ReleasePlanForCore1.1

Here is a list of things that it would be nice to get done before the next (1.1) release of Orchestra Core.

Facelets support for all orchestra tags

It would be nice for orchestra tags to be usable with Facelets.

SpringViewControllerScope

This class currently has jsf-specific code in it, which it should not.

```
****DONE***
```

examples

Add new very simple example without persistence

```
****DONE***
```

Fix broken examples

At least one example just does not run. Others have very serious layout problems.

Improve Example Documentation

Examples are for beginners to learn from. But the current examples have no information on what they are trying to demonstrate. Each example really needs to tell the user:

- what orchestra features are used
- what those orchestra features achieve
- how they can tell from the example app that those features are working

Modify "access" scope duration

```
****DONE***
```

Currently, access scope (formerly called flash scope) lasts until a complete postback/render cycle occurs without that bean being accessed.

But that means that in a master/detail setup, a postback to the detail page that forwards to the master followed by a postback to the master that navigates to the detail keeps the detail bean alive. Ecch.

Instead, access scope should expire if a *render* cycle has no access to it. This then means that a postback to the detail which forwards to the master will expire the detail backing bean because the master page does not refer to it. It also means that the bean is cleaned up earlier which is nice.

Note that when <redirect/> is used in a nav rule then this point is moot as a forward never occurs.

This issue was first reported by Matthias in his "facesgoodies" project.

Review reverts to Spring ScopeManager stuff

A while ago I (simon) made some cleanups in the scope-manager code, particularly to do with scoped-proxy beans where code "drills down" from the proxy to the real bean. This broke some code in the core15 module to do with the ViewControllerScope, so Mario reverted it. However I think the cleanup was worthwhile, and want to look at whether the original changes can be put back, and the issue fixed in the ViewControllerScope instead.

Drilling down from proxy to real bean

An orchestra user wanted to retrieve the "real" bean that an orchestra-generated proxy is proxying, in order to serialize it and pass it to an external system (the proxy is not serializable). He eventually found a partial solution, but it would be nice to look into whether a proper solution is possible.

```
****DONE***
```

View Scope

It was suggested on the mail-list that a "view" scope might be useful, ie a bean that lasts as long as the viewid does not change. This is rather like the existing "access" scope, but a bit more limited. It sounds quite useful and could possibly be more efficient than "access" scope.

Optional; could go into a later release.

Allow multiple-conversation-context stuff to be disabled

Orchestra appends a conversation-context query param to each generated URL in order to support a separate conversation-context per window. This is a nice feature, but clutters the URL and not everyone will want it. So add a config parameter to disable this behaviour.

It would be nice if orchestra could avoid generating this for forms (postbacks) at least. For postbacks the conversationContext value can just be stored in the view-tree and it will be restored automatically. For GET queries, ie h:outputLink and raw urls, the conversationContext does still need to be a query-param on the url. The problem is just knowing when we are generating a URL due to an h:form, and when the url is being generated in other situations.

When using <redirect/> navigation rules, the returning url will need that param too, though - unless something like the RedirectTracker approach is used, where the next request after a redirect is assumed to be the corresponding fetch (within some sane timelimit like 5 seconds). This should be ok, and just means catching ExternalContext.redirect or setting up a custom NavigationHandler.

Portals

A user has been trying to use Orchestra in a Portal application with no success. Getting Orchestra working for portals would be nice.

Clustering and Serialization

Currently there are a number of Orchestra classes that just disable serialization completely. These need to be reviewed to see whether we can properly support this, so that clustering and hot-restart work as well as environments where user sessions can get passivated under load.

Classes with serialization problems:

- ConversationManager
- ScopedBeanTargetSource
- RequestParameterProviderManager

Building with JSF 1.1

```
****DONE***
```

Currently, Orchestra binaries support JSF1.1, but must be built against JSF1.2. This is ugly.

The reason is that Orchestra currently uses a custom Application object in order to do some startup configuration, reducing the number of filters/listeners /etc that need to be defined in a web.xml file. However due to some odd design decisions in the JSF1.2 spec, it appears to be impossible to write an Application subclass that compiles against JSF1.1 and works correctly in a JSF1.2 environment.

However building against 1.2 has significant disadvantages. First, it is just weird. Second, it is possible for jsf12-specific code to accidentally creep in; thorough testing of the binary against 1.1 is needed to prevent this but orchestra has no such checks. Third, it means that lots of deprecation warnings occur in the code because we must call jsf1.1 features while compiling against jsf1.2.

Perhaps initialisation can be done in some other way? Or maybe a workaround can be found to allow an Application build against 1.1 to run correctly with 1.2.

Docs

Improve docs on orchestra site to show more clearly how multiple beans can live in the same conversation

Reports

bugs/issues in reports

```
****DONE***
```

The maven reports point out a lot of things that need fixing. See FindBugs, Checkstyle, JDepend, etc.

changelog report

```
****NOT POSSIBLE: ***
```

Also need to use revision numbers for svn-related reports rather than dates, as date-searching in the Apache svn repo does not work.

Unfortunately, the changelog plugin just does not support svn range numbers 😞. I've filed an enhancement request for the plugin, but don't expect a rapid fix.

Improve ViewController

There are a couple of features in the Shale ViewController that are missing in the Orchestra one.

In particular, Shale supports controllers for f:subView pages, which is very nice.

Minor issues

- convert tabs to spaces

  ```
  ****DONE***
  ```

- check svn:eol-style settings, esp. in site directories
- create a package.html file for each package