Item 60
git_comments:

1. For security reasons it's a bad idea to allow a leading '/', ex: /select?qt=/update see SOLR-3161 There was no restriction from Solr 1.4 thru 3.5 and it's now only supported for SearchHandlers.
2. Echo the request contents back to the client
3. **comment:** this one has handleSelect=true which a test here needs
   label: test
4. !! should *fail* above for security purposes
5. don't let superclass substitute qt for the path
6. * SOLR-3161 * Technically stream.body isn't remote streaming, but there wasn't a better place for this test method.
7. for anything
8. sneaky sneaky

git_commits:

1. **summary:** SOLR-3161 limit qt=/... (leading /) to refer to a SearchHandler for safety
   message: SOLR-3161 limit qt=/... (leading /) to refer to a SearchHandler for safety git-svn-id: https://svn.apache.org/repos/asf/lucene/dev/trunk@1305217 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Use of 'qt' should be restricted to searching and should not start with a '/'
   description: I haven't yet looked at the code involved for suggestions here; I'm speaking based on how I think things should work and not work, based on intuitiveness and security. In general I feel it is best practice to use '/' leading request handler names and not use "qt", but I don't hate it enough when used in limited (search-only) circumstances to propose its demise. But if someone proposes its deprecation that then I am +1 for that. Here is my proposal: Solr should error if the parameter "qt" is supplied with a leading '/'. (trunk only) Solr should only honor "qt" if the target request handler extends solr.SearchHandler. The new admin UI should only use 'qt' when it has to. For the query screen, it could present a little pop-up menu of handlers to choose from, including "/select?qt=mycustom" for handlers that aren't named with a leading '/'. This choice should be positioned at the top. And before I forget, me or someone should investigate if there are any similar security problems with the shards.qt parameter. Perhaps shards.qt can abide by the same rules outlined above. Does anyone foresee any problems with this proposal? On a related subject, I think the notion of a default request handler is bad - the default="true" thing. Honestly I'm not sure what it does, since I noticed Solr trunk redirects '/solr/' to the new admin UI at '/solr/#/'. Assuming it doesn't do anything useful anymore, I think it would be clearer to use <requestHandler name="/select" class="solr.SearchHandler"> instead of what's there now. The delta is to put the leading '/' on this request handler name, and remove the "default" attribute.

jira_issues_comments:

1. For some context on the security ramifications, see my comment on the linked issue: https://issues.apache.org/jira/browse/SOLR-1233?focusedCommentId=13169425&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-13169425
2. **body:** How about we just get rid of "qt" altogether and make everything be path-based? Internally, request handlers can be defined with a "rh_name" but it'll implicitly be the path. (and if it is prefixed with a / then we'll just leave it as it works now as a path, of course). In other words, we can be flexible about whether the name itself has a / in front or not, but same effect either way. Is there a need to continue to support qt at all? Why? And in the example configuration we can simply register a "select" request handler (that would be mapped to /select). I concur, no need to have a default setting... /select can be defined and used, otherwise only the request handlers defined can be used.
   label: code-design
3. +1 then we can get rid of the "handleSelect" logic in RequestDispatcher (this was added when we converted from Servlet to Filter and allowed path based handlers)
4. **body:** Well that's a total reversal of opinion Erik (being the one behind SOLR-1233), and I'm glad! +1 for getting rid of "qt" altogether. But that may be difficult: * The PingRequestHandler uses 'qt' as is clear in the solrconfig.xml. * firstSearcher & newSearcher configured in solrconfig.xml can't pick a request handler then. It would be nice if they could, they really should IMO. Perhaps the solution to the above is "qt" becomes an internal-only feature that doesn't work from the dispatching servlet filter -- the filter would throw an error if present. And heck, why don't we rename this thing to "requestHandler" while we're at it?
   label: code-design
5. **body:** bq. Well that's a total reversal of opinion Erik (being the one behind SOLR-1233), and I'm glad! Not quite a reversal - but rather a cleaning up. If we're going to support qt, then it should support _any_ request handler, IMO. But qt is 1) not an intuitive name at all, and 2) silly given that we should use paths instead for so many reasons. bq. why don't we rename this thing to "requestHandler" while we're at it? I'd be ok with just "rh" bq. PingRequestHandler and firstSearcher/newSearcher I see the argument for an internal only "rh" parameter, but still seems it shouldn't be necessary to even have that, such that we could

dispatch to a request handler internally directly without having to specify qt/rh/requestHandler as a query parameter. How we do that, I'm not sure yet though.
label: code-design

6. Eh... if we keep "qt" in any capacity, it should still be named "qt".
7. **body:** bq. I see the argument for an internal only "rh" parameter, but still seems it shouldn't be necessary to even have that, such that we could dispatch to a request handler internally directly without having to specify qt/rh/requestHandler as a query parameter. How we do that, I'm not sure yet though. It seems unavoidable that an internal request handler parameter name be used for the config file in these couple places. I don't think that's a big deal and I think it can be retained in a way that still brings some overall code clarity, security, and even lines-of-code reduction. And I guess Yonik's right about keeping the old name, even though it's a bad one.
label: code-design
8. **body:** bq. And I guess Yonik's right about keeping the old name, even though it's a bad one. I don't think its fair to call it a "bad" one -- it is just no longer the most appropriate name. "qt" came from before RequestHandlers existed. Old names is the tradeoff we have with strong back-compatibility
label: code-design
9. I'm +1 to the original improvements in this ticket, i.e. restricting qt for searching and disallowing qt's starting with "/". However, I can't see how forcing people into a path-only style of selecting RH configs is a benefit. The ability to stay at /select and switch between various named RHs through a simple request param is a strength in my experience.
10. **body:** bq. However, I can't see how forcing people into a path-only style of selecting RH configs is a benefit. The ability to stay at /select and switch between various named RHs through a simple request param is a strength in my experience. I've thought this way too, because it does make it convenient to just use /select for everything. However, for filtering and security of requests, path-based is much better. I'm of the opinion that if we have qt, it should not be restricted, but even better is to do away with qt and let request handlers define request paths. Ultimately I'd like to see Solr's request handling be much simpler and cleaner, less "magic".
label: code-design
11. **body:** Since when did we start focusing on security in Solr's APIs? The policy has been that Solr is by design insecure and must be locked down. I agree that not knowing that it is possible to do {{/select?qt=/update}} may lead someone to expose full access to Solr to external clients, and it would not hurt to plug that hole. But in my opinion {{qt=<name>}} is as much part of Solr's public APIs and toolset as is {{hl=true}}, and we should think before simplifying this. We're not talking about using {{/select}} for "everything", but have the ability to use for querying. I've had customers with external frameworks supporting Solr, but not supporting multiple different query URLs. Solr query URL is typically configured once in a config. However, the ability to plug in {{qt}} dynaically for different parts of the GUI saves the day.
label: code-design
12. -0 1) there are plenty of people who are happily using "qt" to dynamicly pick their request handler who don't care about securing their solr instances -- we shouldn't break things for them if we can avoid it. 2) assuming qt should be allowed only if it is an instance of solr.SearchHandler seems narrow minded to me -- it puts a totally arbitrary limitation on the ability for people to have their own request handlers that are treated as "first class citizens" and seems just as likely to lead to suprise and frustration as it is to appreciation for the "safety" of the feature (not to mention it procludes perfectly safe "query" type handlers like MLTHnadler and AnalysisRequestHandler if he root goal is "make solr safer for people who don't want/expect "qt" based requests then unless i'm overlooking something it seems like there is a far simpler and more straightforward solution... a) change the example solrocnfig to use handleSelect="false" b) remove the (long ago deprecated) SolrServlet if handleSelect == false, then the request dispatcher won't look at "/select" requests at all (unless someone has a handler named "/select") and it would do dispatching based on the "qt" param. currently if that's false the logic falls throough to the SolrServlet, but if that's been removed then the request will just fail. So new users who copy the example will have only path based request handlers by default, and will have to go out of their way to set handleSelect=true to get qt based dispatching. Bonus points: someone can write a DispatchingRequestHandler that can optionally be configured with some name (such as "/select") and does nothing put look for a "qt" param and forward to the handler with that name -- but it can have configuration options indicating which names are permitted (and any other names would be rejected) ...on the whole, compared to the original suggestion in this issue, that seems a lot safer for people who want safety, and a lot simpler to document. comments?
13. bq. Since when did we start focusing on security in Solr's APIs? The policy has been that Solr is by design insecure and must be locked down. +1 Security and Accessibility often fight, and we've generally chosen the side of making things easy when it does come to a conflict. As always though, it depends on the specific issue.
14. **body:** I thinks its nice that you can currently set up various handler with all of the different parameters, etc set up in your config and then clients don't have to worry about setting. (ie...what is the secret sauce for relevance, anyway, and which spelling dictionary goes with which "qf" list?, etc) Its just easier to have this all in the configuration. This is the beauty of "qt" so whatever solution we find here, I'd really like it if this beauty doesn't get spoiled. By the way, when we were converting an app from Endeca, we used "qt" to roughly emulate Endeca's "search interface" concept, which is basically like a dismax request handler that behaves as if it were a field. Imagine having multiple "qt"s (Request Handlers) set up, each with its own "qf", spelling config, highlighter config, etc, and then being able to do something like this: q=Handler1:(+this +that) AND Handler2:(something else) . Someday I would love to see this kind of enhancement (best I could tell you can't do anything like this even with local params). But if we lock down qt too much or eliminate it altogether, we might make it harder to have this kind of possibility in the future.
label: code-design
15. I'll just leave this with my final thoughts on it from an architecture/elegance perspective... paths represent resources, or perhaps rather "views" of resources. Solr serves up sets of content, basically. Treating these views into sets/subsets as first class "resources" HTTP-wise (HTTP being something we really tout in Solr-land as a big fat value-add over other networking protocols, caching/proxies/etc-wise, tried/true APIs everywhere) to me just seems like the Right Thing to do. Admittedly a RESTafarian, I realize it's a toematoe/tamahtoe-qt/path here, as path/request-param, 6 of 1 down low. As always Hoss has a way of proposing solutions to disagreements with some poignant clarity. So I'm +1 to this approach: {quote} if he root goal is "make solr safer for people who don't want/expect "qt" based requests then unless i'm overlooking something it seems like there is a far simpler and more straightforward solution... a) change the example solrocnfig to use handleSelect="false" b) remove the (long ago deprecated) SolrServlet if handleSelect == false, then the request dispatcher won't look at "/select" requests at all (unless someone has a handler named "/select") and it would do dispatching based on the "qt" param. currently if that's false the logic falls throough to the SolrServlet, but if that's been removed then the request will just fail. So new users who copy the example will have only path based

request handlers by default, and will have to go out of their way to set handleSelect=true to get qt based dispatching. {quote} I'll give this a trunk whirl myself shortly and see how it goes. I like it the compromise and the example enforcement of "best practices". bq. Bonus points: someone can write a DispatchingRequestHandler that can optionally be configured with some name (such as "/select") and does nothing put look for a "qt" param and forward to the handler with that name – but it can have configuration options indicating which names are permitted (and any other names would be rejected) LOL - true true. Better that it exist as an explicit Dispatching implementation, if you ask me, though. Ironicly, once upon a time... [LookupDispatchAction|http://struts.apache.org/1.x/struts-extras/apidocs/org/apache/struts/actions/LookupDispatchAction.html] Definitely +1 to including the Bonus Points into the equation too. Thanks Hoss for stating the nicely obvious way forward.

16. bq. Since when did we start focusing on security in Solr's APIs? !! (jaw dropping awe) Not soon enough! Apparently its when I started looking into it, some 5+ years after Solr was released. Jan, if I submitted an optional feature to Solr, perhaps a servlet filter that errors if parameters don't meet certain patterns, would you -1 it on principle? Yes, steps need to be taken to lock Solr down; I don't disagree with that. I think there is a reasonable intuitive expectation that /select will only query data and surprisingly this is false. I've made this incorrect assumption both in how I've deployed Solr, how I've explained how Solr be secured in my book, and how I've seen others explain how Solr can be secured. I don't think this is simply an issue of awareness (i.e. documentation). The bare minimum I will not "-1" (not that my vote counts, I'm not a PMC member), is for "qt" to not work with a leading slash, particularly in the default config, and right away in 3x. Consequently, the admin UI should not use it as such in order for the search screen to work. So there's enough interest here to keep "qt". Okay, fine with me. What I would like to see is for it to be brain-dead simple to explain how a request handler is chosen. The <requestDispatcher handleSelect="true"> setting and <requestHandler default="true"> setting both seem to complicate the explaination and so I think they should go away. All out of the box solrconfig.xml request handlers should have a name starting with '/'. On a request handler that supports "qt", an attribute should need to be added enableQt="true" added to the request handler. When qt is enabled, it dispatches to the named request handler, but it must not start with a leading '/'. Isn't this a simple explanation, and yet is still customizable with 'qt'? If agreed upon, I imagine this proposal would target 4.0.
17. bq. !! (jaw dropping awe) Not soon enough! Apparently its when I started looking into it, some 5+ years after Solr was released. Jan, if I submitted an optional feature to Solr, perhaps a servlet filter that errors if parameters don't meet certain patterns, would you -1 it on principle? No I wouldn't. Security is very important in some environments and in fact I'd like us to start supporting those use cases better, such as writing a generic document-level security component for various connector frameworks (such as MCF) to hook in to. But I'm very clear on that we should *not* start documenting ways to secure Solr/Tomcat/Jetty in a way that is suitable for public exposure - simply because that is a slippery slope and would give users the impression that Solr is secure and needs no further locking down. But for this issue I'm more interested in the functionality aspect. I must admit that the way I use {{qt}} in my projects is nothing more than a way to select a named instance of a *Search*RequestHandler with request-param defaults. So the fact that {{qt}} can completely switch to any RequestHandler is really too generic and seldom used. In that context your suggestion for a {{enableQt="true"}} param could make sense. If you enable it for all RH's, QT will work as today, or you can pick a few.
18. **body:** Hoss, In your view, is there any value in retaining the <requestDispatcher handleSelect="true"> setting and <requestHandler default="true"> setting for trunk? They seem legacy to me; I've never messed with them and I doubt anyone does but I don't know. Whatever their defaults may be or should be, I think their presence complicates keeping things simple, yet I'm unsure if their omission would result in any real loss of capability. I'm wondering what you think of my simplified proposal where I started saying "What I would like to see is for it to be brain-dead simple"...).
    label: code-design
19. I just attached a patch as an example along the lines of what Hoss proposed. I removed default="true", renamed "search" to "/select", and set handleSelect="false". Then I added some request handlers: * lazy - a lazy loaded search request handler * notlazy - a concrete (not lazy loaded) search request handler * /dispatch - a DispatchingRequestHandler that uses qt to look up a non-lazy loaded *SearchRequestHandler* and dispatches to that * /handlers - just a quick/easy way for me to see the defined request handlers (using the handlers.vm) template Here are some requests and their effect: http://localhost:8983/solr/handlers {code} - [/admin/plugins] - org.apache.solr.handler.admin.PluginInfoHandler@428d5aad - [/admin/system] - org.apache.solr.handler.admin.SystemInfoHandler@4e3c35fd - [notlazy] - org.apache.solr.handler.component.SearchHandler@52fc9d2b - [/admin/file] - org.apache.solr.handler.admin.ShowFileRequestHandler@46b29c9d - [/dispatch] - org.apache.solr.handler.component.DispatchingRequestHandler@78482bad - [/admin/luke] - org.apache.solr.handler.admin.LukeRequestHandler@4a2ba88c - [/update/javabin] - org.apache.solr.handler.BinaryUpdateRequestHandler@7846a55e - [/update] - org.apache.solr.handler.XmlUpdateRequestHandler@6612fc02 - [/terms] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@685f1ba8 - [/admin/threads] - org.apache.solr.handler.admin.ThreadDumpHandler@3c10e820 - [/replication] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@79f7abae - [/analysis/field] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@73286b10 - [lazy] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@628d2280 - [/browse] - org.apache.solr.handler.component.SearchHandler@50c7833c - [/admin/ping] - org.apache.solr.handler.PingRequestHandler@3e5646a5 - [/analysis/document] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@5da5e65f - [/select] - org.apache.solr.handler.component.SearchHandler@36b79701 - [/admin/mbeans] - org.apache.solr.handler.admin.SolrInfoMBeanHandler@4f1adeb7 - [/update/csv] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@6d13e8f3 - [/handlers] - org.apache.solr.handler.DumpRequestHandler@3622e177 - [/elevate] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@2c006765 - [/update/xslt] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@4e842e74 - [/update/json] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@4805e9f1 - [/get] - org.apache.solr.handler.RealTimeGetHandler@7c41f227 - [/update/extract] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@4d811e2c - [/admin/properties] - org.apache.solr.handler.admin.PropertiesRequestHandler@57e40274 - [tvrh] - org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@3a5d3ac0 - [/spell] -

org.apache.solr.core.RequestHandlers$LazyRequestHandlerWrapper@3ebc312f - [/debug/dump] - org.apache.solr.handler.DumpRequestHandler@354124d6 {code} http://localhost:8983/solr/select?q=*:* - returns our tried and true Solr response http://localhost:8983/solr/dispatch?q=*:*&qt=lazy - returns HTTP 400 with "Must be a SearchHandler: lazy" http://localhost:8983/solr/dispatch?q=*:*&qt=notlazy - returns HTTP 200 using the "notlazy" request handler's response And of course you can with this patch, but maybe that's silly to allow, do this request http://localhost:8983/solr/dispatch?q=*:*&qt=/select - which dispatches to /select, basically the same as http://localhost:8983/solr/select?q=*:* I personally really like the idea of qt not being special at all, yet if you do want to use something like that that you wire in a DispatchingRequestHandler. In fact, I'll go so far as to say that Solr's main dispatching filter shouldn't use any query string parameters, and only request handlers themselves use them. (that begs the question about wt, but there's HTTP mechanisms for specifying the format you desire a "resource" in :). I'll digress on that last point. The gist here is that with some simple config tweaks and a dispatcher, you can have your cake and eat it too.

20. p.s. What's "SolrServlet"? That must be way legacy out of trunk kinda thing? I think setting handleSelect="false" itself does the trick in preventing qt from dispatching without a DispatchingRequestHandler.
21. Erik, Cool. I hope you verified that /select?qt=/.... fails. What are your thoughts on my question to Hoss?
22. bq. I hope you verified that /select?qt=/ Did you mean that literally, as a single /? Anyway, with my patch (handleSelect=false and "/select" replacing the default "search" request handler), qt is nothing special. So /select?qt=/ is the same as /select since /select handler itself ignores qt. bq. What are your thoughts on my question to Hoss? IMO, if we're going the way of this DispatchingRequestHandler, then handleSelect should always be "false" and there should be no default request handler. Just map what you want to "/select" to make it the "default" since that's what the convention is.
23. **body:** I meant qt=/update for example, or anything with a leading slash really. I understand now that qt won't be interpreted in this case. When you say: bq. handleSelect should always be "false" and there should be no default request handler To clarify, are you saying not only the configuration but whatever code underlies these old concepts/features? That sounds great to me since it would ease code maintenance and would simplify explaining how request handlers are dispatched.
    label: code-design
24. **body:** bq. I understand now that qt won't be interpreted in this case. Right, in my patch, only DispatchingRequestHandler does anything with qt. If a request comes into /select with a qt, it might as well be /select?foo=bar and is just ignored. bq. "handleSelect should always be "false" and there should be no default request handler" - To clarify, are you saying not only the configuration but whatever code underlies these old concepts/features? I'm going to revise my patch where handleSelect defaults to false and thus won't even be needed in the example config, so I'm removing some comments about that in there. I realize we may want to kitchen-sink the example config and call this out, or not. I like the idea of a super lean "example"/"minimal" config with barely anything spelled out in there because our defaults just cover the best practices for the 80% or so of cases.
    label: code-design
25. **body:** And I'm not really sure what a "default" setting on a handleSelect="false" scenario means. If you want to use /select as the gateway, have a /select handler declared, without any other "magic" involved. Does default have any operation (I'm having trouble following SolrDispatchFilter's logic fully) when handleSelect="false"? I'm not seeing it yet.
    label: code-design
26. **body:** updated patch with handleSelect defaulting to false and removed from example solrconfig. Tidied /handlers view a little too.
    label: code-design
27. **body:** +1 Erik. Like the "/handlers" view (should it be /admin/handlers?) It is then possible to register the dispatchingRH as /select with a default qt set to the standard handler, and achieve backward compat for old applications. Do we want to provide backward compat based on LuceneMatchVersion for these changes? If someone with a 3.x config upgrades to 4.0 (or 3.6), their search should work as before even if they don't have an explicit "/select" handler in their config. The extra logic we need for this is appx: {code} if luceneVersion < LUCENE_36: $defaultHander = the handler with default="true"; else "search" register a "/select" handler being a Dispatching one with qt=$defaultHandler } {code} This way we emulate the old behavior without keeping the old code, just adding some backcompat code.
    label: code-design
28. I think we need to come up with something for 3x soon. I think and hope we can find broad agreement on some things. I think we can take a 2-3 incremental steps, starting with a better default configuration. I took a closer look at Erik's patch as well as some of the Solr code that makes this stuff work. Judging from the underlying code, if you already have a "/select" registered, then the <requestDispatcher handleSelect="..."> setting and the notion of a default request handler, and "qt" is effectively unused/ignored. "qt" is used internally still such as for specifying a handler in query warming. I really like this and it merely requires changes to the example solrconfig.xml which both simplify it and secure it. There could be a little comment on how to get "qt" to work, if you so choose. There is actually one other important thing to do which is to make the admin's query UI not use the 'qt'. I'll sign up to make that happen. Is anyone against (-1) me developing a simple patch doing the above and committing to 3x? I will of course attach the patch first for review and it would only include what I refer to in this JIRA comment. Subsequent to the above, I'm still concerned that someone's existing configuration is susceptible to /select?qt=/update. IMO if the path is /select, then qt should not start with a leading '/', thereby safeguarding people's existing default configuration. If you are -1 to my opinion, speak up.
29. Attached is a patch for 3x called "SOLR-3161-disable-qt-by-default". The tests pass. Aside from the changes I said I did, there were a couple other changes you will notice in the patch: * I made handleSelect default to false, as Erik did in his patch. I did this because if someone were to unwittingly rename /select to /whatever, they would be very surprised to discover that /select still works. And besides, it's basically legacy behavior in my view. I added a comment on how to get 'qt' to work in solrconfig. * In the admin UI query form, I renamed "Query Type" to "Request Handler" with a default choice of "/select" and I moved it to the top where it belongs. I wrote some JavaScript to make it switch the form's action to this value when it starts with a '/'. I tested in FF & Safari and I queried several times using the back-button in-between to ensure there were no lingering state issue with modifying the form. * I couldn't resist; in the admin UI query form, I renamed the label for 'q' from "Solr/Lucene Statement" to be "Query String" which is the same label seen on the front page. It should be noted that my patch proposal and Erik's before it are about improving the default configuration such that qt doesn't even work without taking steps to enable it, vs making 'qt' safer which is what the title of this issue is. I'll commit this ~ Tuesday 11am (GMT-5) unless someone objects.
30. **body:** It's not clear... do these proposals eliminate the possibility of using qt=/myhandler? I'm not sure we should be removing functionality that many have found useful. edit: a quick look at the patch suggests that (after enabling qt) that leading-slash named handlers will still be accessible via qt - but I couldn't discern the final intention from the various comments in this issue.
    label: code-design

31. That's right Yonik; this patch is just the first step though. In the next step, I am hoping to safeguard user's existing configuration from unintentional qt=/update. Two solutions come to mind: # Throw an error if qt starts with '/'. # Throw an error if qt points to a request handler that is NOT an instance of SearchHandler (subclasses are ok). Or some combination of the above, including adding a config flag making qt=/update possible. Gosh that's messed up though, IMO. What do you think?
32. **body:** bq. It's not clear... do these proposals eliminate the possibility of using qt=/myhandler? I'm not sure we should be removing functionality that many have found useful. My DispatchingRequestHandler currently (but only as a first draft) handles /-prefixed handlers to dispatch to them. But *only* if they are instanceof SearchHandler. Without DispatchingRequestHandler, and handleSelect=false, no qt dispatching is possible. bq. edit: a quick look at the patch suggests that (after enabling qt) that leading-slash named handlers will still be accessible via qt - but I couldn't discern the final intention from the various comments in this issue. Again, yes, currently, but only if it is a SearchHandler. There's a TODO in there to consider eliminating dispatching to /-prefixed handlers. I'm +1 on that. But it could be made even more flexible such that a list of allowed handlers is provided, or a regex pattern, or ... whatever. I just like that the crazy dispatching logic is moved out of SolrDispatchFilter and simplified, making it entirely up to the designer of the configuration to determine whether to wire in the DispatchingRequestHandler as /select or not. I'd generally prefer not, leaving everything as /-prefixed handlers that can only be dispatched to directly with no qt magic capability whatsoever.
    label: code-design
33. **body:** I don't think we should remove the ability to use qt with /-prefixed handlers, esp since the current patch here would disable "qt" by default.
    label: code-design
34. **body:** BTW I forgot to add a CHANGES.txt in my patch pending commit; it will be as follows: {quote} Upgrading from Solr 3.5 ---------------------- SOLR-3161: <requestDispatcher handleSelect="false"> is now the default. An existing config will probably work as-is because handleSelect was explicitly enabled in default configs. HandleSelect makes /select work as well as enables the 'qt' parameter. Instead, consider explicitly configuring /select as is done in the example solrconfig.xml, and register your other search handlers with a leading '/' which is a recommended practice. {quote} Erik, I very much agree with your sentiment that the dispatching logic get moved out of SolrDispatchFilter and get simplified and made configurable and clear in the solrconfig.xml. IMO, the current situation should stay in place in 3x (notwithstanding additions of safety checks to prevent qt=/update). 4x/trunk could then get your DispatchingRequestHandler (credit to Hoss for the idea) and _removal_ of SolrRequestDispatcher's handleSelect. By the way, I _suspect_ that assuming "/select" works is so ingrained into various parts of Solr, that we may want to log a warning if there isn't one registered at this path. Maybe I'm wrong but we'll see as we proceed.
    label: code-design
35. bq. Yonik: I don't think we should remove the ability to use qt with /-prefixed handlers, esp since the current patch here would disable "qt" by default. Ok. In my view, the worst part of qt=/update is that it's not actually a search. I'd love adding the SearchHandler instanceof restriction. However Hoss says he doesn't like it: bq. 2) assuming qt should be allowed only if it is an instance of solr.SearchHandler seems narrow minded to me – it puts a totally arbitrary limitation on the ability for people to have their own request handlers that are treated as "first class citizens" and seems just as likely to lead to suprise and frustration as it is to appreciation for the "safety" of the feature (not to mention it procludes perfectly safe "query" type handlers like MLTHnadler and AnalysisRequestHandler Hoss, my answer to you is to not use 'qt' for these cases; register the handlers with a leading '/', and for that matter, Erik and I _suggest_ that 'qt' not get used at all although we acknowledge it's still there for those that love it. Perhaps the restriction to appease most people's wishes should be to error IFF qt starts with a '/' AND it doesn't extend SearchHandler. ~ David
36. David - I see you committed a bit of this to 3.x. For the record, I'm -0 on this going to 3.x simply because it's possibly going to cause traumatic issues for some (or many?) environments. I worry that there will be projects that don't have handleSelect="true" in their config, yet they rely on /select?qt=.... We'll see. Were you going to make the same changes to 4.x?
37. I gave notice. As I understand it, handleSelect="true" has been explicitly in the default configs, and as such I think it's safe. If you'd like it to be safer, we could make the default value be dependent on the lucene version set in the config file like what Jan suggested: https://issues.apache.org/jira/browse/SOLR-3161?focusedCommentId=13221383&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-13221383 If this were done, perhaps the CHANGES.txt isn't even necessary? I am waiting committing it to 4x until SOLR-3232 so that the admin UI query will work. Kinda important :-) Regarding qt=/update what is your opinion on a compromise way forward? That's the next step.
38. bq. Regarding qt=/update what is your opinion on a compromise way forward? The most flexible way would be to add a method to SolrRequestHandler like "boolean indirectDispatch(req)" that would work for any type of request handler (and update handlers would default to false). But if one doesn't want this feature bleeding into the whole request handler architecture, we could distinguish update handlers from other handlers by adding a new base class or a marker interface.
39. bq. I gave notice. I know... I was just adding that I think it's possible we'll see some backlash to that change on 3.x but maybe not given that folks will most likely have handleSelect="true" in their configs. bq. Regarding qt=/update what is your opinion on a compromise way forward? I guess you mean on 3.x.... no opinion there. On 4.x, my opinion is qt is for the birds, get rid of it :)
40. **body:** bq. The most flexible way would be to add a method to SolrRequestHandler like "boolean indirectDispatch(req)" that would work for any type of request handler (and update handlers would default to false). Personally I think using the DispatchingRequestHandler iff you want qt behavior is the way forward here, cleaning up and streamlining the dispatching code. We already have a base class that can be used for determining if qt should dispatch in this case: SearchHandler. Adding what you propose adds more complexity than it's worth, IMO.
    label: code-design
41. As hoss points out, not all searching request handlers inherit from SearchHandler. bq. Adding what you propose adds more complexity than it's worth, IMO. The ability to distinguish an update handler from a request handler doesn't sound complex... the majority of the cases could be handled by adding "implements SolrUpdateHandler" to ContentStreamHandlerBase.
42. **body:** bq. As hoss points out, not all searching request handlers inherit from SearchHandler. Then use /-prefixed handlers for those rather than qt. Or, simply add whatever logic to the DispatchingRequestHandler that makes sense and let qt dispatching happen there, not from SDF. The DispatchingRequestHandler in my patch was merely an example; I really don't care what logic is in there to determine what can be dispatched to, as I'd never use it myself. bq. The ability to distinguish an update handler from a request handler doesn't sound complex Again, I'd say stuff whatever smarts desired down into a dispatching request handler rather than

making Solr's top-level dispatching logic more complicated than need be. But, I will say that having a better separated class hierarchy for search vs. update handlers is a good thing in general.
label: code-design

43. Note that "qt" is needed even for distrib search in some cases... I was just trying to debug the query elevation component by hitting /elevate, and that turned around and hit /select (which doesn't have QEC by default). So I needed to use shards.qt=/elevate to get it to work. We really need to think about if we really want to disable "qt" by default...
44. As long as you provide a leading '/' to shards.qt, there is no problem because the sharded request will use that as the path and not use 'qt'. The smarts that make that happen is largely due to the logic in QueryRequest.getPath(). I just played around with this in tests and stepped through the code to prove it out. This does remind me of another attack vector of sorts for what started all this. Even with qt disabled, this still leaves the possibility of /mysearch?q=...&shards=...&shards.qt=/update....
45. **body:** Choices, choices. I should point out that qt with a leading '/' was disallowed once upon a time [and the reason was for security|https://issues.apache.org/jira/browse/SOLR-1233?focusedCommentId=13215821&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-13215821]. It unfortunately came back [for no good reason (it simplified Erik's work on dataimport.jsp)|https://issues.apache.org/jira/browse/SOLR-1233], not because users didn't like this restriction. *In this light*, I think reverting to the old behavior is fine -- dataimport.jsp doesn't use qt with a leading '/' anymore, and hence the rationale for supporting a leading '/' is _gone_. To avoid breaking anyone who unwittingly depends on this in 3x, an exception can be made when the target is a SearchHandler. I repeat, nobody asked for "qt=/..." nor does anyone need it. The shards.qt parameter is not quite qt and it does need to support a leading '/' but Solr dispatches this as the path so the sharded request will never see qt=/update. But this does need protection somehow -- its actually riskier than 'qt' since it can reach out to a user-specified arbitrary server. I don't know why shards.qt exists since I don't see what could go wrong if a sharded request were to go to a path other than the original request, but I digress -- it's here. In this case, I think the SolrDispatchFilter can check if isShard=true and if so then mandate that the target handler extends SearchHandler -- which it should since only a SearchHandler has the shard logic.
label: code-design
46. **body:** Attached is a patch for 3x that does two things: * SolrDispatchFilter will now throw an error if 'qt' starts with a '/' and doesn't extend from SearchHandler. I added a test ensuring qt=/update errors. * SolrCore checks if isShard=true and then requires the target request handler to be a SearchHandler. I added a test for this. The check for this seemed a little out of place but I can't think of anything better. All tests pass. The changes.txt entry is: * SOLR-3161: Don't use the 'qt' parameter with a leading '/'. It probably won't work in 4.0 and it's now limited in 3.6 to SearchHandler subclasses that aren't lazy-loaded. I propose I commit this to both 3x and trunk in ~2 days to allow time for review. For 4.0, I think Erik and I (and Hoss?) are of like minds. Introduce the uber-flexible DispatchingRequestHandler to handle 'qt', AND remove handleSelect and related code that DispatchingRequestHandler renders obsolete. Erik and I would never use it but some of you clearly like 'qt'. Deprecated warnings would need to be added to 3.x to warn people about handleSelect=true going away. I suspect the hardest part of this may be updating old tests that want to use 'qt'. Thoughts?
label: code-design
47. I committed the last patch (limit qt=/... to SearchHandler) to 3x and trunk as I said I would. Should DispatchRequestHandler be committed to trunk, in the form that Erik coded it in his last patch or should it see certain improvements first? I won't be using it so I don't care that much.
48. **body:** I was looking at the logs of Solr's example config, unmodified and I noticed something troubling that I haven't noticed before: {noformat} WARNING: [] Null Request Handler 'null': {event=firstSearcher&q=static+firstSearcher+warming+in+solrconfig.xml} {noformat} It turns out that the default="true" attribute on the request handler actually does something :-) In RequestHandlers.initHandlersFromConfig() line 164 it checks this flag and registers the handler under the "" name. QuerySenderListener.newSearcher() line 59 does a lookup of "qt" from the its configuration (e.g. firstSearcher) getting null and then calling SolrCore.execute(null,...). Given that "/select" is referenced as a default in about a dozen places in Solr (.java source, excluding tests), it seems to me that if no request handler is marked as default then "/select" should become the default automatically (if present). Interestingly a similar check occurs as the very last line of RequestHandlers.initHandlersFromConfig() but for a request handler named "standard" -- which seems like a very old way things used to be (seems like a candidate for updating in Solr 4 to "/select"). So I propose that if "standard" isn't found, then it tries "/select".
label: code-design
49. **body:** I attached a simple patch for 3x, and it also marks DEFAULT_REQUEST_HANDLER as deprecated. If 4.0, I propose I simply eliminate DEFAULT_REQUEST_HANDLER since the constant is basically not used and "/select" is used seemingly everywhere as a string literal. I thought about a CHANGES.txt entry but it seems too minor.
label: code-design
50. I'll commit the patch ~midnight EST.
51. **body:** I committed it to 3x and 4x. In 4x I removed DEFAULT_REQUEST_HANDLER="standard" constant which wasn't used anywhere after I removed the only reference within the class. In 4x if a request handler is named "standard" it doesn't have an implied default anymore. I propose this issue get resolved and Erik & Hoss's DispatchingRequestHandler be a separate issue.
label: code-design
52. Reverted on 4x - tests fail.
53. As an aside, this default request handler patch should probably have been another JIRA issue instead of this JIRA issue including several different things including clarifying the default config. I stupidly committed the patch without testing and it broke the build because solr/core/src/test-files/solr/conf/solrconfig.xml (and possibly other test configs) has a <requestHandler name="standard"> (with no default="true"). Either the config(s) should be updated to be "/select" based (which I don't mind doing) or I could add back "standard" as eligible for default request handler nomination. I committed the latter now because it's the safest. /select should takes precedence. I also committed a warning message if there is none registered: {code} if(get("") == null) log.warn("no default request handler is registered (either '/select' or 'standard')"); {code}
54. I finally committed the remaining piece that was pending to trunk that had already made it to 3.6. This was for the example solrconfig.xml to not use the handleSelect logic, and to make handleSelect false by default. I could do this now because the new UI's query form is finally dispatching queries properly instead of always using 'qt' (SOLR-3317). Oh, and I freshened up the wiki page on how the dispatch works: http://wiki.apache.org/solr/SolrRequestHandler#preview Trunk commit: r1328798 Closing issue and marked for 3.6 as well.

55. I've reverted this commit, David -- it breaks the tests. Don't know how to fix, so revert seems sensible to avoid spamming the mailing list with jenkins failures.
56. It's caused enough problems for our tests - and it will cause a number of people issues while trying to upgrade. I think we should leave the ability to use "qt" enabled by default. A number of people have expressed the that they find it useful.
57. **body:** I swear I ran the tests before committing (which passed) -- I'll try and be more careful somehow. I used the IntelliJ test runner for all of Solr and I see it fails now. Strange. So the tests that failed did so because of one test config file solrconfig-tlog.xml doesn't have a handleSelect=true on it nor "/select" registered, just the old "StandardRequestHandler" registered as the name "standard". This file is so short at ~26 lines of XML from <config> to </config>, it clearly was never copied from a Solr example solrconfig.xml. Interestingly there are some other configuration files done similarly but they have yet to pose a problem. I looked into why and the reason is that those tests use embedded Solr and thus the default handler is looked up as such via core.getRequestHandler(null) no matter what its name actually is. No test has failed because it wanted to use "qt" but couldn't. Instead, tests have failed regarding "/select" not working. The handleSelect option intertwines the two which is unfortunate and unnecessary. *Here's a proposal:* Make handleSelect do just what it's name implies -- handles "/select" when "/select" isn't registered. And this is, in effect, to register the default handler under an alias of "/select". The 'qt' feature should be a distinctly separate option, and IMO disabled by default.
    label: code-design
58. bq. I swear I ran the tests before committing (which passed) I'm not saying you didn't, they are randomized after all (don't know if this particular case depends on the seed but it may). I just felt like you might be asleep at the time and jenkins was going bonkers with trying to repeat that build.
59. bq.I used the IntelliJ test runner for all of Solr and I see it fails now. Strange. I always run the Ant build to make sure I haven't broken the build. The IntelliJ build differs in some respects, so errors in one system may not manifest in the other.
60. bq. It's caused enough problems for our tests - and it will cause a number of people issues while trying to upgrade. I think we should leave the ability to use "qt" enabled by default. A number of people have expressed the that they find it useful. Agreed -- I see no advantage in changing the _default_ behavior when "handleSelect" isn't specified at all in the solrconfig.xml (specificly: [this change to SolrRequestParser|http://svn.apache.org/viewvc/lucene/dev/trunk/solr/core/src/java/org/apache/solr/servlet/SolrRequestParsers.java?r1=1328798&r2=1328797&pathrev=1328798] that was 1/2 of what Dawid reverted) because it causes pain on upgrade (using existing configs) w/o any obvious value that i see. I do however think the spirit of what David is proposing is a good idea: the _example_ configs that we ship should have handleSelect="false" with a comment explaining what that means, and encouraging people to use path based request handlers instead of setting it to true. That said: i don't think the comments added in [the other 1/2 of that commit|http://svn.apache.org/viewvc/lucene/dev/trunk/solr/example/solr/conf/solrconfig.xml?r1=1328798&r2=1328797&pathrev=1328798] are really the best way to go, because they are only meaningful to someone who upgrades -- not to a new user looking at the config. Those comments provides no education about handleSelect (and the tradeoffs of using it) to anyone (old user or new). I would suggest as an alternative something like... {noformat} <!-- Request Dispatcher This section contains instructions for how the SolrDispatchFilter should behave when processing requests for this SolrCore. handleSelect is a legacy option that affects the behavior of requests such as /select?qt=XXX handleSelect="true" will cause the SolrDispatchFilter to process the request and dispatch the query to a handler specified by the "qt" param handleSelect="false" will cause the SolrDispatchFilter to ignore "/select" requests, resulting in a 404 unless a handler is explicitly registered with the name "/select" handleSelect="true" is not recommended for new users, but is the default for backwards compatibility --> <requestDispatcher handleSelect="false" > {noformat} ...and later on... {noformat} <!-- Request Handlers http://wiki.apache.org/solr/SolrRequestHandler Incoming queries will be dispatched to a specific handler by name based on the path specified in the request. If handleSelect="true" has been specified in the requestDispatcher, then handlers using names without a leading '/' can be accessed with: http://host/app/[core/]select?qt=name If handleSelect="true" and a /select request is processed with out a qt param specified, then the requestHandler that declares default="true" will be used. If a Request Handler is declared with startup="lazy", then it will not be initialized until the first request that uses it. --> {noformat}
61. v4.0 is the biggest release ever and I thought it would be an opportune time to reconsider defaults of the past. If this isn't the time, then I don't think there will ever be a time to do it. And so I'll settle with Hoss's suggestion -- handleSelect is true by default in the code (as it has been) but not the config file (as was changed in v3.6), and the new suggested comments are fine. Should this proposed change be to the v3.6 branch too?
62. Attached is a patch to solrconfig.xml with comments based off of what Hoss suggested. I'll commit these to 4x & trunk in ~24 hours if no further comment is given.
63. Committed solrconfig.xml changes to 4x & trunk. Closing issue.