ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Professional NgRx

4 - Testing

End-to-End Tests

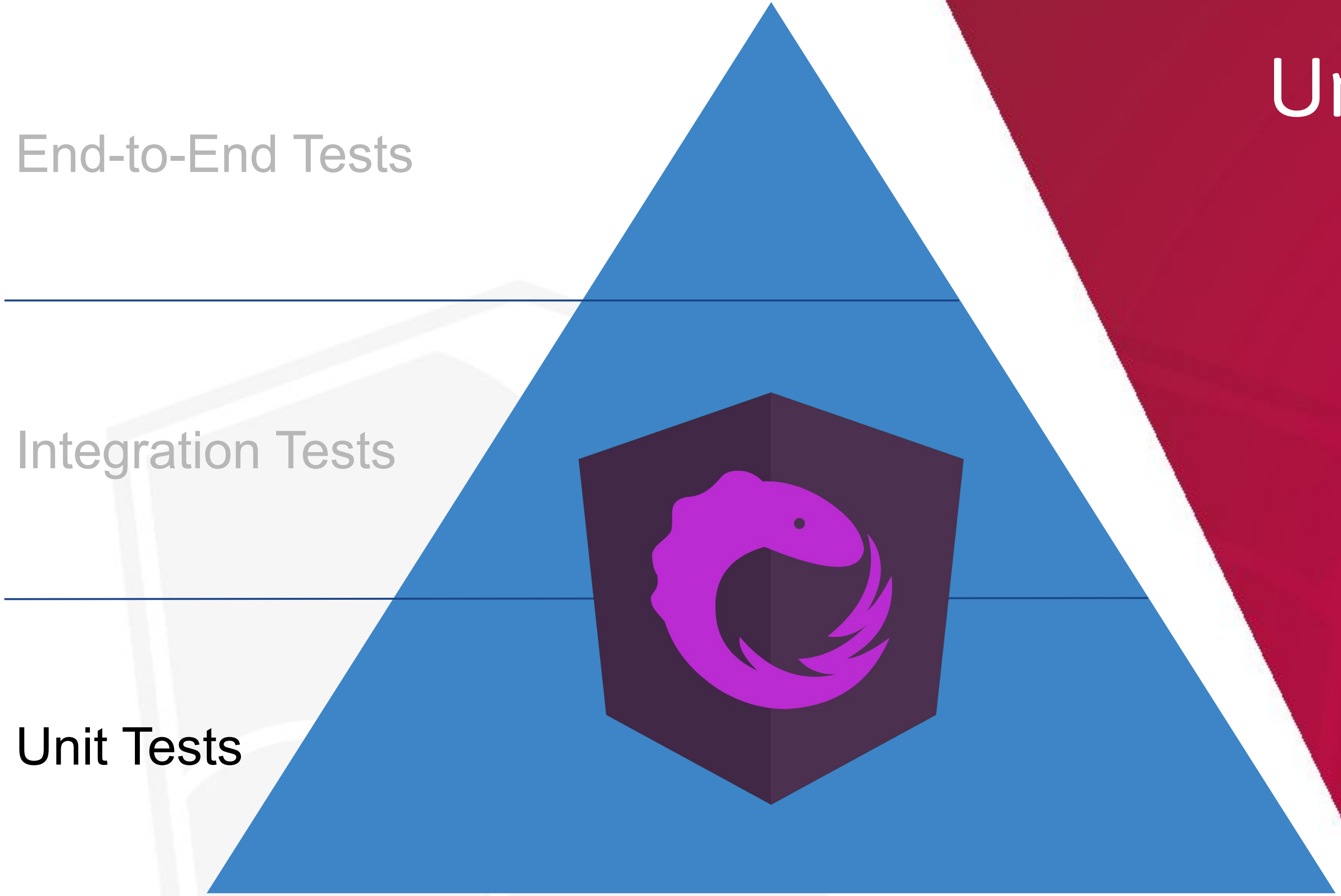
Integration Tests

Unit Tests

Testing Types

- Unit Tests
  - Reducer
  - Selectors
  - Effects
- Integration Tests
  - NgRx elements altogether (but isolated)
  - In combination with components/services
- NgRx as Dependency

Unit Tests



Reducer 1/2

```
export const holidaysFeature = createFeature({
  name: "holidays",
  reducer: createReducer(
    initialState,
    on(loaded, (state, { holidays }) => ({
      ...state,
      loadStatus: "loaded",
      holidays,
    }))
    // ...
  ),
});
```

Reducer 2/2

```
it("should add the holidays on loaded", () => {
  const holidays = createHolidays(
    { title: "Pyramids" },
    { title: "Tower Bridge" }
  );

  const state = holidaysFeature.reducer(
    { holidays: [], loadStatus: "not loaded" },
    loaded({ holidays })
  );

  expect(state).toEqual({ holidays, loadStatus: "loaded" });
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Selectors 1/2

```
const { selectHolidays } = holidaysFeature;


const selectIdTitles = createSelector(selectHolidays, (holidays) =>
  holidays.map(({ id, title }) => ({ id, title }))
);


export const fromHolidays = {
  get: holidaysFeature.selectHolidays,
  selectIdTitles,
};
```

ANGULAR ARCHITECTS INSIDE KNOWLEDGE

Selectors 2/2

```
it('should select the holidays with ids and titles', () => {
  const state: HolidaysState = {
    holidays: createHolidays({ title: 'Pyramids' }, { title: 'Tower Bridge' }),
    loadStatus: 'not loaded',
  };

  expect(fromHolidays.selectIdTitles.projector(state.holidays)).toEqual([
    { id: 1, title: 'Pyramids' },
    { id: 2, title: 'Tower Bridge' },
  ]);
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Effects

```
@Injectable()
export class HolidaysEffects {
  load$ = createEffect(() =>
    this.actions$.pipe(
      ofType(actions.load),
      switchMap(() => this.httpClient.get<Holiday[]>("/holiday")),
      map((holidays) =>
        holidays.map((holiday) => ({
          ...holiday,
          imageUrl: `${this.config.baseUrl}${holiday.imageUrl}`,
        }))
      ),
      map((holidays) => actions.loaded({ holidays }))
    )
  );

  constructor(
    private actions$: Actions,
    private httpClient: HttpClient,
    private config: Configuration,
    private store: Store
  ) {}
}
```

Effects

```
describe("Holidays Effects", () => {
  let httpClient: Mock<HttpClient>;
  const config = new Configuration("https://www.host.com/");
  let store: Mock<Store>;
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Effects

```
describe("Holidays Effects", () => {
  let httpClient: Mock<HttpClient>;
  const config = new Configuration("https://www.host.com/");
  let store: Mock<Store>;

  beforeEach(() => {
    httpClient = createMock(HttpClient);
    store = createMock(Store);
  });
});
```

Effects

```
describe("Holidays Effects", () => {
  let httpClient: Mock<HttpClient>;
  const config = new Configuration("https://www.host.com/");
  let store: Mock<Store>;

  beforeEach(() => {
    httpClient = createMock(HttpClient);
    store = createMock(Store);
  });

  const createEffect = (actions$: Actions) =>
    new HolidaysEffects(actions$, httpClient, config, store);
});
```

Effects

```
describe("Holidays Effects", () => {
  let httpClient: Mock<HttpClient>;
  const config = new Configuration("https://www.host.com/");
  let store: Mock<Store>;

  beforeEach(() => {
    httpClient = createMock(HttpClient);
    store = createMock(Store);
  });

  const createEffect = (actions$: Actions) =>
    new HolidaysEffects(actions$, httpClient, config, store);

  it("should load holidays", async () => {
    const holidays = createHolidays(
      { imageUrl: "pyramids.jpg" },
      { imageUrl: "tower-bridge.jpg" }
    );
    httpClient.get.mockReturnValue(of(holidays));
    const effects = createEffect(of(load));
  });
});
```

Effects

```
describe("Holidays Effects", () => {
  // ...

  it("should load holidays", async () => {
    const holidays = createHolidays(
      { imageUrl: "pyramids.jpg" },
      { imageUrl: "tower-bridge.jpg" }
    );
    httpClient.get.mockReturnValue(of(holidays));
    const effects = createEffect(of(load));

    expect(await firstValueFrom(effects.load$)).toEqual(
      loaded({
        holidays: holidays.map((holiday) => ({
          ...holiday,
          imageUrl: `https://www.host.com/${holiday.imageUrl}`,
        })),
      })
    );
  });
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

RxJS Marbles

- Special Notation
- Primarily made for internal usage
- Use Cases:
  - Complex operator and multiple values
  - Custom operators
- No support for asynchronity outside of operators
  - Promises
  - setTimeout, setInterval

Marble Diagram

🔥 or 🧊

```
const observable = m.cold(

  '--a-b-c',

  { a: 2, b: 10, c: 25 }

);
```

sequence of elements
(1 frame = 1 virtual millisecond)

values of elements

Testing Structure

```
import { marbles } from 'rxjs-marbles/jest';
import { map } from 'rxjs/operators';

test(
 'default check',
 marbles((m) => {
   const source$ = m.cold('--a-b-c', { a: 2, b: 10, c: 25 });

   const destination$ = source$.pipe(map((n) => n * 2));

   m.expect(destination$).toBeObservable(
     '--x-y-z', { x: 4, y: 20, z: 50, });
 })
);
```

Effects

```
// config and setup code

it(
  "should load holidays with rxjs-marbles",
  marbles((m) => {

  })
);
```

Effects

```
// config and setup code

it(
  "should load holidays with rxjs-marbles",
  marbles((m) => {
    const holidays = createHolidays(
      { imageUrl: "pyramids.jpg" },
      { imageUrl: "tower-bridge.jpg" }
    );
    httpClient.get.mockReturnValue(m.cold("250ms h", { h: holidays }));
  })
);
```

Effects

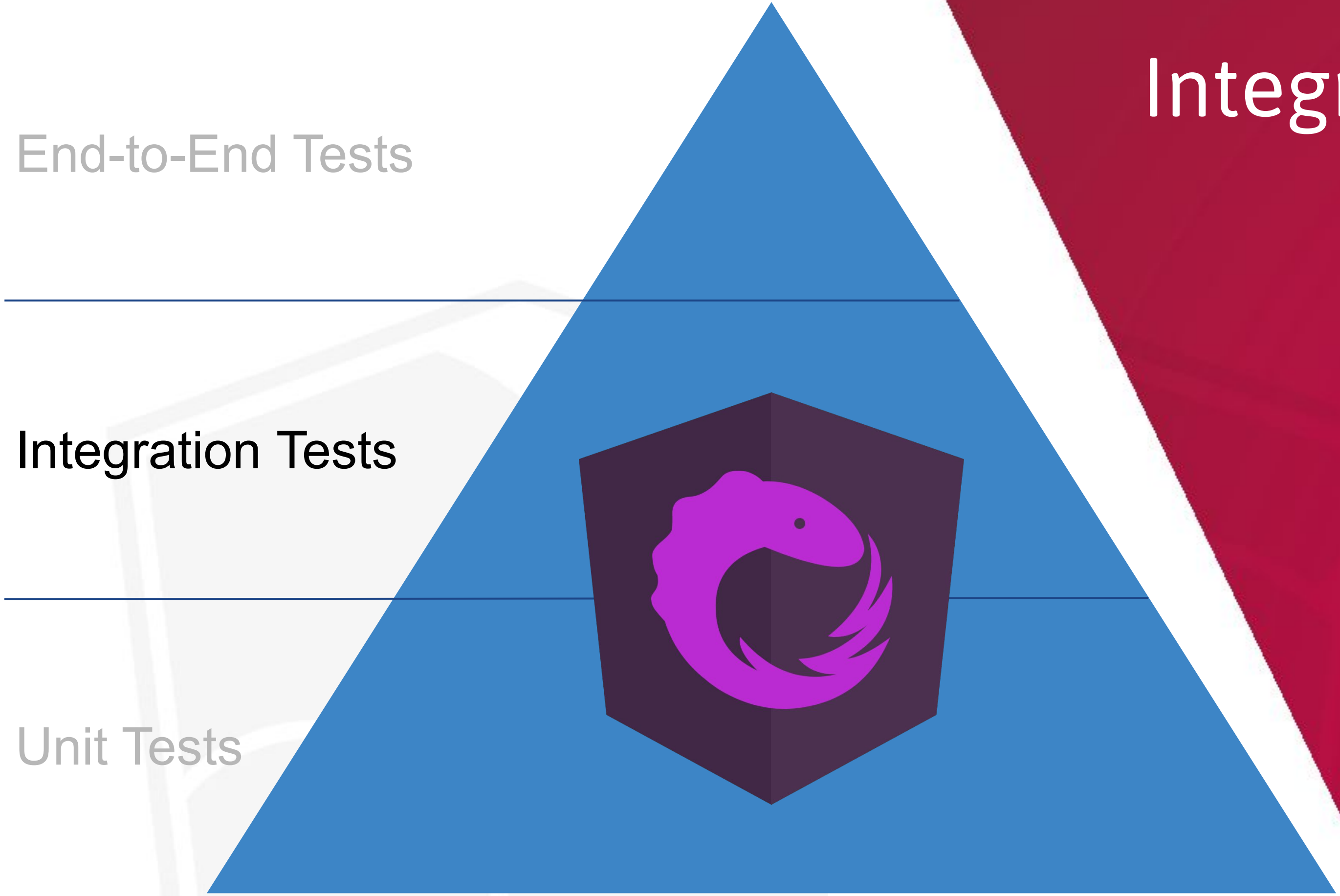
```
// config and setup code

it(
  "should load holidays with rxjs-marbles",
  marbles((m) => {
    const holidays = createHolidays(
      { imageUrl: "pyramids.jpg" },
      { imageUrl: "tower-bridge.jpg" }
    );
    httpClient.get.mockReturnValue(m.cold("250ms h", { h: holidays }));

    const effects = createEffect(m.cold("500ms l", { l: load() }));
  })
);
```

Effects

```
// config and setup code

it(
  "should load holidays with rxjs-marbles",
  marbles((m) => {
    const holidays = createHolidays(
      { imageUrl: "pyramids.jpg" },
      { imageUrl: "tower-bridge.jpg" }
    );
    httpClient.get.mockReturnValue(m.cold("250ms h", { h: holidays }));

    const effects = createEffect(m.cold("500ms l", { l: load() }));

    m.expect(effects.load$).toBeObservable("750ms r", {
      r: loaded({
        holidays: holidays.map((holiday) => ({
          ...holiday,
          imageUrl: `https://www.host.com/${holiday.imageUrl}`,
        })),
      }),
    });
  })
);
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Integration Tests



Initialising NgRx

```
describe("Holidays Data", () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [
        StoreModule.forRoot({}),
      ]
    });
  });
});
```

Initialising Effects

```
describe("Holidays Data", () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [
        StoreModule.forRoot({}),
        EffectsModule.forRoot([]),
      ]
    });
  });
});
```

HttpClient dependency

```
describe("Holidays Data", () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [
        StoreModule.forRoot({}),
        EffectsModule.forRoot([]),
        HttpClientTestingModule,
      ]
    });
  });
});
```

Initialise feature store

```
describe("Holidays Data", () => {
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [
        StoreModule.forRoot({}),
        EffectsModule.forRoot([]),
        HttpClientTestingModule,
        StoreModule.forFeature(holidaysFeature),
        EffectsModule.forFeature([HolidaysEffects]),
      ],
    });
  });
});
```

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Providing services

```
describe("Holidays Data", () => {
  let store: Store;
  let httpCtrl: HttpTestingController;

  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [
        StoreModule.forRoot({}),
        StoreModule.forFeature(holidaysFeature),
        EffectsModule.forRoot([]),
        EffectsModule.forFeature([HolidaysEffects]),
        HttpClientTestingModule,
      ],
      providers: [
        {
          provide: Configuration,
          useValue: new Configuration("https://www.host.com/"),
        },
      ],
    });
```

Finish setup

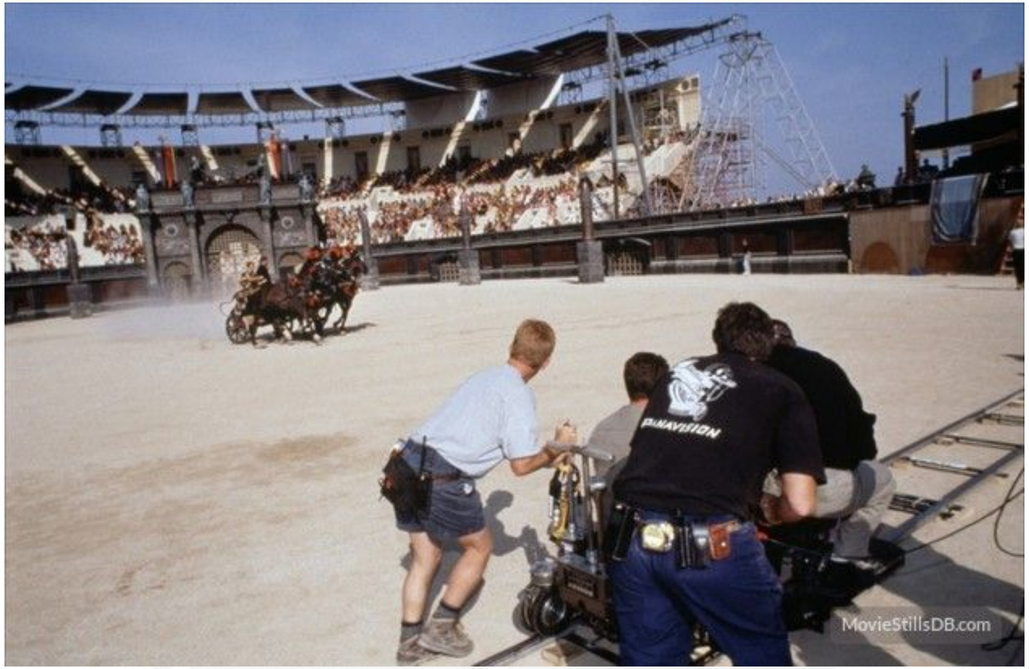
```
describe("Holidays Data", () => {
  let store: Store;
  let httpCtrl: HttpTestingController;

  beforeEach(() => {
    TestBed.configureTestingModule({
      // ...
    });

    httpCtrl = TestBed.inject(HttpTestingController);
    store = TestBed.inject(Store);
  });
});
```

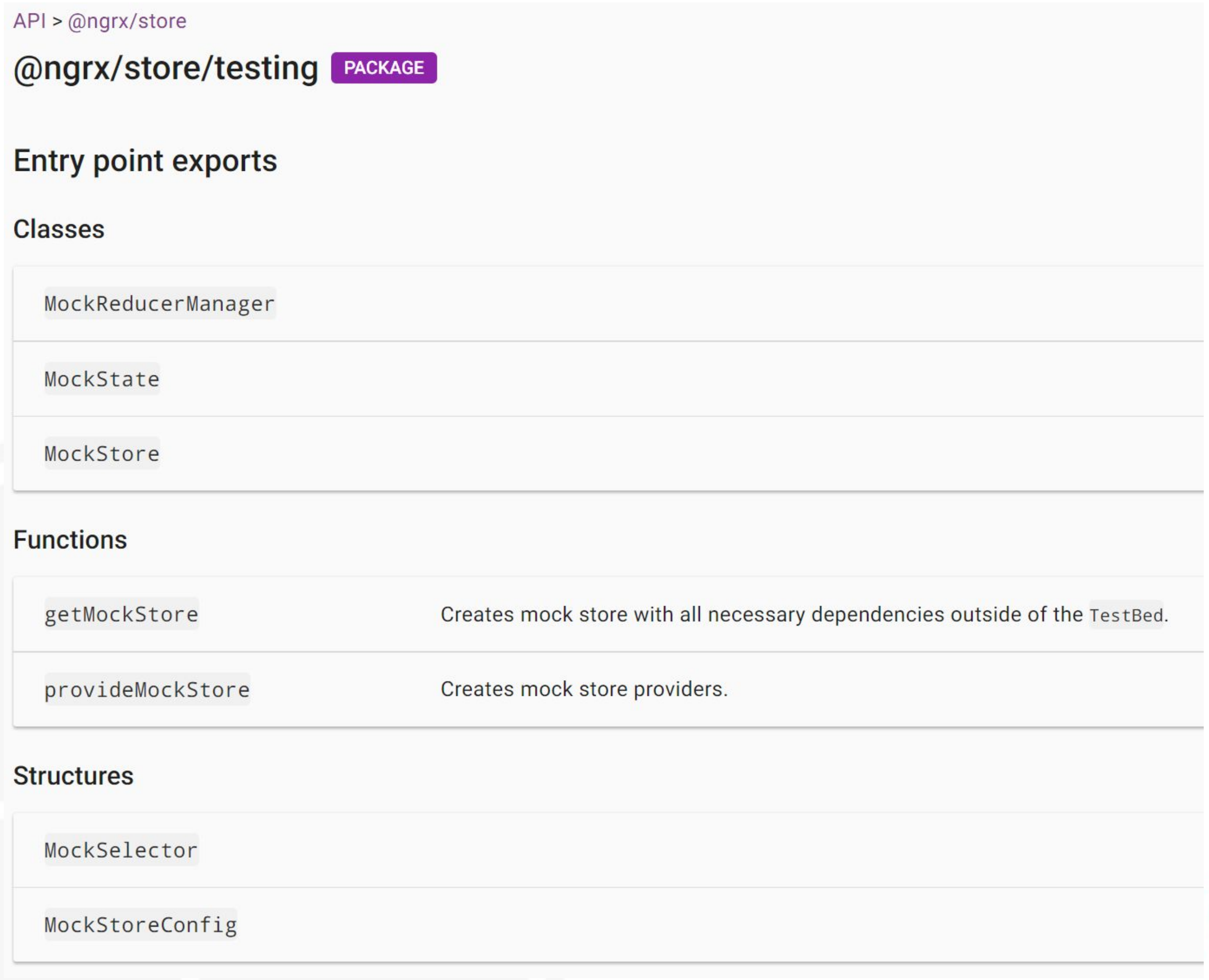
ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

The actual test

```
it("should load holidays", async () => {
  const holidays = createHolidays(
    { title: "Pyramids" },
    { title: "Tower Bridge" }
  );

  store.dispatch(get());
  httpCtrl.expectOne("/holiday").flush(holidays);

  expect(await firstValueFrom(store.select(fromHolidays.get))).toEqual(
    holidays.map((holiday) => ({
      ...holiday,
      imageUrl: `https://www.host.com/${holiday.imageUrl}`,
    }))
  );
  httpCtrl.verify(); // no outstanding http requests
});
```

Mocking NgRx




API > @ngrx/store

@ngrx/store/testing PACKAGE

Entry point exports

Classes

`MockReducerManager`
`MockState`
`MockStore`

Functions

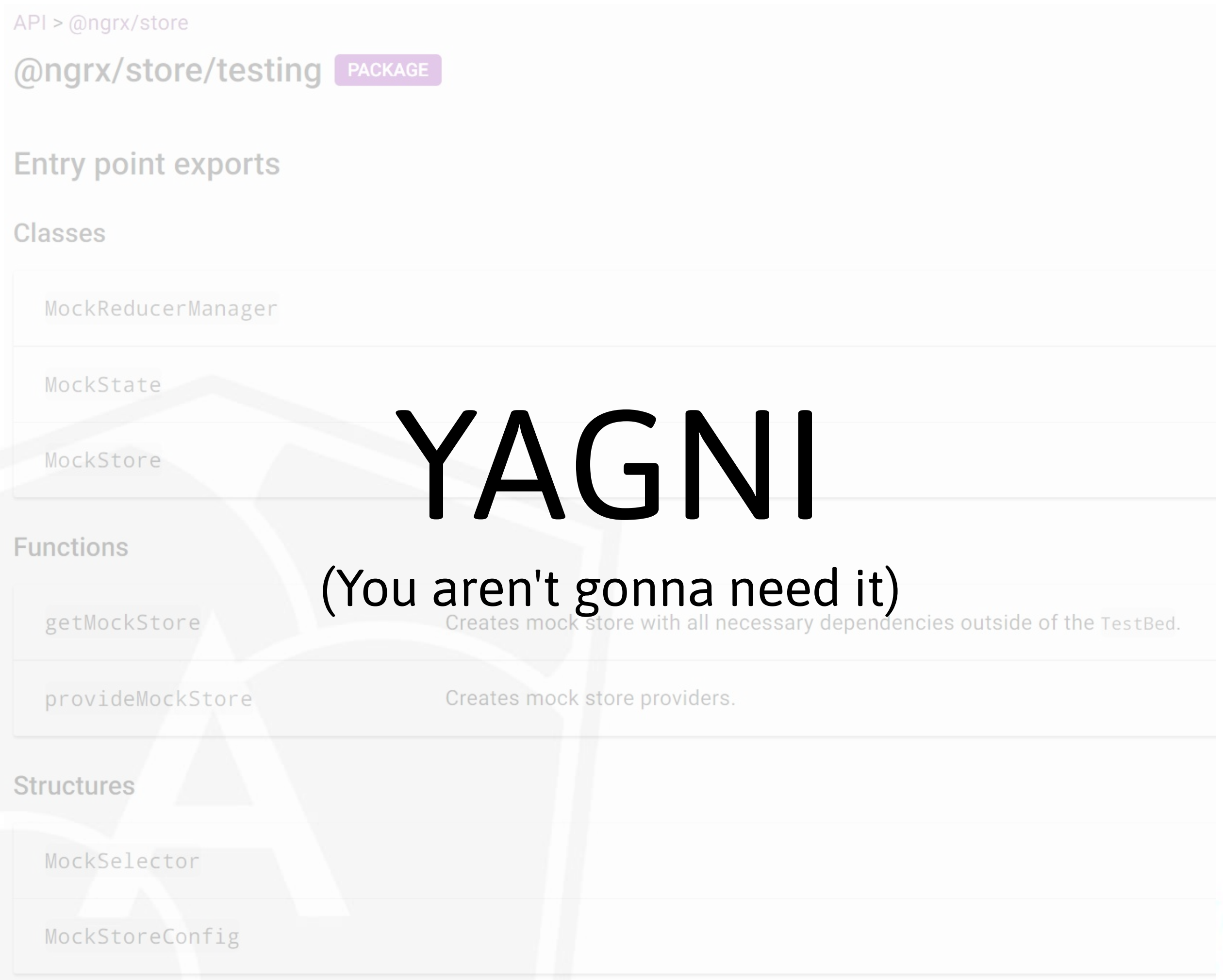
`getMockStore`	Creates mock store with all necessary dependencies outside of the `TestBed`.
`provideMockStore`	Creates mock store providers.

Structures

`MockSelector`
`MockStoreConfig`

YAGNI

(You aren't gonna need it)



API > @ngrx/store

@ngrx/store/testing PACKAGE

Entry point exports

Classes

`MockReducerManager`
`MockState`
`MockStore`

Functions

`getMockStore`	Creates mock store with all necessary dependencies outside of the `TestBed`.
`provideMockStore`	Creates mock store providers.

Structures

`MockSelector`
`MockStoreConfig`

Apply the repository pattern and mock the service