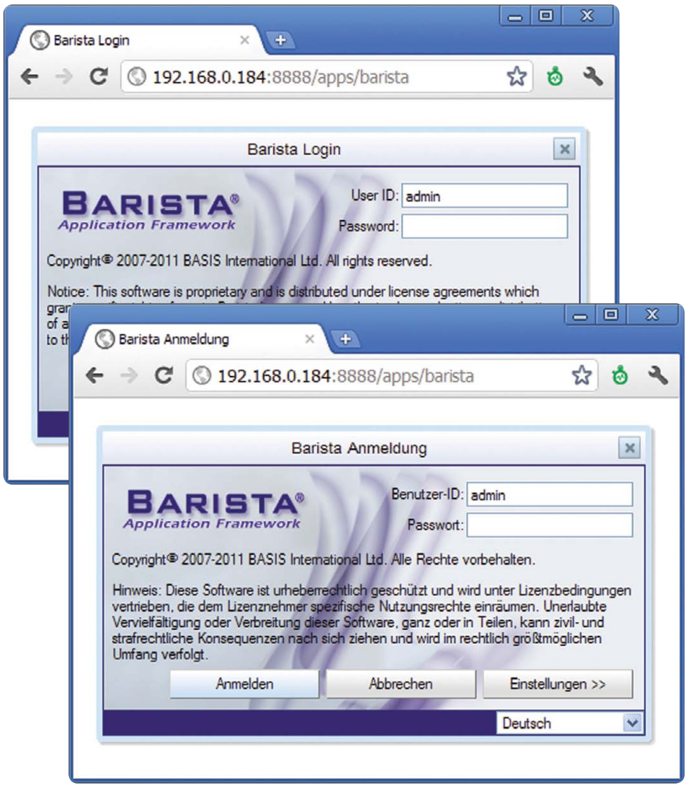
Parlez-BUI Français?

Localization is what we do to make our applications interact with users in their language and using currency, date, and other formats that are appropriate for their country and language. While there are some aspects of this process that can affect the back-end application such as currency conversions, localization is primarily user-focused. This article introduces the tools that BBj® and its browser user interface (BUI) provide to help build fully localized applications.

Language

The most obvious part of localization is language translation. The Barista® Application Framework login screen in **Figure 1** provides a convenient example of two languages.

Developers can implement language translation with a combination of approaches. For static screens stored in resource files, create multiple copies of each resource file, one for each supported language. If an application starts with a login screen, consider creating **`login_en.arc`**, **`login_de.arc`**, **`login_fr.arc`**, and so on. Dynamic text is stored in Java-style property resource bundles (refer to links.basis.com/fjizn). A property resource bundle is a set of text files containing key/value pairs for various locales. Files are named in the format **`login.properties`**, such as **`login_en.properties`**, **`login_de.properties`**, and so on. They contain key/value pairs like the samples shown in **Figure 2**. > >



Barista Login

192.168.0.184:8888/apps/barista

Barista Login

User ID: admin

Password:

Copyright® 2007-2011 BASIS International Ltd. All rights reserved.

Notice: This software is proprietary and is distributed under license agreements which

Barista Anmeldung

192.168.0.184:8888/apps/barista

Barista Anmeldung

Benutzer-ID: admin

Passwort:

Copyright® 2007-2011 BASIS International Ltd. Alle Rechte vorbehalten.

Hinweis: Diese Software ist urheberrechtlich geschützt und wird unter Lizenzbedingungen vertrieben, die dem Lizenznehmer spezifische Nutzungsrechte einräumen. Unerlaubte Vervielfältigung oder Verbreitung dieser Software, ganz oder in Teilen, kann zivil- und strafrechtliche Konsequenzen nach sich ziehen und wird im rechtlich größtmöglichen Umfang verfolgt.

Figure 1. Barista login in English and German

Properties file	Property (key=value)
login.properties	`userid=User ID:` `password=Password:`
login_de.properties	`userid=Benutzer-ID:` `password=Passwort:`
login_fr.properties	`userid=ID Utilisateur:` `password=Mot de passe:`

Figure 2. Sample properties

By Jim Douglas
Software Developer

The BBTranslator utility can manipulate Java properties files. If those properties files exist in the current directory, retrieve the German text like this:

```
use ::bbtranslator.bbj::BBTranslator
Login! = BBTranslator.getInstance("login","de","",dsk("")+dir(""))
print login!.getTranslation("userid")
print login!.getTranslation("password")
```

```
>run
Benutzer-ID:
Passwort:
```

Non-English text will almost always contain text that cannot be expressed in simple ASCII. To ensure that properties files will work correctly on any system, use the native2ascii Java utility to convert text from a local character set to a cross-platform encoding format that will work with any character set.

Character Set

Applications that deal only with English, or run only on a single platform (like Microsoft Windows), often ignore character set issues, implicitly assuming that the character set is windows-1252. Applications might assume, for example, that the following code will always set euro$ equal to the euro (€) symbol:

```
euro$ = $80$
euro$ = "€"
```

The first line explicitly assumes the Windows-1252 character set, which encodes € as $80$. The second line implicitly assumes the character set used when the program was created (probably Windows-1252 if the program was created on Windows). Both of these will fail when run on any other character set. For example, $80$ maps to Ä in the Mac OS X Mac Roman character set. To ensure that an application will work correctly with any character set, one should rather code something like this:

```
euro$ = new String($20ac$,"UTF-16"); rem ' Unicode UTF-16 encoding
euro$ = new String($e282ac$,"UTF-8"); rem ' Unicode UTF-8 encoding
euro$ = new String($80$,"windows-1252"); rem ' Windows Codepage 1252
```

The windows-1252 version will probably work across all platforms; the Unicode versions are guaranteed to work.

It is often possible to use HTML encoding. For example, this will popup a msgbox() with the € symbol as the message:

```
print msgbox("<html>&euro;")
```

BASIS resource (`.arc`) files embed their character set as a specially formatted comment on the first line (e.g., `//#charset: windows-1252`), so they are inherently cross-platform-safe.

Server and Client Locales

Over time, BBx® has accumulated a range of features to allow for building localized applications. The oldest of these is SETOPTS byte 3, bit $20$, combined with bytes 5 and 6, which can be used to localize numeric and currency formatting of the decimal point and the grouping separator. STBL("!DATE") can be used to customize date formats and text for the DATE() function. Those localization features were originally output-only; they were later extended to user input with the INPUTN and INPUTD GUI controls. As the BBj version of the BBx language has evolved, it has incorporated the Java Locale as STBL("!LOCALE"). A Locale is a one-, two- or three-part code used to identify a language, language+country, or language+country+variant. Language codes are lowercase two-letter ISO 639-1 codes. Country codes are uppercase two-letter ISO 3166-1 codes. Variant codes are added when >>

additional subdivisions beyond language+country are needed. When STBL("!LOCALE") is set, STBL("!DATE"), MSGBOX() button text, the print preview user interface, and the internationalized currency masking values are all updated to correspond with the specified locale (run the demo or try the sample code referenced at the end of this article). BBj supports these seven language codes:

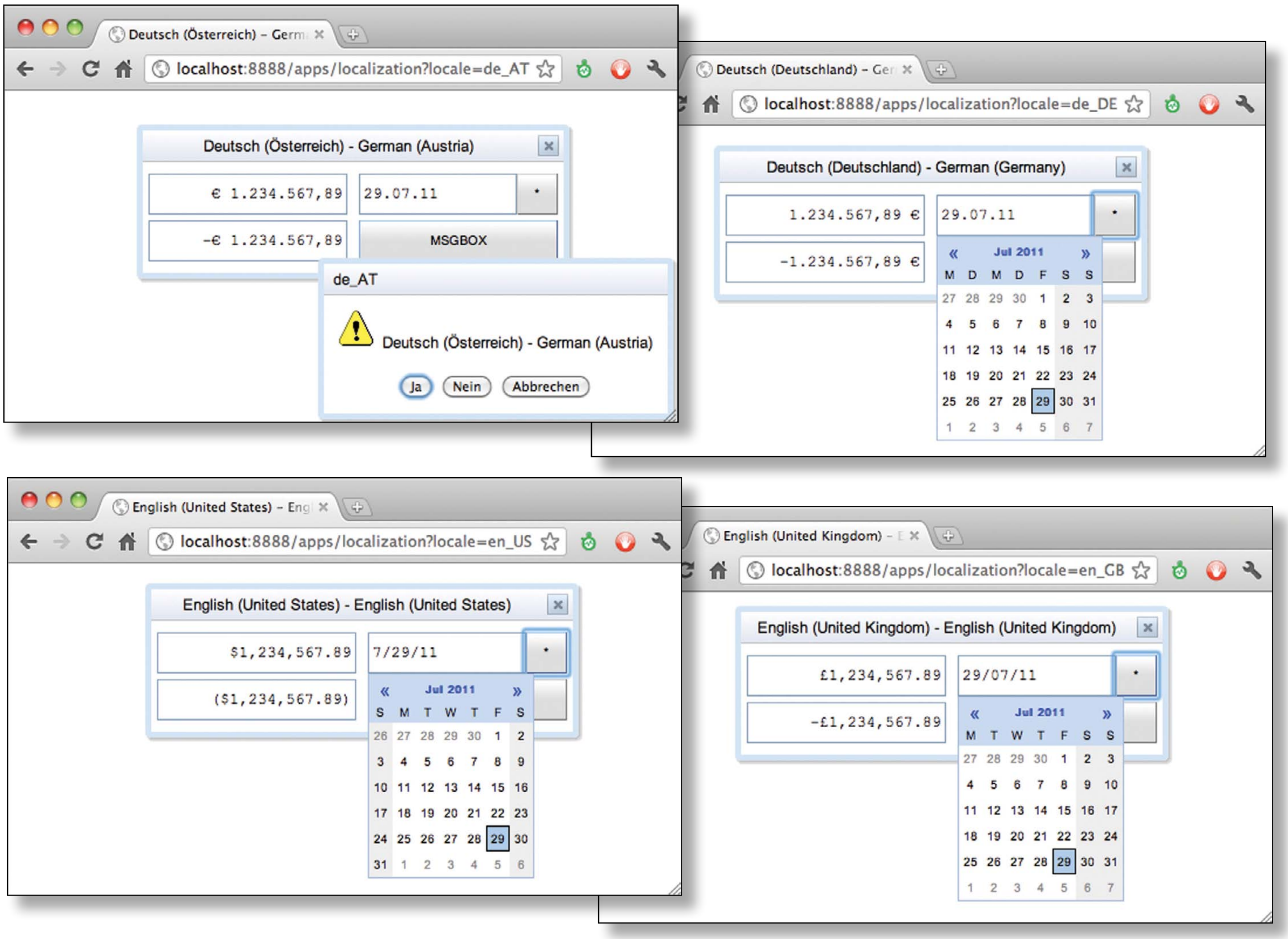
Code	Language	Locales
de	Deutsch (German)	de_AT (Austria), de_CH (Switzerland), de_DE (Germany), de_LU (Luxembourg)
en	English	en_AU (Australia), en_CA (Canada), en_GB (Great Britain), en_IE (Ireland), en_IN (India), en_MT (Malta), en_NZ (New Zealand), en_PH (Philippines), en_SG (Singapore), en_US (United States), en_ZA (South Africa)
es	Español (Spanish)	es_AR (Argentina), es_BO (Bolivia), es_CL (Chile), es_CO (Colombia), es_CR (Costa Rica), es_DO (Dominican Republic), es_EC (Ecuador), es_ES (Spain), es_GT (Guatemala), es_HN (Honduras), es_MX (Mexico), es_NI (Nicaragua), es_PA (Panama), es_PE (Peru), es_PR (Puerto Rico), es_PY (Paraguay), es_SV (El Salvador), es_US (United States), es_UY (Uruguay), es_VE (Venezuela)
fr	Français (French)	fr_BE (Belgium), fr_CA (Canada), fr_CH (Switzerland), fr_FR (France), fr_LU (Luxembourg)
it	Italiano (Italian)	it_CH (Switzerland), it_IT (Italy)
nl	Nederlands (Dutch)	nl_BE (Belgium), nl_NL (Netherlands)
sv	Svenska (Swedish)	sv_SE (Sweden)

Figure 3. Supported languages and locales

BBj has always been a client/server system, with potentially different locales on the client and the server. In BBj 11.0 and above, the BBjThinClient::getClientLocale() function returns the client locale from the GUI or BUI client. The BUI client locale is always set to one of the BBj supported language codes listed in **Figure 3**. If the browser is configured for one of those values, it is used as the default, but the default can also be overridden by specifying a locale value in the URL (e.g.: `?locale=de_DE`). The client locale determines the language that will be used for assorted end-user messages, including the optional User Authentication login screen and text on the various chooser controls.

Example

The Localization.txt sample program, located at links.basis.com/11samples, demonstrates a way to implement user input that automatically adjusts to the client locale. It sets the BBj server-side STBL("!LOCALE") to match the client locale, and it generates BBj INPUTN numeric masks based on the standard currency format for that locale. **Figure 4** shows the program running with different locales, such as en_GB displaying United Kingdom.


Deutsch (Österreich) – Germ
localhost:8888/apps/localization?locale=de_AT

Deutsch (Österreich) - German (Austria)

€ 1.234.567,89 | 29.07.11

-€ 1.234.567,89 | MSGBOX

de_AT

Deutsch (Österreich) - German (Austria)

Ja
Nein
Abbrechen
Deutsch (Deutschland) – Ger
localhost:8888/apps/localization?locale=de_DE

Deutsch (Deutschland) - German (Germany)

1.234.567,89 € | 29.07.11

-1.234.567,89 €

Jul 2011

M	D	M	D	F	S	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

English (United States) – Eng
localhost:8888/apps/localization?locale=en_US

English (United States) - English (United States)

$1,234,567.89 | 7/29/11

($1,234,567.89)

Jul 2011

S	M	T	W	T	F	S
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

English (United Kingdom) – E
localhost:8888/apps/localization?locale=en_GB

English (United Kingdom) - English (United Kingdom)

£1,234,567.89 | 29/07/11

-£1,234,567.89

Jul 2011

M	T	W	T	F	S	S
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Figure 4. Localization demos in German (top) using Austrian and German locales and English (bottom) using Great Britain and United States locales

The Past Through Tomorrow

Our company name, BASIS International Ltd., has always been a recognition that software is global. With the continuing rapid growth of the Internet, particularly the mobile Internet, this is more true now than ever before. BASIS' BUI technology offers a seamless path to get your application up and running in web browsers all over the world. BBj and BUI make localization painless, with functionality and tools that simplify the process. Barista, BASIS' data dictionary-driven development framework and runtime engine, leads by example and runs in multiple languages. Get started on localizing your app today to expand your potential customer base all over the world!

- Download and launch the msgbox.txt sample program that runs the msgboxdemo
- Read Can Your App Speak to Your Customer? for other BASIS language translation utilities