家谱开放数据接口说明

1. 综述

家谱开放数据接口目前提供以下两种形式的数据接口。

1. 通过访问资源URI获取数据：根据标准API接口，通过访问单个资源的URI，获得该资源的所有RDF三元组(属性和值)。
2. 通过查询接口获取特定类型的数据：通过特定API接口，获取“姓氏”、“先祖名人”、“地点”、“机构”、“朝代”、“书目”的数据。（每次最多取得20条数据）
3. 通过Sparql Endpoint获取数据

注：使用家谱开放数据接口时需要提供APIKey进行验证。开发人员请在上海图书馆数据开放平台进行用户注册，并获取独立的APIKey。

注册网址：http://data.library.sh.cn/jp/userlogin/tologin

1. 接口调用方式说明
2. 通过访问资源URI获取数据

功能：输入资源URI，获取资源的RDF数据。如返回结果数据中存在其他资源的URI时，可通过该接口获取其他资源的RDF数据。

API接口： http://data.library.sh.cn/jp/data/json?uri=[参数1]&key=[参数2]

输入：

[参数1]：资源URI

[参数2]：用户的APIKey

输出：包含资源及其属性和值的JSON-LD数据

资源类型与URI设计规范：

|  |  |
| --- | --- |
| 对象 | 命名空间 |
| 家谱文献题名 | http://data.library.sh.cn/jp/authority/title/ |
| 实例（bf:Instance） | http://data.library.sh.cn/jp/resource/instance/ |
| 朝代 | http://data.library.sh.cn/authority/temporal/ |
| 版本类型取值词表 | http://data.library.sh.cn/vocab/edition/ |
| 作品（bf:Work） | http://data.library.sh.cn/jp/resource/work/ |
| 谱籍地 | http://data.library.sh.cn/entity/place/ |
| 机构 | http://data.library.sh.cn/entity/organization/ |
| 单件（bf:Item） | http://data.library.sh.cn/jp/resource/item/ |
| 人物 | http://data.library.sh.cn/jp/entity/person/ |
| 堂号 | http://data.library.sh.cn/jp/authority/titleofancestraltemple/ |
| 姓氏 | http://data.library.sh.cn/authority/familyname/ |

例如：

输入：http://data.library.sh.cn/jp/data/json?

uri=http://data.library.sh.cn/jp/authority/title/huk223doncajistd

&key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"@id": "http://data.library.sh.cn/jp/authority/title/huk223doncajistd",

"@type": [

"http://bibframe.org/vocab/WorkTitle",

"http://bibframe.org/vocab/Title"

],

"label": [

{

"@language": "cht",

"@value": "豐原陳氏家譜"

},

{

"@language": "chs",

"@value": "丰原陈氏家谱"

}

],

"@context": {

"label": "http://bibframe.org/vocab/label"

}

}

1. 通过查询接口获取特定类型的数据
2. 姓氏

功能：输入姓氏获取对应的资源数据。数据匹配方式为完全匹配。

API接口： http://data.library.sh.cn/jp/familyname/[参数1]?key=[参数2]

输入：

[参数1]：姓氏

[参数2]：用户的APIKey

输出：包含姓氏属性和值的JSON-LD数据

例如：

输入：[http://data.library.sh.cn/jp/familyname/](http://data.library.sh.cn/topic/familyname/)陈?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"@id": "http://data.library.sh.cn/authority/familyname/9i5ukkrs9ebh6qmz",

"@type": "http://www.library.sh.cn/ontology/FamilyName",

"label": [

{

"@language": "chs",

"@value": "陈"

},

{

"@language": "en",

"@value": "chen"

},

{

"@language": "cht",

"@value": "陳"

}

],

"description": "见于《世本》。西汉《急就章》列为汉代常见姓氏之一。春秋时陈国有陈亢，为孔门弟子。春秋时齐国有陈乞，事景公为大夫，又战国时楚国有陈学良，学者。汉代有陈平。阳武人，开国元勋。陈氏为中国古今最常见的六大姓氏（王李张刘陈赵）之一。《中国人名大辞典》收有陈氏1012例。宋《百家姓》列为第010姓。",

"@context": {

"description": "http://www.library.sh.cn/ontology/description",

"label": "http://bibframe.org/vocab/label"

}

}

返回属性说明：

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 说明 |
| label | literial | value：姓名 language：语言 “chs”：中文简体 “cht”：中文繁体 “en”：英文 |
| description | literial | 姓氏描述 |

1. 先祖名人

功能：输入先祖名人的姓名，获取对应的资源数据。数据匹配方式为模糊匹配。

API接口：http://data.library.sh.cn/jp/person/[参数1]?key=[参数2]

输入：

[参数1]：先祖名人姓名

[参数2]：用户的APIKey

输出：包含先祖名人属性和值的JSON-LD数据（数组格式）

例如：

输入：http://data.library.sh.cn/jp/person/丁丙?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

[

{

"@id": "http://data.library.sh.cn/jp/entity/person/etrd44w3m3g1vncn",

"@type": "http://www.library.sh.cn/ontology/Person",

"label": [

{

"@language": "cht",

"@value": "丁丙"

},

{

"@language": "chs",

"@value": "丁丙"

}

],

"relatedWork": "http://data.library.sh.cn/jp/resource/work/848hidvhqj3x9uk7",

"roleOfFamily": "http://data.library.sh.cn/jp/vocab/ancestor/xian-zu",

"familyName": "http://data.library.sh.cn/authority/familyname/68n959cf8zdfkz3v",

"@context": {

"familyName": {

"@id": "http://xmlns.com/foaf/0.1/familyName",

"@type": "@id"

},

"relatedWork": {

"@id": "http://www.library.sh.cn/ontology/relatedWork",

"@type": "@id"

},

"label": "http://bibframe.org/vocab/label",

"roleOfFamily": {

"@id": "http://www.library.sh.cn/ontology/roleOfFamily",

"@type": "@id"

}

}

},

{

"@id": "http://data.library.sh.cn/jp/entity/person/jyrdohlzleg26j23",

"@type": "http://www.library.sh.cn/ontology/Person",

"label": [

{

"@language": "cht",

"@value": "丁丙"

},

{

"@language": "chs",

"@value": "丁丙"

}

],

"relatedWork": "http://data.library.sh.cn/jp/resource/work/d1gdypba7z7di1c9",

"roleOfFamily": "http://data.library.sh.cn/jp/vocab/ancestor/ming-ren",

"familyName": "http://data.library.sh.cn/authority/familyname/68n959cf8zdfkz3v",

"@context": {

"familyName": {

"@id": "http://xmlns.com/foaf/0.1/familyName",

"@type": "@id"

},

"label": "http://bibframe.org/vocab/label",

"roleOfFamily": {

"@id": "http://www.library.sh.cn/ontology/roleOfFamily",

"@type": "@id"

},

"relatedWork": {

"@id": "http://www.library.sh.cn/ontology/relatedWork",

"@type": "@id"

}

}

}

]

返回属性说明：

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 说明 |
| label | literial | value：姓名 language：语言 “chs”：中文简体 “cht”：中文繁体 “en”：英文 |
| relatedWork | URI | 相关家谱文献的URI |
| roleOfFamily | URI | 角色URI（人在家族中的角色，如始祖、始迁祖等先祖类型。） |
| familyName | URI | 姓氏URI |
| gender | literial | 性别 |
| family | URI | 家族URI |
| childOf | URI | 父亲URI |
| spouseOf | URI | 配偶URI |
| genealogyName | literial | 谱名（人在家谱上记载的谱名。） |
| courtesyName | literial | 字 |
| pseudonym | literial | 号 |
| orderOfSeniority | literial | 排行 |
| generationCharacter | literial | 字辈 |
| posthumousTitle | literial | 谥号 |
| birthday | literial | 生于 |
| deathday | literial | 卒于 |
| description | literial | 人物描述 |
| temporalValue | literial | 朝代描述 |
| temporal | URI | 朝代URI |

1. 地名

功能：输入地名，获取对应的资源数据。数据匹配方式为完全匹配。

API接口：http://data.library.sh.cn/jp/place/[参数1]?key=[参数2]

输入：

[参数1]：地名

[参数2]：用户的APIKey

输出：包含地名属性和值的JSON-LD数据（数组格式）

例如：

输入：http://data.library.sh.cn/jp/place/杞县?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"@id": "http://data.library.sh.cn/entity/place/n5sfke9d88qj3iyp",

"@type": "http://www.library.sh.cn/ontology/Place",

"label": [

{

"@language": "chs",

"@value": "杞县"

},

{

"@language": "cht",

"@value": "杞縣"

}

],

"city": "开封市",

"country": [

{

"@language": "chs",

"@value": "中国"

},

{

"@language": "cht",

"@value": "中國"

}

],

"county": "杞县",

"province": "河南省",

"sameAs": "http://www.cba.ac.cn/point/410205",

"@context": {

"country": "http://www.library.sh.cn/ontology/country",

"city": "http://www.library.sh.cn/ontology/city",

"county": "http://www.library.sh.cn/ontology/county",

"sameAs": {

"@id": "http://www.w3.org/2002/07/owl#sameAs",

"@type": "@id"

},

"label": "http://bibframe.org/vocab/label",

"province": "http://www.library.sh.cn/ontology/province"

}

}

返回属性说明：

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 说明 |
| label | literial | value：地名（对应“国家”、“省”、“市”、“县”中的最小行政区划） language：语言 “chs”：中文简体 “cht”：中文繁体 “en”：英文 |
| country | literial | 国家 |
| province | literial | 省 |
| city | literial | 市 |
| county | literial | 县 |
| sameAs | URI | 经纬度URI |
| description | literial | 地名描述 |

1. 机构

功能：输入机构的简称或全称，获取对应的资源数据。数据匹配方式为模糊匹配。

API接口：http://data.library.sh.cn/jp/organization/[参数1]?key=[参数2]

输入：

[参数1]：机构简称/全称

[参数2]：用户的APIKey

输出：包含机构属性和值的JSON-LD数据（数组格式）

例如：

输入：http://data.library.sh.cn/jp/organization/上图?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"@id": "http://data.library.sh.cn/entity/organization/brvqlrg8y55v1b5q",

"@type": "http://www.library.sh.cn/ontology/Organization",

"label": [

{

"@language": "chs",

"@value": "上海图书馆"

},

{

"@language": "cht",

"@value": "上海圖書館"

},

{

"@language": "en",

"@value": "Shanghai library"

}

],

"abbreviateName": [

{

"@language": "chs",

"@value": "上图"

},

{

"@language": "cht",

"@value": "上圖"

}

],

"address": {

"@language": "chs",

"@value": "上海市 徐汇区 淮海中路1555号"

},

"region": "http://data.library.sh.cn/entity/place/ntwya73hddzoeonr",

"@context": {

"address": "http://www.library.sh.cn/ontology/address",

"abbreviateName": "http://www.library.sh.cn/ontology/abbreviateName",

"label": "http://bibframe.org/vocab/label",

"region": {

"@id": "http://www.library.sh.cn/ontology/region",

"@type": "@id"

}

}

}

返回属性说明：

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 说明 |
| label | literial | value：机构名称 language：语言 “chs”：中文简体 “cht”：中文繁体 “en”：英文 |
| label | literial | 机构全称 |
| abbreviateName | literial | 机构简称 |
| address | literial | 地址 |
| region | URI | 机构所在地URI |

1. 朝代

功能：

1. 输入朝代、年号，获取朝代起止年数据。
2. 输入年号纪年，获取公元年数据。
3. 输入公元年，返回朝代纪年。
4. 输入朝代，返回年号、帝王、起止年
5. 获取所有朝代

API接口：

http://data.library.sh.cn/jp/data/[参数1].json?key=[参数2]

输入方式1：

[参数1]：朝代或朝代年号。

[参数2]：用户的APIKey

输出：朝代起止年的JSON-LD数据（数组格式）

例如：

输入：http://data.library.sh.cn/jp/data/明.json?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"result": {

"data": "1368~1644",

"uri": "http://data.library.sh.cn/authority/temporal/yex4deivsad41p9q"

}

}

输入方式2：

[参数1]：朝代纪年。

[参数2]：用户的APIKey

输出：公元年的JSON-LD数据（数组格式）

例如：

输入：http://data.library.sh.cn/jp/data/明洪武2年.json?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"result": {

"data": "1369",

"uri": "http://data.library.sh.cn/authority/temporal/3rwxdjxxfz5bhff9"

}

}

输入方式3：

[参数1]：公元年。

[参数2]：用户的APIKey

输出：朝代纪年的JSON-LD数据（数组格式）

例如：

输入：http://data.library.sh.cn/jp/data/1369.json?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"result": {

"data": "明,元至正29年,明2年,明洪武2年",

"uri": "http://data.library.sh.cn/authority/temporal/p77tfazo3es795ad"

}

}

输入方式4：

[参数1]：朝代。

[参数2]：用户的APIKey

输出：朝代纪年的JSON-LD数据（数组格式）

例如：

输入：http://data.library.sh.cn/jp/temporal/秦.json?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"result": [

{

"monarch": "",

"reignTitle": "",

"monarchName": "",

"label": "秦",

"uri": "http://data.library.sh.cn/authority/temporal/qs36fnjpp26jtw8s",

"dynasty": "秦",

"end": "-206",

"begin": "-221"

},

{

"monarch": "始皇帝",

"reignTitle": "",

"monarchName": "嬴政",

"label": "秦",

"uri": "http://data.library.sh.cn/authority/temporal/ty2nvzpe6atov9rz",

"dynasty": "秦",

"end": "-210",

"begin": "-221"

},

{

"monarch": "二世",

"reignTitle": "",

"monarchName": "嬴胡亥",

"label": "秦",

"uri": "http://data.library.sh.cn/authority/temporal/r3iiwqb8wltcenxp",

"dynasty": "秦",

"end": "-207",

"begin": "-209"

},

{

"monarch": "",

"reignTitle": "",

"monarchName": "嬴子婴",

"label": "秦",

"uri": "http://data.library.sh.cn/authority/temporal/k823rq1gc2a82h98",

"dynasty": "秦",

"end": "-206",

"begin": "-206"

},

{

"monarch": "",

"reignTitle": "秦兴",

"monarchName": "薛举",

"label": "秦秦兴",

"uri": "http://data.library.sh.cn/authority/temporal/41ngdn53nn6owyqr",

"dynasty": "秦",

"end": "618",

"begin": "617"

}

]

}

输入方式5：

[参数1]：temporal.json

[参数2]：用户的APIKey

输出：朝代纪年的JSON-LD数据（数组格式）

例如：

输入：http://data.library.sh.cn/jp/temporal.json?key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"data": [

{

"uri": "http://data.library.sh.cn/authority/temporal/4alljneqiivh5691",

"label": "夏",

"end": "",

"begin": "-1989"

},

{

"uri": "http://data.library.sh.cn/authority/temporal/5et552ry5g8t8t1m",

"label": "商",

"end": "",

"begin": "-1559"

},

...

]

}

返回属性

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 说明 |
| label | literial | 朝代名称 |
| begin | literial | 朝代开始时间 |
| end | literial | 朝代截止时间 |
| dynasty | literial | 朝代 |
| monarch | literial | 帝王 |
| monarchName | literial | 帝王姓名 |
| reignTitle | literial | 年号 |

1. 书目数据

功能：输入题名、责任者、姓氏、先祖名人姓名、谱籍地名、堂号、馆藏机构、摘要中的关键词的任意组合，返回所有匹配的家谱数据。数据匹配方式为模糊匹配。

API接口：

http://data.library.sh.cn/jp/work/data?[参数1]&key=[参数2]

输入：

[参数1]：详见\*参数1表。

[参数2]：用户的APIKey

输出：家谱的JSON-LD数据（数组格式）

\*参数1表

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 说明 |
| title | string | 标题 |
| creator | string | 责任者 |
| familyName | string | 姓氏 |
| place | string | 谱籍地 |
| titleOfA | string | 堂号 |
| org | string | 收藏机构 |
| person | string | 先祖名人姓名 |
| des | string | 摘要 |

输入：http://data.library.sh.cn/jp/work/data?title=侯氏家乘不分卷&key=02cdb77b436d4dc383f1b64exxxxxxxxx

输出：

{

"@graph": [

{

"@id": "http://data.library.sh.cn/jp/resource/instance/mnuejsn18xgdjtfu",

"@type": "http://bibframe.org/vocab/Instance",

"category": "http://data.library.sh.cn/vocab/binding/xian-zhuang",

"edition": "http://data.library.sh.cn/vocab/edition/ke-ben",

"extent": "三册",

"instanceOf": "http://data.library.sh.cn/jp/resource/work/n2sl8uxa2dkjxrjs",

"temporal": "http://data.library.sh.cn/authority/temporal/7ase6ple2nud826q",

"temporal:Value": [

1788,

"清乾隆五十三年（1788）"

]

},

{

"@id": "http://data.library.sh.cn/jp/resource/item/9one9teelogwdu2t",

"@type": "http://bibframe.org/vocab/Item",

"heldBy": "http://data.library.sh.cn/entity/organization/brvqlrg8y55v1b5q",

"itemOf": "http://data.library.sh.cn/jp/resource/instance/mnuejsn18xgdjtfu",

"shelfMark": "XP2191-2193",

"description": "STJP016852"

},

{

"@id": "http://data.library.sh.cn/jp/resource/item/hsp9axrzbfa3urs2",

"@type": "http://bibframe.org/vocab/Item",

"heldBy": "http://data.library.sh.cn/entity/organization/u7rvfi69xguxwc5f",

"itemOf": "http://data.library.sh.cn/jp/resource/instance/mnuejsn18xgdjtfu"

},

{

"@id": "http://data.library.sh.cn/jp/resource/item/xms8ymr24lsl1yij",

"@type": "http://bibframe.org/vocab/Item",

"heldBy": "http://data.library.sh.cn/entity/organization/uiewno27vhzgoadn",

"itemOf": "http://data.library.sh.cn/jp/resource/instance/mnuejsn18xgdjtfu"

},

{

"@id": "http://data.library.sh.cn/jp/resource/work/n2sl8uxa2dkjxrjs",

"@type": "http://bibframe.org/vocab/Work",

"creator": "http://data.library.sh.cn/jp/entity/person/nk8l1rk65ilq428q",

"identifiedBy": "2340035",

"subject": "http://data.library.sh.cn/authority/familyname/fkffey53a7wazp5e",

"http://bibframe.org/vocab/title": {

"@id": "http://data.library.sh.cn/jp/authority/title/cheyaod9ij75fb36"

},

"title": [

{

"@language": "cht",

"@value": "侯氏家乘不分卷（河南省開封市杞縣）"

},

{

"@language": "chs",

"@value": "侯氏家乘不分卷（河南省开封市杞县）"

}

],

"description": "一世祖從義原籍安徽鳳阳府潁上县，繼迁河南開封，元末避亂轉徙杞县，卜居於鳴雁鄉焦刺苑，是为侯氏迁杞之始祖。谱载小序及始祖以下十三世之世系。书名据卷端、书签题。",

"place": "http://data.library.sh.cn/entity/place/n5sfke9d88qj3iyp"

}

],

"@context": {

"title": "http://purl.org/dc/elements/1.1/title",

"creator": {

"@id": "http://bibframe.org/vocab/creator",

"@type": "@id"

},

"subject": {

"@id": "http://bibframe.org/vocab/subject",

"@type": "@id"

},

"description": "http://www.library.sh.cn/ontology/description",

"place": {

"@id": "http://www.library.sh.cn/ontology/place",

"@type": "@id"

},

"identifiedBy": "http://bibframe.org/vocab/identifiedBy",

"edition": {

"@id": "http://bibframe.org/vocab/edition",

"@type": "@id"

},

"instanceOf": {

"@id": "http://bibframe.org/vocab/instanceOf",

"@type": "@id"

},

"temporalValue": {

"@id": "http://www.library.sh.cn/ontology/temporalValue",

"@type": "http://www.w3.org/2001/XMLSchema#integer"

},

"extent": "http://bibframe.org/vocab/extent",

"temporal": {

"@id": "http://www.library.sh.cn/ontology/temporal",

"@type": "@id"

},

"category": {

"@id": "http://bibframe.org/vocab/category",

"@type": "@id"

},

"itemOf": {

"@id": "http://bibframe.org/vocab/itemOf",

"@type": "@id"

},

"shelfMark": "http://bibframe.org/vocab/shelfMark",

"heldBy": {

"@id": "http://bibframe.org/vocab/heldBy",

"@type": "@id"

}

}

}

返回属性说明：

|  |  |  |
| --- | --- | --- |
| 属性 | 类型 | 说明 |
| 作品（graph="http://data.library.sh.cn/jp/resource/work/") | | |
| title | literial | value：正书名 language：语言 “chs”：中文简体 “cht”：中文繁体 |
| http://bibframe.org/vocab/title | URI | 题名URI |
| creator | URI | 责任者URI |
| contributor | URI | 其他责任者URI |
| subject | URI | 姓氏URI ("http://data.library.sh.cn/authority/familyname/") 或 堂号URI ("http://data.library.sh.cn/authority/titleofancestraltemple/") |
| place | URI | 谱籍地URI |
| description | literial | 摘要 |
| 实例（graph="http://data.library.sh.cn/jp/resource/instance/") | | |
| category | URI | 分类URI（平装、线装、精装...） |
| edition | URI | 版本URI（抄本、刻本...） |
| extent | literial | 数量 |
| temporal | URI | 出版年代URI |
| temporal:Value | literial | 出版年代描述 |
| instanceOf | URI | 书目URI |
| 单件（graph="http://data.library.sh.cn/jp/resource/item/") \*一个作品可能关联多个单件 | | |
| heldBy | URI | 馆藏机构URI |
| itemOf | URI | 版本信息URI |
| shelfMark | literial | 索书号 |
| description | literial | DOI |

1. 通过Sparql Endpoint获取数据

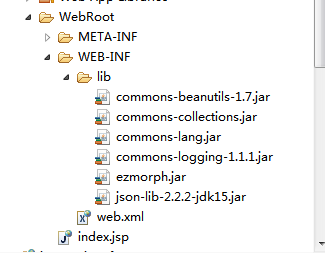
地址：http://data.library.sh.cn:8890/sparql

输入：SPARQL查询语句

输出：HTML, RDF/XML, JSON, Turtle等

三，JSON-LD解析

1. JSON （JavaScript Object Notation）一种简单的数据格式，比xml更轻巧。  
    Json建构于两种结构：  
        1、“名称/值”对的集合（A collection of name/value pairs）。不同的语言中，它被理解为对象（object），纪录（record），结构（struct），字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。 如：       
           {  
               “name”:”jackson”,  
               “age”:100  
            }  
     
     
       2、值的有序列表（An ordered list of values）。在大部分语言中，它被理解为数组（array）如：  
        {  
           “students”:  
           [  
               {“name”:”jackson”,“age”:100},  
               {“name”:”michael”,”age”:51}  
           ]  
        }  
   **二、解析JSON步骤（以JAVA 为例）**  
       **A、服务器端将数据转换成json字符串**  
         首先、服务器端项目要导入json的jar包和json所依赖的jar包至builtPath路径下（这些可以到JSON-lib官网下载：<http://json-lib.sourceforge.net/>）  
      
      
       然后将数据转为json字符串，核心函数是：  
    public static String createJsonString(String key, Object value)  
       {  
           JSONObject jsonObject = new JSONObject();  
           jsonObject.put(key, value);  
           return jsonObject.toString();  
       }  
   **B、客户端将json字符串转换为相应的javaBean**  
   **1、客户端获取json字符串**



public class HttpUtil  
{  
     
    public static String getJsonContent(String urlStr)  
    {  
        try  
        {// 获取HttpURLConnection连接对象  
            URL url = new URL(urlStr);  
            HttpURLConnection httpConn = (HttpURLConnection) url  
                    .openConnection();  
            // 设置连接属性  
            httpConn.setConnectTimeout(3000);  
            httpConn.setDoInput(true);  
            httpConn.setRequestMethod("GET");  
            // 获取相应码  
            int respCode = httpConn.getResponseCode();  
            if (respCode == 200)  
            {  
                return ConvertStream2Json(httpConn.getInputStream());  
            }  
        }  
        catch (MalformedURLException e)  
        {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
        catch (IOException e)  
        {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
        return "";  
    }  
  
     
    private static String ConvertStream2Json(InputStream inputStream)  
    {  
        String jsonStr = "";  
        // ByteArrayOutputStream相当于内存输出流  
        ByteArrayOutputStream out = new ByteArrayOutputStream();  
        byte[] buffer = new byte[1024];  
        int len = 0;  
        // 将输入流转移到内存输出流中  
        try  
        {  
            while ((len = inputStream.read(buffer, 0, buffer.length)) != -1)  
            {  
                out.write(buffer, 0, len);  
            }  
            // 将内存流转换为字符串  
            jsonStr = new String(out.toByteArray());  
        }  
        catch (IOException e)  
        {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
        return jsonStr;  
    }  
}

**2、获取javaBean**  
    public static Person getPerson(String jsonStr)  
    {  
        Person person = new Person();  
        try  
        {// 将json字符串转换为json对象  
            JSONObject jsonObj = new JSONObject(jsonStr);  
            // 得到指定json key对象的value对象  
            JSONObject personObj = jsonObj.getJSONObject("person");  
            // 获取之对象的所有属性  
            person.setId(personObj.getInt("id"));  
            person.setName(personObj.getString("name"));  
            person.setAddress(personObj.getString("address"));  
        }  
        catch (JSONException e)  
        {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
  
        return person;  
    }  
  
    public static List<Person> getPersons(String jsonStr)  
    {  
        List<Person> list = new ArrayList<Person>();  
  
        JSONObject jsonObj;  
        try  
        {// 将json字符串转换为json对象  
            jsonObj = new JSONObject(jsonStr);  
            // 得到指定json key对象的value对象  
            JSONArray personList = jsonObj.getJSONArray("persons");  
            // 遍历jsonArray  
            for (int i = 0; i < personList.length(); i++)  
            {  
                // 获取每一个json对象  
                JSONObject jsonItem = personList.getJSONObject(i);  
                // 获取每一个json对象的值  
                Person person = new Person();  
                person.setId(jsonItem.getInt("id"));  
                person.setName(jsonItem.getString("name"));  
                person.setAddress(jsonItem.getString("address"));  
                list.add(person);  
            }  
        }  
        catch (JSONException e)  
        {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
  
        return list;  
    }