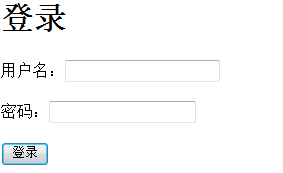
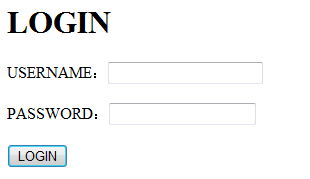
# 国际化（i18n）

* 国际化（Internationalization）指的是同一个网站可以支持多种不同的语言，以方便不同国家，不同语种的用户访问。
* 关于国际化我们想到的最简单的方案就是为不同的国家创建不同的网站，比如苹果公司，他的英文官网是：http://www.apple.com而中国官网是http://www.apple.com/cn
* 苹果公司这种方案并不适合全部公司，而我们希望相同的一个网站，而不同人访问的时候可以根据用户所在的区域显示不同的语言文字，而网站的布局样式等不发生改变。
* 于是就有了我们说的国际化，国际化总的来说就是同一个网站不同国家的人来访问可以显示出不同的语言。但实际上这种需求并不强烈，一般真的有国际化需求的公司，主流采用的依然是苹果公司的那种方案，为不同的国家创建不同的页面。所以国际化的内容我们了解一下即可。
* 国际化的英文Internationalization，但是由于拼写过长，老外想了一个简单的写法叫做I18N，代表的是Internationalization这个单词，以I开头，以N结尾，而中间是18个字母，所以简写为I18N。以后我们说I18N和国际化是一个意思。
* 当然国际化不仅仅是语言的国际化，还包括日期、时间、数字等。

## 原理

* 国际化最常见的应用就是登录页面，像这种国际化的页面，如果在英文环境下登录则显示英文内容，中文环境下显示中文内容。



* 那我们说这个是如何做到的呢?
  + 首先我们需要知道，用户使用的是哪种语言，这个问题最好解决，因为浏览器在发送请求时会发送如下请求头：
    - Accept-Language: zh-CN,en-US
    - 这个头的意思是告诉服务器，当前浏览器优先使用的语言是zh-CN.
      * zh 代表中文
      * CN 代表中国
      * zh-CN 连起来代表简体中文
      * 而en-US是我们第二种可以使用的语言
    - 通过该头就可以判断用户使用的语言
* 已经知道用户使用的语言了，下一步需要我们根据不同的语言显示不同的内容，那我们想一下，如果说我们在页面中把这些诸如：登录、用户名、密码等信息都写死了。那么，将不能再修改，所以在页面中这些需要国际化的内容都是需要动态获取的，也就是他们实际上都是变量。
* 我们一般采取的方式是将这些需要国际化的内容写到一个资源配置文件中，然后程序根据不同的语言加载不同的文件，就达到了一个国际化的目的。

## 核心类

* java.util.Locale
  + 作用：Locale表示一个语言环境的类，通过该类可以创建和获取指定的表示一个国家和语言的对象，国际化的操作主要使用Locale对象来表示当前语言环境。
  + 常量：
    - static public final Locale CHINA
    - static public final Locale GERMANY
    - static public final Locale US
      * 这些常量可以直接返回指定的语言和国家Locale对象
  + 方法：
    - public static Locale getDefault()
    - 获得当前语言环境
  + 构造器：
    - public Locale(String language, String country)
      * 两个参数的构造器，用来根据语言和国家创建Locale对象。
      * 比如：new Locale(“zh”,”CN”)就是创建了一个简体中文的环境。
    - public Locale(String language)
      * 一个参数的构造器，用来根据语言创建Locale对象
      * 比如：new Locale(“en”)创建了一个英文环境对象
    - 常见的语言和国家

|  |  |  |  |
| --- | --- | --- | --- |
| 语言 | 代码 | 国家 | 代码 |
| 简体中文 | zh | 中国 | CN |
| 英语 | en | 美国 | US |
| 德语 | de | 德国 | DE |
| 法语 | fr | 法国 | FR |
| 日语 | ja | 日本 | JP |

* java.util.ResourceBundle
  + 作用：根据不同的Locale对象加载国际化资源文件
  + 方法：
    - public static final ResourceBundle getBundle(String baseName)
      * 根据资源包的基础名加载资源文件（使用系统默认的Locale对象）
    - public static final ResourceBundle getBundle(String baseName,Locale locale)
      * 根据资源包的基础名和Locale对象加载资源
      * 这两个方法返回的都是一个ResourceBundle对象
    - public final String getString(String key)
      * 根据资源的key获取到资源的值，返回一个字符串
  + 外部资源文件
    - 外部资源文件我们一般放到src目录下
    - 资源文件是一个properties类型的文件
    - 文件名的规则是：
      * 基础名\_语言\_国家.properties
      * 如：
        + res\_zh\_CN.properties
        + res\_en\_US.propreties
        + 这样ResourceBundle将会根据不同的Bundle加载不同的资源文件
      * 文件中是键值对的结构：

|  |
| --- |
| username=USERNAME  password=PASSWORD  login=LOGIN |

* java.text.DateFormat
  + 作用：格式化日期
  + 常用方法：
    - public final static DateFormat getDateInstance(int style)
      * style是代表日期的风格
      * 不设置Loacle默认当前服务器的语言环境
    - public final static DateFormat getDateInstance(int style, Locale aLocale)
      * style是代表日期的风格
      * aLocale代表当前服务器语言环境
    - style需要通过DateFormat中的常量来设置：
      * public static final int FULL = 0;
      * public static final int LONG = 1;
      * public static final int MEDIUM = 2;
      * public static final int SHORT = 3;
      * public static final int DEFAULT = MEDIUM;
* java.text.NumberFormat
  + 作用：格式化数字
  + 方法：
    - public static NumberFormat getNumberInstance(Locale inLocale)
      * 获取NumberFormat对象
    - public final String format(double number)
      * 格式化一个数字，返回String类型