第二回ドン引き勉強会

Phobos Code Readings

2006年12月12日(火)

びぎねっと@渋谷

高井 直人 <takai@recompile.net>

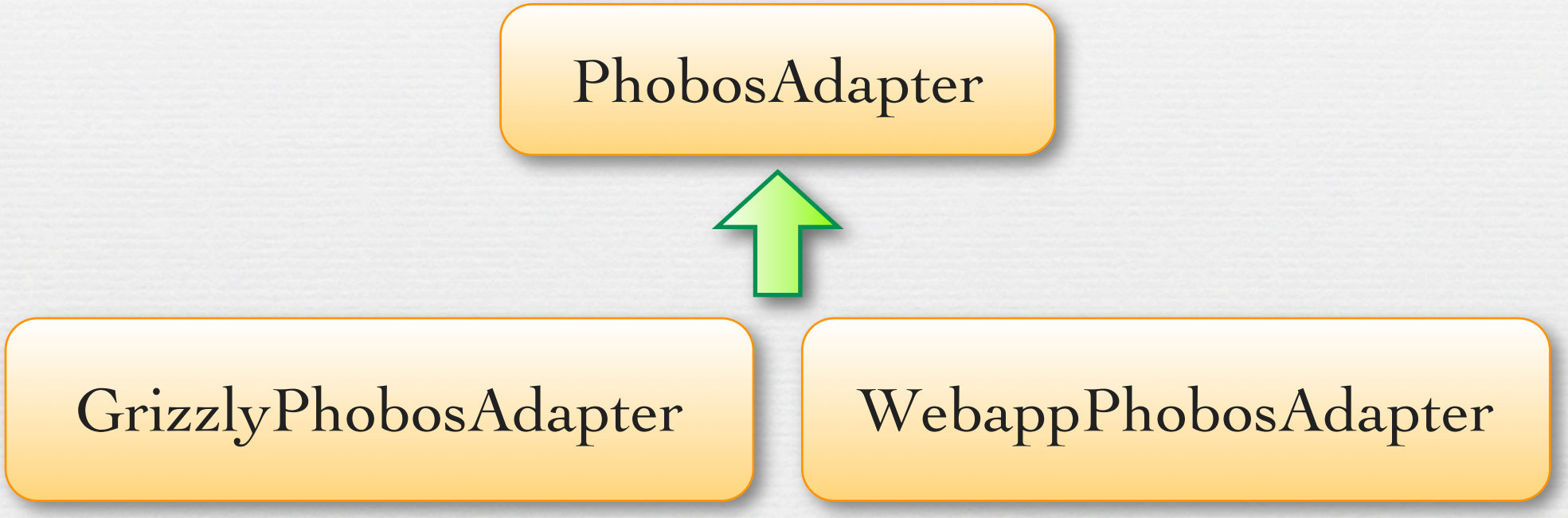
The Goal

- 単体動作可能なウェブアプリケーションの仕組み
  - どのような抽象化を行なっているのか？
- スクリプティングフレームワークの組込み
  - Javaとスクリプト言語との橋渡しは？

What's Phobos

- ウェブアプリケーションフレームワーク
  - 単体でも実行可能
    - Grizzlyを利用
  - スクリプティングフレームワークの組み込み
    - Ruby on Railsを意識

PhobosAdapter



PhobosAdapterを継承

PhobosAdapter

- GrizzlyPhobosAdapter
  - org.apache.coyote.Adapterの実装
  - Grizzlyから呼ばれる
- WebappPhobosAdapter
  - PhobosServletから呼ばれる

```
public abstract class
    PhobosAdapter<RequestType, ResponseType>
{
   public void service(RequestType request,
                       ResponseType response)
   {
```

Coyote

```
public class GrizzlyPhobosAdapter
    extends PhobosAdapter<Request, Response>
                    implements Adapter
{
```

Servlet

```
public class WebappPhobosAdapter
   extends PhobosAdapter<HttpServletRequest,
                         HttpServletResponse>
{
```

GrizzlyPhobosAdapter

```
public class Main extends AbstractMain {
...
    public void start() {
...
        selectorThread = new SelectorThread();
...
        selectorThread.setAdapter(adapter);
        Thread thread = new Thread() {
            public void run() {
                try {
                    selectorThread.initEndpoint();
                    selectorThread.startEndpoint();
```

WebappPhobosAdapter

```
public class PhobosServlet extends HttpServlet {
...
    public void service(HttpServletRequest request,
                        HttpServletResponse response)
      throws IOException, ServletException {
...
        WebappPhobosAdapter adapter =
            (WebappPhobosAdapter)
                servletContext.getAttribute(ADAPTER);
        adapter.service(request, response);
```

Initializing Adapter

- PhobosAdapter#startup(String baseDir)
  - ScriptingServiceの初期化
  - ResourceServiceの初期化
  - PhobosRuntimeで利用するための
    グローバルコンテキストの設定
  - フレームワーク用スクリプトの実行

PhobosAdapter Flow

- PhobosAdapter#service(RequestType, ResponseType)
  - リクエスト、レスポンスオブジェクトをラップ
  - スクリプト(/framework/service.js)を実行

総称型

```
public void service(RequestType request,
                    ResponseType response)
{
    RequestWrapper<RequestType> requestWrapper
        = createRequestWrapper(request);
    ResponseWrapper<ResponseType> responseWrapper
        = createResponseWrapper(response);
...
}
protected abstract RequestWrapper<RequestType>
    createRequestWrapper(RequestType request);
protected abstract ResponseWrapper<ResponseType>
    createResponseWrapper(ResponseType response);
```

抽象
メソッド宣言

RequestWrappter

```
public abstract class RequestWrapper<T> {

    protected RequestWrapper(T request);
    public T unwrap();
    public Object getRequestObject();

    public abstract String getRequestURI();
    public abstract Enumeration getHeaderNames();
    public abstract String getHeader(String name);

    protected T request;
}
```

ResponseWrapper

```
public abstract class ResponseWrapper<T> {

    protected ResponseWrapper(T response);
    public T unwrap();
    public Object getResponseObject();

    public abstract void setStatus(int code);
    public abstract void setContentType(String contentType);
    public abstract PrintWriter getWriter() throws IOException;
    public abstract boolean isCommitted();
    public abstract void reset();

    protected T response;
}
```

Running Script

- ScriptEngineの取得
- リクエストコンテキストの設定
  リクエスト、レスポンス、その他情報
- PhobosRuntimeの設定
  スレッドローカルなコンテキスト
- スクリプト(/framework/service.js) を評価

Conclusion

- Adapterを利用して、さまざまな環境でのウェブアプリケーションフレームワーク実行を実現
- JavaScriptで実装された部分は、かなり自前で処理を行なっている
- きれいな抽象化がされているとはいえ、スクリプト言語の助けがなければ不十分かも