



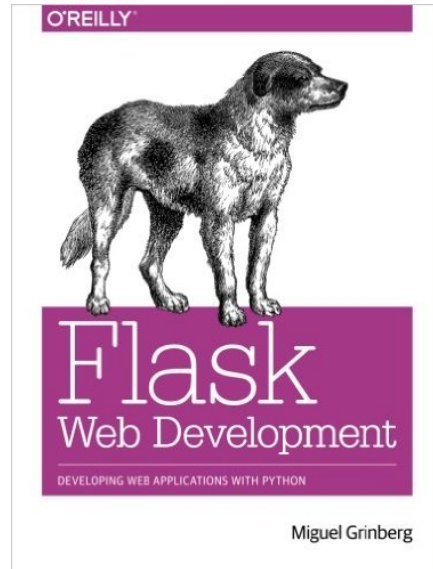
## Flask V: Ajax

**Harry J. Wang, Ph.D.**

University of Delaware

# Resources

- Flask Web Development: Developing Web Applications with Python by Miguel Grinberg (1st Edition)
- Chapter 5 and 14
- Use ``git checkout v5`` to see the delete code with Ajax
- Use ``git checkout v5.1`` to see the form posting code with Ajax



- Sample Course Application: SongBase:  
<https://github.com/udmis/songbase>

# Ajax: Asynchronous JavaScript And XML

- So far, we have been using forms to POST data to the server, which reloads the current page or redirects to another page.
- AJAX enables sending/retrieving data to/from the server asynchronously (in the background) without interfering with the display and behavior of the existing page.
- JSON (JavaScript Object Notation) is often the data format used for AJAX. See <https://en.wikipedia.org/wiki/JSON>
- AJAX is often used to interact with APIs. See Chapter 14 (API design)

# Ajax via jQuery

- Warning: jQuery slim version does not support \$.ajax() – the full version must be used!
- By default, Bootstrap uses the slim version of jQuery, which must be changed.

```
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js">
```



The full version must be used!

```
<!-- jQuery full version first, then Popper.js, then Bootstrap JS -->  
<script src="https://code.jquery.com/jquery-3.2.1.js"></script>
```

# Steps for Using Ajax (with jQuery and Flask)


1. Import jQuery full version
2. Backend: write a new route/api for the Ajax call in Flask, e.g., an API for deleting a song
3. Frontend: In the HTML file, use `$.ajax()` to issue an Ajax call to the API and use jQuery to manipulate the page if needed after the call.

The next few slides use deleting a song as an example to illustrate Ajax

# Backend Example

```
from flask import Flask, jsonify
```

```
@app.route('/api/song/<int:id>', methods=['DELETE'])
def delete_ajax_song(id):
    song = Song.query.get_or_404(id)
    db.session.delete(song)
    db.session.commit()
    return jsonify({"id": str(song.id), "name": song.name})
```



APIs often return  
data in json  
format

# Frontend Example (song-all.html)



```
<button class="btn btn-warning delete_ajax_btn" data-song-id={{song.id}}>Delete Ajax</button>
```

Class for jQuery

Data attribute for jQuery

Button click event

Get data from the data attribute

```
$(".delete_ajax_btn").click(function(event) {  
    var song_id = $(this).data("song-id");
```

Once the Ajax call succeeds, i.e., the song is deleted, the corresponding row needs to be removed from the page

1	Yellow	2000	Coldplay	<button>View</button>	<button>Edit</button>	<button>Delete</button>	<button>Delete Ajax</button>
---	--------	------	----------	-----------------------	-----------------------	-------------------------	------------------------------

Find the row from the button click event – this refers to the button

```
// get the table row of this song  
var row = $(this).parent().parent();
```

Remove the row w/o reloading the page

```
$.ajax({  
    type: "DELETE",  
    url: "/api/song/" + song_id,  
    success: function(response) {  
        console.log(response);  
        row.remove(); // remove the table row
```

# Ajax Call Example (song-all.html)

- The following is the basic structure of a typical Ajax call

```
$.ajax({  
  type: "DELETE", // GET POST DELETE PUT  
  url: "/api/song/" + song_id, // API Route  
  success: function(response) {  
    // success callback function  
    // what to do after ajax call succeeds goes here  
  }  
})  
.fail(function(error) {  
  // fail callback function  
  // what to do after ajax call fails goes here  
});
```

- Different messages are returned to the user in the success/fail callback functions using SweetAlert library as explained next



```
<!-- SweetAlert JS -->  
<script src="https://cdnjs.cloudflare.com/ajax/libs/sweetalert/1.1.3/sweetalert.min.js"></script>
```

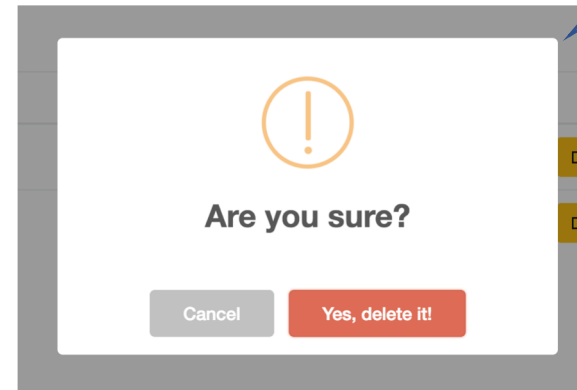
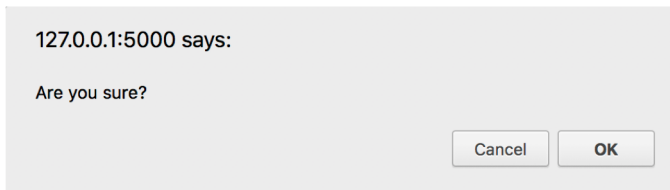
Include this in  
base template

# SweetAlert Example (song-all.html)

<https://sweetalert.js.org>

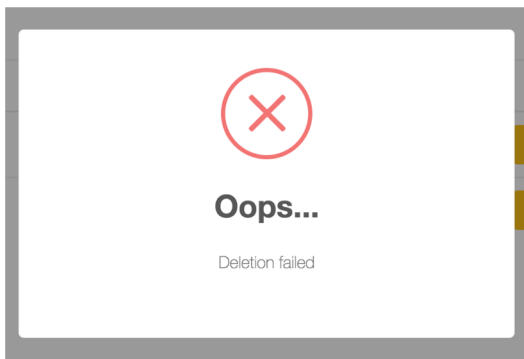
- SweetAlert is a useful JavaScript library to show pretty popup messages, e.g, default alert windows vs. SweetAlert window

Popup window  
with user  
confirmation



```
swal({  
  title: "Oops...",  
  text: "Deletion failed",  
  type: "error",  
  timer: 2000,  
  showConfirmButton: false  
});
```

Simple  
popup  
window



```
swal({  
  title: "Are you sure?",  
  type: "warning",  
  html: true,  
  showCancelButton: true,  
  confirmButtonColor: "#DD6B55",  
  confirmButtonText: "Yes, delete it!",  
  closeOnConfirm: false,  
  showLoaderOnConfirm: true  
},  
function(isConfirm) { // check whether user confirms or not  
  if (isConfirm) {  
    // ajax call code goes here  
  }  
});
```

# Ajax Post Example (artist-add.html)

## Add Artists:

Name

About



- Post Form using Ajax

- Prevent button default behavior:
- Form data serialization:

```
event.preventDefault(); // prevent the button to submit the form
```

```
$.ajax({  
  type: "POST",  
  url: "/api/artist/add",  
  data: $("#add-artist-form").serialize(),  
});
```

- Message flashing in Flask:

<http://flask.pocoo.org/docs/0.12/patterns/flashing>

```
# flash message type: success, info, warning, and danger from bootstrap  
flash('Artist Inserted', 'success')
```

- Toastr: a JS library for simple toast notifications

<https://codeseven.github.io/toastr/>

