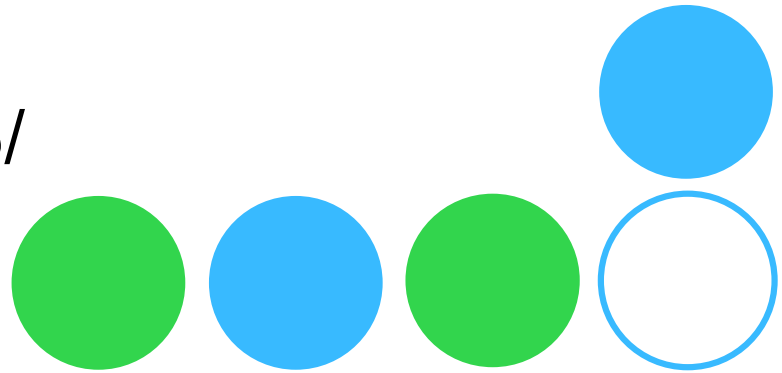
T2 –The WEB Connector-

T2プロジェクト
(http://code.google.com/p/t-2/)
大谷　晋平(shot6)

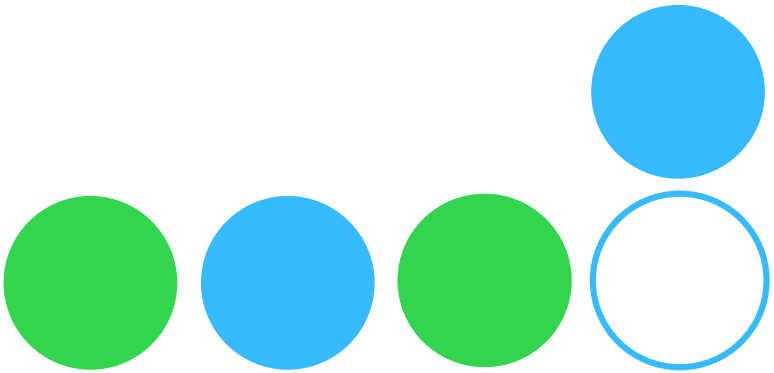
自己紹介

- おなまえ
  - おおたに　しんぺい(shot6)
- おところ
  - Webのどこか、またはISIDという会社
- 職業
  - プログラマ
- ブログ
  - http://d.hatena.ne.jp/shot6/



自己紹介つづき

- しょぞく
  - T2プロジェクト
  - 何でもやる的オープンソースプロジェクト
    - http://t-2.googlecode.com
    - GoogleCodeで1・2を争うかもしれないコミットっぷり
    - 主な活動はフレームワークを作る、アプリを作る
    - 勉強会や交流する場を作る
    - なぜかマスコットも作る



知っ得！
納得！
Webフレームワーク

マスコット「テツ」

知っ得！
納得！
Webフレームワーク

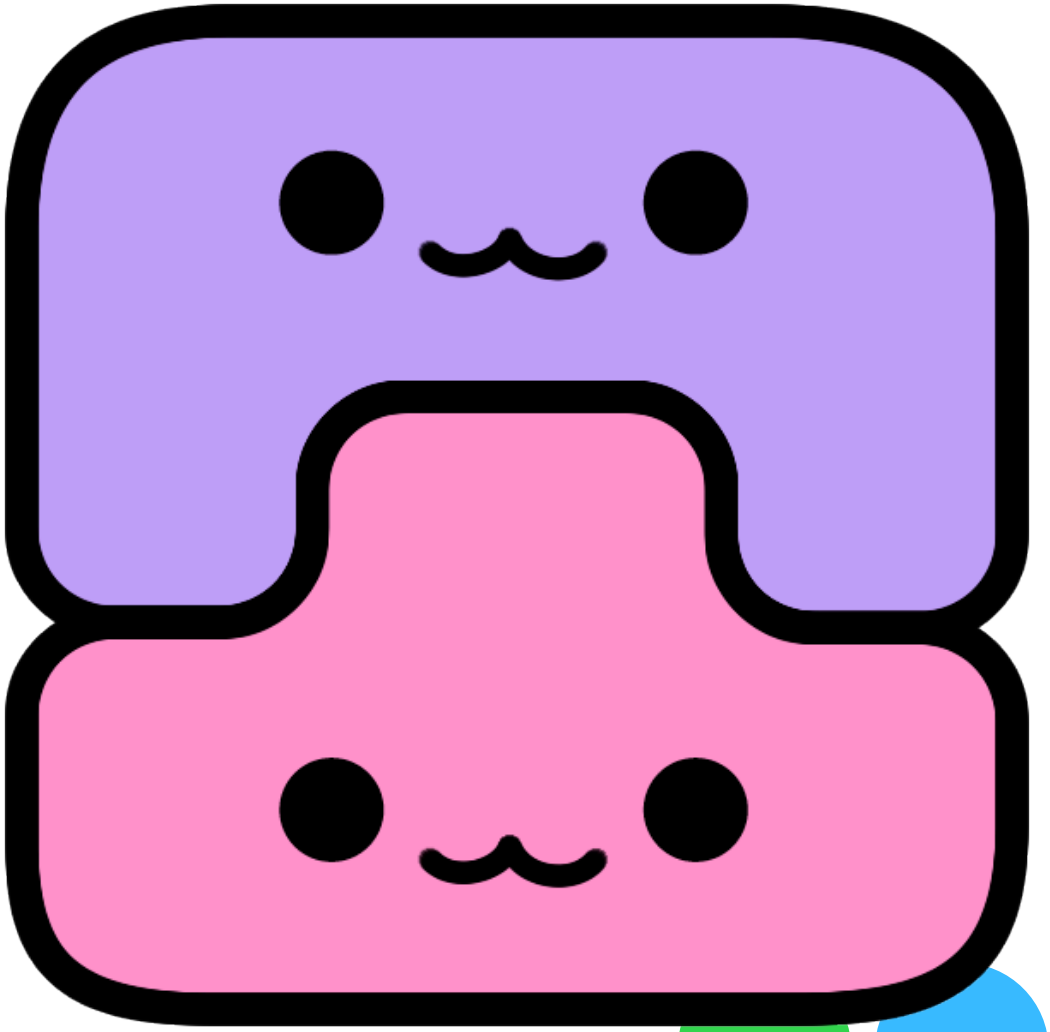
マスコット「テツ、外に出る」

知っ得！
納得！
Webフレームワーク

マスコット「テツ、あいたー」

知っ得！
納得！
Webフレームワーク

マスコット「イーダ」



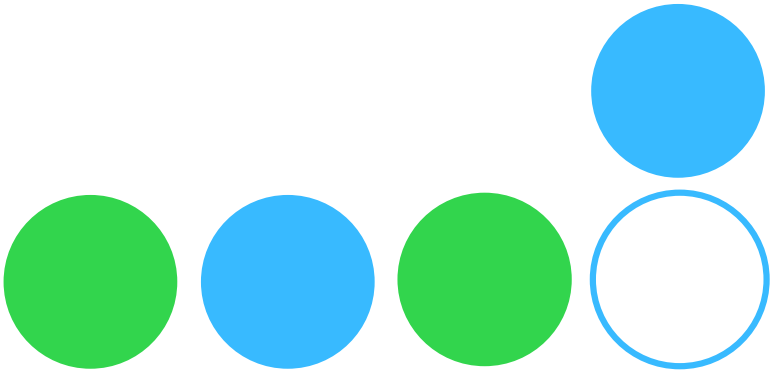
マスコット　デザイン

- デザイナ
  - カネウチカズコさん
  - ねこびーんの作者

なぜこんなことをするのか

- なぜこんなことをしとるの？

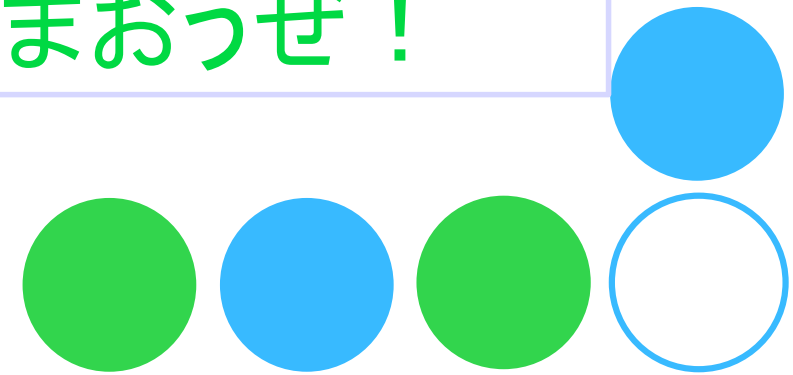
楽しいから！



T2プロジェクトのモットー

- 楽しく、現場で使える物を作って使おう！
  - *Webフレームワーク　T2*
  - DIコンテナ　Lucy
  - 結合テストツール Yonex
  - DaoフレームワークとかFlexフレームワークとか

誰かが作るのを待つんじゃなくて、
自分達でつくっちまおうぜ！

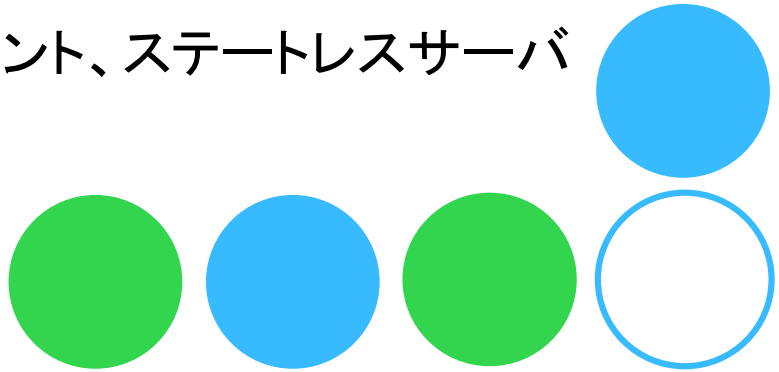


Webフレームワーク T2

- T2ってなんだ？
  - Webにつなげる・つながる事に特化
  - 部品化指向
    - 組み込まれビリティ重要
    - 大掛かりな仕組みではなく、小さく組み込みやすく。
  - Web2.0フレームワーク
    - 現代的なクライアントからのリクエストを淡々とさばく
      - 通常のFormサブミット
      - XmlHttpRequest
      - FlexからのAMF形式のリクエスト
      - SilverlightからのXML形式のリクエスト

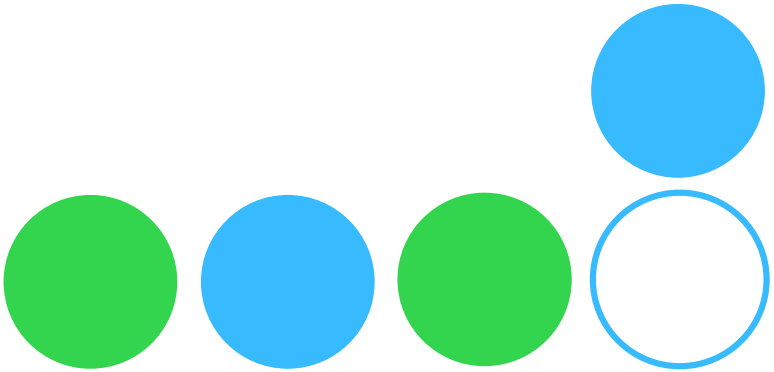
Webフレームワーク T2

- T2ってなんだ？つづき
  - アノテーションベース
    - クラス、メソッド、引数アノテーションを使う
  - 設定レス
  - DIコンテナ非依存
  - ステートレスなフレームワーク
    - ステートフルにするのは自由
    - でもステートはクライアントに持たせる方がよい
      - いわゆるステートフルクライアント、ステートレスサーバ



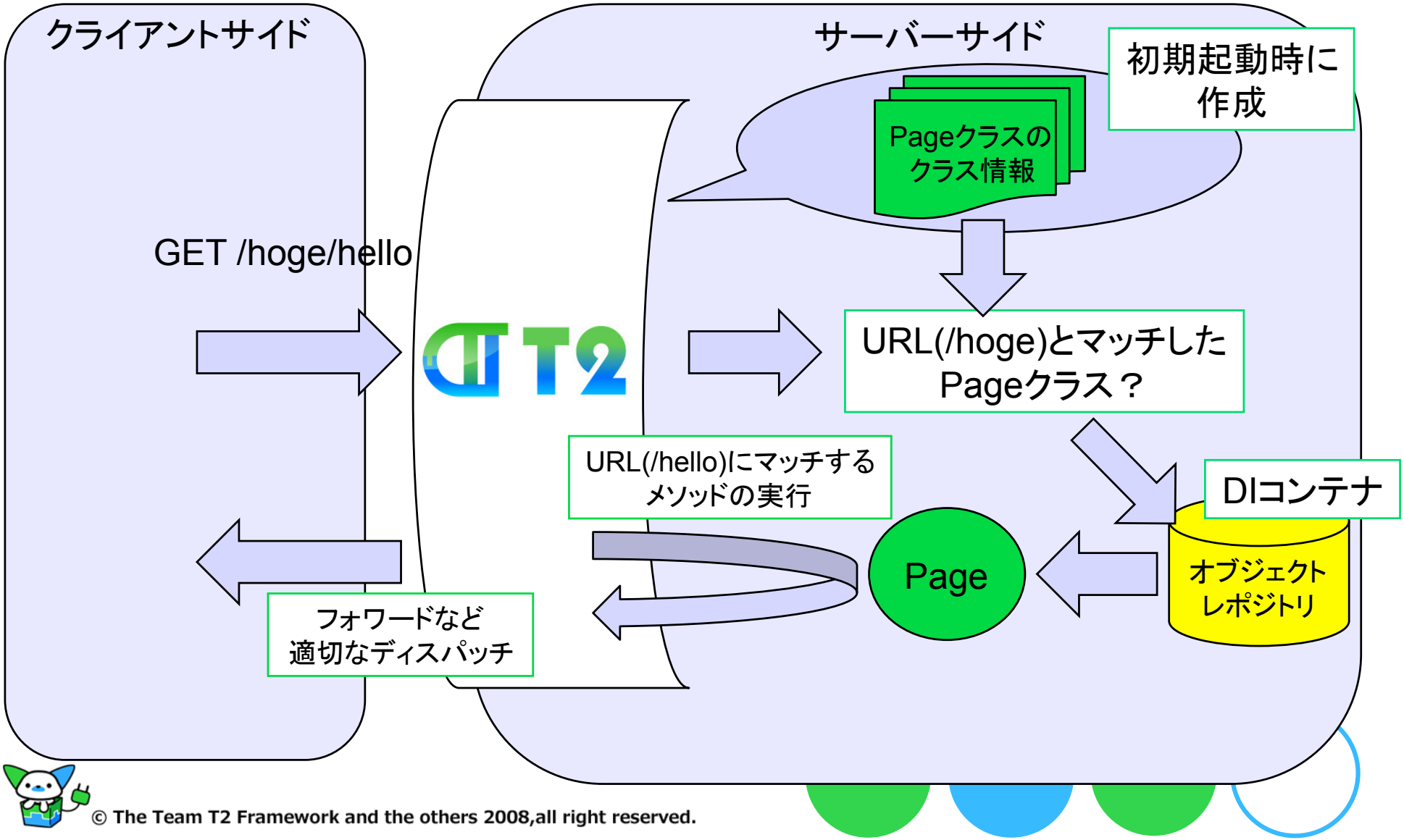
Webフレームワーク T2

- T2ってなんだ？つづき
  - きれいなURL
    - RESTライクなURLをもてる
  - マルチビューフレームワーク
    - JSPやテンプレートエンジンを選ばない
  - フィルタ指向
    - 処理の差し込みはしやすい
    - Cubby/Ymirとかと同じ



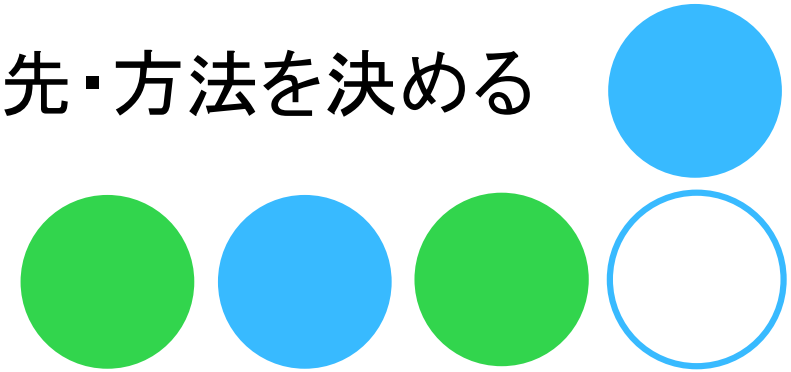
知っ得！
納得！
Webフレームワーク

T2の仕組み

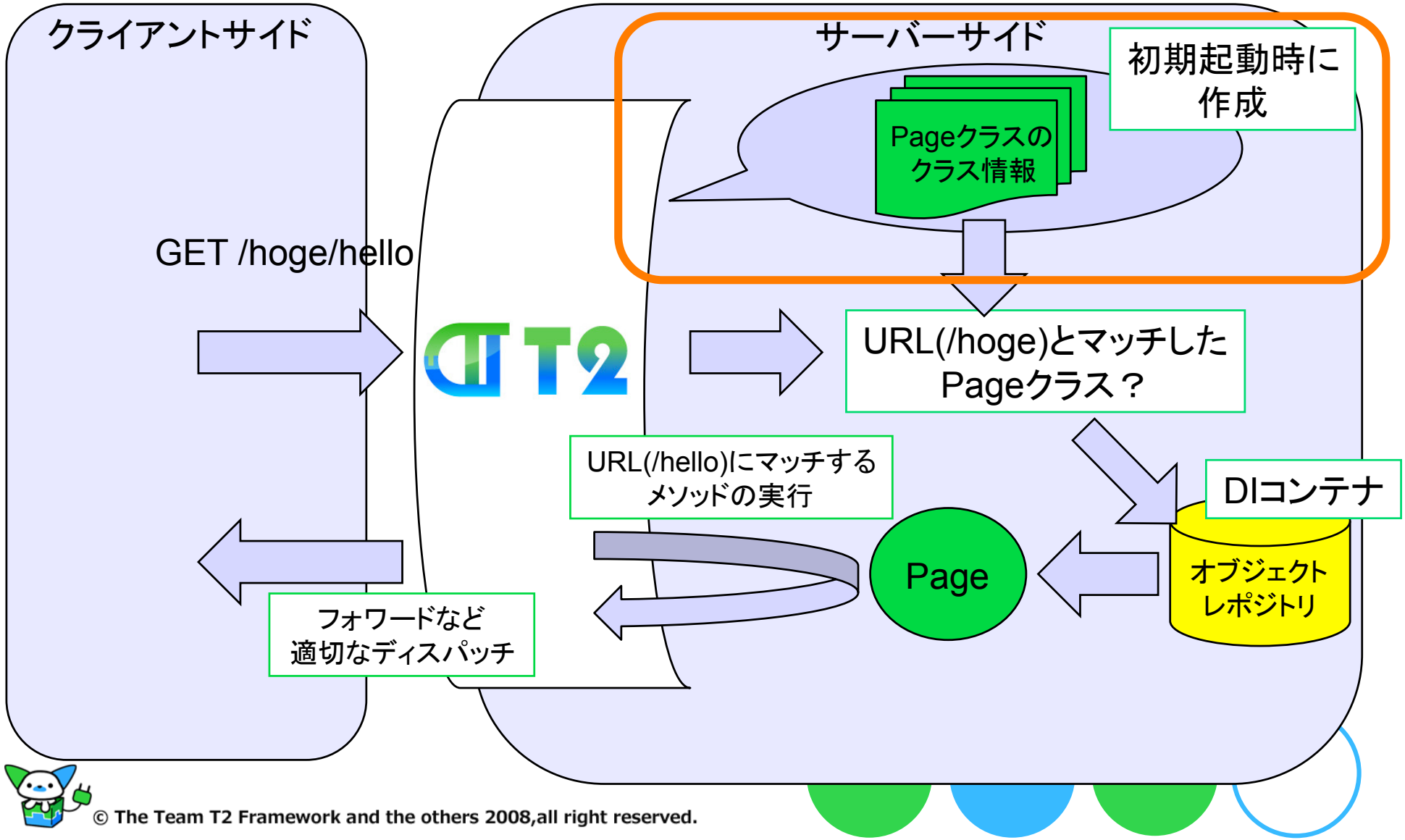


実行フロー

- 実行フローはこんな感じ
  - Pageクラスをかきあつめる
  - リクエストURLにマッチしたPageクラスをインスタンス化する
  - そのPageクラス内メソッドのアノテーションをみて、メソッドを実行準備
  - 引数をみて、引数に値をインジェクト
  - メソッド実行
  - 戻り値のNavigationで遷移先・方法を決める



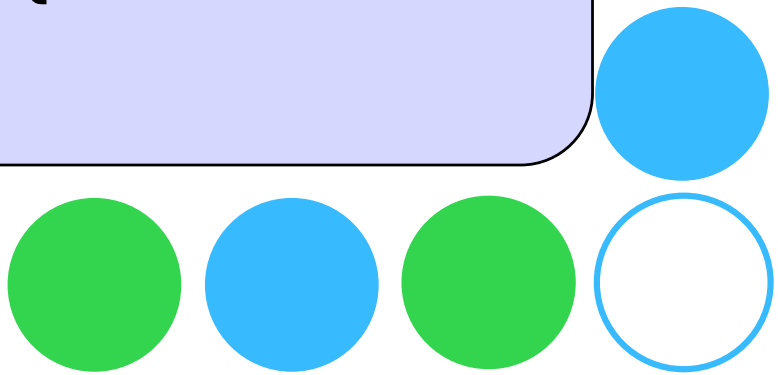
T2の仕組み（初期起動時）



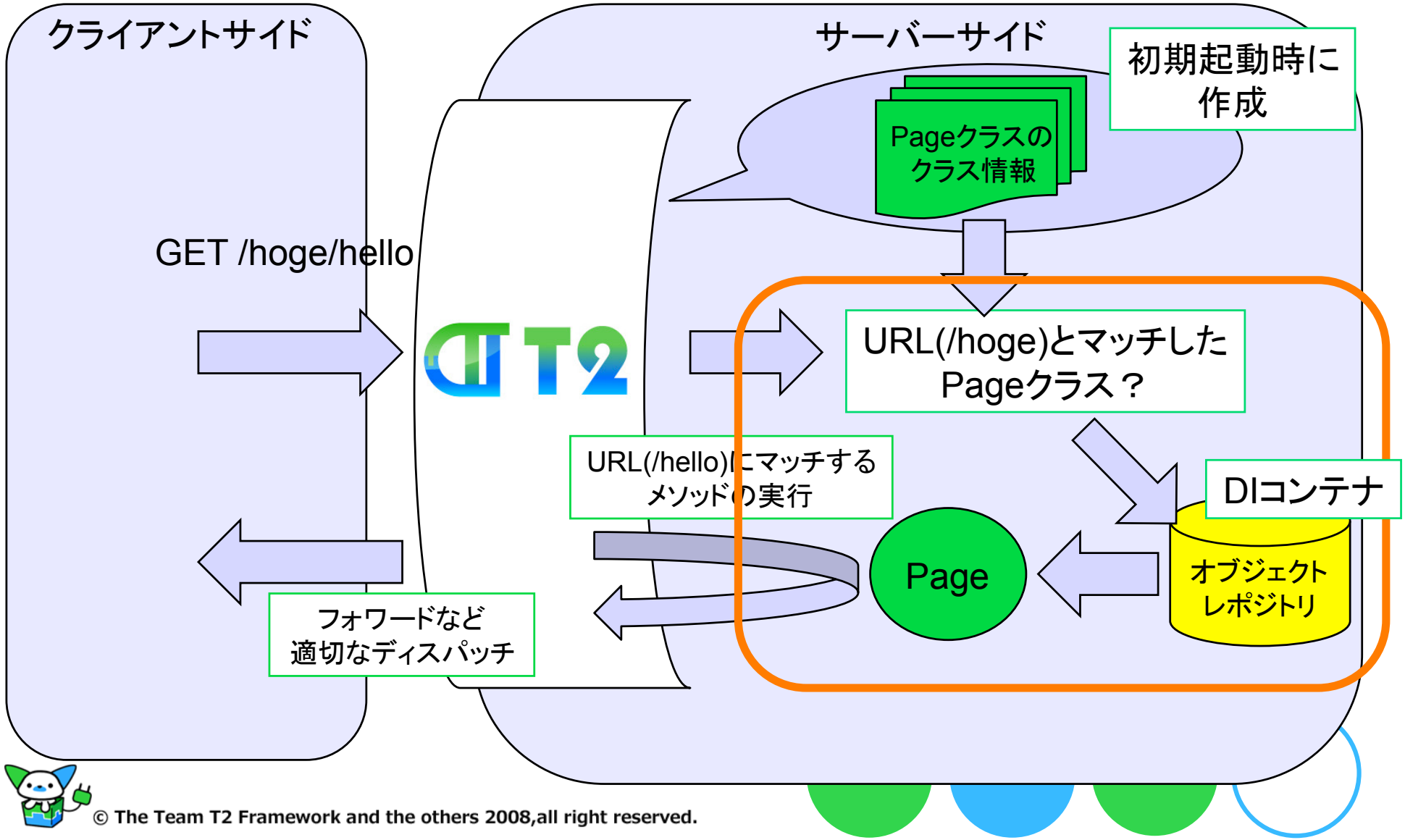
すべてはURL

- T2の初期起動時
  - Pageクラスをかき集める
    - @Pageとついたクラス
    - 指定の特定パッケージ以下
  - @Pageの部分URLでそのPageクラスを特定

```
@Page(“add”) // /context-root/add
public class AddPage {…
}
```

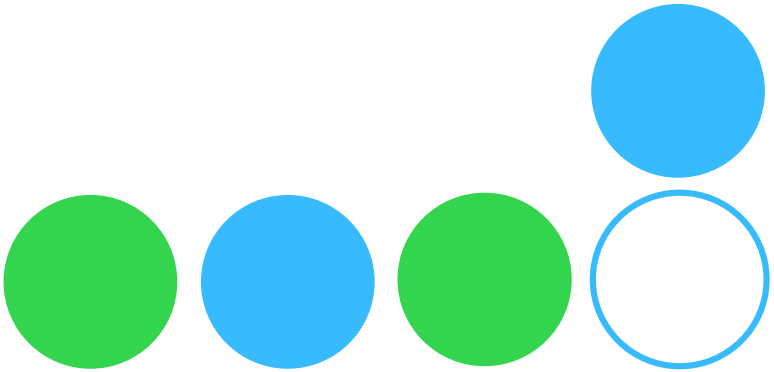


T2の仕組み（リクエスト処理）



リクエスト処理開始

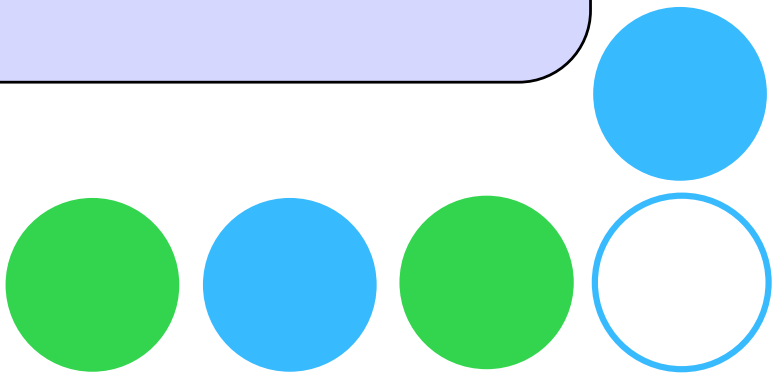
- @PageとURLのマッチング
  - マッチした場合のみT2が処理すべきリクエスト
    - Pageクラスを生成する
      - DIコンテナを抽象化したアダプタに処理を委譲
      - インスタンスの生成・管理はすべておまかせ
      - いまのところLucy/Seasar2/Springに対応
        - Guiceも近いうちに対応
    - 更にURLを深堀りして、次はメソッドとマッチ
      - 次ページにつづく。。。。



メソッドアノテーション

- メソッドアノテーションの条件とのマッチ
  - マッチしたメソッドを実行
  - その1：@ActionPath
    - 特定のURLとマッチしたら、そのメソッドを実行

```
@ActionPath // [page-url]/execute
public Navigation execute() {…
}
```



@ActionPathサンプル

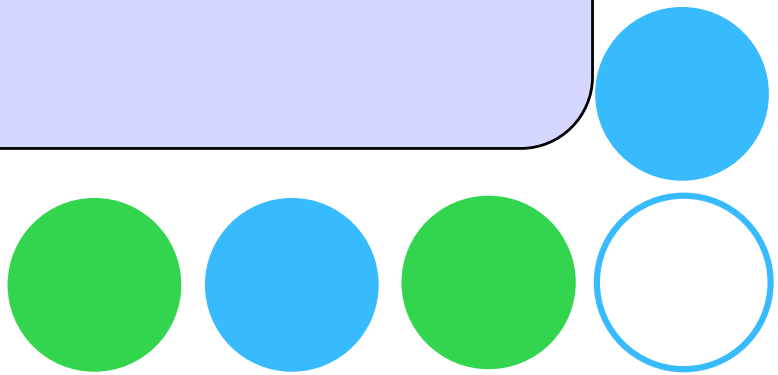
- こんな感じになります

```
 * {@.ja @ActionPath は、URL指定でメソッドが呼べるか呼べないかを指定するアノテーションです.
 * @Page でつけたURL断片＋@ActionPathをつけたメソッド名なURLにリクエストすれば呼ばれます.
 * HelloPageのこのメソッド(requestメソッド)だと、
 *      http://yourdomain/t2-samples/hello/request
 * で呼ばれます. }
 *
 */
@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");
}
```

メソッドアノテーション2

- メソッドアノテーションの条件とのマッチ
  - その２:@ActionParam
    - リクエストパラメータのキーとマッチしたら、そのメソッドを実行する
    - 端的にいえば、押されたボタンがわかる

```
@ActionParam
public Navigation submit() {…
}
```



@ActionParamサンプル

- こんな感じになります

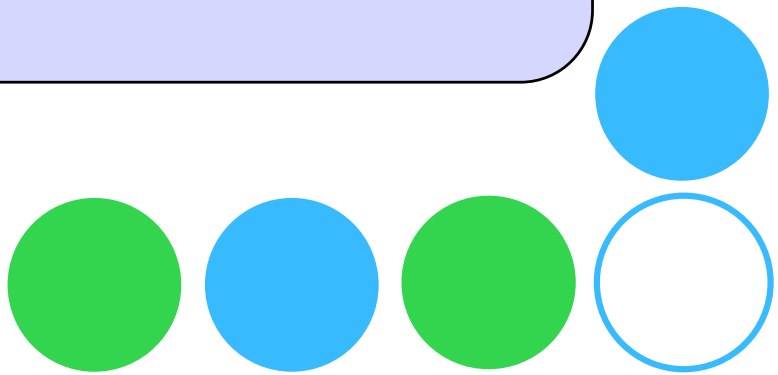
```
     *
     * {@.ja @ActionParam は画面からサブミットされたFORMの中の入力項目に
     * メソッド名と同じname属性を持つ項目があれば、対象メソッドを呼ぶアノテーションです.
     * このアノテーションを使えば、押されたボタンを認識して、処理をユーザのアクション単位できれいに
     * 切り分けられます.
     *
     * @POST の使い方は、{@code GetAnd
     *
     * 総じて、この例では以下のような場合
     *  <li>画面側のFORMのactionが/cor
     *   <li>POSTでサブミットされる</li
     *   <li>サブミットされたフォーム内の
     * }
     */
    @POST
    @ActionParam
    public Navigation add(WebContext context) {
        final Request request = context.getRequest();
        RequestUtil.save("arg1", request);
        RequestUtil.save("arg2", request);
```

<input type="submit" name="add" value="足し算サブミット" />

メソッドアノテーション3

- メソッドアノテーションの条件とのマッチ
  - その3：HTTPメソッドアノテーション
    - HTTPメソッドしばり
    - 以下ならPOSTメソッドだけ。

```
@POST
public Navigation submit() {…
}
```



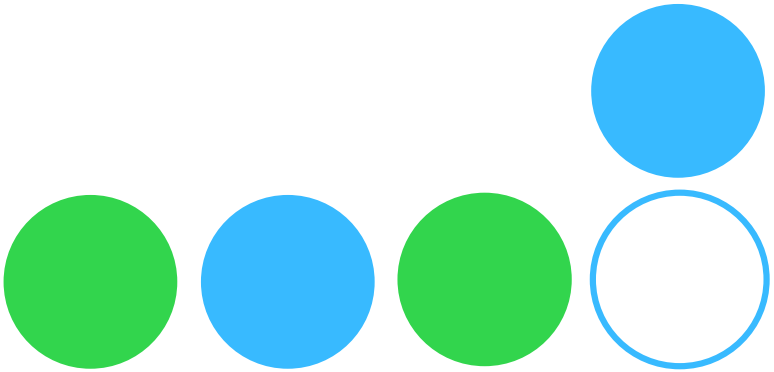
引数アノテーション1

- メソッドが確定したら、引数のインジェクト
  - 暗黙的なもの
    - HttpServletRequest/HttpServletResponse
    - HttpSession
    - ServletContext
    - WebContext(T2独自コンテキスト)

```
@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");
}
```

引数アノテーション2

- メソッドが確定したら、引数のインジェクト
  - 明示的なもの
    - @Form : サブミットされたFormをPOJOでインジェクト
    - @RequestParam/@SessionAttr :リクエスト、セッションの値をインジェクト
    - @Upload : アップロードされたファイル

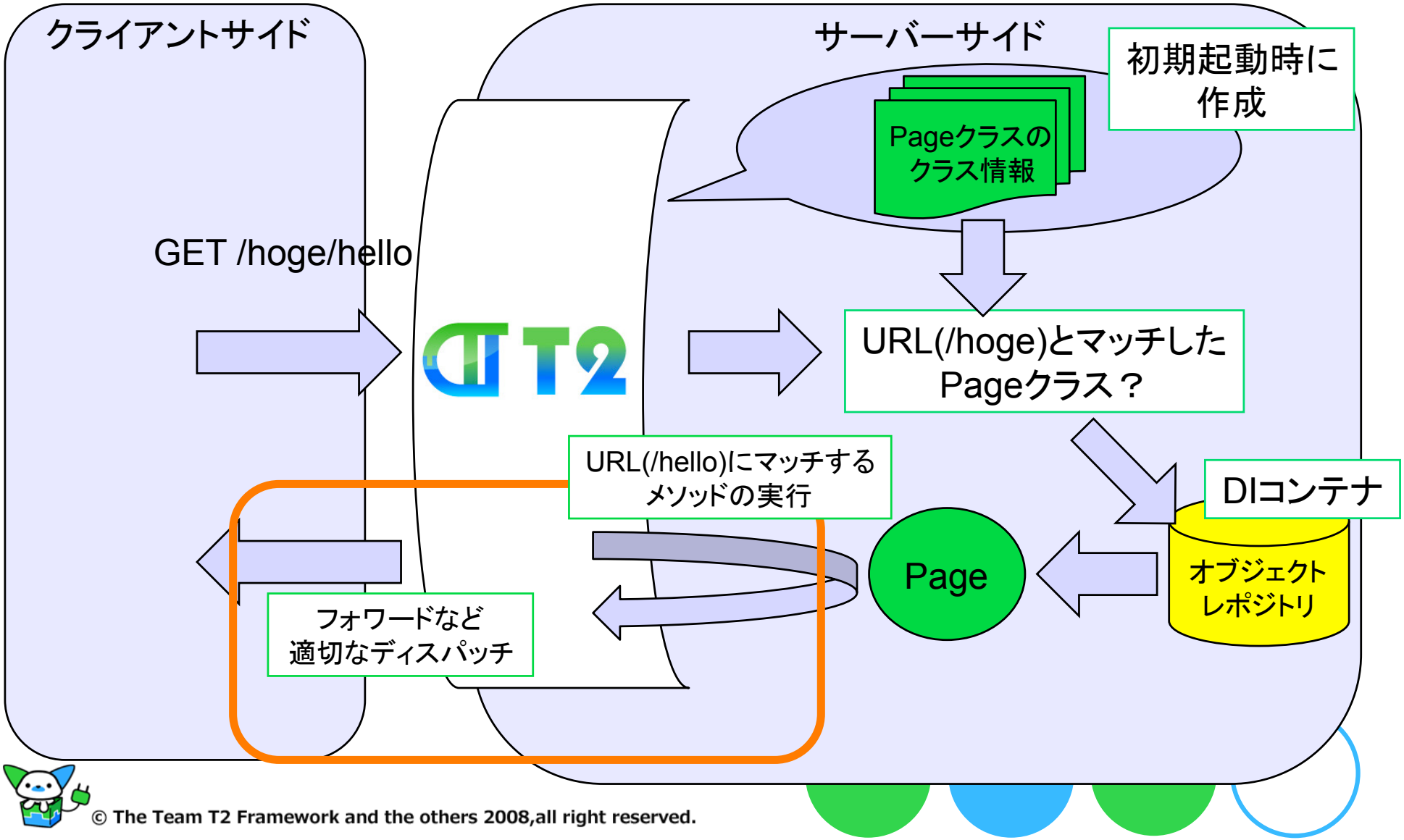


引数アノテーションのサンプル

- サンプルはこんな感じです

```
 * {@.ja @Form は、サブミットされたFORMの値全てをJavaのオブジェクトにマッピングしてもらうためのアノテーションです.
 * 型があわないなどの変換エラーが発生した場合、その値はnullのままになります.@Formで変換エラーが出たときの情報は{@code
 * ErrorInfo}によって 取得することが可能です. @Form はバージョン0.4からの追加機能です.
 *
 * @POST と @ActionParam については、上記addメソッドをご覧ください.}
 *
 * @since 0.4
 */
@POST
@ActionParam
public Navigation addWithForm(@Form AddForm dto, WebContext context,
        ErrorInfo errorInfo) {
    Request request = context.getRequest();
    RequestUtil.save("arg1", request);
    RequestUtil.save("arg2", request);
    if (errorInfo.hasError()) {
```

T2の仕組み（遷移処理）



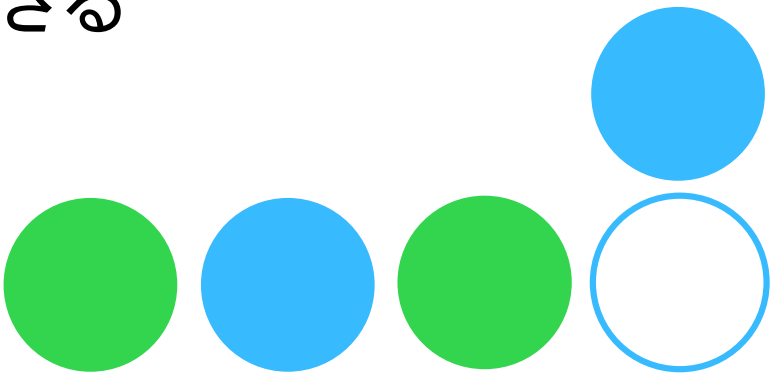
遷移先の決定

- Pageクラス、メソッド、その引数が確定したら、メソッドを実行する
  - T2のメソッドは戻り値はNavigation
  - Navigationはどのクラスかで遷移方法を表現
  - Navigationは遷移先を引数で渡す

```
Forward.to(“/hoge/next.jsp”);
Redirect.to(“/index.jsp”);
Direct.from(file);
```

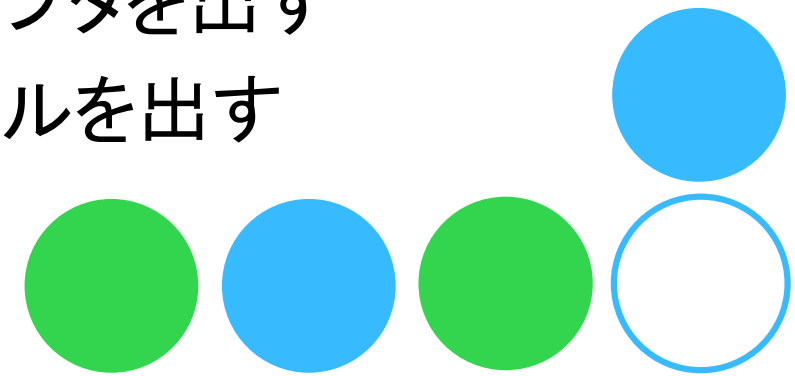
拡張機能

- プラグイン機能
  - 実行タイミングの割り込みと、処理の差し込み
    - 初期化メソッドとか
    - インタセプタしかけるとか
- メソッドアノテーションと引数アノテーションの処理クラスの差し引き
  - 自前のアノテーションとその振る舞いを足せる
  - 既存の使わない物を省略できる
- 例外ハンドラ



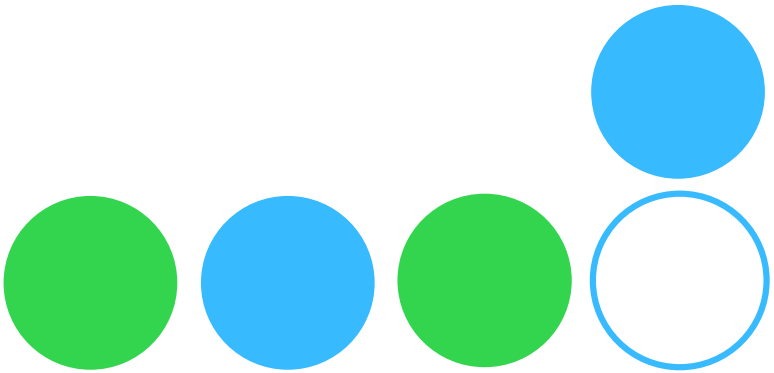
T2 0.5の新機能

- Ajax対応
  - ほとんどのフレームワークがAjaxなリクエストを判別できない
    - XmlHttpRequest/HttpRequestは分けるべき
  - T2はヘッダをみて、判別する
- T2 Extensionモジュール
  - コンテナアダプタやプラグインのプロジェクト
  - 次のリリースで、Guiceアダプタを出す
  - T2+Spring+iBatisなサンプルを出す
  - 自動ITツールの提供



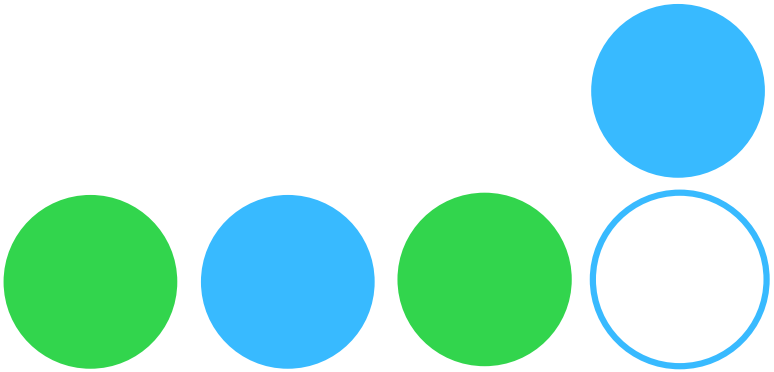
T2 0.6の新機能

- Flex/AIR対応
  - AMFによる高速なバイナリデータのやりとり
  - リッチクライアントをきちんとサポートする



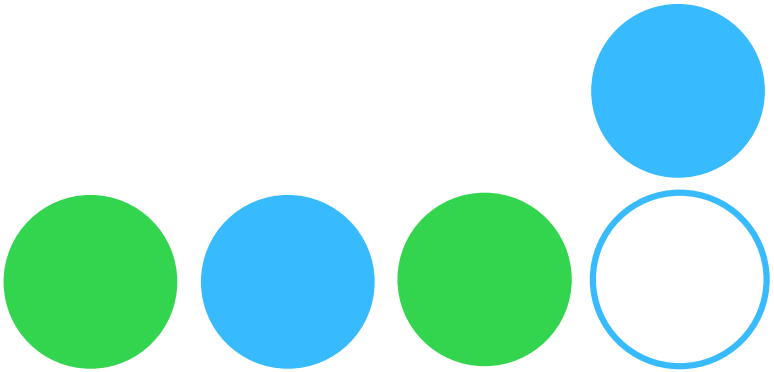
サンプルアプリケーション

- アプリの種類大きく分けて2つあります。
  - ショウケース
    - 機能をひとつづつ紹介するデモ
    - JavaDocでドキュメントを埋め込んでいます
  - サンプル
    - 機能紹介でなく、アプリケーションとしてのより現実路線
    - イントラアプリである、従業員管理アプリ
    - Webサービス系なニュースメーカー
    - などなど



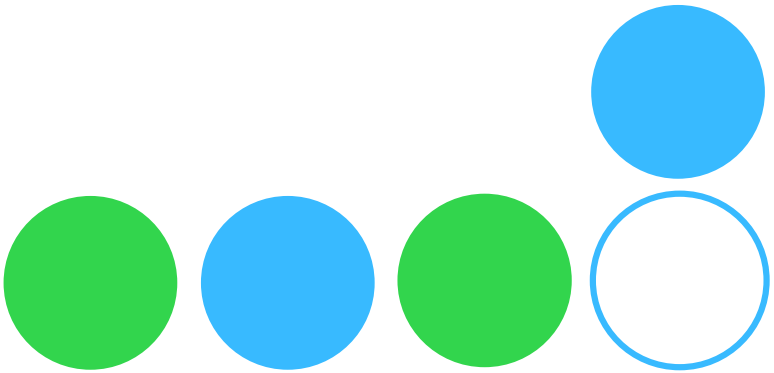
デモ

- 従業員管理アプリ
  - jQuery + T2 + Seasar2 + S2Dao
  - 機能的にも徐々に追加します



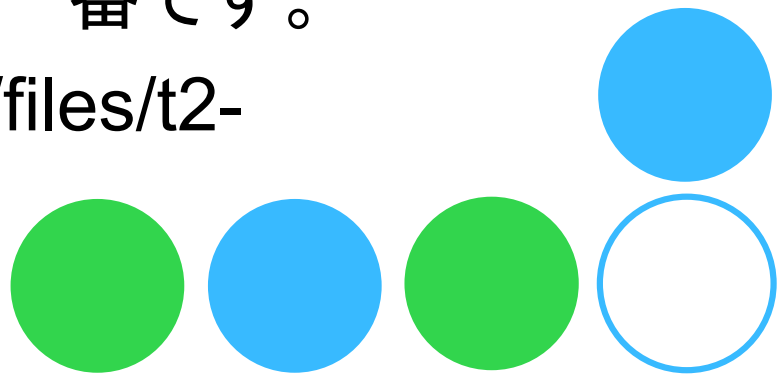
Lucy

- 軽量DIコンテナ
- 本日は割愛！
  - 来年どこかで作ってわかるDIコンテナ勉強会をやりたい。
  - Lucyもそのときにでも。



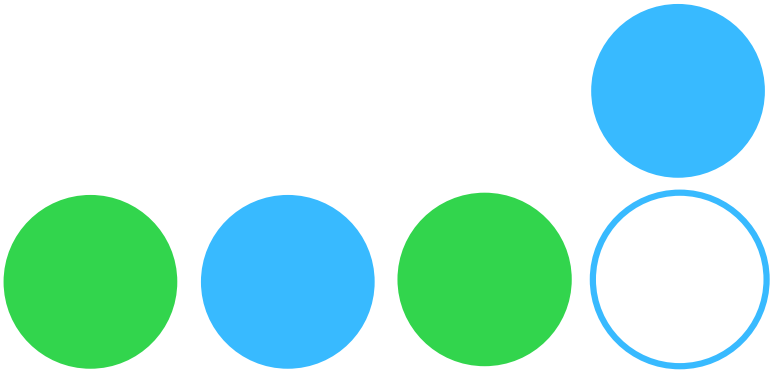
ガイド、ドキュメントなど

- T2ユーザーガイド
  - http://t-2.googlecode.com/files/T2_UserGuide_Japanese%28ver0.4%29.pdf
- T2ユーザーML
  - http://groups.google.com/group/t2-users
- T2の機能がさらに知りたければ・・・
  - T2 ショウケースをみるのが一番です。
  - http://t-2.googlecode.com/files/t2-samples-0.4.0.zip



まとめ

- T2をご紹介しました^o^
  - シンプルな部品化指向Webフレームワーク
  - アノテーションベース
  - ステートレスなフレームワーク
  - Web2.0的なものを当初から考慮



ありがとうございました