```
const bodyParser = require('body-parser');
const cors = require('cors');
const errorhandler = require('errorhandler');
const express = require('express');
const apiRouter = require('./api/api');
const app = express();
const PORT = process.env.PORT || 4000;

app.use(bodyParser.json());
app.use(cors());
app.use('/api', apiRouter);
app.use(errorhandler());

app.listen(PORT, () => {
  console.log('Listening on port: ' + PORT);
});

module.exports = app;

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

const express = require('express');
const apiRouter = express.Router();
const artistsRouter = require('./artists.js');
const seriesRouter = require('./series.js');

apiRouter.use('/artists', artistsRouter);
apiRouter.use('/series', seriesRouter);

module.exports = apiRouter;

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =


    :::::::            ::::::::
    ::::::: SOLUTION   ::::::::
    ::::::: SOLUTION   ::::::::
    :::::::            ::::::::



const express = require('express');
const seriesRouter = express.Router();

const sqlite3 = require('sqlite3');
const db = new sqlite3.Database(process.env.TEST_DATABASE ||
'./database.sqlite');

const issuesRouter = require('./issues.js');

seriesRouter.param('seriesId', (req, res, next, seriesId) => {
  const sql = 'SELECT * FROM Series WHERE Series.id = $seriesId';
  const values = {$seriesId: seriesId};
  db.get(sql, values, (error, series) => {
    if (error) {
      next(error);
```

```
    } else if (series) {
      req.series = series;
      next();
    } else {
      res.sendStatus(404);
    }
  });
});

seriesRouter.use('/:seriesId/issues', issuesRouter);
```

- GET
  - Returns a 200 response containing all saved series on the `series` property of the response body

```
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
seriesRouter.get('/', (req, res, next) => {
  db.all('SELECT * FROM Series', (err, series) => {
    if (err) {
      next(err);
    } else {
      res.status(200).json({series: series});
    }
  });
});
```

/api/series/:seriesId
- GET
  - Returns a 200 response containing the series with the supplied series ID on the `series` property of the response body
  - If a series with the supplied series ID doesn't exist, returns a 404 response

```
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
seriesRouter.get('/:seriesId', (req, res, next) => {
  res.status(200).json({series: req.series});
});
```

- POST
  - Creates a new series with the information from the `series` property of the request body and saves it to the database. Returns a 201 response with the newly-created series on the `series` property of the response body
  - If any required fields are missing, returns a 400 response

```
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
seriesRouter.post('/', (req, res, next) => {
  const name = req.body.series.name,
        description = req.body.series.description;
  if (!name || !description) {
    return res.sendStatus(400);
  }
```

```
  const sql = 'INSERT INTO Series (name, description) VALUES ($name,
$description)';
  const values = {
    $name: name,
    $description: description
  };

  db.run(sql, values, function(error) {
    if (error) {
      next(error);
    } else {
      db.get(`SELECT * FROM Series WHERE Series.id = ${this.lastID}`,
        (error, series) => {
          res.status(201).json({series: series});
        });
    }
  });
});
```

- PUT
  - Updates the series with the specified series ID using the information from the `series` property of the request body and saves it to the database. Returns a 200 response with the updated series on the `series` property of the response body
  - If any required fields are missing, returns a 400 response
  - If a series with the supplied series ID doesn't exist, returns a 404 response

```
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
seriesRouter.put('/:seriesId', (req, res, next) => {
  const name = req.body.series.name,
        description = req.body.series.description;
  if (!name || !description) {
    return res.sendStatus(400);
  }

  const sql = 'UPDATE Series SET name = $name, description =
$description ' +
      'WHERE Series.id = $seriesId';
  const values = {
    $name: name,
    $description: description,
    $seriesId: req.params.seriesId
  };

  db.run(sql, values, (error) => {
    if (error) {
      next(error);
    } else {
      db.get(`SELECT * FROM Series WHERE Series.id = $
{req.params.seriesId}`,
        (error, series) => {
          res.status(200).json({series: series});
        });
    }
```

```
  });
});

- DELETE
  - Deletes the series with the supplied series ID from the database
if that series has no related issues. Returns a 204 response.
  - If the series with the supplied series ID has related issues,
returns a 400 response.
  - If a series with the supplied series ID doesn't exist, returns a
404 response
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
    ::::    ::::    ::::    ::::
seriesRouter.delete('/:seriesId', (req, res, next) => {
  const issueSql = 'SELECT * FROM Issue WHERE Issue.series_id =
$seriesId';
  const issueValues = {$seriesId: req.params.seriesId};
  db.get(issueSql, issueValues, (error, issue) => {
    if (error) {
      next(error);
    } else if (issue) {
      res.sendStatus(400);
    } else {
      const deleteSql = 'DELETE FROM Series WHERE Series.id =
$seriesId';
      const deleteValues = {$seriesId: req.params.seriesId};

      db.run(deleteSql, deleteValues, (error) => {
        if (error) {
          next(error);
        } else {
          res.sendStatus(204);
        }
      });
    }
  });
});

module.exports = seriesRouter;
```