Nuxeo **Web**Engine unveiled

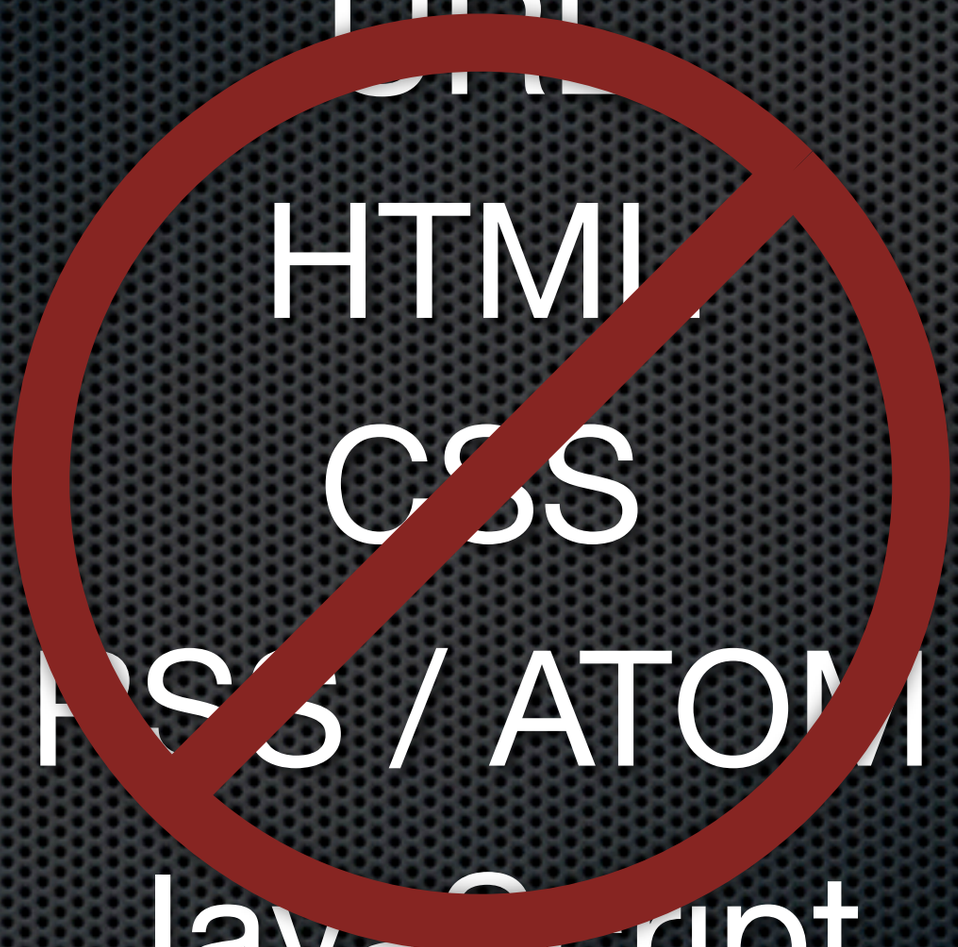
The content-centric web framework

*"Nuxeo WebEngine is a **lightweight**, **content-centric** web framework to quickly build and deliver **slices of web**"*

what the **web**
really is about?

URL
HTML
CSS
RSS / ATOM
JavaScript

URL



HTML

CSS

RSS / ATOM

JavaScript

content

content

content

why WebEngine?

because **URLs** matter

because you have **content**

to **expose** on the **web**

because you want **more**

than web publishing

content repository
ecm platform
many **services** and features
flexible and modular
components everywhere

do you need **anything else**
to **create** your **content**-
oriented **web apps**?

sure!

dynamic
horizontal **scaling**
easy programing model
designed for the **web**

and what about **that**?

lightweight framework
content-centric
extensible with **scripting**
components architecture
lightspeed startup

Nuxeo **Web**Engine

less abstraction, more control

Overview

- Easier and **faster** development for **content-oriented** web **applications**
- leverage a complete **ECM platform**
- ...and a powerful **component model**

less abstraction

- focus on content
- built on REST (HTTP means something)
- no JSF, EJBs, Java EE required
- no magic

more control

- the **browser** is a platform
- **widgets** are the key to expose your **content**
- **scripting** for business logic
- **fast** code/test cycle

smart & elegant

Designed by web's children, for the web you like

- Content is **king**
- URL **matters**
- **REST** everywhere, because the web has a **soul**
- and... in developers we **trust**

WebEngine Essentials

- easy **MVC**
- **smart** URL dispatcher
- **flexible** views on content
- powerful templating language — Freemarker
- **scripting** for logic — Groovy, JS, Python, Ruby, etc.

a real MVC model

Model

Content

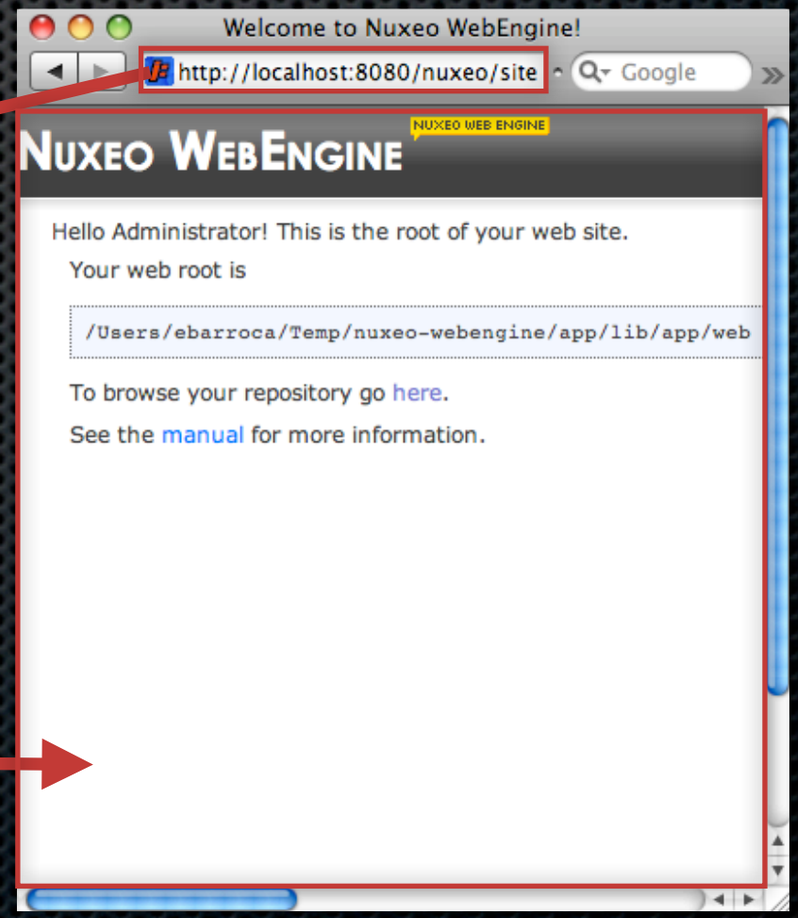
DocumentModel
from Nuxeo Core

Controler

Scripts
or Java Code

View

Templates
(Freemarker)



smart URLs

`/articles/cars/porsche/cs/overview@@view?param=value`

app ID | **content path** (in the repository) | **action selector** | **parameters**

or **simply**:

`/articles/cars/bmw/s3/overview`

(because the *view* action is implied)

smart URLs: mappings

- You can also define mappings to control your URLs

```
<mapping pattern="/user/(?username:[a-zA-Z0-9]+)$">
    <script>/users/user_detail.groovy</script>
</mapping>
```

Will make the URL `/user/JohnDoe` call the script `user_detail.groovy`, with the variable `username` (having the value“`JohnDoe`”) automatically passed to it.

@@actions

- an action...
  - points to a template, a script or a Java class
  - has a guard (ex: permission)
  - is bound to a *content type*
  - belongs to a *category*
  - is prefixed in the URL by **@@** (ex: *@@print*)
- actions represent a **powerful** way to bind **views** and **logic** to **content**

@@actions

```
<object id="WikiPage" extends="WikiObject">
  <actions>
    <action id="view_content"
        enabled="true">
      <permission expression="Read"/>
      <category>tab</category>
    </action>

    <action id="show_comments"
        enabled="true"
        script="show_comments.groovy">
      <permission expression="Read"/>
      <category>tab</category>
    </action>
  </actions>
</object>
```

New action **view_content** on **WikiObjects**.
WebEngine will look for a template called **view_content.ftl** and use it as view.

New action **show_comments** on **WikiObjects**.
As a script is defined, WebEngine will execute the script show_comments.groovy and return the result.

#templates

- based on the **FreeMarker** engine
- **template inheritance**
- **easy** access to content
- extensible context (variable **injection**)
- and... extensible Nuxeo's style! You can use your **preferred** template engine (PHP anyone?)

Scripting

- scripts can access all services of the **Nuxeo Platform services**
- **several** scripting **languages** included — Groovy, Python, JavaScript, Ruby, etc. (thank you, JSR-233!)
- **easy logic**, powerful features
- and... the strength of the **Java VM**

here is how you list the comments on a document

```
//get the current document as commentable document
cdoc = Document.getAdapter(CommentableDocument)

//render the template passing named variables
Context.render("comments/show_comments.ftl",
                ['comments': cdoc.comments])
```

or perform a query

```
//define your query
pquery = "SELECT * FROM Document WHERE
          (dc:created BETWEEN DATE '2008-01-01' AND DATE '2008-12-31')
          AND (ecm:path STARTSWITH '/')"

//perform the query and get the results
results = Request.query(pquery)
```

or list a folder's content

```
//Document is the current content object
docchildren = Document.children
```

MVC in your hands

DocumentModel
(from the core)

Model

@@action

Controler

#template

View

Based on a complete ECM platform

- **content store** (Nuxeo Core, JCR-based)
- advanced **Access Control** (through permissions)
- **enterprise-class** authentication & user/group management
- indexing and **search**
- comments, relations, etc.
- and a **dozen more**! ;-)

Components everywhere

- hot-reloadable **extension points**
- **compose** your apps **dynamically** with **plugins**
- **works** with Jetty, JBoss and GlassFish3
- and... **Nuxeo style**! :-)

showtime

```
<@extends src="wiki/base.ftl">

<@block name="content">
<div class="summary-entries">
<#list Document.children?reverse as entry>
  <div class="summary-entry">
    <h2 class="summary-title"><a
href="${Root.urlPath}/${entry.name}">${entry.title}</a></h2>
    <div class="summary-content">
        <@wiki>${entry.wikiPage.content}</@wiki>
    </div>
  </div>
  <div class="summary-byline">${entry.dublincore.modified?datetime} by
${entry.dublincore.creator}</div>
</#list>
</div>

</@block>
</@extends>
```

http://localhost:8080/nuxeo/site/docs/index.ftl

Google

Nuxeo WebEngine Documentation

NUXEO WEB ENGINE

Note that this is the documentation embedded within the Nuxeo Web Engine and may not be up to date. It is only provided for convenience. To access the full documentation please visit http://www.nuxeo.org/webengine

Hi Guest! You are not logged in.

Username ••••••••

Log In

Help

- About
- Documentation

Nuxeo WebEngine

The WebEngine is a framework to build web applications on top of the Nuxeo ECM product.

1 Overview

The main goal for the WebEngine framework is to let users to build web applications quickly and easly and in a traditional way without having knowledge on advanced Java topics such as JSF or Seam.
The second goal is to be able to develop web pages while the server is running, without the need of restarting the server each time the configuration or the code is modified.

The web pages are build using Freemarker templates for the presentation part and optionally scripts (like groovy, jython etc) or custom Java request handlers for the logical part.
Thus a web application is made from:

1. **Configuration** - based on the nuxeo extension points mechanism
2. **Templates** - freemarker is used as the template engine
3. **Scripts** - any script supported by java scripting engine.
4. **Custom Java Code** - plugged-in through extension points

Here is an example of a simple template page that is outputing Q:My name is CURRENT_USER_NAME:

```
My name is ${Context.principal.name}!
```

The same can be done from a script: (in this example a groovy script)

```
Context.print("My name is ${Context.principal.name}")!
```

1.1 Bridging with Nuxeo ECM

The WebEngine was created to work over a Nuxeo ECM server. Using it outside Nuxeo ECM is possible but is very limiting and you will loose many of the cool features of the WebEngine. If you don't plan to use a Nuxeo ECM as the

Thank you!

www.nuxeo.com

www.nuxeo.org/webengine

www.nuxeo.org/discussions