CSP ベースの Web アプリケーションの作成

Version 2024.1
2024-06-03

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

CSP ベースの Web アプリケーションの作成
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

テーブル一覧

1

CSP ベースの Web アプリケーションの概要

InterSystems IRIS® データ・プラットフォームは、Web ユーザ・インタフェースを作成できるようにするテクノロジを提供します。これまでの経緯から、このテクノロジは CSP として知られ、その成果が CSP ベースの Web アプリケーションです。

CSP は、ブラウザが認識する前に途中で HTTP 要求を捉え、それをブラウザに直接書き込む簡単な方法を提供します。これは、ユーザが生成する Web ページを完全に制御できるようになることを意味します。必要な JavaScript をページに追加するだけで済むため、必要に応じて CSP をサードパーティの JavaScript ライブラリと組み合わせて使用できます。

1.1 CSP ベースの Web アプリケーションのコンポーネント

アプリケーション開発者の観点からすると、CSP ベースの Web アプリケーションは InterSystems IRIS サーバ上の以下の要素で構成されます。

- REST オプションではなく CSP/ZEN オプションを使用するように構成された Web アプリケーション定義。

  Web アプリケーション定義は、許可される認証メカニズム、コードを実行する InterSystems IRIS ネームスペース、特定の目的に使用するカスタム・ページ、タイムアウト、静的ファイルの処理などを制御します。

- 1 つ以上の CSP ページ・クラス。これらのクラスは、HTTP 要求への応答として完全な HTTP 要求を生成します。CSP ページ・クラスでは、要求およびセッション情報に簡単にアクセスすることができ、ユーティリティ・メソッドによって Cookie の管理や CGI 変数へのアクセスなどが可能になります。

- 外部の JavaScript ファイル、スタイル・シート、画像ファイル、静的 HTML ファイル、および必要に応じてその他のリソース。これらのリソースは同じサーバ上に置くことも、他のサーバ上に置くことも、あるいはその組み合わせで使用することもできます。

アーキテクチャには、以下の追加のコンポーネントも存在します。

- Web サーバ (InterSystems IRIS での使用がサポートされているもの)。

- Web ゲートウェイ。Web サーバと InterSystems IRIS サーバ間の仲介として機能します。Web ゲートウェイは、これらのサーバへの接続プールを維持します。

  Web ゲートウェイの構成には、InterSystems IRIS サーバで実行中の Web アプリケーションの定義が含まれます。これらの定義により、Web ゲートウェイは InterSystems IRIS サーバ上の正しい Web アプリケーションに要求をルーティングできます (次に説明するように、CSP サーバを介して要求を送信します)。

- CSP サーバ。これは、InterSystems IRIS サーバで実行される専用プロセスで、Web ゲートウェイからの要求を待機し、必要に応じてそれらを処理します。各 InterSystems IRIS サーバは、必要な数だけ CSP サーバ・プロセスを実行できます (マシン・タイプによる制限は適用されます。ライセンスの計算では、CSP サーバはカウントされません)。

1.2 情報の流れ

CSP ベースの Web アプリケーションでは、要求と応答のプロセスは以下のようになります。

1. HTTP クライアント (通常は Web ブラウザ) は、HTTP を使用して Web サーバにページを要求します。
2. Web サーバの構成方法によって、Web サーバは要求を CSP 要求として認識し、それを Web ゲートウェイに転送します。
3. Web ゲートウェイは、通信する InterSystems IRIS サーバを決定し、そのターゲット・システム上の CSP サーバに要求を転送します。
4. CSP サーバは要求を処理し、要求が静的ファイルに対するものか CSP クラスに対するものかを判断します。

   CSP サーバには、**%CSP.StreamServer** クラスであるストリーム・サーバが含まれます。ストリーム・サーバは、ファイルやその他のストリームの処理を行います。要求が静的ファイルに対するものであれば、ストリーム・サーバはローカル・ファイル・システム内でファイルを検索し、それをエンコードする方法を決定してパッケージ化します。

   要求がクラスに対するものであれば、CSP サーバはクラスの Page() メソッドを呼び出し、それを受けてこのメソッドがそのクラスで定義されているコールバック・メソッドを呼び出します。
5. CSP サーバは要求されたコンテンツを Web ゲートウェイに返し、Web ゲートウェイはそれを Web サーバに渡します。
6. Web サーバはコンテンツをブラウザに送信し、ブラウザでコンテンツが表示されます。

2

CSP ベースの Web アプリケーションの構成

その他のアプリケーションを構成するのと同じ管理ポータルのページ (および同じ API) を使用して、CSP ベースの Web アプリケーションを構成できます。このページでは、主に CSP ベースのアプリケーションに固有の設定について取り上げます。

CSP ベースの Web アプリケーションを構成するには、以下の手順を実行します。

1. アプリケーション定義を表示 (必要に応じてその前に作成) するには、"アプリケーションの作成および編集" の手順に従います。
2. [一般] タブで、以下の情報を使用して Web アプリケーションを定義します。
3. アプリケーション定義のその他の部分については、"アプリケーションの作成および編集" を参照してください。
4. この Web アプリケーションへのアクセスを提供するよう Web ゲートウェイ構成も更新するようにしてください (基礎的な情報は、"CSP ベースの Web アプリケーションのコンポーネント" を参照してください)。

2.1 CSP ベースの Web アプリケーションの一般設定

CSP ベースの Web アプリケーションの場合、[一般] タブの最初のセクションに以下のように値を指定します。

名前

アプリケーションの識別子を指定します。名前の先頭にはスラッシュ (/) を含める必要があります (例えば、/myorg/myapp アプリケーションのようにします)。

名前 /csp/docbook は予約されていることに注意してください。

説明

アプリケーションの説明テキストを指定します。

ネームスペース

このアプリケーションが実行されるネームスペースを指定します。別のネームスペースを選択すると、そのネームスペースの既定アプリケーションがこのドロップダウン・メニューの右側に即座に表示されます。

ネームスペースのデフォルト・アプリケーション

アプリケーションがこのネームスペースの既定アプリケーションかどうかを指定します。%System.CSP.GetDefaultApp() メソッドは、ネームスペースの既定アプリケーションを返します。$system.OBJ.Load や $system.OBJ.ImportDir などの InterSystems IRIS® データ・プラットフォームのインポート関数は、関連付けられているアプリケーションなしでページをインポートする際に、この既定アプリケーションを使用します。

アプリケーション有効

アプリケーションを使用できるかどうかを指定します。有効になっていれば、認証および承認されたユーザはアプリケーションを使用できます。無効の場合は使用できません。

[REST] または [CSP/ZEN] の有効化

[CSP/ZEN] を選択します。

アナリティクス

このアプリケーション内で Business Intelligence と自然言語処理の使用を有効にするかどうかを指定します。

着信 Web サービス

このアプリケーション内で SOAP 要求を処理するかどうかを指定します。無効にするには、チェックを外します。

ログイン CSRF 攻撃を防ぐ

アプリケーションでクロスサイト・リクエスト・フォージェリ (CSRF) 攻撃を自動的に防ぐかどうかを指定します。すべての新しいアプリケーションでこのオプションを有効にすることをお勧めします。また、そのアプリケーションのページをプログラムによって要求するコードが存在する場合を除き、すべての既存のアプリケーションについてもそうすることをお勧めします。

2.2 セキュリティの設定

CSP ベースの Web アプリケーションについては、以下のように [セキュリティの設定] を指定します。

必要なリソース

ユーザがアプリケーションを実行するために Use 許可を保持していなければならないリソース。

CSP ページのクラス内で必要なリソースを指定することもできます。両方のメカニズムが適用されます。

ID でグループ化

By-ID グループで使用して、複数の Web アプリケーション間で認証を共有することを可能にします。このアプリケーションのグループ名を入力して、このグループ名を持つその他すべてのアプリケーションと認証特権を共有します。

許可された認証方法

一連の定義済みのメカニズムの中で、このアプリケーションでサポートされている認証メカニズム。

許可したクラス

このアプリケーション内で実行できるメソッドを持つクラスを指定します。これを行うには、以下の 3 つの方法があります。

- ObjectScript パターン・マッチを使用します。例：`1"myclass".3N` は、`myclass123.cls` がこのアプリケーションで実行されることを許可しますが、`myclassxy.cls` は許可しません。
- 先頭に @ が付いた、ブーリアン値に評価される ObjectScript 式を使用します。要求されたクラス名は、class という名前の変数として渡されます。例：`@class = "PermittedClasses.PermittedPage"`
- クラス・メソッドへの呼び出しを使用します (@syntax も使用できます)。例：`##class(MyPackage).CheckClassIsPermitted(class)`

"ページおよびクラスへのアクセスの有効化" も参照してください。

2.3 セッションの設定

CSP ベースの Web アプリケーションについては、以下のように **[セッションの設定]** を指定します。

セッションタイムアウト

既定のセッション・タイムアウトを秒単位で指定します。この値はプログラムによりオーバーライドできます。

ユーザがセッション中に Web アプリケーション間を移動しても、タイムアウト時間はそのセッションの間に使用された最初の Web アプリケーションによって制御されます。例えば、既定のタイムアウト値が 900 秒の Web アプリケーション A でセッションが開始し、その後、既定のタイムアウト値が 1800 秒の Web アプリケーション B に移動した場合、セッションは 900 秒後にタイムアウトになります。そのような場合、セッション・タイムアウトを更新するためにセッション・イベント・クラスを定義することができます。

イベントクラス

タイムアウトやセッションの終了など、Web アプリケーションを処理するクラスの既定名。

セッションにクッキーを使用する

アプリケーションでブラウザ・セッションを追跡するために cookie を使用するかどうかの設定。選択肢は以下のとおりです。

- **常時** – 既定。ブラウザ・セッションを追跡するために常に cookie を使用します。
- **なし** – ブラウザ・セッションを追跡するために cookie を使用しません。
- **自動検出** – クライアント・ブラウザで無効になっている場合を除き、ブラウザ・セッションを追跡するために cookie を使用します。ユーザが cookie を無効にしている場合、アプリケーションではブラウザ・セッションを追跡するために URL 書き換えが使用されます。

このオプションは、アプリケーションが cookie を使用するかどうかを設定するものではなく、ユーザの設定に従ってアプリケーションがどのようにセッションを管理するかを制御するものです。さらに、値が **[常時]** または **[自動検出]** であっても、アプリケーションは、そうするよう記述された固有のコードが存在する場合にのみ cookie を使用します。

セッションクッキーパス

このアプリケーションに関するセッション Cookie をブラウザから InterSystems IRIS に返送する際に使用する URL の一部。このフィールドの値を指定しない場合、アプリケーションは、**[名前]** フィールドの値の先頭と末尾にスラッシュを付けたものを既定のスコープとして使用します。したがって、ここで値を指定しない場合、`myapp` という名前のアプリケーションのスコープは `/myapp/` になります。

アプリケーションは、指定されたスコープ内にあるページの cookie のみを送信します。スコープを 1 つの Web アプリケーションで必要なページに制限すると、このマシン上の他の Web アプリケーションがこのセッション cookie を使用するのを防ぐことができます。また、この Web サーバ上の他の Web アプリケーションが cookie を参照するのも防ぐことができます。

1 つのセッション cookie を同時に共有しながら、プライマリ・アプリケーションとそのサブアプリケーションとで異なるセキュリティ設定を使用できます (すべてのアプリケーションがプライマリ・アプリケーションのパスを使用している場合)。

セッションCookieのスコープ

セッション cookie の SameSite 属性の既定値を制御します。

ユーザCookieスコープ

アプリケーション固有の cookie の SameSite 属性の既定値を制御します。

2.3.1 SameSite 属性について

SameSite 属性では、サードパーティ・アプリケーションに関連する Cookie をアプリケーションでどのように処理するかを指定します (クロスサイト・リクエストともいいます)。SameSite には、以下の値を設定できます。

- [None] – アプリケーションはクロスサイト・リクエストに応じて Cookie を送信します。SameSite の値が [None] の場合、ブラウザにより、アプリケーションで HTTPS 接続を使用するよう要求されることがあります。
- [Lax] – アプリケーションは、安全な最上位のクロスサイト・ナビゲーションを使用して Cookie を送信します。
- [Strict] – アプリケーションはクロスサイト・リクエストに応じて Cookie を送信することはありません(システム Web アプリケーション、および新規またはアップグレードしたユーザ・アプリケーションの既定値)。

CSP ページのクラス内で、この属性をプログラムによってオーバーライドできます。

SameSite 属性は IETF の計画の一環であり、IETF のいくつものドキュメントで扱われています。

2.4 CSP ファイルの設定

CSP サーバは CSP ページによって生成されたコンテンツを戻すことに加えて、静的ファイルを処理できます。CSP ベースの Web アプリケーションの場合、[CSP ファイルの設定] は、CSP サーバが静的ファイルを処理する方法を制御します。"CSP サーバで静的ファイルを処理する方法" も参照してください。

Web サーバから静的ページを提供する従来の構成を使用できます (この場合、ここで説明する設定は無関係です)。これは特定の状況では望ましい場合があります。例えば、1 つの Web サーバで複数のリモート InterSystems IRIS インスタンスを処理しているシステムがある場合、この Web サーバ マシンにとってローカルな共通の場所からこれらのファイルを提供する方が効率的である可能性があります。

ただし、静的ファイルを提供するよう Web サーバを構成することで問題が生じる可能性があることに注意してください。例えば、共通の Web サーバが異なるバージョンの InterSystems IRIS を提供する場合、同じファイルの 2 つの異なるバージョン間で競合が発生する可能性があります (ハイパーベントのブローカ・コンポーネントなど)。各 CSP サーバが独自のアプリケーションの静的コンテンツを提供する場合は、そのような競合は発生しません。また、Web サーバ自体で静的ファイルを提供するように構成する場合は、その静的コンテンツがシステム内のすべての Web サーバに存在するようにしてください。

静的ファイルの提供

[物理パス] によって指定されたディレクトリから静的ファイルを提供するかどうかを制御します。

- **いいえ** ― このアプリケーション・パスからファイルを提供しません。
- **常時** ― 既定。常にこのアプリケーション・パスからファイルを提供し、静的ファイルのこのパスの CSP セキュリティ設定を無視します。新規アプリケーションではこれが既定です。この設定には以前に Web サーバから静的ファイルが提供されていたアプリケーションとの後方互換性があります。
- **常時かつキャッシュ** ― 常にこのアプリケーション・パスからファイルを提供し、Web ゲートウェイがこれらのファイルをキャッシュして、InterSystems IRIS からファイルを要求せずに済むようにします。これは、導入されたアプリケーションで使用することが想定されるモードです。
- **CSPセキュリティ使用** ― このアプリケーションで CSP ページを表示する許可がある場合は、静的ファイルも表示することができます。ページを表示する許可がない場合は、「`404 page not found`」というメッセージが表示されます。

静的ファイルの提供タイムアウト

ブラウザが静的ファイルをキャッシュする時間の長さ (秒単位) を指定します。既定値は 3600 です。

物理パス

この Web アプリケーションへのファイルの提供元である InterSystems IRIS サーバ上のディレクトリ。パスは install-dir/**csp**/ ディレクトリを基準にした相対パスです。

タグベースの開発にのみ適用されるオプションである **[パッケージ名]**、**[デフォルトスーパークラス]**、**[再帰]**、**[自動コンパイル]**、および **[CSP名ロック]** は無視してください。(必要に応じて、Caché/Ensemble のドキュメントの "アプリケーションの作成および編集：[一般] タブ" を参照してください。)

2.5 カスタム・ページ

CSP ベースの Web アプリケーションについては、以下のように **[カスタム・ページ]** を指定します。

ログインページ

必要に応じて、CSP ページ・クラスの名前を指定します。この名前の前には Web アプリケーションのフル・パスを付ける場合があります。例えば、`/csp/user/MyApp.LoginPage.cls` のようになります。

通常、ログイン・ページはユーザが InterSystems IRIS にログインする前にロードされるため、要求プロセスは **CSPSystem** ユーザ (または CSP ゲートウェイを InterSystems IRIS に接続する任意のユーザ) の下で実行されます。その結果として、**CSPSystem** ユーザにはログイン・ページのコードをロードして実行するのに十分な特権が必要になります。これには通常ログイン・ページが配置されているデータベースを保護するリソースの READ 許可が含まれます。

パスワード変更ページ

必要に応じて、パスワードを変更する際に使用するページのクラス名を指定します。

カスタムエラーページ

必要に応じて、このアプリケーション内でページを生成するときにエラーが発生した場合に表示する CSP エラー・ページ・クラスの名前を指定します。

2.6 ページおよびクラスへのアクセスの有効化

Web アプリケーションからページとクラスへのアクセスには、以下の規則が適用されます。

1. 既定では、ユーザ・アプリケーションは以下のページへのアクセスが許可されます。
   - /csp/sys/ アプリケーションとそのすべてのサブアプリケーションのページが許可されます。
   - /isc/studio/templates/ および /isc/studio/usertemplates/ アプリケーションのページが許可されます。
2. 既定では、ユーザ・アプリケーションは現在のネームスペース内のすべての非 % クラスへのアクセスが許可されます。
3. また、ユーザ・アプリケーションは以下のクラスにもアクセスできます。
   - %CSP.Broker、%CSP.StreamServer、%CSP.Login、%CSP.PasswordChange、%CSP.PageLookup が許可されています。
   - %ZEN.SVGComponent.svgPage および %ZEN.Dialog.* は許可されていますが、以下の追加条件が伴います。
     - その他すべての %ZEN.* クラスは許可されません。
     - その他すべての %Z* クラスは許可されます。
   - すべての %z* クラスは許可されます。

Web アプリケーション内の設定のチェックに加えて、許可されるクラスのチェックが実行されます。

追加のクラスへのアクセスを許可するには、%SYS ネームスペースでグローバル ^SYS("Security","CSP","category") を構成します。ここで、category は AllowClass、AllowPrefix、または AllowPercent です。後続のセクションでは、これらのプロセスについて説明します。

重要　チェックは、まず既定のルールを適用し、その後リストされている順序でカテゴリを適用します。

また、各キーワードは複数回呼び出すことができます。つまり、パッケージ全体をアクセス可能にして、その後そのパッケージ内の 1 つのクラスにアクセスを制限することができます。

2.6.1 ^SYS グローバルについての背景情報

^SYS グローバルは %SYS ネームスペースで利用でき、構成情報が含まれています。最初にこのグローバルの関連する部分の現在のコンテンツを調べると役立つ場合があります。そのためには、ターミナルを開いて、%SYS ネームスペースに切り替えます。次に、以下のコマンドを入力します。

ObjectScript

```
zw ^SYS("Security", "CSP")
```

各ノードに 1 行が表示され、その現在の値が表示されます。以下に例を示します。

```
^SYS("Security","CSP")=1
^SYS("Security","CSP","AllowClass","/csp/samples/","%CSP.UI.Portal.About")=1
^SYS("Security","CSP","AllowClass","/csp/samples/","%SOAP.WebServiceInfo")=1
^SYS("Security","CSP","AllowClass","/csp/samples/","%SOAP.WebServiceInvoke")=1
^SYS("Security","CSP","AllowPrefix","/csp/samples/","%DeepSee.")=1
```

2.6.2 カテゴリ： AllowClass

アプリケーションが特定のクラスの呼び出しに依存している場合、AllowClass オプションを使用してそのクラスを利用可能にします。

重要　　アプリケーションが "ページおよびクラスへのアクセスの有効化" の最初に許可済みとしてリストされているクラス以外のクラスの呼び出しに依存している場合は、使用するのは安全ではない可能性があります。このクラスの呼び出しが必要かどうかを判断し、導入環境のリスク評価を実施して、そのクラスを利用可能にすることによる影響を把握することをお勧めします。

特定の Web アプリケーションが特定のクラスを呼び出せるようにするには、%SYS ネームスペースで以下のコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowClass", "web-app-name", "package.class") = value
```

以下はその説明です。

- web-app-name は、Web アプリケーションの名前です。Web アプリケーション名は小文字で、先頭と末尾の両方にスラッシュを付ける必要があります。

  すべての Web アプリケーションで特定のクラスまたはパッケージを使用できるようにするには、web-app-name を 0 に指定します。この場合、二重引用符で囲む必要はありません。

- package.class は、クラスの完全修飾名です。class を省略すると、指定されたパッケージのすべてのクラスが許可されます。

- value は 1 または 0 です。

  これに 1 を指定すると、Web アプリケーションはこのクラス (またはパッケージ) を呼び出すことができます。

  これに 0 を指定すると、Web アプリケーションはこのクラス (またはパッケージ) を呼び出せません。

例えば、/csp/webapps アプリケーションがクラス %User.Page を使用できるようにするには、以下のコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowClass", "/csp/webapps/", "%User.Page") = 1
```

または、すべての Web アプリケーションが %User.Page を使用できるようにするには、以下のコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowClass", 0, "%User.Page") = 1
```

別の例として、/csp/myapp アプリケーションが、%User パッケージ内の %User.Other クラスを除くすべてのクラスを使用できるようにするには、以下の 2 つのコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowClass", "/csp/myapp/", "%User") = 1
Set ^SYS("Security", "CSP", "AllowClass", "/csp/myapp/", "%User.Other") = 0
```

2.6.3 カテゴリ： AllowPrefix

アプリケーションが、同じ文字セットで始まる複数のクラスまたはパッケージの呼び出しに依存している場合、AllowPrefix オプションを使用します。

重要　　アプリケーションが上記でリストされているクラス以外のクラスの呼び出しに依存している場合は、使用するのは安全ではない可能性があります。このクラスの呼び出しが必要かどうかを判断し、導入環境のリスク評価を実施して、そのクラスを利用可能にすることによる影響を把握することをお勧めします。

特定の Web アプリケーションが、同じ文字セットで始まるクラスまたはパッケージを呼び出せるようにするには、%SYS ネームスペースで以下のコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowPrefix", "web-app-name", "prefix") = value
```

以下はその説明です。

- web-app-name は、Web アプリケーションの名前です。Web アプリケーション名は小文字で、先頭と末尾の両方にスラッシュを付ける必要があります。

  すべての Web アプリケーションが特定のクラスまたはパッケージを使用できるようにするには、web-app-name を 0 に指定します。この場合、引用符で囲む必要はありません。

- prefix は、名前の接頭語です。

- value は 1 または 0 です。

  これに 1 を指定すると、Web アプリケーションはこれらのクラス (またはパッケージ) を呼び出すことができます。

  これに 0 を指定すると、Web アプリケーションはこれらのクラス (またはパッケージ) を呼び出せません。

例えば、/csp/webapps アプリケーションが MyApp パッケージ全体を呼び出せるようにするには、以下のコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowPrefix", "/csp/webapps/", "MyApp.") = 1
```

prefix は "MyApp." であることに注意してください。接頭語にピリオドが含まれるため、Web アプリケーションは MyAppUtils のようなパッケージにはアクセスできません。しかし、Web アプリケーションは MyApp.Utils や MyApp.UnitTests のようなパッケージにはアクセスできます。

別の例として、すべてのアプリケーションが My で始まるすべてのパッケージにアクセスできるようにするには、以下のコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowPrefix", 0, "My") = 1
```

さらに別の例として、/csp/myapp アプリケーションが %MyPkg パッケージ内の %MyPkg.Class1 クラスを除くすべてのクラスにアクセスできるようにする必要があるとします。その場合、以下のコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowClass", "/csp/myapp/", "%MyPkg.Class1") = 0
Set ^SYS("Security", "CSP", "AllowPrefix", "/csp/myapp/", "%MyPkg.") = 1
```

2.6.4 カテゴリ : AllowPercent

アプリケーションが、通常 % 文字で始まるパッケージの呼び出しに依存している場合、AllowPercent オプションによってこれらのクラスが利用可能になります。

重要　　アプリケーションが上記でリストされているクラス以外のクラスの呼び出しに依存している場合は、使用するのは安全ではない可能性があります。このクラスの呼び出しが必要かどうかを判断し、導入環境のリスク評価を実施して、そのクラスを利用可能にすることによる影響を把握することをお勧めします。

すべての Web アプリケーションが、% 文字で始まるすべてのパッケージを使用できるようにするには、%SYS ネームスペースで以下のコマンドを使用します。

```
Set ^SYS("Security", "CSP", "AllowPercent") = 1
```

注釈　　または、値 0 を使用して、すべての Web アプリケーションがこれらのパッケージにアクセスすることを明示的に禁止します。

2.7 CSP サーバで静的ファイルを処理する方法

Web アプリケーションが静的ファイルを提供するよう構成されている場合、CSP サーバ (特にストリーム・サーバ・コンポーネント) は静的ファイルを処理します。CSP サーバは、ファイル拡張子を使用して以下を特定します。

- ファイル・タイプ (MIME タイプ)
- ファイルがバイナリ・ファイルであるかどうか
- ファイルの文字エンコード (該当する場合)

JavaScript ファイルの場合、ストリーム・サーバは、以下に説明する主要な Web サーバと同じ方法で文字エンコードを決定します。必要に応じて既定の動作をオーバーライドできます。

2.7.1 JavaScript ファイルの文字エンコード

最近の慣例では、すべての JavaScript ファイルが `application/javascript` のコンテンツ・タイプとしてマークされます。JavaScript ファイルがこのようにマークされている場合、

- ファイルに BOM (バイト・オーダー・マーク) が含まれていれば、これをブラウザが自動的に検出し、適切な文字セットを利用してこれを読み取ります。
- ファイルに BOM が含まれていなければ、ブラウザはこのファイルを UTF-8 と見なします。

この動作をオーバーライドして、JavaScript ファイルに文字セットを指定する必要がある場合は、グローバル ^%SYS("CSP","MimeFileClassify","JS") をリスト値 $listbuild(contenttype, binary, charset) に設定します。以下に例を示します。

```
Set list=$listbuild("text/javascript", 0 ,"ISO-8859-1")
Set ^%SYS("CSP", "MimeFileClassify", "JS") = list
```

これにより、古い Content-Type が設定され、ISO-8859-1 文字セットが使用されます。また、既定の InterSystems IRIS 変換テーブルが空の文字列以外に定義されている場合、またはグローバル・ノード ^%SYS("CSP","DefaultFileCharset") が NULL 値に設定されている場合、InterSystems IRIS はすべての JavaScript およびその他のテキスト・ファイルにこの文字セットを使用します。既定では、これらのグローバル・ノードはどちらも設定されていません。

3

CSP ページ・クラスの作成

CSP ページ・クラスは要求を処理して HTML を生成し、その HTML をブラウザに直接書き込みます。CSP ベースの Web アプリケーションを作成する場合、主なタスクはページ・クラスの OnPage() コールバック・メソッドを定義することです。

3.1 基本情報

CSP ページ・クラスを作成する手順は以下のとおりです。

1. **%CSP.Page** のサブクラスを作成します。
2. このサブクラス内に、OnPage() コールバック・メソッドを実装します。このメソッドは、以下の変数にアクセスできます。
   - %request: 要求についての情報を持つプロパティが含まれます。この変数は **%CSP.Request** のインスタンスです。
   - %session: ブラウザ・セッションについての情報が含まれます。この変数は **%CSP.Session** のインスタンスです。
   - %response : 既定の HTTP 応答についての情報が含まれます。この変数は **%CSP.Response** のインスタンスです。

   メソッドでは必要に応じてその情報を調べ、HTML ページを文字列として生成し、Write コマンドを使用してその文字列を書き込む必要があります。InterSystems IRIS® データ・プラットフォームは、標準の出力デバイス ($IO) を自動的にリダイレクトして、すべての Write 出力が HTTP クライアントに送り返されるようにします。

   %CSP.Page は、HTML および JavaScript コンテキストで使用する文字列をエスケープおよびエスケープ解除するために使用できる便利なクラス・メソッドを提供します。

   他のアプローチについては、"特殊な HTML 指示文" も参照してください。
3. オプションで、**%CSP.Page** のパラメータをオーバーライドします。これにより、以下を制御できるようになります。
   - クライアントに送信される既定の応答ヘッダ
   - ページへのアクセス
   - 暗号化
   - 特定のシナリオで表示されるエラー・ページ

   使用可能なすべてのクラス・パラメータのリストは、"**%CSP.Page**" のドキュメントを参照してください。
4. オプションで、他のコールバック・メソッドをオーバーライドすると、別のときに処理を制御できます。

以下に例を示します。

Class Definition

```
Class GCSP.Basic Extends %CSP.Page
{

ClassMethod OnPage() As %Status
{
   Set html="<!DOCTYPE html>"
           _"<html lang=""en"" dir=""ltr"">"
           _"<body>"
           _"<h1>Basic Page</h1>"

   // examine %request object and
   // write more output...

   Set html=html_"</body>"
                _"</html>"

   Write html //finally, write the page
   Quit $$$OK
}

}
```

この例では、ページ全体を単一の文字列として連結してから書き込んでいます。これは、長い文字列長の制限を下回ることが予想されるページでは有効です。以下のバリエーションによっても、同じ HTML が生成されます。

Class Definition

```
Class GCSP.Basic Extends %CSP.Page
{

ClassMethod OnPage() As %Status
{
   write "<!DOCTYPE html>"
           _"<html lang=""en"" dir=""ltr"">"
           _"<body>"
           _"<h1>Basic Page</h1>"

   // examine %request object and
   // write some more html...

   write "</body>"
         _"</html>"

   Quit $$$OK
}

}
```

全体的な外観が同じである必要のある複数のページが存在する場合、ページ開始、HTML メタデータを含む <head>、JavaScript、スタイル・シートへのリンク、ナビゲーション 、フッタ、ページ終了などを構築するのに、テスト可能なヘルパー・メソッドを個別に使用することをお勧めします。

3.2 既定の応答ヘッダの制御

CSP ページ・クラスのクラス・パラメータは、クライアントに送信される既定の HTTP 応答ヘッダを決定します。

ブラウザに返されるコンテンツのタイプを制御するには、CONTENTTYPE クラス・パラメータを指定します。以下に例を示します。

Class Member

```
Parameter CONTENTTYPE = "application/vnd.ms-excel";
```

同様に、使用する文字セットを制御するには、CHARSET クラス・パラメータを指定します。

Class Member

```
Parameter CHARSET = "UTF-8";
```

使用可能なすべてのクラス・パラメータのリストは、"%CSP.Page" のドキュメントを参照してください。

また、次に説明するように、OnPreHTTP() コールバック内の %response オブジェクトのプロパティを設定することで、応答の HTTP ヘッダも制御できます。

3.3 その他のコールバック

InterSystems IRIS® データ・プラットフォームで要求を処理する %CSP.Page クラスを決定する際、そのクラスの Page() メソッドが呼び出され、このメソッドによりこれらのコールバック・メソッドが以下の順序で呼び出されます。

1. OnPreHTTP()
2. 上記の OnPage()
3. OnPostHTTP()

3.3.1 OnPreHTTP()

応答の HTTP ヘッダをきめ細かく制御するには、OnPreHTTP() を実装できます。

具体的には、選択した既定とは異なるものが必要な場合、ContentType プロパティなどの %response オブジェクトのプロパティを設定できます。

```
set %response.ContentType="text/html"
```

その後、InterSystems IRIS は、クライアントに書き込む際にこれらの値を使用します。

また、OnPreHTTP() メソッドでは、必要に応じて、リダイレクトを定義することもできます。以下のように Redirect プロパティを設定することで、通常のクライアント側のリダイレクトを作成できます。

```
set %response.Redirect="https://someotherurl"
```

または、別の CSP ページを呼び出す場合は、代わりに ServerSideRedirect プロパティを設定できます。これにより、CSP サーバで指定されたクラス内の Page() メソッドが呼び出されます。

```
set %response.Redirect="GCSP.OtherPage.cls"
```

サーバ側リダイレクトによって、ブラウザで表示される URL が変更されることはありません。

3.3.2 OnPostHTTP()

OnPostHTTP() メソッドは、HTTP 要求の処理が完了した後に任意の操作を実行する場所として提供されます。

3.4 リンクのベスト・プラクティス

HTML <A> アンカー・リンクを含める際は、%CSP.Page の Link() メソッドを使用して URL を作成します。このメソッドは URL エスケープを実行し、該当する場合には URL パラメータの暗号化も行います (ターゲット・ページの定義による)。

このメソッドには、以下のシグニチャがあります。

```
classmethod Link(link As %String, ByRef query As %String,
                 addQ As %Boolean = 0) as %String
```

以下はその説明です。

- link は、ベース URL です。
- query は任意の URL パラメータを含む多次元配列です。
- addQ は、戻り値の最後に ? または & を（必要に応じて）含めるかどうかを指定するブーリアン値です。このオプションによって、追加のクエリ・パラメータを付加できます。

以下に例を示します。

ObjectScript

```
 Set origurl="GCSP.EncryptPage2.cls"
 Set urlparms("SAMPLEPARM")="sample value"
 Set tURL = ##class(%CSP.Page).Link(origurl,.urlparms)
 Set html="<p>Link to page 2: <a href="""_tURL_""">Link</a>"_"</p>"
```

%response オブジェクトの Context プロパティを使用して、すべてのリンクとフォームに自動的に値を挿入することもできます。詳細は、クラス・リファレンスを参照してください。

3.5 タグベースの開発（従来のアプリケーション）

従来のアプリケーションにも .csp ファイルを含めることができます。これは、タグベースの開発で使用されます。タグベースの開発モデルでは、開発者は Web アプリケーションによりアクセスされるディレクトリ構造に含まれる .csp ファイルを作成します。ファイルには、サーバとの通信を可能にする HTML タグと特殊タグの組み合わせが含まれています。CSP コンパイラはファイルを読み取り、それらからクラス定義を生成します。次いでクラス定義は実際のランタイム HTML を生成します。タグベースの開発の詳細は、Caché/Ensemble ドキュメントの "Tag-based Development with CSP" を参照してください。

4

要求の検証

CSP サーバは、HTTP 要求に応答する際、受信要求に関する情報を %request オブジェクトにパッケージ化します。これは、すべての CSP ページで利用できます。この変数は **%CSP.Request** のインスタンスです。

CSP ページでは、%session にもアクセスできます。このオブジェクトを使用して、ページからページに追加のデータを渡すことができます。

4.1 URL

受信 HTTP 要求の URL (クエリ文字列は含まない) を見つけるには、%request オブジェクトの **URL** プロパティを使用します。

```
Write "URL: ", %request.URL
```

4.2 URL パラメータ

URL にはパラメータのリスト (URL クエリとも呼ばれる) が含まれることがあります。%request オブジェクトは、**Data** プロパティを介してこれらのパラメータを利用可能にします。

例えば、受信 URL に以下が含まれているとします。

```
/csp/user/MyPage.csp?A=10&a=20&B=30&B=40
```

以下を使用して、サーバ上でこれらのパラメータを取得できます。

```
Write %request.Data("A",1)    // this is 10
Write %request.Data("a",1)    // this is 20
Write %request.Data("B",1)    // this is 30
Write %request.Data("B",2)    // this is 40
```

Data は多次元プロパティで、その中に格納される各値には、パラメータ名とパラメータのインデックス番号の 2 つの添え字があります (パラメータは上記の **B** のように 1 つの URL 内で複数回出現する場合があります)。このパラメータ名は大文字と小文字を区別しますので、注意してください。

また、受信 HTTP 要求が **GET** 要求であるか **POST** 要求であるかは重要ではないことにも注意してください。**Data** プロパティはどちらの場合もまったく同じようにパラメータ値を表します。

ObjectScript $Data 関数を使用して、指定のパラメータ値が定義されているかどうかをテストできます。

```
If ($Data(%request.Data("parm",1))) {
}
```

パラメータを参照したいが、それが定義されているかどうかわからない場合は、ObjectScript $Get 関数を使用できます。

```
Write $Get(%request.Data("parm",1))
```

以下のように %request オブジェクトの Count メソッドを使用して、特定のパラメータ名にいくつの値が定義されているかを調べることができます。

```
For i = 1:1:%request.Count("parm") {
   Write %request.Data("parm",i)
}
```

パラメータ値が暗号化されている場合でも、同じ手法を使用できます。

4.3 フォーム・データ

CSP サーバは、フォーム送信要求を受け取ると、すべての名前/値のペアを %request オブジェクトの多次元 Data プロパティに配置します。ここで、Data の添え字は名前です。

例えば、以下のようなフォームの場合、

```
<form name="edit" method="post" action="GCSP.EncryptPage3.cls">
<p>Value A: <input type="text" name="inputA"></p>
<p>Value B: <input type="text" name="inputB"></p>
<p><input type="submit" value="Submit This" ></p>
</form>
```

サーバ上で、以下を使用して値 A と値 B の入力値を取得します。

```
set myvalueA=$GET(%request.Data("inputA",1))
set myvalueB=$GET(%request.Data("inputB",1))
```

InterSystems IRIS では、文字列長の制限を超える値を受け取ると、自動的にストリーム (%CSP.CharacterStream のインスタンス) を作成し、そのストリームに値を書き込み、実際の値の代わりにストリーム OREF を Data プロパティに配置します。つまり、非常に長い文字列を受け取る可能性がある場合は、コードによってその値を検証してそれが OREF であるかどうかを確認し、それに従って処理を行う必要があるということです。

ObjectScript

```
Set value=%request.Data("fieldname",1)
If $isobject(value) {
   ; Treat this as a stream
} Else {
   ; Treat this as a regular string
}
```

4.4 CGI 変数

Web サーバは CGI (Common Gateway Interface) 環境変数と呼ばれる一連の値を提供します。この変数には HTTP クライアントと Web サーバについての情報が含まれます。これらの CGI 環境変数の値にアクセスできるようにするには、%request オブジェクトの多次元プロパティ CgiEnvs を使用します。これは、Data プロパティと同じ方法で使用できます。

例えば、どのようなタイプのブラウザが HTTP 要求を行っているかを特定するには、以下のように CGI 環境変数 HTTP_USER_AGENT の値を調べます。

```
Write %request.CgiEnvs("HTTP_USER_AGENT")
```

利用可能な CGI 環境変数の詳細は、"CGI Environment Variables" を参照してください。

4.5 MIME データ

受信要求には、MIME (Multipurpose Internet Mail Extensions) データが含まれる場合があります。これは一般的に、ファイルのようなサイズの大きな情報に使用されます。MIME データは、%request オブジェクトの多次元プロパティ MimeData を使用して取得できます。

5
セッションの管理

セッションとは、特定の期間に及ぶ特定のクライアントから特定のアプリケーションへの一連の要求を表します。InterSystems IRIS® データ・プラットフォームは、セッションの追跡を自動的に提供します。CSP ベースの Web アプリケーション内で、%CSP.Session のインスタンスである %session オブジェクトを調べることで、現在のセッションについての情報にアクセスできます。

アプリケーション間での認証セッションやデータの共有の詳細は、"認証の共有方法" を参照してください。"セッションの終了" も参照してください。

5.1 セッションの作成

セッションは、HTTP クライアントが Web アプリケーションに最初の要求を行うと開始されます。その後、InterSystems IRIS サーバは以下を実行します。

1. 新しいセッション ID 番号を作成します。
2. 必要に応じてライセンスのチェックを実行します。
3. %session オブジェクト (永続) を作成します。
4. セッション・イベント・クラスの OnStartSession() メソッドを呼び出します (存在する場合)。
5. セッション期間中に HTTP クライアントから後続の要求を追跡するためのセッション Cookie を作成します。クライアント・ブラウザで Cookie を無効にしている場合、InterSystems IRIS はセッションを追跡するために自動的に URL 書き換え (各 URL に特殊な値を挿入する) を使用します。

5.2 基本プロパティ

セッションの最初の要求で、%session オブジェクトの NewSession プロパティは、1 に設定されます。後続のすべての要求では、このプロパティは 0 に設定されます。

```
If (%session.NewSession = 1) {
   // this is a new session
}
```

%session オブジェクトの SessionId プロパティには、セッションの一意の識別子が含まれます。

5.3 セッション・データの管理

%session オブジェクトは、Cookie や URL パラメータを使用することなくセッション全体でアプリケーション固有のデータを保存するための簡単な方法を提供します。特に、このオブジェクトの Data プロパティは、コードで使用するための多次元配列プロパティです。例えば、以下の一連のキーと値のペアを保存するとします。

キー	値
product	"widgets"
quantity	100
unitofmeasure	"cases"

このためには、以下のように Data プロパティに値を保存します。

ObjectScript

```
set %session.Data("product")="widgets"
set %session.Data("quantity")=100
set %session.Data("unitofmeasure")="cases"
```

同様に、おそらくアプリケーションの異なるページで、以下のように値を取得できます。

ObjectScript

```
set productodisplay=$GET(%session.Data("product"))
set quantityodisplay=$GET(%session.Data("quantity"))
set uomtodisplay=$GET(%session.Data("unitofmeasure"))
```

この例では、$GET を使用して、特定の添え字が定義されていないときにエラーを防止していることに注意してください。そのようなエラーを防止するには、$GET (または $DATA) を使用するのがベスト・プラクティスです。

また、この手法ではリテラル・データ (オブジェクト参照ではなく) のみを格納でき、配列内の各値は文字列長の制限を下回る必要があることにも注意してください。

5.4 セッション・データの削除

Data プロパティからデータを削除するには、以下のように ObjectScript Kill コマンドを使用します。

```
Kill %session.Data("MyData")
```

5.5 セッション処理のカスタマイズ (イベント・クラス)

さまざまなセッション・イベントが発生したときの動作をカスタマイズするには、以下の操作を実行します。

1. セッション・イベント・クラスを定義します。これは、%CSP.SessionEvents のサブクラスである必要があります。
2. そのクラスで、コールバックを実装して、セッションの開始時、タイムアウト時、またはその他のイベントの発生時のアプリケーションの動作をカスタマイズします。

   コールバック・メソッドには以下のようなものがあります。

- OnStartSession() – セッションの開始時の動作を制御します。
- OnSessionEnd() – セッションの終了時の動作を制御します。すべてのセッションの終了時に呼び出されます。
- OnTimeout() – セッションのタイムアウトの発生時の動作を制御します。タイムアウトの場合にのみ呼び出されます。
- OnApplicationChange() – セッション内でユーザがあるアプリケーションから別のアプリケーションに移動したときの動作を制御します (そのような場合は、セッション・タイムアウト値を更新できます)。

詳細は、"%CSP.SessionEvents" を参照してください。

3. このセッション・イベント・クラスを使用するよう Web アプリケーションを構成します。

   または、アプリケーション・コード内で、%session オブジェクトの EventClass プロパティを設定することで、セッション・イベント・クラスをプログラムによって指定します。

5.6 コンテキストの保持

既定では、ある要求から次の要求まで CSP サーバによって保持される処理コンテキストのみが、%session オブジェクト内に保持されます。CSP サーバは、要求間ですべての処理コンテキスト変数、インスタンス化されたオブジェクト、データベース・ロック、開いているデバイスを保持するメカニズムを提供します。これはコンテキスト保持モードと呼ばれます。%session オブジェクトの Preserve プロパティの値を設定することで、いつでも CSP アプリケーション内でコンテキスト保持をオンまたはオフにできます。あるプロセスを 1 つのセッションに結び付けると、スケーラビリティが失われるため、このオプションを使用することは非常にまれであることに注意してください。

6

セッションの終了

InterSystems IRIS® データ・プラットフォームの CSP ベースの Web アプリケーション内で、セッションは、ユーザがログアウトした場合、サーバがセッションを明示的に終了した場合、またはセッションがタイムアウトした場合に終了することがあります。

6.1 ログアウト・オプションの提供

標準的な方法としては、ユーザがログアウトできるようにするリンクまたはボタンを提供します。

このリンクかボタンを、アプリケーションのホーム・ページにリンクするように定義し、リンクの URL に `IrisLogout=end` を含めることをお勧めします。これで、このサーバはホーム・ページの実行を試みる前に、現在のセッションを終了します。

6.2 サーバにセッションを終了させる

アプリケーション内から、以下の方法でセッションを明示的に終了できます。

- セッションを終了する (クライアントが停止している場合や、新しいサイトに移動する場合など) :

  ObjectScript

  ```
  set %session.EndSession=1
  ```

- ユーザをログアウトする :

  ObjectScript

  ```
  do %session.Logout()
  ```

これらの手法では、サーバ上で利用可能な %session オブジェクトを使用します。これは、%CSP.Session のインスタンスです。

6.3 セッション・タイムアウト

セッション・タイムアウトでは、指定されたセッション・タイムアウト期間内に要求を受信しなかった場合にセッションが終了します。

既定では、セッション・タイムアウトは 900 秒 (15 分) に設定されています。これは、Web アプリケーション定義により制御されます。

6.3.1 プログラムによるタイムアウトの変更

アプリケーション内から、%session オブジェクトの AppTimeout プロパティを設定することで、タイムアウトを変更できます。以下に例を示します。

ObjectScript

```
Set %session.AppTimeout = 3600 // set timeout to 1 hour
```

セッション・タイムアウトを無効にするには、タイムアウト値を 0 に設定します。

セッションが存続期間中に Web アプリケーションを変更しても、セッションの移動先のアプリケーションで定義されている既定のタイムアウトに従ってタイムアウト値が更新されることはありません。例えば、既定のタイムアウト値が 900 秒の Web アプリケーション A でセッションが開始し、その後、既定のタイムアウト値が 1800 秒の Web アプリケーション B に移動した場合、セッションは 900 秒後にタイムアウトになります。

アプリケーションの変更により、セッション・タイムアウトが新しいアプリケーションに合わせて更新されるようにするには、セッション・イベント・クラスを定義します。そのクラスで、OnApplicationChange() コールバック・メソッドをオーバーライドして、%session オブジェクトの AppTimeout プロパティの更新を処理するコードを追加します。

6.4 終了動作のカスタマイズ

セッションの終了時の動作をカスタマイズするには、セッション・イベント・クラスを定義して、そのクラスの OnEndSession() コールバック・メソッドを実装します。

同様に、セッション・タイムアウト時の動作をカスタマイズするには、セッション・イベント・クラスを定義して、そのクラスの OnTimeout() コールバック・メソッドを実装します。

6.5 セッション終了の詳細

セッションが終了すると、サーバは永続的な %CSP.Session オブジェクトを削除し、必要に応じてセッションのライセンス・カウントをデクリメントします。

また、既存のセッション・データを削除し、セッションのセキュリティ・コンテキストも削除します。

セッションがタイムアウトまたはサーバ・アクションによって終了した場合は、サーバはセッション・イベント・クラスの OnEndSession() メソッドも呼び出します (存在する場合)。

7

Cookie の保存と使用

このページでは、CSP ベースの Web アプリケーション内での Cookie の保存および使用方法について説明します。

Cookie は、クライアント・ブラウザ内に格納される名前と値のペアです。クライアントからの後続の要求すべてには、それより前のすべての Cookie の値が含まれます。

Cookie 内に情報を格納すると、セッションの終了後に覚えておきたい情報がある場合に便利です (既定ではブラウザが閉じると Cookie は終了するため、これを行うには有効期限を設定する必要があります)。例えば、Cookie にユーザ名を保存しておくと、後続のセッションでこの情報を再度入力する必要がなくなります。

7.1 Cookie の保存

Cookie を保存するには、以下の例に示すように %responseの SetCookie() メソッドを使用します。

ObjectScript

```
Do %response.SetCookie("UserName",name)
```

Cookie 定義には、以下の形式で有効期限とパスを含めることができます。

ObjectScript

```
Do %response.SetCookie("NAME","VALUE",expireData,path)
```

空白の expireData フィールドは、メモリ内の Cookie を定義します (現在のセッションでのみ利用できます)。ただし、expireData フィールドに値を指定すると、これは指定された日時に削除される永続的な Cookie になります。expireData フィールドの形式は、`Wdy, DD-Mon-YYYY HH:MM:SS GMT` です。例えば、`Wednesday, 24-Mar-2024 18:12:00 GMT` のようになります。

詳細は、クラス・リファレンスの "%CSP.Response" を参照してください。

7.1.1 SameSite 属性

Cookie の作成時に、SameSite 引数を指定できます。この引数は、サードパーティ・アプリケーションに関連する Cookie をアプリケーションでどのように処理するかを指定します (クロスサイト・リクエストともいいます)。この引数は、Web アプリケーションによって指定される既定の SameSite の値をオーバーライドします。

Cookie の SameSite の値を [None] に指定する場合は、HTTPS 接続を使用する必要があります。

7.2 Cookie へのアクセス

Cookie は、%requestの Cookies プロパティにあります。このプロパティは多次元プロパティで、その添え字が Cookie の名前です。

%request オブジェクトは、Cookie のカウントや反復処理を行う方法も提供します。"%CSP.Request" の "GetCookie()"、"NextCookie()"、および "CountCookie()" を参照してください。例えば、以下の単純なページ・クラスは、すべての Cookie とその値を表示します。

Class Definition

```
Class Sample.CookieDemo Extends %CSP.Page
{

ClassMethod OnPage() As %Status
{
   Set html="<!DOCTYPE html>"
            _"<html lang=""en"" dir=""ltr"">"
            _"<body>"
            _"<p>COOKIES:</p>"
            _"<ul>"

   Set cookie=%request.NextCookie("")
   While cookie'="" {
      For count=1:1:%request.CountCookie(cookie) {
         Set html=html_"<li>"_cookie_" - "
                     _..EscapeHTML(%request.GetCookie(cookie,count))
                     _"</li>"
       }
       Set cookie=%request.NextCookie(cookie)
   }
   Set html=html_"</ul>"
               _"</body>"
               _"</html>"

   Write html
   Quit $$$OK
}

}
```

8

再ロードなしのページの更新

最新のブラウザはすべて、サーバからデータを要求するための組み込みの XMLHttpRequest オブジェクトを備えています。このオブジェクトによって、ページの読み込み後にサーバと対話し、再ロードすることなくページを更新することが可能になります。

InterSystems IRIS® データ・プラットフォームでは、CSP ページ内でこのオブジェクトを使用して、CSP サーバと直接通信するための簡単なシステムを提供します。

8.1 基本情報

このシステムには、以下の 3 つの部分があります。

- HTML ページの <head> 部分を生成する際に、HyperEventHead() を呼び出します。これにより、このシステムで必要とされる 2 つの連続した <script> 要素で構成される文字列が返されます。メソッド・シグニチャは以下のとおりです。

  ```
  classmethod HyperEventHead(iframeOnly As %Boolean,
                             strict As %Boolean = 0) as %String
  ```

  以下はその説明です。

  - iFrameOnly 引数は無視されますが、互換性のために存在しています。
  - strict が 1 の場合、返される文字列では厳密な HTML 4 形式の <script> タグが使用されます。

- 呼び出しを行う (サーバ側の) メソッドを作成します。このメソッドはインスタンス・メソッドである必要があり、リテラル値 (OREF ではない) を返すことができます。参照渡しにより、または出力として引数を渡すことはできません。

  Tipヒント XMLHttpRequest を使用する一般的な理由はページを変更することであるため、このメソッドがページに追加する完全に形成された HTML を返すと役に立つ場合があります。

- 該当する場合は、ページの HTML の一部として、HyperEventCall() を使用してメソッドを呼び出し、その呼び出しの結果を使用する JavaScript 関数を定義してください。

  HyperEventCall() メソッドには、以下のシグニチャがあります。

  ```
  classmethod HyperEventCall(methodName As %String,
                             args As %String,
                             type As %Integer = 0) as %String
  ```

  以下はその説明です。

- methodName はサーバ・メソッドに対する参照で、メソッドが同じクラスに存在する場合は `..MethodName()` の形式、存在しなければ、より長い `Package.Class.MethodName` の形式を取ります。
- args には、メソッドに渡すすべての引数が含まれます。args は、変数のコンマ区切りリストを含む引用符で囲まれた文字列です (JavaScript 関数内で定義済み)。
- type が 1 の場合 (推奨)、呼び出しは非同期になります。

HyperEventCall() は、サーバ・メソッドにより返される値を返します。

8.2 例

これらがどのように組み合わされるかを例で示します。このシナリオでは、ツリー・コントロールの追加部分を取得する関数を定義します。この関数は、%CSP.Documatic.Helper という名前のクラスの GetChildren() メソッドを使用します。このメソッドは、name、parent、および ns の 3 つの文字列引数を取り、ツリー・コントロール内のページに追加するための完全に形成された HTML を返します (ここでは、このツリー・コントロールが具体的にどのように機能するか、またはこのメソッドが厳密に何を行うかということは重要ではありません)。

この HTML ページの <head> を生成するコード内で、以下のように HyperEventHead() と HyperEventCall() の両方を呼び出します。

```
//Add hyperevent-related scripts for left navigation
set headhtml=_..HyperEventHead()
 _"<script>function addChildrenAfter(item,name,Id,ns) {"
 _"var h="_..HyperEventCall("%CSP.Documatic.Helper.GetChildren","name,Id,ns",1)_";"
 _"if (h!==null) {"
 _"item.insertAdjacentHTML('afterend',h); } else {"
 _"location.reload();}"
 _"return false;"
 _"}</script>"
```

結果として生成される HTML は以下のようになります (読みやすくするため改行が追加され、編集されています)。

```
<script type="text/javascript" src="/somelocation/csp/broker/cspxmlhttp.js">
</script>
<script type="text/javascript" src="/somelocation/csp/broker/cspbroker.js">
</script>
<script>function addChildrenAfter(item,name,Id,ns)
{var h=cspHttpServerMethod("pyK473ekNn0...very long...DOKepQ",name,Id,ns);
if (h!==null) {item.insertAdjacentHTML('afterend',h);
 } else {location.reload();
}return false;}</script>
```

この HTML にはサーバ・メソッドの名前は含まれませんが、代わりにサーバが実行するコードの特定に使用する長いトークンが含まれます。また、この HTML には `cspHttpServerMethod` も含まれることに注意してください。これは、HyperEventHead() により提供される JavaScript 関数です。

9

エラー処理

すべての Web アプリケーションに対して、InterSystems IRIS® データ・プラットフォームは、アプリケーション・エラーの発生時にユーザにメッセージを表示する既定のエラー・ページを提供します。この代わりに、独自のカスタム・エラー・ページを提供することもできます。エラーが発生して、ライセンスが取得されていない (そのためサーバ・コードを実行できない) 場合の特別なオプションがあります。

このページでは、ユーザが ObjectScript のエラー処理とエラー・ログに精通していることを前提としています。

9.1 カスタム・エラー・ページの追加

カスタム・エラー・ページを追加するには：

1. %CSP.Error のサブクラスを作成し、OnPage() コールバック・メソッドをカスタマイズします。

   このクラスでは、%request、%response、および %session オブジェクトを通常通り使用できます。

   特に、%request.Get("Error:ErrorCode") には、エラー情報が含まれています。DecomposeError() を使用して、以下のようにエラーのテキストを含む多次元配列を取得します。

   ```
   Do ..DecomposeError(%request.Get("Error:ErrorCode"),.ErrorInfo)
   ```

   次に、以下のように ErrorInfo 変数 (参照渡しで返される) をループ処理できます。

   ```
   For i=1:1:ErrorInfo {
        if (i=1) {
             set return="<p>"_ErrorInfo(i,"Desc")_"</p>"
        } else {
             set return=return_$CHAR(13,10)_"<p>"_ErrorInfo(i,"Desc")_"</p>"
        }
    }
   ```

   この例では、HTML に含まれるように文字列を構築します (エラー・メッセージごとに 1 つのパラグラフとなっています)。

   エラーから情報をプルするための追加の方法については、"%CSP.Error" も参照してください。

2. オプションとして、ライセンスが付与されていないときにカスタム・ページを表示するには、このクラスのパラメータをオーバーライドしてください。

3. このエラー・クラスを使用するよう Web アプリケーションを構成します。

9.2 ライセンスが付与される前のエラーの処理

Web アプリケーションにまだライセンスが付与されておらず、エラーが発生した場合、InterSystems IRIS は標準の Web の「HTTP/1.1 404 Page Not Found」エラー・メッセージが既定で表示されます。

エラー・ページ・クラスで以下のパラメータを指定することで、そのような場合にエラーが発生したときに表示されるページの内容を変更できます。

LICENSEERRORPAGE

このパラメータは、ライセンスが付与されないときに InterSystems IRIS が実行する処理を制御します。パラメータには以下の 2 つの値のいずれかを指定できます。

- "" または NULL ー「HTTP/1.1 404 Page Not Found」エラーを返します (既定)。
- 静的 HTML ファイルへのパス ー /csp/myapp/static.html などの名前付きファイルを表示します。

PAGENOTFOUNDERRORPAGE

このパラメータは、以下のいずれかのエラーが発生した場合に InterSystems IRIS が実行する処理を制御します。

- ライセンスを付与できない
- クラスが存在しない
- メソッドが存在しない
- Web アプリケーションが存在しない (既定のエラー・ページでパラメータを設定)
- CSP ページが存在しない
- ファイルが存在しない
- ネームスペースが存在しない
- 不正な要求
- ファイルを開けない
- セッション・タイムアウト

パラメータには以下の値のいずれかを指定できます。

- "" ー「HTTP/1.1 404 Page Not Found」エラーを返します (既定)。
- 1 ー ライセンスを取得し、標準のエラー・ページを表示します。
- 静的 HTML ファイルへのパス ー /csp/myapp/static.html などの名前付きファイルを表示します。

OTHERSTATICERRORPAGE

このパラメータは、その他のエラーが発生した場合に InterSystems IRIS が実行する処理を制御します。パラメータには以下の値のいずれかを指定できます。

- "" ー ライセンスを取得し、標準のエラー・ページを表示します (既定)。
- 1 ー「404 Page not found」エラーを出力します。
- 静的 HTML ファイルへのパス ー /csp/myapp/static.html などの名前付きファイルを表示します。

10

CSP ページへのアクセスの制御

認証の追加（ここでは明示的には説明していません）に加えて、ページをプライベートにして、ページを使用する権限を要求することができます。

これらのオプションは、CSP ページが実行される Web アプリケーションのセキュリティ設定と組み合わせて使用できます。

10.1 ページをプライベートにする

ページをプライベートにした場合、ユーザがページを表示しようとすると、ブラウザに「Forbidden」というメッセージが表示されます。

ページをプライベートにして、他の CSP ページからのリンクを介してのみアクセスできるようにするには、以下のように PRIVATE クラス・パラメータに 1 を指定します。

Class Member

```
Parameter PRIVATE = 1;
```

既定では、ページはパブリックです。

10.2 ページを使用する許可の要求

SECURITYRESOURCE クラス・パラメータを使用して、CSP ページへのアクセスを制限します。以下に例を示します。

Class Member

```
Parameter SECURITYRESOURCE = "%Development:USE";
```

SECURITYRESOURCE パラメータは、システム・リソースとそれぞれのリソースに必要な許可のコンマ区切りのリストである必要があります。垂直バー (|) を使用することで OR 条件を指定し、コンマ (,) を使用することで AND 条件を指定することができます。このページを表示するか、クライアントからサーバ側のメソッドのいずれかを呼び出すには、ユーザは指定されたすべてのリソースで指定された許可を保持している必要があります。

リストの項目の形式は、以下のとおりです。

```
Resource[:Permission]
```

Resource は、サーバで定義された任意のリソースです。リソースのリストを確認するには、**[システム管理]→[セキュリティ]→[リソース]** に移動します。

Permission は、USE、READ、WRITE のいずれかです。オプションであり、既定値は USE です。

別の例を示します。

```
Parameter SECURITYRESOURCE = "R1,R2|R3,R3|R4" ;
```

この例は、ユーザが R1 かつ (R2 または R3 のいずれか) かつ (R3 または R4 のいずれか) のリソースを持っている必要があることを意味しています。ユーザに R1 および R3 がある場合、ユーザはページを実行できます。ユーザに R1 および R4 がある場合、ユーザは R2 または R3 のいずれかの条件を満たしていないため、ページを実行できません。垂直バー (|) の OR 条件は、コンマ (,) の AND 条件よりも優先されます。

11

暗号化

CSP ページで、%sessionの一意のセッション・キーを使用することで、他の CSP ページに送信される URL パラメータを含め、ブラウザに送信される値を暗号化できます。セッション・キーが HTTP アカウントに送信されることはないため、このメカニズムは安全です。

11.1 値の暗号化と解読

値を暗号化するには、ページ・クラスの Encrypt() メソッドを使用します。このメソッドは、**%CSP.Page** のスーパークラスから継承されます。同じセッション内で、Decrypt() メソッドを使用することでこの値を解読できます。どちらのクラスでも、このメソッドは自動的にセッション・キーを使用することに注意してください。

11.2 URL パラメータの暗号化

ある CSP ページから同じセッション内の別の CSP ページへの HTML **<A>** アンカー・リンクを含める場合、URL パラメータを暗号化できます。ブラウザに表示される URL には、以下のように元のパラメータではなく CSPToken URL パラメータが含まれます (以下の例では切り詰められています)。

```
GCSP.EncryptPage2.cls?CSPToken=1nz1Q1kNd$fJPuzngVKhsKrO...
```

システムには、以下の 2 つの部分があります。

- リンクの URL を作成するページで、**%CSP.Page** の Link() メソッドを使用して URL を作成します。

  暗号化されるかどうかに関係なく、すべての URL に対して Link() を使用するのがベスト・プラクティスです。

  ObjectScript

  ```
  Set origurl="GCSP.EncryptPage2.cls"
  Set urlparms("SAMPLEPARM")="sample value"
  Set tURL = ##class(%CSP.Page).Link(origurl,.urlparms)
  Set html="<p>Link to page 2: <a href="""_tURL_""">Link</a>"_"</p>"
  ```

- ターゲット・ページで、ENCODED クラス・パラメータを 1 または 2 に指定します。このクラス・パラメータには、以下のいずれかの値を指定できます。

  - ENCODED=0 ― クエリ・パラメータは暗号化されません。ブラウザはこれらをそのまま受け取ります。

- ENCODED=1 ー すべてのクエリ・パラメータは暗号化され、CSPToken URL パラメータ内で渡されます (例を参照)。
- ENCODED=2 ー 暗号化されないパラメータ (これは、Link() メソッドで追加されたパラメータとは異なり、URL に手動で追加された URL パラメータです) があることを除き、1 と同じです。暗号化されないパラメータは %request から削除されます。

以下に例を示します。

Class Member

```
Parameter ENCODED = 2;
```

ここに記載されているように URL パラメータが暗号化されている場合でも、通常の方法でパラメータの値を取得できます。以下に例を示します。

ObjectScript

```
 set urlparm=$GET(%request.Data("SAMPLEPARM",1))
```

InterSystems IRIS® データ・プラットフォームでは、必要な場合は常に自動的に値を解読します。

12

CSP ページでのテキストのローカライズ

アプリケーションのテキストのローカライズに関する一般的な情報は、"文字列のローカライズとメッセージ・ディクショナリ"を参照してください。

CSP ベースの Web アプリケーションの場合、%response オブジェクトを使用する追加のオプションがあります。

12.1 既定のランタイム言語の設定

$$$Text、$$$TextJS、および $$$TextHTML マクロを介してローカライズされたテキストの場合、既定のランタイム言語は、%response オブジェクトの Language プロパティによって決定されます (明示的にこれを設定している場合)。Language プロパティが明示的に設定されていない場合、ランタイム言語はブラウザの設定によって決まります。

Language プロパティを設定する方法としては、以下の例に示すように、%Library.MessageDictionary の MatchLanguage() メソッドを使用することをお勧めします。

```
Set lang = ##class(%MessageDictionary).MatchLanguage(languages,domain,flag)
Set %response.Language=lang
```

言語のリストとドメイン名を指定すると、このメソッドは HTTP 1.1 マッチング・ルール (RFC2616) を使用して、ドメイン内で最も一致する言語を見つけます。このメソッドには、以下のシグニチャがあります。

```
classmethod MatchLanguage(languages As %String,
                          domain As %String = "",
                          flag As %String = "") as %String
```

以下はその説明です。

- languages は、RFC1766 形式の言語名のコンマ区切りリストです。リスト内の各言語には、関連付けられた品質の値を指定できます。この値は、言語リストによって指定された言語に対するユーザの好みの見積もりを表します。品質値の既定値は q=1 です。

  例えば、da, en-gb;q=0.8, en;q=0.7 は、「 」という意味になります。リスト内の言語がサポートされている言語タグに一致するのは、それがタグと正確に一致しているか、タグの接頭語と正確に一致する (接頭語の後の最初のタグ文字がハイフンである) 場合です。特殊言語のアスタリスク (*) が入力リスト内に存在する場合、これは、リスト内に存在する他のどの言語とも一致しない、すべてのサポート対象言語と一致します。

  サポート対象言語タグに割り当てられた言語品質係数は、その言語タグに一致するリスト内の最長言語の品質値です。返される言語は、最も高い品質係数が割り当てられたサポート対象言語です。

- domain は、マッチングを実行するローカリゼーション・ドメインです。

- flag は、システム・メッセージまたはアプリケーション・メッセージのどちらに対してマッチングされるかを示すオプションのフラグです。

12.2 %response.GetText() メソッド

%response オブジェクトには、GetText() インスタンス・メソッドが含まれます。このメソッドを使用することで、メッセージ・ディクショナリからテキストを取得して、メッセージに指定可能な引数の値を置換できます。

メソッド・シグニチャは以下のとおりです。

```
method GetText(language As %String = "",
               domain As %String = "",
               id As %String,
               default As %String,
               args...) returns %String
```

以下はその説明です。

- language は、言語を指定するオプションの RFC1766 コードです。InterSystems IRIS ではこの文字列はすべて小文字に変換されます。既定の language は %response の **Language** プロパティです。**Language** プロパティが明示的に設定されていない場合、使用される言語はブラウザの設定によって決まります。
- domain はメッセージのドメインを指定するオプションの文字列です。指定されない場合、domain は既定で %response オブジェクトの **Domain** プロパティに設定されます。
- id はメッセージ ID です。
- default は、language、domain、および id で識別されるメッセージが見つからない場合に返される文字列です。
- arg1、arg2 などは、メッセージの引数の置換テキストです。これらはすべてオプションなので、メッセージに引数がない場合でも GetText() を使用できます。

13

認証の共有方法

このページでは、複数の CSP ベースの Web アプリケーションをグループとして動作するよう構成する 2 つの方法を説明します。

- 認証の共有 : アプリケーションが認証を共有しない場合、ユーザは別のアプリケーションによってリンクされた各アプリケーションに別個にログインする必要があります。認証の共有により、ユーザは 1 回のログインでリンクされたすべてのアプリケーションに入ることができます。
- データの共有 : 複数のアプリケーションがグローバル状態の情報を共有および調整できます。

13.1 認証のアプローチ

認証の共有には以下のアプローチを使用できます。

- ログイン Cookie の 1 回限りの共有。同じ ID を持つすべてのアプリケーションが認証を共有します。
- (By-Session グループを持つ) セッションの共有による継続的な共有。このシステムでは、グループのアプリケーションの認証は 1 つのユニットとして移動します。グループ内のアプリケーションのユーザが新規ユーザとしてログインすると、すべてのアプリケーションがそのユーザに移動します。1 つのアプリケーションでログアウトすると、すべてのアプリケーションでログアウトされます。

  アプリケーションはセッションで実行されます。各セッションには、それに関連付けられたセキュリティ・コンテキストがあります。

  いくつかのアプリケーションが同じセッション内に配置されると、それらのアプリケーションは認証を共有します。これは By-Session グループ (セッション共有) と呼ばれます。また、セッションはユーザ定義のデータを含むことができます。
- グループの識別子を一致させることによる継続的な共有。これは By-ID グループと呼ばれます。このシステムでは、グループはセキュリティ・コンテキストを共有します。各アプリケーションは通常別個のセッションとなります。

  重要　　このシステムでは、グループはユーザ・データを管理せず、認証のみが扱われます。

13.2 1 回限りの共有 : ログイン Cookie

ログイン Cookie は最近ログインしたユーザについての情報を保持します。ユーザが頻繁にログインしなくてもよいようにする一方で、アプリケーションが分離され、接続されていない状態を保ちたい場合は、ログイン Cookie を使用します。これは、Web アプリケーションの [**ログイン Cookie**] オプションに対応します。

ログイン Cookie では、各アプリケーションを別個のセッションに配置します。これにより、認証はアプリケーションに最初に入ったときにのみ共有されます。ログイン Cookie のアプリケーションはグループを形成しません。そのため、ログイン後、あるアプリケーションで認証を変更しても、その他のアプリケーションには影響しません。

ユーザがパスワードを使用してログインすると、その認証は Cookie で保存されます。ログイン Cookie が有効にされた別のアプリケーションに (初めて) 入ると、Cookie に保存された認証が使用されます。ユーザがログイン Cookie が有効にされていない 3 つ目のアプリケーションに (初めて) 入る場合、ユーザはユーザ名/パスワードを入力する必要があります。

13.3 By-Session グループ (セッション共有)

アプリケーション間で認証とデータを共有する 1 つの方法は、By-Session グループを定義することです。このシステムでは、グループのアプリケーションの認証は 1 つのユニットとして移動します。グループ内のアプリケーションのユーザが新規ユーザとしてログインすると、すべてのアプリケーションがそのユーザに移動します。1 つのアプリケーションでログアウトすると、すべてのアプリケーションでログアウトされます。

アプリケーションでセッションが共有されると、セッション・オブジェクトを介して認証とデータの両方が共有されます。

セッションの共有には潜在的な問題があります。セッション・イベントは元の Web アプリケーションからのみ取得されます。リンクが異なるセッション・イベントを必要とするページに飛ぶ場合、これらのセッション・イベントは実行されません。また、異なるセキュリティ・コンテキストを持つ別の Web アプリケーションでページを実行すると、ログインが必要になる可能性があります。ログインにより、元の Web アプリケーションで実行中のページのセキュリティ・コンテキストが変更される可能性があります。By-Session グループを使用することに決定する前に、以下の "方針を選択する際の考慮事項" をお読みください。

セッションを共有するには、以下の 2 つの方法があります。

- セッション Cookie パス : 正確に一致するセッション Cookie パスを持つすべてのアプリケーションが、同じセッションに配置されます。
- CSPSHARE : アプリケーション・ページへのリンクに CSPSHARE=1 を設定します。ソース・アプリケーションのセッション Cookie パスが、ターゲットのセッション Cookie パスと異なる場合には、これを使用してください。

By-Session 共有が必要な場合、最適なソリューションは、同じセッション Cookie パスを割り当てることができるようすべてのアプリケーションに名前を付けることです。セッション Cookie パスはアプリケーション名の部分文字列である必要があるため、アプリケーションの名前を変更する必要がある場合もあります。

これを実行できず、セッション共有が必要な場合は、アプリケーション間でジャンプするリンクに CSPSHARE パラメータを配置する必要があります。ターゲット・アプリケーション・ページは、ソース・アプリケーション・ページと同じセッションに配置されます。ソース・セッションは CSPCHD パラメータまたはセッション Cookie のいずれかから決定されます。

13.3.1 CSPSHARE

CSP はブラウザから要求を受け取ると、受け取った **sessionId** が有効なものであるかどうかを確認するため、一連のチェックを行います。これらのチェックには、以下が含まれます。

- ユーザ – エージェントがこの **sessionId** からの以前の要求のものと同じであるかどうか

- Cookie が使用されている場合、この **sessionId** が Cookie に由来するものなのか、CSPCHD パラメータに由来するものなのか

CSPSHARE=1 のクエリ・パラメータを渡すと、[ログイン CSRF 攻撃を防ぐ] オプションがチェックされている場合、アプリケーションが Cookie のみを使用するよう構成されていても、URL を介して **sessionId** を渡すことはできません。この動作を採用することをお勧めします。

[ログイン CSRF 攻撃を防ぐ] オプションがチェックされておらず、**CSPSHARE=1** である場合、CSP はチェックを実施せず、ユーザは別のアプリケーションへのリンクを構築し、**CSPCHD**=sessionId を使用して現在の **sessionId** を含めることができるため、このリンクは既存のページと同じセッションを実行します。また、リンクを構築する際に **CSPSHARE=1** である場合、CSP は自動的にリンクに **CSPCHD**=sessionId を挿入します。**Write** 文を使用して手動でリンクを挿入する場合は、**sessionId** を手動で挿入する必要がある場合があります。例えば、`https` ページから `http` ページ (または `http` ページから `https` ページ) を要求するアプリケーションがある場合、以下のように **CSPSHARE=1** をリンクに追加します。

```
#(..Link()(%request.URL_"?CSPSHARE=1"))#
```

CSPSHARE=1 はリンクの構築で CSPCHD が強制的に追加されるようにし、InterSystems IRIS で Cookie が有効であることが検出された場合でも、**sessionId** が共有されます。

詳細は、"CSPSHARE に関する考慮事項" を参照してください。

13.4 By-ID グループ

アプリケーション間で認証とデータを共有するもう 1 つの方法は、以下のように By-ID グループを定義することです。

1. 管理ポータルで、**[システム管理]→[セキュリティ]→[アプリケーション]→[ウェブ・アプリケーション]** に移動します。
2. **[ID でグループ化]** フィールドで、アプリケーションに共通のグループ名を付けます。この名前により、開いているアプリケーションが 1 つにグループ化されます。

グループは異なるセッションに分かれます。アプリケーションはデータを共有しません。

グループ名は、ネームスペースではなくアプリケーションに付けられます。同じグループ名を持つアプリケーションは、ネームスペースに関係なく認証を共有します。

認証は単一のブラウザでのみ共有されます。

13.5 認証アーキテクチャ

13.5.1 セキュリティ・コンテキストと Sticky Login

アプリケーションはセッションで実行されます。セッションでは、アプリケーションを実行するセキュリティ・コンテキストを必要とします。セキュリティ・コンテキストには、認証の状態が含まれます。

By-Session グループと By-ID グループには、Sticky Login があり、セッションまたはグループで最後に使用されたアプリケーションのセキュリティ・コンテキストが記憶されます。グループのアプリケーション内のユーザが異なるユーザとしてログインした場合、Sticky Login は更新されます (ユーザが認証のないアプリケーションにログインした場合は、Sticky Login は更新されません)。

セッションでアプリケーションにジャンプすると、そのセッションはターゲット・アプリケーションに適した Sticky Login の使用を試みます。Sticky Login がセッションの現在のセキュリティ・コンテキストに一致せず、アプリケーションが Sticky Login

の認証方法を受け入れることができる場合は、セッションのセキュリティ・コンテキストは Sticky Login のコンテキストに切り替えられます。

セッションが終了すると、セッションの Sticky Login は失われます。グループのアプリケーションのいずれかを含むセッションがすべて終了すると、グループの Sticky Login は失われます。

最初のログイン後、グループには関連付けられた Sticky Login オブジェクトが存在します。グループのアプリケーションのいずれかに入る際に、このオブジェクトの使用が試みられます。グループ内のアプリケーションに UnknownUser として入ると、グループ内のその他すべてのアプリケーションを認証なしのセキュリティ・コンテキストに移動することになるため、Sticky Login は更新されません。

Sticky Login に 2 要素認証で認証されたユーザが含まれる場合、2 要素以外のアプリケーションでもその 2 要素認証が使用されます (ただし、ユーザ名認証がその 2 つのアプリケーションで一致する場合)。

13.5.2 カスケード認証

アプリケーションの認証情報の取得を試みる際、CSP サーバは優先度を使用します。CSP サーバは、以下のイベントごとに新しい認証情報を取得しようとします。

- 新規セッションへの最初の要求の場合
- セッション内でアプリケーションの変更があった場合
- アプリケーションが By-ID グループの一部であり、セッションの現在のセキュリティ・コンテキストがグループの Sticky Login のコンテキストと一致しない場合
- 要求にユーザ名/パスワードのペアが含まれる場合

CSP サーバは、以下の順序で新しい認証情報を取得しようとします。

1. 明示的なログイン : ユーザが認証されたユーザ名/パスワードを入力したかどうかをチェックします。そうである場合、システムはアプリケーションの認証グループのコンテキストを更新します (これによりグループの Sticky Login が設定されます)。
2. Sticky Login : アプリケーションのグループの Sticky Login のコンテキストが取得されます。Sticky Login と By-Session グループがない場合、セッションの現在のコンテキストを使用します。
3. ログイン Cookie : このアプリケーションでこれが存在し、有効化されている場合に使用します。
4. 認証なし : アプリケーションで有効にされている場合に不明ユーザを使用します。
5. ログイン・ページの表示 : 上記すべてが失敗した場合、ユーザからユーザ名/パスワードを要求します。%CSP.Session API から呼び出された場合、ユーザ名/パスワードのみが試行されます。ログイン後、UnknownUser としてログインした場合を除き、グループの Sticky Login を更新します。

13.5.3 ログアウトまたはセッションの終了

セッションがログアウトまたは終了されると、認証は失われます。セッションをログアウトまたは終了するには、以下の %CSP.Session メソッドを使用できます。

推奨 : CacheLogout=end

CSP セッションをログアウトするには、文字列 `CacheLogout=end` を含む URL を渡して、アプリケーションのホーム・ページにリンクすることをお勧めします。これにより、ホーム・ページの実行が試行される前に、現在のセッションが終了します (つまり、取得したライセンスが解放され、既存のセッション・データが削除され、セッションのセキュリティ・コンテキストが削除されます)。

この Web アプリケーションで認証が必要な場合、セッションも認証されたユーザも存在しません。この場合、IRIS ではホーム・ページのロジックを実行せず、代わりにログイン・ページを表示します。ユーザが有効なログインを実行すると、新しいセッションが開始され、ホーム・ページが表示されます。

Set EndSession? =1

これによりセッションは強制終了されます。セッションの Sticky Login のコンテキストは破棄されます。OnEndSession() が呼び出されます。セッションに By-Session グループが含まれる場合、このグループは破棄されます。セッションに By-ID アプリケーションが含まれる場合、そのアプリケーションはグループから削除され、それがグループ内の唯一のアプリケーションでない限り、グループは存在し続けます。ログイン Cookie は影響を受けません。By-Session グループのデータは失われます。ただし、By-ID グループの場合、グループの Sticky Login は一度の破棄では影響を受けず、グループのその他のメンバは引き続きログインしたままになります。

さらに、By-Session グループでは、破棄によりグループのメンバは分散され、メンバのアプリケーションに再び入った場合、同じ新規セッションに再度統合されるか、(CSPSHARE を使用してグループ化されていた場合) さまざまなセッションに送られるかは保証されません。

Session Logout

セッションはログアウトされます。Sticky Login のコンテキストは破棄されます。セッションに By-Session グループが含まれる場合、グループ内のすべてのアプリケーションの認証が失われます。セッションに、By-ID グループのアプリケーションが含まれる場合、グループは Sticky Login のコンテキストを失い、そのグループ内のすべてのアプリケーションはログアウトされます。

また、OnLogout が呼び出されます。ログイン Cookie が破棄されます。

セッションは存続するため、データは By-Session グループ用に保持されます。

すべてのセッションのログアウト

特定のユーザとして現在認証されているすべてのセッションをログアウトすることができます。

これにより、ログイン Cookie は破棄されます。

セッションは存続しますが、認証は失われます。

13.6 方針を選択する際の考慮事項

このセクションには、方針を選択する際に検討すべきいくつかのポイントが含まれています。

13.6.1 ログイン Cookie の考慮事項

ログイン Cookie を介して共有するかどうかを決定する際は、以下の点について検討してください。

- ログイン Cookie は、新規ユーザがパスワードを入力してログインするたびに、そのユーザに更新されます。
- ログイン Cookie は、認証なしのログイン (UnknownUser として) に対しては生成されません。
- ログイン Cookie は、API 呼び出しを介したログイン時には生成されません。
- ログイン Cookie のセッションは、セッションが認証されると独立します。そのため、あるセッションからログアウトするか、そのセッションがタイムアウトしても、その他のセッションには影響しません。
- ログイン Cookie アプリケーションからの認証は、パスワードのみの (ログイン Cookie を使用しない) アプリケーションとは共有できません。グループ内の認証ありのアプリケーションの一貫した動作を確保するには、すべてのアプリケーションにログイン Cookie を使用するか、またはどれにも使用しないでください。

13.6.2 グループの考慮事項

このセクションには、認証を共有する認証グループを作成する際に検討すべきいくつかのポイントが含まれています。

- セッション共有は、セッション・オブジェクトを介してデータを共有する必要があると判断した場合にのみ使用してください。By-ID およびログイン Cookie による共有が、より堅牢で、予測可能です。
- ターゲット・ユーザに対して統一された動作を生み出すために、グループを作成する際は可能な限り一貫性を維持するようにしてください。1 つのアプリケーションを、By-ID グループと By-Session グループの両方に配置しないでください。異なる認証方針を使用すると、予期しない動作を引き起こす可能性があります。By-ID は、By-Session より優先されます。そのため、あるアプリケーションに両方のグループがある場合、By-ID での同期が維持されます。
- グループのすべてのメンバに対して同じ認証タイプを使用してください。特に、グループ内の一部のアプリケーションがログイン Cookie を許可し、その他のアプリケーションが許可しない場合、ユーザ名/パスワードを介してグループに入るとグループ全体が認証されますが、ログイン Cookie を介して入ると一部のアプリケーションのみが認証されることになります。このため、ログインが要求される場合と要求されない場合があることについて、ユーザ間で混乱が生じる可能性があります。
- CSP サーバでは、すべてのアプリケーションが認証グループに属しているものと見なします。そのため、1 つのセッション内の単独のアプリケーションは、単一エンティティの By-Session 認証グループを形成します。
- By-ID グループに、認証なしのみのアプリケーションを配置しないようにしてください。
- By-Session グループは脆弱です。By-ID グループの方が堅牢なアプローチです。共有データを含め、あるグループについてのすべての情報が単一のセッションに含まれるため、グループは容易に失われる可能性があります。これは、セッションはタイムアウトする可能性があるためです。つまり、特定の期間が経過すると、セッションは自動的に破棄されます。ユーザがコンピュータから離れるか、By-Session グループに含まれていないアプリケーションを使用すると、セッションはタイムアウトする可能性があります。グループ内のアプリケーションの 1 つが ENDSESSION=1 とマークすると、グループは分散されます。
- ブラウザで、分散したアプリケーションのページを含むタブが開かれている場合 (特に、それらが元々 CSPSHARE=1 を使用してグループ化されていた場合)、それをクリックすると複数のログインが必要になる場合があります。いずれにしても、元のセッションのデータは永久的に失われます。

  グループが認証を失った場合、グループのアプリケーションから開いているページを更新するか、そのページに移動すると、ユーザの再ログインが必要になります。
- By-Session アプリケーションを含むセッションを終了すると、その By-Session グループ内のアプリケーションのページを更新する際に、ユーザの再ログインが必要になります。By-ID アプリケーションを含むセッションを強制終了する場合は、そのセッションのアプリケーションがグループの唯一のメンバでない限り、ログインは必要ありません。
- セッションをログアウトすると、そのセッションのグループのすべてのメンバが (異なるセッションに属していても) ログアウトされます。グループのいずれかのページを更新するには、新規のログインが必要になります。ただし、By-ID グループでは、1 回のログインで、グループ全体にログインします。By-Session グループの場合、Web ゲートウェイが分散したアプリケーションを新規に構築されたセッション・オブジェクトに戻すことができるならば、1 回のログインでグループ全体にログインします。
- ログアウトしてもセッションは破棄されないため、セッション・データはすべて存続します。
- 同じブラウザの異なるタブで、2 人の異なるユーザが同じアプリケーションにログインすることはできません。
- 認証は単一のブラウザでのみ共有されます。このランタイム識別子は、**%Session** オブジェクトに格納されます。
- グループ化により、同じグループ (By-ID) または同じセッション (By-Session) 内のユーザ間で認証を共有することが可能になります。指定のグループ外のアプリケーションからの認証を共有する場合は、ログイン Cookie を使用してください。指定のグループ外のアプリケーションに認証を送信する場合は、CSPSHARE=1 を使用してください ("CSPSHARE に関する考慮事項" を参照してください)。

13.6.3 CSPSHARE に関する考慮事項

CSPSHARE は、最終的な手段として使用してください。

By-Session アプリケーションのリンクは、以下の状況では CSPSHARE=1 を必要としません。

- ソース・アプリケーションとターゲット・アプリケーションが同じグループ ID を持っている場合。
- ターゲット・ページがソース・ページと同じアプリケーションに存在する場合。
- ターゲット・ページのアプリケーションのセッション Cookie パスが、ソース・アプリケーションのセッション Cookie パスと一致する場合。

13.6.4 データの共有

By-Session グループは、セッション・オブジェクトを介してデータを共有できます。

By-ID グループは、独自のデータを管理する必要があります。例えば、データがグローバルに格納されている場合、現在のユーザ、$Username、またはグループのランタイム ID を使用して、データにキーを設定できます。CSP サーバは各ブラウザにブラウザ ID の Cookie を割り当てます。By-ID グループが作成されると、そのグループにキーが割り当てられます。このキーは、ブラウザ ID とグループ ID が連結されたものです。これにより一意のキーである %CSP.Session.BrowserId が作成されます。これはデータを格納する際のキーとして使用できます。

14
ログの有効化

このページでは、CSP アクティビティを記録するログを有効化する方法を説明します。このログは、CSP ベースの Web アプリケーションをトラブルシューティングする際に役立ちます。

14.1 ログの有効化と無効化

ログを有効化するには、ターミナルで以下のコマンドを入力します。

ObjectScript

```
Set ^%ISCLOG = 2
```

^ISCLOG グローバルでログ情報を表示できます。

ログを無効にするには、以下のコマンドのいずれかを使用できます。

ObjectScript

```
Set ^%ISCLOG = 0
Kill ^%ISCLOG
```

14.2 ログ・レベル

参考のために、ログ・レベルを以下に示します。

- 0 ー InterSystems IRIS® データ・プラットフォームはログ記録を実行しません。
- 1 ー InterSystems IRIS は例外的なイベントのみをログ記録します (エラー・メッセージなど)。
- 2 ー InterSystems IRIS は詳細な情報をログ記録します (`method ABC invoked with parameters X,Y,Z and returned 1234` など)。
- 3 ー InterSystems IRIS は HTTP 要求から受け取ったデータなど、未処理の情報をログ記録します。
- 5 ー InterSystems IRIS は OAuth 2.0 情報をログ記録します。

14.3 ISCLog の詳細

ISCLOG では、一部のエントリは以下のようにイベント・ログ・ヘッダと一致します。

ISCLOG	イベント・ログ
Job	Cache-PID
SessionId	Session-ID
Tag	Request-ID

以下の表に、ISCLOG のフィールドと定義を示します。

テーブル 14-1: ISCLOG のフィールド

フィールド	定義
%category	CSPServer：cspServer、cspServer2、%request、%response からログが記録されます。
	CSPSession：セッションを処理する %session および cspServer と cspServer2 の一部からログが記録されます。これにより、セッションのライフサイクルの監視が可能になります。
	CSPLicense：ライセンスを処理する cspServer および cspServer2 の一部からログが記録されます。
	Gateway Request：ゲートウェイ要求を処理する GatewayMgr、GatewayRegistry、ゲートウェイ要求ハンドラ、および cspServer2 の一部からログが記録されます。
%level	1 = 例外およびエラー。
	2 = CSPSession 情報。CSPLicense 情報。cspServer からの情報：%response、%session、および %request の設定後の要求処理の一部。これには、認証、ライセンスの処理、リダイレクト、および CSPpage の呼び出しが含まれます。
	3 = cspServer2 からの情報：%response、%session、%request、および Web ゲートウェイとのハンドシェイク/データ転送を設定する要求の処理の一部。
%job	ISCLOG 要求が行われたときの $job の値。イベント・ログのヘッダの Cache-PID フィールドと一致します。
%sessionid	利用可能な場合に入力されます。ISCLOG 要求が行われたときの sessionid の値。イベント・ログのヘッダの Session-ID フィールドと一致します。

フィールド	定義

%tag	CSP サーバの場合、タグにはゲートウェイからの要求 ID が含まれます（利用可能な場合）。これは、イベント・ログのヘッダの Request-ID フィールドと一致します。その他のロガーでは、この値は任意の値に設定される可能性があります。
	ISCLOG エントリの作成者が使用できます。Web ゲートウェイによって送信される要求の ID を格納します。ISCLOG エントリの生成のためのフィルタとして使用できます。
	
	`Set ^%ISCLOG("Tag","mytagvalue1")=1`
	`Set ^%ISCLOG("Tag","mytagvalue2")=1`
	
	タグが付いていないか、"mytagvalue1" または "mytagvalue2" のタグが付いた ISCLOG 要求のみが記録されます。

%routine	現在実行中のルーチンの名前。
%message	以下の "メッセージ形式" を参照してください。

14.4 メッセージ形式

メッセージは、現在実行中のタグ・ラベルまたはメソッドの名前で開始されます。この名前は角括弧で囲まれます。例えば、`[MyMethod] rest of messages` のようになります。

`CSPSession` カテゴリ内のメッセージには、メソッド名の後に `CSPSession-Id=sessid` も含まれます。これが必要なのは、セッション・イベントはセッションが作成される前か、セッションが破棄された後（つまり、ISCLOG エントリ内の SessionId フィールドは空）に記録できるためです。

```
[MyMethod] CSPSession-Id: 12ty34ui22
```

`GatewayRegistry` カテゴリ内のメッセージには、メソッド名の後に `CSPID=cspid` も含まれます（利用可能な場合）。これにより、ゲートウェイ要求ハンドラを介して API 呼び出しからの個々のゲートウェイ要求を追跡することが可能になります。

```
[MyMethod]CSPID:334r43345 rest of message
```

A

予約された URL パラメータ

以下の URL パラメータは、CSP ベースの Web アプリケーションのコンテキストで予約されています。

IrisUserName

ログイン・ページからの、ログインするためのユーザ名を含みます。

IrisPassword

ログイン・ページからの、`IrisUserName` により指定されるユーザのパスワードを含みます。

IrisOldPassword

`IrisUserName` および `IrisPassword` と共に渡される場合、ユーザの現在のパスワードを含みます。セキュリティ・ルーチンにより、ユーザのパスワードは `IrisPassword` からの新しい値に変更されます（例：`IrisOldPassword=fredsAboutToBeChangedPwd`）。パスワードが変更されると、ユーザは新しいパスワードを使用してログインされます。

IrisLogout

`IrisLogout` に値が指定されないか、`cookie` 以外の値が指定されると、この要求のセッションがログアウトされます（破棄はされません）。ログアウトにより、現在のログイン Cookie が破棄され、このセッションで宙に浮いた 2 要素認証のセキュリティ・トークンは削除されます。

`IrisLogout=cookie` は現在のログイン Cookie を破棄します。

IrisSecurityToken

`IrisSecurityToken` には、ログイン・セキュリティ・トークン・ページから送信されたセキュリティ・トークンの値が含まれます（例：`IrisSecurityToken=12345678`）。

IrisSecuritySubmit

この名前が存在することは、ユーザが `IrisSecurityToken` と関連付けられた値を持つセキュリティ・トークンを送信していることを意味します。

IrisSecurityCancel

この名前が存在することは、ユーザがログイン・セキュリティ・トークン・ページをキャンセルしたことを意味します。

IrisLoginPage

カスタム・ログイン・ページを含むログイン・ページには、2 つのサブページが含まれます。1 つはログインのため、もう 1 つはセキュリティ・トークンの値を返すためのページです。ページは IrisLoginPage の値をチェックして、どのサブページを表示するかを決定します。IrisLoginPage=1 は、ログインのためのサブページを表示する必要があることを示します。

IrisNoRedirect

ページ P が要求されましたが、このページは認証されていないため、ログイン・ページが表示されます。ユーザがログイン・ページから情報を送信すると、P のページ要求は通常元のブラウザにリダイレクトされます (これにより、ブラウザが P を表示する前に <Resend> ボタンを押すようにユーザに求めることがなくなります)。この動作は、IrisNoRedirect=1 を渡すことで簡略化できます。

B

特殊な HTML 指示文

CSP ページ・クラス内の OnPage() コールバックで、特殊な指示文である &html<> を使用して HTML を記述できます。この指示文では、より簡単に HTML を構築できますが、その結果はユニット・テストには適していません (ブラウザを使用せずに結果を検証できないため)。

B.1 &html<> の基本

特殊な指示文 &html<> には、複数行の HTML など、任意の有効な HTML を含めることができます。以下に例を示します。

Class Definition

```
Class GCSP.HTML Extends %CSP.Page
{

ClassMethod OnPage() As %Status
{
   &html<<!DOCTYPE html>
   <html lang="en" dir="ltr">
   <body>
   <h1>Basic Page</h1>
   <div>No double quotes needed "here"</div>
   </body>
   </html>>

   Quit $$$OK
}

}
```

B.2 &html<> 内の式

指示文 &html<> に含まれる HTML 内に式を含めると便利な場合があります。以下の 2 つの可能性があります。

- コンパイル時に評価される式。この場合、次の構文を使用できます。

  ```
  ##(expression)##
  ```

  この種類の式は、CSP コンパイル時式と呼ばれます。

- 実行時に評価される式。この場合、次の構文を使用できます。

  ```
  #(expression)#
  ```

ここで、expression は任意の ObjectScript 式です。この種類の式は、CSP 実行時式と呼ばれます。

どちらの場合も、式が必要な場所に構文を含めるだけです (いかなる連結も必要ありません)。以下に例を示します。

```
Class GCSP.HTML1 Extends %CSP.Page
{

ClassMethod OnPage() As %Status
{
   &html<<!DOCTYPE html>
   <html lang="en" dir="ltr">
   <body>
   <h1>Basic Page</h1>
   <div>This class was compiled at ##($zdatetime($h,3))##</div>
   <div>This class was viewed at #($zdatetime($h,3))#</div>
   </body>
   </html>>

   Quit $$$OK
}

}
```

C

CSP エラー・コード

このページでは、CSP ベースの Web アプリケーションで発生する可能性のあるエラーの原因と、その解決方法について説明します。これらのエラーの一部はタグベースの開発にのみ適用されますが、簡素化するためにここに含められています。

テーブル III-1: CSP エラー・コード、エラー・メッセージ、および報告されるタイミング

エラー・コード	エラー・メッセージ	報告されるタイミング
5902	ルール %1 は存在しません。	%apiCSP を呼び出してルールに属性を追加するときに、存在しないルール名を指定すると報告されます。
5903	ルール名が必要です。	ルールを追加または削除しようとしたが、ルール名を指定しなかった場合に報告されます。
5904	行番号 %3 のタグ '<%1>' には属性 '%2' が必要です。	CSP ページのタグに必要な属性を指定しなかった場合に報告されます。この必須の属性がなければ、ページはコンパイルできません。
5905	行番号 %3 にある属性 %1 の値 %2 は無効です。	CSP ページ内の属性の値が有効な選択肢でない場合に報告されます。例えば、<script language="Cache" runat="XXXXX"> と定義した場合、runat の値が有効な選択肢ではありません。CSP コンパイラはこのページをコンパイルできず、このエラーが報告されます。
5906	セッション ID が見つかりません。	%New() メソッドでセッション ID を指定せずに %CSP.Session インスタンスを作成しようとすると報告されます。例えば、Set session=##class(%CSP.Session).%New()' ではこのエラーが報告されますが、Set session=##class(%CSP.Session).%New(1234)' ではセッション ID 1234 を渡すためエラーは報告されません。
5907	セッション ID %1 は存在しません。	既存の %CSP.Session をロードしようとしたが、InterSystems IRIS® データ・プラットフォームに格納されていないセッション ID を %OpenId() に渡すと報告されます。
5908	クラス %1: %2 の作成に失敗しました。	CSP コンパイラが、CSP ページに対応するクラスを作成できない場合に報告されます。
5909	行番号 %2 のタグ <%1> に終了のタグがありません。	CSP コンパイラで、開始タグがあるが終了タグがないことが検出された場合に報告されます（ルール定義でタグに終了タグが必要であることが指定されている場合）。

エラー・コード	エラー・メッセージ	報告されるタイミング
5911	文字セット %1 がインストールされていないため、文字セットの変換を実行できません。	このページを出力するために CSP ページで指定された文字セットが、InterSystems IRIS にインストールされていない場合に報告されます。これは、OnPreHTTP() メソッドの %response.CharSet プロパティで指定された文字セットである可能性があります。クラス "%CSP.Page" の "charset" プロパティを参照してください。エラーで報告された文字セットを使用する予定であったことを確認し、もしそうであれば、それが InterSystems IRIS にインストールされていることをチェックします。または、OnPreHTTP() メソッドの %response.CharSet プロパティを設定します。
5912	CSP ページ '%1' が存在しません	存在しない CSP ページを要求した場合に報告されます。URL を誤入力したか、または別の CSP ページ上のリンクが間違っていた可能性があります。ページがサーバに存在するかどうかを確認し、存在しない場合は、リンクがどこから来たか探します。ページが存在するはずである場合、Web アプリケーションの設定が適切なディレクトリに正しく設定されていることと、CSP ファイルがディスク上に存在することを確認してください。このエラーは、autocompile オプションがオンになっており、CSP エンジンがこのページをコンパイルしようとして、ファイルを見つけられない場合に発生します。
5914	CSPアプリケーション '%1' が存在しません	URL のアプリケーションの部分が Web アプリケーションのリストで見つからない場合に報告されます。例えば、ページ /cspx/samples/menu.csp を csp ではなく cspx のタイプでロードしようとすると、InterSystems IRIS はその Web アプリケーションを見つけることができません。管理ポータルで [システム管理]→[セキュリティ]→[アプリケーション]→[ウェブ・アプリケーション] に移動して、アプリケーションのリストで、コマンドに誤りがないか確認してください。
5915	ライセンスを割り当てることができません。	ライセンス制限に達しているため、CSP セッションの新規の要求が受け入れられない場合に報告されます。Web アプリケーションの構成で指定された CSP セッションの既定のタイムアウトを短縮するか、ライセンスを追加で購入することを検討する必要があります。
5916	不正な CSP 要求です。	プライベート・ページへのアクセスを許可する暗号化トークンを含む別の CSP ページからリダイレクトされる代わりに、URL を入力してこのページにアクセスしようとしたか、このプライベート・ページへのアクセスを許可するために無効な暗号化トークンを使用した場合に報告されます。
5917	CSP は HTTP メソッド %1 をサポートしていません。	サポートされていない HTTP メソッドを使用しようとした場合に報告されます。サポートされる HTTP メソッドは GET、POST、HEAD です。現時点で、CSP サーバでその他の HTTP メソッドはサポートされていません。このエラーは、CSP サーバと対話する Web ゲートウェイのバージョンが適合していない場合にも発生する可能性があります。

エラー・コード	エラー・メッセージ	報告されるタイミング
5918	ログアウトされているため、動作を実行できません。	CSP 要求に暗号化されたデータが含まれるが、セッションが新規のセッションであり、解読キーを暗号化されたデータと一致させる方法がない場合に報告されます。通常これはセッションがタイムアウトした後に、ユーザがブラウザで別の要求を生じさせる動作を実行したために発生します。セッション・タイムアウト値を大きくするか、ユーザを最初のページにリダイレクトして動作を再度開始できるようにするエラー・メカニズムを使用できます。
5919	要求した動作は無効です。	通常、暗号化された文字列を CSP ページから InterSystems IRIS に渡したが、解読キーがこのデータを暗号化するのに使用されたキーと一致しない場合に報告されます。これは、ユーザが URL を手動で改ざんしたか、他の何らかの原因で暗号化された文字列の値が InterSystems IRIS で生成されてから次の HTTP メッセージで InterSystems IRIS に戻されるまでに変更された場合に生じる可能性があります。
5920	このCSPページはネームスペース '%1' から実行する必要があります	各 Web アプリケーションは、InterSystems IRIS の特定のネームスペースに結び付けられています。このエラーは、/csp/samples アプリケーションが SAMPLES ネームスペースに結び付けられている場合に、USER ネームスペース内の /csp/samples/loop.csp からページをコンパイルするといった動作を試みた場合に報告されます。
5921	CSPアプリケーション '%1' は、稼動するネームスペースを指定しなければなりません	Web アプリケーションの構成でネームスペースが欠落している場合に報告されます。管理ポータルでは Web アプリケーションがネームスペースなしで作成されることは許可されないため、これは通常 CPF ファイルが手動で誤って編集されたことを示します。
5922	返答待ち時間のタイムアウトです。	%Net.HttpRequest オブジェクトが、対話している HTTP サーバからの返答待ちでタイムアウトしたときに、このオブジェクトにより報告されます。
5923	%1 回リダイレクトされました。リダイレクションループと考えられます。	1 ページで 4 回を超えるリダイレクトが検出された場合に報告されます。コンパイラはその状態をループが存在していると見なします。CSP ページが ServerSideRedirect を使用して別のページにジャンプすると、ページ A.csp が B.csp にリダイレクトされ、それが A.csp にリダイレクトされることでループが発生する可能性があります。
5924	エラーが発生しましたが指定されたエラーページが表示できませんでした – Web マスタに通知してください。	実行時に CSP ページでエラーが発生すると、CSP エンジンは、任意の方法でエラーを処理できるユーザが指定したエラー・ページにリダイレクトします。しかし、ユーザ指定のエラー・ページが存在しないか、このエラー・ページを生成する際にエラーが発生すると、CSP エンジンは BACK^%ETN の使用に何らかの問題があることをログ記録し、このエラー・メッセージを報告します。このエラーはプロダクション・システムで表示される可能性があるため、ユーザ作成のエラー・ページにバグがある場合を考慮し、メッセージは意図的にあいまいな表現になっています。このエラーを解決するには、まず Web アプリケーションで指定されたエラー・ページが存在するかどうかを確認し、次にこのエラー・ページに潜在的なバグがないかどうか調べてください。

エラー・コード	エラー・メッセージ	報告されるタイミング
5925	行番号 %1 で <SCRIPT LANGUAGE=CACHE> タグに RUNATかMETHOD属性が指定されていません	<script language="Cache"> タグに、このコードを実行するべきタイミングを CSP コンパイラに指示するための必須の属性 runat、または新しいメソッドを作成するための method 属性が欠落している場合に報告されます。
5926	HTTP ヘッダが記述されフラッシュされたため、転送できません。	データがブラウザに書き込まれてから、サーバ側リダイレクトを使用しようとすると報告されます。%response.ServerSideRedirect 機能を使用して別のページにリダイレクトを試みる場合、これはデータがブラウザに戻される前に行う必要があります。つまり、通常これはページの OnPreHTTP() メソッドで行う必要があります。
5927	ページ %1 のクラス名と既にロードされているクラス %2 とが競合するため、このページをロードできません。	同じネームスペース内の異なるアプリケーションに、同一の名前を持つ 2 つの CSP ファイルがある場合に報告されます。例えば、USER ネームスペースに /test と /anothertest という 2 つの CSP アプリケーションがあり、これらのアプリケーションが InterSystems IRIS サーバ上の異なるディレクトリにある場合、それぞれに test.csp ファイルが存在します。autocompile を有効にして、URL /test/test.csp を入力すると、CSP コンパイラはこのページをクラス csp.test にコンパイルします。URL /anothertest/test.csp を入力すると、これはこのページをロードしてクラス csp.test を作成しようとし、それが異なるアプリケーションに既に存在することを検出して、このエラーを報告します。これが行われない場合、各要求でページ全体をリコンパイルするため、パフォーマンスが非常に悪くなります。同じネームスペースで同一のファイル名を使用しないようにするか、Web アプリケーションで定義されたパッケージ (既定では CSP) を変更してください。例えば、/anothertest をパッケージ名 package を使用するよう変更します。これにより、test.csp のコンパイル時に、クラス名 package.test が作成されます。この名前は、csp.test を使用する他のアプリケーションと競合しません。
5931	ページが表示される前にのみ、OnPreHTTP() にこのメソッドを呼び出すか、値を設定できます。	データがブラウザに出力される前に一部のパラメータを変更できるように、ページの OnPreHTTP() メソッドで呼び出す必要のある関数を呼び出した場合に報告されます。これを解決するには、この呼び出しを OnPreHTTP() メソッドに移動します。
5932	Web サーバ上のこの Web ゲートウェイ・バージョンで、動作は無効です。	使用している Web ゲートウェイのバージョンでこの動作がサポートされていない場合に報告されます。この機能を使用しないようにするか、Web ゲートウェイのバージョンを新しいバージョンにアップグレードしてください。
5933	CSPサーバで内部エラーが発生しました: %1	CSP エンジン内で予期しないエラー状態が発生している場合に報告されます。このエラーをインターシステムズのサポートに報告してください。
5954	CSP ページのロックに失敗しました。	CSP ページが自動コンパイルされる場合、2 つのジョブが同時に同じページのコンパイルを試みることがないよう、ページがまずロックされます。ロックがもう一方のジョブにより 60 秒以内に解除されない場合、コンパイルが何らかの理由で失敗したと見なされ、このエラー・メッセージが報告されます。スタジオからこのページのリコンパイルを試行して、エラーが報告されるかどうか確認してください。

エラー・コード	エラー・メッセージ	報告されるタイミング
5955	CSPAppList クエリ : Fetch() に無効なデータがあります。	CSP アプリケーションのリストを判断するクエリが無効である場合に報告されます。このエラーは稼働中のシステムでは表示されてはならないものです。
5956	CSPアプリケーション '%2' のディレクトリ '%1' が存在しません	Web アプリケーションによりポイントされているディレクトリがファイル・システム内に存在しない場合に報告されます。
5961	文字セット %1 を変換できません。	ブラウザからの要求を受け取ったときに報告されます。ブラウザにより送信された情報は現在の InterSystems IRIS の既定のロケールに変換されるため、エラーが発生します。変換をデバッグするには、ブラウザにより送信された情報を分離し、テスト・プログラム内で手動で文字セットから変換します。
5962	新規セッションを割り当てることができません。	%session.ForceNewSession() の呼び出し時に、このセッション ID に新しいスロットがない場合に報告されます。
5963	無効な SysLog レベルです: %1	内部ログ・レベルの設定時に、そのレベルが許容範囲外である場合に報告されます。