Advanced Apache Roller

Dave Johnson
Sun Microsystems, Inc.

Goals of session

Understand how to setup and configure Roller for best performance and user experience.

Learn where to find resources and documentation for customizing your Roller installation(s).

Agenda: 50 minutes to discuss

- Advanced installation
- Page and feed caching
- Authentication options
- Deployment architecture
- Caching with Memcached
- Scripting and automation
- Plugging in new functionality

Caveats, disclaimers, etc.

- Everything applies to Apache Roller 4.0
  - The most recent official ASF release
- Plugins discussed (memcached, Groovy, etc.) are *not* official Apache releases
- Your mileage may vary
  - Don't tase me bro!

RTFM

- Roller has pretty extensive documentation
  - Installation Guide
  - User Guide
  - Template Guide
- Download them here:
  - http://roller.apache.org/download.cgi
- I'm not going to repeat what's in the docs

Installation is now pretty easy

- There's even a five minute install
  - Download Roller
  - Create a database
  - Create a `roller-custom.properties` file
  - Add JDBC driver and mail jars
  - Deploy & start Roller
  - Roller creates/updates tables automatically
- But it bypasses container management

Simple database & mail config

- No need to click around your App Server's console or hack its configuration files.
- Add database and mail setup to your `roller-custom.properties` file.

```
installation.type=auto
database.configurationType=jdbc
database.jdbc.driverClass=com.mysql.jdbc.Driver
database.jdbc.connectionURL=\
    jdbc:mysql://localhost:3306/rollerdb
database.jdbc.username=scott
database.jdbc.password=tiger
mail.configurationType=properties
mail.hostname=smtp-server.example.com
mail.username=scott
mail.password=tiger
```

A little too easy?

- Easy install bypasses container management of resources including:
  - JDBC connections and pools
  - Mail sessions
  - Authentication
- This is not necessarily the best idea

Advanced Installation

- To use container managed resources, leave the database setup out of your `roller-custom.properties` file.
- If you want to use your own JNDI names

```
database.jndi.name=jdbc/MyResourceName
mail.jndi.name=mail/MyMailSessionName
```

- See Install Guide Tips and Tricks section
- What about Container Managed Auth?
  - Roller 4.1 will make CMA possible

TIP: customize default blogroll

- We appreciate all the links!

```
newuser.blogroll=\
Dave Johnson|http://rollerweblogger.org/roller,\
Matt Raible|http://raibledesigns.com/page/rd,\
Lance Lavandowska|http://rollerweblogger.org/lance,\
Elias Torres|http://torrez.us/,\
Jeff Blattman|http://blogs.sun.com/jtb,\
blogs.sun.com|http://blogs.sun.com,\
jroller.com|http://jroller.com
```

- But you're not required to pay homage to the Roller development team so change them to links relevant to your site

Photo by http://flickr.com/photos/fastlanedaily

Page and feed caching

- Caches are critical to Roller performance
  - Rendering a blog page is expensive
  - Dozens of database queries required
- Configure Roller's caches for best results
  - Defaults are for 100 blog system
  - Built in cache is in-memory LRU
  - That may not be what you want

Built-in cache implementations

- Roller includes two in-memory Least Recently Used (LRU) cache implementations:
  - Expiring LRU: entries expire after timeout
  - Non-expiring LRU: entries never expire

- Default cache is Expiring LRU

```
cache.defaultFactory=\
org.apache.roller.weblogger.util.cache.ExpiringLRUCacheFactoryImpl
```

Roller's four caches

- Four separately configurable caches
  - **weblogpage**: for all weblog pages
  - **weblogfeed**: for weblog feeds
  - **sitewide**: for front-page blog and feeds
  - **planet**: for aggregator feeds
- For each you can set:
  - **enabled**: turn on and off the cache
  - **size**: max number of cache entries to store
  - **timeout**: cache entry timeout in seconds
  - **factory**: override default cache factory

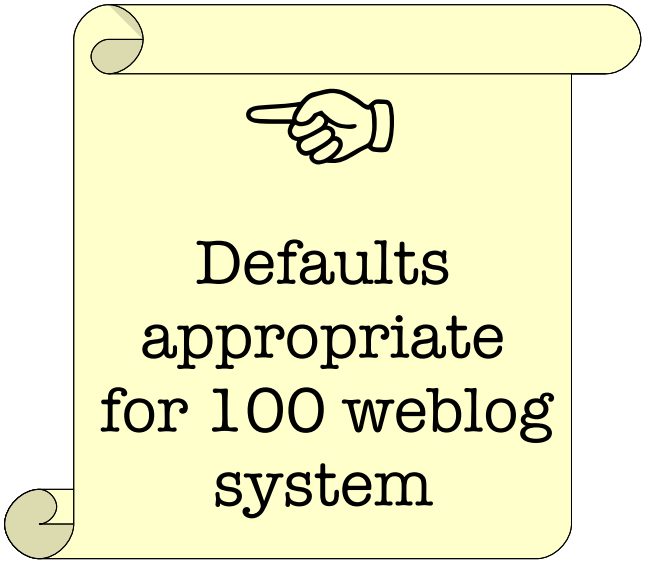
Default cache settings

```
cache.sitewide.enabled=true
cache.sitewide.size=50
cache.sitewide.timeout=1800

cache.weblogpage.enabled=true
cache.weblogpage.size=400
cache.weblogpage.timeout=3600

cache.weblogfeed.enabled=true
cache.weblogfeed.size=200
cache.weblogfeed.timeout=3600

cache.planet.enabled=true
cache.planet.size=10
cache.planet.timeout=1800
```



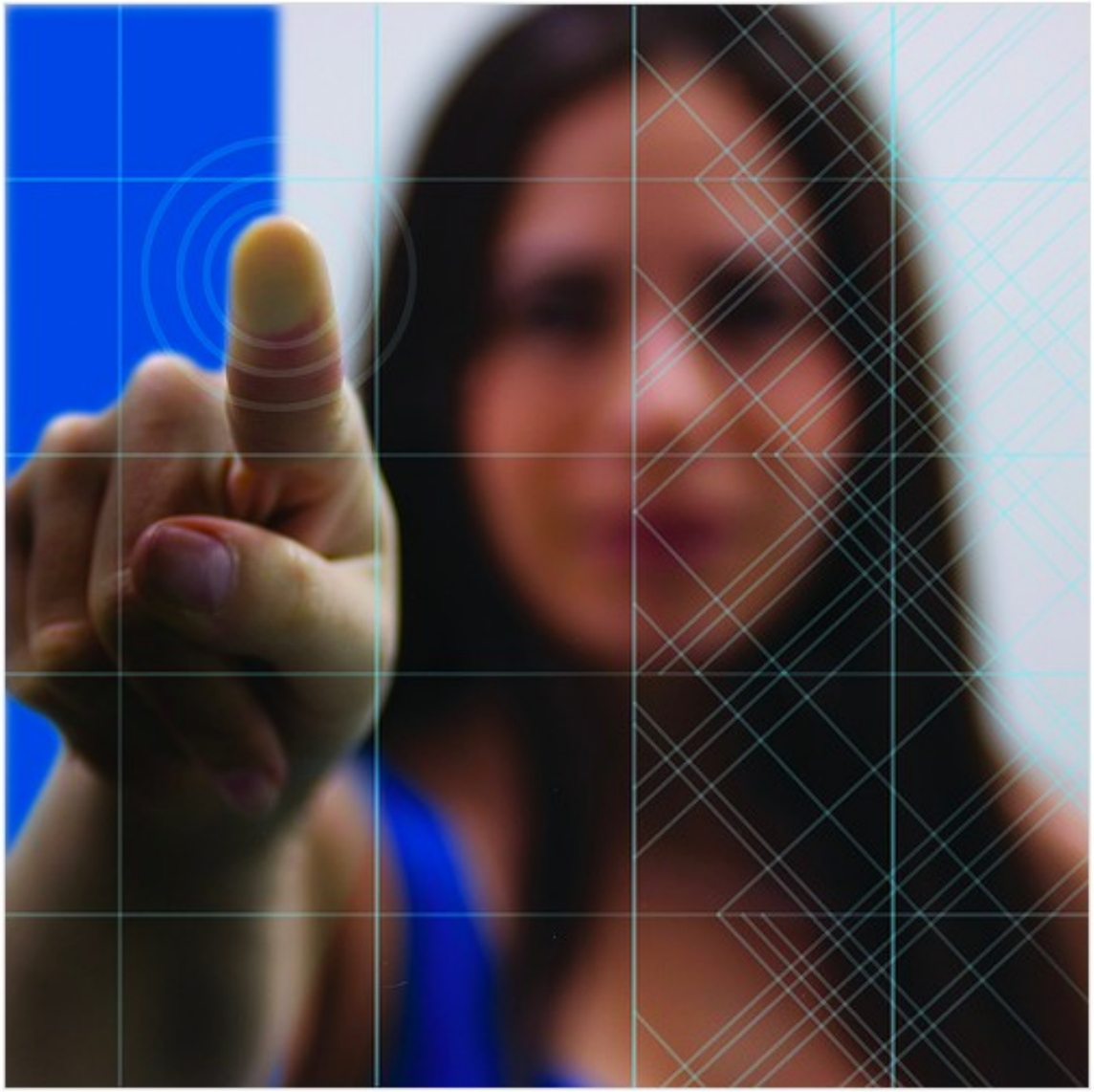
Caching: Conclusion

- For small sites, defaults will do
- For larger sites, set larger cache sizes
- For really big sites, consider memcached
  - We'll discuss it later...

TIP: choose themes carefully

- You can get additional themes at:
  - http://roller.dev.java.net
  - http://rollerthemes.com

- **But beware!** If you're running a big Roller site and supporting lots of users, it's easier to support small set of quality themes. So choose carefully.
- If you want to be able to control user's themes then consider disabling theme customization.



By Tio http://flickr.com/photos/66179962@N00

Authentication options

- Default is to authenticate against database
  - Via the `rolleruser` and `userrole` tables
- You can change that, but you have to understand Spring Acegi to do so
  - Acegi config is `WEB-INF/security.xml`
  - You can setup SSO and use LDAP
    - See Matt Raible's HOWTO
    - http://cwiki.apache.org/confluence/x/Yg4B

Why Spring Acegi?

- Roller uses Spring Acegi for Authentication
  - Bypassing Container Managed Authentication
- But why? Acegi makes installation easier
  - No need to setup “realm” in App Server
  - For most users no changes necessary

Auth. & Auth. Limitations

- No support for password reset
  - But admins can do it via UI
- No way to delete users via UI
  - But you can disable them
- No support for private blogs

User Management changes in 4.1

- We'll make it possible to use CMA
- And to plugin your own:
  - User management
  - Permissions management
- And hopefully we'll address some of those limitations... wanna help?

TIP: separate themes directory

- By default themes are stored in Roller's context directory under `/themes`

```
# The directory in which Roller will look for themes
themes.dir=${webapp.context}
```

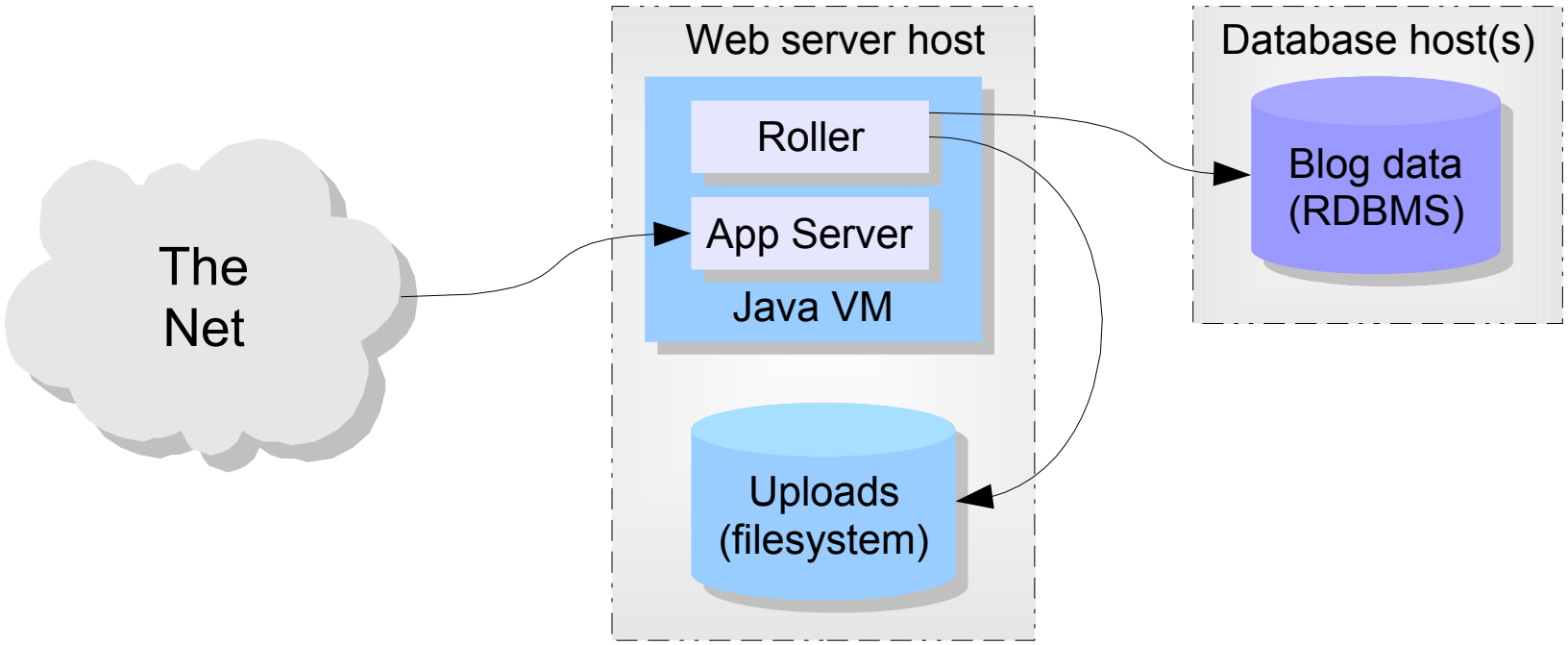
- If you've got your own customized themes, then protect them from upgrade by keeping them in a separate directory.

Deployment options

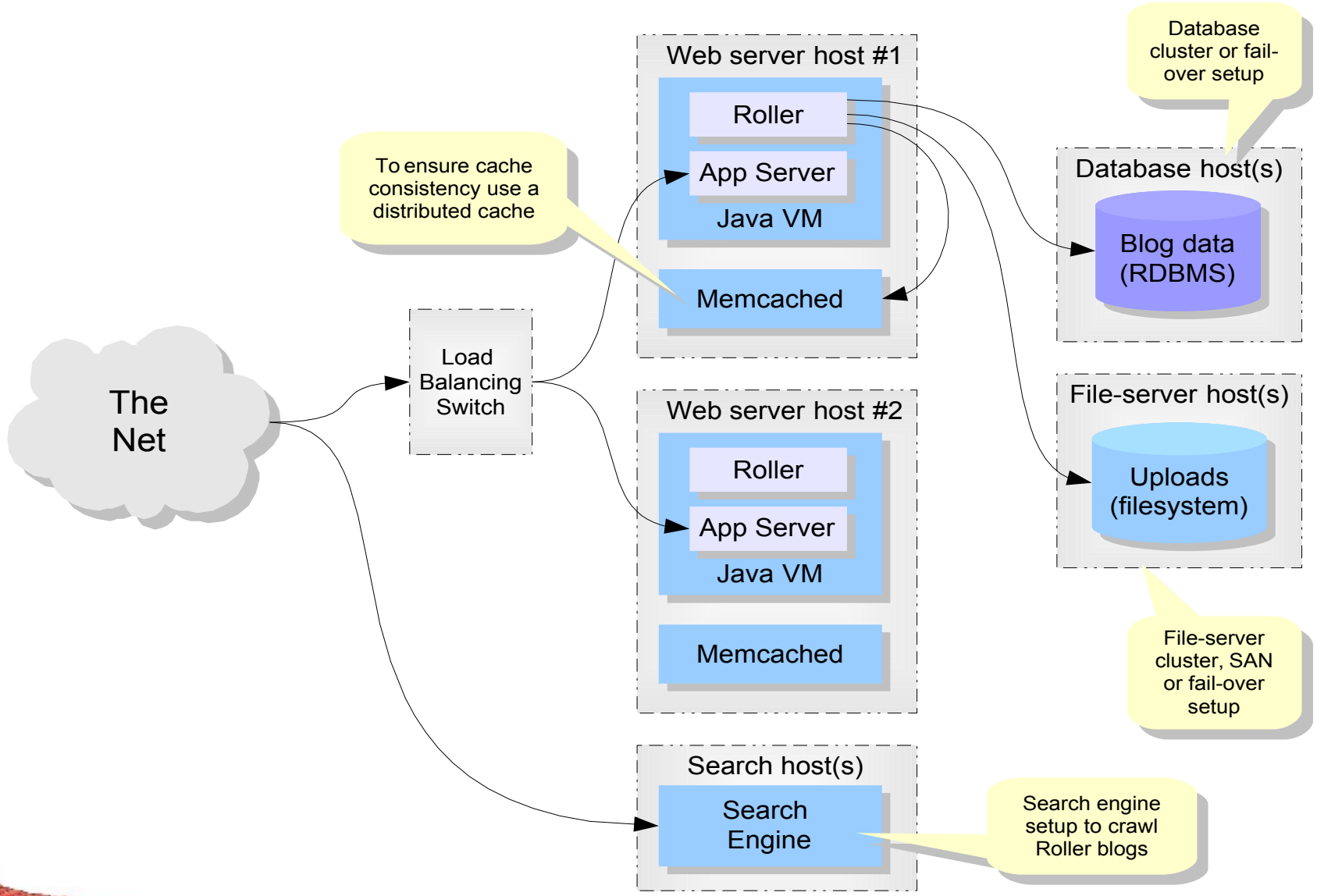
Photo by Tony Welch - http://blogs.sun.com/frosty/entry/new_blogs_server

Deployment architecture: small

- Here's a typical setup for a small site



Deployment architecture: large



Deployment architecture: sizing

- Don't have good sizing data, but...
- blogs.sun.com* handles 1.6 million hits/day with very low load:
  - Two SunFire T2000 servers
  - Each running Sun Web Server 7
    - Java VM heap size 3GB
  - Each running Memcached
    - Memory max set to 6GB
  - MySQL database cluster
  - Sun OneSearch Search Engine

** information based on public mailing list and blog posts*

TIP: check the Roller Support project

Support Project

Themes, plugins and add-ons for Apache Roller

The Roller Support Project provides themes, plugins and other add-ons for the Apache Roller blog server. If you've got a theme or plugin you'd like to contribute then speak up on the Apache Roller mailing lists.

The Java.Net Roller Support site hosts sample applications and related components based on the Apache Roller blog and planet servers. Disclaimer: This site is not affiliated with the Apache Software Foundation.

Themes for Roller
Additional Roller themes for use on your site.

Plugins for Roller
Additonal weblog entry plugins (e.g. JSPWiki plugin, Textile plugin, etc.)

Editor-addins for Roller
Alternatives to the Text and Rich Text editors included in Roller

Caching with Memcached

- If you're running Roller in a cluster then consider using Memcached
- What is it?

  *“memcached is a high-performance, **distributed memory object caching system**, generic in nature, but intended for use in speeding up dynamic web applications by alleviating database load.*
- Who uses it?
  - Wikipedia, LiveJournal, Slashdot, SourceForge, blogs.sun.com and more...

Getting Memcached

- Downloads and docs here
  - http://www.danga.com/memcached
- Installing can be as easy as this:
  - `apt-get install memcached` (on Debian or Ubuntu)
- Or this, on Solaris with Blastwave:
  - `pkg-get install memcached`
- There's a Windows version too
  - http://jehiah.cz/projects/memcached-win32

Setting up Memcached

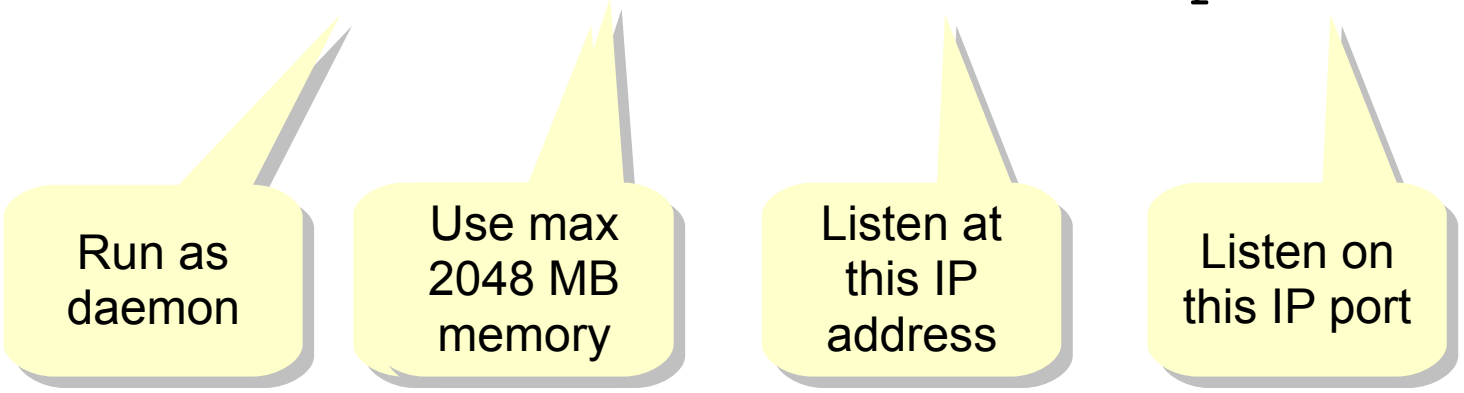
- Run one or more memcached daemons

- For example, on one machine:

```
$ ./memcached -d -m 2048 -l 10.0.0.40 -p 11211
```

- And on another:

```
$ ./memcached -d -m 2048 -l 10.0.0.41 -p 11211
```



Setting up Roller-Memcached Plugin

- Download from Roller Support project
  - https://roller.dev.java.net/servlets/ProjectDocumentList
  - Under Roller Support 4.0 / Plugins you'll find the file:
    - **roller-memcached-4.0.tar.gz**

- Unzip it and add jars to Roller

```
$ tar xzvf roller-memcached-4.0.tar.gz
$ cp roller-memcached/* roller/WEB-INF/lib
```

Config. the Roller-Memcached Plugin

- Via `roller-custom.properties`
- You can set Memcached as the default

```
cache.defaultFactory=\
   net.java.rolller.tools.cache.memcached.MemcachedLRUCacheFactory
cache.memcached.default.servers=\
    10.0.0.40:11211, 10.0.0.41:11211
```

- Or use different servers for different caches

```
cache.memcached.weblogfeed.servers=\
    10.0.0.40:11211, 10.0.0.41:11211
cache.memcached.weblogfeed.servers=\
    10.0.0.40:11211, 10.0.0.41:11211
```

When things go wrong? Logging!

TIP: enabling debug logging

- Roller uses Commons Logging, backed by Log4J
- Log4J properties are read from Roller properties file

- Enable DEBUG logging for a package by adding a line line this to your `roller-custom.properties` file:

```
log4j.category.<package-name>=DEBUG
```

TIP: how to enable debugging

- Key packages for debugging
  - `org.apache.roller.weblogger.business`
  - `org.apache.roller.weblogger.business.startup`
  - `org.apache.roller.weblogger.business.search`
  - `org.apache.roller.weblogger.config`
  - `org.apache.roller.weblogger.pojo`
  - `org.apache.roller.weblogger.planet`
  - `org.apache.roller.weblogger.ui.core`
  - `org.apache.roller.weblogger.ui.struts2`
  - `org.apache.roller.weblogger.ui.rendering`
  - `org.apache.roller.weblogger.webservices`
- See Javadocs for package listing
  - http://people.apache.org/~snoopdave/javadocs/roller40

Scripting and automation

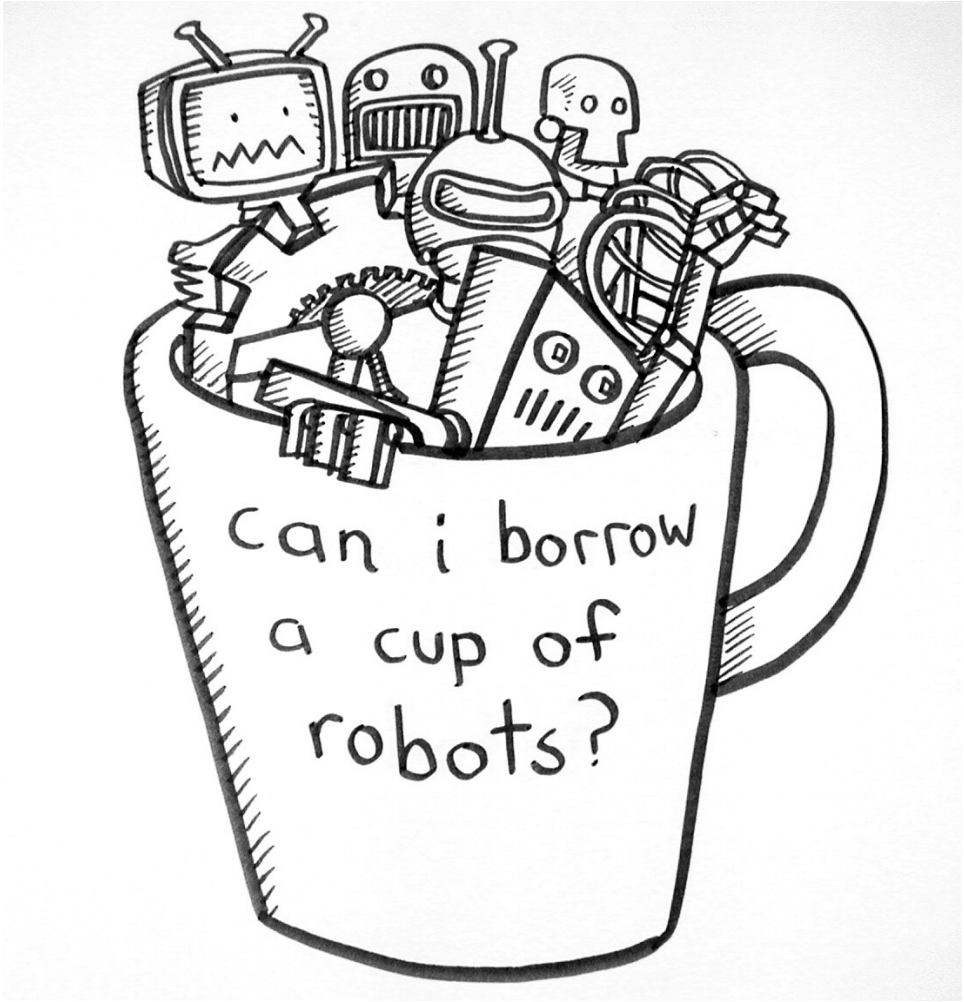


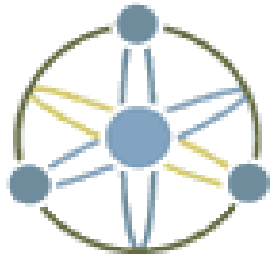
Photo by http://flickr.com/photos/striatic

Scripting and automation

- Sometimes you need to automate
- You might need to do provisioning
- Or automatically post weblog entries
- Or react when new content is posted

Automation options in Roller

- First, let's talk about posting
- Atom Publishing Protocol (RFC-5023)
  - In Roller this means
    - Full CRUD for blog entries and file uploads
    - ... and nothing else
  - Lots of options for client libraries
    - Apache Abdera (incubating)
      - http://incubator.apache.org/abdera
    - ROME Propono
      - http://wiki.java.net/bin/view/Javawsxml/RomePropono



Automation options, continued

- Now let's talk about administration
- Roller Admin Protocol (RAP)
  - REST API for user and blog provisioning
- The “Roller API” i.e. call Roller's Java classes
  - Put Roller jars in classpath and call Roller's Java classes directly
  - If you know what you're doing you can do just about anything this way

Roller Admin API (RAP)

- Developed for Sun's Portal product
  - http://rollerweblogger.org/wiki/Wiki.jsp?page=DeveloperGuide
- AtomPub-like REST API
- Enable via roller-custom.properties

```
webservices.adminprotocol.enabled=true
```

- End-point returns Service Doc

```
http://localhost:8080/roller/roller-services/rap
```

- Let's take a look

RAP service doc

```
<service xmlns="http://purl.org/roller/rap#">
<workspace title="Collections for administration">
    <collection title="Weblog administration entries"
      href="http://suntoy:8080/roller/rap/weblogs">
      <member-type>weblog</member-type>
    </collection>
    <collection title="User administration entries"
      href="http://suntoy:8080/roller/rap/users">
      <member-type>user</member-type>
    </collection>
    <collection title="Member administration entries"
      href="http://suntoy:8080/roller/rap/members">
      <member-type>member</member-type>
    </collection>
  </workspace>
</service>
```

Scripting RAP with Groovy

- Download the Groovy RAP examples
  - Not an official Apache release, get it here:
  - http://people.apache.org/~snoopdave/apache-roller-4.0/groovy
  - There you'll find the file: **roller-4.0-grooy-rap.tar.gz**
- The RAP SDK is included and examples:
  - listcollections.gy
  - listusers.gy
  - createuser.gy
  - deleteuser.gy

- Instructions are in the README.txt

Scripting the Roller API

- Download the Groovy Roller API examples
  - Not an official Apache release, get it here:
  - http://people.apache.org/~snoopdave/apache-roller-4.0/groovy
  - There you'll find the file:
    - **roller-4.0-grooy-rollerapi.tar.gz**
- Eight example scripts are provided
- Instructions are in the README.txt

Groovy Roller example

```
user = new User();
user.setUserName(username);
user.setScreenName(username);
user.setPassword(password);
user.setFullName(fullName);
user.setEmailAddress(email);
user.setLocale(locale);
user.setTimeZone(timeZone);
user.setDateCreated(new java.util.Date());

WebloggerStartup.prepare();
WebloggerFactory.bootstrap();
roller = WebloggerFactory.getWeblogger();
umgr = roller.getUserManager();

umgr.addUser(user);
roller.flush();
```

Photo by http://flickr.com/photos/shoebappa

TIP: create your own Roller build

- Get Subversion and the Java 5 SDK
- And do this:

```
$ svn co https://svn.apache.org/repos/asf/roller/trunk roller_trunk
$ cd roller_trunk/apps/weblogger
$ ant dist apache-release
```

- You'll find release files in `./dist`
  - apache-roller-4.0.tar.gz
- And the WAR directory in `./build/webapp`

Plugging in new functionality

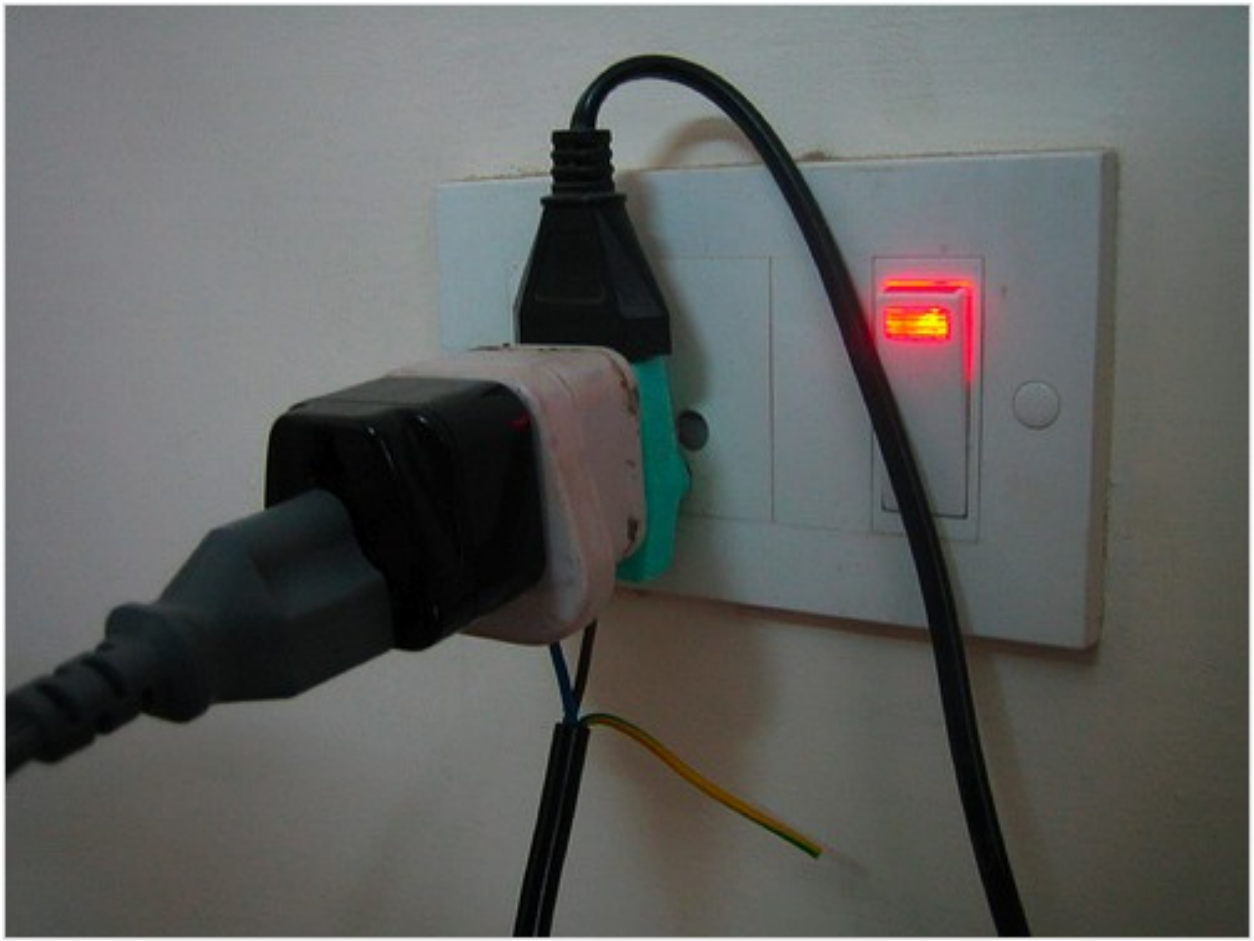


Photo by http://flickr.com/photos/verseguru

Ten types of Roller plugins

1) Page Model Plugin
2) Weblog Editor UI Plugin
3) Weblog Entry Plugin
4) Weblog Entry Comment Plugin
5) Comment Authenticator Plugin
6) Comment Validator Plugin
7) Renderer Plugin
8) Request Mapper Plugin
9) Cache System Plugin
10) Repeatable Task

Enough material for several talks...
Very little documentation available

See also http://rollerweblogger.org/roller/entry/roller_plugins

Implementing a Page Model Plugin

- Make data available in templates
- Implement the `Model` interface

```
public interface Model {
    public String getModelName();
    public void init(Map params);
}
```

The init() method is called on every request. Use these parameters to initialize your model.

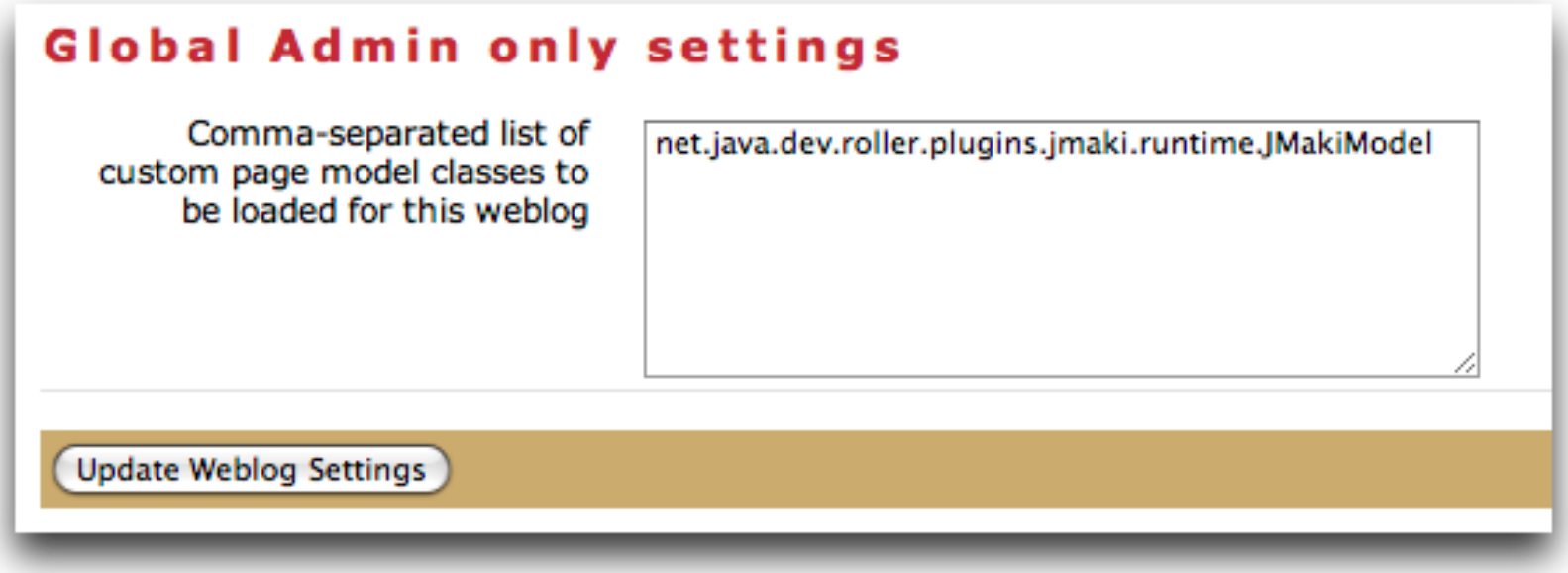
urlStrategy – URLStrategy for forming URLs to Roller resources.

pageContext – JSP PageContext object with Servlet Request and Response.

- And add getters for your data
- Compile against roller-web.jar
- Add your class(es) to Roller classpath
  - Via jar in WEB-INF/lib or classes directory

Configuring your Page Model

- Login as a Global Admin and add your page model to specific blogs via the Weblog Settings page:



Configuring your Page Model

- Or, to make model available to all blogs:
- Via `roller-custom.properties`
- Add Model classname to properties
  - To make it available in feed templates, add to

```
rendering.feedModels=
```

  - To make it available in all blog pages, add to

```
rendering.pageModels=
rendering.searchModels=
rendering.previewModels=
```

Configuring your Page Model

```
rendering.pageModels=\
org.apache.roller.weblogger.ui.rendering.model.PageModel,\
org.apache.roller.weblogger.ui.rendering.model.ConfigModel,\
org.apache.roller.weblogger.ui.rendering.model.UtilitiesModel,\
org.apache.roller.weblogger.ui.rendering.model.URLModel,\
org.apache.roller.weblogger.ui.rendering.model.MessageModel,\
org.apache.roller.weblogger.ui.rendering.model.CalendarModel,\
org.apache.roller.weblogger.ui.rendering.model.MenuModel,\
net.java.dev.plugins.jmaki.JMakiRuntime

rendering.searchModels=\
org.apache.roller.weblogger.ui.rendering.model.SearchResultsModel,\
org.apache.roller.weblogger.ui.rendering.model.ConfigModel,\
org.apache.roller.weblogger.ui.rendering.model.UtilitiesModel,\
org.apache.roller.weblogger.ui.rendering.model.URLModel,\
org.apache.roller.weblogger.ui.rendering.model.MessageModel,\
org.apache.roller.weblogger.ui.rendering.model.CalendarModel,\
org.apache.roller.weblogger.ui.rendering.model.MenuModel,\
net.java.dev.plugins.jmaki.JMakiRuntime

rendering.previewModels=\
org.apache.roller.weblogger.ui.rendering.model.PreviewPageModel,\
org.apache.roller.weblogger.ui.rendering.model.ConfigModel,\
org.apache.roller.weblogger.ui.rendering.model.UtilitiesModel,\
org.apache.roller.weblogger.ui.rendering.model.PreviewURLModel,\
org.apache.roller.weblogger.ui.rendering.model.MessageModel,\
org.apache.roller.weblogger.ui.rendering.model.CalendarModel,\
org.apache.roller.weblogger.ui.rendering.model.MenuModel,\
net.java.dev.plugins.jmaki.JMakiRuntime

rendering.siteModels=\
org.apache.roller.weblogger.ui.rendering.model.SiteModel,\
org.apache.roller.weblogger.ui.rendering.model.PlanetModel,\
net.java.dev.plugins.jmaki.JMakiRuntime
```

TIP: Create your own themes

- Another way to add functionality is to add your own new themes to Roller
- The Template Guide explains the macros and model objects you can use
- Two part blog post explains the details of packaging your theme and creating the theme definition file:
  - http://rollerweblogger.org/roller/entry/how_to_create_a_roller
  - http://rollerweblogger.org/roller/entry/roller_themes_part_2

Questions?

- For more information:
  - Apache Roller sites
    - http://roller.apache.org - official homepage
    - http://cwiki.apache.org/ROLLER - project wiki
    - http://rollerweblogger.org - project blog
  - The Roller Support project (not an ASF site)
    - http://roller.dev.java.net
  - Other Roller related (non ASF) sites
    - http://rollerweblogger.org/roller
    - http://people.apache.org/~snoopdave