(_ _)m
  - AMF機能が0.6の最大のメリット
- T2 0.7：Webサービス連携
- T2 0.8：SSO対応、プラグイン改善
- T2 0.9：例外ハンドリング機能強化
- テストフレームワークの改善
- Web2.0系Viewの簡単開発機能

T2プロジェクト

- T2プロジェクト
  - http://code.google.com/p/t-2/
- シンプルで有用なフレームワークやツールセットを提供するOSSプロジェクト。特にJavaに限っているわけではない。
- T2プロジェクトの製品
  - Commons : 汎用共通ライブラリ
  - Lucy : 軽量DIコンテナ
  - Yonex : SeleniumベースITテストツール
  - Vili : 汎用プロジェクト作成支援Eclipseプラグイン

T2プロジェクト

- committer
  - shot6
    - http://d.hatena.ne.jp/shot6/
  - skirnir
    - http://d.hatena.ne.jp/skirnir/
  - c9katayama
    - http://d.hatena.ne.jp/c9katayama/
  - yone098
    - http://d.hatena.ne.jp/yone098/



謝辞

御清聴
ありがとう
ございました