VelocityTools2 Planning

Ideas For VelocityTools 2.x

1. TransparentOnDemandToolsLoading: Instead of a standard HashMap, the idea here is to have a Toolbox, which will instantiate a tool from its tool-info only when the generic getter is called. This should be a quite interesting performance gain in some situations. If it seems useful, we could add a special attribute to force a tool to be instantiated at toolbox initialization ('instantiate-on-load' ?). The Toolboxes may then pool or at least hold on to instantiated tools for subsequent requests from the template or from other parts of the webapp (see idea 2 below). (STATUS: done except for the 'instantiate-on-load' attribute)

2. EasierToolAccessOutsideTemplates: Other objects in my webapp are often jealous of the VelocityViewServlet. They also would like to use some of the tools. Session scoped tools can be reached via the session, but it's not the case for application or request scoped tools. To achieve this, there would be a few things to do:

- create a separate Toolbox for each scope and store it in the attributes of the corresponding servlet API object (request->ServletRequest, session->HttpSession, application->ServletContext). (STATUS: done)
- the ViewToolContext (successor of ChainedContext) will search the attributes for these Toolboxes and pass requests for the tools on to them. (STATUS: done)
- We could create a simple utility to help other classes retrieve tools, so they needn't all duplicate the code for finding the toolbox in the attributes and then requesting the tool from the toolbox... (STATUS: done)

3. SimplifiedToolboxXML: In line with the ideas above, we could cut out some repetition in toolbox.xml by better organizing it and using XML more appropriately. For instance, the toolbox.xml for the "simple" example could be simplified further to something like:

{{{<tools>
<data type="number" key="version" value="1.1"/>
<data type="boolean" key="isSimple" value="true"/>
<data key="foo" value="this is foo."/>
<data key="bar">this is bar.</data>
<toolbox scope="request" xhtml="true">
<tool key="toytool" class="ToyTool" restrictTo="index.vm"/>
</toolbox>
<toolbox scope="session">
<property name="create-session" value="true" type="boolean"/>
<tool key="map" class="java.util.HashMap"/>
</toolbox>
<toolbox scope="application">
<tool class="org.apache.velocity.tools.generic.DateTool"/>
</toolbox>
</tools>
}}}
(STATUS: done, and now our tools come with a @DefaultKey annotation to make it even simpler)

4. FactorOutBasicVelocityViewStuff: This would create a better basis for bring the Veltag into the project as a sibling of the VelocityViewServlet. (STATUS: done)

5. SupportAlternateToolboxConfiguration: Not everyone likes XML. I'd like us to make Toolbox management pluggable with provided support for configuration via properties file as well as XML, and i want it to be easier to configure and manage in Java as well. (STATUS: done, but needs more testing)

6. Simplify the package structure of VelocityTools. There are a lot of sub-packages that seem redundant or unnecessary. I'd like to shift things about as much as possible to reduce that. (STATUS: done)

Backwards Compatibility For VelocityTools 2.x

1. Basic user interfaces - these are things that people use directly without extending
    - Our tools - the biggest thing here is the tools themselves. we shouldn't make users have to change their templates at all. we might consider finding some way to provide deprecation notices for things we'd like to change, but i don't think there's many places we'll need to do that. in the meantime, the most we may do is move packages and put deprecated subclasses with init() support in the old tool locations. (STATUS: DONE)
    - Custom tools - the trick will be to support the init() method. Not sure how this will work, but i'm pretty determined to find a way. (STATUS: done)
    - Servlets - A lot of people directly use the VVS and VLS. So, while i've moved them to a new package, there are deprecated subclasses at their old location to make this a seamless transition for these folks. (STATUS: done)
    - toolbox.xml - We need to create a FactoryConfiguration that can translate the old xml configuration format to fit the new tool management structure. This should not be too difficult. The trickier part is where we should automatically check for the old format (probably in the deprecated VVS and deprecated VLS, at least). Some thought may need to be put into this. Anyway, the goal is to not force people to update their configurations right away. (STATUS: done, though not quite as described above)
    - Logging stuff - The old CommonsLog-LogSystem bridges are deprecated and now extend their successors, the CommonsLog-LogChute implementations. I now think that CommonsLogLogChute belongs in Engine, not Tools, but it will have to have a copy here until at least the Engine 1.6 release. Also, ServletLogger is deprecated and extends its successor, ServletLogChute. For all of these, the transition should be seamless for users also using Engine 1.5. (STATUS: done)
    - WebappLoader - This has been deprecated and now extends its successor, WebappResourceLoader, which is better named and has logging improvements to match those of the other loaders in Engine 1.5. Again, upgrading to 2.x should not break anything for users also using Engine 1.5. (STATUS: done)

2. Common extension points - places where people extend the tools, servlets

- Servlet subclasses - we need to support this as much as possible. (STATUS: partly done)
- Tool subclasses that change init() or configure() - if we meet all the goals in #1 above, then this should happen automatically for the most part. we just need to leave our init() and configure() methods in place, so that calls to super.init() or super.configure() do not fail. (STATUS: done)

3. Advanced API users - those that use tool management without using the servlets (e.g. Spring MVC or "standalone" toolbox users)

- ViewContext users - can use deprecated one at old location for now (STATUS: done) though anyone who implemented their own version will find that the tools no longer recognize it. they need to update to the new ViewContext package name.
- ChainedContext users - ChainedContext extends ViewToolContext, everything should work superficially as it did before. (STATUS: done)
- XMLToolboxManager users - may be possible to hack up a version that reads old toolbox.xml format, and returns a Map of initialized tools for getToolbox(initData), but that initData part is tricky. partial support is probably the best we can do here, unless we leave all the old code intact and just deprecate it. (STATUS: deprecated, but left otherwise as-is. also, old tools still have deprecated but functional init() methods so they work with it)
- ServletToolboxManager users - similar situation to that of XMLToolboxManager, but worse. again, probably a choice between partial integration with new infrastructure or else leaving both infrastructures side-by-side with one deprecated. (STATUS: same as XMLToolboxManager)

4. Anyone else that digs further into the tool management API for 1.x - Unless we decide to only deprecate the old tool management infrastructure, these people shouldn't upgrade until they're ready to make a lot of changes. 🙂