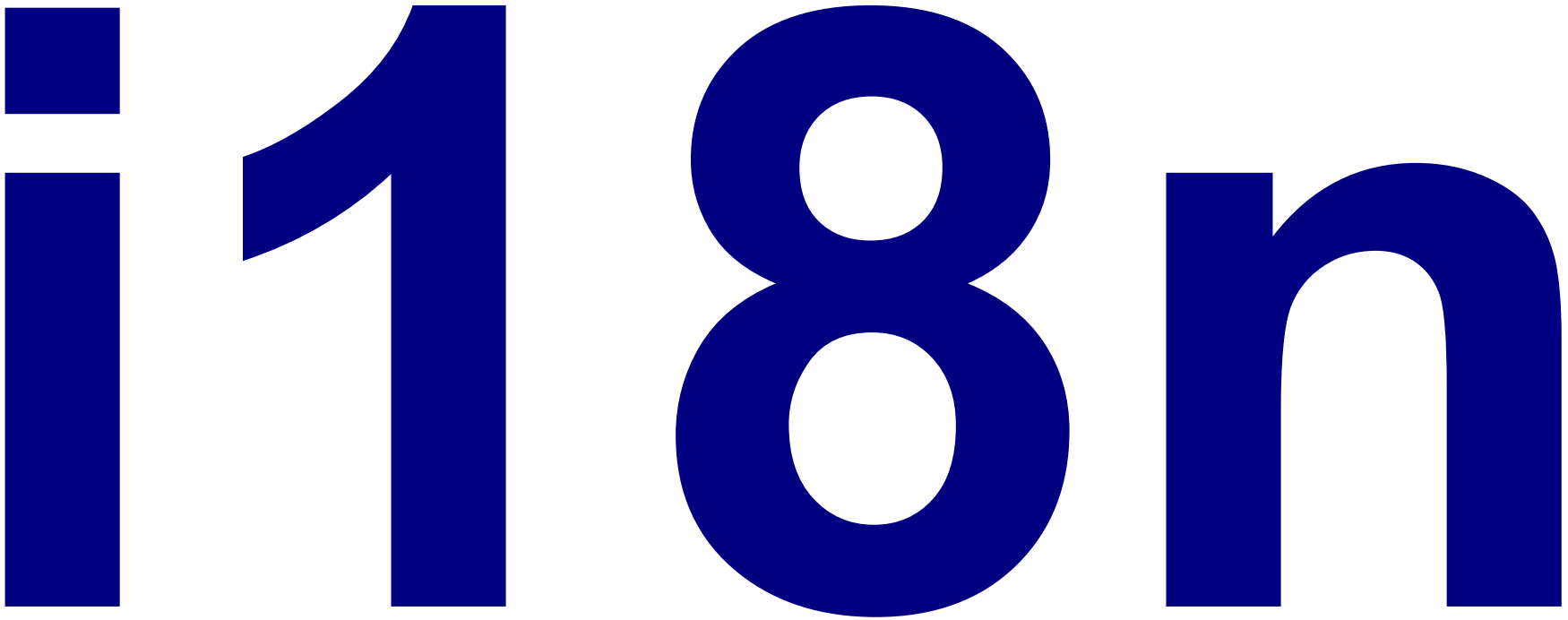

i18n

L10n

Bedeutung von i18n, L10n

- i18n == Internationalization

- L10n == Localization

Definitionen i18n, L10n

- i18n == Internationalization is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes.
- L10n == Localization

Definitionen i18n, L10n

- i18n == Internationalization is the process of designing a software application so that it can be adapted to various languages and regions without engineering changes.
- L10n == Localization is the process of adapting internationalized software for a specific region or language by adding locale-specific components and translating text.

http://en.wikipedia.org/wiki/Internationalization_and_localization

Ablauf der Übersetzung

1)
2)
3)
4)

Request trifft bei unserer Applikation ein, keine Response im Cache

Stell' Dir vor unser System wäre bereits vollständig internationalisiert und lokalisiert.

Ablauf der Übersetzung

1) Frage an Dich: Wie könnte der Ablauf aussehen?
2)
3)
4)

Ablauf der Übersetzung

1) Locale des aktuellen Request wird ermittelt
2)
3)
4)

Ablauf der Übersetzung

1) Locale des aktuellen Request wird ermittelt
2) MessageCatalogue für (1) wird geladen
3)
4)

Ablauf der Übersetzung

1) Locale des aktuellen Request wird ermittelt
2) MessageCatalogue für (1) wird geladen
3) MessageCatalogue für fallback Locale wird geladen und verwendet um alle Nachrichten zu ergänzen die in 2 nicht vorhanden sind, so entsteht im RAM ein „Dictionary“
4)

Ablauf der Übersetzung

1) Locale des aktuellen Request wird ermittelt
2) MessageCatalogue für (1) wird geladen
3) MessageCatalogue für fallback Locale wird geladen und verwendet um alle Nachrichten zu ergänzen die in 2 nicht vorhanden sind, so entsteht im RAM ein „Dictionary“
4) Jede zu übersetzende Nachricht wird im Dictionary gesucht. Falls nicht vorhanden, wird die Nachricht unverändert zurück gegeben.

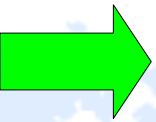
Erforderliche Vorbereitung

- Frage an Dich: Was ist alles zu tun für i18n + L10n
-
-
-

Erforderliche Vorbereitung

- Ausgabe Strings ("messages") abstrahieren
- Translator Komponente konfigurieren
- Locale des aktuellen Request ermitteln
- „Translation resource“ (z.B. Datei) für jede unterstützte Locale bereit stellen

Erforderliche Vorbereitung



- Ausgabe Strings ("messages") abstrahieren
- Translator Komponente konfigurieren
- Locale des aktuellen Request ermitteln
- Translation resources für alle Locales bereit stellen

Ausgabe Strings abstrahieren

```
$message =  'Hello World';
echo $message;
echo $translator->trans($message);
```

```
{{ message }}
{{ message|trans }}
```

```
{% trans %} Hello World {% endtrans %}
```

Erforderliche Vorbereitung

- Ausgabe Strings ("messages") abstrahieren
- Translator Komponente konfigurieren
- Locale des aktuellen Request ermitteln
- Translation resources für alle Locales bereit stellen

Translator konfigurieren

```
# app/config/config.yml
framework:
    translator:
        enabled: true
```

Translator konfigurieren

```
# app/config/config.yml
framework:
    translator:
        fallback: en
```

Translator konfigurieren

```
# app/config/config.yml
framework:
    default_locale: en
    translator:
        fallback: en
```

Translator konfigurieren

```
# app/config/config.yml
framework:
    default_locale: en
    translator:
        fallback: en
```

Erforderliche Vorbereitung

- Ausgabe Strings ("messages") abstrahieren
- Translator Komponente konfigurieren
- Locale des aktuellen Request ermitteln
- Translation resources für alle Locales bereit stellen

Locale des aktuellen Request

```
/**
 * @Route("/{_locale}/contact/{id}")
 */
public function showAction($id)
{
    // ...
}
```

Locale des aktuellen Request

```
/**
 * @Route("/{_locale}/contact/{id}",
        requirements={“_locale” = "en|fr|de"},
        defaults={“_locale”: “en”})
 */
public function showAction($id)
{
    // ...
```

Locale des aktuellen Request

```
/**
 * @Route("/{_locale}/contact/{id}")
 */
public function showAction($id)
{
    // ...
}
```

Locale des aktuellen Request

```
/**
 * @Route("/{_locale}/contact/{id}")
 */
public function showAction($id, Request $request)
{
    $locale = $request->getLocale();
    // ...
    $request->setLocale('en_US');
```

Sticky Locale mittels Session

- Technisch möglich, aber schlechte Idee, warum?
- Bessere Idee: Redirect auf z.B. /de/...

Sticky Locale mittels Session

- Technisch möglich, aber schlechte Idee, denn:

  Ein Uniform Resource Locator identifiziert und lokalisiert eine Ressource
- Bessere Idee: Redirect auf z.B. /de/...

Sticky Locale mittels Session

- Technisch möglich, aber schlechte Idee, denn:

  Ein Uniform Resource Locator identifiziert und lokalisiert eine Ressource
- Bessere Idee: Redirect auf z.B. /de/...
- Wenn Sticky Locale dennoch gewünscht:
  - Setzen
    - $this->get('session')->set('_locale', 'en_US');
  - Auswerten

http://symfony.com/doc/current/cookbook/session/locale_sticky_session.html

LuneticsLocaleBundle

- Erlaubt Locale Ermittlung basierend auf
  - Query parameter
  - Route parameters
  - Browser preferences
  - Cookie or the Session
  - Subdomain hostname
  - Custom guesser
- https://packagist.org/packages/lunetics/locale-bundle
- https://github.com/lunetics/LocaleBundle/blob/master/Resources/doc/index.markdown
- https://github.com/lunetics/LocaleBundle/blob/master/Resources/doc/guesser.md

JMSI18nRoutingBundle

- Sehr nützlich um bestehende Apps nachträglich zu internationalisieren
- Unterstützt z.B. das Schema
  - / (read Accept-Language Header)
  - /de/ (de_DE)
  - /en/ (en_US)
- http://jmsyst.com/bundles/JMSI18nRoutingBundle
- https://packagist.org/packages/jms/i18n-routing-bundle

JMSI18nRoutingBundle

JMS == Johannes Schmitt,
Symfony Core Contributor
=> Gute Qualität

- Sehr le Apps nachträglich zu inter
- Unterstützt z.B. das Schema
  - / (read Accept-Language Header)
  - /de/ (de_DE)
  - /en/ (en_US)
- http://jmsyst.com/bundles/JMSI18nRoutingBundle
- https://packagist.org/packages/jms/i18n-routing-bundle

Erforderliche Vorbereitung

- Ausgabe Strings ("messages") abstrahieren
- Translator Komponente konfigurieren
- Locale des aktuellen Request ermitteln
- Translation resources für alle Locales bereit stellen

Translation resources für alle Locales bereit stellen

- Datei
- dynamische Quelle (DB, API etc.)
  - Für gute Performance
    - Alle Übersetzungen auf einmal laden
    - Caching als Datei oder Blob in KeyValue Store

Translation resources für alle Locales bereit stellen

- Datei
- dynamische Quelle (DB, API etc.)
  - Für gute Performance
    - Alle Übersetzungen auf einmal laden
    - Caching als Datei oder Blob in KeyValue Store

Übersetzungsdateien: Ordner Rangfolge

- <bunde directory>/Resources/translations/
- app/Resources/<bundle name>/translations
- app/Resources/translations
- Frage an Dich: Warum drei mögliche Locations!?

Übersetzungsdateien: Ordner Rangfolge

- <bunde directory>/Resources/translations/
- app/Resources/<bundle name>/translations
- app/Resources/translations
- Messages aus (3rd Party) Bundles können auf Ebende der App-Config überschrieben werden

Übersetzungsdateien: Ordner Rangfolge

- <bunde directory>/Resources/translations/
- app/Resources/<bundle name>/translations
- app/Resources/translations
- Messages aus (3rd Party) Bundles können auf Ebende der App-Config überschrieben werden
- Einzelne Messages werden sukzessive ersetzt, d.h. nur veränderte erforderlich in app/...

Übersetzungsdateien: Ordner Rangfolge

- <bunde directory>/Resources/translations/
- app/Resources/<bundle name>/translations
- app/Resources/translations
- Messages aus (3rd Party) Bundles können auf Ebende der App-Config überschrieben werden
- Einzelne Messages werden sukzessive ersetzt, d.h. nur veränderte erforderlich in app/...
- Fallback vom Locale (de_AT) zur Sprache (de) und erst dann zum Fallback Locale (en_US)

Übersetzungsdateien: Namen

- domain.locale.loader
  - Domain: Optional, z.B. admin, navigation
    - Die Default Domain ist „messages“
  - Locale: z.B. en_GB, en_US, en, de_AT
  - Loader: xliff, php or yml

Übersetzungsdateien: Namen

- domain.locale.loader
  - Domain: Optional, z.B. admin, navigation
    - Die Default Domain ist „messages“
  - Locale: z.B. en_GB, en_US, en, de_AT
  - Loader: xliff, php or yml
- Messages aus nicht default Domain verwenden:
  $translator->trans('Hallo Foo', [], 'admin');

Übersetzungsdateien: Formate

- yml
- php
- xliff
- beliebige weitere (eigenen Loader schreiben)

Übersetzungsdateien: Formate

- yml

```
Symfony2 is great: J'aime Symfony2
```

- php
- xliff
- beliebige weitere (eigenen Loader schreiben)

Übersetzungsdateien: Formate

- yml

```
Symfony2 is great: J'aime Symfony2
```

- php

```
return array(
    'Symfony2 is great' => 'J\'aime Symfony2',
);
```

- xliff
- beliebige weitere (eigenen Loader schreiben)

Übersetzungsdateien: Formate

- yml

```
Symfony2 is great: J'aime Symfony2
```

- php

```
return array(
    'Symfony2 is great' => 'J\'aime Symfony2',
);
```

- xliff

```
<?xml version="1.0"?> ...
```

```
<!-- messages.fr.xliff -->
<?xml version="1.0"?>
<xliff version="1.2"
xmlns="urn:oasis:names:tc:xliff:document:1.2">
<file source-language="en" datatype="plaintext"
original="file.ext">
<body>
    <trans-unit id="1">
        <source>Symfony2 is great</source>
        <target>J'aime Symfony2</target>
    </trans-unit>
</body>
</file>
</xliff>
```

Übersetzungsdateien: Formate

- yml

```
Symfony2 is great: J'aime Symfony2
```

- php

```
return array(
    'Symfony2 is great' => 'J\'aime Symfony2',
);
```

- xliff

```
<?xml version="1.0"?> ...
```

Messages: Real or Keyword

```
$translator->trans('Symfony is great');
$translator->trans('symfony.great');

$translator->trans('Administration Area');
$translator->trans('admin.title');
```

Keyword Messages Kurzform in yml und php, nicht xml Dateien

```
symfony2:
    is:
        great: Symfony2 is great
        amazing: Symfony2 is amazing
    has:
        bundles: Symfony2 has bundles
```

konvertiert intern zu:

```
symfony2.is.great: Symfony2 is great
symfony2.is.amazing: Symfony2 is amazing
symfony2.has.bundles: Symfony2 has bundles
```

Keyword Messages Kurzform in yml und php, nicht xml Dateien

```
symfony2:
    is:
        great: Symfony2 is great
        amazing: Symfony2 is amazing
    has:
        bundles: Symfony2 has bundles
```

konvertiert intern zu:

```
symfony2.is.great: Symfony2 is great
symfony2.is.amazing: Symfony2 is amazing
symfony2.has.bundles: Symfony2 has bundles
```

Vorteil: Leichtes Suchen mit Editor

Platzhalter in Messages

```
$translated = $translator->trans(
    'Hallo %name%, wie geht\'s Dir heute?',
    array('%name%' => $name)
);
```

Pluralisierung in Messages

```
$numberOfApples = 10;
$translated = $translator->transChoice(
    'There is one apple|There are %count% apples',
    $numberOfApples,
    array('%count%' => $numberOfApples)
);
```

Pluralisierung in Messages

```
$numberOfApples = 10;
$translated = $translator->transChoice(
    'There is one apple|There are %count% apples',
    $numbe  fApples,
    arra    %count%' => $number    pples)
);
```

Erste Variante: 1 Objekt

Zweite Variante:
Mehrere Objekte

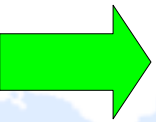
Pluralisierung in Messages

```
{% transchoice count %}
    {0} There are no apples|
    {1} There is one apple|
    ]1,Inf] There are %count% apples
{% endtranschoice %}
```

Erforderliche Vorbereitung

- Ausgabe Strings ("messages") abstrahieren
- Translator Komponente konfigurieren
- Locale des aktuellen Request ermitteln
- Translation resources für alle Locales bereit stellen

Weitere Möglichkeiten



- Custom Loader zum Laden aus DB, API etc.
- Datenbank Inhalte übersetzten
- Constraint Messages Übersetzten
- Locale manuell festlegen

Custom Loader zum Laden aus DB, API etc.

```
interface LoaderInterface
{
    /**
     * @param mixed  $resource A resource
     * @param string $locale   A locale
     * @param string $domain   The domain
     *
     * @return MessageCatalogue
     * @throws NotFoundResourceException
     * @throws InvalidResourceException
     */
    public function load($resource, $locale, $domain = 'messages');
}
```

Custom Loader zum Laden aus DB, API etc.

```
class MyCustomLoader implements LoaderInterface
{
    public function load($resource, $locale, $domain = 'messages')
    {
        // Übersetzungen Laden aus DB, API etc.

        $catalogue = new MessageCatalogue($locale);
        $catalogue->set('hello.world', 'Hello World!', $domain);
        return $catalogue;
    }
}
```

Custom Loader zum Laden aus DB, API etc.

```
services:
    main.translation.my_custom_loader:
        class: Acme\MainBundle\Translation\MyCustomLoader
        tags:
            - { name: translation.loader, alias: db }
```

Custom Loader zum Laden aus DB, API etc.

```
services:
    main.translation.my_custom_loader:
        class: Acme\MainBundle\Translation\MyCustomLoader
        tags:
            - { name: translation.loader, alias: db }
```

- app/Resources/translations/messages.de.db (etc.) activiert den Custom Loader.

Custom Loader zum Laden aus DB, API etc.

```
services:
    main.translation.my_custom_loader:
        class: Acme\MainBundle\Translation\MyCustomLoader
        tags:
            - { name: translation.loader, alias: db }
```

- app/Resources/translations/messages.de.db (etc.) activiert den Custom Loader.
- Die Datei selbst wird als Resource übergeben und kann wahlweise z.B. Parameter für einen API Aufruf o.ä. enthalten

Custom Loader zum Laden aus DB, API etc.

```
services:
    main.translation.my_custom_loader:
        class: Acme\MainBundle\Translation\MyCustomLoader
        tags:
            - { name: translation.loader, alias: db }
```

- app/Resources/translations/messages.de.db (etc.) activiert den Custom Loader.
- Die Datei selbst wird als Resource übergeben und kann wahlweise z.B. Parameter für einen API Aufruf o.ä. enthalten

http://symfony.com/doc/current/reference/dic_tags.html#dic-tags-translation-loader

Weitere Möglichkeiten

- Custom Loader zum Laden aus DB, API etc.
- Datenbank Inhalte übersetzten
- Constraint Messages Übersetzten
- Locale manuell festlegen

Datenbank Inhalte übersetzten

```
class Category
{
    /**
     * @Gedmo\Translatable
     * @ORM\Column(length=64)
     */
    private $title;
```

Datenbank Inhalte übersetzten

```
/**
 * @Gedmo\TranslationEntity(class="Entity\CategoryTranslation")
 */
class Category
{
    /**
     * @Gedmo\Translatable
     * @ORM\Column(length=64)
     */
    private $title;
```

https://github.com/l3pp4rd/DoctrineExtensions/blob/master/example/app/Entity/CategoryTranslation.php

Weitere Möglichkeiten

- Custom Loader zum Laden aus DB, API etc.
- Datenbank Inhalte übersetzten
- Constraint Messages Übersetzten
- Locale manuell festlegen

Constraint Messages Übersetzten

```
// src/Acme/BlogBundle/Entity/Author.php
class Author
{
    /**
     * @Assert\NotBlank(message = "author.name.not_blank")
     */
    public $name;
    ...
}

# validators.en.yml
author.name.not_blank: Please enter an author name.
```

Weitere Möglichkeiten

- Custom Loader zum Laden aus DB, API etc.
- Datenbank Inhalte übersetzten
- Constraint Messages Übersetzten
- Locale manuell festlegen

```
$this->get('translator')->trans(
    'Symfony2 is great',
    array(),
    'messages',
    'fr_FR'
);

$this->get('translator')->transChoice(
    '{0} There are no apples|{1} There is one apple|]1,Inf[ There are
%count% apples',
    10,
    array('%count%' => 10),
    'messages',
    'fr_FR'
);
```

Zusammenfassung

- Ausgabe Strings ("messages") abstrahieren
- Translator Komponente konfigurieren
- Locale des aktuellen Request ermitteln
- Translation resources für alle Locales bereit stellen
- Custom Loader zum Laden aus DB, API etc.
- Datenbank Inhalte übersetzten
- Constraint Messages Übersetzten
- Locale manuell festlegen

Weiterführende Information

- http://symfony.com/doc/current/book/translation.html
- http://symfony.com/doc/current/components/translation/
- More loaders: Click here for list of 13 built in translation loaders
- Bundle for editing translations (excellent featureset, active project)
  - https://github.com/Kunstmaan/KunstmaanTranslatorBundle

Vielen Dank
für Eure Aufmerksamkeit!

Fragen?

Ideen, Wünsche, Anmerkungen?