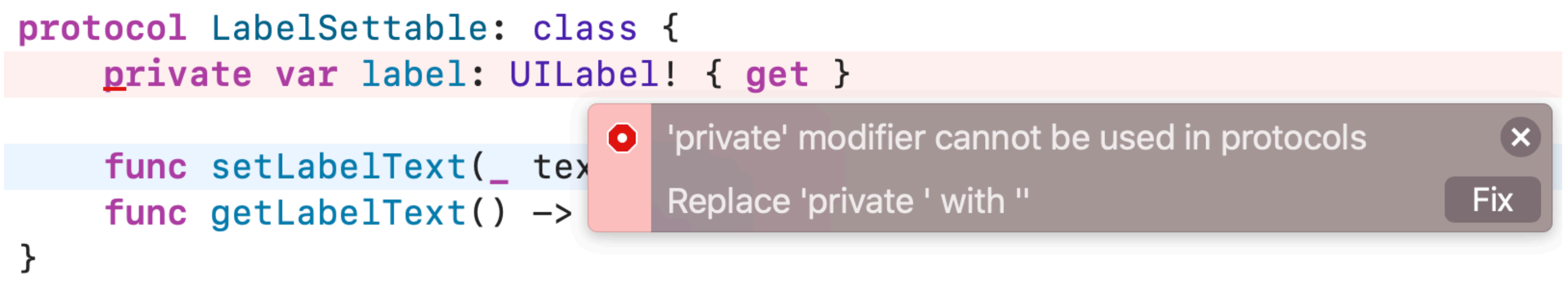
Private Properties in Protocols

Viranchee Lotia
@code_magician

```swift
protocol LabelSettable: class {
    private var label: UILabel! { get }

    func setLabelText(_ text: String)
    func getLabelText() -> String
}
```

```swift
protocol LabelSettable: class {
    private var label: UILabel! { get }

    func setLabelText(_ text: String)
    func getLabelText() -> String
}
```

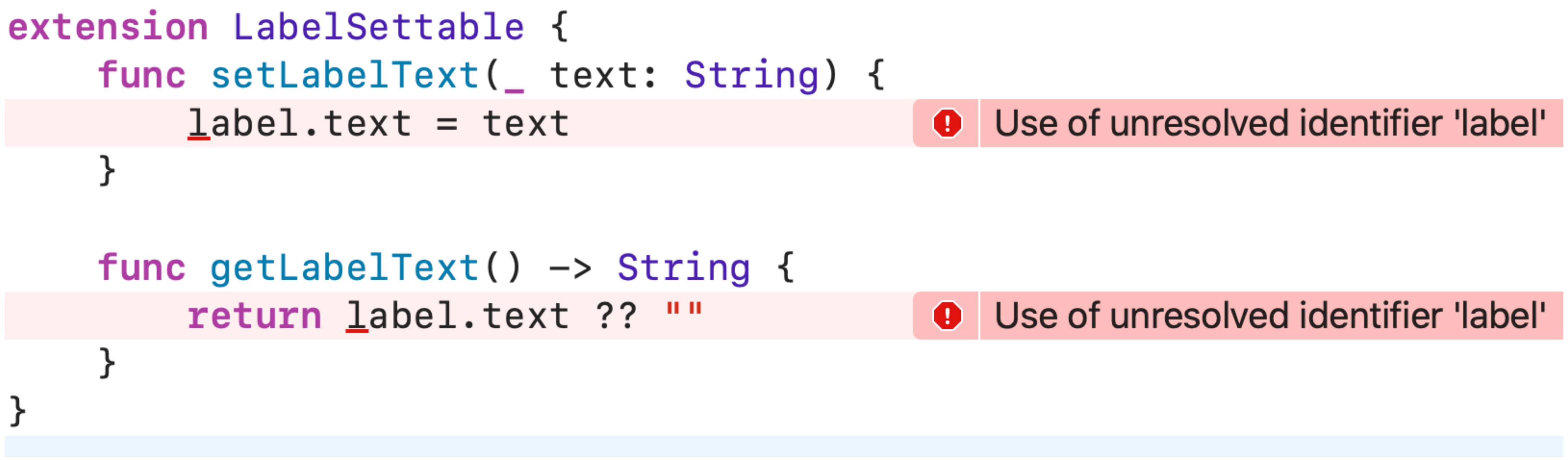
```swift
extension LabelSettable {

    func setLabelText(_ text: String) {
        label.text = text
    }

    func getLabelText() -> String {
        return label.text ?? ""
    }

}
```



```
protocol LabelSettable: class {
    private var label: UILabel! { get }

    func setLabelText(_ tex
    func getLabelText() ->
}
```

'private' modifier cannot be used in protocols

Replace 'private ' with ''



```
extension LabelSettable {
    func setLabelText(_ text: String) {
        label.text = text                    Use of unresolved identifier 'label'
    }

    func getLabelText() -> String {
        return label.text ?? ""              Use of unresolved identifier 'label'
    }
}
```

```
protocol LabelSettable: class {
    ~~private~~ var label: UILabel! { get }

    func setLabelText(_ text: String)
    func getLabelText() -> String
}
```

```swift
extension LabelSettable {

    func setLabelText(_ text: String) {
        label.text = text
    }

    func getLabelText() -> String {
        return label.text ?? ""
    }

}
```

Now, let’s make properties private

Copy Paste the functionality in all classes!!

Note: Extensions can only access properties available at that scope.

```swift
class CustomView: UIView {
    private var label = UILabel()
}

extension CustomView: LabelSettable {
    func setLabelText(_ text: String) {
        label.text = text
    }

    func getLabelText() -> String {
        return label.text ?? ""
    }
}
```

Automate It! 😎

Sourcery / GYB

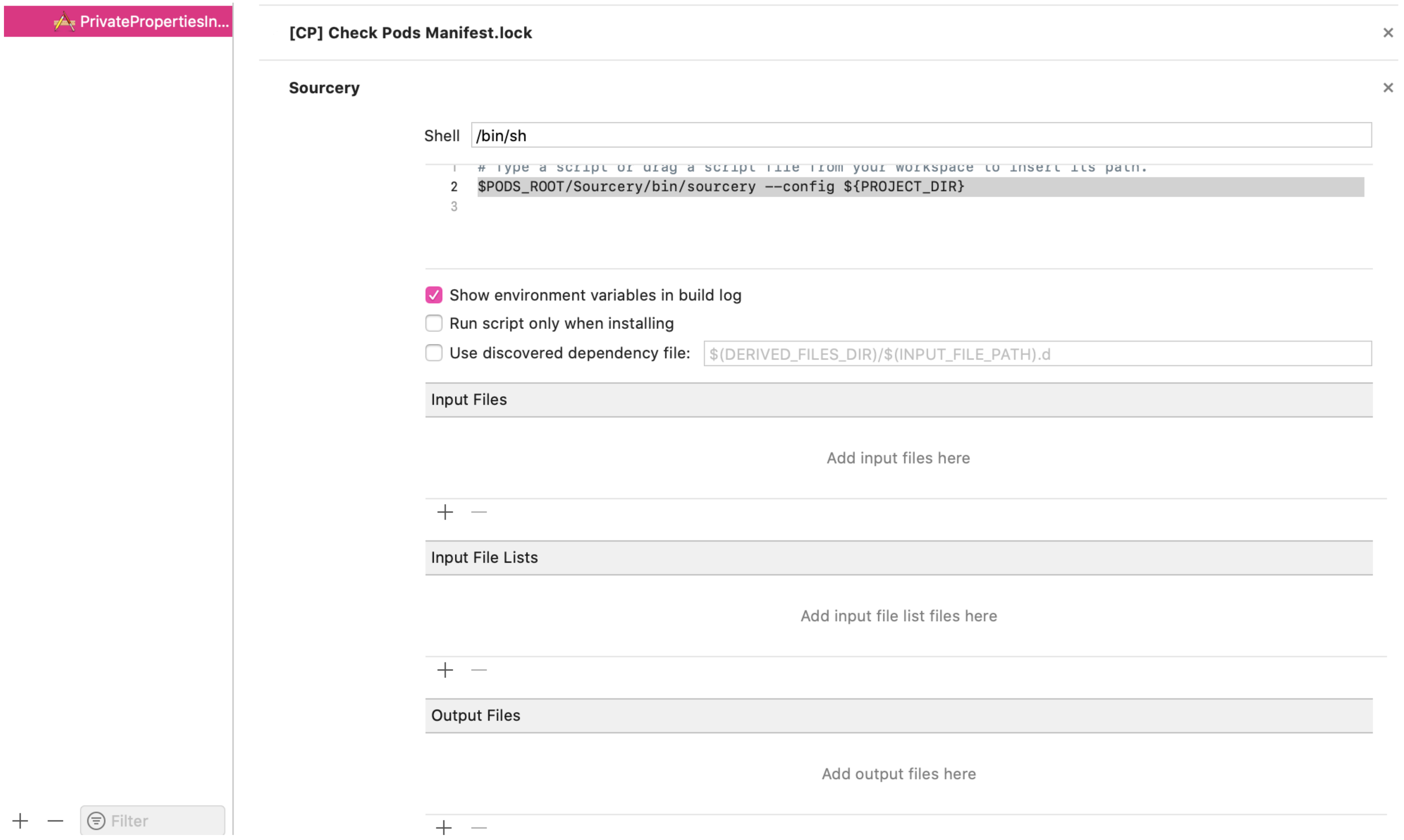
File: Project/SourceryTemplates/LabelSettable.stencil

```
{% for type in types.implementing.UILabelSettable %}

// sourcery:inline:auto:TableCellTableViewCell.LabelSettable
// MARK: - Sourcery LabelSettable

        func setLabelText(_ text: String) {
            label.text = text
        }

        func getLabelText() -> String {
            return label.text ?? ""
        }

// sourcery:end

{% endfor %}
```

File: .sourcery.yml

```yaml
sources:
    - Project/Views
    - Project/Helpers
templates:
    - Project/SourceryTemplates
output: Project/SourceryGenerated
```

Xcode Build Phase



PrivatePropertiesIn...

[CP] Check Pods Manifest.lock

Sourcery

Shell /bin/sh

```
# Type a script or drag a script file from your workspace to insert its path.
$PODS_ROOT/Sourcery/bin/sourcery --config ${PROJECT_DIR}
```

- [✓] Show environment variables in build log
- [ ] Run script only when installing
- [ ] Use discovered dependency file: $(DERIVED_FILES_DIR)/$(INPUT_FILE_PATH).d

Input Files

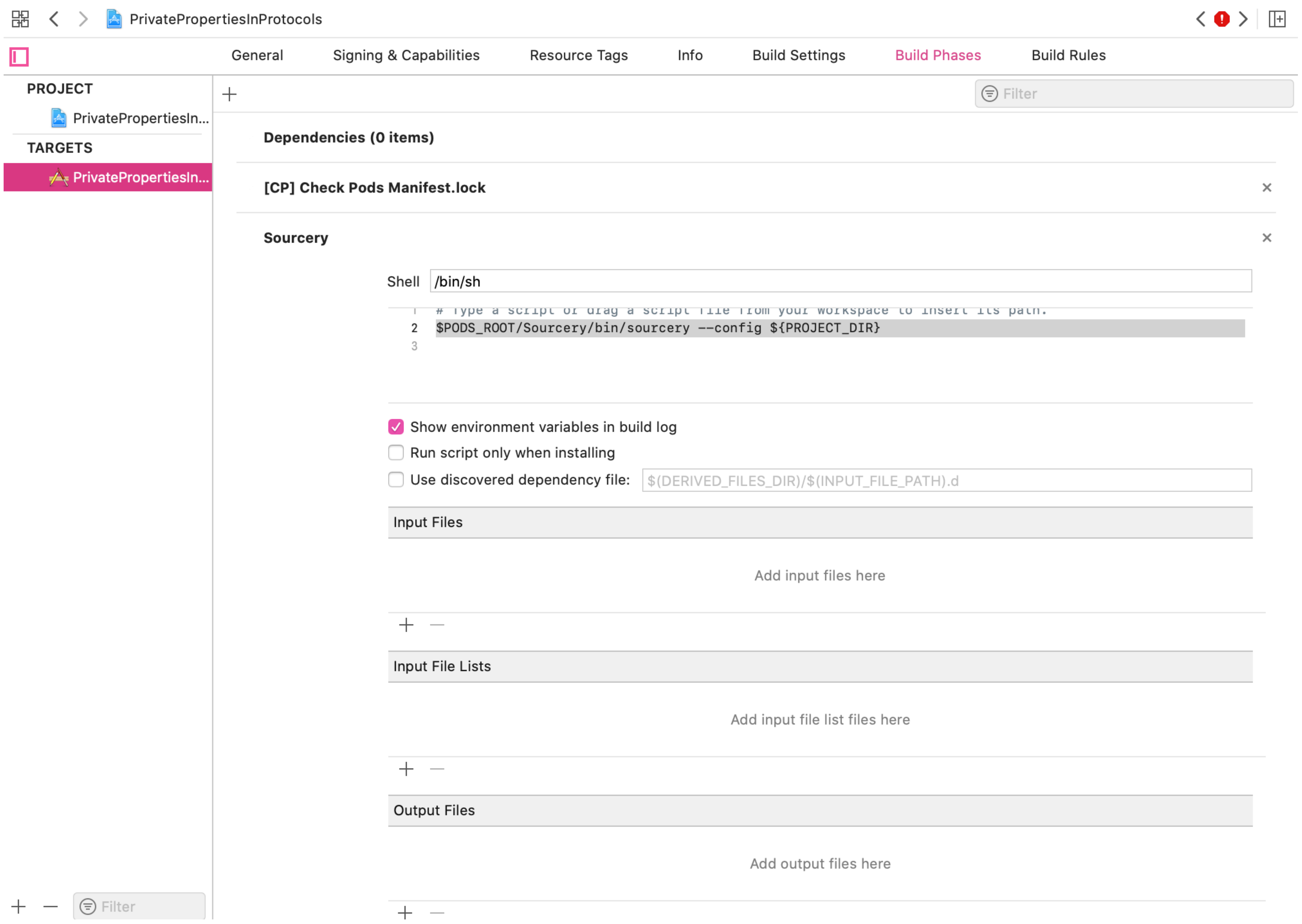
Add input files here

Input File Lists

Add input file list files here

Output Files

Add output files here



Dependencies (0 items)

[CP] Check Pods Manifest.lock

Sourcery

Shell /bin/sh

```
# Type a script or drag a script file from your workspace to insert its path.
$PODS_ROOT/Sourcery/bin/sourcery --config ${PROJECT_DIR}
```

- [x] Show environment variables in build log
- [ ] Run script only when installing
- [ ] Use discovered dependency file: $(DERIVED_FILES_DIR)/$(INPUT_FILE_PATH).d

Input Files

Add input files here

Input File Lists

Add input file list files here

Output Files

Add output files here

Output

```swift
class CustomCell: UITableViewCell, LabelSettable {

    @IBOutlet weak private var label: UILabel!

// sourcery:inline:auto:TableCellTableViewCell.UILabelSettable
// MARK: - Sourcery UILabelSettable

        func setLabelText(_ text: String) {
            label.text = text
        }

        func getLabelText() -> String {
            return label.text ?? ""
        }
// sourcery:end
}
```

- Step 1: Make those properties public
- Step 2: Implement Protocol Extension, Code compiles
- Step 3: Copy it to Sourcery template configured to Inline that code
- Step 4: Revert Step 1 & 2

Existing Examples

File: AVProtocol1

```swift
//  VideoPlayable
protocol VideoPlayable {

    ///startPlayback
    func startPlayback(with options: VideoPlaybackOptions)

    ///stopPlayback
    @discardableResult
    func stopPlayback() -> VideoPlaybackOptions
}

///  Use this Protocol when the VideoPlayer implementation is of AVFoundation
protocol AVVideoPlayable: VideoPlayable { }
```

File: Protocol1.stencil

```
{% for type in types.implementing.AVVideoPlayable %}
// sourcery:inline:auto:PoppinTableViewCell.AVVideoPlayable
// MARK: - Sourcery VideoPlayable Conformance

    private func setMuteImage(_ button: UIButton) {
        let image = (avPlayer?.isMuted ?? true) ? Asset.Reactions.mute.image :
Asset.Reactions.unMute.image
        button.setImage(image, for: .normal)
    }

    @discardableResult
    func stopPlayback() -> VideoPlaybackOptions {
        self.avPlayer?.pause()
        return avPlayer?.playerConfiguration ?? VideoPlaybackOptions()
    }

    func startPlayback(with options: VideoPlaybackOptions) {
        self.avPlayer?.isMuted = options.mute
        setMuteImage(muteButton)
        self.avPlayer?.play()
    }
// sourcery:end
}

{% endfor %}
```

File: Protocol2

```
///  A contract that a particular Class, possibly a ViewController, is able to play videos automatically in a
tableView on scroll.
protocol VideoPlayableController: class {

    ///  The Index Path which is visible, and on which video is being played on
    var visibleIndexPath: IndexPath? { get set }

    ///  Call this method in ScrollViewDidScroll
    /// - Parameter tableView:  The tableView which needs to be given autoplay logic
    func autoplayVideosIn(tableView: UITableView)

    ///  Call this method to pause video on current cell
    func pauseVideo()

    ///  Call this method to play video on current cell
    func playVideo()

}


///  Sub type  specialised for ViewControllers with a TableView
protocol VideoPlayableOnTableViewController: VideoPlayableController { }
```

File: Protocol2.stencil

```
{% for type in types.implementing.VideoPlayableOnTableViewController %}
// sourcery:inline:auto:PoppinTableViewCell.VideoPlayableOnTableViewController
// MARK: - Sourcery VideoPlayableOnTableViewController Conformance

    func pauseVideo() {
        if let playingIndexPath = visibleIndexPath {
            guard let cell = tableView.cellForRow(at: playingIndexPath) as?
                                                       VideoPlayable else { return }
            cell.stopPlayback()
        }
    }
    func playVideo() {
        if let playingIndexPath = visibleIndexPath {
            guard let cell = tableView.cellForRow(at: playingIndexPath) as?
VideoPlayable else { return }
            cell.startPlayback(with: .init(mute: true))
        }
    }
// sourcery:end
}
{% endfor %}
```

File: Protocol3

```swift
/// Contract to receive Rating of a media
protocol PopoverRateDelegate: AnyObject {

    /// Click on rate button
    /// - Parameter rating: media object
    func didClick(rating: Media.Rating)
}
```

File: Protocol3.stencil

```
{% for type in types.implementing.PopoverRateDelegate %}
// sourcery:inline:auto:PoppinTableViewCell.PopoverRateDelegate

// MARK: - Sourcery PopoverRateDelegate Conformance

    func didClick(rating: Media.Rating) {
        guard media.concreteUserRating != rating else { return }
        sendRating(rating)
    }
    ///  Send rating to network and update image of rating button
    /// - Parameter rating:  New Media rating provided/ to update
    private func sendRating(_ rating: Media.Rating) {
        if rating == .notRated { return }
        let mediaAndRate = MediaAndRate(id: media.id, rating: 
rating)
        requestManager.rateMedia(mediaAndRate) { [weak self] (_) in
            guard let self = self else { return }
            self.media.concreteUserRating = rating
            Notifications.refreshMedia(media: self.media).post()
        }
    }
// sourcery:end
{% endfor %}
```

Learning Sourcery

```
pod 'Sourcery', ‘0.17'
```

```
→  bash: sourcery --help
Usage:

    $ sourcery

Options:
    --watch [default: false] - Watch template for changes and regenerate
as needed.
    --disableCache [default: false] - Stops using cache.
    --verbose [default: false] - Turn on verbose logging
    --quiet [default: false] - Turn off any logging, only emmit errors.
    --prune [default: false] - Remove empty generated files
    --sources - Path to a source swift files. File or Directory.
    --exclude-sources - Path to a source swift files to exclude. File or
Directory.
    --templates - Path to templates. File or Directory.
    --exclude-templates - Path to templates to exclude. File or Directory.
    --output - Path to output. File or Directory. Default is current path.
    --config - Path to config file. File or Directory. Default is current
path.
    --force-parse - File extensions that Sourcery will be forced to parse,
even if they were generated by Sourcery.
    --args - Custom values to pass to templates.
    --ejsPath - Path to EJS file for JavaScript templates.
```

File: .sourcery.yml

```
sources:
    - Project/Views
    - Project/Helpers
templates:
    - Project/SourceryTemplates
output: Project/SourceryGenerated
```