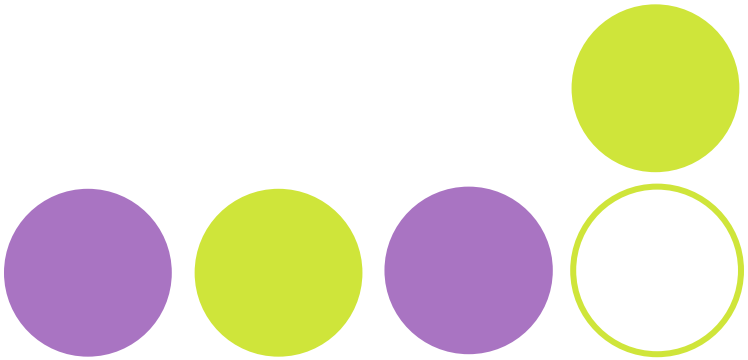
Seasar Conference
2009 Autumn

T2のいま そしてこれから

T2 project
http://t-2.googlecode.com/
大谷 晋平(shot6)

アジェンダ

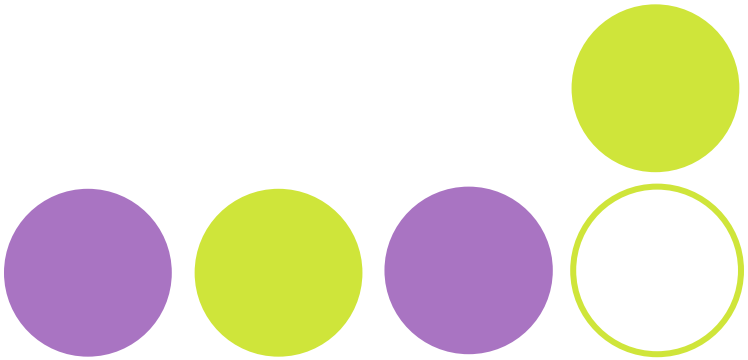
- 自己紹介とチームの紹介
- Web開発でいまおきていること
- T2概要
- T2主要コンポーネント
- デモ
- まとめ



Seasar Conference 2009 Autumn

自己紹介

- 大谷 晋平(おおたに しんぺい)
- オープンソースプログラマ
  - Java/ActionScript/JavaScript/C#...
- ISIDってところで働いてます
- Blog
  - http://d.hatena.ne.jp/shot6/
- Twitter
  - http://twitter.com/shot6/



T2チームの紹介

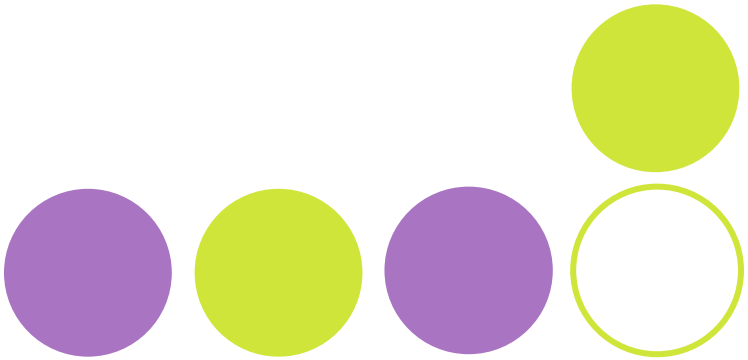
- 五人のコミッタ
  - 四人は遭遇率高い、一人レアキャラ
- 全員プログラマ、特にJava
- 全員がフレームワークとアプリの両方をばりばり作る人達。
  - 五人それぞれの経験をT2に注入

Web開発でいまおきていること

- Webアプリケーション再考のときが来た
  - RIAは既に普通の選択肢のひとつ
    - Ajax, Flex, Silverlightとか普通。
  - クラウド環境への適用
  - リアルタイム性の確保、スマートフォン
  - HTTPベースなプロトコルやフォーマット全盛期の到来
    - AMF, XMPP, reversehttp, pubsubhubbub
    - またはSOAP

では何が必要になるのか？

- RIAからRIPへ
- ユニファイドプロトコルHTTP
- クラウド可用性

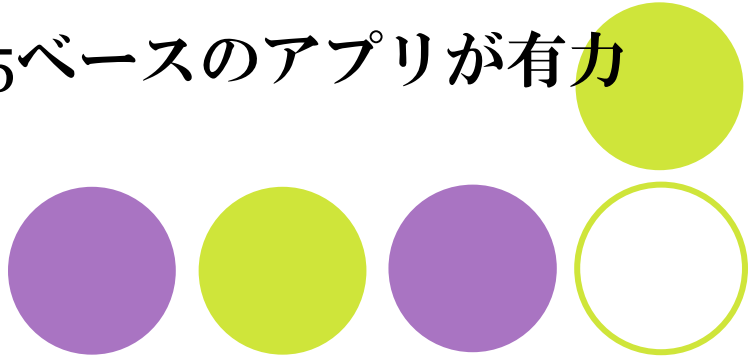


RIP –Rich Internet Platform-

- RIP(Rich Internet Platform)
  - 複数のRIAをサポートするプラットフォームが重要(HTML5, Ajax, Flex, Silverlight...).
  - マルチパラダイム指向
  - RIAによって、サーバサイドが影響を受けない
  - HTML5がその中でも最も重要
    - HTML4とは違い, HTML5はプラットフォーム.
    - 特にランタイムが重要
      - **WebSockets、WebStorage、WebWorker CacheManifest、Geolocation API**

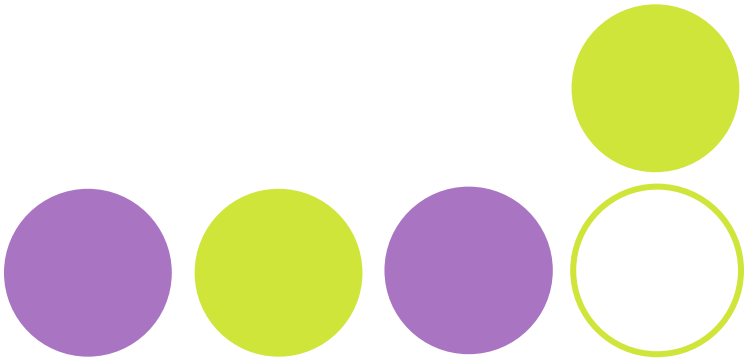
ユニファイドプロトコルHTTP

- HTTP/HTTPS最強説
  - どんな環境でもHTTPを喋るように。
    - 簡単にやりとりが可能。ブラウザのみじゃない。
  - HTTPベースのフォーマットも同様に重要
    - FlexのAMFや, jabberやGoogleTalkなどのXMPP(BOSH)、PubSubHubbubなど。
    - インタラクティブ性やリアルタイム性が求められる場合に、フォーマットやプロトコルが重要
    - スマートフォンでもHTML5ベースのアプリが有力



クラウド可用性

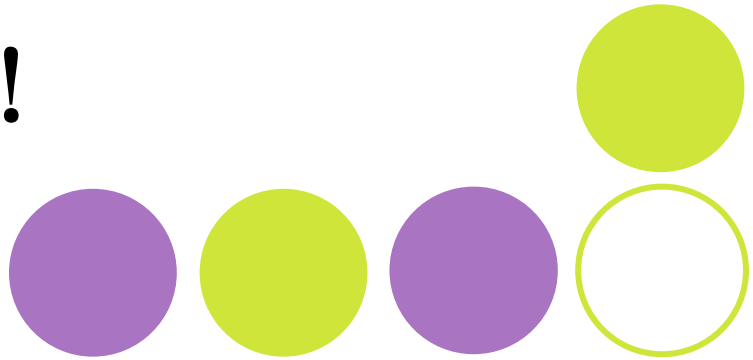
- クラウド可用性
  - クラウド環境で上手く動作する
  - 上手く動作するとは:
    - デプロイが簡単
    - スケールしやすいように原則ステートレス
    - 複数のクラウド環境で同じように動く
      - GAEやAmazonだけでなく、通常のアプリ環境もあわせて同じように動く事が重要.



Webアプリ次のステップ

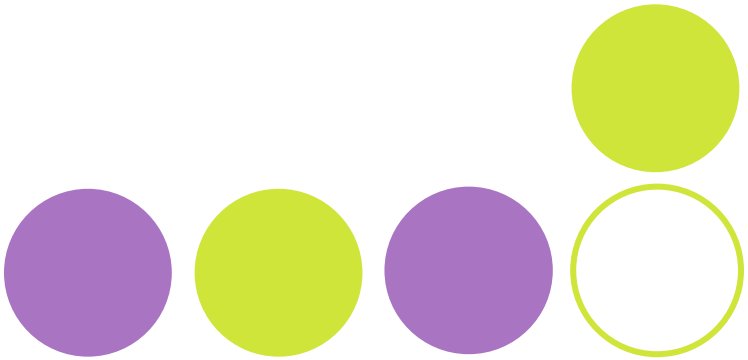
- これらを指向して、シンプルに実現するフレームワーク選びはとても重要:
  - RIAでなくRIP
  - HTTPベースの複数フォーマットや仕組みをサポートする
  - クラウド可用性

そこで T2 ですよ！

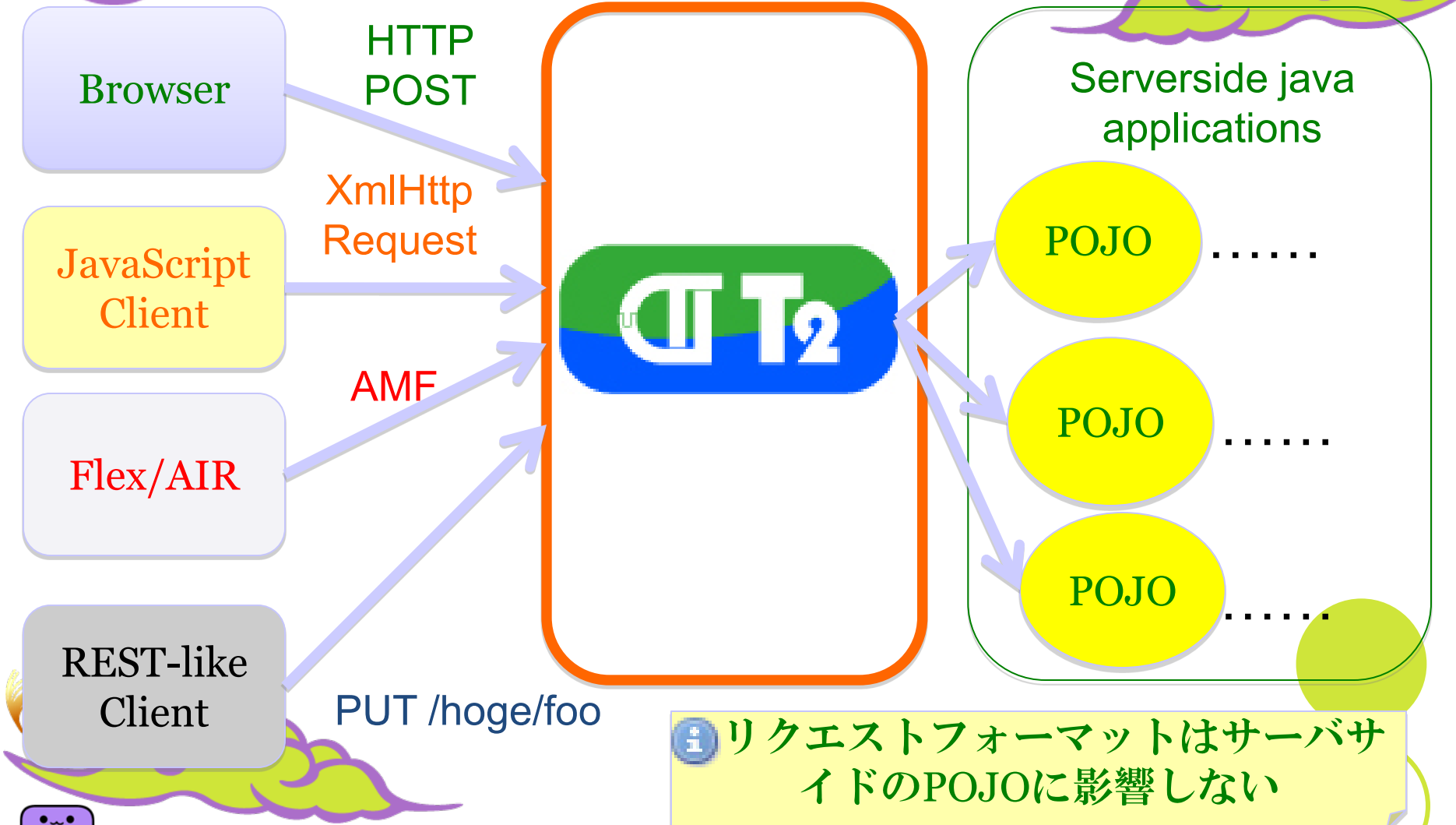


T2とは

- T2とはシンプルで使いやすい現代的なWebフレームワーク
  - アノテーションを使って簡単開発&メンテ
  - 最新バージョンは0.6.1-ga. ASL2ライセンス.
  - http://code.google.com/p/t-2/ で開発中.
  - 勿論旧来のWebアプリも作りやすい！
  - 拡張可能かつ組み込みやすい



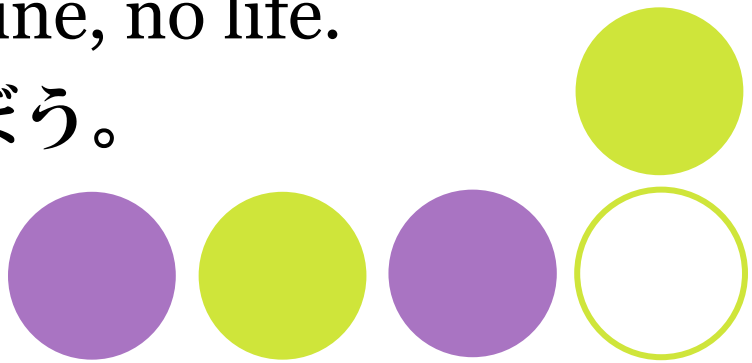
T2概要図



リクエストフォーマットはサーバサイドのPOJOに影響しない

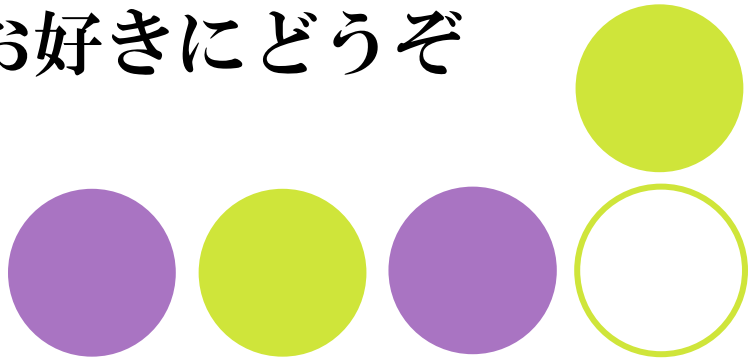
何故T2を作ったのか

- 何故T2を作ったのか
  - JavaのWebフレームワークは開発も保守も難しすぎる。
  - 近代的なWebクライアントのサポートが無い
    - Ajax, AMF, RESTリクエストなんかを同一のサーバサイドのモデルで開発できる？
  - No clean url, no life.
  - No choice of template engine, no life.
  - CoCより少ない設定を選ぼう。



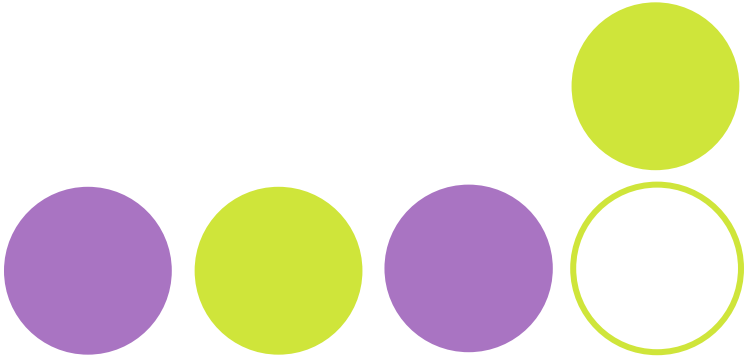
何故T2を作ったのか(続き)

- Webフレームワークってこうあるべき
  - もっともっと簡単でいい
  - 単一の簡単で使いやすいモデルを提供すべき
  - 色々なリクエストを上手く扱おう
    - Ajax、AMF、RESTなど
  - 綺麗なURLを提供しよう
    - 強制はしないけど
  - テンプレートエンジンはお好きにどうぞ



コンセプト

- コンセプト
  - 一つの事をうまくやろう！
    - T2はHTTPクライアントからの接続を適切なPOJOに適切なフォーマットで渡すだけ。他の事はしない。
  - WEBの多様性を受け入れよう
    - 色々なクライアントがこれから出てくる。それぞれのやり方を受け入れ、でもサーバサイドは同じ。
    - 一つだけ確かな事は、HTTP をしゃべり、URLを持つ事。それだけが大事。

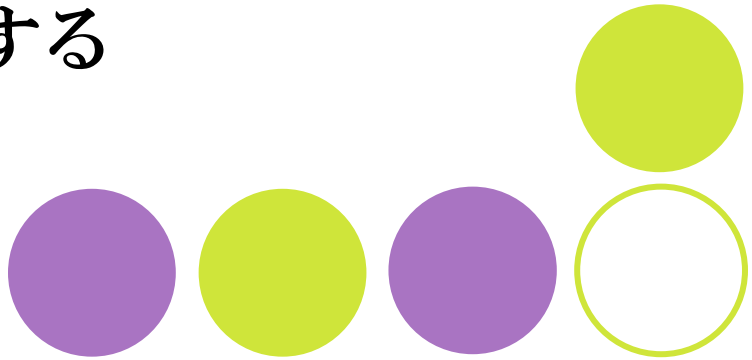


想定ユーザ

- T2を使う想定ユーザ
  - こんな開発者の方々:
    - 自分で作るWebアプリを用意に開発&コントロールできるようにしたい.
    - クライアントサイドとサーバサイドをキレイに分離したい
    - 自前フレームワークをシンプルかつ拡張容易でコントローラブルにしたい
    - RESTやAjax, Flexなどを使ってRIAアプリをさっくり作りたい
    - クラウドベースのアプリをさっくり作りたい

プログラミングモデル

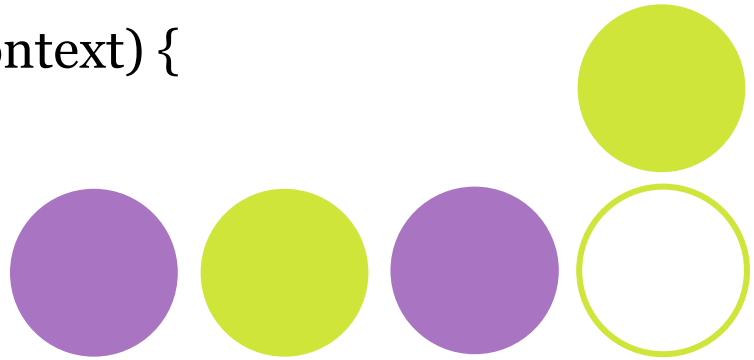
- Pageモデル
  - T2を使ったアプリの基本
  - 基本的にPOJOだけど、アノテーションでエンハンスする
  - Pageは1URLと1POJOをマッピングしたもの
  - Pageは アクションメソッド を持つ
  - アクションパラメータ はアクションメソッドの引数でT2がインジェクトする



プログラミングモデル(続き)

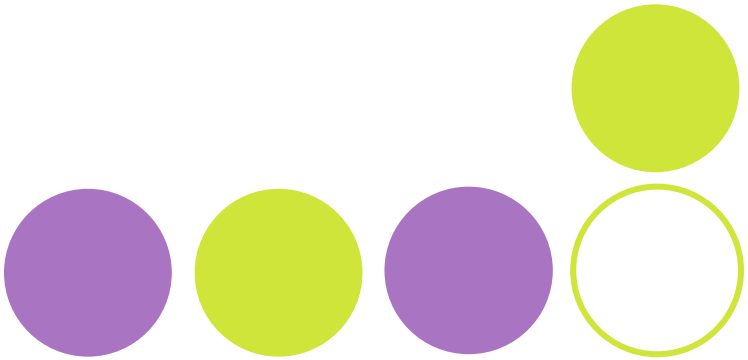
- Example:

```
@Page("employee")
public class EmployeePage {
  @Default
  public Navigation index(WebContext context) {
      ......
  }
  @GET
  @ActionPath("list/detail")
  public Navigation detail(WebContext context) {
      ......
  }
}
```



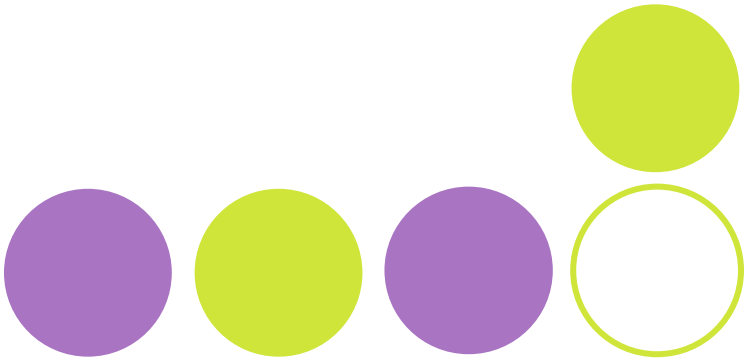
アクションメソッドデザイン

- Funnel(ろうと)デザイン
  - Pageは大枠から詳細へ徐々に設定するように作る事が出来る
    - PageにアノテーションでURLをつける
    - ->アクションメソッドにHTTPメソッドをつける
    - ->アクションメソッドにURLをつける
    - ->->インジェクトしてほしい引数を並べる



アクションメソッドデザイン

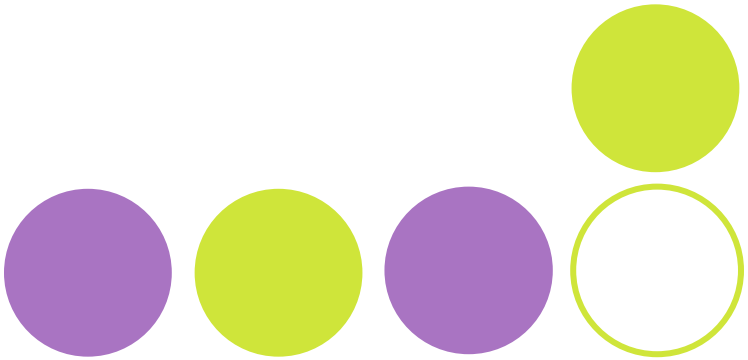
- Example: HTTP GET /hoge/foo?bar=2
  - Pageクラスが/hogeをURLとする
  - アクションメソッドで@GETをつける
  - アクションメソッドで/fooをURLとする
  - アクションパラメータbarをStringで受け取る



アクションメソッドデザイン

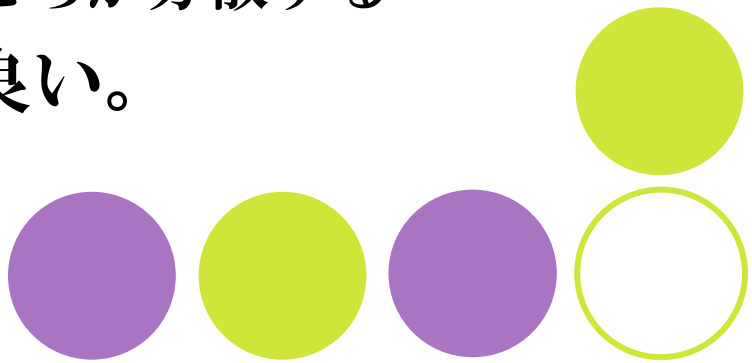
- こんなかんじ:

```
@Page("/hoge”)
public YourPage {
  @GET
  @ActionPath(“hoge”)
  public Navigation hoge(@RequestParam("bar") String bar) {
    ……
```



アクションパラメータデザイン

- Struts 1.xを思い出そう
  - Actionは引数で固定のものがわたってきた
  - あれはあれで良いモデル
- T2は？
  - もう少しバリエーションをつけたが原則同じ
  - パラメータで貰えれば一番わかりやすい
    - Setterで貰うと見るべきところが分散する
  - 対象メソッドだけ見れば良い。



アクションパラメータデザイン

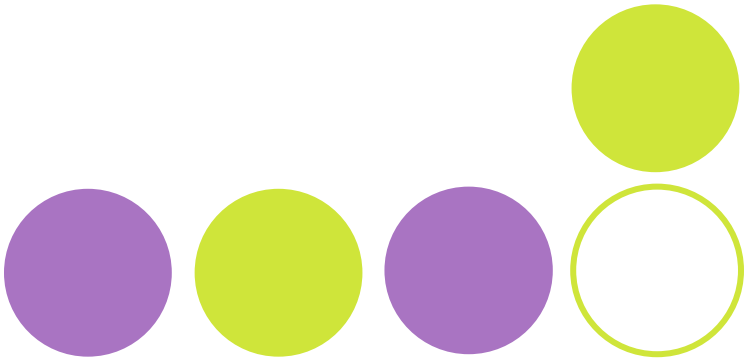
- 以下のようなアクションパラメータ
  - T2が準備したコンテキスト
    - WebContext/Request/Response...
  - Servletのコンテキスト
    - HttpServletRequest/HttpServletResponse...
  - フォームデータなどをPOJOに変換したもの
    - From HTTP POST, or from Flex AMF
  - その他便利なもの
    - REST-like なURLの部分、foreachのインデクス、ファイルアップロード

コードはこんな感じ!

```
@Page(”sample")
public class SamplePage {
  @Default
  public Navigation index(WebContext context) {...
  @Amf
  public Navigation execute(HogeDto hoge) {...
 @ActionParam
  public Navigation submit(@Form FooDto foo) {...
 @ActionPath(“fuga/{moge}”)
  public Navigation submit(@Var(“moge”) String s) {...
```

主要コンポーネント

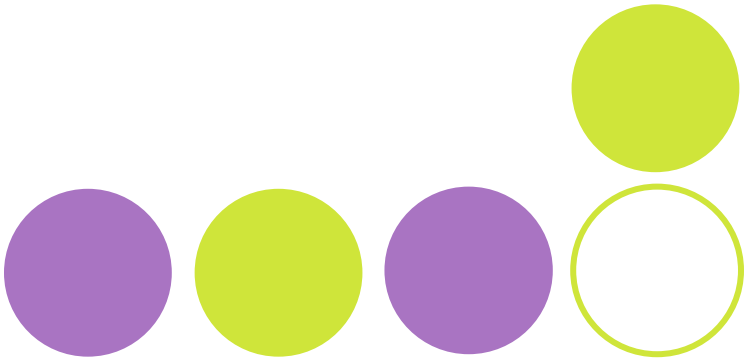
- @Page
- HTTPハンドル機能 @GET/@POST
- URLパターン機能 @ActionPath
- データマッチング機能 @ActionParam
- タイプマッチング機能 @Ajax/@Amf
- Navigation
- ContainerAdapter



Seasar Conference 2009 Autumn

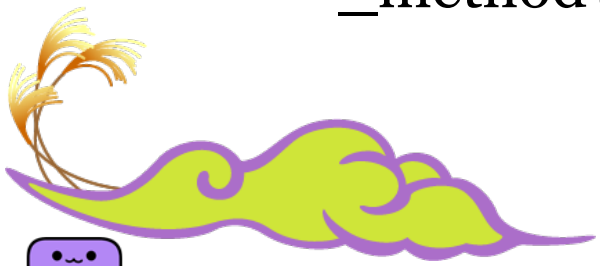
@Page

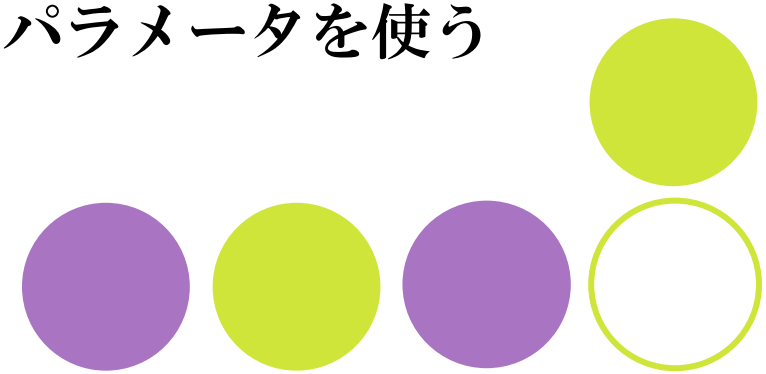
- T2 Pageクラスのルートアノテーション
  - @Pageは必ずつけないと駄目
  - T2は最小限設定にするために、web.xmlのルートパッケージ設定と@Pageだけで設定完了
  - @Pageで大枠のURLを指定する



HTTPマッチング

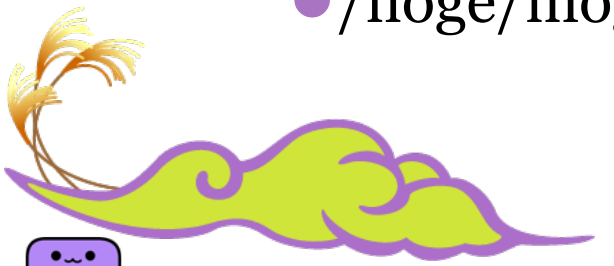
- HTTPメソッドごとに呼び出すメソッドを変える
  - GET, POST, PUT, DELETE, またはHEAD.
  - 各アノテーションで区別される
  - 原則アクションメソッドにはつける
    - HTTPメソッドごとに処理内容は違う
    - PUT/DELETEはブラウザから呼べないので、_methodというリクエストパラメータを使う

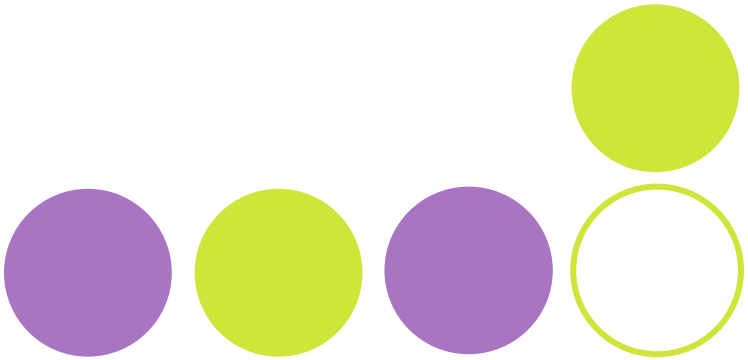




URLマッチング

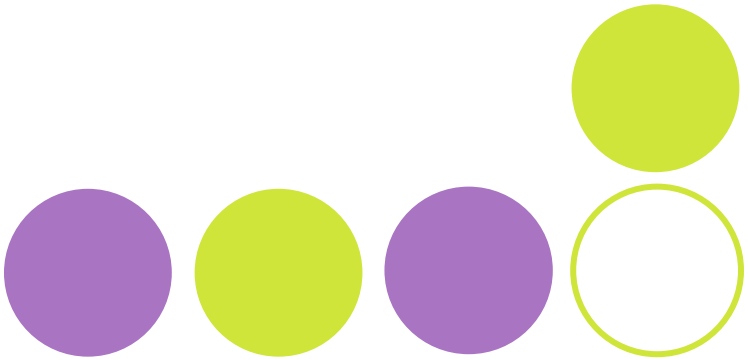
- リクエストされたURLごとに呼び出すメソッドを変える
  - @ActionPathがその中心
  - @Page(“hoge”) + @ActionPath(“foo”)
    - /hoge/fooがアクセスされるURL
  - @Page(“hoge”) + @ActionPath(“foo/{bar}”)
    - /hoge/foo/aaaはOK
    - /hoge/moge/aaaはNG





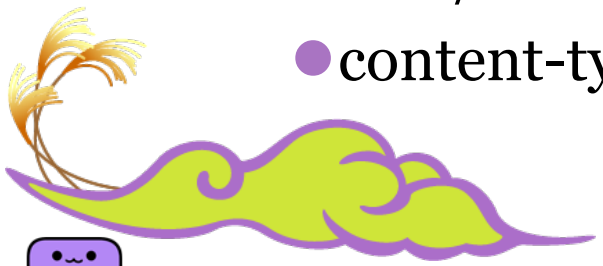
データマッチング

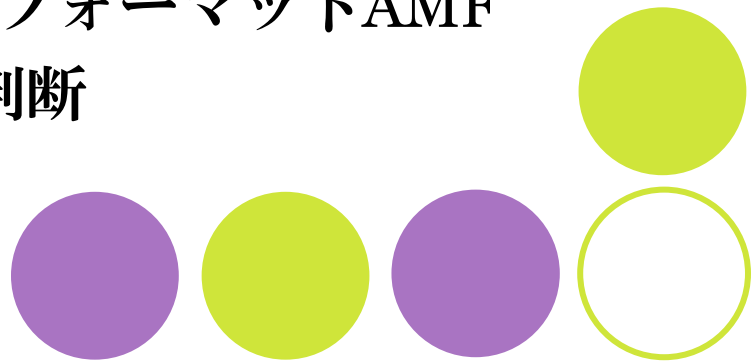
- 送信されてきたデータで呼び出すメソッドを変える
  - @ActionParamがその中心
  - リクエストパラメータ名とメソッド名が同一なら呼び出す
    - Submitボタンがおされたらsubmitメソッドを呼ぶ
    - 直感的でわかりやすい



リクエストタイプマッチング

- リクエストタイプによって呼び出すメソッドを変える
  - @Ajax
    - XmlHttpRequestのみを受け付ける
    - 近代的jsフレームワークの習慣的につけるヘッダである“X-Requested-With”を利用
  - @Amf
    - Flex/AIRの高効率バイナリフォーマットAMF
    - content-typeヘッダなどで判断



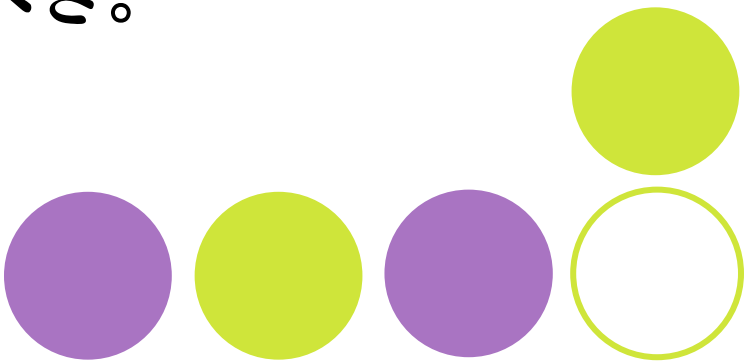


リクエストタイプマッチング

- リクエストタイプはとても重要！
  - Ajax呼び出しとフォームのサブミットは求められる特性が違う。
    - 同期、非同期とパフォーマンスを意識する
    - T2では明示的に分ける工夫がされている

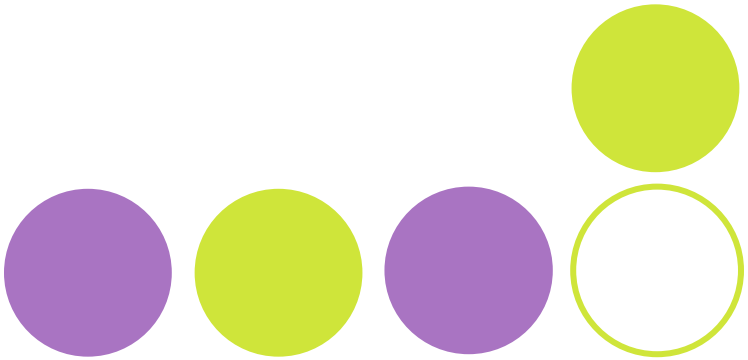
デフォルトメソッド

- デフォルトメソッドは結構重要
  - 変なエラーを起こさない
  - Pageごとに適切なデフォルト動作をセット
- @Defaultがそのアノテーション
  - 一般的にはそのPageが受け持つ遷移先を返す
  - 状態のクリアなんかも必要な場合も。
  - Pageに一個定義しておくべき。



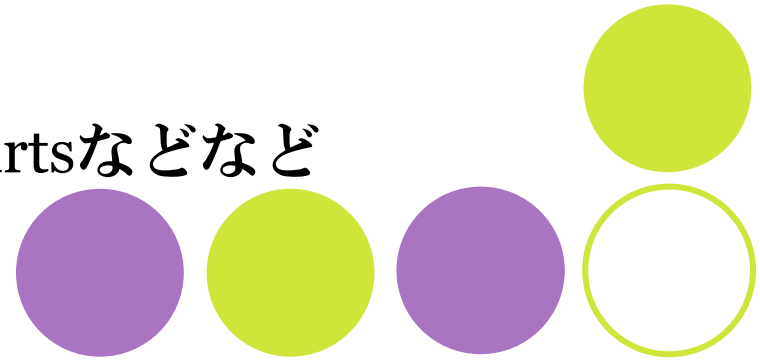
Navigation

- Navigationは開発者が最も使うAPI:
  - T2に次にどこに遷移するか伝える
  - T2にどのようにレスポンスに書くのかを伝える
- 特徴
  - 簡単に使える
  - 拡張や自前実装も簡単



用意されているNavigation

- 大体こんなもんがある
  - Forward/Redirect/Proceed(携帯用仮想のリダイレクト)
  - Json(またはSecureJson)
  - Direct(ストリームレスポンス)
  - AmfResponse(Flex/AIR AMF)
  - CacheManifest(Generate HTML5 manifest)
  - 将来は
    - CVS, PDF, GoogleMap, Chartsなどなど



ContainerAdapter

- DIコンテナは今やデファクトな技術
  - T2はDIコンテナへのアダプタ機能を提供
    - Spring/Guice/Seasar2またはLucy
  - web.xmlで設定

```
<init-param>
  <param-name>t2.container.adapter</param-name>
  <param-
value>org.t2framework.t2.adapter.SpringAdapter</param-
value>
</init-param>
```

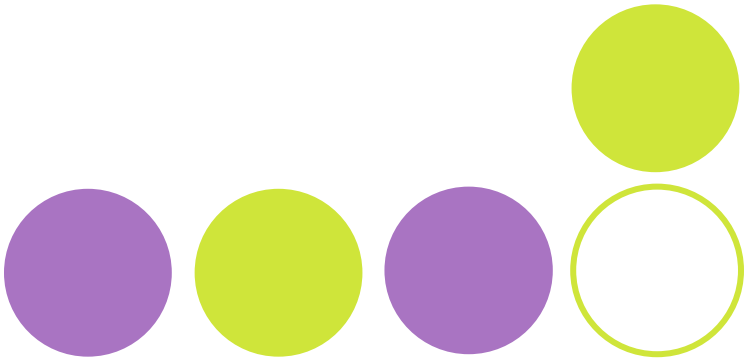
デモ

- RIA+Cloudのデモ
  - Flex + GAEJ + T2
  - T2が持っているAMF3実装を使用
- 僕のiPhoneはどこいった！？デモ
  - Geolocation + Ajax + T2 + iPhone Webサーバ (Serversman)
- Mayaa連携
- PureMVC(Flex) + T2 + Springデモ

時間あれば。

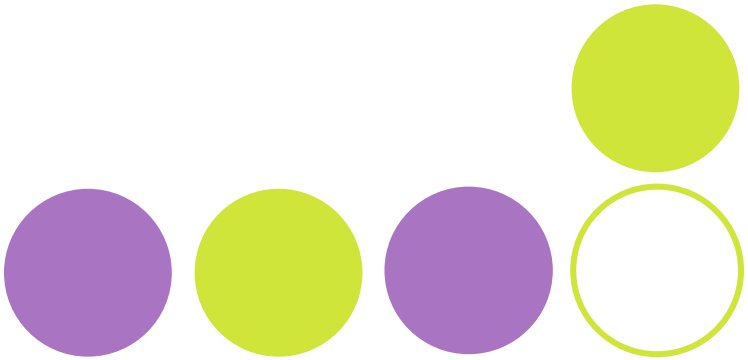
ロードマップ

- T2 0.6.x
  - コア機能は固まった。
  - しばらく0.6系でバグフィックスする予定
  - サービスブラウザを追加する
  - RESTサポート。POSTオーバロードなど
  - T2AMFの機能強化



ロードマップ

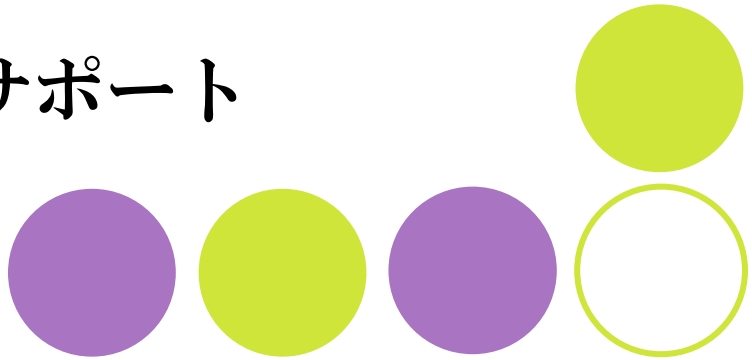
- T2 0.7以降予定
  - Navigationをより豊富に。
    - RSS, CSV, PDF, GoogleMap, Chartなど。
  - HTML5対応
  - WebService対応
    - @WebServiceのようなやつ
  - XMPP対応
  - SSO/OpenID対応
    - OpenSSO/OpenID4Java



Seasar Conference 2009 Autumn

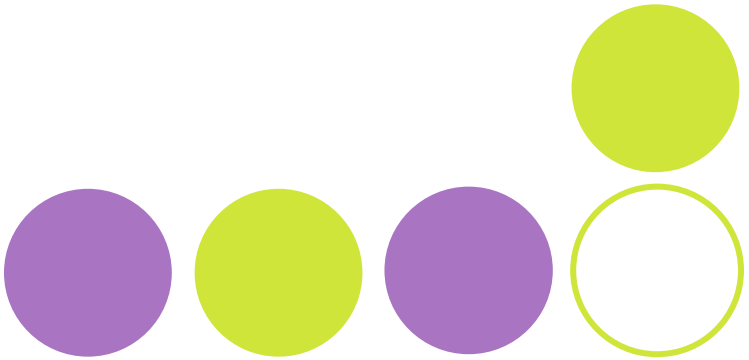
まとめ

- Webアプリケーションは再考の時期に来ている。変化の時期が来ている。
  - RIAからRIPへ。中心はHTML5
  - クラウド環境
  - HTTP最強説と拡張フォーマットの利用
- T2はシンプルで開発しやすく、変化に強いフレームワーク
  - 現環境も新環境も強力にサポート



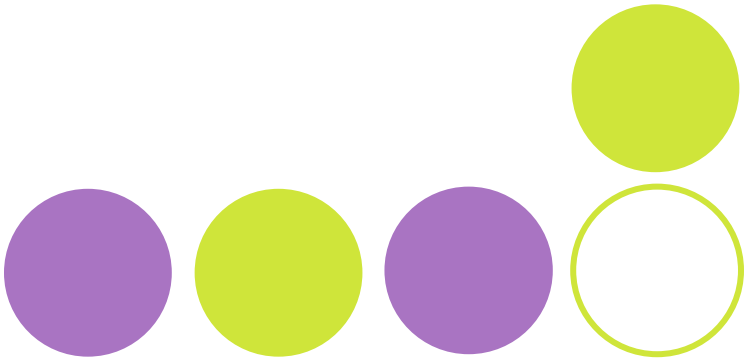
リソース

- サイト
  - http://code.google.com/p/t-2/
- ダウンロード
  - http://code.google.com/p/t-2/downloads/list
- Mavenサイト
  - http://maven.t2framework.org/maven2/



リソース

- ドキュメント
  - http://code.google.com/p/t-2/wiki/Index
- メーリングリスト
  - http://groups.google.com/group/t2-users-en
- ソース
  - http://code.google.com/p/t-2/source/checkout



Seasar Conference
2009 Autumn

ありがとうございました

ご清聴ありがとうございました。
これからもT2を宜しくお願いします！

次のセッションも来てね^^