



中山大學  
SUN YAT-SEN UNIVERSITY

# Lecture 06.

## Multiple Pages Web Application

**SE-386 Software Process Improvement**

<http://my.ss.sysu.edu.cn/wiki/display/SPSP>

[ericwangqing@gmail.com](mailto:ericwangqing@gmail.com)

School of Software, Sun Yat-sen University

# Outline

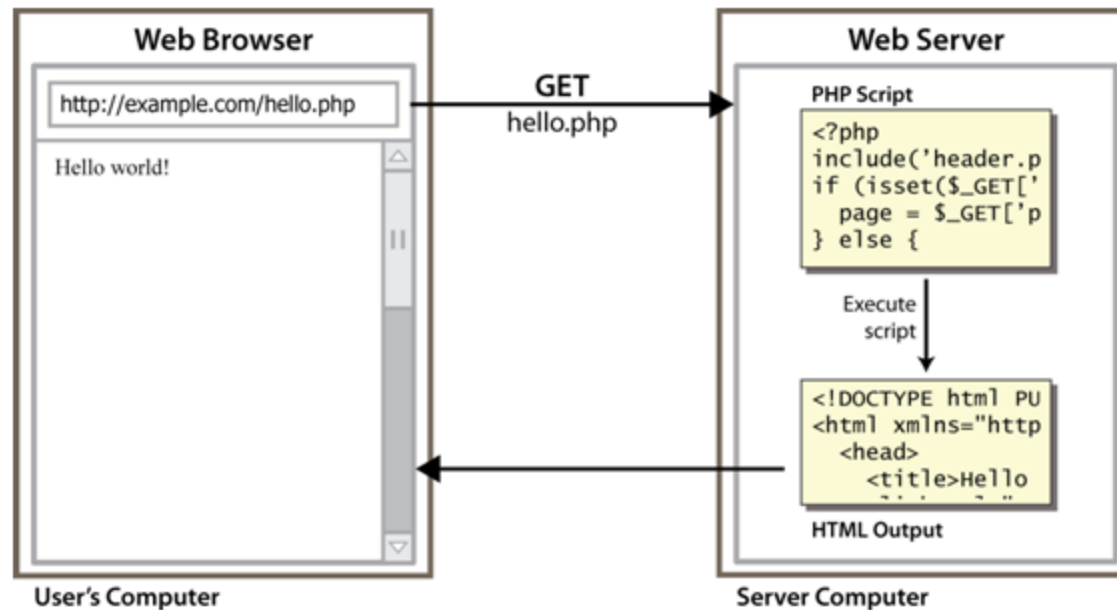
---

- **Multiple Page Appliation**
- NodeJS
- ExpressJS
- Code in Practice

# Mechanism of browsers



# Multiple Page Applications



- An application is composed by a set of pages (urls)
- Server generate a page based on:
  - application (global) state
  - session
  - request parameters (query string, post body)

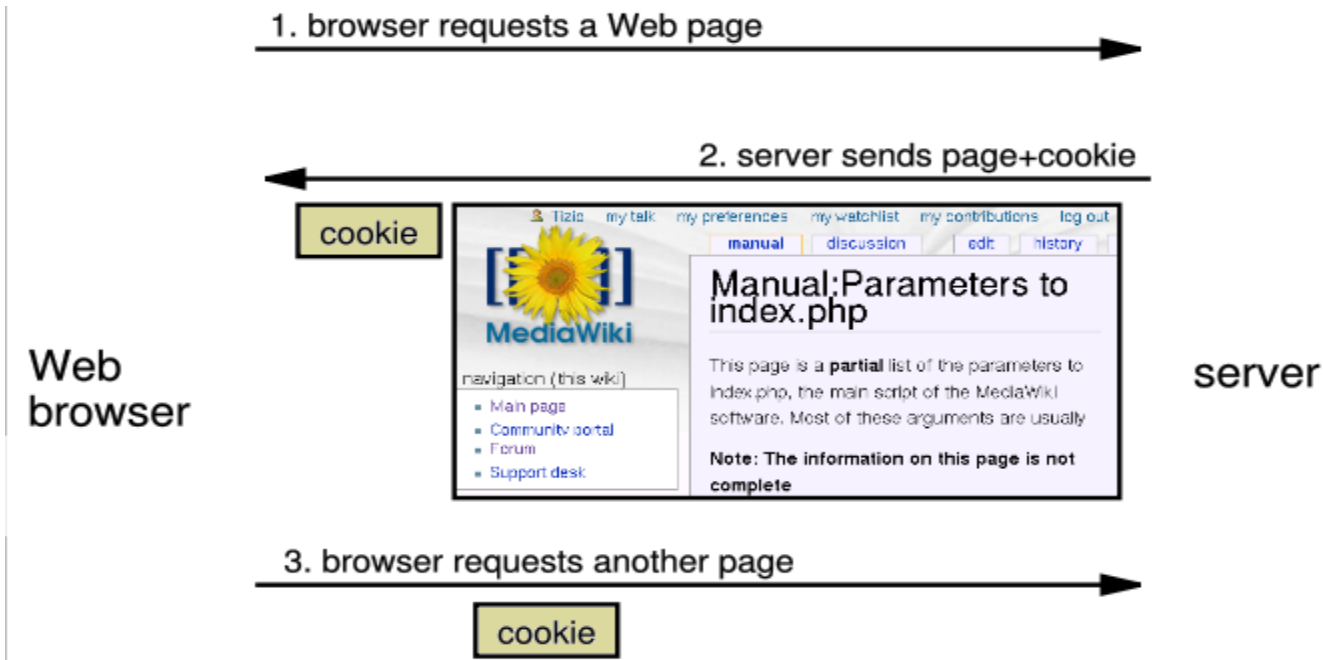
# What is a Cookie?

---

- cookie: a small amount of information sent by a server to a browser, and then sent back by the browser on future page requests
- cookies have many uses:
  - authentication
  - user tracking
  - maintaining user preferences, shopping carts, etc.
- a cookie's data consists of a single name/value pair, sent in the header of the client's HTTP GET or POST request



# How cookies are sent



- when the browser requests a page, the server may send back a cookie(s) with it
- if your server has previously sent any cookies to the browser, the browser will send them back on subsequent requests
- alternate model: client-side JS code can set/get cookies

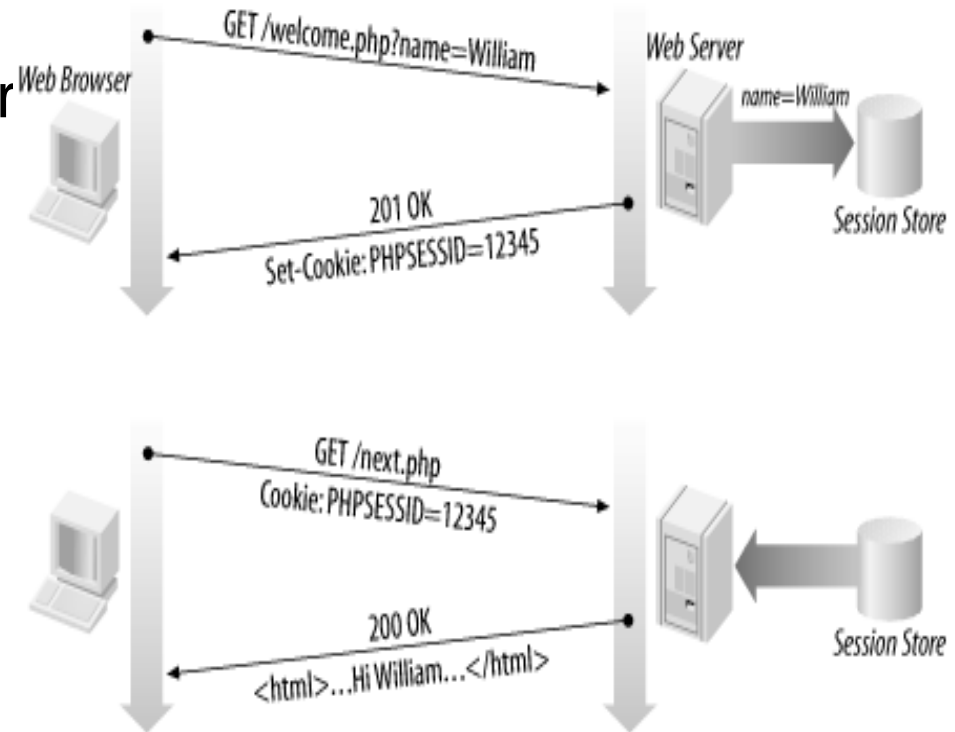
# What is a session?

---

- **session**: an abstract concept to represent a series of HTTP requests and responses between a specific Web browser and server
  - HTTP doesn't support the notion of a session, but PHP does
- sessions vs. cookies:
  - a cookie is data stored on the client
  - a session's data is stored on the server (only 1 session per client)
- sessions are often built on top of cookies:
  - the only data the client stores is a cookie holding a unique **session ID**
  - on each page request, the client sends its session ID cookie, and the server uses this to find and retrieve the client's session data

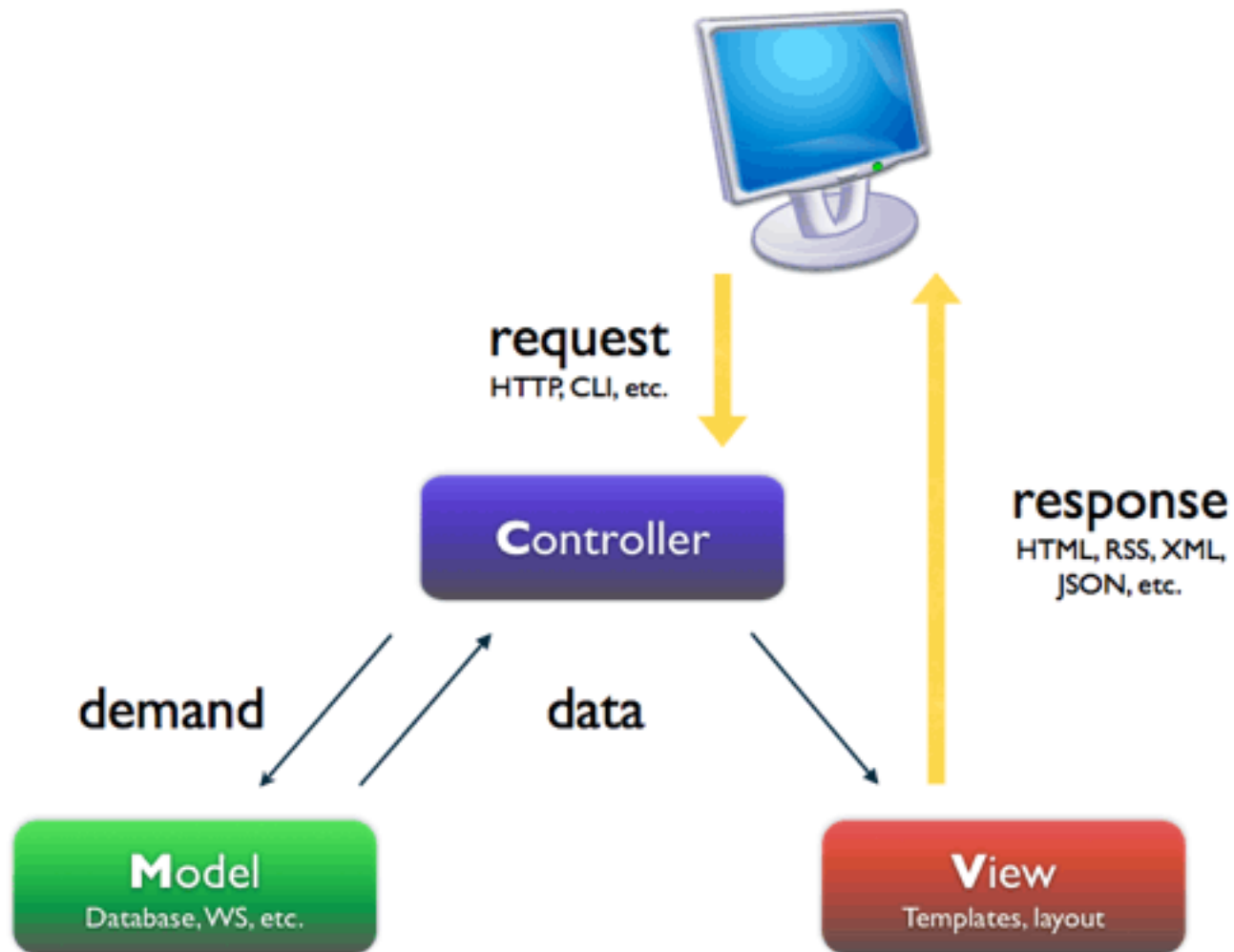
# How sessions are established

- client's browser makes an initial request to the server
- server notes client's IP address/browser, stores some local session data, and sends a session ID back to client
- client sends that same session ID back to server on future requests
- server uses session ID to retrieve the data for the client's session later, like a ticket given at a coat-check room

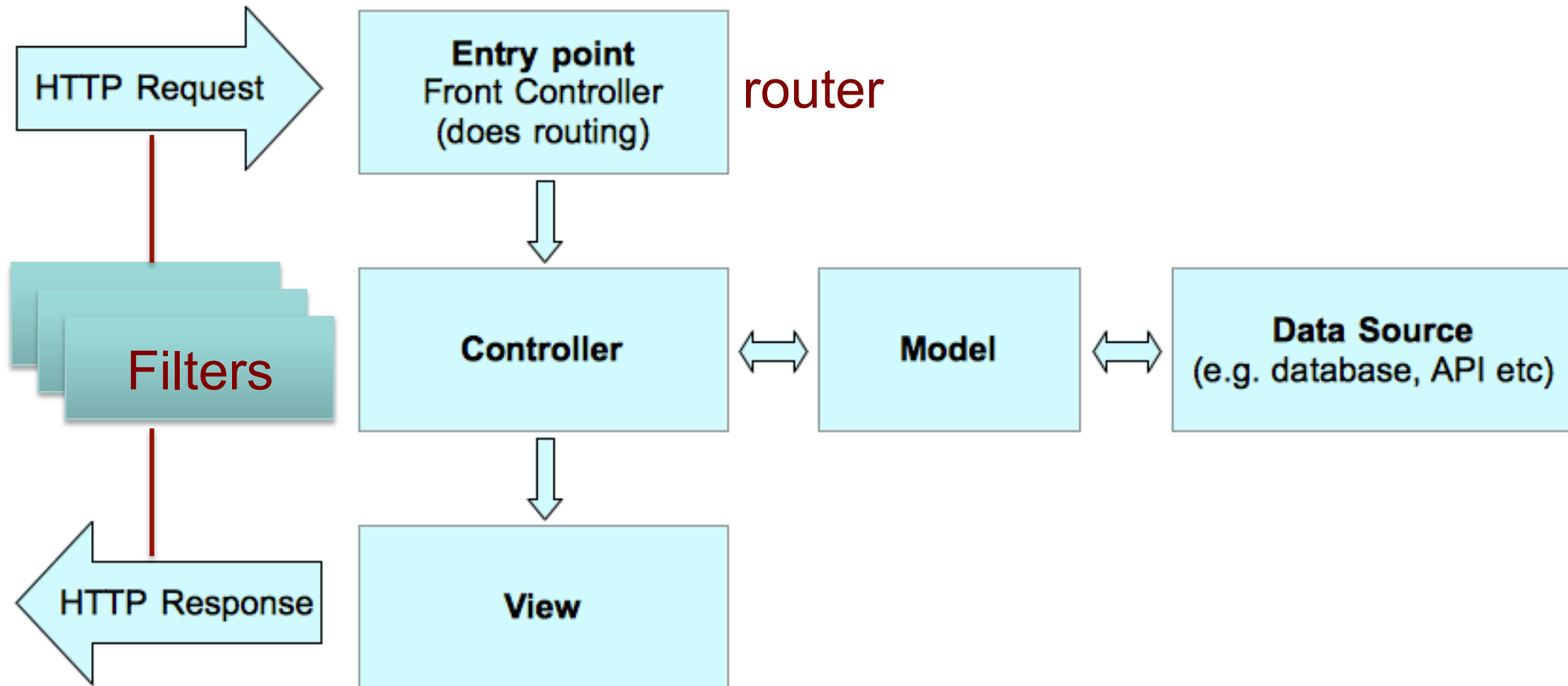




# MVC



# MVC + Front Controller (Router) + Filters



# Outline

---

- Multiple Page Appliation
- **NodeJS**
- ExpressJS
- Code in Practice

# Node.js

---

nodejs.pdf

# Outline

---

- Multiple Page Appliation
- NodeJS
- **ExpressJS**
- Code in Practice

# ExpressJS

<http://expressjs.com/>

## Express

Fast, unopinionated, minimalist  
web framework for **Node.js**

```
$ npm install express --save
```

Express docs available in other languages: [Spanish](#), [Japanese](#), [Russian](#), [Chinese](#).

### Web Applications

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

### APIs


With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy.

### Performance

Express provides a thin layer of fundamental web application features, without obscuring Node features that you know and love.

### LoopBack

Develop model-driven apps with an Express-based framework. Find out more at [loopback.io](http://loopback.io).

 Star 18,245

 The Express project is sponsored by [StrongLoop](#).  [Fork the website on GitHub](#)

# Building Web Apps with Express

By Aaron Stannard

Startup Developer Evangelist, Microsoft Corporation

# What is Express?

---

- Sinatra-inspired MVC framework for Node.JS
- Built on Connect Middleware
- Minimalist

express



# What Express Does

---

- Parses arguments and headers
- Routing
- Views
  - Partials
  - Layouts
- Configuration
- Sessions

# Simple Express App

---

```
var express = require('express');  
var app = module.exports = express.createServer();
```

Loads Express module

Instantiates Express  
server

```
app.configure(function() {  
  app.set('views', __dirname + '/views');  
  app.set('view engine', 'jade');  
  app.use(express.bodyParser());  
  app.use(express.methodOverride());  
  app.use(app.router);  
  app.use(express.static(__dirname + '/public'));  
});
```

Global Configuration

```
// Routes  
require('./routes/site')(app);
```

Loads and binds routes  
(defined in separate  
module)

```
app.listen(process.env.PORT);  
console.log("Express server listening on port %d in %s mode",  
  app.address().port, app.settings.env);
```

# Getting Started with Express

---

- Installing Express

```
C:\> npm install express
```

```
C:\> npm install express -g
```

- Creating Express Applications

```
C:\> express [new project name]
```

```
C:\> cd [new project name]
```

```
C:\[new project name]> npm install -d
```

```
C:\[new project name]> node app.js
```

```
Express server listening on port 3000 in development mode
```

# Express Project Structure

---

<b>/projectroot/</b>	
package.json	Tells Node and NPM what packages are required
readme.txt	
<b>web/</b>	Root of your actual application
app.js	
<b>views/</b>	The main entry point for the Express application
layout.jade	
index.jade	
<b>models/</b>	Data model
post.js	
<b>public/</b>	Static content
images	
stylesheets	
scripts	
<b>test/</b>	Directory for unit tests
route-test.js	
post-test.js	
<b>node_modules/</b>	Output directory for all NPM installations

```
C:\[projectroot]> node web/app.js
```

```
Node server listenening on port 3000 in development mode
```

# Express Configuration (app.js)

---

```
app.configure(function(){  
  app.set('views', __dirname + '/views');  
  app.set('view engine', 'jade');  
  app.use(express.bodyParser());  
  app.use(express.methodOverride());  
  app.use(express.cookieParser());  
  app.use(sessions({secret: 'adfasdf34efsdffs34sefsdf'}));  
  app.use(app.router);  
  app.use(express.static(__dirname + '/public'));  
});
```

Global configuration setter  
View Engine and Views directory  
Session Key  
Router and Static Content

Development-environment configurations

```
app.configure('development', function(){  
  app.use(express.errorHandler({ dumpExceptions: true, showStack: true }));  
});
```

Production-environment configurations

```
app.configure('production', function(){  
  app.use(express.errorHandler());  
});
```

# Routing

---

```
//Catch-all  
app.all('/app(/*)?', requiresLogin);
```

Catch-all – works for all HTTP verbs

```
// Routes  
app.get('/', routes.index);  
app.get('/about', routes.about);  
app.get('/contact', routes.contact);  
app.get('/app/list', routes.listapps);  
app.get('/app/new', routes.newapp);  
app.post('/app/new', routes.saveapp);  
app.get('/app/:app', routes.getapp);  
app.get('/app/:app/edit', routes.editapp);
```

HTTP GET request

HTTP POST request

Accepts :app route argument

Syntax follows the pattern:

App.[verb](path, function(req,res), [function(req,res)]);

# Creating Route Modules (Style 1)

---

## server.js

```
var express = require('express')  
    , routes = require('./routes');
```

...

```
app.get('/', routes.index);
```

## route.js

```
// GET the homepage  
exports.index = function(req, res){  
    res.render('index', { title: Home' });  
};
```

# Creating Route Modules (Style 2)

---

## server.js

```
// Routes  
require('./routes/site')(app);
```

## routes/site.js

```
/*  
 * Module dependencies  
 */  
  
module.exports = function (app) {  
  
  // GET home page  
  app.get('/', function (req, res) {  
    res.render('index', { title: 'Home' })  
  });  
}
```



# Request Object

---

- Req.Param
  - Req.Query
  - Req.Body
- Req.Session
- Req.Flash
- Req.Headers
- Can bind usable JSON payloads

# Response Object

---

- Res.Redirect
- Res.Write
- Res.End
- Res.Download
- Local variables and object binding

# Route Pre-Conditions

---

```
app.param('app', function(req, res, next, id){
  appProvider.getAppById(req.session.userName, id,
    function(error, app){
      if(error) return next(error);
      if(!app) return next(new
                                Error('Failed to find
app with                        id '+ id));
      req.app = app;
      next();
    });
});
```

# Route Filters

---

```
//Catch-all
app.all('/app(/*)?', function(req, res, next) {
  if(req.session && req.session.userName) {
    next();
  } else {
    res.redirect('/login?redir=' + req.url);
  }
});
```

# Views

---

- Support for multiple view engines
- Layout support
- Partials
- Dynamic Helpers

# Jade

---

- Basic Syntax

```
ul
  li
    a(href='/') Home
```

- Variables

```
title= title
```

- Conditionals

```
if flash.info
  span.info= flash.info
```

- Iterations

```
each user in users
  div.user_id= user.username
```

# View Helpers

---

```
app.dynamicHelpers({  
  session: function (req, res) {  
    return req.session;  
  },  
  flash: function (req, res) {  
    if (typeof (req.session) == 'undefined') {  
      return undefined;  
    }  
    else {  
      return req.flash();  
    }  
  }  
});
```

# Session Management

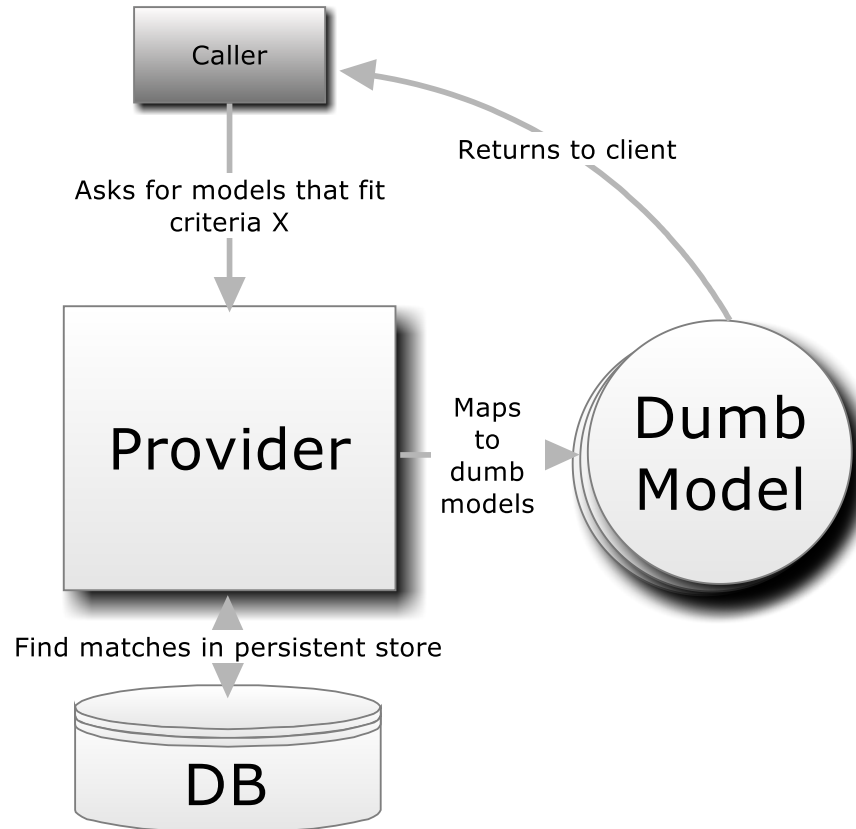
---

- Session State Providers
  - Cookie + Back-end Session Store
- Session Cookies
  - cookie-sessions NPM package

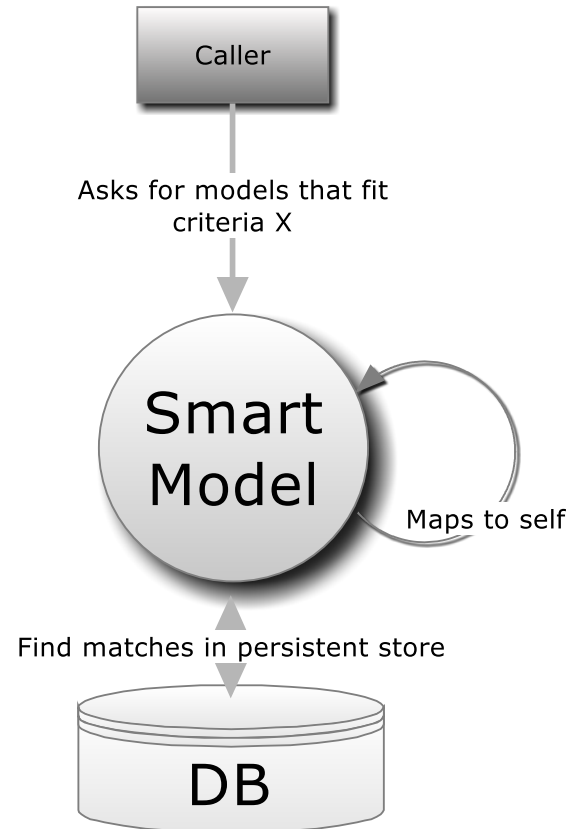
```
//Assign username of logged in user to session  
req.session.userName = username;
```



# Model Structure (OOP)



# Model Structure (Common)



# Express on Windows Azure

---

- Rename app.js to server.js
- (You're done)
- Caveats
  - Only a limited number of session state providers
  - Many popular data stores not available

# Further Reference

---

- [ExpressJS.com](http://ExpressJS.com) - Official Express Homepage
- [Github.com/visionmedia/jade](https://github.com/visionmedia/jade) - Jade
- [Github.com/senchalabs/connect](https://github.com/senchalabs/connect) - Connect
- [Github.com/WindowsAzure](https://github.com/WindowsAzure) – Azure bits

# Outline

---

- Multiple Page Appliation
- NodeJS
- ExpressJS
- **Code in Practice**



## Lab 06. MyHomework

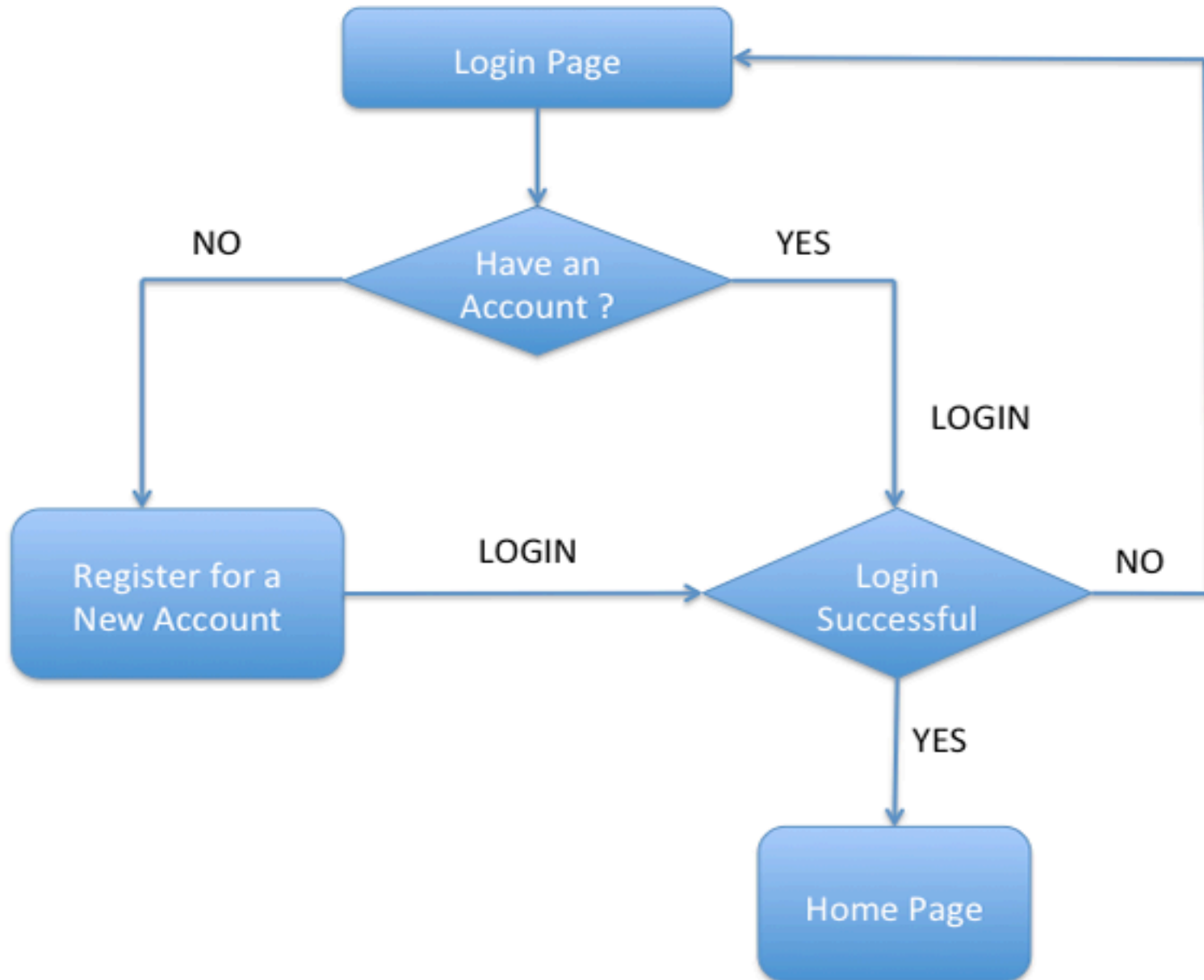
🔗 7 Added by 王青, last edited by 王青 on Apr 08, 2015 ([view change](#))

# 简介

**MyHomework** 是一个基于ExpressJS的Web 2.0应用，老师可以发布作业，学生可以提交作业。

1. 角色: 学生，老师。
2. 访问管理:
  - a. 只有选定了本课程老师和学生才能够访问使用本系统。
  - b. 老师可以看到所有的作业要求和所有学生提交的作业。
  - c. 学生能看到所有的作业要求，但只能看到自己的作业。
3. 发布作业要求: 老师可以发布作业要求，也可以修改一个已发布但是尚未截止的作业要求。
4. 提交作业: 学生可以提交作业（可以多次提交作业，系统将保留最新的版本）。
5. **deadline**: 老师可以设定/修改作业要求的截止时间，截止时间到达后，任何学生都将无法提交作业。
6. 作业评分: 截止时间到达之后，老师可以批改作业给出分数。

# MyHomework



<http://passportjs.org/>

# Passport

Simple, unobtrusive authentication for Node.js.

## About Passport

Passport is authentication middleware for [Node.js](#). Extremely flexible and modular, Passport can be unobtrusively dropped in to any [Express](#)-based web application. A comprehensive set of strategies support authentication using a username and password, Facebook, Twitter, and more.

[Read the Guide »](#)

## Features

- 140+ authentication strategies
- Single sign-on with OpenID and OAuth
- Easily handle success and failure
- Supports persistent sessions
- Dynamic scope and permissions
- Pick and choose required strategies
- Implement custom strategies
- Does not mount routes in application
- Lightweight code base

```
$ npm install passport
```

## Community

- [GitHub](#)
- [Twitter](#)
- [Google Groups](#)
- [Stack Overflow](#)
- IRC: [#passport.js](#)

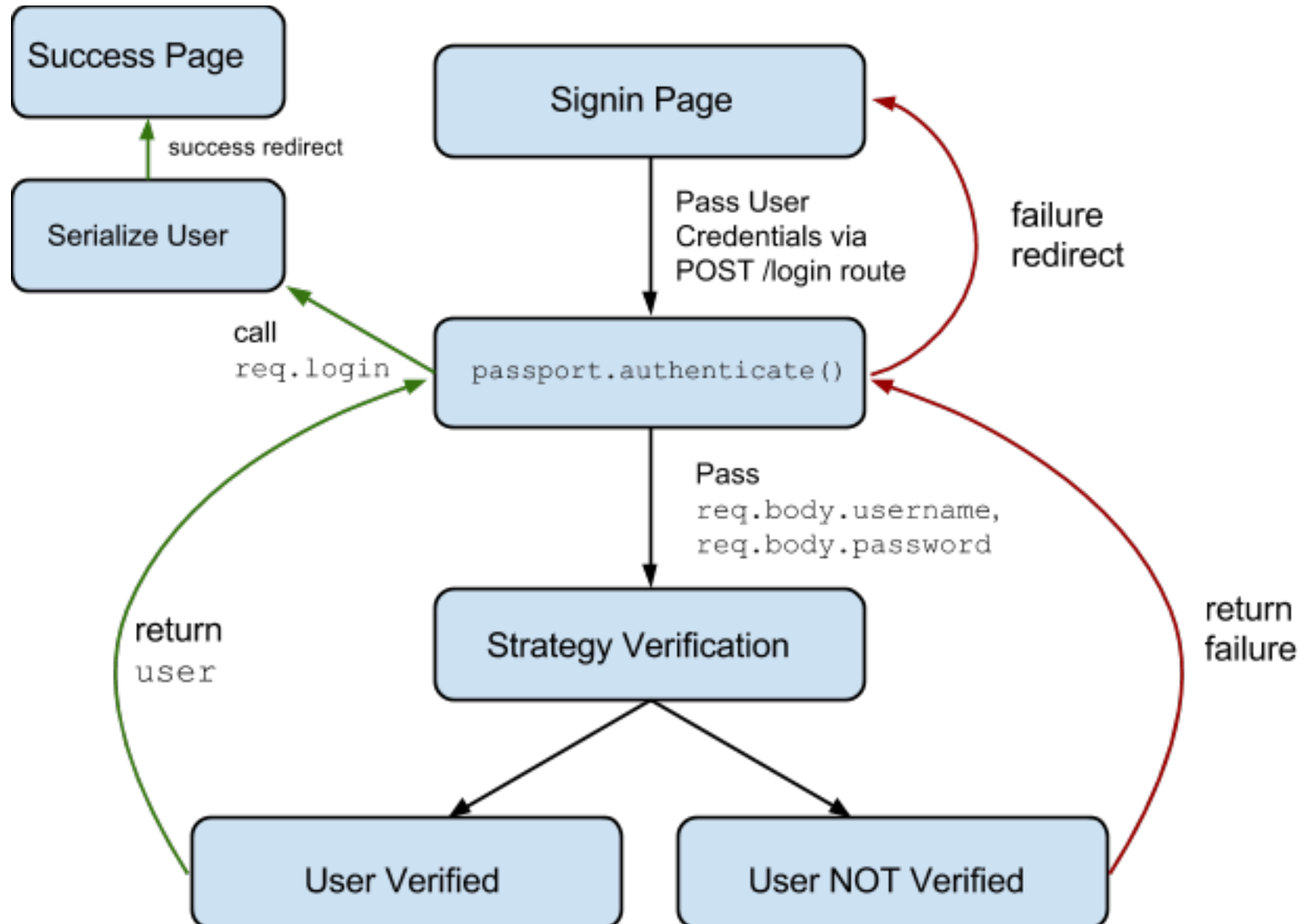
## Also

[Locomotive](#)

Powerful MVC web framework.



# MyHomework



# Thank you!

