Web Application Development

Approaches

Choices

- Server Side

PHP
ASP
Ruby
Python
CGI
Java Servlets
Perl

Choices

- Client Side

Javascript
VBScript
ASP

Language basics - always the same

- Embedding in / outside Web pages
- Whitespace and Line breaks
- Statements and semicolons
- Case sensitivity
- Comments
- Literals
- Identifiers
- Keywords
- Data types

Development approaches

Procedural / imperative style
Object Oriented data structure with imperative style
MVC style
Pre built MVC frameworks
Pre-built platforms

Imperative/Procedural style

```
{$REGION 'Font: PROGGY CLEAN 12' }

procedure MakeBestProgram(
  const what : TObject;
  var how : string);
var
  cnt : integer;
begin
  //loop it
  for cnt := 0 to 100 do
  begin
    if Assigned(Controls[cnt]) then Break;
  end;
end;

{$ENDREGION}
```

Write all the code

Use logical blocks to enhance reuse

Guaranteed to produce the functionality required

Object Oriented Style

- Combine values and processes that act on those values in a single structure

Person
firstname lastname age
set_firstname set_lastname set_age get_firstname get_lastname get_age

- Claimed to be closer to real world structures

History - Object Oriented Programming

Difficult to be precise

PDP-1 at MIT

Sketchpad (Sutherland 63)

Simula 67

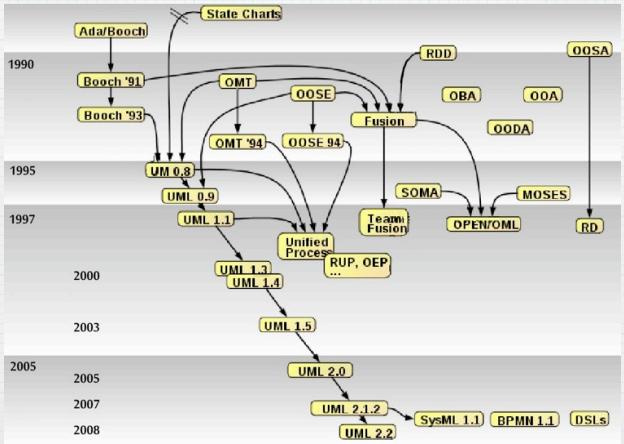
Smalltalk, Xerox PARC by Alan Kay 1970s , introduced the term object-oriented programming

Lisp

Ada, C++ and many other languages

History - Object Oriented Design (UML)

Many different versions and variants



Object Data Model - Characteristics

Encapsulation

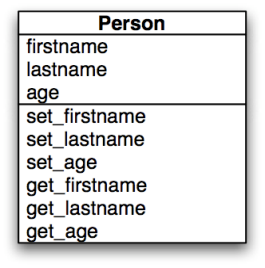
Inheritance

Polymorphism

The design of an object is done in a class

Example:- a Person Class

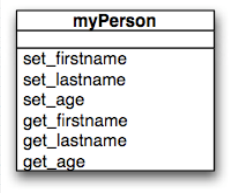
- Three properties
- Six methods



Using the Person class

To use the class, create an Object of that class

```
myPerson = new Person
```

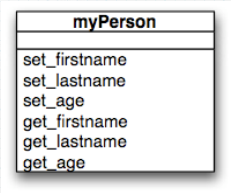


myPerson is an Object of type Person

myPerson is an instance of a Person

Using the Person instance

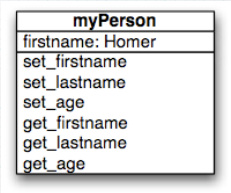
```
myPerson.set_firstname('Homer');
myPerson.set_lastname('Simpson');
myPerson.set_age(38);
```



The set_ methods pass values into the objects properties

Using the Person instance

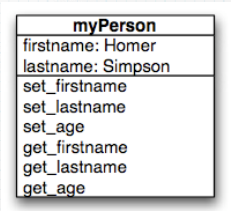
```
myPerson.set_firstname('Homer');
myPerson.set_lastname('Simpson');
myPerson.set_age(38);
```



The set_ methods pass values into the objects properties

Using the Person instance

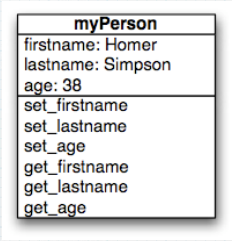
```
myPerson.set_firstname('Homer');
myPerson.set_lastname('Simpson');
myPerson.set_age(38);
```



The set_ methods pass values into the objects properties

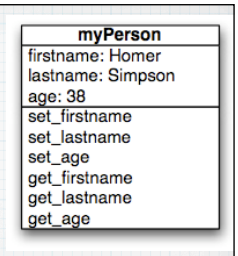
Using the Person instance

```
myPerson.set_firstname('Homer');
myPerson.set_lastname('Simpson');
myPerson.set_age(38);
```



The set_ methods pass values into the objects properties

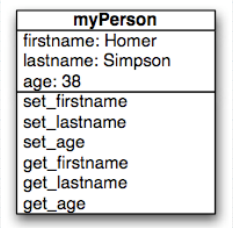
Using the Person instance



The get_ methods get values out of the object

```
print myPerson.get_firstname();      Homer
print myPerson.get_lastname();       Simpson
print myPerson.get_age();            38
```

Encapsulation



You should always access the properties through the methods of the object

```
myPerson.age=42;
```

Don't do this

```
myPerson.set_age(42);
```

Do this

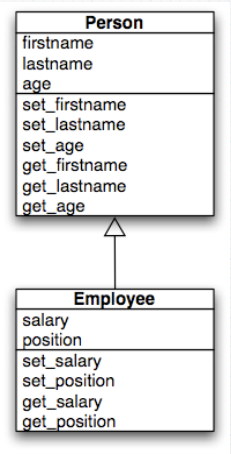
Inheritance

- inheritance allows new classes to be built out of existing classes
- Properties and Methods may be added to child classes

Inheritance

Person is the parent

Employee is the descendant of Person and inherits all the properties and methods



Polymorphism

Operators or methods can vary the way they work depending on the context

Example: happyBirthday() applied to a person

```
tomObject.setAge(42);
happyBirthday(tomObject);
```

→ age becomes 43

Example: happyBirthday() applied to an employee

```
tomObject.setAge(42);
happyBirthday(philObject);
```

→ age becomes 43

→ phil gets email reminding him to buy cakes

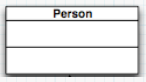
happyBirthday behaves differently depending on the type passed into it

Object Data Model

Encapsulation	Promotes cleaner code
Inheritance	Promotes reuse
Polymorphism	Promotes design strategies (from UML)

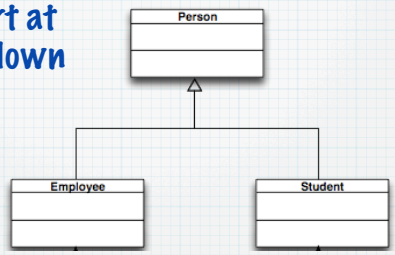
Inheritance - Specialisation design style

Specialisation start at the top and work down



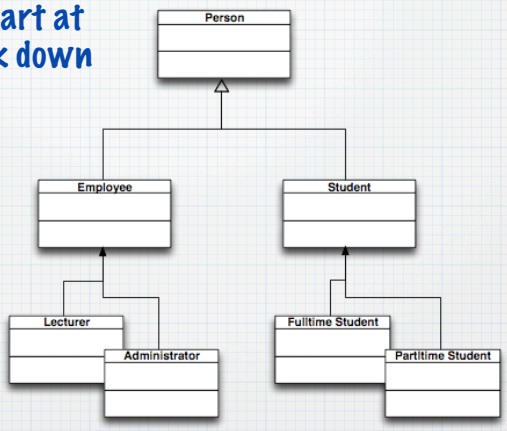
Inheritance - Specialisation design style

Specialisation start at the top and work down



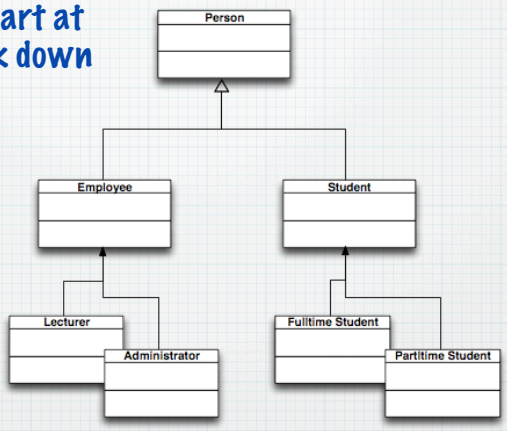
Inheritance - Specialisation design style

Specialisation start at the top and work down



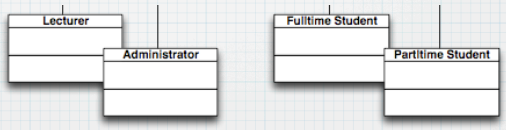
Inheritance - Specialisation design style

Specialisation start at the top and work down



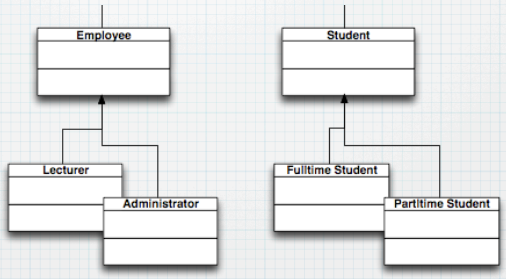
Inheritance - Generalisation design style

Generalisation start at
the bottom and work up



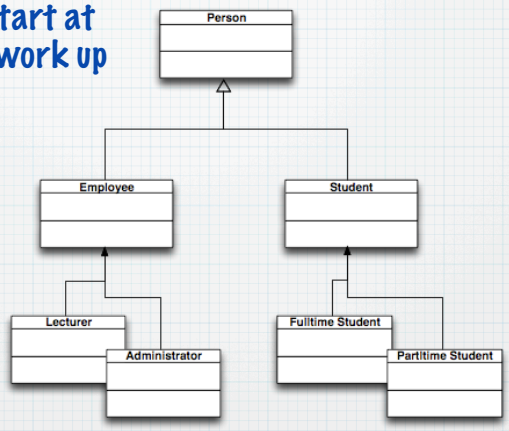
Inheritance - Generalisation design style

Generalisation start at
the bottom and work up



Inheritance - Generalisation design style

Generalisation start at
the bottom and work up



Pre existing class libraries

- To generate the interface (HTML)
- To provide persistence (database storage)
- etc..

... many prebuilt libraries easily added

MVC - design pattern

A way of structuring an application into different component parts

First described by Trygve Reenskau working at Xerox research labs (1979)

Influential paper: Applications Programming in Smalltalk-80 - How to use Model-View-Controller

Design Patterns

- Certain programs reuse the same basic structure or set of ideas
- These regularly occurring structures have been called Design Patterns

Using MVC

Model

Classes that are core to the application

person
invoice
car
booking

Using MVC

Model

Classes that are core to the application

person
invoice
car
booking

View

Classes that are key to the presentation / interaction with the user

HTML classes
forms

Using MVC

Model

Classes that are core to the application

person
invoice
car
booking

View

Classes that are key to the presentation / interaction with the user

HTML classes
forms

Controller Code that uses the Model and View

MVC choices

- Build it all yourself
- Use pre existing MVC frameworks (typically class and object based)

Many different frameworks

PHP Framework	PHP4	PHP5	MVC	Multiple DB's	ORM	DB Objects	Templates	Caching	Validation	Ajax	Auth Module	Modules	EDP
Akelos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
ash.MVC	-	✓	✓	-	-	✓	✓	-	✓	-	✓	✓	-
CakePHP	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
CodeIgniter	✓	✓	✓	✓	-	✓	✓	✓	✓	-	-	-	-
DIY	-	✓	✓	-	✓	✓	✓	✓	-	✓	-	-	-
eZ Components	-	✓	-	✓	-	✓	✓	✓	✓	-	-	-	-
Fusebox	✓	✓	✓	✓	-	-	-	✓	-	✓	-	✓	-
PHP on TRAX	-	✓	✓	✓	✓	✓	-	-	✓	✓	-	✓	-
PHPDevShell	-	✓	-	-	-	-	✓	-	-	✓	✓	✓	-
PhpOpenbiz	-	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	-	-
Prado	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
QPHP	✓	✓	✓	✓	-	✓	✓	-	✓	✓	✓	✓	✓
Seagull	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Symfony Project	-	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	-
WACT	✓	✓	✓	✓	-	✓	✓	-	✓	-	-	✓	-
WASP	-	✓	✓	-	-	✓	✓	-	✓	✓	✓	✓	-
Yii	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Zend	-	✓	✓	✓	✓	✓	-	✓	✓	-	✓	✓	-
ZooP	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓	-	-

- http://www.phpframeworks.com/

cakePHP

MVC frameworks

- Have very strict rules for use
- Names must match certain dictated patterns
- Files must be place in certain places
- Classes in your application inherit from the MVC classes

Creating tables for an app

Users table standard fields for a user registration site

```
mysql> CREATE TABLE `users` (
    -> `id` INT( 10 ) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
    -> `username` VARCHAR( 40 ) NOT NULL ,
    -> `password` VARCHAR( 40 ) NOT NULL ,
    -> `email` VARCHAR( 255 ) NOT NULL ,
    -> `first_name` VARCHAR( 40 ) NOT NULL ,
    -> `last_name` VARCHAR( 40 ) NOT NULL ,
    ->    UNIQUE  (username , email )
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql>
```

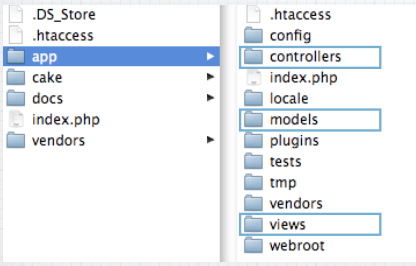
Some cakePHP specific rules about names

All tables must have a primary key id

All tablenames must be plural

File locations

Files we create need to go in certain distinct places to work



Add model, view and controllers in appropriate folders

Creating user model for the app

user.php in models folder

```
<?php
class User extends AppModel
{
var $name = 'User';
}
?>
```

Creates a new class derived from cakePHP AppModel class

CakePHP requires us to create a $name property

All model filenames and classnames must be singular and correspond to the database table

Creating user controller for the app

users_controller.php in controller folder

```
<?php
class UsersController extends AppController
{



}
?>
```

Creates a new class derived from cakePHP AppController class

All controller names should be plural versions of the database name appended to Controller

`UsersController`

Adding behaviour to the Controller

```
<?php
class UsersController extends AppController
{
    function index(){
    }

    function register(){
    }

}
?>
```

Each function will tie to a URL and a view

Adding behaviour to the Controller

Each function (behaviour) will tie to a URL and a view

```
<?php
class UsersController extends AppController
{
    function index(){
    }

    function register(){
    }

}
?>
```

http://localhost:8888/users/

http://localhost:8888/users/register

Adding a register View

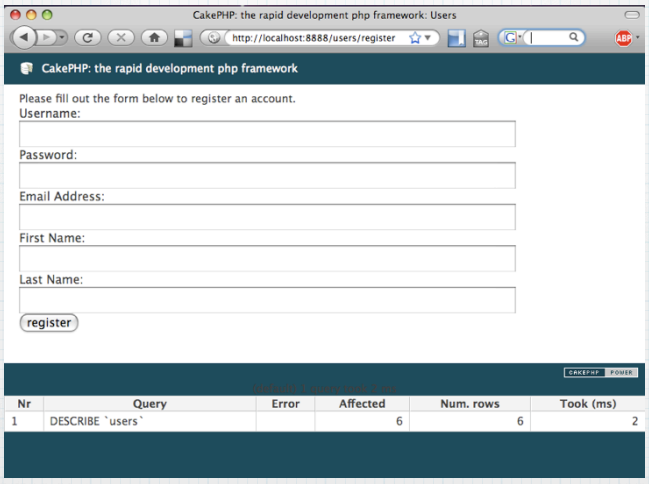
register.ctp in app/views/users

```
<form action="/users/register" method="post">
<p>Please fill out the form below to register an account.</p>
<label>Username:</label><input name="username" size="40" />

<label>Password:</label><input type="password" name="password" size="40" />

<label>Email Address:</label><input name="email" size="40" maxlength="255" />

<label>First Name:</label><input name="first_name" size="40" />

<label>Last Name:</label><input name="last_name" size="40" />

<input type="submit" value="register" />
</form>
```

Note that each form item corresponds to a field in the table (same name)

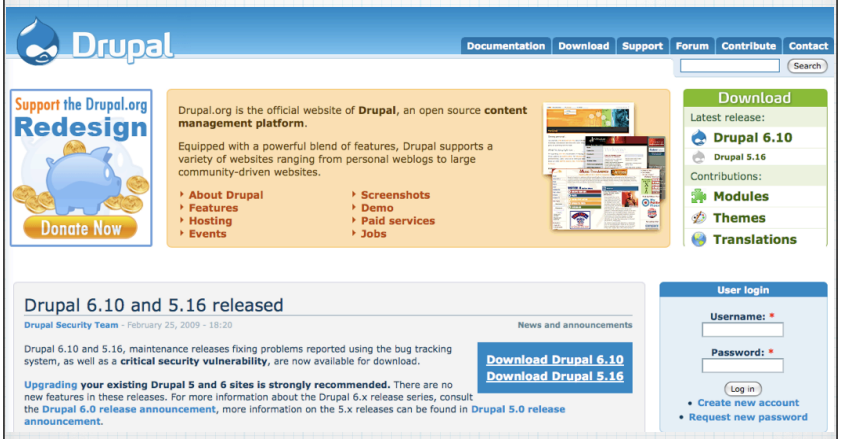
Adding a register View

register.ctp in app/views/users



Prebuilt platforms

- php, python, mysql based (typically)
- Installation simplistic
- Add-ons for particular functionality
- Joomla, Drupal and others



- Utility (734)
- Content (689)
- Content display (680)
- Third-party integration (609)
- Administration (426)
- Developer (309)
- Content Construction Kit (CCK) (301)
- Community (279)
- Media (264)
- User access/authentication (225)
- User management (215)
- Filters/editors (207)
- Taxonomy (204)
- Views (204)
- Javascript Utilities (203)
- Theme related (193)
- e-Commerce (184)
- Mail (160)
- Commerce / advertising (153)
- File management (119)
- Import/export (125)
- Search (110)
- Syndication (110)
- Evaluation/rating (99)
- Site navigation (94)
- Security (89)
- Organic Groups (88)
- Location (81)
- Event (69)
- Multilingual (64)
- Statistics (52)
- RDF (38)
- Games and Amusements (33)
- Performance and Scalability (29)
- Paging (26)

- 100's of add-ons/extensions

Access & Security
Authentication (34)
Authentication External (17)
Backend Access Control (7)
Backup (8)
Captcha (12)
Content Restriction (33)
Frontend Access Control (28)
Site Access (19)
Site Security (18)

Administration
Admin add-ons (60)
Admin Interface (13)

Ads & Affiliates
Affiliate Advertising (46)
Amazon (17)
Banner Management (46)
Classified Ads (6)
Corner Banners (5)
Google Ads (39)
Jobs & Recruitment (13)
Text & Link Ads (21)

Authoring
Authors (12)
Content Submission (19)
Planned Content (9)

Contacts & Feedback
Articles comments (24)
Contact Details (17)
Contact forms (13)
Email (14)
Forms (23)
Greetings (12)
Guest Book (14)
Polls (10)
Quiz & Surveys (15)
Testimonials & Suggestions (12)

Content & News
Articles Slideshow (35)
Blog (19)
Frontend News (24)
Latest News (37)
Microblogging (7)
News Tickers & Scrollers (31)
Newsflash (12)
Popular Content (9)
Quotes (15)
Random News (8)
Related Items (23)
RSS Syndicate (26)

Core Enhancements
Content Management (18)

Extension Specific
AdsManager extensions (10)
Agora Extensions (6)
AlphaContent extensions (5)
BannersManager extensions (5)
Community Builder extensions (185)
DOCMan extensions (17)
ECJC extensions (7)
Fireboard Forum extensions (24)
Freeway extensions (5)
Hot Property extensions (24)
JCE extensions (9)
jDownloads extensions (8)
Joom!Fish extensions (18)
Joomap extensions (6)
JoomLeague Plugins (9)
JoomSEF extensions (76)
Kunena Forum extensions (3)
Migrator Extensions (7)
Pages-and-Items extensions (12)
Phoca Download extensions (7)
Phoca Gallery extensions (8)
PUArcade extensions (27)
Remository Extensions (7)

Photos & Images
Articles Images (32)
eCards (2)
Images (27)
Images Rotators (30)
Images Slideshow (54)
Maps (35)
Panorama (8)
Photo Flash Gallery (17)
Photo Gallery (41)
Photo Gallery add-ons (17)

Search & Indexing
Domain Search (9)
Extensions Search (29)
Search Results (15)
Site Search (28)
Tags & Clouds (36)

Site Management
Cache (12)
Content Statistics (12)
Credits (17)
Data Reports (10)
Ranks (19)
SEF (16)
SEO & Metadata (49)
Site Analytics (28)

Development approaches

Procedural / imperative	More coding
Object Oriented data structure with imperative style	↓
MVC style	
Pre built MVC frameworks	
Pre-built platforms	Less coding

Development approaches

Procedural / imperative	Get exactly what you want
Object Oriented data structure with imperative style	↓
MVC style	
Pre built MVC frameworks	
Pre-built platforms	May have to compromise