What's the best way to make a modular java web application [closed]

Asked 16 years, 3 months ago Modified 7 years, 3 months ago

Viewed 2k times

6

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed 7 years ago.

Improve this question

I'm building small web site in Java (Spring MVC with JSP views) and am trying to find best solution for making and including few reusable modules (like "latest news" "upcoming events"...).

So the question is: Portlets, tiles or some other technology?

java spring-mvc module

Share

Improve this question

Follow

edited Sep 6, 2017 at 20:30

Vadim Kotov

8,274 ● 8 ● 50 ● 63

asked Sep 16, 2008 at 12:25

moriarty

63 ● 4

7 Answers

Sorted by: Highest score (default)

4

If you are using Spring MVC, then I would recommend using Portlets. In Spring, portlets are just lightweight controllers since they are only responsible for a fragment of the whole page, and are very easy to write. If you are using Spring 2.5, then you can enjoy all the benefits of the new annotation support, and they fit nicely in the whole Spring application with dependency injection and the other benefits of using Spring.

A portlet controller is pretty much the same as a servlet controller, here is a simple example:

```
@RequestMapping("VIEW")
@Controller
public class NewsPortlet {

    private NewsService newsService;

    @Autowired
    public NewsPortlet(NewsService newsService) {
        this.newsService = newsService;
    }
```

```
    @RequestMapping(method = RequestMethod.GET)
    public String view(Model model) {
        model.addAttribute(newsService.getLatests(10))
        return "news";
    }
}
```

Here, a NewsService will be automatically injected into the controller. The view method adds a List object to the model, which will be available as ${newsList} in the JSP. Spring will look for a view named news.jsp based on the return value of the method. The RequestMapping tells Spring that this contoller is for the VIEW mode of the portlet.

The XML configuration only needs to specify where the view and controllers are located:

```
<!-- look for controllers and services here -->
<context:component-scan base-package="com.example.news

<!-- look for views here -->
<bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalRe
    <property name="prefix" value="/WEB-INF/jsp/news/"
    <property name="suffix" value=".jsp"/>
</bean>
```

If you want to simply embed the portlets in your existing application, the you can bundle a portlet container, such as eXo, Sun, or Apache. If you want to build your application as a set of portlets, the you might want to consider a full blown portlal solution, such as Liferay Portal.

Share Improve this answer Follow

edited Sep 17, 2008 at 8:55

answered Sep 16, 2008 at 13:24

pjesi
4,051 ● 3 ● 23 ● 16

thnx for useful answer, do you have any experience with Spring portlets MVC? Some tutorials, guidances? Maybe can recommend a portal server? – moriarty Sep 16, 2008 at 14:12

Developing them is my primary job so I have plenty of experience. There is not much specific documentation online though. You can get pretty far by following the Spring MVC documentation since the Porlet framework is almost identical. Liferay is a good Portal, but might be more than you need. – pjesi Sep 16, 2008 at 20:15

2

Tiles can be a pain. Vast improvement over what came before (i.e. nothing), but rather limiting.

Wicket might be more what you're looking for, unless you've settled on JSP.

Share Improve this answer Follow

answered Sep 16, 2008 at 12:32

sblundy
61.4k ● 22 ● 123 ● 124

Can you comment on why "Tiles can be a pain"? – matt b Feb 3, 2009 at 18:04

2

I don't recommend using Portlets unless your application is truly a web portal.

If you just want "reusable components" use JSP tagfiles, they are dead simple yet extremely powerful, since they are the same as regular JSPs.

I've had experience using tiles and the complexity involved simply isn't worth it.

Share Improve this answer

Follow

answered Jan 8, 2009 at 19:28

bpapa

21.5k ● 25 ● 100 ● 147

1

I'm a big fan of GWT. It lets you write your components as normal Java classes and then you can insert them into your pages at will. The whole thing ends up being compiled to Javascript.

Here's an example:

```
public class MyApplication implements EntryPoint, Hist
{
    static final String INIT_STATE = "status";

    /**
     * This is the entry point method.  Instantiates t
     */
    public void onModuleLoad ()
    {
        RootPanel.get ().setStyleName ("root");
        initHistorySupport ();
    }
```

```
    private void initHistorySupport ()
    {
        History.addHistoryListener (this);

        // check to see if there are any tokens passed
browser's URI
        String token = History.getToken ();
        if (token.length () == 0)
        {
            onHistoryChanged (INIT_STATE);
        }
        else
        {
            onHistoryChanged (token);
        }
    }


    /**
     * Fired when the user clicks the browser's 'back'
     *
     * @param historyToken the token representing the
     */
    public void onHistoryChanged (String historyToken)
    {
        RootPanel.get ().clear ();
        Page page;
        if (Page1.TOKEN.equalsIgnoreCase (historyToken
        {
            page = new Page1 ();
        }
        else if (Page2.TOKEN.equalsIgnoreCase (history
        {
            page = new Page2 ();
        }
        else if (Page3.TOKEN.equalsIgnoreCase (history
        {
            page = new Page3 ();
        }
        RootPanel.get ().add (page);
    }
}
```

Share Improve this answer

Follow

answered Sep 16, 2008 at 12:45

jodonnell

50.4k ● 11 ● 64 ● 67

1

I had a lot of experience with portlets in conjunction with Ajax JSF (IceFaces) and Liferay Portal and I wouldn't recommend them to anyone - everything looks good when reading tutorial and real hell in practice. Of course I think they are much more convenient and lightweight with Spring MVC and JSP, but anyway, portlets aren't well supported technology imho.

Share Improve this answer

Follow

answered Jul 21, 2009 at 15:33

Phil Gounbin

41 ● 1 ● 4

0

I'm not 100% sure what "reusable components" means in this context, but if you mean that you want certain common elements to appear on every page, such as banner, footer, navigation links, etc., then look no further than SiteMesh. My team has used it successfully on a couple of internationalised web applications.

Share Improve this answer

Follow

answered Sep 17, 2008 at 9:04

Andrew Swan

13.6k ● 23 ● 73 ● 99

Tapestry is a Java web app framework with an emphasis on easily creating reusable components.

0

I have used sitemesh, and it is good for wrapping a set of pages in standard headers and footers, but Tapestry is better for creating components which are used on many pages, possibly many times per page. Tapestry components can take other components as parameters, which allows the Sitemesh style wrapping.

Share Improve this answer Follow

edited Sep 17, 2008 at 10:23

answered Sep 16, 2008 at 12:33

tgdavies

11.4k ● 4 ● 39 ● 43

Hm, Tapestry looks promising... they have nice screencasts... tapestry.apache.org/tapestry5/screencast.html Will check it out, thnx a lot. – moriarty Sep 16, 2008 at 13:01