UsingCustomResourceSource

NOTE: This is outdated information that applies only to Tapestry 4.

Using Custom Resource Source

Thanks to HiveMind, using a custom resource source is as simple as providing your own implementation of the *tapestry.ComponentMessagesSource* service.

0. Assumptions

In the following example, we assume the resources a stored in a database with a structure similar to this:

label_id	key	locale	value
1	username	en_US	User Name
2	username	fr_CA	Nom d'Utilisateur
3	username	zn_CN	
4	password	en_US	Password
5	password	fr_CA	Mot de Passe
6	...	...	...

A label manager will be used to retrieve the message for a given key, depending of the requester's locale.

I am using Spring to access my DAOs, but you don't have to. Feel free to use your own approach.

Let's get started!

1. Provide an implementation of the ComponentMessagesSource Interface

Below is the DbComponentMessagesSourceImpl class, my implementation of the ComponentMessagesSource interface.

```
package foo.web.tapestry;

import java.util.Locale;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.hivemind.Messages;
import org.apache.tapestry.IComponent;
import org.apache.tapestry.services.ComponentMessagesSource;

import foo.service.ILabelManager;

/**
 *
 *
 * @author Serge Eby
 * @version $Rev: 84 $
 *
 */
public class DbComponentMessagesSourceImpl implements ComponentMessagesSource {

    private static Log
        _logger = LogFactory.getLog(DbComponentMessagesSourceImpl.class);
    private ILabelManager _labelManager;

    /**
     * Retrieves the label manager service
     * @return ILabelManager the current label manager service
     */
    public ILabelManager getLabelManager() {
        return _labelManager;
    }


    /**
     * Sets the label manager service
     * @param aService the label manager service to set
     */
    public void setLabelManager(ILabelManager aService) {
        _labelManager = aService;
    }

    /*
     * @see org.apache.tapestry.services.ComponentMessagesSource#getMessages(
     * org.apache.tapestry.IComponent)
     */
    public Messages getMessages(IComponent aComponent) {
        Locale locale = null;
        if (aComponent == null) {
            if (_logger.isDebugEnabled()) {
                _logger.info("Component is null, default locale will be used");
            }
        }
        else {
            locale = aComponent.getPage().getLocale();
            if (_logger.isDebugEnabled()) {
                _logger.info("Component Locale is " + locale );
            }
        }
        return new DbComponentMessages(locale, getLabelManager());
    }

}
```

2. Extend the AbstractMessages class

Since the Properties object defined in the default *org.apache.tapestry.services.impl.ComponentMessages* class cannot be null, you have to provide your own version of the AbstractMessages class. The DbComponentMessages class below rely on the ILabelManager service to pull the messages instead of the Properties object:

```
package foo.web.tapestry;

import java.util.Locale;


import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.hivemind.impl.AbstractMessages;
import foo.service.ILabelManager;

/**
 * This class retrieve messages from a database
 *
 * @author Serge Eby
 * @version $Rev: 84 $
 *
 */
public class DbComponentMessages extends AbstractMessages {

    private static Log    _logger = LogFactory.getLog(DbComponentMessages.class);
    private ILabelManager _labelManager;
    private Locale        _locale;

    /**
     * Constructor
     * @param aLocale the current locale
     * @param aSerivce the label manager service
     */
    public DbComponentMessages(Locale aLocale, ILabelManager aService) {
        if (aService == null) {
            if (_logger.isDebugEnabled()) {
                _logger.debug("Label manager service is null");
            }
        }
        if (aLocale == null) {
            _logger.debug("Locale is null");
        }

        _labelManager = aService;
        _locale       = aLocale;
        aService.setLocale(aLocale);
    }

    /**
     * Retrieve the locale
     */
    @Override
    protected Locale getLocale() {
        return _locale;
    }

    /**
     * Get the value for a given key
     * @param aKey the key to use when retrieving messages from the database
     * @return String the value found
     */
    @Override
    protected String findMessage(String aKey) {
        return _labelManager.getLabelValue(aKey);
    }

}
```

3. Configure HiveMind

Implement the service in the hivemodule.xml configuration file. Below is the relevant information:

```
<?xml version="1.0"?>
<module id="foo" version="1.0.0">
    <!-- Wires spring beans -->
          ...

     <!-- ASOs -->
          ...

    <!-- Localization -->
    <implementation service-id="tapestry.ComponentMessagesSource">
        <invoke-factory>
            <construct class="foo.web.tapestry.DbComponentMessagesSourceImpl">
                <set-object property="labelManager" value="spring:labelManager"/>
            </construct>
        </invoke-factory>
    </implementation>


</module>
```

Here, the labelManager is a Spring bean defined in the applicationContext.xml. Please refer to Tapestry4Spring for details on how to wire Spring into Tapestry4.

4. Implement the ILabelManager Interface

Here is the ILabelManager Interface:

```
package foo.service;

import java.util.List;
import java.util.Locale;

import foo.model.Label;

/**
 *
 *
 * @author Serge Eby
 * @version $Rev: 84 $
 *
 */
public interface ILabelManager {
    /**
     * Sets the locale or default locale if null
     * @param aLocale the locale to set
     */
    public void setLocale(Locale aLocale);

    /**
     * Retrieves the label value
     * @param aLabelKey the key to search for
     * @return the label string
     */
    public String getLabelValue(String aLabelKey);
    ...
}
```

Provide your implementation of the ILabelManager and configure your Spring applicationContext File appropriately (if using Spring of course). For better performance, the label manager implementation should have a caching mechanism to avoid accessing the database for each request.

```
package foo.service.impl;

import java.util.Map;
import foo.service.ILabelManager;

/**
 *
 *
 * @author Serge Eby
 * @version $Rev: 84 $
 *
 */
public class LabelManagerImpl implements ILabelManager {
   private Map _cache;
     ...
}
```

5. Use it

You are now all set! In your html template, you can do:

```
<span key="username">Username</span>

or

<span jwcid="@Insert" value="message:username">Username</span>
```

If using Tapestry annotations, doing:

```
public abstract FooPage extends BasePage {
    @Message
    public abstract String getUsername();

    ...
}
```

will retrieve the localized username value from the database.

Feel free to send any comments or suggestions.

[LutzHuehnken] asks: If I understand correctly, this will affect *all* components. Is there a way to achieve this behaviour just for my own, custom components?