User Interface - Feature #4286

support user-specific offline storage that is provided by the client's UI driver

09/04/2019 10:18 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Sergey Ivanovskiy	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to User Interface - Feature #3880: enhanced browse 3rd phase of impro...	**WIP**
Related to User Interface - Feature #4517: optionally back the 4GL features f...	**Closed**
Related to User Interface - Feature #4854: origin affinity	**Closed**

History

#1 - 09/04/2019 10:44 AM - Greg Shah

In #3261, #3706 and #3880 we have seen a requirement to store data about the state of the user's browse layout/customization selections. An approach was made to implement directory.xml storage using the user's FWD account.

This approach does not work when the same FWD account is shared between many users. For example, the default approach to security in FWD is to emulate the Progress approach which means that there is a default account shared by all users. This breaks the browse configuration persistence approach.

A better approach must be implemented:

- For web clients, the web browser's offline storage API can be used for this purpose. I think the Window.localStorage is the way to go.
- For the other clients, the user's OS account can be used in the same way. I was thinking java.util/prefs.Preferences.userRoot() might make sense.

I believe we need the ability to store and retrieve data in a hierarchical structure.

Stanislav: Can you please provide some requirements in regard to the data types and usage patterns needed?

The idea here is to expose this to both the FWD client (e.g. GUI browse widget) and the FWD server (we may need to provide a mechanism for this out to 4GL code). I want to keep the API simple.

Then in #3880, we would remap our current usage into this client-local offline storage instead of in the directory.

#2 - 09/04/2019 10:51 AM - Greg Shah

- Related to Feature #3880: enhanced browse 3rd phase of improvements added

#3 - 09/04/2019 11:36 AM - Stanislav Lomany

> Stanislav: Can you please provide some requirements in regard to the data types and usage patterns needed?

Not sure what should I advise. One can call BrowseModel.applyEnhancedConfig on the client side or BrowseWidget.applyEnhancedConfig on the server side to apply an enhanced config.

#4 - 09/04/2019 11:54 AM - Greg Shah

Please document what data types are needed. For example, do you need boolean, integer, decimal, date, string....? Document any important structure that is needed. For example, how deep can the hierarchy be? Do you use arrays?

We need the to ensure that the data structures you need will be supported by this new facility.

#5 - 09/04/2019 11:58 AM - Constantin Asofiei

Saving things at browser side would mean the user will not see these if he switches computers/browsers.

Can we instead add a resource plugin which can be implemented by the customer code to provide more details to whom the settings belong?

#6 - 09/04/2019 12:37 PM - Greg Shah

> Saving things at browser side would mean the user will not see these if he switches computers/browsers.

I understand this limitation. The existing customer systems already have this same limit. It is OK for now.

> Can we instead add a resource plugin which can be implemented by the customer code to provide more details to whom the settings belong?

I really want to avoid the directory storage. It is very complicated because it requires a large set of tricky ACLs to open up editing for the right users. I think it was a mistake to put it there and the local storage is a better idea.

#7 - 09/04/2019 05:00 PM - Stanislav Lomany

> Please document what data types are needed. For example, do you need boolean, integer, decimal, date, string....? Document any important structure that is needed. For example, how deep can the hierarchy be? Do you use arrays?

We need string, integer and boolean. We store a set of enhanced browse configs. Each browse config has a set of column configs.

#8 - 09/04/2019 05:41 PM - Greg Shah

> We store a set of enhanced browse configs. Each browse config has a set of column configs.

It is not clear how each item in the "set" is differentiated. Are there unique names for each browse config or each column?

Perhaps it will help if you give an example of the stored data.

#9 - 09/05/2019 09:49 AM - Stanislav Lomany

> It is not clear how each item in the "set" is differentiated. Are there unique names for each browse config or each column?

Each browse config has an unique arbitrary name just for storage purposes. proc_name and browse_name specify the target browse. If not specified, config is applied to all browses. user_name specifies the user name. If not specified, config is applied for all users.

Column configs are keyed by column_key which normally is the name of the backing field.

> Perhaps it will help if you give an example of the stored data.

```
<node class="enhancedBrowseConfig" name="1">
   <node-attribute name="proc_name" value="browse.p"/>
   <node-attribute name="browse_name" value="br"/>
   <node-attribute name="user_name" value="thebogus"/>
   <node-attribute name="fgcolor" value="#8080ff"/>
   <node-attribute name="font" value="DejaVu Sans,11,false,false,false"/>
   <node-attribute name="row_separators" value="FALSE"/>
   <node-attribute name="column_separators" value="TRUE"/>
   <node-attribute name="row_height_px" value="14"/>
   <node-attribute name="labels_height_px" value="17"/>
   <node class="container" name="columns">
      <node class="enhancedColumnConfig" name="1">
         <node-attribute name="column_key" value="tt.f1"/>
```

```
         <node-attribute name="ordinal" value="1"/>
         <node-attribute name="visible" value="TRUE"/>
         <node-attribute name="width_px" value="70"/>
         <node-attribute name="bgcolor" value="#b7ffdb"/>
         <node-attribute name="label_font" value="DejaVu Sans,11,true,false,false"/>
      </node>
      <node class="enhancedColumnConfig" name="2">
         <node-attribute name="column_key" value="tt.f2"/>
         <node-attribute name="ordinal" value="0"/>
         <node-attribute name="visible" value="TRUE"/>
         <node-attribute name="width_px" value="21"/>
      </node>
   </node>
</node>
```

#10 - 09/10/2019 08:38 AM - Sergey Ivanovskiy

EnhancedBrowseConfigManager reads the enhanced browse configuration from the directory. It seems that this task requires to implement the service that can be provided to EnhancedBrowseConfigManager. This service provides methods to read/write from/to the directory if its back end is a directory, otherwise its methods can read/write from/to the client. The bottleneck can be that the read and write operations can frozen the client UI for a while.
Is this understanding correct?
p2j.js already uses the loacalStorage.

```
   /**
    * Save the target JS object serialized to a string value with the given key.
    *
    * @param    {String} key
    *           The assigned key.
    * @param    {Object} obj
    *           The target JS object.
    * @param    {Boolean} local
    *           True value indicates that the object is persisted between opened and new sessions,
    *           otherwise the object is stored during the current session.
    */
   function saveObject(key, obj, local)
   {
      var sObj = JSON.stringify(obj);
      var storage;
      if (local)
      {
         storage = localStorage;
      }
      else
      {
         storage = sessionStorage;
      }
      storage.setItem(key, sObj);
      return sObj;
   }

   me.saveObject = saveObject;

   /**
    * Try to restore a JS object from the retrieved string value for the given key.
    *
    * @param    {String} key
    *           The provided key.
```

```
     * @param    {Boolean} local
     *           True value indicates that the local storage is used, otherwise the session
     *           storage is used.
     *
     * @return   {Object} obj
     *           The target JS object or null.
     */
    function restoreObject(key, local)
    {
       var storage;
       if (local)
       {
          storage = localStorage;
       }
       else
       {
          storage = sessionStorage;
       }
       var sObj = storage.getItem(key);
       if (sObj)
       {
          var obj = JSON.parse(sObj);
          return obj;
       }
       return null;
    }

    me.restoreObject = restoreObject;

    /**
     * Delete a JS object by the given key.
     *
     * @param    {String} key
     *           The provided key.
     * @param    {Boolean} local
     *           True value indicates that the local storage is used, otherwise the session
     *           storage is used.
     */
    function deleteObject(key, local)
    {
       var storage;
       if (local)
       {
          storage = localStorage;
       }
       else
       {
          storage = sessionStorage;
       }
       storage.removeItem(key);
    }

    me.deleteObject = deleteObject;
```

It is a key and value storage. For the java client we are going to use Preferences.userRoot() that is also a key and value storage.

Does it require to implement special service methods to persist tree data structures if we can persist any serializable java object?

#11 - 09/10/2019 08:39 AM - Sergey Ivanovskiy

Created task branch 4286a for this task.

#12 - 09/10/2019 09:52 AM - Greg Shah

> This service provides methods to read/write from/to the directory if its back end is a directory, otherwise its methods can read/write from/to the client.

I think in practice, we will only ever store some defaults in the directory. The ACL/permissions on editing the directory are not workable for per-user or per-group storage. I think it is perfectly acceptable to hand-code the enhanced browse default settings in the directory.

Stanislav: With this in mind, do we need the storage service to provide both directory and client-local storage back ends?

> The bottleneck can be that the read and write operations can frozen the client UI for a while.

Is there really so much data that it will take any appreciable amount of time to read?

> Does it require to implement special service methods to persist tree data structures if we can persist any serializable java object?

I agree this is probably OK.

We can serialize to JSON and store the result as a string. I think we use JSON for serialization **even for the Swing client case**. This will keep the result simple and easy to read. There is no reason to use XML or a binary Java serialization approach.

#13 - 09/10/2019 03:08 PM - Stanislav Lomany

> Stanislav: With this in mind, do we need the storage service to provide both directory and client-local storage back ends?

So, as far as I get, we need read/write functionality for the client and read functionality for the directory. Do we need write functionality for the

directory? Why not, especially considering that it already works?

#14 - 09/10/2019 03:31 PM - Greg Shah

> Why not, especially considering that it already works?

It is my impression that configuring directory edit permissions is prohibitively complex for users/groups. This would suggest that only admin users would be practical to enable for this mode. Please confirm this or correct my mis-understanding.

Still, even with this limit the admin case may still be useful.

On the down side, the directory storage uses its own schema changes instead of storing a single JSON string (which has the entire hierarchical object graph encoded in a single string). This means the back end may be a bit more complex to handle both cases.

OK, we can go ahead with this.

#15 - 09/10/2019 05:02 PM - Stanislav Lomany

> It is my impression that configuring directory edit permissions is prohibitively complex for users/groups.

We do admin/non-admin check for global setting. And use SecurityManager.getUserId for user-specific settings. Config is stored in a separate section where normal directory security is not applied.

#16 - 09/11/2019 01:53 AM - Sergey Ivanovskiy

Stanislav Lomany wrote:

> > Stanislav: With this in mind, do we need the storage service to provide both directory and client-local storage back ends?
>
> So, as far as I get, we need read/write functionality for the client and read functionality for the directory. Do we need write functionality for the directory? Why not, especially considering that it already works?

Please explain where this new storage service is expected to use. I suppose that it must be used to store enhanced browse settings on the client side but this note confuses me. Please explain what write functionality are required for this storage service. If this service persists into the directory and the read functionality are implemented, then please help to understand what are implementation java classes and what are absent that should be

implemented.

#17 - 09/11/2019 05:39 AM - Stanislav Lomany

> Please explain where this new storage service is expected to use.

I think that Greg can provide a better vision here, but as I see it, admin can visually set global settings for all browses (e.g. make all browses striped) and save them in the directory, while users will store their settings on the client side.

> If this service persists into the directory and the read functionality are implemented, then please help to understand what are implementation java classes and what are absent that should be implemented.

Directory read/write works. You have to add client-side storage.

#18 - 09/11/2019 07:13 AM - Greg Shah

Stanislav Lomany wrote:

> > Please explain where this new storage service is expected to use.
>
> I think that Greg can provide a better vision here, but as I see it, admin can visually set global settings for all browses (e.g. make all browses striped) and save them in the directory, while users will store their settings on the client side.

Yes, exactly.

> > If this service persists into the directory and the read functionality are implemented, then please help to understand what are implementation java classes and what are absent that should be implemented.
>
> Directory read/write works. You have to add client-side storage.

The "both directory and client local storage will work" is something for the enhanced browse to implement. For your task, just focus on implementing the client local storage facility.

#19 - 09/11/2019 07:54 AM - Sergey Ivanovskiy

Stanislav Lomany wrote:

> Please explain where this new storage service is expected to use.

I think that Greg can provide a better vision here, but as I see it, admin can visually set global settings for all browses (e.g. make all browses striped) and save them in the directory, while users will store their settings on the client side.

The admin can reset all settings for a given user even if this user has a personal setting on the client side, correct? It looks like an integration task for the local storage. Should it be done within this task?

> Directory read/write works. You have to add client-side storage.

OK.

#20 - 09/11/2019 07:59 AM - Sergey Ivanovskiy

Greg Shah wrote:

> If this service persists into the directory and the read functionality are implemented, then please help to understand what are implementation java classes and what are absent that should be implemented.

> Directory read/write works. You have to add client-side storage.

The "both directory and client local storage will work" is something for the enhanced browse to implement. For your task, just focus on implementing the client local storage facility.

OK. As an independent task it should be easy to implement. Are there additional requirements for this storage service in order to be integrated with the enhanced browse?

#21 - 09/11/2019 08:01 AM - Stanislav Lomany

> The admin can reset all settings for a given user even if this user has a personal setting on the client side, correct?

At this point behavior is the opposite: if we have an user-specific configuration, it overwrites global "admin" config. But when a user creates new config, it inherits the current "admin" config. So, we do not "merge" configs and admin have no access to user configs except than editing config with hands.

#22 - 09/11/2019 08:56 AM - Greg Shah

> The admin can reset all settings for a given user even if this user has a personal setting on the client side, correct?

As Stanislav notes, we don't have to do that.

On the other hand, it does seem prudent to provide a way to delete an entry.

#23 - 09/11/2019 10:15 AM - Sergey Ivanovskiy

I was going through the directory settings for hotel_gui and testcases and could find only these settings

```
    <node class="container" name="server">
      <node class="container" name="default">
        <node class="container" name="enhancedBrowse">
          <node class="container" name="embedded">
            <node class="boolean" name="applyLayout">
              <node-attribute name="value" value="TRUE"/>
            </node>
            <node class="boolean" name="changeLayout">
              <node-attribute name="value" value="FALSE"/>
            </node>
            <node class="boolean" name="enhancedSorting">
              <node-attribute name="value" value="TRUE"/>
            </node>
            <node class="boolean" name="enhancedFiltering">
              <node-attribute name="value" value="TRUE"/>
            </node>
            <node class="boolean" name="exporting">
              <node-attribute name="value" value="TRUE"/>
            </node>
            <node class="boolean" name="hyperlinking">
              <node-attribute name="value" value="TRUE"/>
            </node>
            <node class="container" name="configurations">
              <node class="enhancedBrowseConfig" name="1">
```

```
                <node-attribute name="alt_bgcolor" value="#ddeffe"/>
                <node-attribute name="row_separators" value="FALSE"/>
                <node-attribute name="column_separators" value="TRUE"/>
                <node-attribute name="separators_color" value="#b0b0b0"/>
              </node>
            </node>
          </node>
        </node>
```

Stanislav, could you help with testcases from the testcases project that explore these functionality for saving and restore enhanced browse configurations?

#24 - 09/11/2019 10:19 AM - Greg Shah

There are no such testcases. Stanislav will handle the enhanced browse part. I think you can focus on defining the API and providing the API to the client and on the server.

#25 - 09/11/2019 10:21 AM - Stanislav Lomany

> Stanislav, could you help with testcases from the testcases project that explore these functionality for saving and restore enhanced browse configurations?

1. Enable enhanced layout using config from "Quick-out" section https://proj.goldencode.com/projects/p2j/wiki/Browse_Widget_Enhancements
2. You can edit any browse using browse right-click menu.

#26 - 09/11/2019 10:37 AM - Sergey Ivanovskiy

Stanislav Lomany wrote:

> > Stanislav, could you help with testcases from the testcases project that explore these functionality for saving and restore enhanced browse configurations?
>
> 1. Enable enhanced layout using config from "Quick-out" section https://proj.goldencode.com/projects/p2j/wiki/Browse_Widget_Enhancements
> 2. You can edit any browse using browse right-click menu.

Thank you. It will help me to test new functionality.

#27 - 09/12/2019 05:01 AM - Sergey Ivanovskiy

It seems that the web client can use java.util.prefs.Preferences.userRoot() as the other clients. Does it really need to persist data on the web client browser side? This question is just to clarify the positive and negative arguments for using this solution.

#28 - 09/12/2019 08:37 AM - Greg Shah

Sergey Ivanovskiy wrote:

> It seems that the web client can use java.util.prefs.Preferences.userRoot() as the other clients. Does it really need to persist data on the web client browser side? This question is just to clarify the positive and negative arguments for using this solution.

With the web client, there may be many users sharing the same OS user. If we use java.util.prefs.Preferences.userRoot(), then all of these users would be storing their customizations together, making a big mess. The idea here is that the browser's local storage is as close to a per-user facility as we can get.

#29 - 09/12/2019 12:43 PM - Sergey Ivanovskiy

I didn't check that the Java Preferences users key and value storage has the following limits for the maximum length of string keys, the maximum stored bytes per a value of a key and the maximum length of the node name

```
    /**
     * Maximum length of string allowed as a key (80 characters).
     */
    public static final int MAX_KEY_LENGTH = 80;
```

```
    /**
     * Maximum length of string allowed as a value (8192 characters).
     */
    public static final int MAX_VALUE_LENGTH = 8*1024;
```

```
    /**
     * Maximum length of a node name (80 characters).
     */
    public static final int MAX_NAME_LENGTH = 80;
```

Thus we have limits even if we will map a plain java object into the user Preferences tree with nodes of Preferences but in this case json serialization doesn't help and it needs to traverse each java object by some custom code.

#30 - 09/13/2019 05:46 AM - Sergey Ivanovskiy

- File 4286a_1.patch added

The committed revision 11329 (4286a) implemented the client storage. I didn't test this version, but please review the code to be sure that it is going forward in the correct direction. Testing this code.

#31 - 09/13/2019 05:46 AM - Sergey Ivanovskiy

- Status changed from New to WIP

- % Done changed from 0 to 70

#32 - 09/13/2019 08:26 AM - Sergey Ivanovskiy

The used backed data storages, Preferences and localStorage have the maximal size of bytes that can be stored in one key value. It needs to split the data into chunks and then to collect them or to implement another data storages based on the browser's identity (https://fingerprintjs.com/).

#33 - 09/13/2019 08:46 AM - Greg Shah

Please implement "chunking" where you split the entry into pieces. Hide this behind the FWD client storage API but inside the common Java code in the client. This allows the caller to provide a single larger JSON string to store and the API handles the issue as needed. I don't want the caller of the API to have to deal with this.

#34 - 09/13/2019 09:02 AM - Greg Shah

Code Review Task Branch 4286a Revision 11329

Overall, it is going in the right direction.

1. Why is UserPreferences in the com.goldencode.p2j.ui package. I think it is better in com.goldencode.p2j.ui.client.

2. Please always place your implements and extends on separate lines as required in the coding standards.

3. Please leave a blank line in between members in a class. For example:

```
   /** The client local storage proxy */
   private ObjectStorage clientLocalStorage;  // there should be a blank line between this member and the cons
tructor
   /**
    * Constructor.
    */
```

#35 - 09/15/2019 04:32 PM - Sergey Ivanovskiy

Please review the committed rev 11330 (4286a) that should fix the code according to the first review. I haven't implemented chunks for the web storage yet because it can hold 5Mb of data and it should be enough to store user settings. Another weak place is unique keys that should be used to store chunks of large data.
In the case for Preferences API I solved these problems by using random unique names UUID for the preferences node associated with chunks of data. We can't use the key as a name because a key can contain slash chars which are not accepted as node names. The web local storage case is different because it is a simple map and it requires to generate unique keys for chunks. It seems that the solution with uuid or some special generated

keys for chunks doesn't solve unique keys issue.

#36 - 09/16/2019 06:28 AM - Sergey Ivanovskiy

- Status changed from WIP to Review

- % Done changed from 70 to 100

Please review committed revision 11331 (4286a) that added common class UserKeyValueStorage in order to manage chunks of data.

#37 - 09/17/2019 02:52 PM - Sergey Ivanovskiy

Now with revision 11331 (4286a), we can use these methods to save/retrieve EnhancedBrowseConfig

```
EnhancedBrowseConfig conf = new EnhancedBrowseConfig();

..................................................

ThinClient.getInstance().getClientStorage().setObject("key1", conf);

..................................................

EnhancedBrowseConfig conf = (EnhancedBrowseConfig) ThinClient.getInstance().getClientStorage().getObject("key1
", EnhancedBrowseConfig.class);

..................................................

EnhancedBrowseConfig conf = (EnhancedBrowseConfig) LogicalTerminal.getClientStorage().getObject("key1", Enhanc
edBrowseConfig.class);

..................................................................

LogicalTerminal.getClientStorage().setObject("key1", conf);

..................................................
```

Is it enough for this task?

#38 - 09/17/2019 03:26 PM - Greg Shah

I think so. Can you delete an item using ...setObject("key1", null)?

#39 - 09/17/2019 03:52 PM - Sergey Ivanovskiy

Greg Shah wrote:

> I think so. Can you delete an item using ...setObject("key1", null)?

It doesn't delete the key from this storage. We can use LogicalTerminal.getClientStorage().deleteKeyAndValue("key1") and ThinClient.getInstance().getClientStorage().deleteKeyAndValue("key1").

#40 - 09/17/2019 03:55 PM - Sergey Ivanovskiy

- File EnhancedBrowseConfig.patch added

Sorry, that I didn't note that in order to serialize and desirialize EnhancedBrowseConfig I used this patch EnhancedBrowseConfig.patch.

#41 - 09/17/2019 04:19 PM - Greg Shah

Sergey Ivanovskiy wrote:

> Greg Shah wrote:
>
> > I think so. Can you delete an item using ...setObject("key1", null)?
>
> It doesn't delete the key from this storage. We can use LogicalTerminal.getClientStorage().deleteKeyAndValue("key1") and ThinClient.getInstance().getClientStorage().deleteKeyAndValue("key1").

This is OK. How about changing deleteKeyAndValue() to deleteObject()?

#42 - 09/17/2019 04:21 PM - Greg Shah

Stanislav: Are you OK with this?

#43 - 09/17/2019 04:22 PM - Sergey Ivanovskiy

- File 4286a_summary.patch added

Ok. Please review this summary changes rev 11332 (4286a). The last changes were the fix for EnhancedBrowseConfig to be deserialized by Jackson, the fix for Jackson object mapper that it doesn't fail on unknown property.

#44 - 09/17/2019 04:27 PM - Sergey Ivanovskiy

Greg Shah wrote:

> How about changing deleteKeyAndValue() to deleteObject()?

This name is more convenient for me because this storage is a key and value storage and deleteKeyAndValue can be used for any key and value.

#45 - 09/17/2019 04:30 PM - Sergey Ivanovskiy

I would like to change deleteKeyAndValue(String key) to delete(String key) in this case. Greg, is it OK for you?

#46 - 09/17/2019 04:54 PM - Greg Shah

Sergey Ivanovskiy wrote:

> I would like to change deleteKeyAndValue(String key) to delete(String key) in this case. Greg, is it OK for you?

OK.

#47 - 09/17/2019 05:31 PM - Sergey Ivanovskiy

Committed rev 11333(4286a) renamed deleteKeyAndValue. I tested EnhancedBrowseConfig with the web client too. It seems that it makes sense to implement
toString() method for LogicalTerminal.getClientStorage().setObject(config.getStorageKey().toString(), config);.

#48 - 09/18/2019 05:14 AM - Stanislav Lomany

Can we iterate stored entries or get list of them?

#49 - 09/18/2019 07:05 AM - Sergey Ivanovskiy

Stanislav Lomany wrote:

> Can we iterate stored entries or get list of them?

The web client uses localStorage on the browser side and this key and value storage exposes Storage interface that can return the number of stored items and the key of a stored item given by index. Preferences has keys() method that returns all string keys stored by this node. Now I didn't expose these methods for usages because for a one key we can use several keys to store the value due to the limitation of these storages.

Please help to understand when you need these methods. Especially if we are able to persist arrays of objects and list of objects using this client storage. Actually we can add keys() to our client storage but this method will be expensive because it will iterate with the client storage back end and collect results.

#50 - 09/18/2019 07:33 AM - Sergey Ivanovskiy

Preferences storage has the following requirement on keys that the maximal key length must not exceed 80.

Thus, from #4286-49 we can expose

```
public String[] keys();
```

if it is urgently required for the usage.

#51 - 09/18/2019 05:16 PM - Stanislav Lomany

> Please help to understand when you need these methods.

I think that the order will be following:

1. Server asks client for an appropriate configuration. Depending on result, client-side or server-side configuration can be applied.
2. Greg, do we want to load all of the configs in memory on the client side, as we do on the server side? If not, we can key the configurations in the client storage with a string combined of procedure name and browse name (and user name?). With key length limited to 80, we can use hash of that combination as the key and use list/array capability for collision resolution.
3. In any case, the result of round-trip to client is cached on the server side.
4. We this approach, we don't need to list the entries. But I think that potentially we may need to delete all of the configurations on the client side (or specific subset of them), so it's nice to have keys() for the case.

#52 - 09/18/2019 06:05 PM - Greg Shah

> Greg, do we want to load all of the configs in memory on the client side, as we do on the server side?

I don't think so, unless you have a reason this makes sense. I think each enhanced browse would load its specific config if needed.

#53 - 09/19/2019 01:36 AM - Sergey Ivanovskiy

Greg, Stanislav,
Please clarify my part of this task. Should we add String[] keys(); or expose int getSize(); and String getKey(int index);?
Could I consider my part of this task completed?

#54 - 09/19/2019 06:20 AM - Stanislav Lomany

> I don't think so, unless you have a reason this makes sense. I think each enhanced browse would load its specific config if needed.

So, if we're not going to load it, then we need to be able to find an entry using the key. Entries can be keyed by procedure name + browse name + user name. Procedure name can get easy go over the 80 characters limitation, so looks like we need to hash the keys. In order to resolve collisions, we may put several entries in an array under the specific key. Any better ideas?

#55 - 09/19/2019 06:42 AM - Sergey Ivanovskiy

Stanislav Lomany wrote:

> > I don't think so, unless you have a reason this makes sense. I think each enhanced browse would load its specific config if needed.
>
> So, if we're not going to load it, then we need to be able to find an entry using the key. Entries can be keyed by procedure name + browse name + user name. Procedure name can get easy go over the 80 characters limitation, so looks like we need to hash the keys. In order to resolve collisions, we may put several entries in an array under the specific key. Any better ideas?

Stanislav, we can move to another key and value databases for user settings or can change UserPreferences to work with any keys. Preferences is a tree and a given key can be mapped into the branch of this tree and then the given settings are stored. I would prefer to use some NOSQL embedded database that have an open source license and with help of browse identity it gives a powerful solution. May be such solutions exceed our needs.

#56 - 09/19/2019 09:01 AM - Sergey Ivanovskiy

For the web client localStorage doesn't have such restrictions on its keys. You can use procedure name + browse name + user name.

#57 - 09/19/2019 09:17 AM - Greg Shah

OK, why not do the following?

- keys are optionally nested in a tree using / as the path separator (e.g. "procedure_name/browse_name/user_name")
- for web storage, the key is a simple string without breaking the parts into separate paths
- for user preferences, the individual parts can be used as paths in the tree so that we can properly store with less need for hashing

#58 - 09/19/2019 09:22 AM - Sergey Ivanovskiy

Greg, do you suppose that each one of procedure_name/browse_name/user_name doesn't exceed 80. Is it possible for procedure_name to have a name that exceeds this limit?
What do you think if Berkeley DB can be used to store key and value elements?

#59 - 09/19/2019 09:26 AM - Stanislav Lomany

- for user preferences, the individual parts can be used as paths in the tree so that we can properly store with less need for hashing

Quite elegant! I don't mind (if we'll have API that transparently handles creation/deletion/enumeration stuff without any kind of tree logic).

#60 - 09/19/2019 09:41 AM - Greg Shah

> do you suppose that each one of procedure_name/browse_name/user_name doesn't exceed 80

Yes, this will be a limit of the API. The caller can handle this part.

> What do you think if Berkeley DB can be used to store key and value elements?

For the purpose of this use case, I prefer to avoid the extra dependency. In the future, we will be porting our FWD client to other platforms. For example, a mobile platform is a likely target. Limiting the 3rd party code needed will help us later. We can live with the limit of 80 chars per path element.

> if we'll have API that transparently handles creation/deletion/enumeration stuff without any kind of tree logic

Yes, exactly. Only the user preferences case needs this, so we hide the tree processing behind the API.

#61 - 09/19/2019 12:14 PM - Sergey Ivanovskiy

OK. Committed revision 11334 (4286a) changed UserPreferences in order to use long keys. Planning to add the following two methods in order to iterate over this storage

```
int getSize();
String getKey(int index);
```

#62 - 09/19/2019 04:17 PM - Sergey Ivanovskiy

Sergey Ivanovskiy wrote:

> OK. Committed revision 11334 (4286a) changed UserPreferences in order to use long keys. Planning to add the following two methods in order to iterate over this storage
> [...]

Committed revision 11335, 11336 (4286a) added these methods to iterate over this client storage.

#63 - 09/19/2019 05:03 PM - Sergey Ivanovskiy

For the web client the web storage can use keys to extend the value capacity. If getKey(i) returns null, then this key is used to store some value and can't be used. If key = getKey(i) is not null, then this key can store the value and this value can be overridden by some new value setString(key, newvalue); size = getSize() returns the number of all keys that includes keys to store a large value.

#64 - 09/20/2019 12:30 PM - Sergey Ivanovskiy

- File deleted (4286a_summary.patch)

#65 - 09/20/2019 12:32 PM - Sergey Ivanovskiy

- File 4286a_summary.patch added

Greg, please review the final changes that should fix found issues.

#66 - 09/20/2019 01:09 PM - Greg Shah

Is everything checked in?

#67 - 09/20/2019 01:54 PM - Sergey Ivanovskiy

Yes, the last committed revision is rev 11339.

#68 - 09/20/2019 02:57 PM - Greg Shah

Code Review Task Branch 4286a Revision 11339

I'm OK with the changes.

BrowseKey, EnhancedBrowseConfig, EnhancedBrowseConfigStorageKey each need a history entry.

Stanislav: Do you have any concerns or other requirements?

#69 - 09/23/2019 11:03 AM - Stanislav Lomany

> Stanislav: Do you have any concerns or other requirements?

I'm OK if the storage properly supports long keys and printing some errors with e.printStackTrace(); is acceptable.

Minor issues:

```
getOutputManager().getDriver().getClientStorage();
```

can be replaced with

```
OutputManager.getDriver().getClientStorage();
```

Shouldn't we log something is these cases?

```
catch (IOException e)
{
   return null;
}
```

#70 - 09/23/2019 06:33 PM - Sergey Ivanovskiy

The committed rev. 11341 (4286a) fixed found issues. The web client was tested with this diff

```
=== modified file 'src/com/goldencode/p2j/ui/EnhancedBrowseConfigManager.java'
--- src/com/goldencode/p2j/ui/EnhancedBrowseConfigManager.java    2018-10-18 19:51:17 +0000
+++ src/com/goldencode/p2j/ui/EnhancedBrowseConfigManager.java    2019-09-23 22:20:32 +0000
@@ -681,6 +681,8 @@

          // update in-memory config
          configs.put(config.getStorageKey(), config);
+         LogicalTerminal.getClientStorage().setObject(config.getStorageKey().toString(),
+                                                      config);
       }
       finally
       {
@@ -1083,6 +1085,14 @@

       try
       {
+         Object obj = LogicalTerminal.getClientStorage().getObject(
+                  new EnhancedBrowseConfigStorageKey(userId, configKey).toString(),
+                  EnhancedBrowseConfig.class);
+         if (obj instanceof EnhancedBrowseConfig)
+         {
+            return (EnhancedBrowseConfig) obj;
+         }
+
          return configs.get(new EnhancedBrowseConfigStorageKey(userId, configKey));
       }
       finally

=== modified file 'src/com/goldencode/p2j/ui/EnhancedBrowseConfigStorageKey.java'
--- src/com/goldencode/p2j/ui/EnhancedBrowseConfigStorageKey.java    2019-09-23 20:44:10 +0000
+++ src/com/goldencode/p2j/ui/EnhancedBrowseConfigStorageKey.java    2019-09-23 22:06:23 +0000
@@ -138,4 +138,16 @@
       result = 37 * result + configKey.hashCode();
       return result;
    }
+
+
+   public String toString()
+   {
+      StringBuilder key = new StringBuilder();
+
+      key.append(userName).append("/").append(configKey.ehKeyProcName).append("/")
+      .append(configKey.ehKeyBrowseName);
+
+      return key.toString();
+   }
+
 }
\ No newline at end of file

=== modified file 'src/com/goldencode/p2j/ui/client/gui/driver/web/WebClientKeyValueStorage.java'
--- src/com/goldencode/p2j/ui/client/gui/driver/web/WebClientKeyValueStorage.java    2019-09-20 16:18:45 +0000
+++ src/com/goldencode/p2j/ui/client/gui/driver/web/WebClientKeyValueStorage.java    2019-09-23 22:25:32 +0000
@@ -77,7 +77,7 @@
 extends UserKeyValueStorage
 {
    /** The maximal string length for values */
-   private static final int MAXIMAL_STRING_LENGTH = 4 * 1024 * 1024;
+   private static final int MAXIMAL_STRING_LENGTH = 300 /* 1024*/;

    /** The web socket object that is responsible for the communication with the js client */
    private final GuiWebSocket webSocket;
```

#71 - 09/24/2019 01:19 PM - Sergey Ivanovskiy

The branch 4286a was update over the trunc rev 11329 to revision 11342.

#72 - 09/24/2019 01:56 PM - Sergey Ivanovskiy

Stanislav Lomany wrote:

> Shouldn't we log something is these cases?

```
catch (IOException e)
{
   return null;
}
```

WebClientKeyValueStorage and UserPreferences has such code in the case when they read data from the back end storage. The local storage on the client side can be corrupted or manually deleted. Thus it seems that returning null is enough. And logging ((JsonProcessingException)e).printStackTrace() looks really bad but those exceptions can occur only if an object of Chunk type is not json-serializable. Thus they shouldn't depend on the back end storage and contained data but they depend only on the code of Chunk. If there are no objections, then this branch can be merged into the trunk.

#73 - 09/24/2019 02:01 PM - Greg Shah

Stanislav: It is your call. If you are OK, Sergey can merge to trunk.

#74 - 09/24/2019 03:41 PM - Stanislav Lomany

I'm OK with that.

#75 - 09/24/2019 04:38 PM - Greg Shah

Sergey: OK, please merge 4286a to trunk.

#76 - 09/25/2019 02:46 AM - Sergey Ivanovskiy

The rev. 11345 (4286a) fixed some missed history entries that were found when preparing this branch for the trunc.

The rev. 11345 (4286a) was merged into the trunc as the rev 11330. 4286a was archived.

#77 - 09/25/2019 08:06 AM - Greg Shah

- Status changed from Review to Closed

#78 - 01/23/2020 05:33 PM - Greg Shah

- Related to Feature #4517: optionally back the 4GL features for Registry access with the user-specific offline storage features added

#79 - 08/14/2020 12:53 PM - Greg Shah

- Related to Feature #4854: origin affinity added

Files

4286a_1.patch	49 KB	09/13/2019	Sergey Ivanovskiy
EnhancedBrowseConfig.patch	3.19 KB	09/17/2019	Sergey Ivanovskiy
4286a_summary.patch	81.9 KB	09/20/2019	Sergey Ivanovskiy