HTTP Communication в Tizen (Tizen Ver.2.3)

Весна 2015 г.

Ким Су Донг, ***доктор философии.***
Профессор кафедры информатики
Лаборатория программирования
Университет Сунгсил

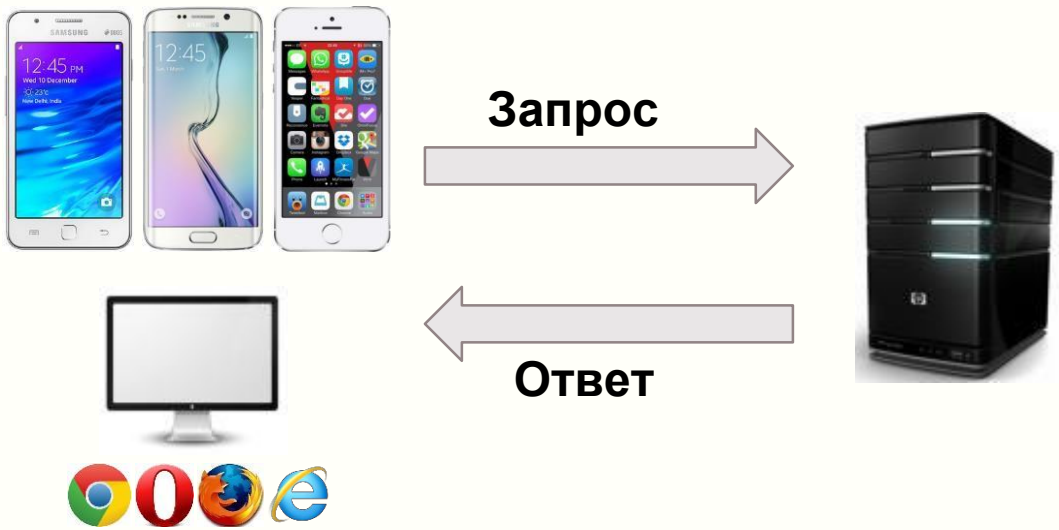
Office 02-820-0909 Mobile 010-7392-2220

sdkim777@gmail.com http://soft.ssu.ac.kr

Часть 7-1.
Обзор HTTP-коммуникаций

Что такое HTTP-комуникация?

- **Протокол приложений для распределенных, совместных, гипермедиа информационных сетей**
- **Основа обмена данными в World Wide Web**



Методы HTTP

- **GET**
  - **Для запроса представления заданного ресурса**
- **POST**
  - **для размещения данных для обработки на заданном ресурсе**
  - **для представления файлов на сервер**
- **PUT**
  - **для загрузки представления заданного ресурса**
- **DELETE**
  - **для удаления заданного ресурса**

REST (Representational State Transfer)

- Стиль архитектуры ПО, состоящий из правил и наилучших примеров по созданию масштабируемых веб-сервисов
- Для коммуникации в протоколе HTTP (Hypertext transfer Protocol) посредством тех же терминов HTTP
  - GET
  - POST
  - PUT
  - DELETE

Использование методов HTTP в REST

- **При работе с данными использованы как функции CRUD**
  - **Относятся ко всем основным функциям работы с данными**

Операция	HTTP
Создать	POST
Вернуть	GET
Обновить	PUT
Удалить	DELETE

Часть 7-2. HTTP на веб-приложении Tizen (Tizen v. 2.3)

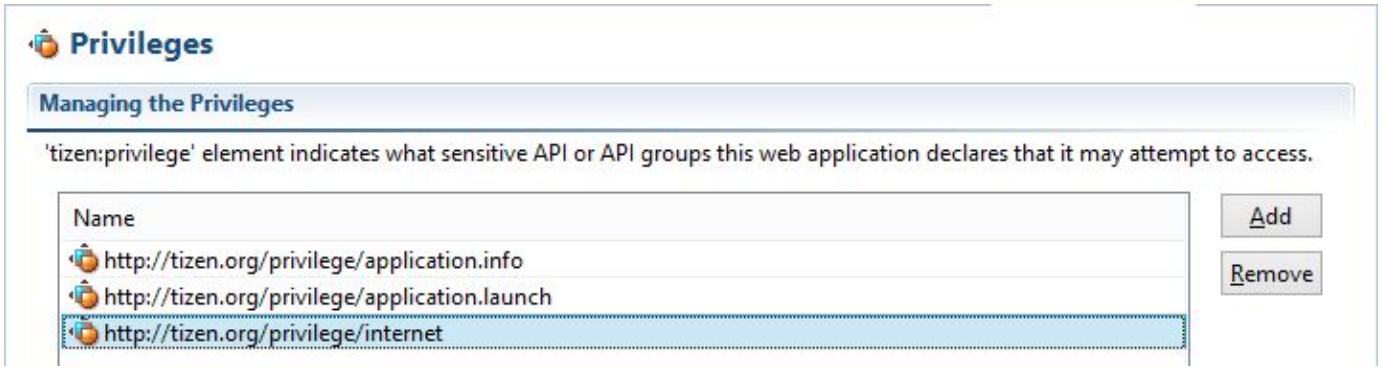
Два пути для HTTP-коммуникации

- **Синхронная коммуникация**
  - Использовать тэги Form
- **Асинхронная коммуникация**
  - Использовать Ajax
  - Чаще всего используется в веб-приложении Tizen

Настройка привилегий

- **Разрешить использовать определенные функции**
- **Добавить привилегию, http://tizen.org/privilege/internet**

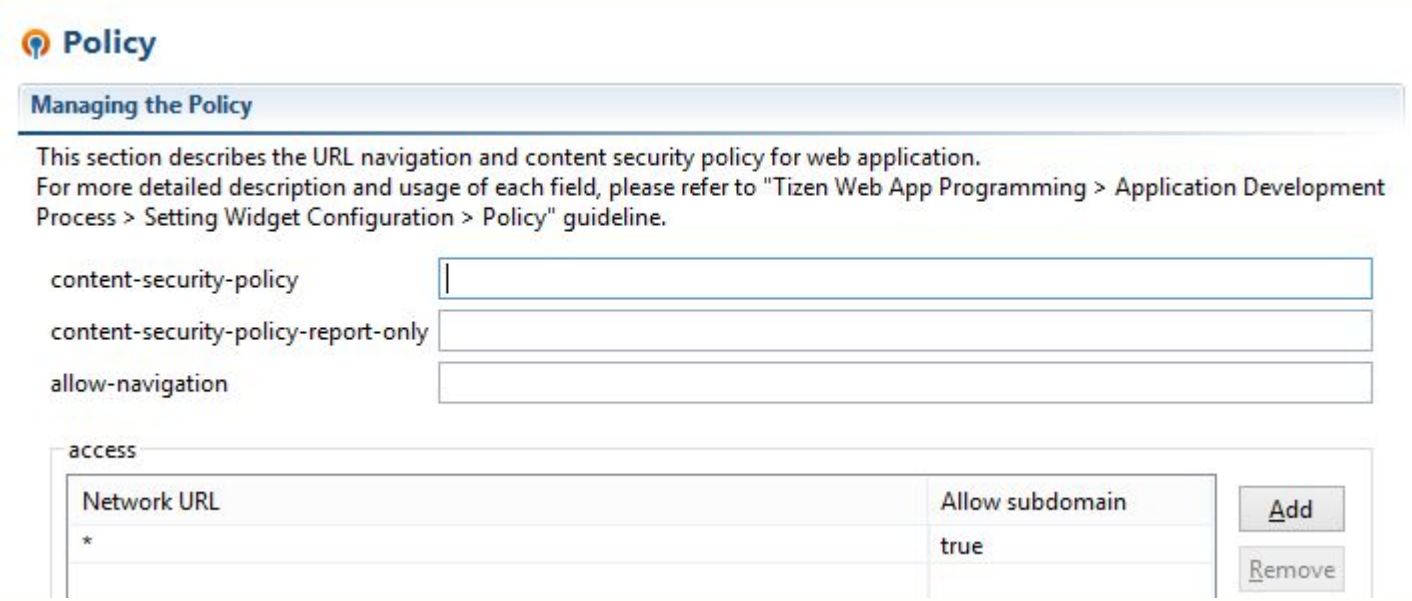
config.xml > Privileges > Add



Настройки политики

- **Разрешить веб-приложению Tizen доступ к данному URL**
- **Добавить политику:**
  - **Сетевой URL: ***
  - **Разрешить субдомен: True**

config.xml > Policy > Add



Тэг Form

- Используется для создания веб-формы HTML для ввода информации пользователем
- Для перезагрузки других веб-страниц
- Ключевые атрибуты
  - Действие
    - Строка, содержащая URL, куда отправляется запрос
  - Метод
    - Метод HTTP для использования в запросе
- Использование

```
<formaction="demo_form.asp" method="get">
  name: <input type="text" name="Gildong Hong"><br>
  <input type="submit" value="Submit">
</form>
```

AJAX (Асинхронный JavaScript и XML)

- Группа взаимосвязанных технологий веб-разработки, используемых на стороне клиента для создания асинхронных веб-приложений
- Это не новый язык программирования, а новый способ применения существующих стандартов
- Для обмена данными с сервером и обновления частей веб-станицы без перезагрузки всей страницы целиком

jQuery Methods for Ajax

- jQuery.ajax()
  - для выполнения асинхронного HTTP-запроса (Ajax)
- jQuery.get()
  - для загрузки данных с сервера по HTTP-запросу GET
- jQuery.post()
  - для загрузки данных с сервера по HTTP-запросу POST

jQuery.ajax() (1)

- jQuery.ajax(url [, settings])
  - Url
    - строка, содержащая URL, на который отправляется запрос
  - Settings
    - Набор пар ключ/значение, настраивающих запрос Ajax

jQuery.ajax() (2)

- **Ключи настройки**
  - Method
    - Метод HTTP для использования в запросе
    - GET, POST, PUT
  - Data
    - Данные для передачи на сервер
  - Datatype
    - Тип данных, которые ожидается получить от сервера
    - XML, JSON, HTML
  - Success
    - Функция, вызываемая при успешном запросе
  - Error
    - Функция, вызываемая при неудаче запроса

jQuery.ajax() (3)

- **Пример**

```
$("#bt-ajax").click(function() {
    $.ajax({
        url : "http://example.com",  method : "GET",  crossDomain :
        true,  dataType : "json",
        success : function(res) {
            $("#p-ajax-res").html(JSON.stringify(res));
        },
        error : function(err) {
            $("#p-ajax-res").html(JSON.stringify(err));
        }
    });
});
```

jQuery.get() (1)

- jQuery.get(url [, Settings])
  - Url
    - Строка, содержащая URL, на который отправляется запрос
  - Settings
    - Набор пар ключ/значение, настраивающий запрос Ajax
- Примерная функция Ajax

```
$.get({  url: url,
  data: data,  success:
  success,
  dataType: dataType
});
```

=

```
$.ajax({  url: url,
  method: 'GET'  data:
  data,  success: success,
  dataType: dataType
});
```

jQuery.get() (2)

- **Ключи настройки**
  - Data
    - Простой объект или переменная, которая отправляется на сервер
      с запросом
  - Success
    - Функция, вызываемая при успешном запросе
  - DataType
    - Функция, вызываемая при неудаче запроса

jQuery.post() (1)

- jQuery.post(url [, Settings])
  - Url
    - Строка, содержащая URL, на который отправляется запрос
  - Settings
    - Набор пар ключ/значение, настраивающий запрос Ajax
- Примерная функция Ajax

```
$.post({  url: url,
  data: data,  success:
  success,
  dataType: dataType
});
```

=

```
$.ajax({  url: url,
  method: 'POST'  data:
  data,  success: success,
  dataType: dataType
});
```

jQuery.post() (2)

- **Ключи настройки**
  - Data
    - простой объект или переменная, которая отправляется на сервер
      с запросом
  - Success
    - Функция, вызываемая при успешном запросе
  - DataType
    - Функция, вызываемая при неудаче запроса

Часть 7-3. HTTP на встроенном приложении Tizen (Tizen v.2.3)

Предварительные условия

- **Требуемый заголовок**
  - curl.h
- **Требуемые привилегии**
  - http://tizen.org/privilege/internet

Ключевые функции

- curl_easy_init()
- curl_slist_append()
- curl_easy_setopt ()
- curl_easy_perform ()
- curl_formadd ()
- curl_easy_cleanup()

curl_easy_init()

- **Возвратить обработчик, используемый как ввод**
  для других функций в интерфейсе Curl

- **Возврат**
  - **Ссылка CURL в случае успеха, иначе — значение NULL**

curl_slist_append()

- **Для добавления указанной переменной к ссылаемому списку переменных**
  - **Чтобы создать заголовочную информацию для HTTP-запроса**
  - **Чтобы создать команды для FTP-запроса**
- **Параметры**

Тип	Имя	Описание
struct curl_slist*	list	Указатель на связанный со строкой список
const char*	string	Значение строки, присвоенное связанному списку

- **Возврат**
  - **Новый указатель списка, иначе — значение NULL**

curl_easy_setopt()

- **Настроить опции для обработчика Curl**
- **Параметры**

Тип	Имя	Описание
CURL *	handle	Обработчик Curl
CURLoption	option	Опция задания поведения Curl
void	parameter	Параметр

- **Ключевые опции**

Option Name	Описание
CURLOPT_URL	Определение URL
CURLOPT_WRITEFUNCTION	Определение обратного вызова для полученных данных
CURLOPT_WRITEDATA	Определение указателя на данные для передачи обратному вызову на запись
CURLOPT_POST	Определение HTTP-метода как POST

- **Возврат**
  - **0 при успехе, иначе отличные от нуля значения**

curl_easy_perform ()

- **Для передачи данных, как описано в опциях**
- **Параметры**

Тип	Имя	Описание
CURL *	handle	Обработчик Curl

- **Возврат**
  - **0 при успехе, иначе — ненулевое значение**

curl_formadd ()

- **Для добавления разделов при создании multipart/formdata HTTP POST**
- **Параметры**

Тип	Имя	Описание
Curl_httppost *	firstitem	Обработчик Curl
Curl_httppost	lastitem	Опция для задания поведения Curl
…	…	Не определено

- **Возврат**
  - **0 при успехе, иначе — ненулевое значение**
- **Использование**

```
/*<input type='file', name='sendfile'>*/

curl_formadd(&formpost, &lastptr, CURLFORM_COPYNAME,
  "sendfile", CURLFORM_FILE, "postit2.c", CURLFORM_END)
```

curl_easy_cleanup()

- **Чтобы завершить обработчик libcurl**
- **Параметры**

Тип	Имя	Описание
CURL *	handle	Обработчик Curl

- **Возврат**
  - **Нет**

Пример (1)

- **Чтобы вернуть JSON по HTTP-запросу**
  - **Структура для данных ответа**

```
typedef struct curlResponseData {
  char *response; //Данные отклика
  size_t size;        //Объем данных
} CURL_RES_DATA;
```

  - **Функция инициализации структуры**

```
void init_curl_responseData(CURL_RES_DATA *res) {  res->size = 0;
  res->response = malloc(res->size + 1);

  if (res->response == NULL) {
    dlog_print(DLOG_INFO, "CALL_HTTP_REQUEST", "Allocation failed");
  }

  res->response[0] = '\0';
}
```

Пример (2)

- **Чтобы вернуть JSON по запросу HTTP (продолжение)**
  - **Функция для записи полученных данных в структуру**

```
size_t curl_write_function(void *ptr, size_t size, size_t nmemb,  CURL_RES_DATA *res) {
  size_t newLen = res->size + size * nmemb;
  res->response = realloc(res->response, newLen + 1);  if (res->response == NULL) {
    dlog_print(DLOG_INFO, "CALL_HTTP_REQUEST", "Allocation failed");
  }

  //Копируем данные отклика в структуру
  memcpy(res->response + res->size, ptr, size * nmemb);  res->response[newLen] = '\0';
  res->size = newLen;

  return size * nmemb;
}
```

Пример (3)

- **Чтобы вернуть JSON по запросу HTTP (продолжение)**
  - **Чтобы запросить запрос HTTP**

```
void call_http_request() {  CURLcode res;  CURL_RES_DATA
  res_data;  char* jsonUrl=
"http://www.w3schools.com/json/myTutorials.txt";
  init_curl_responseData(&res_data);

  // Инициализация
  // обработчика Curl
  curl = curl_easy_init();  if (curl) {

    // Инициализация заголовка HTTP
    struct curl_slist *headers = NULL;  curl_slist_append(headers,
    "Accept: application/json");  curl_slist_append(headers,
    "Content-Type:
application/json");
...
```

```
    // Задание опций
    curl_easy_setopt(curl, CURLOPT_URL, jsonUrl);  curl_easy_setopt(curl,
    CURLOPT_HTTPHEADER, headers);  curl_easy_setopt(curl,
    CURLOPT_WRITEFUNCTION,
curl_write_function);
    curl_easy_setopt(curl, CURLOPT_WRITEDATA, &res_data);

  // Отправка HTTP-запроса curl_easy_perform(curl);  free(res_data.
  response);  curl_easy_cleanup(curl);
  }
}
```