КООРДИНАЦИОННЫЙ ЦЕНТР ДОМЕНОВ .RU/.РФ

Поддерживаю .РФ

Введение в поддержку интернационализированных доменных имен и адресов электронной почты для Java-разработчиков

Антон Воршевский, архитектор информационных систем и популяризатор программирования

25 ноября 2020

Подготовлено на основе материалов Universal Acceptance Steering Group:
Universal Acceptance for Java Developers Tutorial

Целевая аудитория, Задачи и Цели

- Целевая аудитория:
  - Java разработчики
  - 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 Java 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 Java 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

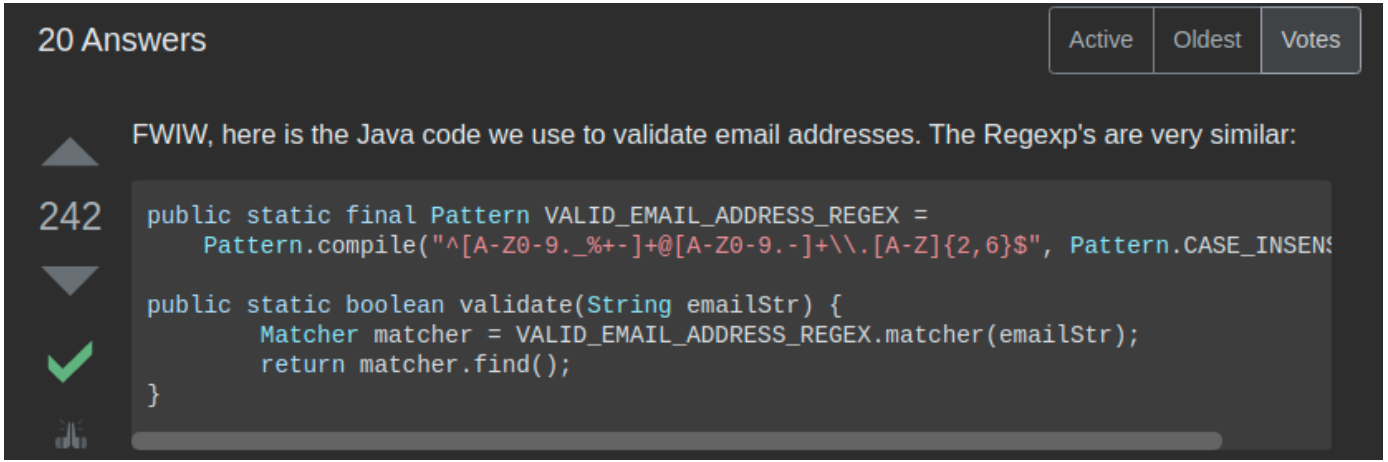
Валидация E-mail: Реальный пример

- 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 (isEmailValid(email)) {
  subscribe();
} else {
  throw new 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 существующий 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 пакет com.sun.mail:javax.mail:1.5.6 используемый для отправки E-mail по SMTP уже имеет функцию "validate". Они переписали метод isEmailValid:

```
public static boolean isEmailValid(String email) {
  try {
    var iEmail = new javax.mail.internet.InternetAddress(email);
    iEmail.validate();
    return true;
  } catch (AddressException e) {
    return false;
  }
}
```

Валидация E-mail

- Однако, они осознали что метод продолжает считать ввод некорректным. Они увидели, что поддержка интернационализации была исправлена в новой версии, поэтому они обновились до -> com.sun.mail:jakarta.mail:1.6.5

```
public static boolean isEmailValid(String email) {
  try {
    var iEmail = new javax.mail.internet.InternetAddress(email);
    iEmail.validate();
    return true;
  } catch (AddressException e) {
    return false;
  }
}
```

- Наконец, изучая эти исправления в javamail и переименованной версии библиотеки jakartamail, они поняли что нужно так же изменить функцию subscribe, а их SMTP сервер 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. добавленные 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 Суффиксов (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 мэйнтейнеров. 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 (Универсальное 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)

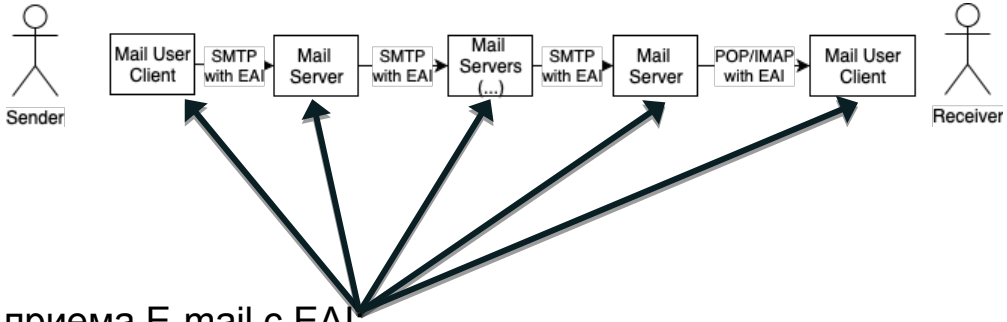
Добавленные и удаленные TLD:

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

Интернационализированные E-mail Адреса (EAI)

- e-mail синтаксис: 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 принимающий сервер его поддерживает. Если нет, письмо возвращается отправителю. Опция SMTPUTF8 используется для этой цели в протоколе SMTP (Simple Mail Transport Protocol)

EAI Путь Доставки

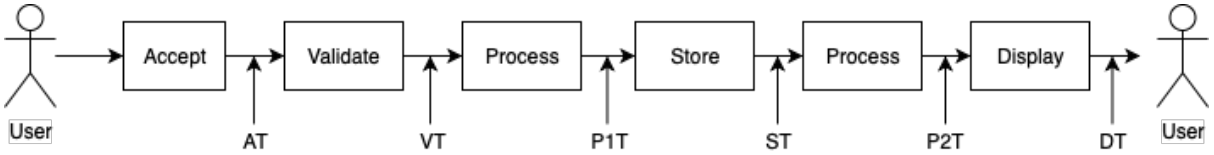


- для отправили и приема E-mail с EAI:
  - все участники по пути доставки E-mail должны обновиться для поддержки EAI
  - если любой SMTP сервер по дороге не поддерживает EAI, тогда E-mail не сможет быть доставлен.

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 не релевантных 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.

Java

Версии Java

- Примеры кода тестировались на Java 11 (версия Oracle) и Android API/SDK 26 где это возможно
- Возможно что старые версии могут иметь проблемы.
- Некоторые библиотеки могут потребовать (не обязательно из-за UA) более свежие версии Java
- Если это явно не сказано, примеры должны работать на VM любого вида: Oracle, OpenJDK или Android (Dalvik/ART)

Версии Библиотек

- Этот семинар демонстрирует множество библиотек встречающихся в дикой природе. Хотя список не исчерпывающий, он достаточно полон, что бы помочь понять какую библиотеку и как вам лучше использовать, особенно если ваш софт уже разрабатывается некоторое время.
  - 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.
- Java тип String хорошо подходит для хранения этих идентификаторов, поскольку нативно поддерживает Unicode. Поэтому большинство библиотек ожидают тип String.
- Кодировка по умолчанию (Charset.defaultCharset()) в большинстве систем обычно UTF-8. Проверь (`java -XshowSettings`) или измени дефолтную кодировку для конкретной используемой Java VM.
  - Подробнее по ссылке JEP

Насколько Базовых Тестовых Случаев

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

```
List<String> testDomains = List.of(
  "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.tm",               // ulabel.ulabel
  "exâmple.xn--o3cw4h",        // ulabel.alabel
  "xn--exmple-xta.xn--o3cw4h" // alabel.alabel
);
List<String> testLocalParts = List.of(
  "user",
  "kévin"
);
```

Валидация Доменного Имени

Используя Чистую Java

- Традиционный способ сделать резолв имени хоста и резолв sockets
- `import java.net.InetAddress;`
- `getByName(String host); getAllByName(String host);`
- `Socket(String host, int port);`
  - Внутри использует getByName()
- Кидает UnknownHostException на любую ошибку:
  - у хоста нет IP адресов
  - некорректный хост
  - плохой синтаксис
  - ...
- Передает host String как есть системному вызову OS без валидации .
  - таким образом, невалидные домены (с плохими Ulabels) проходят.
  - результат зависит от реализации внутри OS

Используя Чистую Java: Пример

```
import java.net.InetAddress;

try {

InetAddress[] hosts = getAllByName(input);

    } catch (UnknownHostException e) {

}
```

Используя Чистую Java: Пример

```
import java.net.InetAddress;

try {

InetAddress host = getByName(input);

    } catch (UnknownHostException e) {

}
```

Используя Чистую Java: Пример

```
import java.net.InetAddress;

try {

Socket socket =  new Socket(input, 1234);  // 1234 = port number)

    } catch (UnknownHostException e) {

}
```

Используя Чистую Java: Рекомендация

- Не используйте имя хоста в прямую как есть.
- Вместо этого:
  - Валидируйте имя хоста перед вызовом getByName()
    - 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 конвертировав IDN в A-label) используя другую библиотеку, а уже затем используйте базовые вызовы
    - Ваш код станет более независим от того на какой OS работает VM
  - Или используйте другую библиотеку для подготовки и выполнения запроса

JRE-IDN

- Является частью JRE
- Реализует IDNA2003.
- `import java.net.IDN;`
- String domain = IDN.toASCII(Utf8DomainString);
- Кидает IllegalArgumentException
  - делает только базовую валидацию, не проверяет что UTF-8 строка является валидной меткой

JRE-IDN: Пример

```
import java.net.IDN;

try {

String asciiEncodedDomain = IDN.toASCII(input);

    } catch (IllegalArgumentException e) {


}
```

JRE-IDN: Пример

```
import java.net.IDN;

try {

String unicodeEncodedDomain = IDN.toUnicode(input);

    } catch (IllegalArgumentException e) {


}
```

JRE-IDN: Рекомендация

- Не использовать
  - поскольку базируется IDNA2003

Apache Commons Validator

- Реализует валидаторы для доменов и E-mail
- https://github.com/apache/commons-validator
- Maven Repository:
  - ```
    <dependency>
     <groupId>commons-validator</groupId>
     <artifactId>commons-validator</artifactId>
     <version>1.6</version>
    </dependency>
    ```
    ```

Apache Commons Validator (продолжение)

- Включает статический список TLD в коде
  - Список обновляется с новыми релизами кода
    - Таким образом, между релизами, уже на следующий день после релиза, список не актуален.
    - Даже при релизе с новейшей версией библиотеки, корректность ее поведения будет меняться в зависимости от состояния актуального списка
  - Поэтому будет выдавать неверные результат для некоторых TLD:
    - TLD удаленных после даты синхронизации мэйнтейнерами данных из реестра TLD IANA с их статическим списком в коде
    - TLD добавленных после даты синхронизации мэйнтейнерами данных из реестра TLD IANA с их статическим списком в коде

Apache Commons Validator: Пример

```
import org.apache.commons.validator.routines.DomainValidator;

DomainValidator validator = DomainValidator.getInstance();

    if (validator.isValid(input)) {

}
```

Apache Commons Validator: Рекомендация

- очень хорошая библиотека, но не используйте ее
  - из-за статического списка TLD (всегда устаревшего)
- Не Рекомендуется

International Components for Unicode (ICU)

- Золотой стандарт библиотеки для Unicode. Разработана IBM. Сейчас обслуживается Unicode. Синхронизирована со стандартами Unicode.
- Есть Java версия (ICU4J): http://site.icu-project.org/home
- ICU4J не совсем написана на Java. Это прямой маппинг версии на C. По этой причине Java разработчики могут не любить ее.
- Maven Repository:

```
<dependency>
 <groupId>com.ibm.icu</groupId>
 <artifactId>icu4j</artifactId>
 <version>65.1</version>
</dependency>
```

ICU

- IDNA преобразование базируется на Unicode TR46 (который поддерживает переход с IDNA2003 на IDNA2008). Однако возможно сконфигурировать не поддерживать переходный режим (рекомендуется)
- IDNA преобразование включает нормализацию в соответствии с IDNA (отлично!)
- Не используй IDNA2003 методы (convertTo*)
- Вывод методов может содержать плохие доменные имена, где запрещенные символы заменены на U+FFFD.
- Проверяйте наличие ошибок при преобразовании вызовом info.hasErrors()

ICU: Пример

```
import com.ibm.icu.text.IDNA;

IDNA validator = IDNA.getUTS46Instance(
    IDNA.NONTRANSITIONAL_TO_ASCII
  | IDNA.NONTRANSITIONAL_TO_UNICODE
  | IDNA.CHECK_BIDI
  | IDNA.CHECK_CONTEXTJ
  | IDNA.CHECK_CONTEXTO
  | IDNA.USE_STD3_RULES);
StringBuilder output = new StringBuilder();
IDNA.Info info = new IDNA.Info();
validator.nameToASCII(input, output, info);

if (info.hasErrors()) {}
```

опции не использовать UTS46 transitional feature
и использовать улучшенную валидацию.

ICU: Рекомендация

- Наиболее актуальная библиотека для работы с Unicode
- Для IDN доменов, установите опции что бы ограничить валидацию и использование в соответствии с IDNA2008.

Guava

- Разработана Google. Имеет метод hostname.
- https://github.com/google/guava
- Maven Repository:

```
<dependency>
 <groupId>com.google.guava</groupId>
 <artifactId>guava</artifactId>
 <version>28.2-jre</version>
</dependency>
```

Guava

- isValid(String domain) выполняет только базовую валидацию
  - Невалидный U-label проходит
- Методы from(String domain) и is*Suffix используют публичный список суффиксов (PSL).
  - Поэтому может быть не синхронизирована с текущим списком TLD.
  - PSL берется из используемой версии библиотеки Guava.

Guava: Пример

```
import com.google.common.net.InternetDomainName;

 if (InternetDomainName.isValid(input)) {

}

InternetDomainName domain = InternetDomainName.from(input);

if (domain.isPublicSuffix()) {

}
```

Guava: Рекомендация

- Не годиться для валидации
- При использовании,
  - осознавайте зависимость от PSL, статически вшитого в библиотеку
  - часто обновляете библиотеку

Xcode

- Библиотека разработанная Verisign. Имеет объект "Idna"
- https://www.verisign.com/en_US/channel-resources/domain-registry-products/idn-sdks/index.xhtml
- Нет Maven репозитория (только zip файл с jar файлом внутри)

Xcode: Пример

```
import com.vgrs.xcode.common.Unicode;

import com.vgrs.xcode.idna.Idna;

import com.vgrs.xcode.idna.Punycode;


 Idna idna = new Idna(new Punycode(), true, true);

 int[] output = idna.domainToUnicode(input.toCharArray()); // see domainToAscii for
roundtrip

 String domain = new String(Unicode.decode(output));
```

Xcode: Рекомендация

- Медленно (в тестах обработка домена занимает до 5 секунд)
- нет Maven репозитория
- Но превосходно реализован IDNA2008

Высокоуровневые средства

- HTTP Frameworks
- Могут использовать внутри Java URL/URI

Java URL

- import java.net.URL;
- Поддерживает все протоколы (не только http/https но ftp, file, …)
- Не валидирует часть hostname.
- Когда некорректен, кидает MalformedURLException

Java URL: Пример

```
import java.net.URL;

    try {

      URL url = new URL("http://" + input);

    } catch (MalformedURLException e) {

    }
```

Java URI

- Тоже что для URL
- Когда некорректен, кидает URISyntaxException

Java URI: Пример

```
import java.net.URI;

    try {

      URI uri = new URI("http://" + input);

    } catch (URISyntaxException e) {

    }
```

Java URI/URL Рекомендация

- Можно использовать, но не валидирует hostname
- Используйте другую библиотеку для валидации hostname

Выполнение HTTP Запроса

Java 1.1 HttpURLConnection

- старый способ
- использует java.net.URI
  - поэтому наследует его особенности
- Поскольку сейчас существует множество HTTP библиотек/пакетов/фреймворков, лучше использовать другую.

Apache HTTPClient

- старый способ
- Нет валидации домена
- Maven:

```
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient</artifactId>
  <version>4.5.10</version>
</dependency>
```

Apache HTTPClient: Пример

import org.apache.http.client.methods.HttpGet;

Example:

```
try {
    HttpGet request = new HttpGet("http://" + input);
} catch (IllegalArgumentException e) {
}
```

Apache HTTPClient: Рекомендация

- Поскольку сейчас существует множество HTTP библиотек/пакетов/фреймворков, лучше использовать другую

OkHTTP

- Более свежая, остается актуальной (поддерживает http/2), поддерживается, использует конструкции Builder() и более популярна чем предыдущие. Разработана Square.
- Была написана на Java, но перешла на Kotlin (остается совместимой с Java)
- https://square.github.io/okhttp/
- Maven:

```
<dependency>
        <groupId>com.squareup.okhttp3</groupId>
        <artifactId>okhttp</artifactId>
        <version>4.2.2</version>
 </dependency>
```

OkHTTP

- Предоставляет метод для использования публичного списка суффиксов (но не использует по умолчанию)
- Не валидирует имя хоста в URL
- Автоматически кодирует IDN U-label вызовом java.net.IDN
  - Поэтому наследует особенности java.net.IDN

OkHTTP: Пример

```
import okhttp3.Response;

OkHttpClient httpClient = new OkHttpClient();
Request request = new Request.Builder().url("http://" + input).build();
try {
  Response response = httpClient.newCall(request).execute();
} catch (IOException e) {
}
```

OkHTTP: Пример

```
import okhttp3.HttpUrl;

HttpUrl url = HttpUrl.parse(“http://” + input);

if (url == null) { }
```

OkHTTP: Рекомендации

- Лучше не использовать методы с публичным списком суффиксов
- Валидируйте и подготовьте имена хостов перед использование OkHTTP
- Используйте библиотеку IDNA2008 для преобразования в A-Labels что бы OkHTTP не попыталась конвертировать используя java.net.IDN которая использует IDNA2003.

Java 11 HTTP Client

- Новый HTTP Client с современными конструкциями (Builder,…) встроен в Java 11
- Не поддерживает UTF8 в имени хоста. Кидает IllegalArgumentException
- использует java.net.URI
  - Поэтому наследует характеристики java.net.URL/URI
- Не проверяет валидность IDN (например пропускает невалидный punycode)
- Предполагает классическое имя хоста (ASCII, RFC1035).

Java 11 HTTP Client: Пример

```
import java.net.http.HttpRequest;

try {
  HttpRequest request = HttpRequest.newBuilder()
    .GET()
    .uri(URI.create("http://" + input))
    .build();
} catch (IllegalArgumentException e) {


}
```

Java 11 HTTP Client: Рекомендация

- Поскольку включен в Java, не требует дополнительных библиотек или пакетов, поэтому отсутствуют зависимости и необходимость следить за версиями
- Однако, поскольку отсутствует валидации и подготовка UTF8 IDN, требуются дополнительные шаги по подготовке и валидации имени хоста перед вызовом HTTP client.

Google Java HTTP Client

- Google Library
- Содержит GenericURI() class для работы с URI
  - Который использует java.net.URL/URI внутри
    - поэтому наследует характеристики java.net.URL/URI
- https://github.com/googleapis/google-http-java-client
- Maven:

```
<dependency>
  <groupId>com.google.http-client</groupId>
  <artifactId>google-http-client</artifactId>
</dependency>
```

Google Java HTTP Client: Пример

```
import com.google.api.client.http.GenericUrl;

try {
  HttpTransport HTTP_TRANSPORT = new NetHttpTransport();
  GenericUrl url = new GenericUrl("http://" + input);
  if (url.host = “”) { }
  HttpRequest request = HTTP_TRANSPORT
    .createRequestFactory()
    .buildGetRequest(url);
  HttpResponse response = request.execute();
} catch (IllegalArgumentException e) {}
```

Google Java HTTP Client: Рекомендация

- Поскольку рассчитывает на java.net.URI, не валидирует и не подготавливает имя хоста
- Нужно подготовить и валидировать имя хоста при помощи другой библиотеки перед вызовом GenericURL()

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 (недопустим UTF-8 в локальной части: [a-zA-Z0-9_+&*-])
  - Не поддерживает ASCII TLD длиннее 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 ‘@’.

Jakarta Mail

- Наиболее используемый Java пакет для отправки E-mail
- Также содержит метод validate() для валидации E-mail адреса
- import javax.mail
- Maven:

```
<dependency>
  <groupId>com.sun.mail</groupId>
  <artifactId>jakarta.mail</artifactId>
  <version>1.6.5</version>
</dependency>
```

Jakarta Mail

- validate() хорошо валидирует E-mail адреса, особенно локальную часть.
  - Проверяет наличие недопустимых символов: ()<>,;:"[]\ , различные пробелы и т.д.
  - Проверят что символы это только цифры и буквы по определению классов Unicode.
  - Не валидирует IDN

Jakarta Mail: Пример

```
import javax.mail.internet.InternetAddress;
try {
 InternetAddress emailAddr = new InternetAddress(input);
 emailAddr.validate();
} catch (AddressException e) {}
```

Jakarta Mail: Рекомендация

- Хорошая библиотека для использования.
- Добавьте валидацию и подготовку IDN домена как дополнительный шаг
- Не используйте старый Java Mail пакет (com.sun.mail:javax.mail), с тех пор как Java Mail стал Jakarta Mail было выпущено множество исправлений в работе с UTF-8

Apache Commons Validator

- Содержит валидаторы доменов и E-mail
- Имеет статический список TLD!!! Всегда Устаревший!
- https://github.com/apache/commons-validator
- Maven Repository:
  - ```
    <dependency>
     <groupId>commons-validator</groupId>
     <artifactId>commons-validator</artifactId>
     <version>1.6</version>
    </dependency>
    ```
    ```

Apache Commons Validator

```
EmailValidator validator = EmailValidator.getInstance();
 boolean emailValid = validator.isValid(input);
                if (emailValid) {
```

Apache Commons Validator: Рекомендация

- Не использовать, так как библиотека зависит от статического списка TLD

EmailValidator4J

- https://github.com/egulias/EmailValidator4J
- Заявляет поддержку EAI!
- Состояние разработки и поддержки неизвестно.
- Приглядеться

Отправка E-mail

JakartaMail

- Тоже что и раньше, смотри выше
- Maven:

```
<dependency>
  <groupId>com.sun.mail</groupId>
  <artifactId>jakarta.mail</artifactId>
  <version>1.6.5</version>
</dependency>
```

JakartaMail: Пример отправки E-mail

```
Properties properties = System.getProperties();
properties.setProperty("mail.smtp.host", host);
Session session = Session.getDefaultInstance(properties);
try {
  MimeMessage message = new MimeMessage(session);
  message.setFrom(new InternetAddress(from));
  message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
  message.setSubject("This is the Subject Line!");
  message.setText("This is actual message");
  Transport.send(message);
} catch (MessagingException e) {
}
```

JakartaMail: Рекомендация

- Хорошая библиотека для использования.
- Поддерживает EAI (корректно с версии 1.6.5)
- Смотри разбор про валидацию выше

Simple Java Mail

- Jakarta Mail обертка. Упрощает отправку E-mail.
- Поддерживает много современных возможностей
- Более современные конструкции (Builder)
- https://github.com/bbottema/simple-java-mail/
- http://www.simplejavamail.org
- Maven

```
<dependency>
  <groupId>org.simplejavamail</groupId>
  <artifactId>simple-java-mail</artifactId>
  <version>6.0.5</version>
</dependency>
```

Simple Java Mail Валидация

- Использует другую библиотеку для валидации E-mail
  - https://github.com/bbottema/email-rfc2822-validator.git
- Maven:

```
<dependency>
  <groupId>com.github.bbottema</groupId>
  <artifactId>emailaddress-rfc2822</artifactId>
  <version>2.1.4</version>
</dependency>
```

Simple Java Mail Валидация (продолжение)

- Использует различные регулярные выражения
- Считает любой UTF-8 некорректным, поэтому никаких U-label в доменах, никаких EAI

Simple Java Mail: Пример

```
Mailer mailer = MailerBuilder
  .withSMTPServer("smtp.host.com”)
  .async();
Email email = EmailBuilder.startingBlank()
  .to("user@example.org")
  .buildEmail();
mailer.sendMail(email);
```

Simple Java Mail: Рекомендации

- Хотя предоставляет современный интерфейс для отправки почты, не поддерживает EAI и U-label для доменов.
- Внутренняя валидация основана на устаревшем RFC2822

Frameworks

SpringBoot Framework

- Популярен в мире Java, server side
- https://spring.io
- Умеет @Email аннотации, http запросы, отправку E-mail

SpringBoot HTTP Request

- Использует внутри java.net.URI
  - Наследует характеристики java.net.URI
- Maven:

```
<dependency>
 <groupId>org.springframework.boot</groupId>
 <artifactId>spring-boot-starter-web</artifactId>
 <version>2.2.4.RELEASE</version>
</dependency>
```

SpringBoot HTTP Request

- Не валидирует домены
- Преобразует доменные метки UTF8 в percent encoding -> НЕВЕРНО

SpringBoot HTTP Request: Пример

```
import org.springframework.web.client.RestTemplate
RestTemplate restTemplate = new RestTemplate();
try {
  String result = restTemplate.getForObject("http://" + input, String.class);
} catch (RestClientException e) {}
```

SpringBoot HTTP Request: Рекомендация

- Валидировать и подготовить имя хоста перед использованием этой библиотеки

SpringBoot Send Email

- Обертка для Java Mail.
  - Поэтому наследует характеристики Java Mail
- Не валидирует
- https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/mail.html
- Maven:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-mail</artifactId>
  <version>2.2.4.RELEASE</version>
</dependency>
```

SpringBoot Send Email: Пример

```
import org.springframework.mail.MailException;
import org.springframework.mail.MailSender;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.JavaMailSenderImpl;

SimpleMailMessage msg = new SimpleMailMessage();
msg.setTo(emailAddress);
msg.setText(emailtext);
try{
  mailSender.send(msg);
} catch(MailException ex) {}
```

SpringBoot Send Email: Рекомендация

- Валидировать и подготовить E-mail адреса перед использованием SpringBoot Mail
- Проверьте в версиях зависимостей что используется правильная версия Jakarta Mail поддерживающая EAI.

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

Android

android.icu.text.IDNA

- Та же ICU библиотека, интегрированная в android OS
  - Нет дополнительных зависимостей
- https://developer.android.com/reference/android/icu/text/IDNA
- Те же соображения как при обсуждении библиотеки icu4j

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 String 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)

Best Practices (продолжение)

- При использовании HTTP Requests, большинство фреймворков рассчитывает на лежащий ниже Java URL/URI, который не валидирует и не подготавливает имена хостов.
  - Подготовьте имена хостов при помощи библиотеки с IDNA2008, а затем передайте результат в HTTP фреймворк.

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.

Назад в Компанию: Подготовка E-mail Адреса

- Теперь мы знаем что команда разработчиков выполнила часть задачи. Некоторые E-mail (например customer@фирма.рф) по прежнему отвергаются Jakarta Mail потому что доменная часть не подготовлена. Ниже полный пример

Он подготавливает E-mail адрес с A-label в домене, который затем используется как ввод любой библиотеки или фреймворка. однако локальная часть остается UTF-8 что может вызвать проблемы на пути доставки почты.

```
IDNA validator = IDNA.getUTS46Instance(
    IDNA.NONTRANSITIONAL_TO_ASCII
    | IDNA.NONTRANSITIONAL_TO_UNICODE
    | IDNA.CHECK_BIDI
    | IDNA.CHECK_CONTEXTJ
    | IDNA.CHECK_CONTEXTO
    | IDNA.USE_STD3_RULES);
IDNA.Info info = new IDNA.Info();
String localpart = email.substring(0, email.lastIndexOf("@"));
String domain = email.substring(email.lastIndexOf("@") + 1);
StringBuilder output = new StringBuilder();
validator.nameToASCII(domain, output, info);
email = localpart + "@" + output.toString();

if (isEmailValid(email)) {
  subscribe();
} else {
  throw new Exception("Invalid email address, please review it and submit again");
}
```

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
- Примеры кода из данной презентации можно найти тут
- https://github.com/marcblanchet/ua-tutorial
- https://github.com/icann/ua-java-tutorial

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>.

КООРДИНАЦИОННЫЙ ЦЕНТР ДОМЕНОВ .RU/.РФ

Поддерживаю .РФ

Спасибо за участие!

Подробнее на сайте
Поддерживаю.РФ