Angular Content Projection with Component Interaction

This example demonstrates how to project a full Angular component (`InfoBoxComponent`) into another component (`DialogComponent`) using **content projection**, and how the projected component can be accessed using `@ContentChild`.

🔧 1. InfoBoxComponent

A reusable info card that accepts inputs and exposes a method.

`info-box.component.ts`

```
import { Component, Input } from '@angular/core';

@Component({
  selector: 'app-info-box',
  template: \`
    <div [class.highlighted]="isHighlighted" class="info-box">
      <h3>{{ title }}</h3>
      <p>{{ message }}</p>
    </div>
  \`,
  styles: [\`
    .info-box {
      padding: 12px;
      border: 1px solid #ccc;
      border-radius: 6px;
      background-color: #f9f9f9;
      transition: background-color 0.3s;
    }

    .highlighted {
      background-color: #ffeeba;
      border-color: #ffc107;
    }
  \`]
})
export class InfoBoxComponent {
  @Input() title: string = 'Default Title';
  @Input() message: string = 'Default message content.';
  isHighlighted = false;

  highlight() {
    this.isHighlighted = true;
  }
}
```

🧱 2. DialogComponent

Accepts projected content and interacts with the projected component.

`dialog.component.ts`

```
import {
  Component,
  ContentChild,
  AfterContentInit
} from '@angular/core';
import { InfoBoxComponent } from '../info-box/info-box.component';

@Component({
  selector: 'app-dialog',
  template: \`
    <div class="dialog">
      <h2>Dialog Header</h2>
      <div class="dialog-body">
        <ng-content></ng-content>
      </div>
    </div>
  \`,
  styles: [\`
    .dialog {
      border: 1px solid #666;
      padding: 16px;
      border-radius: 8px;
    }
  \`]
})
export class DialogComponent implements AfterContentInit {
  @ContentChild(InfoBoxComponent) infoBox!: InfoBoxComponent;

  ngAfterContentInit() {
    if (this.infoBox) {
      console.log('InfoBox title:', this.infoBox.title);
      this.infoBox.highlight(); // Interact with projected component
    }
  }
}
```

💡 3. AppComponent (Usage Example)

Project `app-info-box` inside the dialog.

`app.component.html`

```
<app-dialog>
  <app-info-box
    [title]="'Hello from AppComponent'"
    [message]="'This InfoBox is being projected into the dialog.'"
  ></app-info-box>
</app-dialog>
```

☑ Key Concepts

Concept	**Description**
`ng-content`	Used to insert projected content
`@ContentChild(InfoBoxComponent)`	Gets the instance of a projected component
Method interaction	You can call public methods of projected components (`highlight()` in this case)
Input binding	Projected components can have `@Input()` just like normal usage

📌 Pro Tips

- You can project multiple components and access them using `@ContentChildren`.
- This technique is useful in modals, tab systems, cards, and form wrappers.
- You can also use structural directives (e.g. `*ngIf`) around projected components and still access them using `@ContentChild`.