# Kotlin
# Example AsyncTasks

# on server side

HTL Villach
Future Inside

```java
// A POJO class using Lombok annotations instead boilerplate code.
@NoArgsConstructor  // NEVER forget for JSON and XML creations ;-)
@AllArgsConstructor
@Data
// JAX-RS supports automatic mapping form JAXB-annotated class to XML and JSON.
@XmlRootElement
public class Person {
    private int    ssnr;
    private String name;
    private ArrayList<String>  contacts;
    private ArrayList<Address> addresses;
}
```

```java
@NoArgsConstructor
@AllArgsConstructor
@Data
@XmlRootElement
public class Address {
    private int    zip;
    private String town;
    private String street;
}
```

```java
/** <<singleton>> DAO for a person object data model provider in form of an enumeration. */
public enum PersonDAO {
    instance;

    private Map<Integer, Person> contentProvider = new HashMap<Integer, Person>();

    private PersonDAO() {
        ArrayList<String> contacts = new ArrayList<String>();
                          contacts.add("phone 007");
                          contacts.add("email w(at)willy.com");
        ArrayList<Address> addresses = new ArrayList<Address>();
                           addresses.add(new Address(9500, "Villach", "Hauptplatz"));
                           addresses.add(new Address(9500, "Villaco", "Tschinowitscherweg 5"));
                           addresses.add(new Address(9020, "Klagenfurt", "Neuer Platz 1"));
        contentProvider.put(4711, new Person(4711, "Windel Willy",  contacts, addresses));
        contentProvider.put(4812, new Person(4812, "Kurt C. Hose",  contacts, null));
        contentProvider.put(4913, new Person(4913, "Iris Gleichen", null,     addresses));
        contentProvider.put(5014, new Person(5014, "Axel Schweiß",  null,     null));
    }

    public Map<Integer, Person> getModel() { return contentProvider; }
}
```

# on server side – cont'd.

```java
@Path("/persons")                          // maps the resource to the URL persons
public class PersonsResource {
      // Enables contextual objects, e.g. ServletContext, Request, Response, UriInfo
      @Context
      UriInfo uriInfo;
      @Context
      Request request;

      // GET a list of persons for applications in form of a json string
      @GET
      @Produces(MediaType.APPLICATION_JSON)
      public List<Person> getPersons() {
        List<Person> persons = new ArrayList<Person>();
        persons.addAll(PersonDAO.instance.getModel().values());
        return persons;
      }
```

192.168.192.22:8080/WebServiceSimple/App/persons

JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON


```
0:
  addresses:
    0:
      street:   "Hauptplatz"
      town:     "Villach"
      zip:      9500
    1:
      street:   "Tschinowitscherweg 5"
```


localhost:8080/WebServiceSimple/App/persons

JSON Raw Data Headers

Save Copy


```
0:
  addresses:
    0:
      street:   "Hauptplatz"
      town:     "Villach"
      zip:      9500
    1:
      street:   "Tschinowitscherweg 5"
      town:     "Villaco"
      zip:      9500
    2:
      street:   "Neuer Platz 1"
      town:     "Klagenfurt"
      zip:      9020
  name:         "Iris Gleichen"
  ssnr:         4913
1:
  name:         "Axel Schweiß"
  ssnr:         5014
2:
```

# Kotlin Data Classes

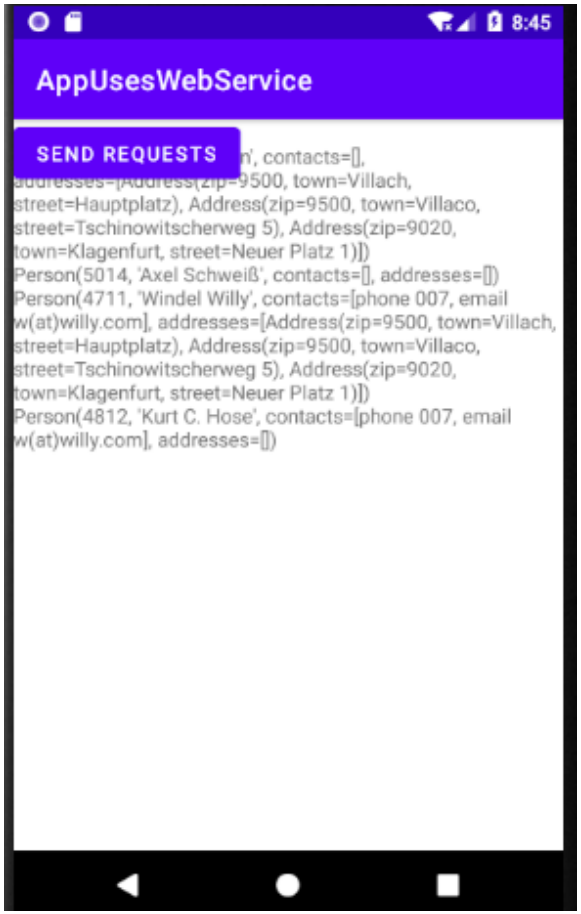
HTL Villach
Future Inside

```
package data.com.examples.che.data

import com.examples.che.data.Address

class Person (val ssnr: Int, var name: String,
              var contacts: ArrayList<String>, var addresses: ArrayList<Address>) {

    override fun toString(): String {
        return "Person($ssnr, '$name', contacts=$contacts, addresses=$addresses)"
    }
}
```

```
package com.examples.che.data

data class Address (val zip: Int, val town: String, val street: String)
```

software inside

# Kotlin Activity




```kotlin
fun onClickGetData(view: View) {
    // val url = "http://192.168.192.22:8080/WebServiceSimple/App/persons"
    val url = binding.tvWebService.text.toString()
    val get = GetPersonsAsyncTask().execute(url)    // and task.cancel() to stop it

    val n    = Random.nextInt()
    val p    = Person(n, name: "name" +n, ArrayList<String>(), ArrayList<Address>())
    val post = PostAsyncTask().execute(url, Gson().toJson(p))
}
```

```kotlin
dependencies {
    implementation 'com.google.code.gson:gson:2.8.5'
```

software inside

# AsyncTask to read data

```kotlin
private inner class GetPersonsAsyncTask : AsyncTask<String, String, String>() {

    override fun doInBackground(vararg urls: String?): String {
        var c: HttpURLConnection? = null
        try {
            val url = URL(urls[0])
            c = url.openConnection() as HttpURLConnection
            c.connectTimeout = CONNECTON_TIMEOUT_MILLISECONDS
            c.readTimeout    = CONNECTON_TIMEOUT_MILLISECONDS
            publishProgress(streamToString(c.inputStream))
        } catch (ex: Exception) {
            Log.i( tag: "ERR", ex.localizedMessage)
        } finally {
            if (c != null) c.disconnect()
        }
        return " "
    }

    override fun onPreExecute() { }
    override fun onPostExecute(result: String?) { }
```

```kotlin
    override fun onProgressUpdate(vararg values: String?) {
        try {
            var a = JSONArray(values[0])
            var s = ""
            for (i in 0 until a.length()) {
                val o = a.getJSONObject(i)
                val p = Person(o.getInt( name: "ssnr"), o.getString( name: "name"),
                               toStringList(o,  item: "contacts"), getAddresses(o))
                Log.i(TAG,  msg: "at " +i +": " +p)
                s += "\n" + p.toString()
            }
            binding.tvMessage.text = s
        } catch (e: Exception) {
            binding.tvMessage.text = e.localizedMessage
        }
    }

    private fun getAddresses(o: JSONObject) : ArrayList<Address> {
        val arr = toJsonArray(o,  item: "addresses")
        var adr = ArrayList<Address>()
        for (i in 0 until arr.length()) {
            val o      = arr.getJSONObject(i)
            val street = o.getString( name: "street")
            val town   = o.getString( name: "town")
            val zip    = o.getInt( name: "zip")
            adr.add(Address(zip, town, street))
        }
        return adr
    }
```

# Miscellaneous functions

HTL Villach
Future Inside

```
// extracts an json array from a json object by a given item which can rise an exception
fun toJsonArray(o: JSONObject, item: String) : JSONArray {
    var a : JSONArray
    try {
        a = o.getJSONArray(item)
    } catch (e: Exception) {
        a = JSONArray()
    }
    return a
}

// get a string list by a given item from a json object which can rise an exce
fun toStringList(o: JSONObject, item: String) : ArrayList<String> {
    val a = toJsonArray(o, item)
    var l = ArrayList<String>()
    for (i in 0 until a.length()) {
        try {
            l.add(a.getString(i))
        } catch (e: Exception) {
            l.add(e.localizedMessage)
        }
    }
    return l
}
```

```
fun streamToString(inputStream: InputStream): String {
    val bufferReader = BufferedReader(InputStreamReader(inputStream))
    var line: String
    var result = ""

    try {
        do {
            line = bufferReader.readLine()
            if (line != null) {
                result += line
            }
        } while (line != null)
        inputStream.close()
    } catch (ex: Exception) {
        result += ex.localizedMessage
    }
    return result
}
```

software inside

# AsyncTask to insert Data

```
private inner class PostAsyncTask() : AsyncTask<String, String, Boolean>() {

    override fun doInBackground(vararg urls: String?): Boolean {
        var c: HttpURLConnection? = null
        var ok = false
        try {
            val url = URL(urls[0])
            c = url.openConnection() as HttpURLConnection
            c.connectTimeout = CONNECTON_TIMEOUT_MILLISECONDS
            c.readTimeout    = CONNECTON_TIMEOUT_MILLISECONDS
            c.setRequestProperty("Content-Type", "application/json")
            c.requestMethod = "POST"
            val wr = OutputStreamWriter(c.outputStream)
            wr.write(urls[1])   // json string as second param ;-)
            wr.flush()
            ok = (c.responseCode == 200)
        } catch (ex: Exception) {
            Log.i( tag: "ERR", ex.localizedMessage)
        } finally {
            if (c != null) c.disconnect()
        }
        return ok
    }
```