Web and Database Computing

adelaide.edu.au

Server Architecture and Routes: Advanced HTTP Requests

GET and POST Requests

Different types of Requests

GET

- Used for requesting a resource from a Server.
- No request body
- Cannot send large amounts of data to server
- Can have URL parameters for small amounts of data

POST

- Used for sending data to a Server
- Contains a request body

Other types (HEAD, PUT, DELETE etc.) less commonly used in websites, but are used in REST APIs

GET request with parameters

- Parameters placed at end of URL
  - Consist of `name=value` pairs, preceeded by `?` and separated by `&`
- Must be a valid URL; may require encoding the names/values
- e.g. `/page.html?param1=value1&param2=more%20complex%2C%20fancy%20%26%20encoded%20value`

Sending a GET request with parameters

Through a Form Submit

- A HTML form, when submitted can be used to send a GET request with additional parameters.
- Browser will load response as a new page.

```
<form action="/page.html" method="get">
  <input type="text" name="q" value="Enter your search term here" />
  <input type="submit" value="Search" />
</form>
```

Creates

Enter your search term Search

Causes the browser to send a GET request for the page at

`/page.html?q=text`

Sending a GET request with parameters

Using AJAX

- Use string concatenation to construct the URL.
- May need to encode values before sending.

```
// Create new AJAX request
var xhttp = new XMLHttpRequest();

// Define behaviour for a response
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        // do stuff if successful
    }
};

// Initiate connection
xhttp.open("GET", "page.html?param1="+encodeURIComponent(value1)+"&param2="+encodeURIComponent(value2), true);

// Send request
xhttp.send();
```

Sending a GET request with parameters

As a link

- Since the parameters are contained in the URL, you can just provide a link.
- Browser will load link as a new page, so response should be a full HTML page.

Handling GET request parameters in Express

Use the request object

- `req.query` is automatically generated as an object who's keys are the request parameters.
- i.e. for a query parameter named `param1`, we can access its value from `req.query.param1`

Defined for us in **app.js**

```
...

app.use(express.urlencoded({ extended: false }));

...
```

Use in any of your routes e.g. **routes/index.js**

```
router.get('/page.html', function(req, res) {
    var q = req.query.param1;
    res.send('You searched for '+q);
});
```

POST Requests

POST Requests

- Can contain large amounts of data
- Held in request body

Sending a POST request

Through a Form Submit

- Similar to GET
- Search terms won't appear in URL
- Preferred method for passwords

```
<form action="/page.html" method="post">
  <input type="text" name="q" value="Enter your search term here" />
  <input type="submit" value="Search" />
</form>
```

Creates

Enter your search term Search

Causes the browser to send a POST request for `/page.html` with the form data in the request body.

Sending a POST request

Using AJAX

- Put content in `send()` call

```
// Create new AJAX request
var xhttp = new XMLHttpRequest();

// Define behaviour for a response
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        // do stuff if successful
    }
};

// Initiate connection
xhttp.open("POST", "page.html"), true);

// Send request
xhttp.send("Some string that is the body of my request");
```

Handling a POST request's body in Express

Use the request object

- `req.body` is automatically generated as a string containing the request body.

Use in any of your routes e.g. **routes/index.js**

```
router.post('/page.html', function(req, res) {
    var text = req.body;
    res.send('You sent '+text);
});
```

Handling different request body encodings

Express generator provides middleware to automatically handle different request encodings.

- JSON or form data requests will automatically be parsed into an object by express if your request is properly set up.

For the request body generated by a form submission

```
<form action="/page.html" method="post">
  <input type="text" name="q" value="Some text" />
  <input type="submit" value="Search" />
</form>
```

`req.body` contains the object parsed:

```
router.post('/page.html', function(req, res) {
    var p1 = req.body.q;       // will be "Some text"
    var p2 = req.body.submit;  // will be "Search"
    res.send();
});
```

Handling different request body encodings

Express generator provides middleware to automatically handle different request encodings.

- JSON or form data requests will automatically be parsed into an object by express if your request is properly set up.

For the JSON request body

```
{
  "param1": "value1",
  "param2": 2
}
```

`req.body` contains the object parsed:

```
router.post('/search.html', function(req, res) {
    var p1 = req.body.param1;  // will be "value1"
    var p2 = req.body.param2;  // will be 2
    res.send();
});
```

Sending JSON encoded POST requests

Express contains middleware to automatically handle different request encodings.

- JSON or form data requests will automatically be parsed **if your request is properly set up**.
- This can be done in AJAX by setting the request's `Content-type` header to `application/json`

```
// Create new AJAX request
var xhttp = new XMLHttpRequest();

// Define behaviour for a response
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        // do stuff if successful
    }
};

// Initiate connection
xhttp.open("POST", "search.html", true);

// Set content type to JSON
xhttp.setRequestHeader("Content-type", "application/json");

// Send request
xhttp.send(JSON.stringify({param1:"value1", param2:2}));
```

Dynamic URLs

Parameterised URLs in Express

Sometimes we want to have custom URL paths
e.g. URLs for each user like `/user/bob` and `/user/alice`

Express provides a method to do this using parameterised URL paths

- Use `:paramName` in the path
- Access via `req.params.paramName`

e.g.

```
router.get('/user/:id', function(req, res) {
    // The value of id is accessible as a variable
    res.send('user ' + req.params.id);
});

router.get('/author/:authorId/page/:pNum', function(req, res) {
    // A route can have multiple parameters in the URL path in different places
});
```

Summary

- GET requests do not have a body, while POST requests do.
- GET requests often have query parateters in the URL that can be accessed using `req.query`.
- Query parameters must be URL encoded.
- POST requests have a body that can be accessed using `req.body`.
- Express will automatically handle parse these requests to objects if the request is properly formatted.
- Express allows for dyanmic paths in URLs.

THE UNIVERSITY
of ADELAIDE