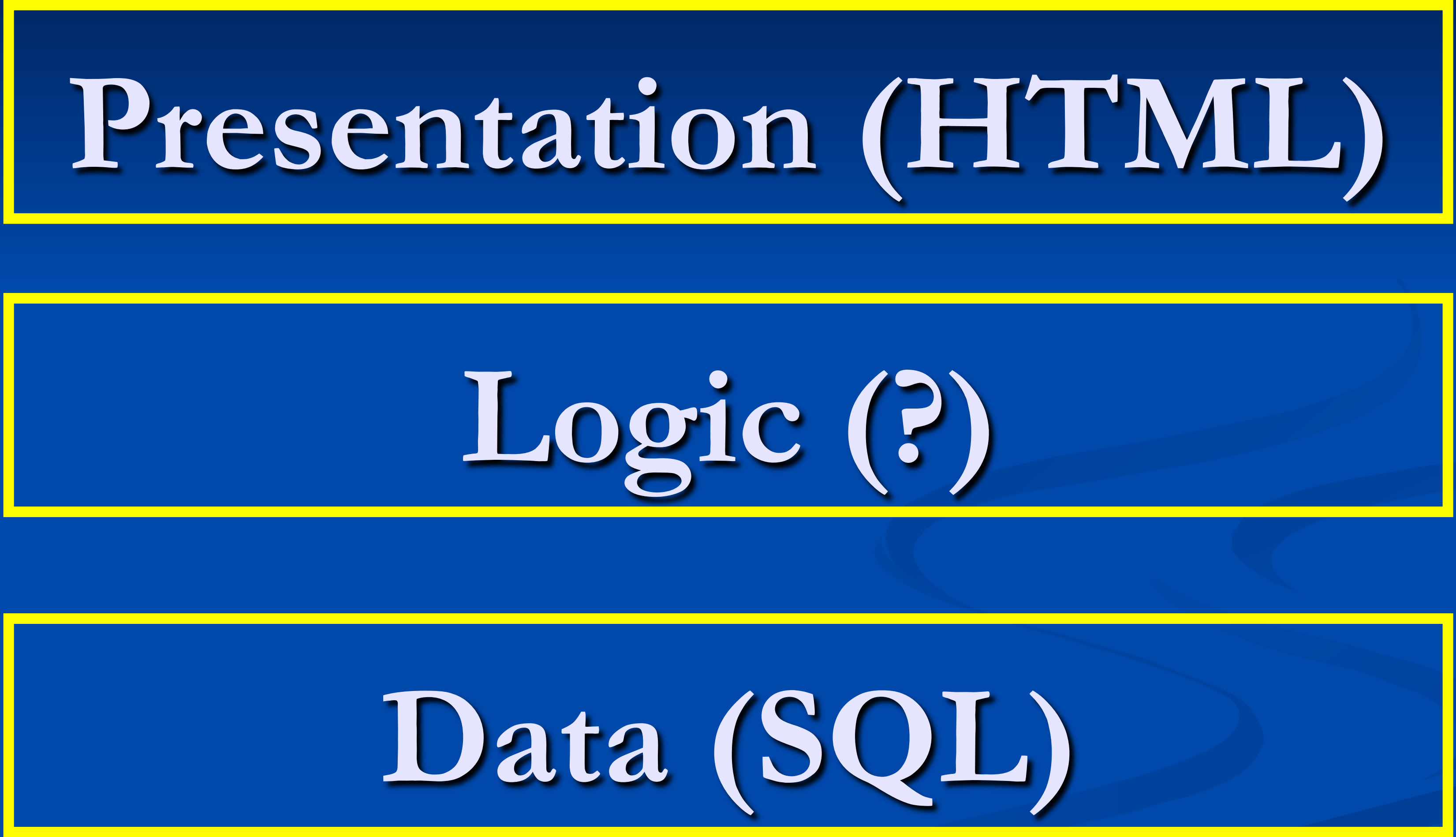
3-tier

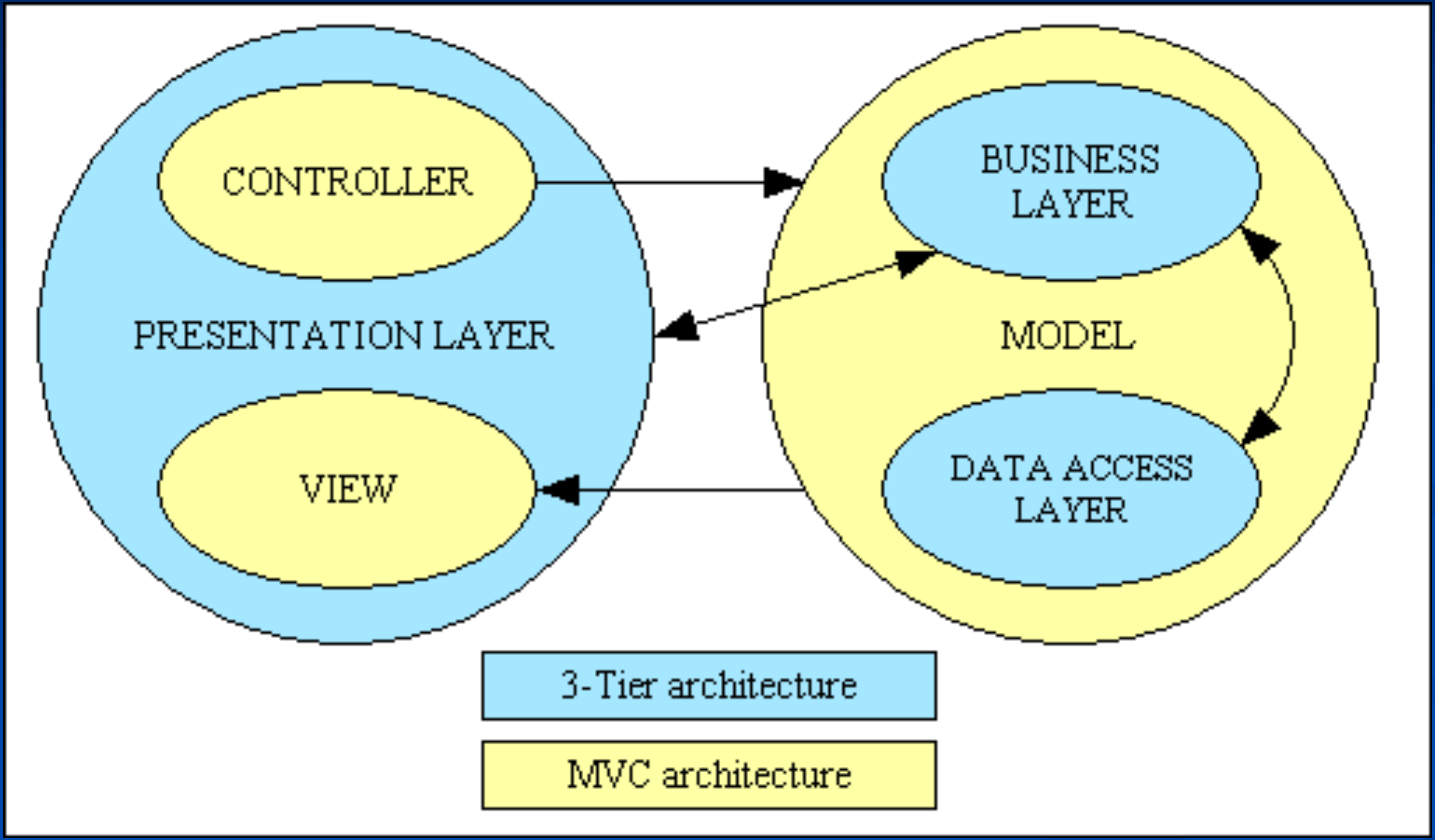


Presentation (HTML)

Logic (?)

Data (SQL)

MVC


CONTROLLER
PRESENTATION LAYER
VIEW
BUSINESS LAYER
MODEL
DATA ACCESS LAYER
3-Tier architecture
MVC architecture

```
<FONT color=red>
  <P onclick=“dothis()”>Teste</P>
</FONT>
```

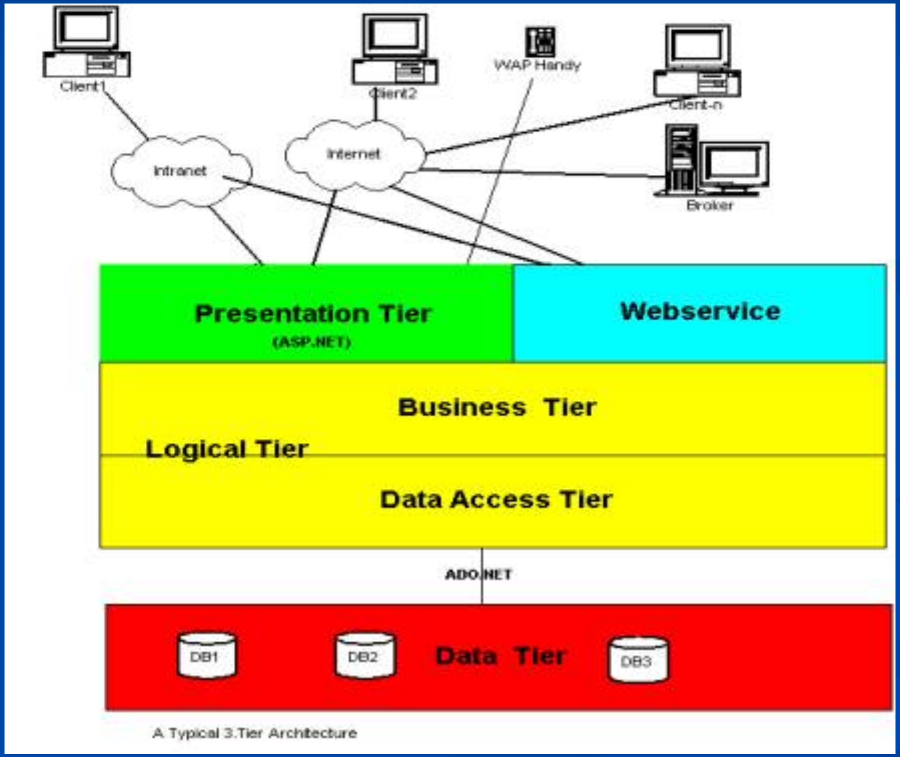
```
.zbr { font-color: red}
```

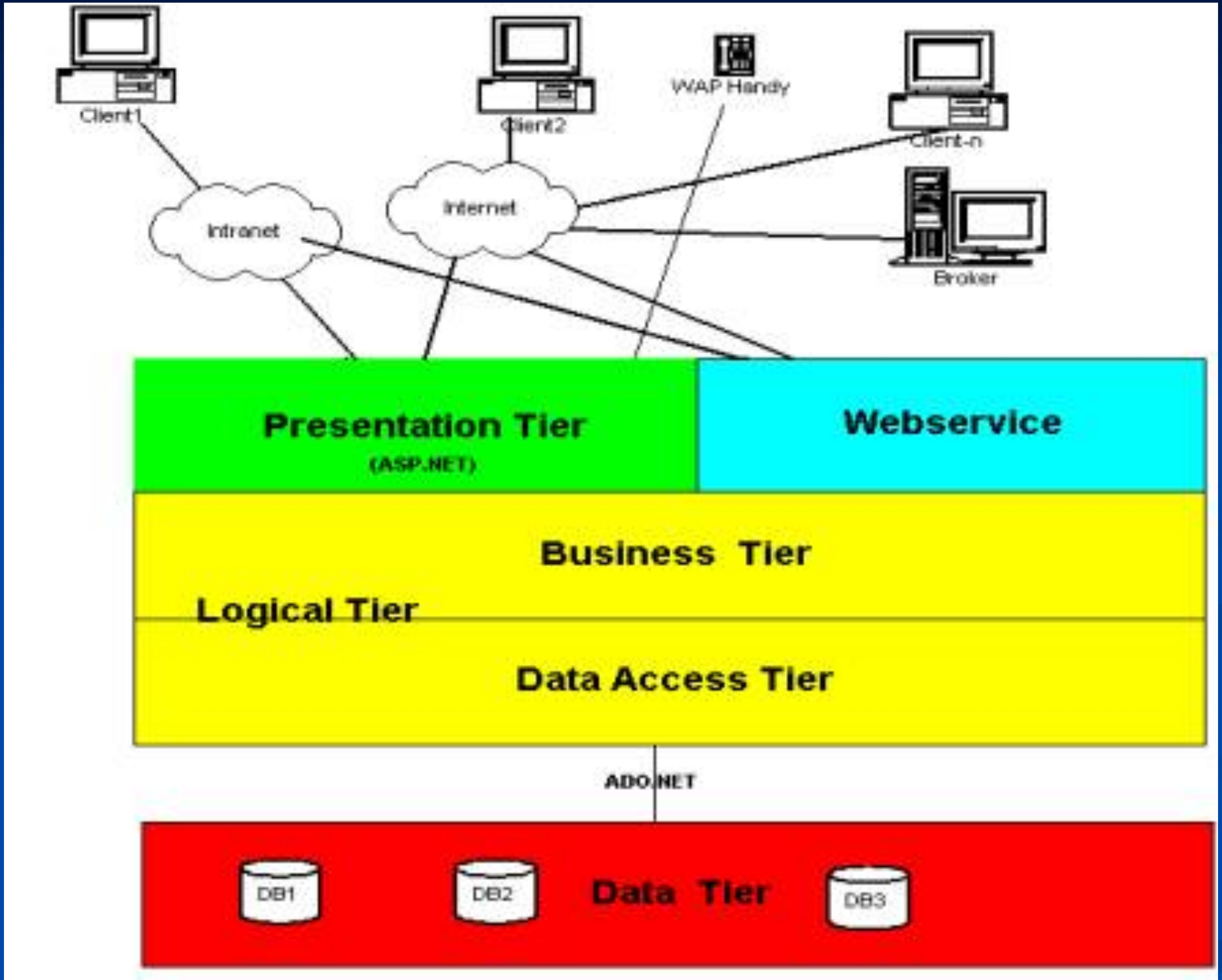
```
dothis()
```

```
<P id=zbr>Teste</P>
```

.NET

IIS

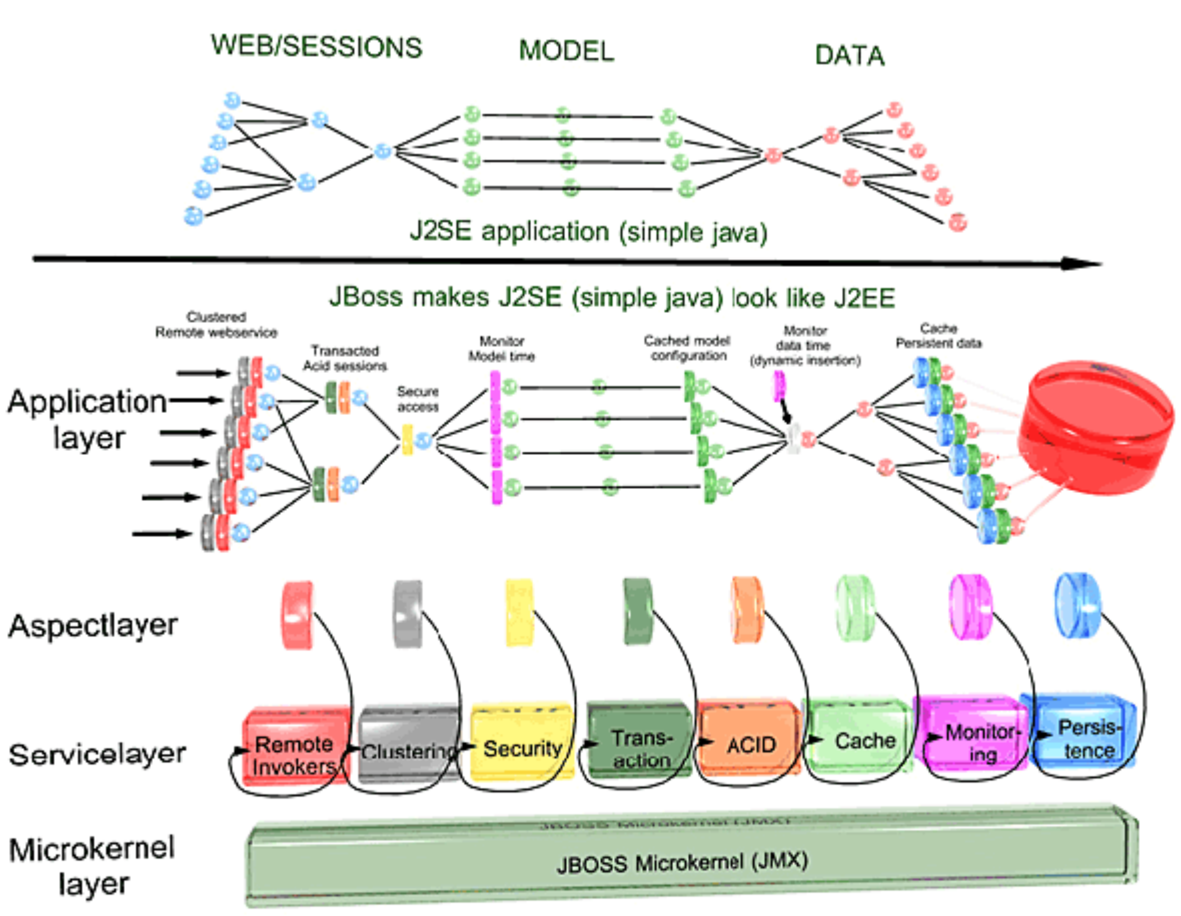

Client1
Client2
WAP Handy
Client-n
Broker
Intranet
Internet
Presentation Tier
(ASP.NET)
Webservice
Business Tier
Logical Tier
Data Access Tier
ADO.NET
DB1
DB2
Data Tier
DB3
A Typical 3.Tier Architecture



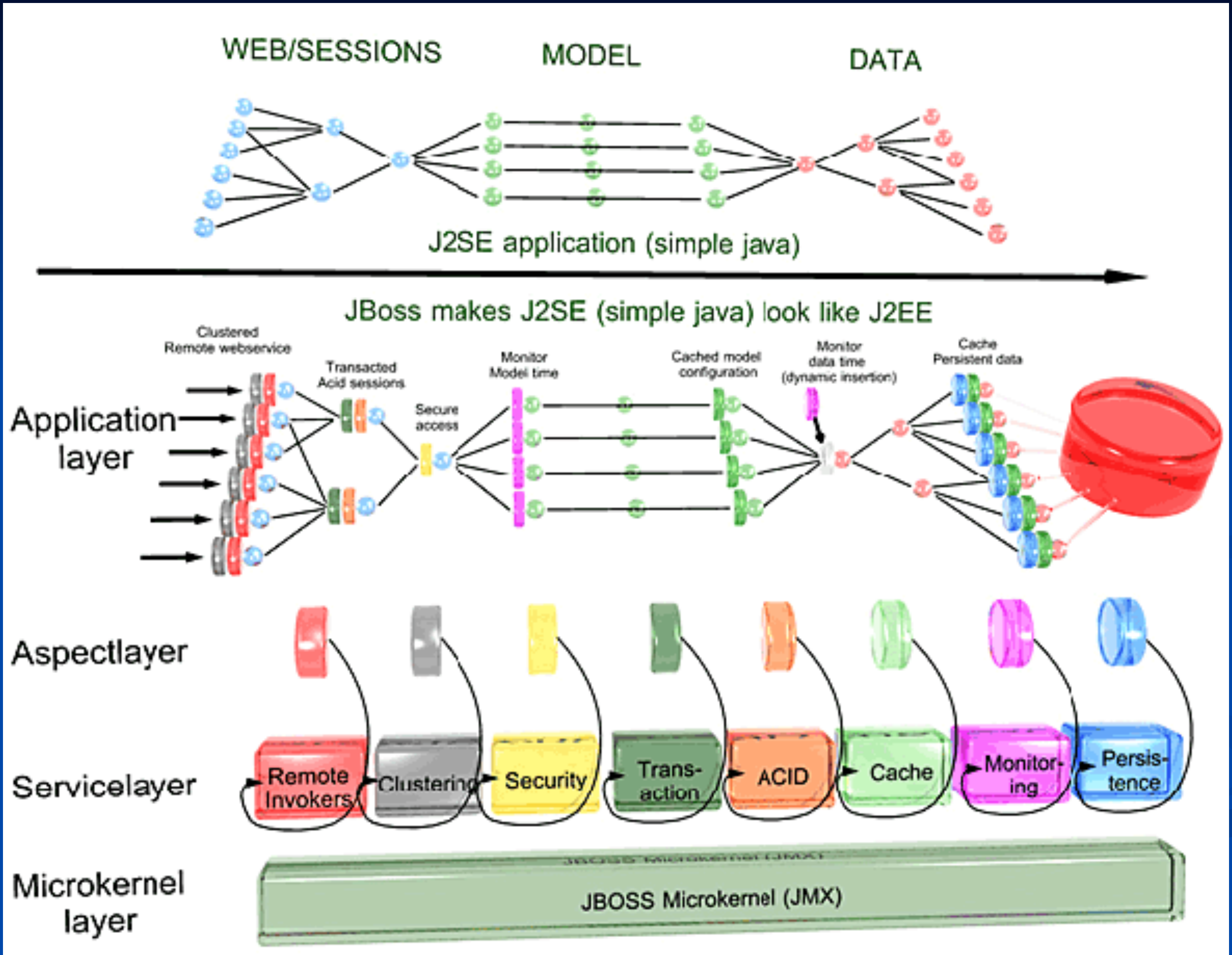
A Typical 3.Tier Architecture

JBoss

J2EE

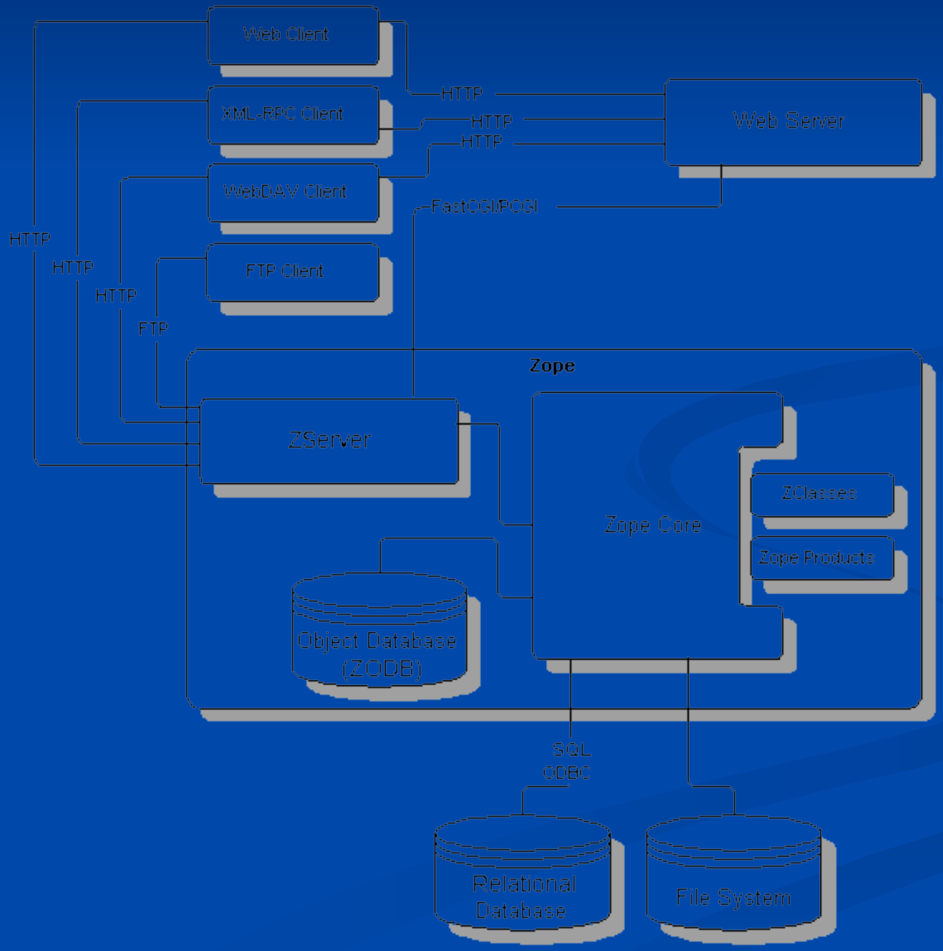


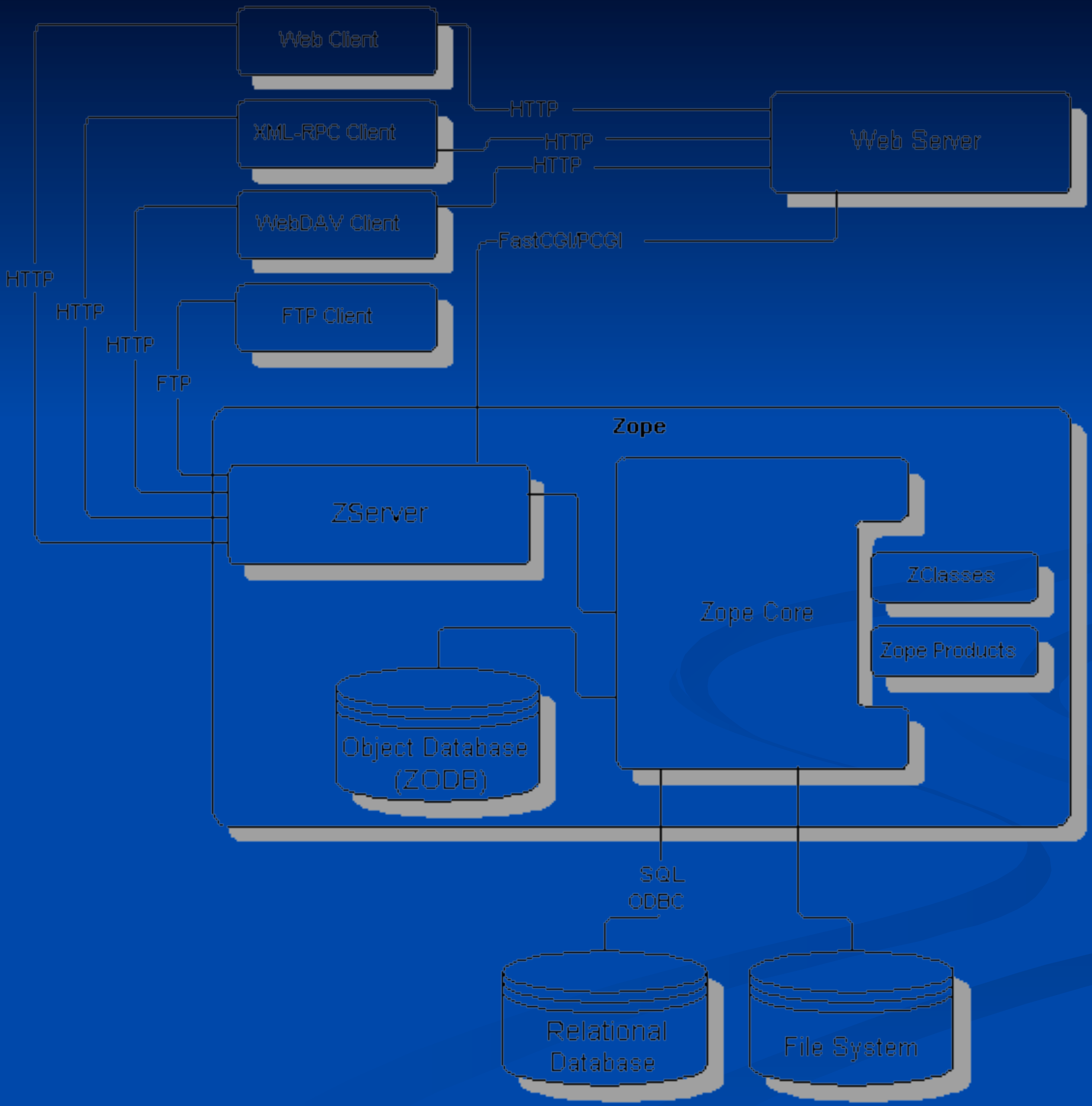
WEB/SESSIONS
MODEL
DATA
J2SE application (simple java)
JBoss makes J2SE (simple java) look like J2EE
Clustered Remote webservice
Transacted Acid sessions
Secure access
Monitor Model time
Cached model configuration
Monitor data time (dynamic insertion)
Cache Persistent data
Application layer
Aspectlayer
Servicelayer
Remote Invokers
Clustering
Security
Trans-action
ACID
Cache
Monitor-ing
Persis-tence
Microkernel layer
JBOSS Microkernel (JMX)


WEB/SESSIONS
MODEL
DATA
J2SE application (simple java)
JBoss makes J2SE (simple java) look like J2EE
Clustered Remote webservice
Transacted Acid sessions
Secure access
Monitor Model time
Cached model configuration
Monitor data time (dynamic insertion)
Cache Persistent data
Application layer
Aspectlayer
Servicelayer
Remote Invokers
Clustering
Security
Trans-action
ACID
Cache
Monitor-ing
Persis-tence
Microkernel layer
JBOSS Microkernel (JMX)

Zope

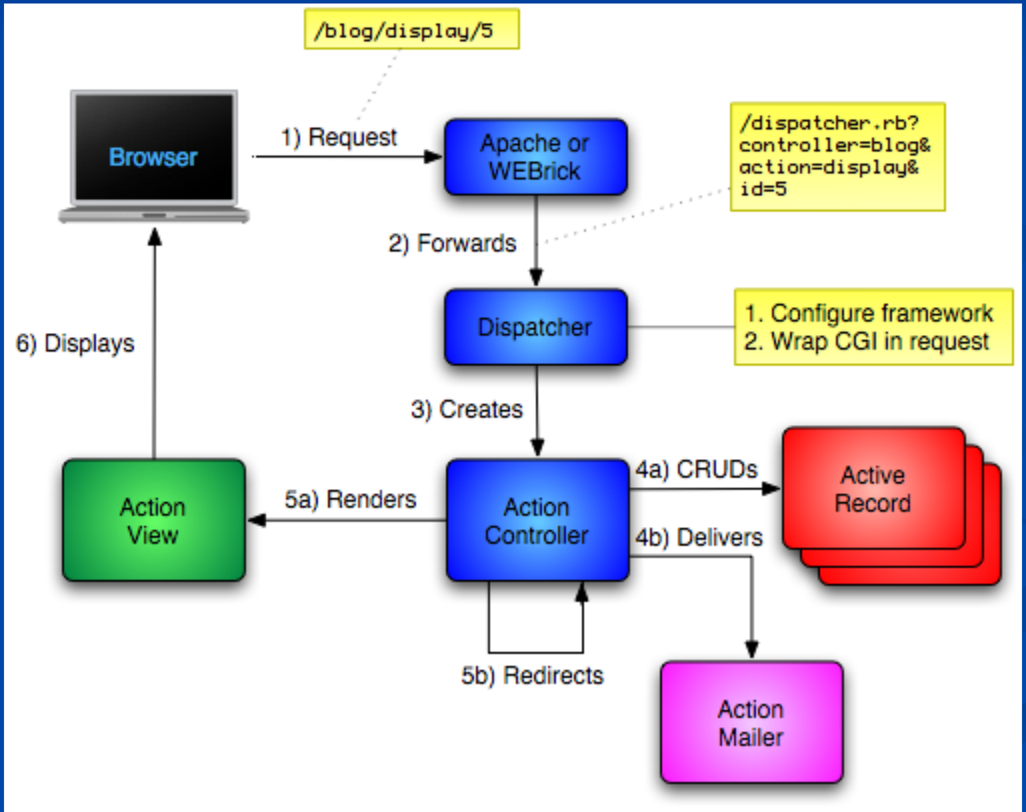
Z Object Publishing

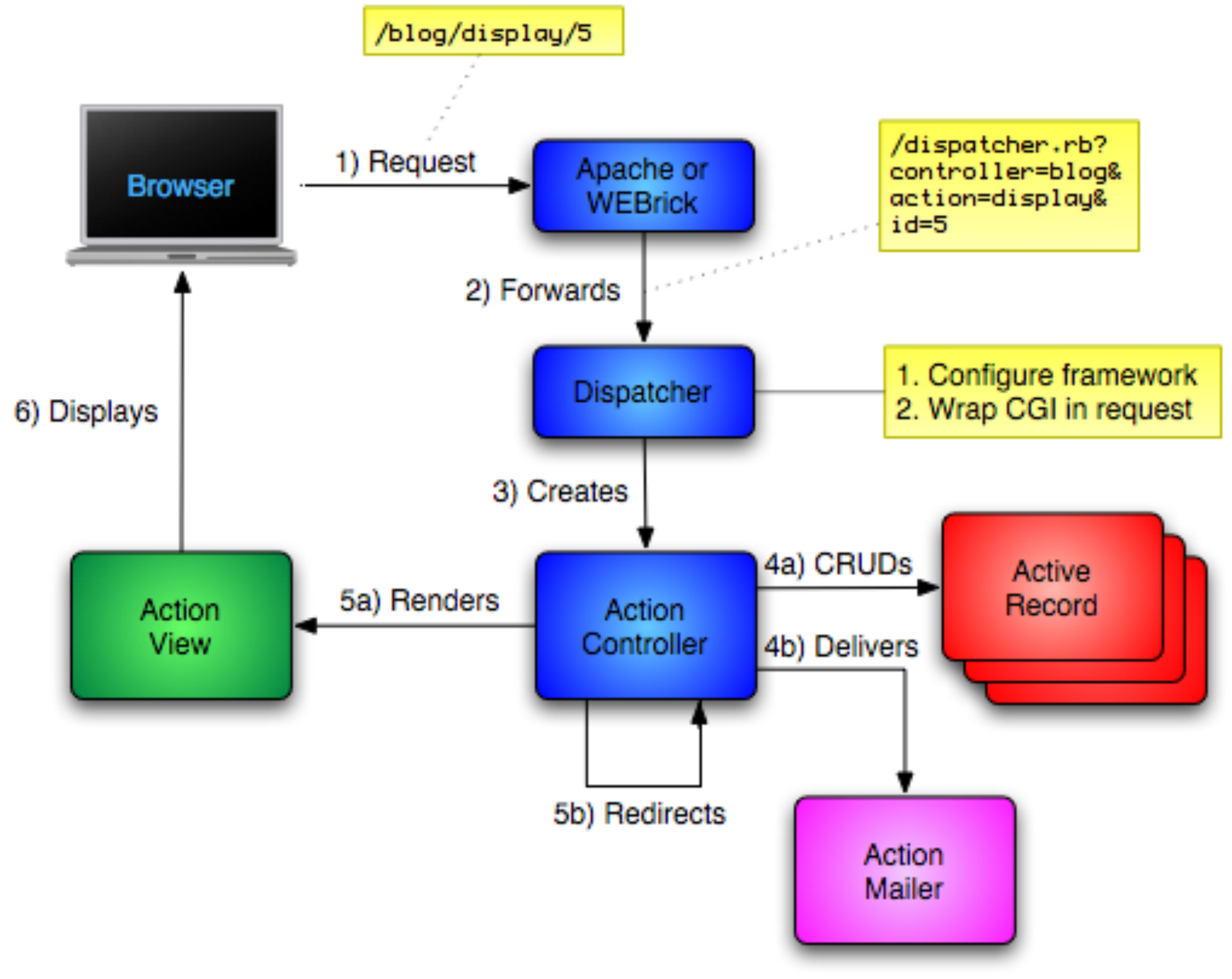



Web Client
XML-RPC Client
WebDAV Client
FTP Client
Web Server
HTTP
HTTP
HTTP
FastCGI/PCGI
HTTP
HTTP
HTTP
FTP
Zope
ZServer
Zope Core
ZClasses
Zope Products
Object Database (ZODB)
SQL
ODBC
Relational Database
File System

Rails

Ruby


/blog/display/5
Browser
1) Request
Apache or WEBrick
/dispatcher.rb?controller=blog&action=display&id=5
2) Forwards
Dispatcher
1. Configure framework
2. Wrap CGI in request
6) Displays
3) Creates
4a) CRUDs
Active Record
Action View
5a) Renders
Action Controller
4b) Delivers
5b) Redirects
Action Mailer



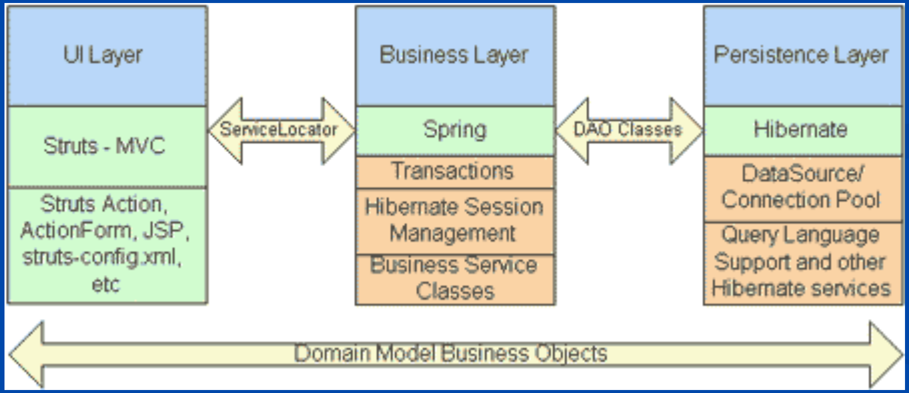
/blog/display/5

/dispatcher.rb?
controller=blog&
action=display&
id=5

1. Configure framework
2. Wrap CGI in request

Java

Struts/Springs/Hibernate



UI Layer

Struts - MVC

Struts Action, ActionForm, JSP, struts-config.xml, etc

ServiceLocator

Business Layer

Spring

Transactions

Hibernate Session Management

Business Service Classes

DAO Classes

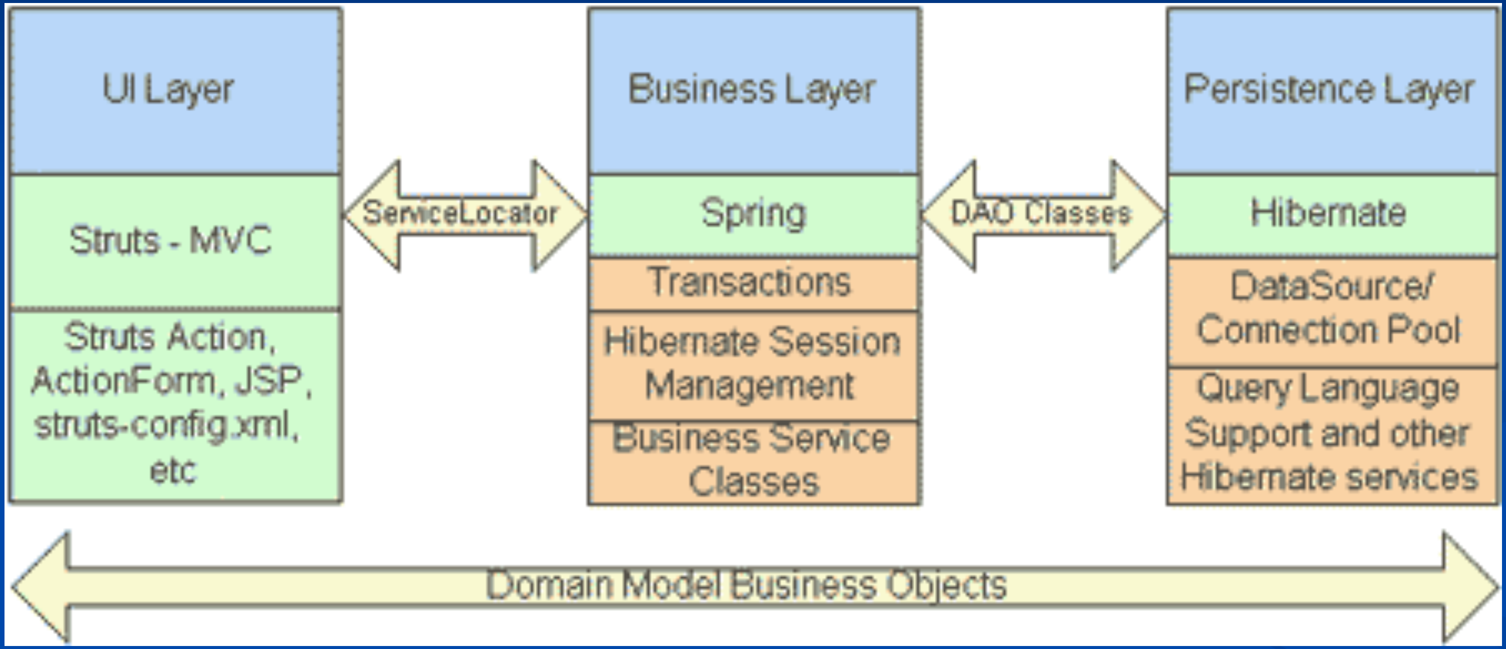
Persistence Layer

Hibernate

DataSource/ Connection Pool

Query Language Support and other Hibernate services

Domain Model Business Objects



UI Layer

Struts - MVC

Struts Action, ActionForm, JSP, struts-config.xml, etc

ServiceLocator

Business Layer

Spring

Transactions

Hibernate Session Management

Business Service Classes

DAO Classes

Persistence Layer

Hibernate

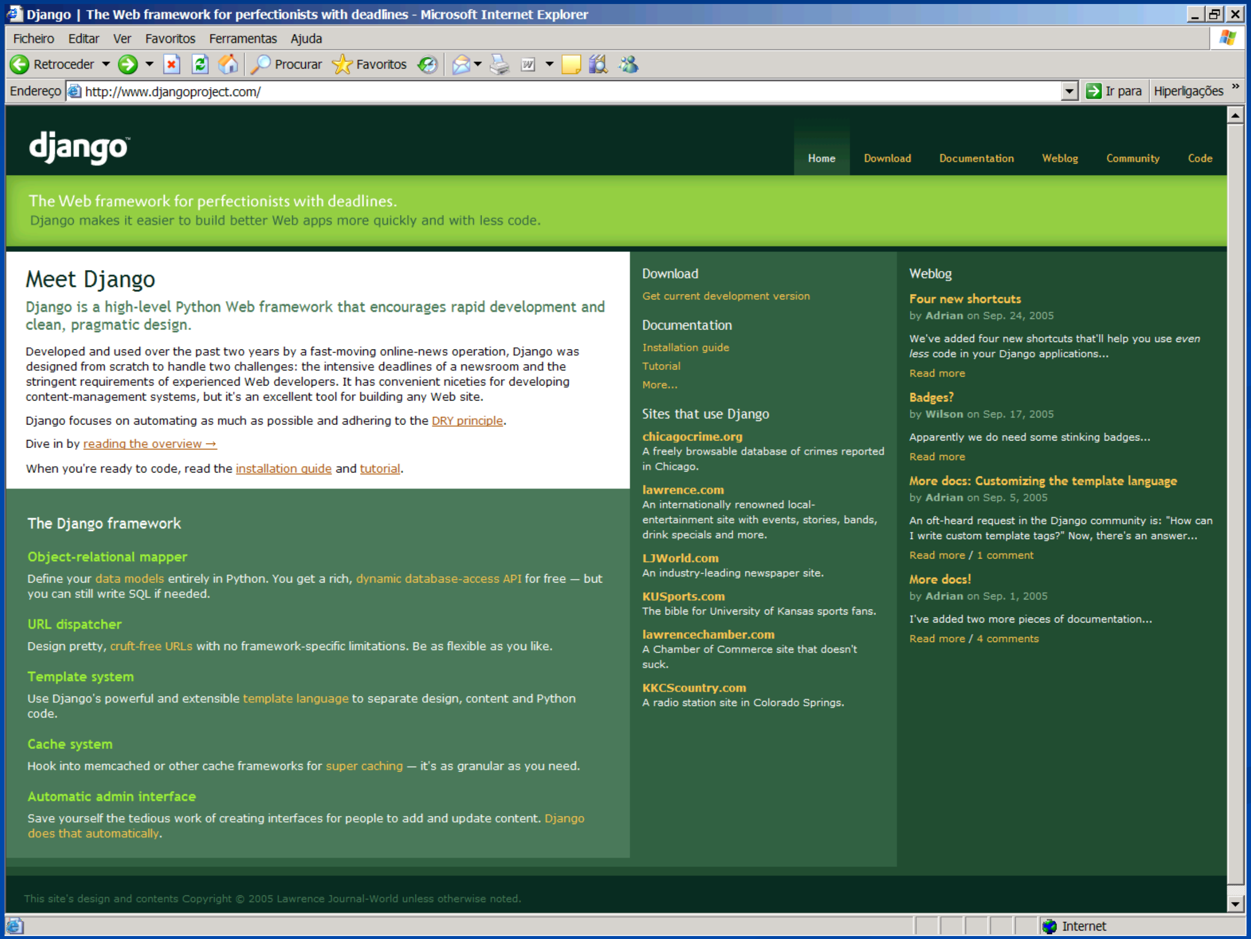
DataSource/ Connection Pool

Query Language Support and other Hibernate services

Domain Model Business Objects

Django

Python



The Web framework for perfectionists with deadlines.

Django makes it easier to build better Web apps more quickly and with less code.

Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Developed and used over the past two years by a fast-moving online-news operation, Django was designed from scratch to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of experienced Web developers. It has convenient niceties for developing content-management systems, but it's an excellent tool for building any Web site.

Django focuses on automating as much as possible and adhering to the DRY principle.

Dive in by reading the overview →

When you're ready to code, read the installation guide and tutorial.

The Django framework

Object-relational mapper

Define your data models entirely in Python. You get a rich, dynamic database-access API for free — but you can still write SQL if needed.

URL dispatcher

Design pretty, cruft-free URLs with no framework-specific limitations. Be as flexible as you like.

Template system

Use Django's powerful and extensible template language to separate design, content and Python code.

Cache system

Hook into memcached or other cache frameworks for super caching — it's as granular as you need.

Automatic admin interface

Save yourself the tedious work of creating interfaces for people to add and update content. Django does that automatically.

Download

Get current development version

Documentation

Installation guide

Tutorial

More...

Sites that use Django

chicagocrime.org
A freely browsable database of crimes reported in Chicago.

lawrence.com
An internationally renowned local-entertainment site with events, stories, bands, drink specials and more.

LJWorld.com
An industry-leading newspaper site.

KUSports.com
The bible for University of Kansas sports fans.

lawrencechamber.com
A Chamber of Commerce site that doesn't suck.

KKCScountry.com
A radio station site in Colorado Springs.

Weblog

Four new shortcuts
by **Adrian** on Sep. 24, 2005

We've added four new shortcuts that'll help you use *even less* code in your Django applications...

Read more

Badges?
by **Wilson** on Sep. 17, 2005

Apparently we do need some stinking badges...

Read more

More docs: Customizing the template language
by **Adrian** on Sep. 5, 2005

An oft-heard request in the Django community is: "How can I write custom template tags?" Now, there's an answer...

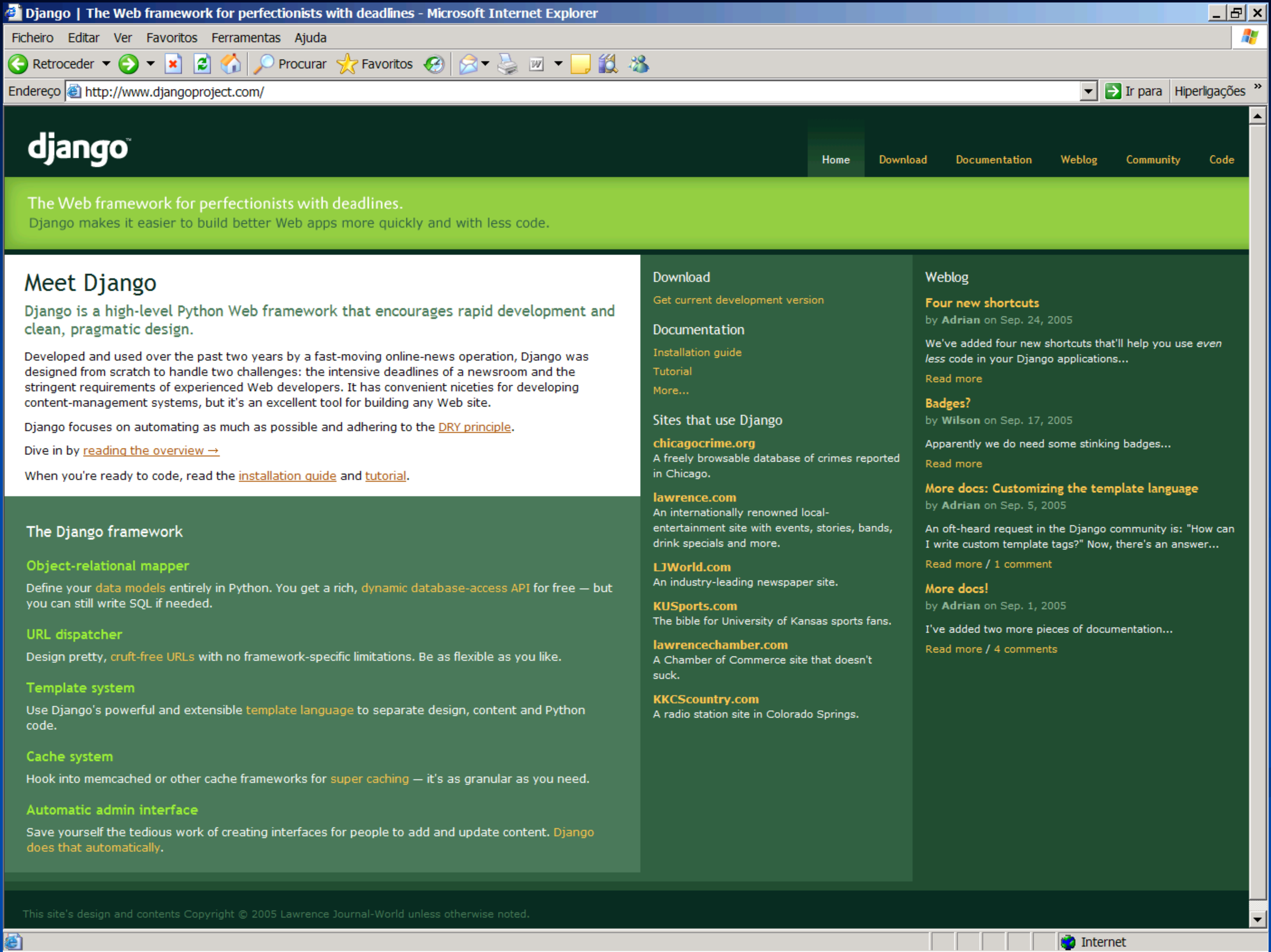
Read more / 1 comment

More docs!
by **Adrian** on Sep. 1, 2005

I've added two more pieces of documentation...

Read more / 4 comments

This site's design and contents Copyright © 2005 Lawrence Journal-World unless otherwise noted.


Django | The Web framework for perfectionists with deadlines - Microsoft Internet Explorer
Ficheiro Editar Ver Favoritos Ferramentas Ajuda
Retroceder
Procurar
Favoritos
Endereço
http://www.djangoproject.com/
Ir para
Hiperligações
Internet

django

Home Download Documentation Weblog Community Code

The Web framework for perfectionists with deadlines.
Django makes it easier to build better Web apps more quickly and with less code.

Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Developed and used over the past two years by a fast-moving online-news operation, Django was designed from scratch to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of experienced Web developers. It has convenient niceties for developing content-management systems, but it's an excellent tool for building any Web site.

Django focuses on automating as much as possible and adhering to the DRY principle.

Dive in by reading the overview →

When you're ready to code, read the installation guide and tutorial.

The Django framework

Object-relational mapper

Define your data models entirely in Python. You get a rich, dynamic database-access API for free — but you can still write SQL if needed.

URL dispatcher

Design pretty, cruft-free URLs with no framework-specific limitations. Be as flexible as you like.

Template system

Use Django's powerful and extensible template language to separate design, content and Python code.

Cache system

Hook into memcached or other cache frameworks for super caching — it's as granular as you need.

Automatic admin interface

Save yourself the tedious work of creating interfaces for people to add and update content. Django does that automatically.

Download

Get current development version

Documentation

Installation guide
Tutorial
More...

Sites that use Django

chicagocrime.org
A freely browsable database of crimes reported in Chicago.

lawrence.com
An internationally renowned local-entertainment site with events, stories, bands, drink specials and more.

LJWorld.com
An industry-leading newspaper site.

KUSports.com
The bible for University of Kansas sports fans.

lawrencechamber.com
A Chamber of Commerce site that doesn't suck.

KKCScountry.com
A radio station site in Colorado Springs.

Weblog

Four new shortcuts
by **Adrian** on Sep. 24, 2005

We've added four new shortcuts that'll help you use *even less* code in your Django applications...

Read more

Badges?
by **Wilson** on Sep. 17, 2005

Apparently we do need some stinking badges...

Read more

More docs: Customizing the template language
by **Adrian** on Sep. 5, 2005

An oft-heard request in the Django community is: "How can I write custom template tags?" Now, there's an answer...

Read more / 1 comment

More docs!
by **Adrian** on Sep. 1, 2005

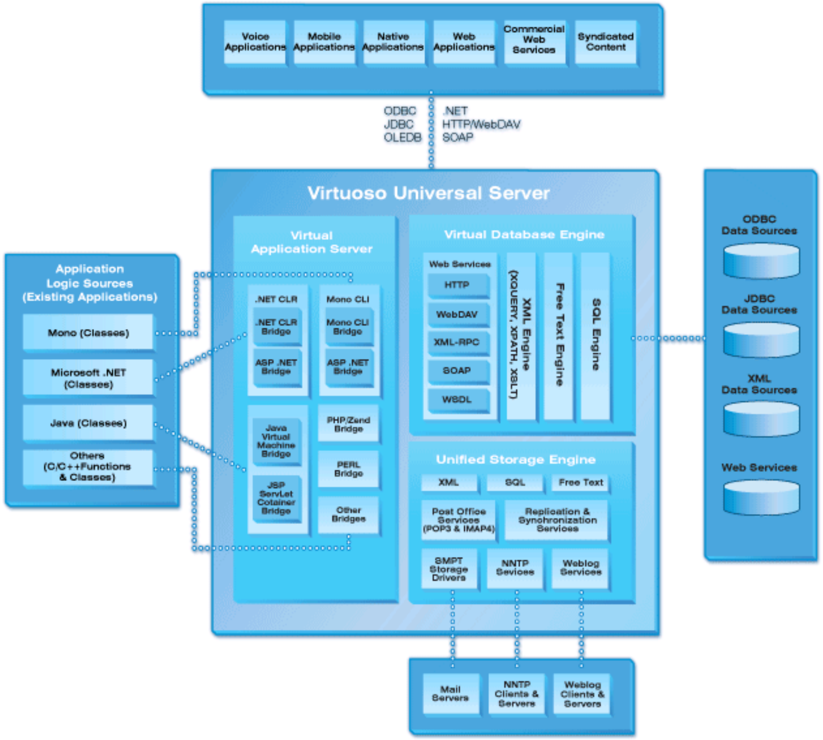
I've added two more pieces of documentation...

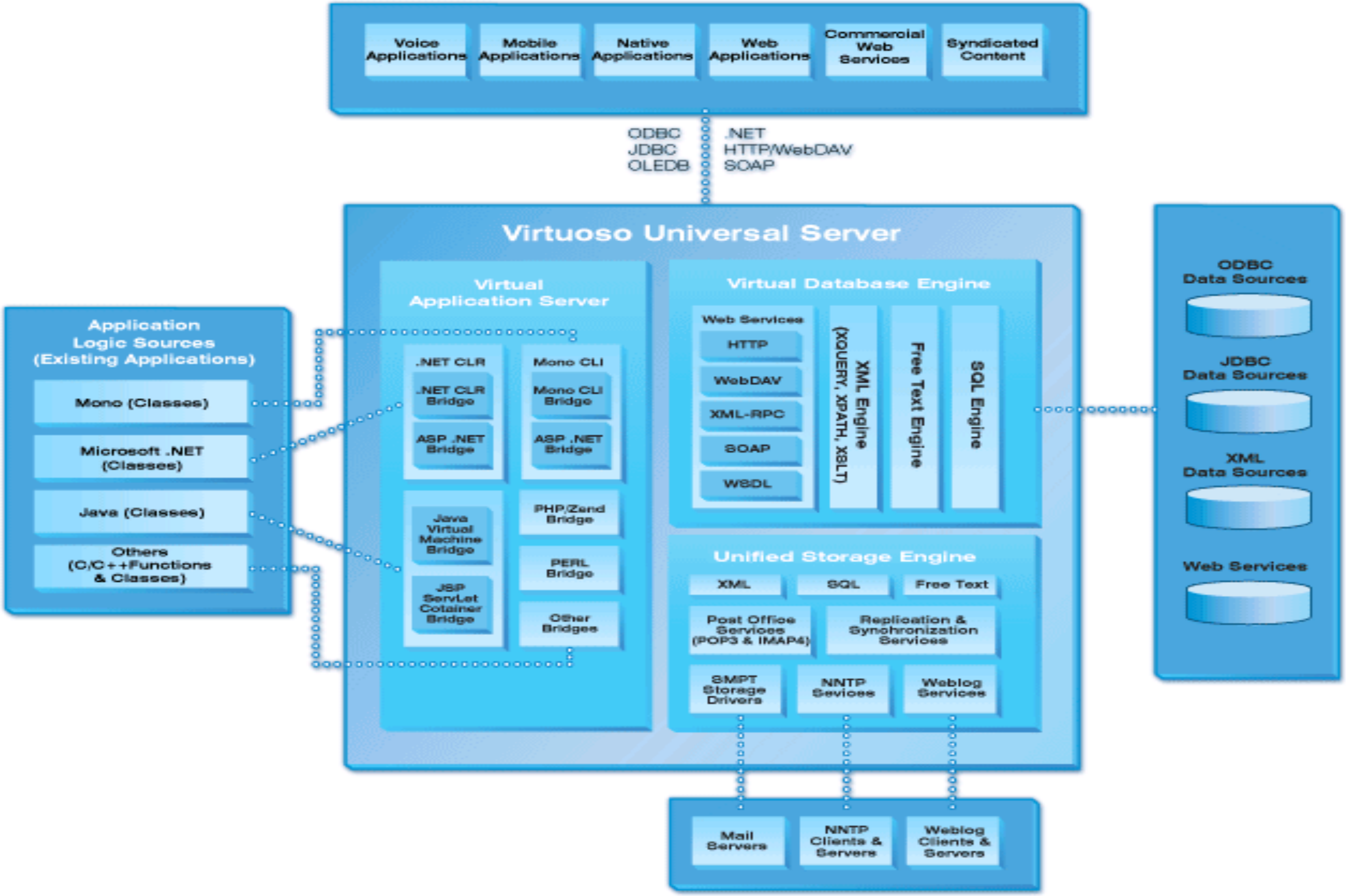
Read more / 4 comments

This site's design and contents Copyright © 2005 Lawrence Journal-World unless otherwise noted.

OpenLink Virtuoso

Universal Server Platform

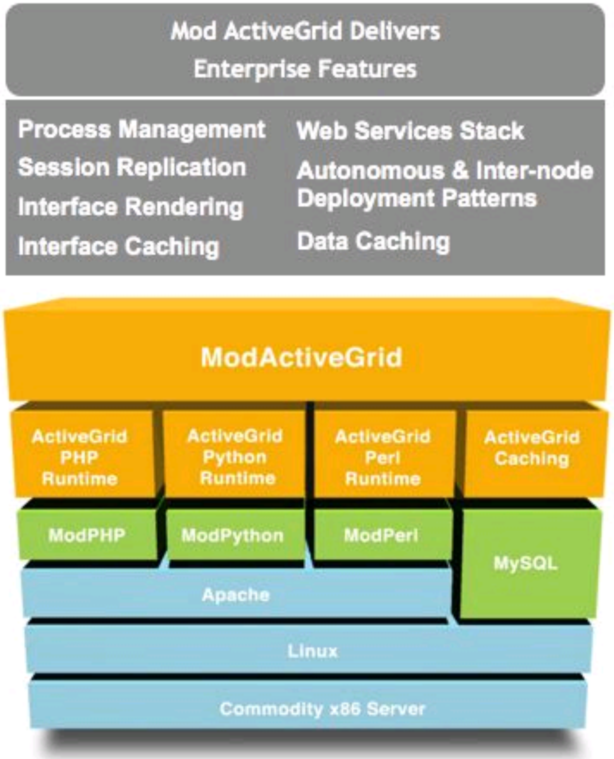




Voice Applications
Mobile Applications
Native Applications
Web Applications
Commercial Web Services
Syndicated Content
ODBC
JDBC
OLEDB
.NET
HTTP/WebDAV
SOAP
Virtuoso Universal Server
Virtual Application Server
.NET CLR
.NET CLR Bridge
ASP .NET Bridge
Mono CLI
Mono CLI Bridge
ASP .NET Bridge
Java Virtual Machine Bridge
JSP ServLet Cotainer Bridge
PHP/Zend Bridge
PERL Bridge
Other Bridges
Virtual Database Engine
Web Services
HTTP
WebDAV
XML-RPC
SOAP
WSDL
XML Engine (XQUERY, XPATH, XSLT)
Free Text Engine
SQL Engine
Unified Storage Engine
XML
SQL
Free Text
Post Office Services (POP3 & IMAP4)
Replication & Synchronization Services
SMPT Storage Drivers
NNTP Sevices
Weblog Services
Application Logic Sources (Existing Applications)
Mono (Classes)
Microsoft .NET (Classes)
Java (Classes)
Others (C/C++Functions & Classes)
ODBC Data Sources
JDBC Data Sources
XML Data Sources
Web Services
Mail Servers
NNTP Clients & Servers
Weblog Clients & Servers

Active Grid

Enterprise LAMP



Mod ActiveGrid Delivers
Enterprise Features

Process Management
Session Replication
Interface Rendering
Interface Caching

Web Services Stack
Autonomous & Inter-node Deployment Patterns
Data Caching

ModActiveGrid

ActiveGrid PHP Runtime

ActiveGrid Python Runtime

ActiveGrid Perl Runtime

ActiveGrid Caching

ModPHP

ModPython

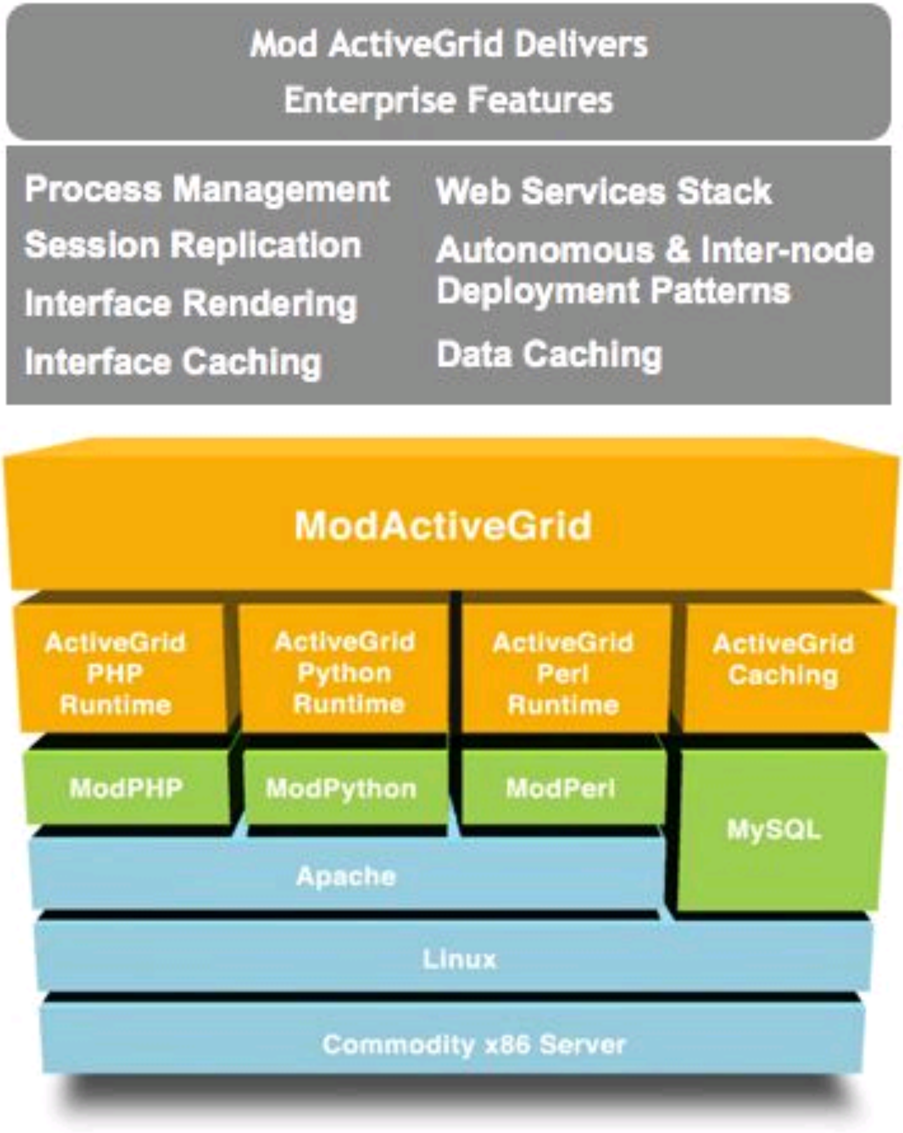
ModPerl

MySQL

Apache

Linux

Commodity x86 Server



Mod ActiveGrid Delivers
Enterprise Features

- Process Management
- Session Replication
- Interface Rendering
- Interface Caching
- Web Services Stack
- Autonomous & Inter-node Deployment Patterns
- Data Caching

ModActiveGrid

ActiveGrid PHP Runtime

ActiveGrid Python Runtime

ActiveGrid Perl Runtime

ActiveGrid Caching

ModPHP

ModPython

ModPerl

MySQL

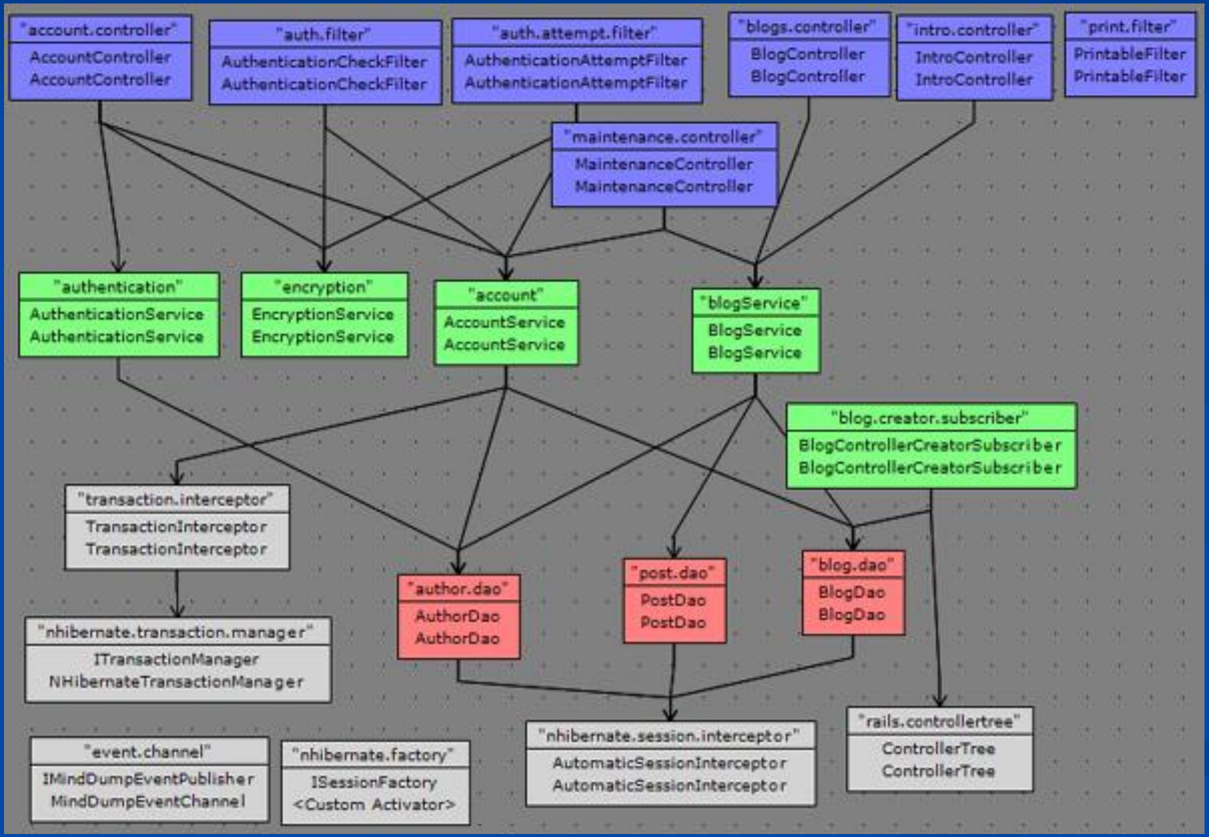
Apache

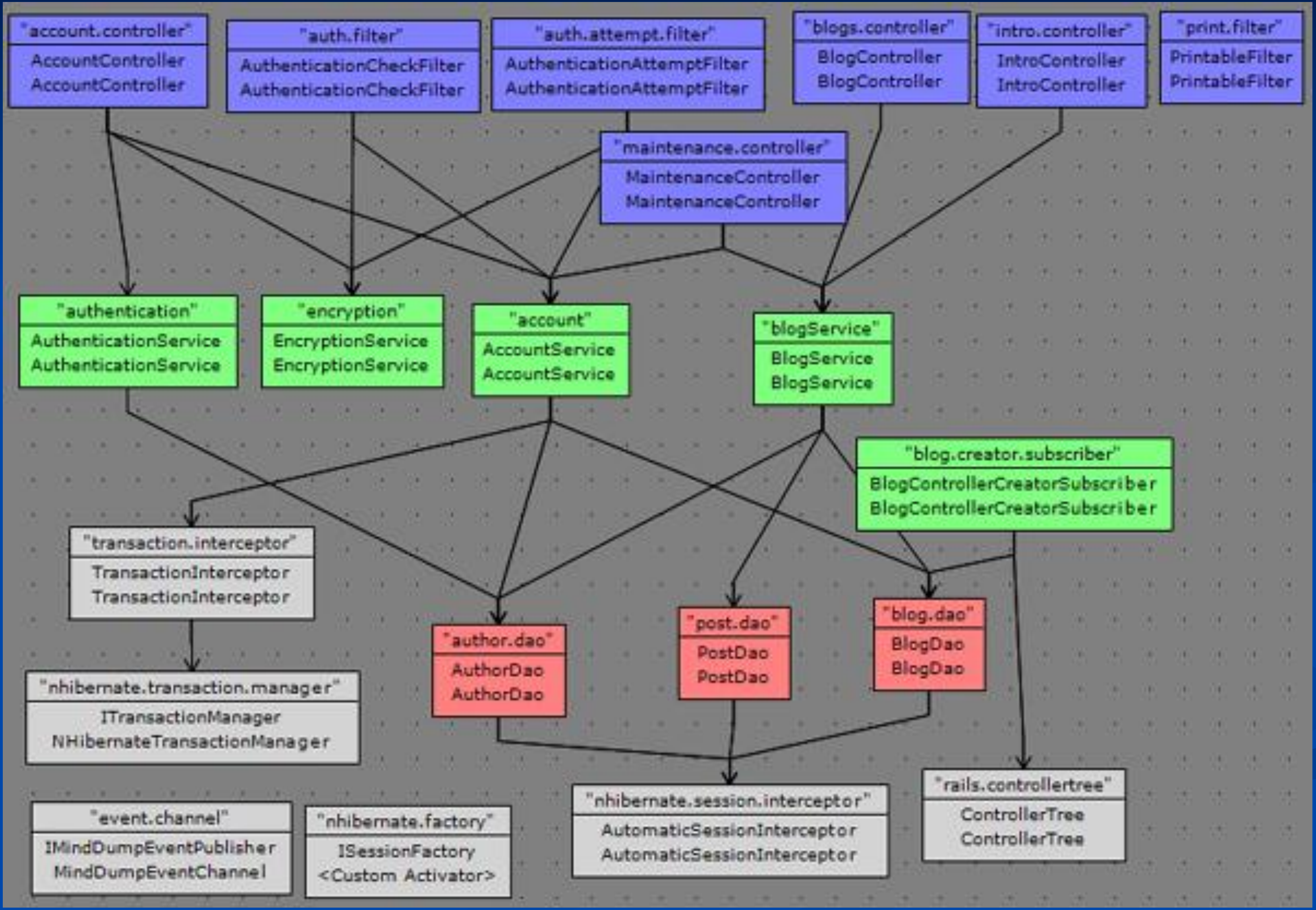
Linux

Commodity x86 Server

Castle

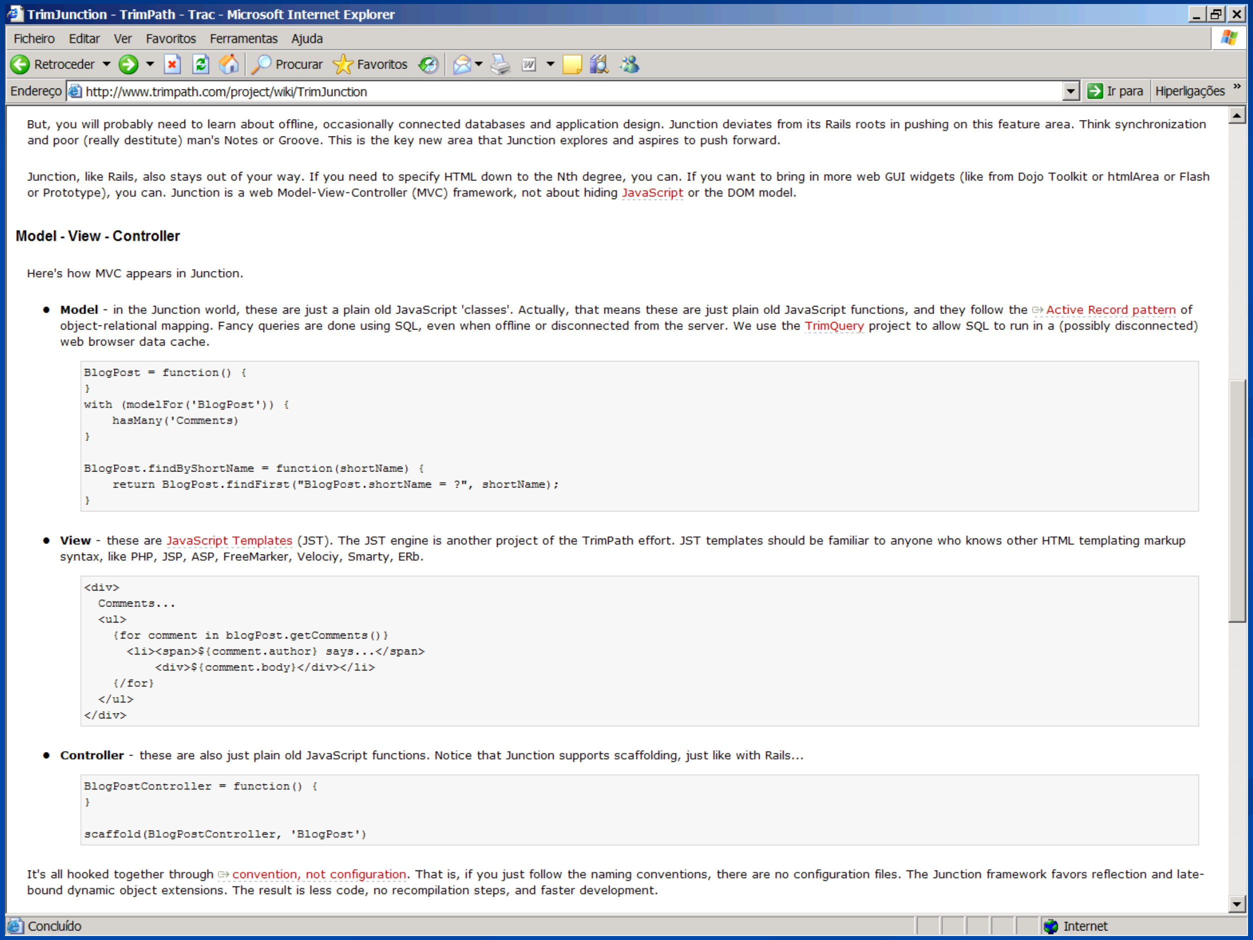
whatever




"account.controller"
AccountController
AccountController
"auth.filter"
AuthenticationCheckFilter
AuthenticationCheckFilter
"auth.attempt.filter"
AuthenticationAttemptFilter
AuthenticationAttemptFilter
"blogs.controller"
BlogController
BlogController
"intro.controller"
IntroController
IntroController
"print.filter"
PrintableFilter
PrintableFilter
"maintenance.controller"
MaintenanceController
MaintenanceController
"authentication"
AuthenticationService
AuthenticationService
"encryption"
EncryptionService
EncryptionService
"account"
AccountService
AccountService
"blogService"
BlogService
BlogService
"blog.creator.subscriber"
BlogControllerCreatorSubscriber
BlogControllerCreatorSubscriber
"transaction.interceptor"
TransactionInterceptor
TransactionInterceptor
"author.dao"
AuthorDao
AuthorDao
"post.dao"
PostDao
PostDao
"blog.dao"
BlogDao
BlogDao
"nhibernate.transaction.manager"
ITransactionManager
NHibernateTransactionManager
"rails.controllertree"
ControllerTree
ControllerTree
"nhibernate.session.interceptor"
AutomaticSessionInterceptor
AutomaticSessionInterceptor
"event.channel"
IMindDumpEventPublisher
MindDumpEventChannel
"nhibernate.factory"
ISessionFactory
<Custom Activator>

TrimPath Junction

Javascript


TrimJunction - TrimPath - Trac - Microsoft Internet Explorer
Ficheiro Editar Ver Favoritos Ferramentas Ajuda
Retroceder
Procurar
Favoritos
Endereço http://www.trimpath.com/project/wiki/TrimJunction
Ir para
Hiperligações
Concluído
Internet

But, you will probably need to learn about offline, occasionally connected databases and application design. Junction deviates from its Rails roots in pushing on this feature area. Think synchronization and poor (really destitute) man's Notes or Groove. This is the key new area that Junction explores and aspires to push forward.

Junction, like Rails, also stays out of your way. If you need to specify HTML down to the Nth degree, you can. If you want to bring in more web GUI widgets (like from Dojo Toolkit or htmlArea or Flash or Prototype), you can. Junction is a web Model-View-Controller (MVC) framework, not about hiding JavaScript or the DOM model.

Model - View - Controller

Here's how MVC appears in Junction.

- **Model** - in the Junction world, these are just a plain old JavaScript 'classes'. Actually, that means these are just plain old JavaScript functions, and they follow the ⇨Active Record pattern of object-relational mapping. Fancy queries are done using SQL, even when offline or disconnected from the server. We use the TrimQuery project to allow SQL to run in a (possibly disconnected) web browser data cache.

```
BlogPost = function() {
}
with (modelFor('BlogPost')) {
    hasMany('Comments')
}

BlogPost.findByShortName = function(shortName) {
    return BlogPost.findFirst("BlogPost.shortName = ?", shortName);
}
```

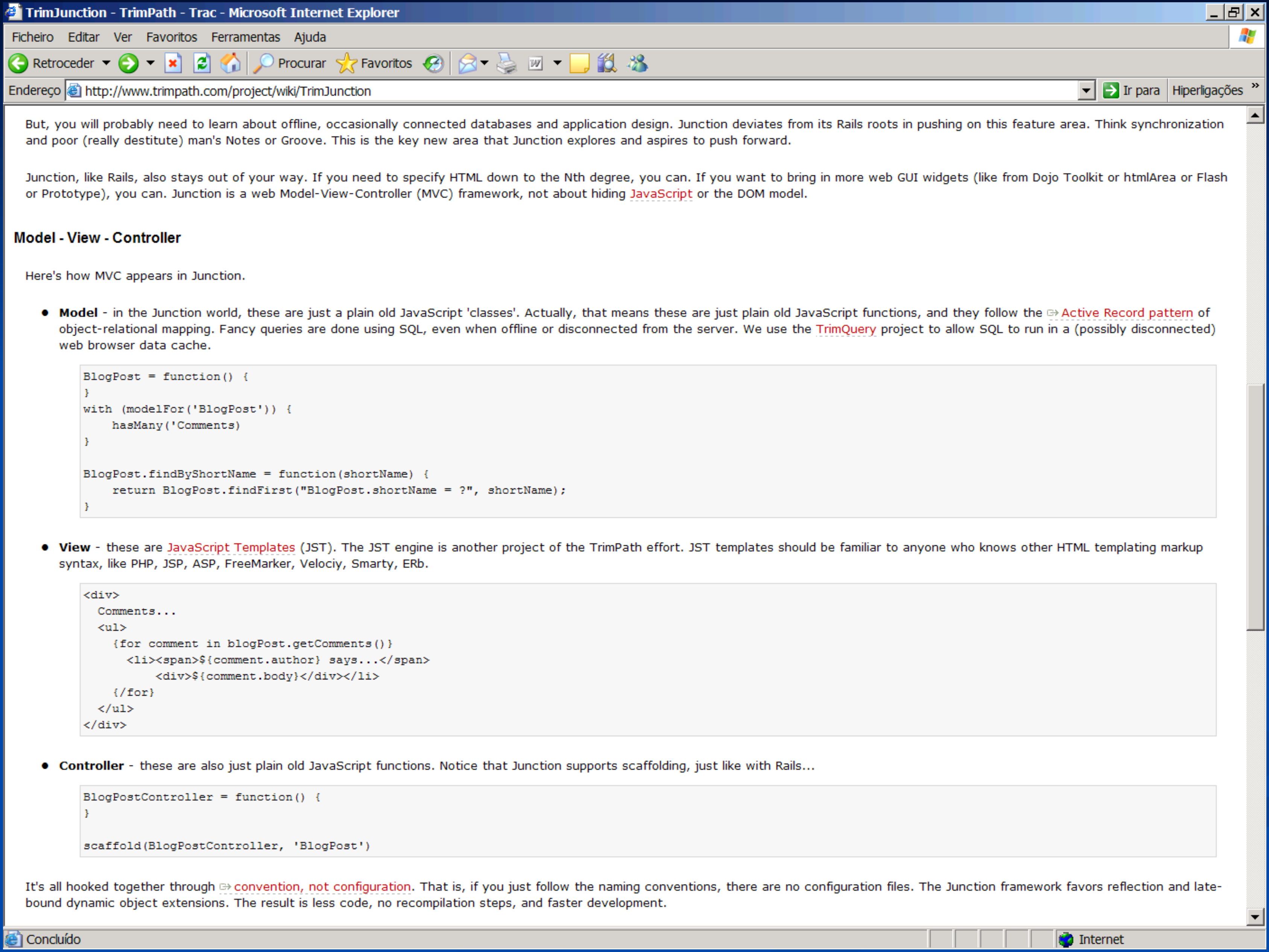
- **View** - these are JavaScript Templates (JST). The JST engine is another project of the TrimPath effort. JST templates should be familiar to anyone who knows other HTML templating markup syntax, like PHP, JSP, ASP, FreeMarker, Velociy, Smarty, ERb.

```
<div>
  Comments...
  <ul>
    {for comment in blogPost.getComments()}
      <li><span>${comment.author} says...</span>
          <div>${comment.body}</div></li>
    {/for}
  </ul>
</div>
```

- **Controller** - these are also just plain old JavaScript functions. Notice that Junction supports scaffolding, just like with Rails...

```
BlogPostController = function() {
}

scaffold(BlogPostController, 'BlogPost')
```

It's all hooked together through ⇨convention, not configuration. That is, if you just follow the naming conventions, there are no configuration files. The Junction framework favors reflection and late-bound dynamic object extensions. The result is less code, no recompilation steps, and faster development.



But, you will probably need to learn about offline, occasionally connected databases and application design. Junction deviates from its Rails roots in pushing on this feature area. Think synchronization and poor (really destitute) man's Notes or Groove. This is the key new area that Junction explores and aspires to push forward.

Junction, like Rails, also stays out of your way. If you need to specify HTML down to the Nth degree, you can. If you want to bring in more web GUI widgets (like from Dojo Toolkit or htmlArea or Flash or Prototype), you can. Junction is a web Model-View-Controller (MVC) framework, not about hiding JavaScript or the DOM model.

Model - View - Controller

Here's how MVC appears in Junction.

- **Model** - in the Junction world, these are just a plain old JavaScript 'classes'. Actually, that means these are just plain old JavaScript functions, and they follow the Active Record pattern of object-relational mapping. Fancy queries are done using SQL, even when offline or disconnected from the server. We use the TrimQuery project to allow SQL to run in a (possibly disconnected) web browser data cache.

```
BlogPost = function() {
}
with (modelFor('BlogPost')) {
    hasMany('Comments)
}

BlogPost.findByShortName = function(shortName) {
    return BlogPost.findFirst("BlogPost.shortName = ?", shortName);
}
```

- **View** - these are JavaScript Templates (JST). The JST engine is another project of the TrimPath effort. JST templates should be familiar to anyone who knows other HTML templating markup syntax, like PHP, JSP, ASP, FreeMarker, Velociy, Smarty, ERb.

```
<div>
  Comments...
  <ul>
    {for comment in blogPost.getComments()}
      <li><span>${comment.author} says...</span>
          <div>${comment.body}</div></li>
    {/for}
  </ul>
</div>
```

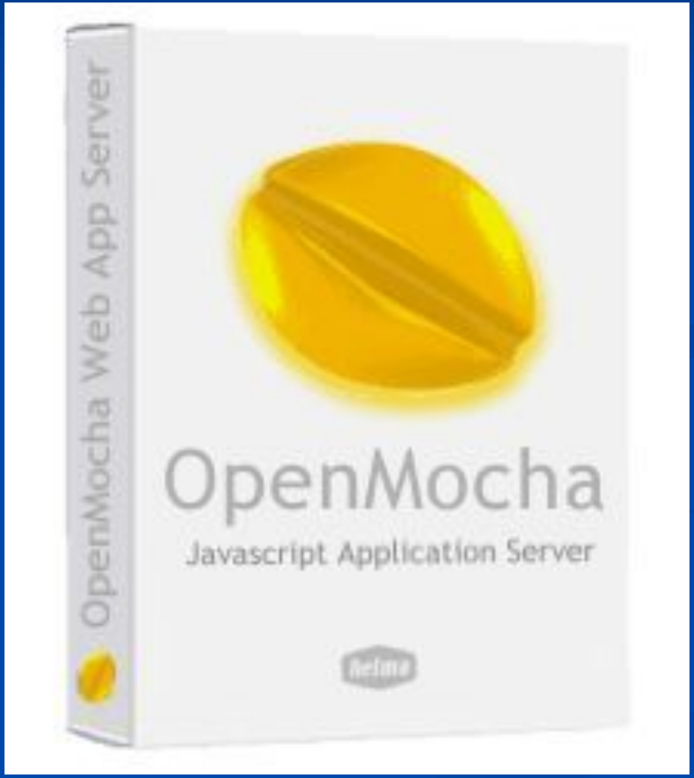
- **Controller** - these are also just plain old JavaScript functions. Notice that Junction supports scaffolding, just like with Rails...

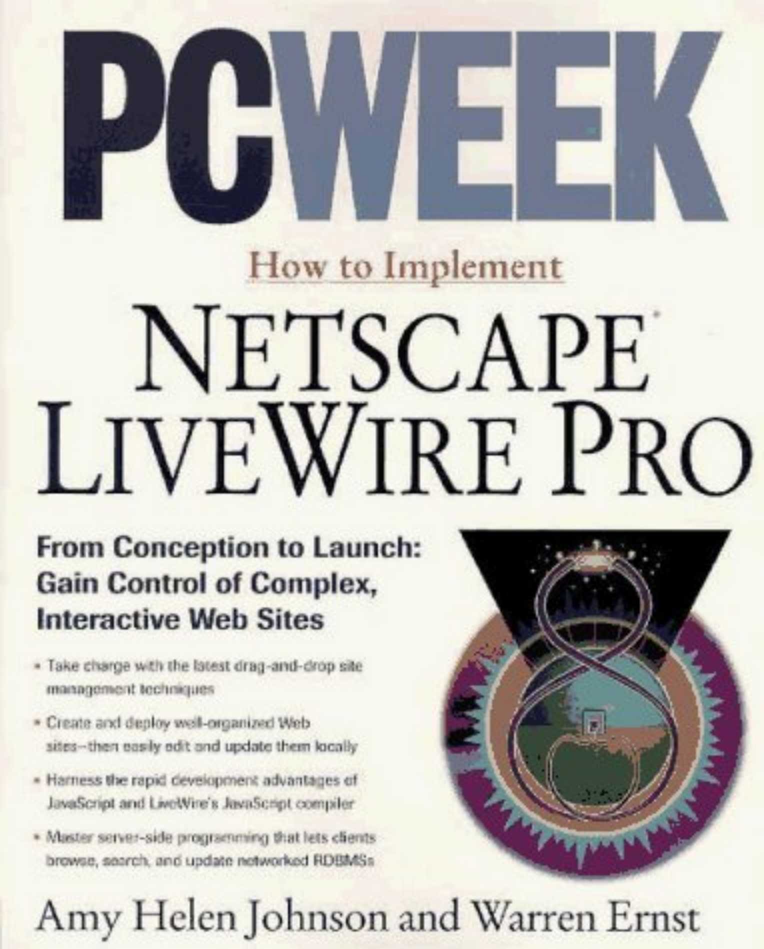
```
BlogPostController = function() {
}

scaffold(BlogPostController, 'BlogPost')
```

It's all hooked together through convention, not configuration. That is, if you just follow the naming conventions, there are no configuration files. The Junction framework favors reflection and late-bound dynamic object extensions. The result is less code, no recompilation steps, and faster development.

OpenMocha

Server Side Javascript





PCWEEK

How to Implement

NETSCAPE LIVEWIRE PRO

From Conception to Launch: Gain Control of Complex, Interactive Web Sites

- Take charge with the latest drag-and-drop site management techniques
- Create and deploy well-organized Web sites—then easily edit and update them locally
- Harness the rapid development advantages of JavaScript and LiveWire's JavaScript compiler
- Master server-side programming that lets clients browse, search, and update networked RDBMSs

Amy Helen Johnson and Warren Ernst