КООРДИНАЦИОННЫЙ ЦЕНТР ДОМЕНОВ .RU/.РФ

Поддерживаю .РФ

Внедрение поддержки интернационализированных доменных имен и адресов электронной почты для Python-разработчиков

Антон Воршевский

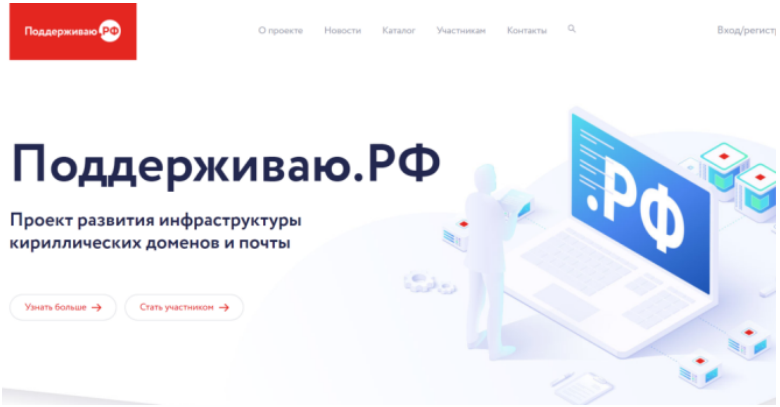
Архитектор информационных систем и популяризатор программирования

10 Марта 2021

Подготовлено на основе материалов Universal Acceptance Steering Group, https://uasg.tech/

ТЕХНИЧЕСКАЯ ПОДДЕРЖКА

КООРДИНАЦИОННЫЙ ЦЕНТР ДОМЕНОВ .RU/.РФ



ПОДДЕРЖИВАЮ.РФ

- **Документация по IDN/EAI/UA**
- **Обучающий курс по UA (+проверочный тест)**
- **Перечень ПО со статусом поддержки IDN/EAI**
- **Багтрекер по доработкам, связанным с поддержкой IDN/EAI**
- **Онлайн-тест на поддержку EAI**
- **Обобщенная методика проверки ПО на соответствие UA**
- **Рекомендации по внедрению поддержки доменных имен и почтовых адресов в .РФ**

ТЕСТОВЫЙ СТЕНД

КООРДИНАЦИОННЫЙ ЦЕНТР ДОМЕНОВ .RU/.РФ

REG.RU

Поддерживаю.РФ

Тестовый стенд на базе почтового ПО с открытым исходным кодом и полной поддержкой работы с кириллической электронной почтой. Почтовые адреса выделяются по запросу, только на время тестирования.

Получите тестовые адреса электронной почты на кириллице:

поддерживаю.рф/контакты

Целевая аудитория, Задачи и Цели

- Целевая аудитория:
  - Разработчики на Python (и иных языках)
  - Менеджеры IT проектов
  - Технические директора
- Задачи:
  - Понять основные концепции относящиеся к интернационализированным доменным именам и E-mail адресам
  - Понять проблемы при использовании чистого кода на Python для проверки и использования интернационализированных доменных имен и E-mail
  - Рассмотреть подходящие для этой цели библиотеки
  - Узнать на примерах как использовать библиотеки
  - Рассмотреть существующие практики разработки приложений с поддержкой UA
- Цель:
  - Разрабатывать Python приложения с поддержкой UA

План

- Суть проблемы
- Базовые ключевые концепции относящиеся к UA
  - Unicode
  - IDN (Интернационализированные Доменные Имена)
  - EAI (Интернационализация E-mail Адресов)
- Валидация ввода UA идентификаторов
- Использование UA идентификаторов:
  - Разименовывание доменных имен
  - Отправка E-mail
- Рекомендуемые практики
- Заключение
- Литература

Суть Проблемы

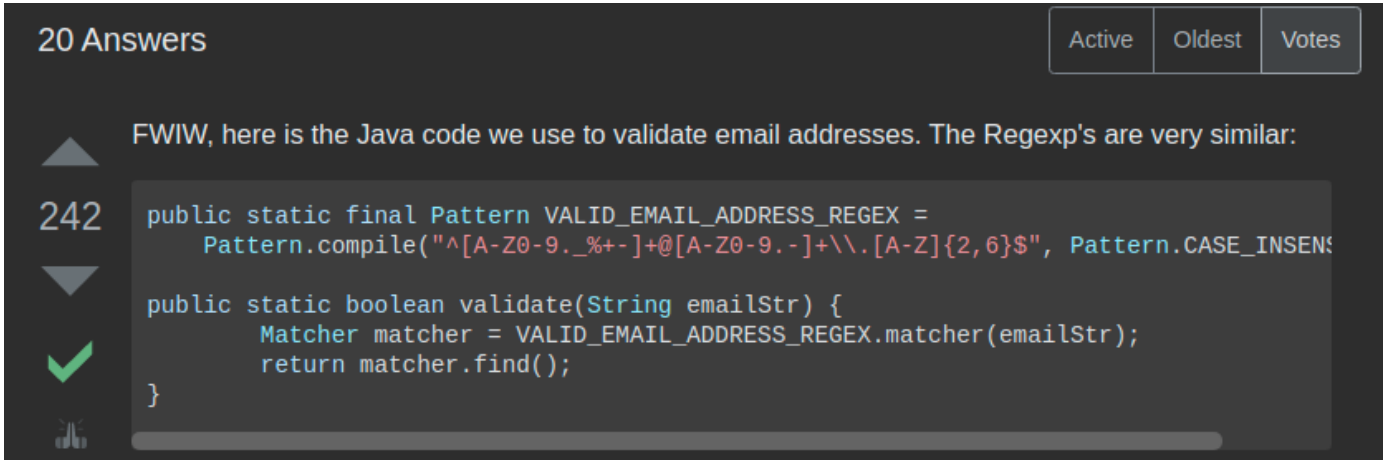
Валидация E-mail: Реальный пример

- Компания сделала вебсайт, где клиенты могут подписаться на получение выгодных предложений по электронной почте
- Поскольку форма подписки является вводом пользователя, разработчики проверяют корректность E-mail адреса перед попыткой отправить E-mail:

```
if is_email_valid( email ):
    subscribe()
else:
    raise Exception('Invalid email address, please review it and submit again')
```

Валидация E-mail

- Разработчики пошли на Stackoverflow и нашли регулярное выражение для валидации:



Валидация E-mail

- Компания стала международной и начала работать не только в англоязычных регионах
- Продажники решили посмотреть, какие проблемы возникли у клиентов, которые не смогли подписаться. Вебсайт, по их словам, всегда возвращает: "Invalid email address, ..." для реально существующего E-mail.
- Разработчики покопались в логах и нашли E-mail воспроизводящий проблему:

普遍接受 - 测试 @ 普遍接受 - 测试 . 世界

- Возможно простое регулярное выражение не настолько хорошая идея и разработчики начали искать правильную библиотеку для валидации E-mail

Валидация E-mail

- Команда разработки осознала что библиотека email-validator уже имеет функцию "validate_email". Они переписали метод is_email_valid:

```
from email_validator import validate_email, EmailNotValidError

def is_email_valid( email ):
    ""
    try:
        valid = validate_email(email)
        return True
    except EmailNotValidError as e:
        return False
```

Валидация E-mail

- Наконец, изучая эти исправления в используемой библиотеки для отправки почты smtplib, они поняли что нужно обновить библиотеку до версии 3.5, так же изменить функцию subscribe, добавив опцию SMTPUTF8, а их SMTP сервер так же должен поддерживать новый флаг "SMTPUTF8". Баг исправлен?

Валидация E-mail

- Позже, был проведен аудит безопасности веб приложения. Внешние аудиторы безопасности выставили плохую оценку за валидацию E-mail и предложили рекомендуемое стандартное исправление, взятое у признанной международной организации в области безопасности: Open Web Application Security Project (OWASP). OWASP рекомендует следующее регулярное выражение для E-mail:
  - ^[a-zA-Z0-9_+&*-]+(?:\.[a-zA-Z0-9_+&*-]+)*@(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,7}$]]
- Должна ли компания реализовать рекомендованное исправление безопасности?

Ключевые Концепции

Unicode

- присваивает коды (codepoint) для символов (glyphs)
- В спецификациях коды представлены как шестнадцатеричные значения в формате U+XXXX
- обычно юникод передается в формате UTF-8
  - переменное количество байтов для одного кода
  - ascii используется как есть
  - золотой стандарт для передачи Unicode в веб и различных протоколах
- несколько способов кодирования символа:
  - “è” = U+00E8
  - “e`” = “è” = U+0065 U+0300
  - Что бы убедиться, что итоговое представление одинаково, независимо от способа ввода символов, нужен процесс Нормализации.
    - в двух примерах выше, Normalization Form C(NFC) даст U+00E8 для обоих

Universal Acceptance (UA)

- Как правильно поддерживать интернационализированные идентификаторы и длинные TLD (домен верхнего уровня)
  - интернационализированные идентификаторы:
    - IDN (Интернационализированные Доменные Имена)
    - EAI (Интернационализация E-mail Адресов)
  - Длинные имена доменов верхнего уровня (TLD):
    - Недавно TLD были длиной 2-3 символа (например .ru, .com). Затем TLD стали длиннее (например .info, .google).
    - Некоторые приложения продолжают проверять что TLD введенный пользователем не больше чем 3 символа
  - Добавленные и удаленные TLD:
    - TLD могут появляться и исчезать ежедневно. Некоторые приложения проверяют корректность TLD по устаревшему списку доменов.

Доменные Имена

- Доменное имя это упорядоченная последовательность меток (labels): a.b.c.d
- Домен верхнего уровня справа
- Domain Name System (DNS) это распределенная база данных и сервис для запроса записей привязанных к доменному имени
- Доменное имя может иметь много таких записей:
  - IPv4 адреса для доменного имени
  - IPv6 адреса для доменного имени
  - Имя хоста почтового сервера отвечающего за доменное имя
  - ...
- DNS зона - это список доменных записей (Resource Records(RR)) для метки внутри домена уровнем выше

Интернационализированные Доменные Имена (IDN)

- Позволяют использовать не-ASCII символы для любой метки (label) в доменном имени
  - Не все метки в доменном имени могут быть интернационализированы

- пример: exâmple.ca

- Пользователь использует IDN версию, но IDN преобразуется в ASCII
  - exâmple => exmple-xta => xn--exmple-xta
  - xn-- префикс добавляется для обозначения кодированного IDN

Интернационализированные Доменные Имена (IDN) (продолжение)

- Пример обработки с использованием IDN:
  - пользователь вводит в браузере: http://exâmple.ca
  - браузер делает нормализацию ввода пользователя
  - браузер преобразует exâmple.ca в ASCII совместимое представление называемое Punycode[RFC3492] и добавляет в начало ‘xn--’.
    - xn--exmple-xta.ca
  - браузер выполняет DNS запрос что бы получить IP адрес для xn--exmple-xta.ca

Интернационализированные Доменные Имена (IDN) (продолжение)

- Стандарт назван IDN for Applications (IDNA)
  - две версии: IDNA2003 и IDNA2008. Сейчас используется последняя.
- U-Label это нативное Unicode представление IDN метки: viagénie
- A-Label это Punycode представление IDN метки: xn--viagnie-eya

Два IDN Стандарта

- Первая версия: называется IDNA2003 (RFC3490)
  - Алгоритмы называются StringPrep(RFC3454) и NamePrep(RFC3491).
  - Кодирование в ASCII использует Punycode (RFC3492).
  - Идентифицирует IDN добавлением xn-- перед punycode
  - Базируется на Unicode 3.2 (март 2002)
  - Пропускает новые символы (т.е. добавленые после Unicode 3.2) как есть

Два IDN Стандарта

- Вторая (и позднейшая) версия: называется IDNA2008. Больше не использует Stringprep и Nameprep, однако кодирование в ASCII продолжает использовать Punycode и идентичный префикс (xn--). IDNA2008 гораздо более гибок по поддержке новых символов добавляемых в Unicode со временем.
- IDNA2008 более строг чем IDNA2003: корректные домены по IDNA2003 могут не быть таковыми в соответствии с IDNA2008.
- Таким образом настоятельно рекомендуется использовать стандарт IDNA2008.
- Рекомендация: убедиться что библиотеки которые вы используете, базируются на IDNA2008.

Два IDN Стандарта

- Пример: ſſ.example (i.e. U+017F U+017F)
  - корректен для IDNA2003 и преобразуется в ss.example
  - недопустим в соответствии с IDNA2008
- Для “облегчения” перехода с IDNA2003 на IDNA2008, Unicode определил процедуру перехода в спецификации UTS 46. Согласно UTS 46, ſſ.example преобразуется в ss.example. IETF не рекомендует использовать UTS 46. ICANN поддерживает только IDNA2008, таким образом IDNA2003 или переходный домен UTS46 не валидны.

Публичный Список Суфиксов (Public Suffix List) (PSL)

- Попытка помочь разработчикам узнать существует или нет конкретный TLD (или любые поддомены). Ведется добровольцами (Mozilla). TLD должен (вручную) зарегистрировать себя и используемые в домене правила у PSL мейнтейнеров. Модель управления примитивная.
- Задача была позволить браузерам проверять корректность доменных имен и TLD прямо в адресной строке и даже предлагать/корректировать TLD согласно статическому списку, вместо отправки DNS запросов.
- https://publicsuffix.org
- В случае использования,
  - Разработчик должен поддерживать локальную копию списка в актуальном состоянии.
  - Любой TLD не в списке будет считаться не существующим.
  - TLD может не быть в PSL потому что регистратор не зарегистрировал его, или по причине какой-то политики PSL, или потому что список суффиксов в приложении устарел.

Universal Acceptance (Всеобщее Принятие) (UA)

- Как правильно поддерживать интернационализированные идентификаторы и длинные TLD

  - интернационализированные идентификаторы:
    - IDN
    - EAI

Universal Acceptance (UA) (продолжение)

- Длинные доменные имена верхнего уровня (TLD):
  - Недавно TLD были длиной 2-3 символа (например .ru, .com). Затем TLD стали длиннее (например .info, .google).
  - Некоторые приложения продолжают проверять что TLD введенный пользователем не больше чем 3 символа

Universal Acceptance (UA) (продолжение)

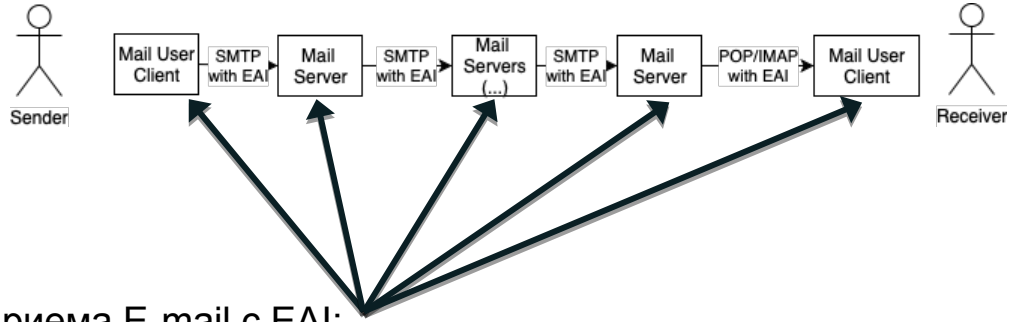
Добавленные и удаленные TLD:

- TLD могут появляться и исчезать ежедневно. Некоторые приложения проверяют корректность TLD по устаревшему списку доменов.

Интернационализированные E-mail Адреса (EAI)

- e-mail синтакс: leftside@domainname
- domainname может быть интернационализированным как IDN
- leftside (так же известно как локальная часть/имя ящика) с Unicode (UTF8) это **EAI**
- примеры: kévin@example.org, вася@фирма.рф, すし @ 快手 . 游戏
- Побочный эффект: Почтовые заголовки тоже должны поддерживать EAI. Почтовые заголовки используются почтовым софтом для получения информации как доставить E-mail.
- Поскольку не все почтовые сервера поддерживают EAI, используется протокол согласования, что бы посылать EAI только когда принимающей сервер его поддерживает. Если нет, письмо возвращается отправителю. Опция SMTPUTF8 используется для этой цели в протоколе SMTP (Simple Mail Transport Protocol)

EAI Путь Доставки

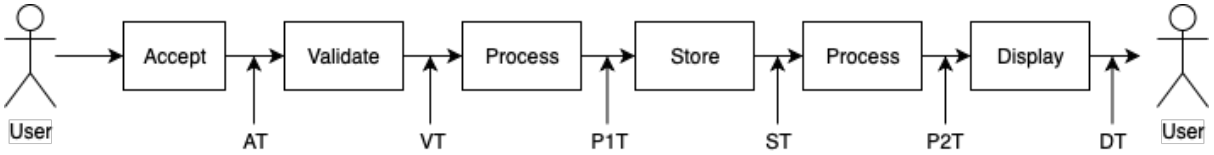


- для отправки и приема E-mail с EAI:
  - все участники по пути доставки E-mail должны обновиться для поддержки EAI
  - если любой SMTP сервер по дороге не поддерживает EAI, тогда E-mail не сможет быть доставлен.

Компоненты Приложения

Модель Компонентов Приложения

- Базируется на UASG026. Это упрощенная модель компонентов приложения сфокусированная на обработке интернационализированных идентификаторов.
- Каждый блок имеет свой набор требований и выполняемых действий.



Валидация Ввода Пользователя

- Валидация ввода пользователя или иного ввода. Очень полезна по следующим причинам: улучшенный user experience, безопасность, исключение нерелевантных проблем
- Валидация E-mail адресов и доменных имен полезна.
- Некоторые методы валидации:
  - синтаксис: синтаксис строки правильный? Например, E-mail адрес должен содержать ‘@’. доменное имя должно содержать ‘.’
  - Доменное имя корректно?
    - домен верхнего уровня (TLD) существует?
    - полное доменное имя существует?
  - E-mail адрес корректен?
    - часть с доменным именем (смотри выше)
    - локальная часть

Валидация Доменного Имени

- Синтаксис:
  - ASCII: RFC1035
  - IDN:
    - используя A-Labels
    - используя U-Labels
- Домен верхнего уровня (TLD) существует?
  - список TLD
  - DNS запрос
- Полное доменное имя существует?
  - DNS запросы

Разименовывание (Resolving) Доменного Имени

- После валидации программа использует идентификатор доменного имени для:
  - выполнение запросов к DNS используя доменное имя
- Таким образом для соответствия UA, приложение должно использовать правильные методы, которые поддерживают UA.
  - например, передача U-Label в традиционный вызов gethostbyname() может закончиться неудачей, поскольку он не ожидает доменное имя в UTF8.

Валидация E-mail Адресов

- E-mail адрес состоит из localpart@domainname
- Для доменных имен смотри раньше
- Синтаксис локальной части:
  - ASCII
  - UTF8 (EAI)
- Доменное имя принимает почту?
- Почтовый ящик localpart принимает почту?

Отправка E-mail

- После валидации программа использует E-mail для:
  - E-mail адрес может быть использован как ID пользователя
  - E-mail адрес может быть использован для отправки почты
- Таким образом для соответствия UA, приложение должно использовать правильные методы, которые поддерживают UA.
  - Например, передача локальной части E-mail адреса в UTF-8 почтовому серверу может не получиться, поскольку тот не ожидает локальную часть адреса в UTF8.

Test Cases

- Исчерпывающий список тестовых случаев для UA документирован в UASG0004
- Разработчику настоятельно рекомендуется использовать эти тестовые наборы для unit и системного тестирования.

Python

Версии Python

- Примеры кода тестировались на Python 3.6 где это возможно
- Возможно что старые версии могут иметь проблемы.
- Некоторые библиотеки могут потребовать (не обязательно из-за UA) более свежие версии Python
- Если это явно не сказано, примеры должны работать на любых платформах

Версии Библиотек

- Этот семинар демонстрирует разные библиотеки встречающихся в дикой природе. Хотя список не исчерпывающий, он достаточно полон, что бы помочь понять какую библиотеку и как вам лучше использовать.
  - Будущие отчеты UASG предоставят детальную информацию о степени соответствия UA конкретных библиотек. Смотри https://uasg.tech
- Библиотеки тестировались на их текущих версиях, доступных на момент написания.
- Возможно что новые версии тех же библиотек уже исправили проблемы и улучшили поддержку, что так же поменяет и наши рекомендации.
- Поэтому при начале разработки проверьте пожалуйста текущий статус библиотек.

Тип для Хранения UA идентификаторов

- UA идентификаторы это доменные имена и почтовые адреса которые могут содержать данные в UTF8.
- Начиная с Python 3.0 стандартный тип str нативно поддерживает Unicode, поэтому хорошо подходит для хранения этих идентификаторов. В Python 2.7 необходимо использовать встроенный тип unicode.
- Кодировка по умолчанию в большинстве систем обычно UTF-8. Проверь или измени дефолтную кодировку если это необходимо.

Насколько Базовых Тестовых Случаев

- Для дальнейших примеров кода мы будем использовать следующие наборы входных данных, которые являются базовыми тестовыми случаями для UA (не полными):

```
testDomains = [
    "example.org",               # ascii.ascii
    "example.undefinedtld",      # unexistant tld
    "example.recentTld",         # recently allocated tld
    "example.accountants",       # allocated longer than 7 char tld
    "exâmple.org",               # ulabel.ascii
    "xn--exmple-xta.org",        # alabel.ascii
    "exâmple.ꝡw",               # ulabel.ulabel
    "exâmple.xn--o3cw4h",        # ulabel.alabel
    "xn--exmple-xta.xn--o3cw4h" # alabel.alabel
    ]
testLocalParts = [
    "user",
    "kévin"
    ]
```

Валидация Доменного Имени

Используя Чистый Python

- Традиционный способ сделать резолв имени хоста
  - `socket.gethostbyname(hostname)`
  - `socket.gethostbyname_ex(hostname)`
  - `socket.getaddrinfo(host, port, family=0, type=0, proto=0, flags=0)`
- Кидает exception socket.herror либо socket.gaierror на любую ошибку:
  - у хоста нет IP адресов
  - некорректный хост
  - плохой синтаксис
  - ...
- Передает host как есть системному вызову OS без валидации .
  - таким образом, невалидные домены (с плохими Ulabels) проходят.
  - результат зависит от реализации внутри OS

Используя Чистый Python: Рекомендация

- Не используйте имя хоста в прямую как есть.
- Вместо этого:
  - Валидируйте имя хоста перед вызовом gethostbyname()
    - что бы избежать ожидания ответов на заведомо некорректные запросы
    - пользователь получит более адекватный фидбэк: можно различить случаи когда имя хоста было не верным и когда оно было корректным, но запрос не вернул данных.
  - Подготовьте hostname (например сконвертировав IDN в A-label) используя другую библиотеку, а уже затем используйте базовые вызовы
    - это сделает ваш код независимым от используемой OS
  - Или используйте другую библиотеку для подготовки и выполнения запроса

encodings.idna

- Нативный конвертер для U-label и A-label
- Основан на IDNA2003
- Не требует установки

encodings.idna: Пример

```
print( 'ботва.домен'.encode('idna') )

print( b'xn--80abd1cu.xn--p1ai'.decode('idna') )
```

encodings.idna: Рекомендация

- Не смотря на то что является стандартной и не требует установки основана на устаревшем стандарте IDNA2003, поэтому не используйте ее
- Не Рекомендуется

idna

- Очень хорошая и наиболее полная реализация стандарта IDNA2008
- https://github.com/kjd/idna
- Старые версии поддерживают Python 2.7, для этого используйте idna<3 в зависимостях
- Используйте исключение idna.IDNAError для обработки ошибок

idna: Пример

```
import idna

try:

    valid = idna.encode(domain)

except idna.IDNAError as e:

    print str(e)
```

idna: Рекомендация

- Наилучшее соответствие стандарту IDNA2008 из всех библиотек
- Настоятельно рекомендуется

Валидация E-mail Адресов

Регулярные Выражения для E-mail (Regex)

- Базовое: something@something
  - ^(.+)@(.+)$
- От owasp.org (безопасность):
  - [^[a-zA-Z0-9_+&*-]+(?:\.[a-zA-Z0-9_+&*-]+)*@(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,7}$]
  - Не поддерживает EAI (i.e. no UTF-8 local parts allowed: [a-zA-Z0-9_+&*-])
  - Не поддерживает ASCII TLD длинее 7 символов: [a-zA-Z]{2,7}
  - Не поддерживает U-Labels в IDN TLD: [a-zA-Z]
  - Но OWASP является _авторитетом_ в области безопасности.
    - Поэтому может возникнуть конфликт с вышей командой безопасности из-за использования совместимого с UA регулярного выражения вместо “стандартного” от OWASP.

Регулярные Выражения для E-mail (Regex) (продолжение)

- Примеры Regex предлагаемые на различных форумах: List of proposals
  - ^[A-Za-z0-9+_.-]+@(.+)$
  - ^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^.-]+@[a-zA-Z0-9.-]+$
  - ^[a-zA-Z0-9_!#$%&'*+/=?`{|}~^-]+(?:\\.[a-zA-Z0-9_!#$%&'*+/=?`{|}~^-]+)*@[a-zA-Z0-9-]+(?:\\.[a-zA-Z0-9-]+)*$
  - ^[\\w!#$%&'*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{|}~^-]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}$
  - Все не соответствуют EAI следующим по причинам:
    - На поддерживают UTF8 в локальной части
    - Имеют ограничение на длину TLD
    - Не поддерживают U-Label

Регулярные Выражения для E-mail (Regex) (продолжение)

- Возможно конечно изобрести regex для EAI-IDN, но только для IDN оно уже будет выглядеть как реализация таблиц IDNA внутри regex!
- Поэтому, учитывая что обе стороны EAI могут содержать UTF8, адекватным regex для EAI может быть .*@.* которое проверяет только наличие символа ‘@’.

email_validator

- Популярная библиотека для валидации и нормализации email адресов
- Очень хорошее соответствие стандартам
- import email_validator
- Не валидирует доменную часть адреса

email_validator: Пример

```
from email_validator import validate_email, EmailNotValidError

try:
    valid = validate_email(email, check_deliverability=False)
    valid_email = valid.email
except EmailNotValidError as e:
    print(str(e))
```

email_validator: Рекомендация

- Очень хорошая библиотека.
- Добавьте валидацию и подготовку IDN домена как дополнительный шаг
- `Выключите check_deliverability что бы не проверять существование домена`
- `Рекомендуется`

Отправка E-mail

smtplib

- Нативная библиотека для отправки email
- Готова к работе с EAI
- Поддерживает отправку с флагом SMTPUTF8
- Не делает валидацию email

smtplib: Пример отправки E-mail

```
import smtplib

headers = ("From: %s\r\nTo: %s\r\n\r\n"
      % (fromaddr, ", ".join(toaddrs)))
msg = headers + body

server = smtplib.SMTP('localhost')
server.set_debuglevel(1)
server.sendmail(fromaddr, toaddrs, msg, mail_options=['SMTPUTF8'])
server.quit()
```

smtplib: Рекомендация

- Использование требует осторожности
- Необходимо предварительно выполнять нормализацию и валидацию email при помощи другой библиотеки (например email_validator)
- Рекомендуется

Frameworks

Django auth

- Популярный пакет для управления авторизацией при помощи email.
- Не совместим с EAI
- Меинтейнеры отвергли соответствующий pull request который бы добавил поддержку EAI

Django auth: Рекомендация

- Не рекомендуется

Использование Баз Данных

Использование Баз Данных

- SQL
  - Доменные имена: максимум: 255 октетов, 63 октета на метку. Однако, в UTF-8 длина переменная.
  - Рекомендуется использовать колонки со строками переменной длины
  - Проверьте что на самом деле использует Object-relational mapping (ORM) драйвер, если у вас такой есть.

- noSQL
  - Уже UTF-8 переменной длины

Best Practices

Best Practices

- Валидация ввода для EAI, IDN, UA это сложно.
- Некогда не полагайтесь на статический список TLD. Они приходят и уходят.
- Не кодируй любой специфический синтаксис кроме того что есть в стандартах. Например, метка, такая как TLD, может быть длиной до 63 октетов в A-Label/ACSII формате и включает в себя префикс ‘xn--’ для IDN,
  - Поэтому кодировать что TLD содержит максимум 6 или 7 октетов просто неверно.
- Используйте тип str для хранения доменного имени и E-mail адреса
- Убедитесь что ввод нормализуется перед любым сохранением, сравнением и обработкой.
- При хранении идентификаторов в базе данных, убедитесь что весь путь данных включая саму базу совместим с UA. Например, кодировка колонки для хранения идентификатора (домена или E-mail) в SQL базах данных должна быть UTF-8.

Best Practices (продолжение)

- Лучше сделать базовую валидацию, а затем DNS запросы с отловом ошибок, чем пытаться сделать слишком много валидации.
- Используйте библиотеку/фреймворк с поддержкой UA (IDNA2008 для доменов)
- Выполняйте системное и юнит тестирование для UA идентификаторов
  - Для начала, используйте наборы тестовых данных из UASG0004.
- Рассмотрите возможность конвертировать доменные имена в их A-Label эквивалент перед передачей библиотекам, так может быть безопаснее для данных при прохождении их по полному пути (который может включать библиотеки зависимостей и которые не поддерживают UA)

Валидация и Разрешение Доменных Имён

- Конвертируйте U-label в A-Label и затем передайте стандартным методам
- Думайте об U-label в контексте отображения.
- Преобразование между A-Label и U-Label является двунаправленным и не теряет данные, поэтому нет необходимости хранить оба вида меток. Можно оставить A-Label так как они лучше поддерживаются везде в коде и зависимостях.
- Однако, U-Label нужны для сортировки, сравнения и поиска, поскольку их сортировка основана на реальном значении: т.е. на строке UTF-8, вместо punycode представления.
- Перед отображением строки, всегда преобразуйте в U-Label, поскольку конечный пользователь ожидает U-Label

Валидация и Отправка E-mail

- Выполните нормализацию локальной части в UTF-8 если получен как ввод
- Всегда используйте нормализованные локальные части при сравнении, сортировке и поиске
- Для доменной части, смотри раньше
- Валидируйте правильной библиотекой перед отправкой
- При отправке E-mail на EAI адрес будьте готовы что:
  - E-mail может быть отвергнут вашим почтовым сервером исходящей почты
  - E-mail может не дойти до получателя, если один из серверов по дороге не поддерживает EAI.

Заключение

- Будьте в курсе, что UA идентификаторы могут не поддерживаться в полной мере программами и библиотеками
- Используйте правильные библиотеки и фреймворки
- Адаптируйте свой код для корректной поддержки UA
- Делайте системное и юнит тестирование используя тестовые данные для UA что бы убедиться что ваша поддержка UA действительно работает

UA References

- http://uasg.tech
- Use Cases for UA Readiness Evaluation, UASG-0004
- Reviewing Programming Languages and Frameworks for Compliance with Universal Acceptance Good Practice, UASG-018
- Evaluation of Software libraries for UA Readiness: http://uasg.tech/software
- Universal Acceptance Readiness Framework, UASG-026

IDN References

- Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.
- Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.
- Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.
- Alvestrand, H., Ed., and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, DOI 10.17487/RFC5893, August 2010, <https://www.rfc-editor.org/info/rfc5893>.
- Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, DOI 10.17487/RFC5894, August 2010, <https://www.rfc-editor.org/info/rfc5894>.
- Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <https://www.rfc-editor.org/info/rfc3492>.

EAI References

- Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.
- Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <https://www.rfc-editor.org/info/rfc6531>.
- Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/info/rfc6532>.
- Hansen, T., Ed., Newman, C., and A. Melnikov, "Internationalized Delivery Status and Disposition Notifications", RFC 6533, DOI 10.17487/RFC6533, February 2012, <https://www.rfc-editor.org/info/rfc6533>.
- Levine, J. and R. Gellens, "Mailing Lists and Non-ASCII Addresses", RFC 6783, DOI 10.17487/RFC6783, November 2012, <https://www.rfc-editor.org/info/rfc6783>.
- Resnick, P., Ed., Newman, C., Ed., and S. Shen, Ed., "IMAP Support for UTF-8", RFC 6855, DOI 10.17487/RFC6855, March 2013, <https://www.rfc-editor.org/info/rfc6855>.
- Gellens, R., Newman, C., Yao, J., and K. Fujiwara, "Post Office Protocol Version 3 (POP3) Support for UTF-8", RFC 6856, DOI 10.17487/RFC6856, March 2013, <https://www.rfc-editor.org/info/rfc6856>.
- Fujiwara, K., "Post-Delivery Message Downgrading for Internationalized Email Messages", RFC 6857, DOI 10.17487/RFC6857, March 2013, <https://www.rfc-editor.org/info/rfc6857>.
- Gulbrandsen, A., "Simplified POP and IMAP Downgrading for Internationalized Email", RFC 6858, DOI 10.17487/RFC6858, March 2013, <https://www.rfc-editor.org/info/rfc6858>.