A Service Based Approach

WS Slides #4

Characterization Review

Application composed of language/platform independent resources

Possible a mashup

Current trend: RESTful single page application using HTML5 is very hot

Putting it Together

Finally need some design

- The resources
- How to distribute the responsibilities, Client vs Server
- MVC

Also

- Separation of concerns (many techniques, “languages” involved)

Issues

- Quality … testing?
- Authorization (optional)

Designing a RESTful Service

Identification of resources

- Which URLs to which resources? Which methods?
- Hard, similar to OO design (if unlucky we end up with a bad design)

Example: Web Service for books and music

```
ex.org/review?type=cd&title=help
ex.org/cd-review?title=help
ex.org/review/cd?title=help
ex.org/review/cd/help/beatles
GET seems ok, but what about PUT, POST?
```

The debate: No verbs in URI's (resources are not operations)? Operations are GET, POST, ...)

Technical View of Serviced Based Approach

In request based approach “everything” on server side (except DOM tree and CSS rendering)

Service Based Approach

- Model still on server but …
- … much more client side; View rendering (manipulate DOM tree inside browser), parts of control and some “representation” of model objects (using JavaScript)

Separation of concerns

Unobtrusive JavaScript =
HTML + CSS + JavaScript in a disciplined manner

- No JavaScript, CSS in HTML
- Style/layout in CSS
- JavaScript in *.js files
- Event handling setup in JavaScript after page load

We always use unobtrusive style!

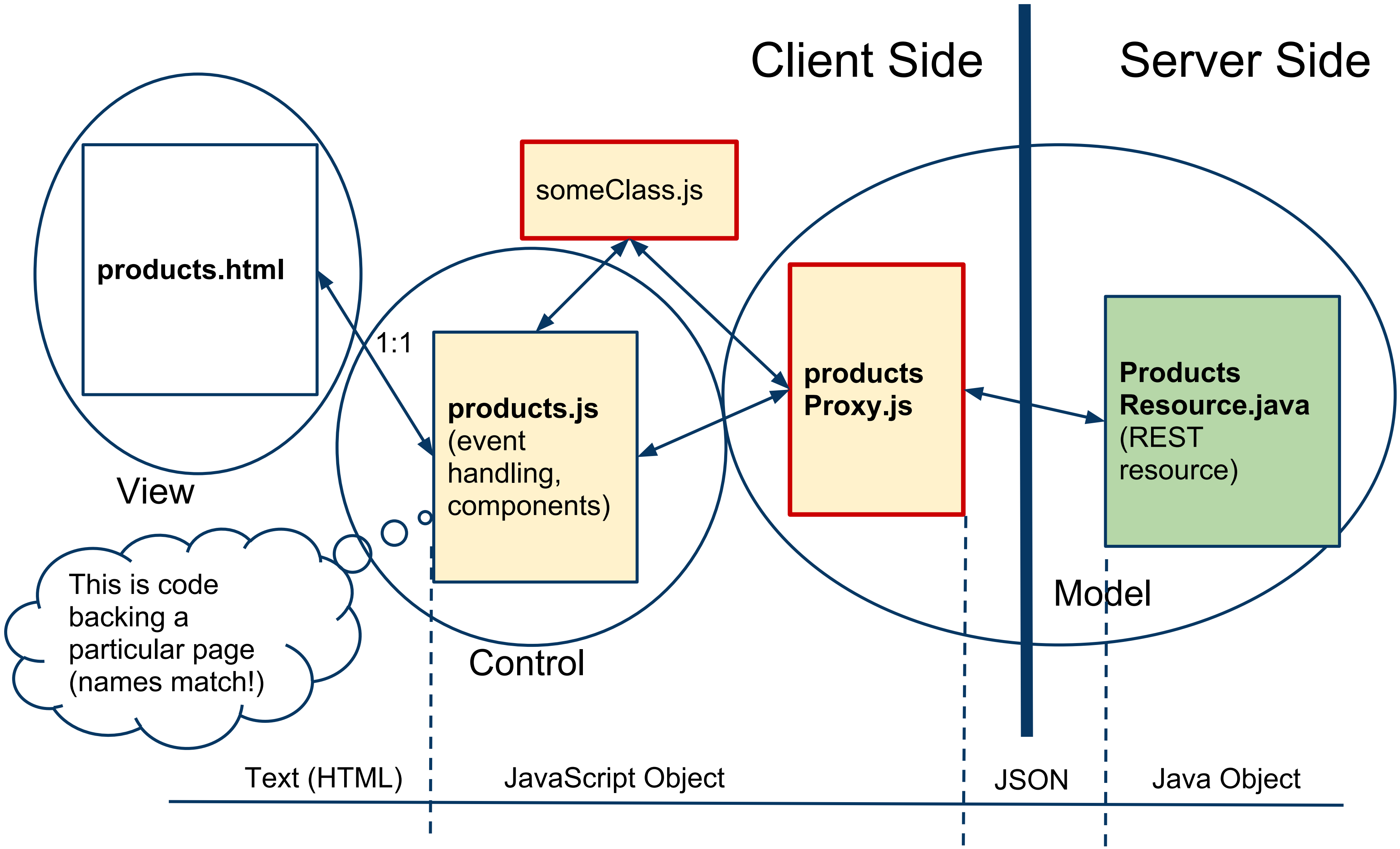
JavaScript MVC

Easy to create a "big ball of mud", mixing event handling, DOM manipulation, application logic.

Separate JS code for
- DOM manipulating (prefer "components")
- Event handling, listeners (page backing code)
- Services (often proxies)

Normally using some framework, **backbone.js** … many more
- No time for frameworks, use in house MVC design

In House MVC Design


Client Side
Server Side
products.html
View
someClass.js
1:1
products.js (event handling, components)
Control
products Proxy.js
Products Resource.java (REST resource)
Model
This is code backing a particular page (names match!)
Text (HTML)
JavaScript Object
JSON
Java Object

Red borders = Pseudo classical + module pattern. Arrows are calls and returns

Java to JavaScript Strategy

We use pseudo classical style as much as possible
- I.e. all functions as methods

Code backing a particular HTML page will use standalone functions or singleton objects (setting up event handling …)

So … we try to do pure OO but in some well defined cases there are freestanding function

JavaScript Testing

JS development heavily dependent on testing, normally need large test suites

Many possibilities we use **QUnit,** http://qunitjs.com/

- See workshop skeleton code for use
- Need a library

REST Authorization problems

Stateless

- REST should be stateful, can't store credentials (name/password) in session
- Solution: Send something (time limited) <u>representing</u> the credentials with each request (very bad to send credentials)

Mashup applications

- The application should be able to call other applications (possible on behalf of user, possible restricted calls)
- Must have <u>single login</u> for user

A proposed solution is **OAuth**

OAuth Confusion

Situation seems very confusing

- Versions: OAuth 1.0 (1.0A a revision of 1.0) and OAuth 2.0 (not backward compatible).
- Are spec <u>really</u> finished (?)
- Spec leader (and others) resigned from OAuth 2.0
- Some API's use 1.0 (Twitter) some use 2.0 (Facebook, Google, Twitter)
- Marked as simple to use ... at least not for a newbie...

Introduction OAuth 1.0 http://hueniverse.com/oauth/guide/workflow/

Authorization using Twitter4J

Twitter4J is a Java client-API adapting Twitters REST API

- Here we'll use the Twitter REST API 1.1 and "Twitter for Websites"

Before use of REST API a bit of setup

- Need account on Twitter
- Need to register your application with Twitter
- Must use OAuth for authorization (we use Twitter to login to our application, if success application can call the Twitter API)
- Our web application is a client of Twitter

Twitter OAuth Flow

Using Twitter4J

1. When application registred we get a **consumer key** and a **consumer secret** (on registration page)
2. Use these in application to get a **request token** (store for later use)
3. Redirect to; requestToken.getAuthenticationURL(). Will show the Twitter login page
4. After login. Twitter will callback to application (we supply an URL)
5. At callback, extract parameter "oauth_verifier"
6. Use request token and parameter to get an **access token** (store for later use)
7. Use access token when calling API (each call)

How to Run a JEE Service

There are many Java EE containers capable of running JAX-RS

As noted before we prefer GlassFish (because it defaults support JAX-RS)