Structure

AssignableDocument

An assignable document is an augmented PDF that allows teachers to mark up the PDF with the intention of students taking the assessment.

iOS 17.4+ | iPadOS 17.4+ | Mac Catalyst 17.4+ | visionOS

```
struct AssignableDocument
```

Overview

The document has several parts, which includes the ability to modify the PDF, annotate the PDF, and define question regions in the PDF.

To add a question to this assignable, you can use `appendQuestion(pageID:rect:maxScore:)` which will take care of associating a box to a page and adding a question connected to that box to the document.

This document is fully mergeable, which means that any copies of this document that are independently mutated can be merged into a determinisitic resulting document. You can merge copies of this document into this one using `merge(_:)`. You can also merge individual parts of copies of this document into this one with `merge(partData:into:)`. For example, if deviceA has documentA and deviceB has documentB, which is a copy of documentA. When a user changes a question in documentB on deviceB, deviceB can export that part's data and send it to deviceA to be merged back into documentA.

You can create as many of these objects as you have memory for. This type assumes single-threaded access.

Topics

Creating an assignable document

`init(pdfURL: URL, id: String?) throws`

Initializes a new assessment document that is based on the PDF located at the provided URL. If the file located at the URL provided cannot be accessed, this initializer throws.

`init(id: AssignableDocument.ID, partData: [AssignableDocument.PartID : MergeablePartData]) async throws`

Construct an instance of this object with the parts data passed in.

`enum MergeablePartData`

~~`init(id: AssignableDocument.ID, partData: [AssignableDocument.PartID : URL]) throws`~~

Construct an instance of this object with the parts data passed in.

Deprecated

~~`init(pdfURL: URL, authors: [some UserIdentity], id: String?) throws`~~

Initializes a new assessment document that is based on the PDF located at the provided URL. If the file located at the URL provided cannot be accessed, this initializer throws.

Deprecated

Inspecting an assignable document

`typealias ID`

A type representing the stable identity of this document.

`var id: AssignableDocument.ID`

The stable identity of this document.

`var isMultiPageDocument: Bool`

`true`, if this document has more than one page; `false`, otherwise.

`var isPartial: Bool`

Denotes whether or not this document is a partial one.

`enum PartIDs`

An enumeration containing the identities of parts managed by this view.

`var partIDs: [AssignableDocument.PartID]`

Returns a collection of identifiers reflecting the manifest of parts available in the document.

`struct Question`

A question in the assignable document.

`struct QuestionBox`

A box on a page for a question.

`var questions: [AssignableDocument.Question]`

A collection of questions defined for this assignable.

`typealias Element`

The type for elements of this document. An element is a component of the document such as a page or question.

`var pagesDebugDescription: String`

`enum Error`

Errors for this document type.

Getting and setting the questions

`func appendQuestion(pageID: AssignableDocument.Page.ID, rect: CGRect, maxScore: Double?) -> AssignableDocument.Question.ID`

Creates a new question and appends it to the document.

`func questions(on: AssignableDocument.Page.ID) -> [AssignableDocument.Question]`

Find questions that exist on the specified page.

~~`func removeQuestion(AssignableDocument.Question.ID) -> AssignableDocument.Question?`~~

Removes a question and its boxes from the document.

Deprecated

Computing the max score

`func computeMaxScore(defaultQuestionMaxScore: Double?) -> Double?`

Computes the maximum possible score for this `AssignableDocument` as defined by each individual question's `maxScore`.

Getting the authors

`var authors: [AnyUserIdentity]`

The set of identities of users that created or modified this assignable. Treated as a set.

Getting the configuration

`typealias Configuration`

The configuration for an assessment which contains options for display of marks and their point values.

`var configuration: some AssignableDocumentConfiguration`

The configuration of this assessment which contains options for display of marks and their point values.

`enum CorrectMarkType`

The glyph to use that represents a correct mark.

Merging the parts

`func merge(AssignableDocument) async throws -> Bool`

Merge another object of this type into this object.

`func merge(partData: MergeablePartData, into: AssignableDocument.PartID) async throws -> Bool`

Merges an individual part into the specified part of this object.

~~`func merge(other: AssignableDocument) throws -> Bool`~~

Merge another object of this type into this object.

Deprecated

~~`func merge(partID: AssignableDocument.PartID, partDataURL: URL) throws -> Bool`~~

Merges an individual part's data into the specified part of this object.

Deprecated

Producing thumbnails

```
func questionThumbnails(visibleParts: [AssignableDocument.PartID])
async -> [AssignableDocument.Question.ID : [AssignableDocument.Question
.Thumbnail]]
```

Produces thumbnails of question regions within the document.

Making the parts

```
func makePart(for: AssignableDocument.PartID) throws -> MergeablePart
Data?
```

Creates data for the part with the given identifier.

Exporting the parts

```
func exportBaseAsPDF() async -> PDFDocument
```

Exports the base part of this document to a `PDFDocument`.

```
func exportParts(identifiedBy: [AssignableDocument.PartID]) async
throws -> [AssignableDocument.PartID : MergeablePartData]
```

Given a set of part identifiers, return a dictionary of part ID to part data.

~~`func export(partIDs: [AssignableDocument.PartID]) async throws -> [AssignableDocument.PartID : URL]`~~

Given a set of part identifiers, return a dictionary of part ID to data objects for the requested layers.

Deprecated

Accessing documents

```
subscript(AssignableDocument.Page.ID) -> AssignableDocument.Page?
```

Access the page that the identifier denotes, if any.

~~`subscript(AssignableDocument.QuestionBox.ID) -> AssignableDocument.QuestionBox?`~~

Access the question box that the identifier denotes, if any.

Deprecated

~~`subscript(AssignableDocument.Question.ID) -> AssignableDocument.Question?`~~

Access the question that the identifier denotes, if any.

Deprecated

Comparing assignable documents

`static func == (AssignableDocument, AssignableDocument) -> Bool`

Returns a Boolean value indicating whether two values are equal.

Hashing the assignable document

`func hash(into: inout Hasher)`

Hashes the essential components of this value by feeding them into the given hasher.

Default Implementations

- Assignable Implementations
- MergeableDocument Implementations

Relationships

Conforms To

```
Assignable
Copyable
Equatable
Hashable
Identifiable
MergeableDocument
MergeablePartsContainer
```

See Also

Assignable document

`struct AssignedWorkDocument`

An assigned work document is a document that contains taker and scorer markup specific to a taker. It also contains a copy of the assignable document upon which it is based.

`protocol Assignable`

Documents conforming to this protocol can be assigned to a user.