UsageEvents

Starting point for aggregating the discussion about usage statistics ...

Mark Wood's Patches:

1927351: Usage event capture (statistics, etc.) JSPUI

1927360: Usage event capture (statistics, etc.) XMLUI

2025998: Usage event capture for 1_5_x JSPUI, XMLUI

DSpace GSoC Statistics Project:

StatisticsAddOn

Recent Discussions on Usage Statistics...

http://sourceforge.net/mailarchive/message.php?msg_id=20080501134422.GA18930%40IUPUI.Edu

http://sourceforge.net/mailarchive/message.php?msg_id=20080327144540.GD12272%40pomona.hpl.hp.com

```
Email Archive: dspace-devel (read-only) Admin Posts
[Dspace-devel] Versioning and statistics for 1.6 and beyond
Exclude From: James Rutherford <james.rutherford@hp...> - 2008-03-27 14:44
Hi all,

Now that the 1.5 release is out of the way (thanks Scott et al) it's
time to crack on with the next! You may recall that we had some pretty
successful output from the summer of code last year, and I think that
the versioning and statistics projects are good candidates for imminent
review & inclusion into trunk. The other work (including the content
integrity service) may be better placed as an add-on, and this is
something we're looking in to.

Does anyone have any immediate thoughts / concerns about putting this
work into trunk? I think it would be great to have this stuff included
before we start the summer of code again this year!

cheers,

Jim
```

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Tim Donohue <tdonohue@ui...> - 2008-03-27 15:04
Jim,

I'm actually curious as to whether the GSOC statistics project from last
year should be integrated with (or take advantage of) all the Statistics
work that University of Minho has done with the StatisticsAddOn:

http://wiki.dspace.org/index.php/StatisticsAddOn

I'm all for getting a better out-of-the-box statistics engine into 1.6.
But, I just want to make sure we're taking advantage of other work
that's been done, or at least do a review of the StatisticsAddOn
alongside the GSOC Statistics to see which is the more advantageous
route. In addition, we obviously need to get one or the other
functional with Manakin, but that can come a bit later :)

All that being said, statistics would be an area that I'd be interested
in helping out with in 1.6...we're planning a move to Manakin, and
desperately want something in place as it is.

- Tim

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Elliot Metsger <emetsger@jh...> - 2008-03-27 15:13
I liked Mark Wood's approach. Gathering the stats, and presenting the
stats are two different things - they should not be coupled. Stat
events get sent to a sink, and something handle the events coming to the
sink. Mark started a thread a while back on his approach...

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Dorothea Salo <dsalo@li...> - 2008-03-27 15:16
On Thu, Mar 27, 2008 at 10:04 AM, Tim Donohue <tdonohue@ui...> wrote:
> Jim,
>
> I'm actually curious as to whether the GSOC statistics project from last
> year should be integrated with (or take advantage of) all the Statistics
> work that University of Minho has done with the StatisticsAddOn:
>
> http://wiki.dspace.org/index.php/StatisticsAddOn

Might we want to look at IRStats as well? For functionality, at least;
I don't know that the code can be integrated.

+1 to a stats system; this is the commonest feature request I receive.
If it's showing up in 1.6, I may call a halt to our plans to integrate
IRStats with 1.5.

Dorothea

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Tim Donohue <tdonohue@ui...> - 2008-03-27 15:31
Dorothea,

IRStats is a good tool for general statistics. But, from what I've seen of it, it doesn't yet provide much detail on statistics *within* a specific Community or Collection in DSpace. The reason is that it relies on the *Apache* DSpace web logs, and doesn't use the DSpace generated logs or understand the general DSpace architecture.

That all being said, I think it's still worth looking at and potentially helping them build out. The only other issues I see are that it's all Perl (we've been trying to stay all Java), and requires dependencies (like AWStats and MaxMind database) which likely couldn't be packaged with DSpace out-of-the-box. But, it'd still be nice to offer as an "add-on", or write up a good how-to to use it. :)

- Tim

I agree. My ideal stats system would be UI agnostic, and would be able to generate statistics that could be displayed in JSPUI, XMLUI, and future UIs (within reason)... :) We'd be asking for trouble if we start down the route of building a separate stats package for each interface.

- Tim

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Mark Diggory <mdiggory@MI...> - 2008-03-27 15:43
The GSoC stuff is actually all ServletFilter based last time I looked... It should be something that can be added to both.

-Mark

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Elliot Metsger <emetsger@jh...> - 2008-03-27 15:46
Tim Donohue wrote:

Ideally, the stats engine is extensible as well. At least no static singletons anywhere that business logic is used.

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Elliot Metsger <emetsger@jh...> - 2008-03-27 15:54
Mark Diggory wrote:

I agree, the Stats API should be usable from a filter. I can see where it would be handy.

I'd also like a mechanism to expose what events have been recorded so that potentially one doesn't record the same event twice. E.g. if Manakin is modified to record stats (using the common stats API), and then someone pops a servlet filter in front of it (using the same API) there is a potential for the filter firing and manakin firing on the same request. Of course, someone who is mucking about with Manakin and servlet filters probably knows what they are doing and would be aware of this.

If you don't expose the number of times an event has fired to the Stat API caller, perhaps the stats engine itself will record when it sees the same event twice so you can filter them out when reporting.

Elliot

If the common Stats API is designed well, hopefully existing engines could write an adapter. I think that should be a use case when considering a common stats api.

Elliot

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Mark H. Wood <mwood@IU...> - 2008-03-27 19:06

Attachments: Message as HTML
See patches 1927351 and 1927360 for what I've been doing since.

I've been working on fitting a version of the statistical code from the University of Rochester to this generic event tap. Our users want it in XMLUI this time, so I've been doing that first. I've got an aspect that seems to handle the events and augment affected DRI documents appropriately, but I now need to extend a theme to lay out the results, and then work out how to package it all as an add-on.

This is all against DSpace 1.4.2 and Manakin 1.1a so far, but I want to bring it up to DSpace 1.5 very soon.

--
Mark H. Wood, Lead System Programmer mwood@IU...
Typically when a software vendor says that a product is "intuitive" he
means the exact opposite.

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Mark Diggory <mdiggory@MI...> - 2008-03-27 19:16
Folks,

I think that we should be providing resources to the community get
this sort of work moving in a collaborative direction...

There is much room in the dspace-sandbox at google-code for projects
such as this... you shouldn't need to be dumping patches into the
patch queue... we need to get away from that approach to
disseminating "addons".

I propose that MarkW and anyone else who wishes to contribute to at
unified statistics solution be given dspace-sandbox rights and we'll
organize a project space for them... This way we can work
collaboratively, centrally and iteratively.

Cheers,
MarkD

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Mark H. Wood <mwood@IU...> - 2008-03-27 20:59

Attachments: Message as HTML
On Thu, Mar 27, 2008 at 12:12:45PM -0700, Mark Diggory wrote:
> There is much room in the dspace-sandbox at google-code for projects
> such as this... you shouldn't need to be dumping patches into the
> patch queue... we need to get away from that approach to
> disseminating "addons".

Sorry, I didn't know. I'd acted on an earlier suggestion and put a
version of my patch* up on the Developer Sandbox page on the wiki, but
haven't seen any subsequent discussion of the code. So, since there
seemed to be some interest today, I tried a model that a lot of
open-source projects use -- you might call it "throw a patch and see
what they say" -- and learned that that's not the custom here.

I would appreciate some guidance. I haven't found a page describing
the feature development lifecycle for the DSpace project. To put it
simply, I don't know how to behave here.

----
* http://wiki.dspace.org/index.php/Modular+Usage+Statistics

--
Mark H. Wood, Lead System Programmer mwood@IU...
Typically when a software vendor says that a product is "intuitive" he
means the exact opposite.

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Mark Diggory <mdiggory@MI...> - 2008-03-27 22:21
On Mar 27, 2008, at 1:58 PM, Mark H. Wood wrote:

No, there really hasn't been a precedent other than throwing things
on the patch queue and in the wiki... And thats the problem... those
places are not very good for this sort of thing...

My goal is to get people thinking that working in someplace like the
sandbox is "better" than having to do all that work to organize stuff
in the wiki and create a patch in the queue with files attached...

Its different if your doing a local customization thats just for your
institution, thats not what I'm recommending this for. I'm
recommending it when a project has something that they want to throw
over the fence and put into the community.

Yep, no docs on this atm... maybe thats a problem that needs working
on... Its more self starting... we dole out svn projects in that
repository when someone from the community suggests they want to
contribute something (sword, dao, srw, AIP, i18n etc). Its mostly on
a case by case basis ATM. We'll continue a discussion and see if
there is an appropriate path to proceed on.

> -------------------------
> * http://wiki.dspace.org/index.php/Modular+Usage+Statistics

Organizing this for 1.5 will more than likely look different than
1.4 it can be a separate addon project. from the standpoint that all
your code changes could go there... But from the standpoint of the
hooks in the DSpace Servlets ... those will need to be placed into
the code in dspace-jspui-api or dspace-xmlui-api... Thats about it...

I have to say its really light and I rather like the approach. My
question is how does it differ from what was going on in the GSoC
project at these points. For instance...

1.) in http://dspace-gsoc.googlecode.com/svn/dspace/branches/2007/
statistics-1_4_x/dspace/src/org/dspace/app/webui/servlet/
BitstreamServlet.java

```
> // Statistics log
> LogEvent logEvent = new LogEvent();
> logEvent.setType(ContentEvent.BITSTREAM_VIEW);
> logEvent.setAttribute("id", ""+bitstream.getID());
> logEvent.setAttribute("ip", request.getRemoteAddr());
> if ((request.getHeader("referer")!=null)&&(!request.getHeader
> ("referer").equals("")))
> logEvent.setAttribute("referer", request.getHeader("referer"));
> if ((request.getLocale().getLanguage()!=null)&&(!
> request.getLocale().getLanguage().equals("")))
> logEvent.setAttribute("language", request.getLocale
> ().getLanguage());
> StatsLogger.logEvent(logEvent);
```

vs your

```
> + AbstractUsageEvent ue = (AbstractUsageEvent)
> PluginManager.getSinglePlugin(AbstractUsageEvent.class);
> + ue.setSessionID(request.getSession().getId());
> + ue.setSource(request.getRemoteAddr());
> + ue.setEperson(context.getCurrentUser());
> + ue.setEventType(AbstractUsageEvent.VIEW); // FIXME move to
> org.dspace.core.Constants?
> + ue.setObjectType(Constants.BITSTREAM);
> + ue.setID(bitstream.getID());
> + ue.fire();
```

these are pretty close...

a.) What I might recommend is because these clients are 99% going to
be in a Servlet Engine... that the Servlet Engine request just be
passed into the Event then events can customized without modification
of this points of delivery...
b.) I like the GSoC solution that encapsulates any interaction with
the PluginManager and just takes the Log event and passes into a
static class. Although I hate static methods and would rather see an

object constructed and fired directly like yours.

So I might like to see something more like a facade:

```
> + UsageEvent ue = new UsageEvent();
> + ue.setRequest(request);
> ue.setServletContext(getServletContext());
> ue.setContex(context);
> + ue.setEventType(AbstractUsageEvent.VIEW); // FIXME move to
> org.dspace.core.Constants?
> + ue.setObjectType(Constants.BITSTREAM);
> + ue.setID(bitstream.getID());
> + ue.fire();
```

Where its configuration and what ever service its interacting with
behind the scenes is encapsulated.

2.) Your implementation actually has a file based solution... we may
want to take that and use it in the GSoC FileLogger...
http://dspace-gsoc.googlecode.com/svn/dspace/branches/2007/
statistics-1_4_x/dspace/src/org/dspace/statistics/FileLogger.java

3.) The GSoC stats example writes to a JMS queue (another service to
setup and manage... not great IMO)... I want something lighter... a
JMS queue is a significant point of failure if it goes down.

Hope some of this critique is helpful,
MarkD

~~~~~~~~~~~~~
Mark R. Diggory - DSpace Developer and Systems Manager
MIT Libraries, Systems and Technology Services
Massachusetts Institute of Technology

Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Elliot Metsger <emetsger@jh...> - 2008-03-28 02:50
Mark Diggory wrote:
> On Mar 27, 2008, at 1:58 PM, Mark H. Wood wrote:
> So I might like to see something more like a facade:
>

```
>> + UsageEvent ue = new UsageEvent();
>> + ue.setRequest(request);
>> ue.setServletContext(getServletContext());
>> ue.setContex(context);
>> + ue.setEventType(AbstractUsageEvent.VIEW); // FIXME move to
>> org.dspace.core.Constants?
>> + ue.setObjectType(Constants.BITSTREAM);
>> + ue.setID(bitstream.getID());
>> + ue.fire();
```

>
>
> Where its configuration and what ever service its interacting with
> behind the scenes is encapsulated.

I agree, but I think there needs to be another layer of abstraction.

I don't think that the caller should be constructing Event objects. The
caller should only have to call one or at most two methods.

This has a few advantages:
1) it makes it easier for the caller - one line of code.
2) prevents each package from creating redundant Event object factory code.

Start with a method signature like:

```
event( StatEvent.VIEW, HttpServletRequest, Context, DSpaceObject )
```
~~~~~~~~~~~~~

```
This has a couple of problems:
1) it assumes the caller has access to the event() method arguments,
like HttpServletRequest. Depending on where the caller is in the stack,
they may not have access to an HttpServletRequest. They may have access
to a Cocoon Request. Or they may not have access to a *Request object
at all, but may have some of the parts (say a URL string).
2) We really shoudn't be passing objects that expose state to something
like a stats engine.

To this:
event( Event.VIEW, HttpServletRequestBean, ContextBean, DSpaceObjectBean )

Where *Bean is simply an analog for the mutable business object.

The problem with this is the caller has to convert the stateful business
objects to their *Bean analogs. Even if that capability is provided by
the Stats API that means another method call.

Perhaps there is a public, final object belonging to the stats api which
has setter methods that accept various business objects. An instance of
this public, final object is the argument to the event() method call.

You would have a wrapper for the Stats API depending where in the stack
you are. If you are in cocoon, you have a stats wrapper that accepts a
Cocoon Request, assembles the encapulated business objects into the
public final object from above, and forwards to the underlying stat api.
If you are in JSPUI (or otherwise have access to httpservletreqeust)
you would use that wrapper instead.

thoughts?



>
> 2.) Your implementation actually has a file based solution... we may
> want to take that and use it in the GSoC FileLogger...
> http://dspace-gsoc.googlecode.com/svn/dspace/branches/2007/
> statistics-1_4_x/dspace/src/org/dspace/statistics/FileLogger.java
>
> 3.) The GSoC stats example writes to a JMS queue (another service to
> setup and manage... not great IMO)... I want something lighter... a
> JMS queue is a significant point of failure if it goes down.
>
> Hope some of this critique is helpful,
> MarkD
>
> ~~~~~~~~~~~~~
> Mark R. Diggory - DSpace Developer and Systems Manager
> MIT Libraries, Systems and Technology Services
> Massachusetts Institute of Technology
>
>
>
>
>
>
> -------------------------------------------------------------------------
> Check out the new SourceForge.net Marketplace.
> It's the best place to buy or sell services for
> just about anything Open Source.
> http://ad.doubleclick.net/clk;164216239;13503038;w?http://sf.net/marketplace
> _______________________________________________
> Dspace-devel mailing list
> Dspace-devel@li...
> https://lists.sourceforge.net/lists/listinfo/dspace-devel
```

```
Re: [Dspace-devel] Versioning and statistics for 1.6 and beyond
Delete From: Mark H. Wood <mwood@IU...> - 2008-03-28 20:22

Attachments: Message as HTML
On Thu, Mar 27, 2008 at 03:17:48PM -0700, Mark Diggory wrote:
> On Mar 27, 2008, at 1:58 PM, Mark H. Wood wrote:
...
> a.) What I might recommend is because these clients are 99% going to
> be in a Servlet Engine... that the Servlet Engine request just be
> passed into the Event then events can customized without modification
> of this points of delivery...
>
> b.) I like the GSoC solution that encapsulates any interaction with
> the PluginManager and just takes the Log event and passes into a
> static class. Although I hate static methods and would rather see an
> object constructed and fired directly like yours.
>
> So I might like to see something more like a facade:
>
> > + UsageEvent ue = new UsageEvent();
> > + ue.setRequest(request);
> > ue.setServletContext(getServletContext());
> > ue.setContex(context);
> > + ue.setEventType(AbstractUsageEvent.VIEW); // FIXME move to
> > org.dspace.core.Constants?
> > + ue.setObjectType(Constants.BITSTREAM);
> > + ue.setID(bitstream.getID());
> > + ue.fire();

I tried to be a good little Java programmer and do things the OO way,
with all of the parallel setters and getters, but I didn't like it much.
If you look at the end of AbstractUsageEvent you'll see an overload of
fire() which takes all of the event data as arguments, and that
version does just take an HttpServletRequest and pluck out the stuff
it wants. I probably should have also reworked the event's properties
so that it isn't carrying around an Eperson, but just what we want to
know about the Eperson, and take that out of the concrete event
handlers.

> Where its configuration and what ever service its interacting with
> behind the scenes is encapsulated.

Okay, I guess you want to push the PluginManager calls inside the
UsageEvent constructor. I like it. Only the event layer should have
to know that the PluginManager is involved.

--
Mark H. Wood, Lead System Programmer mwood@IU...
Typically when a software vendor says that a product is "intuitive" he
means the exact opposite.
```