

Overview of Lift Web Framework

11, 12 Dec. 2009

Pune, India

IndicThreads.com Conference On Java Technology

Vikas Hazrati

www.xebiaindia.com

Today There is a Wide Choice of Web Frameworks

Struts²



tapestry



As Developers It is Hard to Make a Choice



Every Framework Has its Own Set of Pitfalls



But Quick Development is Still an Option





Why is Lift Better?

Convention over configuration

Clean separation of presentation content and logic

Leverage the Scala programming language

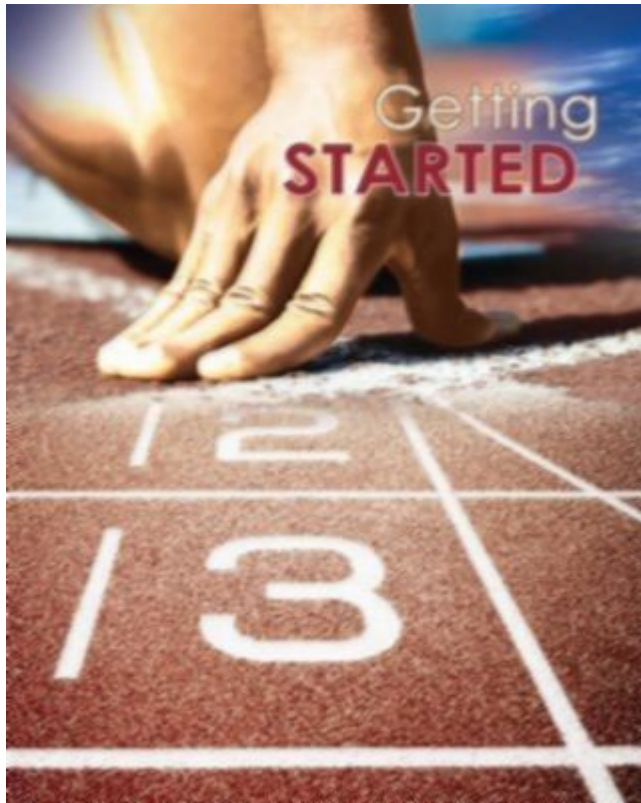


Responsive community

Concise code increases productivity


















Powerful AJAX & Comet Support

Highly Scalable



```
mvn archetype:generate -U \  
  -DarchetypeGroupId=net.liftweb \  
  -DarchetypeArtifactId=lift-archetype-blank \  
  -DarchetypeVersion=1.0 \  
  -DremoteRepositories=http://scala-tools.org/repo-  
releases \  
  -DgroupId=demo.helloworld \  
  -DartifactId=helloworld \  
  -Dversion=1.0-SNAPSHOT
```

```
cd helloworld  
mvn jetty:run
```

- ▼  helloworld
 - ▶  src/test/scala
 - ▶  src/test/resources
 - ▶  src/main/scala
 - ▶  src/main/resources
 - ▶  Referenced Libraries
 - ▶  JRE System Library [java-6-openjdk]
 - ▶  Scala Library version 2.7.7.final
- ▼  src
 - ▼  main
 - ▼  webapp
 - ▶  templates-hidden
 - ▶  WEB-INF
 - ▶  index.html
 - ▶  test
 - ▶  target
 - ▶  pom.xml

index.html

```
<lift:surround with="default" at="content">  
<h2>Welcome to your project!</h2>  
<p><lift:helloWorld.howdy /></p>  
</lift:surround>
```



```
class HelloWorld {  
  def howdy = <span>Welcome to helloworld at {new  
_root_.java.util.Date}</span>  
}
```

default.html

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:lift="http://liftweb.net/">
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-
8" />
<meta name="description" content="" />
<meta name="keywords" content="" />

<title>demo.helloworld:helloworld:1.0-SNAPSHOT</title>
<script id="jquery" src="/classpath/jquery.js"
type="text/javascript"></script>
</head>
<body>
<lift:bind name="content" />
<lift:Menu.builder />
<lift:msgs/>
</body>
</html>
```

Boot Class

```
package bootstrap.liftweb

import _root_.net.liftweb.util._
import _root_.net.liftweb.http._
import _root_.net.liftweb.sitemap._
import _root_.net.liftweb.sitemap.Loc._
import Helpers._

/**
 * A class that's instantiated early and run. It allows the
 * application
 * to modify lift's environment
 */
class Boot {
  def boot {
    // where to search snippet
    LiftRules.addToPackages("demo.helloworld")

    // Build SiteMap
    val entries = Menu(Loc("Home", List("index"), "Home")) ::
Nil
    LiftRules.setSiteMap(SiteMap(entries:_*))
  }
}
```

Lift Entry Point

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
<filter>
  <filter-name>LiftFilter</filter-name>
  <display-name>Lift Filter</display-name>
  <description>The Filter that intercepts lift
calls</description>
  <filter-class>net.liftweb.http.LiftFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>LiftFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

</web-app>
```

Lift's Main Objects

`net.liftweb.http.S`

`net.liftweb.http.SHtml`

`net.liftweb.http.LiftRules`

Template

Snippets

Boot
class

Model

Adding AJAX
Spice

Model

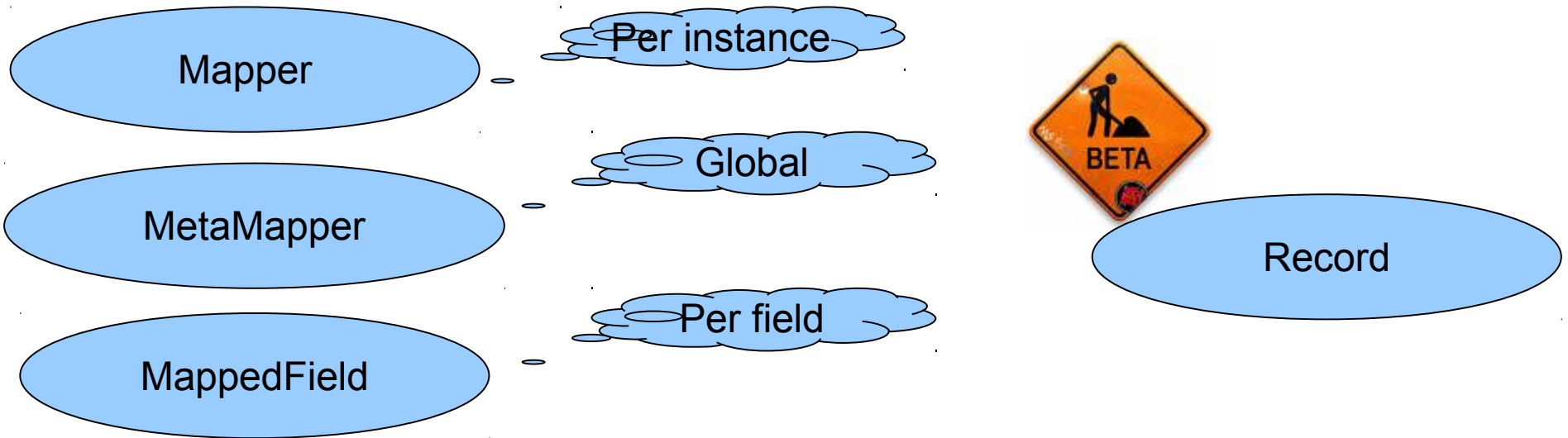


Maps to rdbms

```
class ToDo extends LongKeyedMapper[ToDo] with IdPK {  
  def getSingleton = ToDo  
  object done extends MappedBoolean(this)  
  object owner extends MappedLongForeignKey(this, User)  
  object priority extends MappedInt(this) {  
    override def defaultValue = 5  
  }  
  object desc extends MappedPoliteString(this, 128)  
}  
object ToDo extends ToDo with LongKeyedMetaMapper[ToDo]
```

Provides meta functionality
Like finders etc

Mapper and Record Frameworks



```
<project ...>  
...  
<dependencies>  
...  
<dependency>  
  <groupId>net.liftweb</groupId>  
  <artifactId>lift-mapper</artifactId>  
  <version>1.0</version> <!-- or 1.1-SNAPSHOT, etc -->  
</dependency>  
</dependencies>  
...  
</project>
```

Database Connection

```
class Boot {  
  def boot {  
    ...  
    DB.defineConnectionManager(DefaultConnectionIdentifier, DBVendor)  
    Schemifier.schemify(true, Log.infoF _, User, ToDo)
```

```
import _root_.net.liftweb.mapper._  
import _root_.java.sql._  
object DBVendor extends ConnectionManager {  
  // Force load the driver  
  Class.forName("org.postgresql.Driver")  
  // define methods  
  def newConnection(name : ConnectionIdentifier) = {  
    try {  
      Full(DriverManager.getConnection(  
        "jdbc:postgresql://localhost/mydatabase",  
        "root", "secret"))  
    } catch {  
      case e : Exception => e.printStackTrace; Empty  
    }  
  }  
  def releaseConnection (conn : Connection) { conn.close }
```

```
import _root_.java.math.MathContext
class Expense extends LongKeyedMapper[Expense] with IdPK {
  def getSingleton = Expense
  object dateOf extends MappedDateTime(this)
  object description extends MappedString(this, 100)
  object amount extends MappedDecimal(this, MathContext.DECIMAL64, 2)
  object account extends MappedLongForeignKey(this, Account)
}
```

```
create          save          delete          count          countByInsecureSQL

findAll        findAllByInsecureSQL    findAllByPreparedStatement

findAllFields
```

Templates



Menu and content dynamic



Menu fixed, Content dynamic



Menu and content dynamic



3 columns, all dynamic



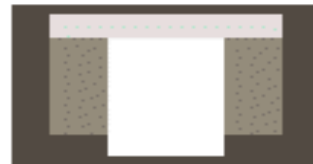
4 columns, all dynamic



Menu floating



Menu fixed, content & header dynamic



3 columns fixed centered



dynamic with header and footer

Templates

Lift tag

```
<lift:surround with="default" at="content">  
  <head><title>Hello!</title></head>  
  <lift:Hello.world />  
</lift:surround>
```

Snippet

Views

```
class ExpenseView extends LiftView {  
  override def dispatch = {  
    case "enumerate" => doEnumerate _  
  }  
  def doEnumerate () : NodeSeq = {  
    ...  
  
    <lift:surround with="default" at="content">  
      { expenseltems.toTable }  
    </lift:surround>  
  }  
}
```

Tags

```
<lift:snippet type="MyClass:render" />  
<lift:MyClass.render />  
<lift:MyClass />
```

snippet

```
<lift:surround with="template_name" at="binding">  
  children  
</lift:surround>
```

surround

```
<lift:bind name="binding_name" />
```

bind

```
<div class="accountUpdates">  
  < lift : comet type="AccountMonitor">  
    <ul><account:entries>  
      <li><entry:time /> : <entry:user /> :  
<entry:amount /></li>  
    </account:entries></ul>  
  </ lift : comet>  
</div>
```

comet

Snippets



View

Logic

```
<lift:Util.out>
  Please Log In <b>Dude</b>
</lift:Util.out>
```

Lift tags

```
package com.liftworkshop.snippet
import scala.xml.{NodeSeq}
import com.liftworkshop._
import model._
class Util {
  def in(html: NodeSeq) =
    if (User.loggedIn_?) html else NodeSeq.Empty
  def out(html: NodeSeq) =
    if (!User.loggedIn_?) html else NodeSeq.Empty
}
```

Corresponding
Snippet
code

View code	Class	Method
<code><lift:foo/></code>	Foo	render
<code><lift:FooBar/></code>	FooBar	render
<code><lift:foo_bar/></code>	FooBar	render
<code><lift:foo_bar.baz/></code>	FooBar	baz

Snippets

```
class Ledger {  
  def balance (content : NodeSeq) : NodeSeq =  
  Text(currentLedger.formattedBalance)  
}
```

```
class Ledger {  
  def balance (content : NodeSeq) : NodeSeq =  
    <p>{currentLedger.formattedBalance}  
    as of <lift:Util.time /></p>  
}
```

```
<lift:Ledger.balance>  
<ledger:balance/> as of <ledger:time />  
</lift:Ledger.balance>
```

```
class Ledger {  
  def balance (content : NodeSeq ) : NodeSeq =  
    bind ("ledger", content,  
          "balance" -> Text(currentLedger.formattedBalance),  
          "time" -> Text((new java.util.Date).toString))  
}
```

Snippets are Stateless



Cookies

SessionVar

RequestVar

StatefulSnippet
Subclass

Form Processing

Please complete the form below. Mandatory fields marked *

Delivery Details

Name *	<input type="text"/>
Address *	<input type="text"/>
Town/City	<input type="text"/>
County *	<input type="text"/>
Postcode *	<input type="text"/>
Is this address also your invoice address? *	
<input type="radio"/> Yes	
<input type="radio"/> No	

Post/Get

AJAX

JSON

Form Processing

```
<html>
...
<lift:Show.myForm form="POST">
  <tr>
    <td>Name</td>
    <td><f:name><input type="text"/></f:name></td>
  </tr>
  <tr>
    <td>Birthyear</td>
    <td><f:year>
      <select><option>2007</option></select>
    </f:year></td>
  </tr>
  <tr>
    <td>&nbsp;</td>
    <td><input type="submit" value="Add"/></td>
  </tr>
</lift:Show.myForm>
</html>
```


Goodies

ProtoUser and MegaProtoUser



```
class User extends ProtoUser[User] {  
  override def shortName = firstName.is  
  override def lastNameDisplayName = "surname"  
}
```

AJAX and Comet

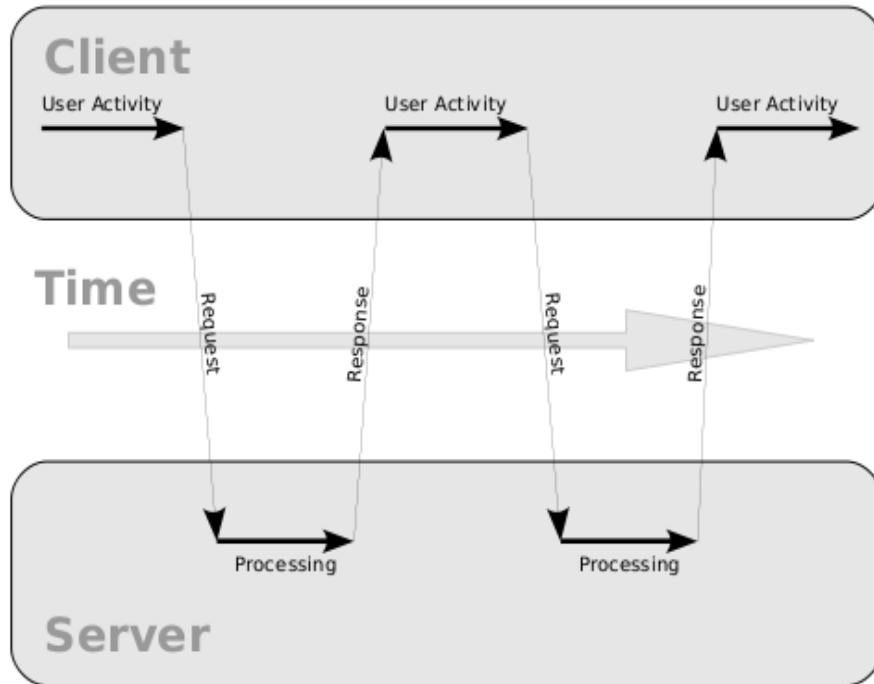


JavaScript

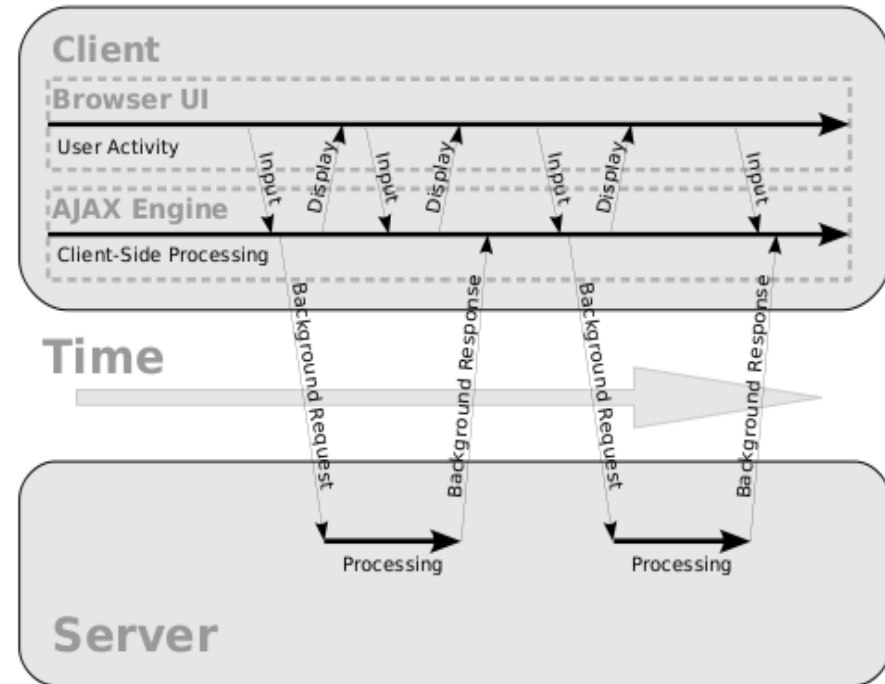
```
import JsCmds._
import JE._
var myName = ""
bind(...
  "name" -> text(myName, myName = _, "id" -> "myName"),
  "submit" -> submit("Save", ..., "onclick" ->
    JsIf(JsEq(ValById("myName"), ""),
      Alert("You must provide a name") & JsReturn(false))
    )
)
```

```
import net.liftweb.http.js.yui.YUIArtifacts
class Boot {
  def boot = {
    ...
    LiftRules.jsArtifacts = YUIArtifacts
    ...
  }
```

AJAX and Comet



(a) Traditional Application Model



(b) AJAX Application Model

Comet Based on Scala Actors

```
class Clock extends ControllerActor {  
  ActorPing.schedule(this, Tick, 10000L)  
  
  def render = bind("time" -> Text(timeNow.toString))  
  
  override def lowPriority: PartialFunction[Any, Unit] = {  
    case Tick =>  
      reRender  
      ActorPing.schedule(this, Tick, 10000L)  
  }  
}
```

schedule a Tick
message in 10 seconds

renders Controller and sends
update to all Page Actors on
which Controller exists

AJAX

```
</lift:TD.list all_id="all_todos">
<div id="all_todos">
  <div>Exclude done <todo:exclude/></div>
  <ul>
    <todo:list>
      <li>
        <todo:check><input type="checkbox"/></todo:check>
        <todo:priority>
          <select><option>1</option></select>
        </todo:priority>
        <todo:desc>To Do</todo:desc>
      </li>
    </todo:list>
  </ul>
</div>
</lift:TD.list>
```

```

def list(html: NodeSeq) = {
  val id = S.attr("all_id").open_!
  def inner(): NodeSeq = {
    def reDraw() = SetHtml(id, inner())
    bind("todo", html,
        "exclude" ->
            ajaxCheckbox(QueryNotDone, v =>
{QueryNotDone(v); reDraw}),
        "list" -> doList(reDraw) _)
  }
  inner()
}

private def doList(reDraw: () => JsCmd)(html: NodeSeq):
NodeSeq =
  toShow.
  flatMap(td =>
    bind("todo", html,
        "check" -> ajaxCheckbox(td.done,
            v => {td.done(v).save; reDraw()}),
        "priority" ->
            ajaxSelect(ToDo.priorityList,
Full(td.priority.toString),
            v => {td.priority(v.toInt).save;
reDraw()}),
        "desc" -> desc(td, reDraw)
    ))

private def desc(td: ToDo, reDraw: () => JsCmd) =
swappable(<span>{td.desc}</span>,
    <span>{ajaxText(td.desc,
        v => {td.desc(v).save; reDraw()})}
    </span>)

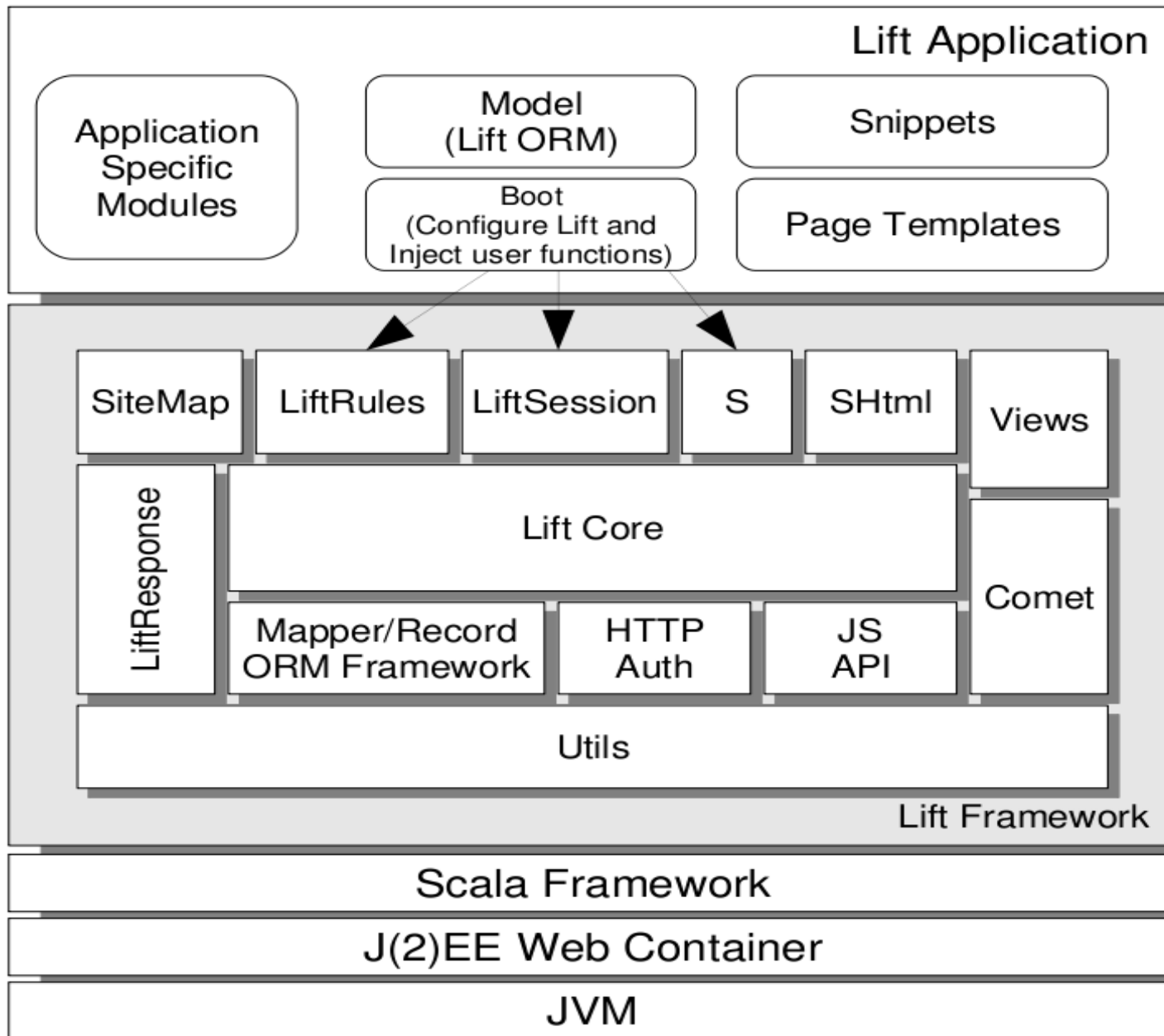
```

Comet

```
<lift:surround with="default" at="content">  
  <lift:comet type="Clock" name="Other">  
    Current Time: <clk:time>Missing Clock</clk:time>  
  </lift:comet>  
</lift:surround>
```

```
class Clock extends CometActor {  
  override def defaultPrefix = Full("clk")  
  def render = bind("time" -> timeSpan)  
  def timeSpan = (<span id="time">{timeNow}</span>)  
  // schedule a ping every 10 seconds so we redraw  
  ActorPing.schedule(this, Tick, 10000L)  
  override def lowPriority : PartialFunction[Any, Unit] = {  
    case Tick => {  
      println("Got tick " + new Date());  
      partialUpdate(SetHtml("time", Text(timeNow.toString)))  
      // schedule an update in 10 seconds  
      ActorPing.schedule(this, Tick, 10000L)  
    }  
  }  
}  
case object Tick
```

Lift Architecture



Rails v/s Lift

For single request processing, the lift code, running inside Tomcat, ran **4 times faster** than the Rails code running inside Mongrel. However, the **CPU utilization was less than 5% in the lift version, where it was 100% of 1 CPU (on a dual core machine) for the Rails version.** For multiple simultaneous requests being made from multiple machines, we're seeing **better than 20x performance of the lift code versus the Rails code** with 5 Mongrel instances. Once again, the lift code is not using very much CPU and the Rails code is pegging both CPUs.

<http://lambda-the-ultimate.org/node/2147>

Lots of Choices

Struts²



tapestry



