Frédéric Heinemann, Christian Rau

SAP® Web Application Server

Entwicklung von Webanwendungen

Galileo Press

Inhalt

1 Einführung

Zielgruppe

Dieses Buch richtet sich – wie der Untertitel zu Recht vermuten lässt – in erster Linie an Entwickler, die webbasierte Applikationen im R/3-Umfeld planen und umsetzen wollen. Hierbei stellt sich aber nicht so sehr die Frage, ob man als Leser des Buches ein R/3-Entwickler sein muss, sondern vielmehr, ob man bereit ist, sich den neuen Konzepten und Technologien im Sinne einer Herausforderung zu stellen. Insofern ist das Buch genauso gut für IT-Entscheider, Projektmanager und Interessierte geschrieben, die sich mit dem Thema auseinander setzen wollen.

Es stellt sich natürlich die Frage, wie man die sicherlich nicht einfache Aufgabe der Darstellung von Zusammenhängen und der Wissensvermittlung für den Fall eines so komplexen Produktes wie den SAP Web Application Server löst. Dies ist eines der wesentlichen Ziele, die wir uns mit diesem Buch gesetzt haben. Es wird viel Wert auf praktische Betrachtungsweisen gelegt. Daher werden Methoden und Technologien anhand einer konkreten Entwicklung und mit zahlreichen Beispielen zweckorientiert vorgestellt und eingesetzt.

Das Buch geht aber bewusst darüber hinaus. IT-Entscheider und Projektmanager stehen heute mehr denn je vor der schwierigen Frage, für welche Technologieplattform sie sich entscheiden sollen, und wenn sie sich für eine entschieden haben, mit welcher der zur Verfügung stehenden konkreten Technologien sie ihr Ziel verwirklichen können. Sie erhalten hier einen tiefer gehenden Einblick in die Architektur und die Einsatzmöglichkeiten des SAP Web Application Servers, um durch die Verbindung von Theorie und Praxis bei der Entscheidungsfindung unterstützt zu werden.

Voraussetzungen

Um die Übungen und auch die Abbildungen des Praxisteils nachvollziehen zu können, wird ein korrekt installierter SAP Web Application Server benötigt. Für die meisten Beispiele in diesem Buch ist die Version 6.10 ausreichend. Einige Übungen im Rahmen der BSP-Extensions und des MVC-Design-Pattern werden allerdings nur mit Version 6.20 und höher funktionieren. Sie benötigen außerdem auf Ihrem Übungssystem die zur Entwicklung erforderlichen Berechtigungen.

Vorkenntnisse in DHTML und JavaScript sind sicherlich hilfreich, Grundkenntnisse der ABAP-Programmierung zum vollständigen Nachvollziehen der Beispiele empfehlenswert. Hierbei werden Neulinge aber nicht alleine gelassen. Sie erfahren Unterstützung im Rahmen der Grundlagenvermittlung.

Inhalt Der Inhalt des Buches ist in thematisch überschaubare Blöcke gegliedert:

- **Kapitel 2 – Übersicht: SAP Web Application Server**
  Die Übersicht über den SAP Web Application Server betrachtet insbesondere die Architektur im ganzheitlichen Sinne, die neuen Erweiterungen wie z. B. den Internet Communication Manager, das Internet Communication Framework, die J2EE-Integration, den SAP Web Dispatcher und vieles mehr, was den Begriff »Web« im Namen des neuen Basissystems rechtfertigt. Nicht vernachlässigt werden in diesem Zusammenhang auch die damit in den Mittelpunkt gerückten Sicherheitsaspekte. Des Weiteren wird auf die Unterschiede zwischen der ABAP und der Java Personality sowie auf den Internet Transaction Server eingegangen. Den Abschluss bildet ein Ausblick auf die kommenden Versionen und die mit ihnen zu erwartenden Neuerungen.

- **Kapitel 3 – Grundlagen: BSP-Applikationen**
  In Kapitel 3 werden die notwendigen Grundlagen vorgestellt, die im Rahmen der Entwicklung mit Business Server Pages erforderlich sind. Nach einem kurzen Ausblick auf das Beispielszenario wird in einem weiteren Schritt insbesondere Neulingen die Gelegenheit gegeben, sich mit den benötigten Sprachen und Standards vertraut zu machen. Diese Abschnitte können von Lesern mit entsprechenden Vorkenntnissen in ABAP, HTML, JavaScript, CSS, XML sowie Internetprotokollen und Cookies übersprungen werden.

  Der zweite Teil widmet sich den eigentlichen Bestandteilen von BSP-Applikationen, der Verarbeitung und dem Modell-View-Controller Design-Pattern.

  Den Abschluss bildet eine Einführung in die Einbindung von Mobile Clients.

- **Kapitel 4 – Entwicklung: Werkzeuge**
  Die Werkzeuge stellen die Schnittstelle des Entwicklers zum System dar und sind somit ein wichtiger Faktor für seine tägliche Arbeit. Die Entwicklungsumgebung des SAP Web Application Servers ist sehr mächtig, da mit ihr nicht nur BSP-Applikationen, sondern auch alle anderen im R/3-Umfeld anfallenden Entwicklungsaufgaben realisiert werden können. Diese Vielfalt mag auf den ersten Blick überwältigend sein. Daher wird im ersten Teil die Entwicklungsumgebung ganzheitlich, aber hinreichend knapp vorgestellt.

  Im weiteren Verlauf konzentrieren wir uns dann vertieft auf die Werkzeuge und Konzepte, die für die Entwicklung von Business-Server-Pages-Applikationen von besonderer Relevanz sind. Das sind nicht nur

der Editor, das MIME-Repository oder die WebDAV-Schnittstelle, sondern auch einige weitere Werkzeuge wie z. B. die Versionsverwaltung oder die Übersetzungswerkzeuge, die den Entwickler maßgeblich bei seiner Arbeit unterstützen.

- **Kapitel 5 – Praxis: Erstellen von BSP-Applikationen**
  Der Praxisteil widmet sich vollständig der Erstellung von Webapplikationen mit dem SAP Web Application Server. Viele der vorher zur Einführung eher theoretisch vorgestellten Werkzeuge und Konzepte werden anschaulich zum Einsatz gebracht, Vor- und Nachteile an praktischen Beispielen demonstriert und auch Überlegungen zum Ausdruck gebracht, die aus den beruflichen Erfahrungen der Autoren resultieren.

  So entsteht aus einer anfänglichen Idee Schritt für Schritt eine Webapplikation, die nicht nur »heruntergeschrieben« ist, sondern auch das notwendige Grundwissen über Navigation, Design, Performanz und Sicherheitsaspekte vermittelt.

- **Anhang**
  Der Anhang bietet umfangreiche Möglichkeiten zum Nachschlagen. Es wurden nicht nur die erforderlichen Klassen und Interfaces, sondern auch das Logging, die BSP-Extensions und vieles mehr zum schnellen Nachschlagen übersichtlich gegliedert. Außerdem finden Sie hier ein Glossar mit wichtigen Begriffen aus der SAP-Welt.

Weitere Informationen zum Buch

Weitere Informationen zum Buch, Updates, ein Diskussionsforum und eventuell notwendige Korrekturen finden Sie im Internet unter *www.sap-press.de*. Dort finden Sie auch den kompletten Transportauftrag der Beispielapplikation zum Download.

Danksagung

Von der Idee, ein Buch zu schreiben, bis zur tatsächlichen Fertigstellung ist es ein langer und oft aufwändiger Weg. Viele Menschen haben uns hierbei unterstützt. Wenn wir im Folgenden den einen oder anderen nicht erwähnen, obwohl auch er seinen Beitrag zum Gelingen dieses Buches beigetragen hat, bitten wir um Nachsicht.

An aller erster Stelle danken wir unseren Kollegen und Vorgesetzten, ohne die dieses Buch nicht zustande gekommen wäre. Unseren besonderen Dank möchten wir Bernd Herth und Dr. Gangolf Haub aussprechen, die uns tatkräftig mit Ratschlägen und Hinweisen unterstützten. Herrn Heinz-Jörg Göbert mochten wir im Speziellen danken, weil wir ohne seine Zustimmung einfach nicht die ohnehin knappe Zeit gefunden hät-

ten, dieses Buch zu realisieren. Und nicht zuletzt wollen wir all den Kollegen danken, die uns oft auch in kleinen Dingen mit Verständnis und manchmal auch unbewusst geholfen haben.

Ein solches Buch benötigt auch immer technische Unterstützung. Hierfür möchten wir uns auf Seiten der SAP AG insbesondere bei Dirk Feeken für die Bereitstellung von Ressourcen und bei Rüdiger Kretschmer, Steffen Knöller und Jürgen Opgenorth für die Durchsicht des Manuskripts und die Beantwortung unserer technischen Fragen bedanken.

So sehr ein solches Buch die Autoren und die technische Unterstützung braucht, so sehr braucht es auch den Verlag, der es betreut. Sonst wären alle vorangehenden Bemühungen sprichwörtlich umsonst. Deshalb geht hier unser Dank an unseren Lektor Florian Zimniak, der insbesondere durch Geduld, Ausdauer und Verständnis das seine zu diesem Buch beigetragen hat. An dieser Stelle wollen wir uns auch bei Monika Hardt und Christel Metke für die unkomplizierte Zusammenarbeit bei der Korrektur bedanken.

Ein Dank geht auch an alle, insbesondere aus dem privaten Umfeld, die nicht erwähnt wurden, aber erwähnenswert gewesen wären.

Der letzte Punkt in Tabelle 3.4 ist der entscheidende. Alle modernen Browser haben die Möglichkeit, Cookies abzulehnen bzw. vollständig abzublocken. Serverseitige Cookies hingegen unterliegen alleine der Kontrolle durch den Server bzw. die Applikation.

Im Umgang mit serverseitigen Cookies ist es erforderlich, dass der Entwickler eine bestimmte Gewissenhaftigkeit und Sensibilität im Umgang mit den Daten an den Tag legt. Insbesondere die Dateninhalte entscheiden zum Beispiel auch darüber, ob der Anwender seine Einwilligung zur Speicherung der Daten willentlich erklären muss!

Aufgrund der technischen Vorteile bietet es sich geradezu an, die serverseitigen Cookies im Rahmen der BSP-Applikationsentwicklung zu verwenden. Deshalb werden sie im Rahmen von Kapitel 5 noch wesentlich ausführlicher besprochen.

3.3 BSP-Applikationen

BSP-Applikationen sind eigenständige Webanwendungen mit Präsentations-, Ablauf- und Anwendungslogik, die funktional in sich abgeschlossen sind. Business Server Pages ähneln in vielerlei Hinsicht den Server-Page-Technologien (*x*SP) anderer Softwarehersteller, wie beispielsweise Active Server Pages (ASP) von Microsoft und Java Server Pages (JSP) von Sun. Diese Technologie hat im Umfeld der Webentwicklung aufgrund ihrer Vorzüge eine relativ weite Verbreitung gefunden.

Vollständige Integration

BSP-Anwendungen werden auf dem SAP Web Application Server mithilfe des Web Application Builders entwickelt. Dieser ist in der Entwicklungstransaktion SE80 integriert. BSP-Applikationen können wie Standard-Applikationen auf Funktionsbausteine, Klassen, BAPIs, die Datenbank usw. zugreifen. Sie sind an das Korrektur- und Transportwesen (KTW) angeschlossen.

Die Präsentationsebene einer solchen BSP-Applikation, auf der die eigentliche Darstellung stattfindet (in diesem Fall auf dem Client-Browser) wird aus einer Abfolge von Webseiten gebildet. Diese setzt sich aus folgenden Elementen zusammen:

- **statische Webseiten**
  Diese Webseiten enthalten kein serverseitiges Scripting.
- **dynamisch generierte Webseiten**
  Diese Webseiten enthalten serverseitiges Scripting und werden erst auf Anforderung zur Laufzeit vom Applikationsserver zu einer fertigen Webseite »zusammengebaut«.

- **MIME-Objekte**
  Dies sind z. B. Bilder, Symbole, Sound-Dateien und Stylesheets, die in die Webseiten eingebunden werden können.

Des Weiteren können clientseitiges JavaScript für dynamische Aktionen ohne Server-Roundtrip[7] und clientseitige Cookies zum Zwischenspeichern von Informationen bei Bedarf eine Rolle spielen.

Damit die Webapplikation tatsächlich auf dem Client ablaufen kann, sind eine Reihe von Komponenten und Mechanismen notwendig, die auf dem Applikationsserver miteinander interagieren und zur Anwendung gebracht werden müssen.

In diesem Kapitel werden die einzelnen Komponenten, die Steuerung und der Verarbeitungsablauf solcher BSP-Applikationen vorgestellt. Des Weiteren wird das *Model-View-Controller-Design-Pattern* (*MVC*) vorgestellt, das eine interessante Alternative zur bisherigen BSP-Programmierung darstellt.

3.3.1 Komponenten

Unterhalb der bereits beschriebenen Präsentationsschicht, d. h. auf dem Applikationsserver, befinden sich die einzelnen Bestandteile der BSP-Applikation. Zur Laufzeit werden diese Bestandteile verarbeitet und an den Client, in einem für ihn verständlichen Format, gesendet. Die Antwort des Clients kann selbstverständlich auch innerhalb der BSP-Applikation verarbeitet werden. Eine BSP-Applikation setzt sich aus mehreren oder allen der folgenden Komponenten zusammen (siehe auch Abbildung 3.8):

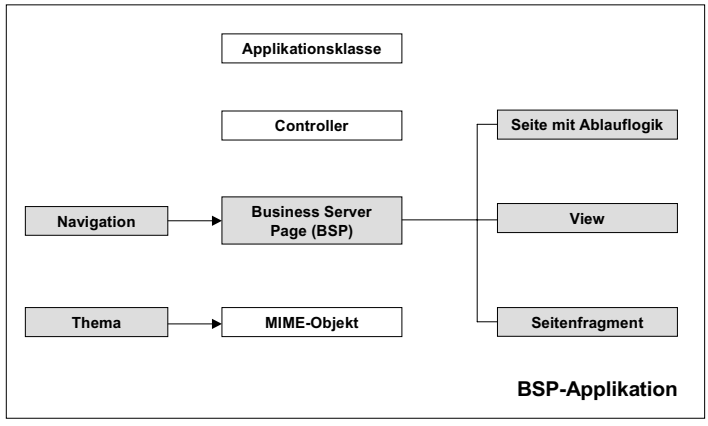

Abbildung 3.8 Bestandteile einer BSP-Applikation

7 *Server-Roundtrip* bedeutet, dass zur Verarbeitung von Interaktionen Daten mit dem Applikationsserver ausgetauscht werden. Dies ist je nach Browser und Coding nicht zwingend notwendig.

Statisches HTML

Natürlich ist es möglich, mit der Entwicklungsumgebung einige statische HTML-Seiten mit Hyperlinks zu verknüpfen, die in einer mehr oder weniger vorgegebenen Reihenfolge vom Anwender aufgerufen werden. Das wäre allerdings eine sehr einfache Applikation, die sich auch, ein wenig Übung vorausgesetzt, mit einem einfachen Texteditor erstellen ließe. Dazu benötigt man keinen SAP Web Application Server, ein wenig Webspace bei einem Internetprovider wäre vollkommen ausreichend. Hierbei stößt man aber sehr schnell an die Grenzen des Möglichen. Sobald das Zurückschreiben von Daten und der Umgang mit dynamischen Inhalten erforderlich ist (wie z. B. aus Datenbanken generiert), kommt man an der Trennung von Layout und Daten sowie weiteren dynamischen Elementen, z. B. für Eingabeprüfung und Auswahlhilfe, nicht vorbei.

Daher ist es in einem ersten Schritt erforderlich, die logische Einheit einer BSP-Applikation in Präsentationskomponenten, Ablaufsteuerung und Anwendungslogik zu unterteilen. Darüber hinaus beinhaltet eine BSP-Applikation eine Reihe von Verwaltungsattributen.

In der tatsächlichen Umsetzung lässt sich die saubere Trennung natürlich nicht immer vollständig umsetzen. Bei der Verwendung von BSP-Seiten wird man beispielsweise selten vermeiden können, Ablauflogik im Layout einzubinden. Das ist ein Umstand, den man mit dem Einsatz des *MVC-Design-Pattern* vermeiden kann. Aufgrund der besonderen Stellung des MVC-Konzepts wird diesem Thema ein gesonderter Abschnitt gewidmet (siehe Abschnitt 3.3.4). Auf eine detaillierte Erläuterung der MVC-Komponenten wird deshalb an dieser Stelle verzichtet.

Alle nachfolgend aufgeführten Objekte sind als Teil der BSP-Applikation in das SAP Korrektur- und Transportwesen (KTW) integriert und werden als logische Einheit behandelt. Damit können alle Objekte einer BSP-Applikation vollständig und konsistent zwischen SAP-Systemen transportiert werden.

Eigenschaften/Verwaltungsattribute

Jede BSP-Applikation besitzt eine Reihe von Verwaltungsattributen, die ihre allgemeinen Eigenschaften beschreiben. Hierzu gehören beispielsweise die Zuordnung von Paket, Thema und Applikationsklasse, das HTTPS-Flag usw. Diese werden auf der Eigenschaftsseite der BSP-Applikation definiert. In Kapitel 5 werden diese Punkte detailliert behandelt.

Präsentationskomponenten

Dieser Abschnitt beschreibt die Komponenten, die zur Erzeugung der grafischen Darstellung auf dem Bildschirm dienen.

BSP-Seite

Die BSP-Seiten sind die Grundlage für die Inhalte, die letztlich im Browser des Clients angezeigt werden. Sie können statischen Web-Code[8] und dynamischen Scripting-Code (ABAP) enthalten. Dieser Scripting-Code wird zum Zeitpunkt der Generierung bzw. Verarbeitung auf dem Server in browserverständlichen Code (z. B. HTML) transformiert. So ist es möglich, das endgültige Aussehen der Seite erst zur Laufzeit, d. h. zum Zeitpunkt der Anforderung, festzulegen.

Eine BSP-Seite kann die folgenden Ausprägungen besitzen:

- **Seite mit Ablauflogik**
  Dies sind Seiten, deren Ablauf durch Eventhandler gesteuert wird (Eventhandler-basiertes Modell). Üblicherweise sollte sich in den Seiten wenig Anwendungslogik befinden. Hierfür ist stattdessen ein spezielles Konstrukt, die Applikationsklasse, zuständig, die die Kapselung der notwendigen Business-Logik erlaubt. Eine Seite mit Ablauflogikimplementiert neben dem Layout über die Eventhandler die Ablauflogik und verfügt außerdem über Seitenattribute und Typdefinitionen.

  Seitenattribut

  Seitenattribute sind global auf einer BSP-Seite zur Verfügung stehende Variablen. Sie stehen im Layout und in allen Eventhandlern zur Verfügung und »merken« sich ihren Wert über die gesamte Laufzeit des Requests. Der Inhalt eines im Event gefüllten Seitenattributs kann also ohne weiteres im Layout-Coding zur Ausgabe gebracht werden. Seitenattribute, bei denen die Eigenschaft `Auto` aktiviert ist, werden nach Erzeugung des Seitenobjekts automatisch mit dem Wert des namensgleichen Parameters gefüllt, wenn ein solcher an die Seite übergeben wurde. Seitenattribute können jeden elementaren Typ (ausser XSTRING), Struktur- und Tabellentyp annehmen.

  Typdefinition

  In den Typdefinition der BSP-Seite können beliebige Typdefinitionen erzeugt werden, auf die zu jedem Zeitpunkt der Seite zugegriffen werden kann. Z. B. kann der Typ einer internen Tabelle definiert werden, um damit ein Seitenattribut zu typisieren.

8 Wenn im weiteren Verlauf in den Beispielen nur HTML-Code verwendet wird, bedeutet das nicht, dass andere Standards wie XML, WML oder XHTML nicht genauso gut zum Einsatz gebracht werden könnten.

Die Seiten des Typs »Seite mit Ablauflogik« sind ausführbar und können über eine URL oder über die Navigation aus einer anderen Seite angesprochen werden. In Kapitel 5 wird auf ihre Programmierung detailliert eingegangen. Eine vollständige BSP-Applikation kann übrigens bei Bedarf ausschließlich aus Seiten mit Ablauflogik und den dazugehörigen Eventhandlern aufgebaut werden.

- **Seitenfragment**
  Seitenfragmente werden wie normale BSP-Seiten angelegt, aber als Seitenfragment gekennzeichnet. Von normalen Seiten unterscheiden sie sich lediglich dadurch, dass sie keine eigenständige Event-Behandlung, keine Typdefinition und keine eigenen Attribute besitzen. Sie werden ausschließlich von anderen BSP-Seiten mittels der `include`-Direktive als einfaches Text-Include eingebunden. Sie erben von diesen die Seitenattribute.

  Seitenfragmente können wiederum selbst Seitenfragmente inkludieren, aber keine Seiten mit Ablauflogik. Seitenfragmente erlauben den modularen Aufbau des Layouts von BSP-Seiten und steigern dadurch den Anteil von wiederverwendbarem Programmcode.

- **View (MVC)**
  Views dienen zur Visualisierung von Daten, die im Rahmen des Model-View-Controller-Design-Pattern zur Verfügung gestellt werden (siehe Abschnitt 3.3.4). Ihr Aufruf erfolgt fast ausschließlich durch Controller. Daher besitzen sie keine eigene URL. Views besitzen wie Seiten mit Ablauflogik Seitenattribute. Im Gegensatz zu diesen können Views allerdings nicht automatisch beim Aufruf über das `Auto`-Flag gefüllt werden. Für das Füllen der Attribute, für die Annahme des Requests und die Ablaufsteuerung ist der zugeordnete Controller zuständig.

MIME-Objekte

MIME (*Multipurpose Internet Mail Extensions*) ist eine Erweiterung des ursprünglichen Internet-E-Mail-Protokolls, das den Austausch unterschiedlicher Datenarten im Internet ermöglicht. Dies beinhaltet u.a. Audio-, Video- und Bilddaten, Cascading Style Sheets, Anwendungsprogramme und ASCII-Dateien. Client-Browser sind in der Lage, diese Objekttypen entweder mittels Plug-Ins oder durch integrierte Anwendungen zu verarbeiten. So können die gängigen Browser z. B. die meisten Grafikformate ohne externe Hilfsmittel anzeigen. Andere Objekttypen wie beispielsweise Flash-Animationen benötigen Plug-Ins.

Mit jeder neu angelegten BSP-Applikation wird ein gleichnamiges Verzeichnis im MIME-Repository angelegt. Es dient als Ablage für alle anwendungsspezifischen MIME-Objekte. Über dieses Repository werden die MIMEs zentral verwaltet.

Themen

Themen dienen dazu, Ersetzungen verwendeter MIME-Objekte zu definieren. Damit kann das Erscheinungsbild von BSP-Applikationen nachträglich verändert werden, ohne Änderungen am Layout-Quelltext durchführen zu müssen. Jedes MIME-Objekt innerhalb einer BSP-Applikation kann so durch ein anderes Objekt aus dem lokalen File-System ersetzt werden. Mit dem Thema-Konzept ist es auf einfache Art und Weise möglich, das Layout der Seiten einer BSP-Applikation – auch nachträglich – an individuelle Wünsche anzupassen.

Ein Thema wird als eigenständiges Entwicklungsobjekt im Web Application Builder angelegt und dient als Container für alle Ersetzungsdefinitionen. Damit sich die Änderungen zur Laufzeit auswirken, muss das Thema der entsprechenden BSP-Applikation explizit zugeordnet werden.

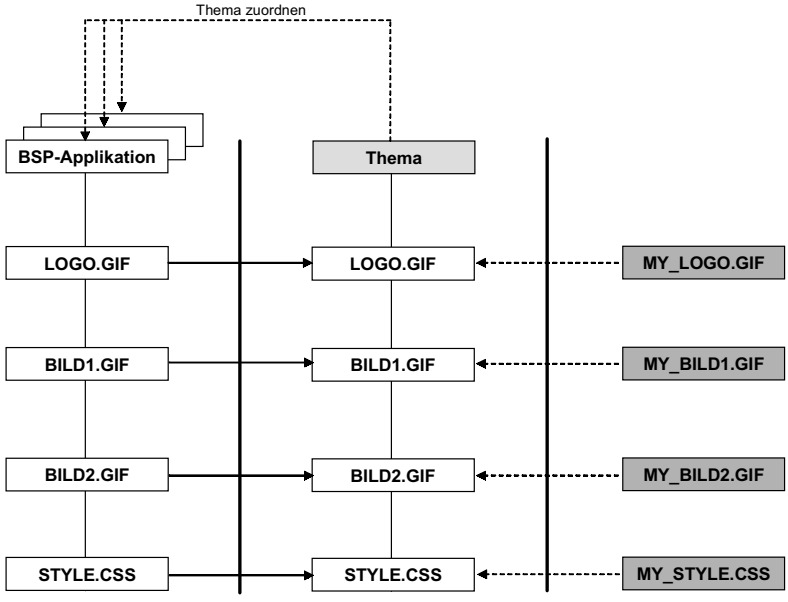

Abbildung 3.9 Ersetzung von MIME-Objekten mittels eines Themas

Ersetzung zur Laufzeit

Sobald ein MIME-Objekt von einer Seite angefordert wird, ermittelt die BSP-Laufzeitumgebung, ob der laufenden BSP-Applikation ein Thema zugeordnet ist. Ist dies der Fall, wird das korrekte Objekt aus der Ersetzungsdefinition ermittelt. Dieses Objekt wird dann anstelle des in der BSP-Applikation eingetragenen Objekts zur Laufzeit der BSP-Applikation zum Client übermittelt. In Abbildung 3.9 würde das zum Client zu sendende MIME-Objekt *LOGO.GIF* zur Laufzeit durch *MY_LOGO.GIF* ersetzt werden, wenn das Thema der aktiven BSP-Applikation zugeordnet ist. Für die Arbeit mit Themen steht der *Themen-Editor* zur Verfügung.

BSP-Extensions

Während des Erstellens einer BSP-Applikation stellt sich immer wieder die Frage, wie das Corporate Design einer solchen Applikation sichergestellt werden kann. Die Cascading Style Sheets sind hier ein sehr bewährtes Konzept. Allerdings droht bei größeren Projekten mit mehreren Entwicklern die Gefahr, dass die definierten Style-Anweisungen nicht korrekt verwendet werden. Im ungünstigsten Fall müssen auf jeder BSP-Seite für jedes HTML-Element die passenden CSS-Elemente immer wieder neu zugewiesen werden. Dies ist eine langwierige Angelegenheit, bei der sich leicht Fehler einschleichen können. Nicht zuletzt leidet auch die Übersichtlichkeit des HTML-Codings darunter, was nachträgliche Änderungen erschwert.

BSP-Extensions helfen hier weiter. Sie sind eine Abstraktionstechnik, mit der sowohl die Syntax als auch die Semantik von HTML-Codeblöcken vereinfacht werden kann. Dieser Mechanismus ist offen strukturiert und kann z. B. auch in XML- und WML-Codestrecken verwendet werden.

Container für BSP-Elemente

Eine BSP-Extension ist ein Container für BSP-Elemente. Jedes Element ist im BSP-Kontext einer ABAP-Klasse zugeordnet. In dieser Klasse ist die Funktionalität eingebettet, mit der das clientseitig zur Ausführung gebrachte Coding erzeugt und dessen Funktionalitäten realisiert werden. Die Definitionen der Elemente und die Abbildung auf die ABAP-Klassen sind flexibel. Mit dieser Technologie kann eine Vielzahl von Anforderungen erfüllt werden, die sich nicht auf grafische Objekte beschränkt.

Extension-Infrastruktur

Der SAP Web Application Server stellt eine Infrastruktur zur Verwaltung von BSP-Extensions zur Verfügung. Daher können diese relativ einfach im Rahmen von BSP-Applikationen eingesetzt werden. Mit dem SAP Web Application Server 6.20 werden viele vordefinierte Extensions ausgeliefert, z. B. HTML Business für BSP (HTMLB). Diese Extensions sind beliebig erweiterbar und können für eigene Zwecke angepasst werden. Selbstver-

ständlich können auch eigene BSP-Extensions entwickelt werden. Die BSP-Extensions können über den in die Entwicklungsumgebung integrierten Editor erstellt und bearbeitet werden.

Verwenden Sie die HTMLB-Extensions als Vorlage, um daraus eigene Extensions zu erzeugen. So können Sie sich einen Teil der Arbeit ersparen.

Die Verwendung von BSP-Extensions in BSP-Seiten erfolgt mittels der `extension`-Direktive. Zusätzlich können Komposit-Elemente erstellt werden, die mehrere BSP-Elemente in einer Untermenge vereinen können. Damit ist es möglich, alle BSP-Elemente bei Layoutänderungen gleichzeitig zu beeinflussen. Das reduziert den Aufwand bei komplizierten BSP-Anwendungen.

Jede BSP-Extension besteht aus einer Sammlung von BSP-Elementen. Jedes dieser Elemente hat definierte Attribute und ist einer ABAP-Klasse zugeordnet. Die im Element bereitgestellten Attribute stellen die Eingabeparameter für die zugeordnete ABAP-Klasse dar und dienen der Beeinflussung des Aussehens, des Eingabeverhaltens und weiterer Funktionalitäten. Das Einfügen von BSP-Elementen auf BSP-Seiten erfolgt in XML-Notation.

Im ausgehenden HTML-Datenstrom schreibt die Elementklasse das serialisierte HTML-Coding auf der Grundlage der vom Element gelieferten Funktionalität. Dabei wird davon ausgegangen, dass alle Elemente in einer Extension einen gemeinsamen Ausgabestil unterstützen.

Vorteile von BSP-Extensions

Die Verwendung von BSP-Extensions bietet folgende Vorteile:

- Das HTML-Coding muss nur einmal entwickelt werden. Änderungen wirken sich unverzüglich auf alle Aufrufe des Elements aus. Das gilt auch über Applikationsgrenzen hinweg. Damit erhöht sich die Wiederverwendbarkeit im Hinblick auf das Corporate Design.
- Die dem Element zugeordnete ABAP-Klasse (Elementklasse) kann zusätzliche Logik beinhalten, um browserabhängigen HTML-Code zu generieren. Das vermeidet browserabhängiges Coding im Layout.
- Auch die Stylesheet-Zuweisungen befinden sich in der Elementklasse. Dadurch, dass die CSS-Zuweisungen an *einer* bestimmten Stelle stattfinden, ist gewährleistet, dass das generierte HTML-Coding korrekte Referenzen auf die Stylesheets setzt.
- Die Standard-XML-Syntax kann – im Gegensatz zum HTML-Coding im Layout – geparst und bereits zum Zeitpunkt der Generierung geprüft werden. Dadurch werden Fehler vermieden.

Neben einer BSP-Extension für Standard-HTML-Elemente wie Buttons, Eingabefelder, Dropdown-Listen u.Ä. können auch stark spezialisierte Extensions realisiert werden. Eine solche Extension könnte beispielsweise ein Komposit-Element beherbergen, das einen vollständigen Newsticker einschließlich Generierungsrahmen und Anwendungslogik realisiert. Der Newsticker kann dann, per `extension`-Direktive mit all seinen Attributen parametrisiert, in beliebigen BSP-Applikationen zur Verfügung gestellt werden.

Die Vorteile von BSP-Extensions werden allerdings durch einen erhöhten Programmieraufwand in der Realisierung derartiger BSP-Extensions erzielt. Daher ist diese Technologie im Normalfall nur für größere Projekte lohnenswert. Wenn allerdings Wert auf ein eindeutiges Corporate Design gelegt wird, sollte in keinem Fall darauf verzichtet werden.

Komponenten der Ablaufsteuerung

Die Ablaufsteuerung eines Programms bestimmt den zeitlichen und logischen Ablauf einer Anwendung. Wann welche Komponenten eines Programms zur Ausführung gebracht werden, ist im Falle von BSP-Applikationen teilweise fest vorgegeben und teilweise durch den Entwickler steuerbar. Zu der ersten Gruppe gehören die Eventhandler, die Bestandteil von BSP-Seiten mit Ablauflogik sind. Zur zweiten Gruppe gehören beispielsweise Navigationsstruktur und Controller.

Eventhandler

Die Eventhandler werden zu bestimmten Zeitpunkten der Laufzeit einer BSP-Seite in fest definierter Reihenfolge zur Ausführung gebracht. Sie werden mit ABAP-Coding gefüllt und erlauben den Zugriff zur Laufzeit auf bestimmte Objekte wie die Applikationsklasse oder auf bestimmte Laufzeitobjekte, um z.B. den Zugriff auf Request-Informationen zu ermöglichen. In Abschnitt 3.3.3 wird dieses Thema ausführlich behandelt.

Navigationsstruktur

Die Navigationsstruktur wird zur Definition von Navigations-Requests verwendet. Diese beschreiben Start und Ziel eines Requests, d.h. von welcher Seite zu welcher Folgeseite navigiert werden soll. Durch die Zuordnung der Seiten über Navigations-Requests wird eine rein formale Beschreibung des Navigationsschemas innerhalb einer BSP-Applikation erreicht. Damit ist es z.B. möglich, die Ablaufsteuerung einer BSP-Applikation ohne Eingriff in das Coding zu ändern.

Controller (MVC)

Controller sind ein weiterer Bestandteil des MVC-Design-Pattern. Sie werten aus den Daten eines eingehenden Requests auf der Grundlage eines Models einen passenden View aus. Dieser wird dann für den Response gerendert. Sie stellen das Bindeglied zwischen Model und View dar.

Komponenten der Anwendungslogik

Die Anwendungslogik (Business-Logik) kümmert sich um die eigentliche Bereitstellung und Verarbeitung der Daten. Sie kann, wie in der herkömmlichen ABAP-Programmierung üblich, in Form von BAPIs, Funktionsbausteinen oder Klassenbibliotheken aus einer BSP-Applikation, vorzugsweise aus den Eventhandlern heraus, angesprochen werden.

Der SAP Web AS bietet hierzu zusätzliche Strukturierungshilfsmittel – die BSP-Applikationsklasse und das MVC – an, die zur Kapselung der benötigten Anwendungslogik genutzt werden können.

Es ist sogar möglich, die Anwendungslogik im Layout-Teil einer BSP-Seite unterzubringen. Davon können wir allerdings nur abraten!

Applikationsklasse

Kapselung der Anwendungslogik

Die Applikationsklasse dient der Kapselung der Anwendungslogik einer BSP-Applikation. Die Realisierung erfolgt mittels einer üblichen globalen ABAP-Klasse. Sie ruft z. B. per BAPI Business-Daten aus Backend-Systemen auf und schreibt diese nach Verarbeitung durch die BSP-Applikation zurück. Eine solche Applikationsklasse wird der BSP-Applikation zugeordnet und steht dann jeder Seite der BSP-Applikation mit ihren Komponenten (Attribute, Methoden usw.) über die typisierte Objektreferenz `application` direkt zur Verfügung. Dies geschieht automatisch. Es ist weder eine »manuelle« Deklaration noch eine Instanziierung vor der Verwendung notwendig.

Die in Applikationsklassen normalerweise realisierten Aufgaben einer BSP-Applikation können beispielsweise folgende sein:

- Seitenübergreifende Speicherung von BSP-Applikationsdaten in Form von Attributen
- Kapselung der Anwendungslogik in Methoden
- Verschalung wiederkehrender Aufgaben (z. B. Prüfung von Berechtigungen, komplexe Eingabeprüfungen, Sichern und Wiederherstellen von Daten mithilfe serverseitiger Cookies) in Methoden

Eine Applikationsklasse kann in beliebig vielen Applikationen und selbstverständlich auch in normalen ABAP-Programmen verwendet werden. Allerdings kann pro BSP-Applikation nur eine Applikationsklasse existieren. Die Zuordnung erfolgt, wie in Abbildung 3.10 dargestellt, über die Eigenschaftsseite der BSP-Applikation.

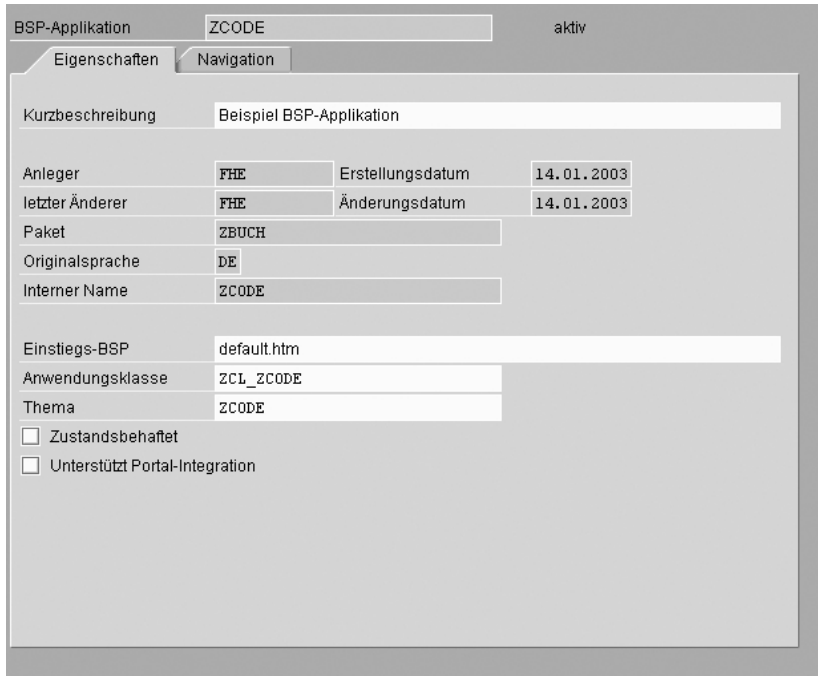

Abbildung 3.10 Zuordnung der Anwendungsklasse

Sie können natürlich neben der Applikationsklasse weitere Klassen in eine BSP-Applikation einbinden.

Innerhalb der Applikationsklasse werden normalerweise bereits existierende Anwendungsfunktionalitäten des SAP Application Servers bzw. der Backend-Systeme verschalt. Das Coding wird dadurch übersichtlicher und wartungsfreundlicher.

Bei der Verwendung von Applikationsklassen ist Folgendes zu berücksichtigen:

- Ist die Applikationsklasse einer *verbindungsorientierten* (*stateful*) BSP-Applikation zugeordnet, ist ihre Lebensdauer genau so lang wie die der BSP-Applikation. Sie ist dann ideal für die Datenhaltung geeignet.

- Ist die Applikationsklasse einer *zustandslosen* (*stateless*) BSP-Applikation zugeordnet, ist ihre Lebensdauer genau so lang wie die einer BSP-Seite, d.h. vom Eingang des Requests bis zum Abschluss des `response`-Objekts. Sie ist für die Datenhaltung ungeeignet, da sie wie alle Objekte direkt im Anschluss zerstört wird. In diesem Fall setzt man für die Datenhaltung in der Regel serverseitige Cookies ein.
- Applikationsklassen sind singleton. Pro Session kann maximal eine Instanz existieren.
- Eine BSP-Applikation kann maximal eine Applikationsklasse zugeordnet bekommen.
- Der Konstruktur muss parameterlos sein.

Im Rahmen der Datenbeschaffung muss man also bereits abwägen, wie man die Implementierung realisieren will. Es ist beispielsweise nicht sehr sinnvoll, in einer stateless BSP-Applikation den kompletten Materialstamm des Backend-Systems in der Applikationsklasse zu beschaffen. Dieser Vorgang würde dann bei jedem Seitenwechsel erfolgen und die Performance nachhaltig einschränken. In einer stateful Applikation wiederum könnte es durchaus sinnvoll sein, denn die Daten würden nur einmal beim ersten Aufruf der Applikation, wenn das `application`-Objekt instanziiert wird, vom Backend geladen werden.

Applikationsereignisse

In der Applikationsklasse kann das Interface `IF_BSP_APPLICATION_EVENTS` implementiert werden. Dadurch werden weitere Verarbeitungszeitpunkte einer BSP-Applikation über Methoden zugänglich gemacht, die eine sehr flexible Steuerung ermöglichen und einen Eingriff in die Verarbeitungslogik gestatten. Die einzelnen Methoden des Interfaces werden in Anhang A.2 vorgestellt.

Applikationsbasisklasse

Es kann sinnvoll sein, die Applikationsklasse von der vordefinierten Basisklasse `CL_BSP_APPLICATION` abzuleiten. Diese Klasse bietet Methoden, mit denen z.B. Session-Timeout, aktuelle URL zur BSP-Applikation, Zustandsmodus usw. abgerufen bzw. gesetzt werden können. Die einzelnen Methoden des zugehörigen Interfaces werden ebenfalls in Anhang A.2 vorgestellt.

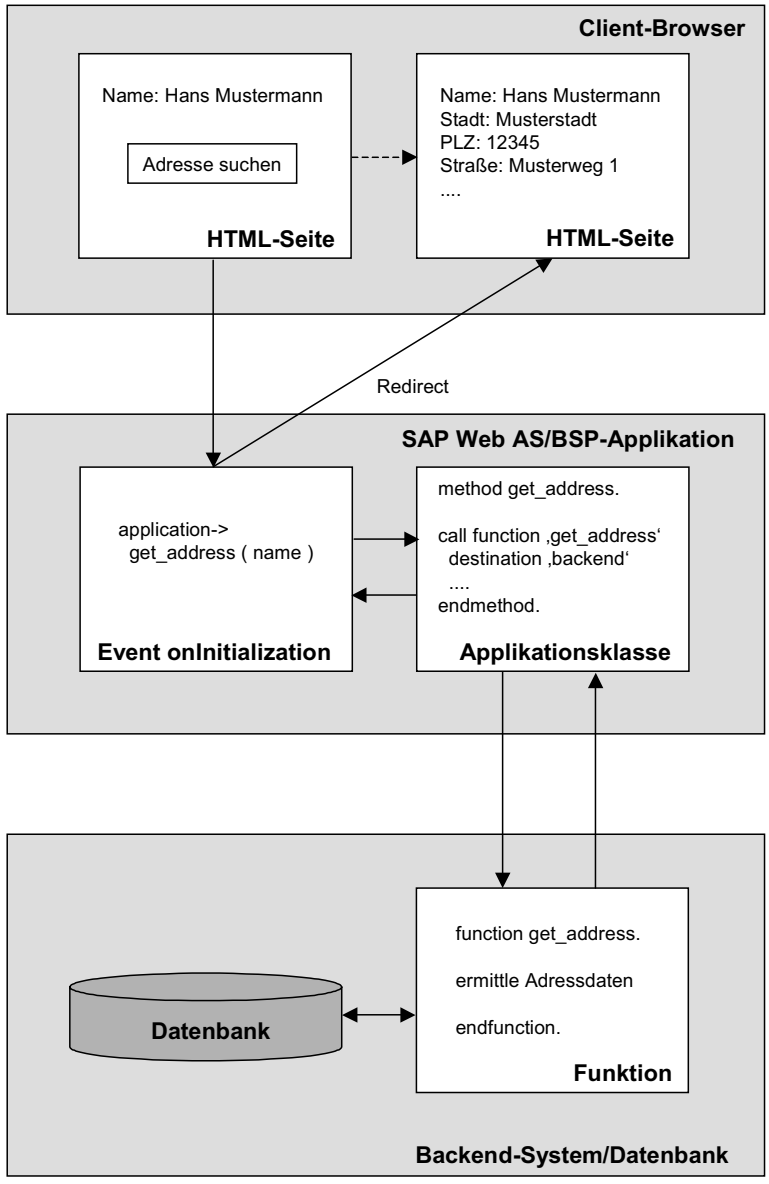

Abbildung 3.11 Typischer Ablauf im Falle gekapselter Anwendungslogik

Ein Beispiel

Abbildung 3.11 verdeutlicht beispielhaft anhand eines vereinfachten Szenarios den Ablauf eines Requests mit gekapselter Anwendungslogik. Auf der HTML-Seite 1 wird die Suche nach einer Adresse ausgelöst. Mittels des Events `onInputProcessing` wird die Anfrage auf die BSP-Seite HTML-Seite 2 gelenkt. Hierbei wird der Name als Auto-Seitenattribut

übergeben. Der Event `onInitialization` ruft die Applikationsklasse auf, die wiederum im Backend einen RFC-Aufruf zur Datenbeschaffung ansteuert. Das Ergebnis wird an die Applikationsklasse zurückgegeben und von dort aus dem Eventhandler zur Verfügung gestellt. Die jetzt gefüllten Attribute werden im Layout-Coding ausgegeben und erscheinen in dieser Form im Client-Browser als HTML-Seite 2.

Model (MVC)
Das Model dient als Anwendungsobjekt der Steuerung des Verhaltens und der Anwendungsdaten. Es antwortet sowohl auf Informationsrequests über seinen Status, die meist vom View kommen, als auch auf Anweisungen für Zustandsänderungen, die in der Regel vom Controller gesendet werden.

Das Model kennt weder seine Views noch seine Controller. Damit dient das Model allein der internen Datenverarbeitung, ohne auf die Anwendung mit ihrer Benutzeroberfläche Bezug zu nehmen. Konkret wird ein Model durch eine Referenz des Controllers auf eine Klasse festgelegt.

3.3.2 Zugriff auf eine BSP-Applikation

Im Folgenden werden die Besonderheiten des Zugriffs auf eine BSP-Applikation erläutert. Einen Überblick über die Standard-Fehlermeldungen gibt der daran anschließende Abschnitt. Zum Abschluss werden die systemspezifischen URL-Parameter vorgestellt, mit denen der Aufruf einer Applikation konfiguriert und den eigenen Bedürfnissen angepasst werden kann.

Adressierung

Eine BSP-Applikation wird durch eine URL (*Uniform Resource Locator*) zur Ausführung gebracht. Man kann die Applikation direkt durch Eingabe der URL in der Adressleiste des Client-Browsers aufrufen.

Es ist auch möglich, die Applikation als so genannten *Favoriten* abzuspeichern und bequem über die Favoritenverwaltung aufzurufen. Sie kann auch wie normale Verknüpfungen auf dem Desktop abgelegt werden. Hierbei können auch Name/Value-Paare (siehe unten) zur Parametrisierung mit angegeben werden. Damit kann die Webapplikation parametrisiert aufgerufen werden.

Die URL einer BSP-Applikation hat folgenden allgemeinen Aufbau:

Bestandteile der URL

```
<Protokoll>://<Host>:<Port>/<Namensraum>/<Applikations-
name>/<Seite>?<URL-Parameter>
```

- **Protokoll**
  BSP-Applikationen unterstützen die Protokolle HTTP und HTTPS. Falls HTTPS eingesetzt werden soll, muss dieses Protokoll als Service im ICM vorhanden sein. Hierzu ist die Konfiguration per Instanzprofil sowie die Installation der entsprechenden Systemdateien erforderlich.[9]
- **Host**
  Für »Host« wird der Name des Applikationsservers eingesetzt, auf dem die Applikation ausgeführt werden soll. Es ist entweder die IP-Adresse des Hosts oder der DNS-Name einschließlich Netzwerkdomäne anzugeben.
- **Port**
  Es folgt die Portnummer. Sie spezifiziert den Port, auf dem die Anwendung laufen soll, falls der voreingestellte Wert umgangen werden soll. Den Ports werden die dazugehörigen Protokolle über die Profileinstellungen (Profilparameter icm/server_port_<xx>) zugewiesen.
- **Namensraum**
  Der Namensraum ist die Namensraumkennung der BSP-Applikation. SAP-Applikationen werden im Namensraum **sap** ausgeliefert. BSP-Applikationen können in einem eigenen Namensraum angelegt werden.
- **Applikationsname**
  Der Applikationsname repräsentiert den Namen einer BSP-Applikation, so wie er in der Entwicklungsumgebung definiert wurde.
- **Seite**
  Hier erfolgt der Name der gewünschten Zielseite der BSP-Applikation. Das kann eine BSP-Seite, eine statische Seite oder ein Controller des MVC sein. Sinnvollerweise sollte man die Einstiegsseite verwenden, um sicherzustellen, dass die Applikation auch korrekt initialisiert wird.

9 Die Erklärung der Vorgehensweise ist nicht Bestandteil dieses Buches. Entsprechende Installationsleitfäden finden sich in der Dokumentation zum SAP Web Application Server unter »Sicherheit beim SAP Web Application Server – Verwendung des Secure-Sockets-Layer-Protokolls« oder im Internet unter *http://service.sap.com/instguides*.
Das Installationspaket steht im SAP Service Marketplace unter *http://service.sap.com/swcenter* autorisierten Kunden für den Einsatz der SAP Cryptographic Library zur Verfügung.

- **URL-Parameter**
  Der Applikation können auch Parameter als so genannte *Name/Value-Paare* beim Aufruf mitgegeben werden. Diese können systemspezifisch oder applikationsbezogen sein (siehe weiter unten in diesem Abschnitt). Die Trennung von der eigentlichen URL erfolgt durch das Fragezeichen (`?`).

Die letzten beiden Bestandteile können auch weggelassen werden. In diesem Fall erfolgt der Einstieg über die Standardseite, die in den BSP-Applikationseigenschaften angegeben wurde. Leerzeichen sind nicht zulässig, können aber über die so genannte *Escaped-URL* dennoch realisiert werden. Hierbei wird das Leerzeichen als `%20` dargestellt.

Eine URL im SAP-Namensraum könnte dann wie folgt aussehen:

http://www.my-webas.de:8080/sap/bc/bsp/sap/zcode/start.html?var_1=init

Da diese langen URLs relativ unbequem sind, besteht die Möglichkeit, einen externen Alias im ICF zu definieren. Der Alias bildet dann die Adresse ab und kann in Bezug auf das letzte Beispiel wie folgt aussehen:

http://www.my-webas.de/zcode

Fehlermeldungen

Für den Fall, dass die BSP-Applikation nicht erreicht oder ausgeführt werden kann, wird vom System ein Standardfehler in Form von HTTP-Error-Codes als Antwort gesendet. Der bekannteste Fehler dürfte der Fehlercode HTTP-Error `404` sein, der gesendet wird, falls die angeforderte Ressource nicht gefunden werden konnte. Der Code `500 - interner Serverfehler` ist während der Entwicklung besonders wichtig. In Anhang A.1 finden Sie eine Auflistung der durch den SAP Web Application Server erzeugten Fehlercodes.

Um aussagekräftige Fehlermeldungen, insbesondere auch während der Entwicklung in Bezug auf den Fehlercode `500`, vom Server zu erhalten, sollte die Option **Kurze HTTP-Fehlermeldungen anzeigen** im Internet Explorer deaktiviert sein (siehe Abbildung 3.12). Ansonsten werden vom Internet Explorer »userfreundlichere« Fehlermeldungen ausgegeben, deren praktischer Nutzen während der Entwicklung leider gegen null strebt.

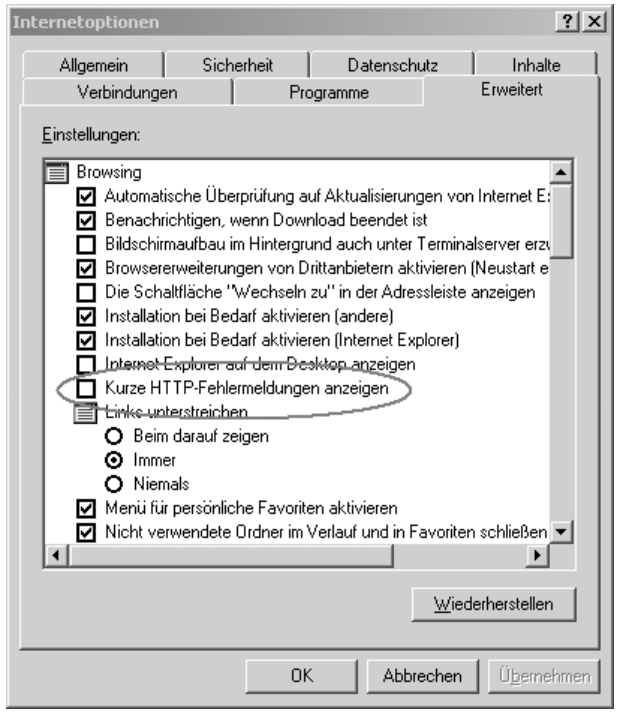

Abbildung 3.12 Anzeige aussagekräftiger Fehlermeldungen des Servers beim Internet Explorer

Reaktion auf Anmelde- und Anwendungsfehler

Der SAP Web Application Server bietet außerdem die Möglichkeit, auf Anmelde- und Anwendungsfehler zu reagieren. Dabei gibt es zwei Alternativen: Zum einen können eigene Fehlerseiten erstellt werden, die dem Anwender weitergehende Informationen bezüglich der aufgetretenen Situation oder Lösungsvorschläge geben können. Zum anderen kann der Anwender mittels eines Redirects zu einer anderen URL-Adresse umgeleitet werden. Bei einem Redirect können zusätzlich die Formfelder, die an die Zielseite bereits übermittelt wurden, an die neue Adresse mit übergeben werden.

Dies ermöglicht eine sofortige Anzeige der Parameter, die beispielsweise für ein Fehlverhalten der Applikation sorgen. Das erspart unter Umständen einiges an Zeitaufwand für lange Debugging-Sitzungen.

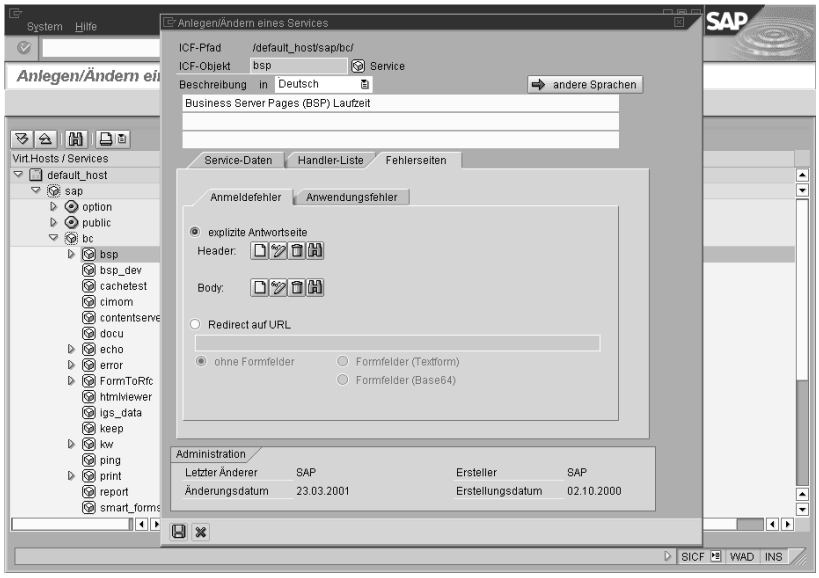

Abbildung 3.13 Anlegen von eigenen Fehlerseiten bzw. eines Redirects in der Transaktion SICF

URL-Parameter

Das Verhalten einer BSP-Applikation lässt sich über die URL steuern. Dazu kann diese URL um einen Query-String erweitert werden. Als Query-String wird der Teil einer URL bezeichnet, der nach einem Fragezeichen (`?`) in der URL aufgeführt wird.

Die Parameternamen und ihre zugehörigen Werte sind case-insensitive. Eine Ausnahme bildet der Parameter `sap-exiturl`, falls auf einen case-sensitiven Server verwiesen wird.

Der Query-String enthält eine Folge von Name/Value-Paaren, die durch das kaufmännische Und (`&`) getrennt sind, z. B.:

http://www.my-webas.de:8080/sap/bc/bsp/sap/zcode/start.htm?zustand=0&org=sap

Hier wird die Adresse der BSP-Applikation um zwei Query-String-Parameter `zustand` und `org` mit den Werten »0« und »sap« erweitert. Dies hat sinngemäß denselben Effekt wie die Wertzuweisung im Coding mittels:

```
zustand = '0'.
org     = 'sap'.
```

Systemspezifische URL-Parameter

Der SAP Web Application Server kennt eine Reihe von systemspezifischen URL-Parametern. Sie werden vom Server automatisch bei jedem Aufruf überprüft und beeinflussen zum Teil entscheidend das Verhalten der Applikation.

Die möglichen Systemparameter haben grundsätzlich die folgende Struktur:

```
sap-<parameter-name>=<wert>
```

Es lassen sich mehrere Systemparameter kombinieren.

Im Folgenden werden die einzelnen Parameter vorgestellt und ihre Einsatzmöglichkeiten an Beispielen veranschaulicht.

- `sap-sessioncmd`
  Mit dem Wert »open« wird eine laufende BSP-Applikation neu bzw. für den Fall, dass sie noch nicht läuft, erstmalig gestartet.

  http://www.my-webas.de:8080/sap/bc/bsp/sap/zcode/start.htm?sap-sessioncmd=open

  Mit dem Wert »close« wird eine laufende BSP-Applikation beendet. Der Browser wird beim Beenden auf eine leere Seite gelenkt. Das Beenden hat die gleiche Wirkung wie die Eingabe des Transaktionskürzels `/n` im SAP-Frontend.

  http://www.my-webas.de:8080/sap/bc/bsp/sap/zcode/start.htm?sap-sessioncmd=close

- `sap-exiturl`
  Mittels des Parameters `sap-exiturl` wird zu der angegebenen URL-Adresse verzweigt.

  Kombiniert man diesen Parameter mit dem Parameter `sap-sessioncmd`, lässt sich das Beenden der Applikation komfortabel gestalten. Falls man also explizit die aktuelle BSP-Applikation im Webbrowser beenden und auf eine Zielseite verzweigen möchte, kann der Aufruf wie folgt aussehen:

  http://www.my-webas.de:8080/sap/bc/bsp/sap/zcode/start.htm?sap-sessioncmd=close&sap-exiturl=logout_success.htm

- `sap-theme`
  Dieser Parameter legt das verwendete Thema für die aufgerufene BSP-Seite fest und beeinflusst damit das Gesamterscheinungsbild der Applikation. Ein Thema ist als Sammlung von Ersetzungsdefinitionen für MIME-Objekte zu verstehen.

Ein als Standard definiertes Thema einer BSP-Applikation wird damit übersteuert.

Ein Beispiel:

http://www.my-webas.de:8080/sap/bc/bsp/sap/zcode/start.htm?sap-sessioncmd=open&sap-theme=vip_customer

- `sap-syscmd`
  Mit dem Wert `nocookie` kann gesteuert werden, dass das Session-Cookie nicht beim Client gespeichert, sondern als Teil der URL übergeben wird. Das Session-Cookie wird dann im URL-Mangling-Code verborgen.

  http://www.my-webas.de:8080/sap/bc/bsp/sap/zcode/start.htm?sap-syscmd=nocookie

Anmelde-parameter

Die nachfolgenden Systemparameter dienen der Steuerung der verschiedenen Anmeldeparameter am SAP Web Application Server.

- `sap-client`
  Dieser Parameter definiert den Mandanten, an dem eine Anmeldung am SAP Web Application Server stattfindet. Ein Mandant, der hier angegeben wird, übersteuert den vordefinierten Standardmandanten. Existiert der Mandant nicht im System, erfolgt eine Fehlermeldung.
- `sap-user`
  Die Anmeldung am System erfolgt unter dem angegebenen Namen. Existiert der Benutzer nicht im System, erfolgt eine Fehlermeldung.
- `sap-password`
  Es wird das Passwort zur Anmeldung des Benutzers übergeben. Ist das Passwort nicht korrekt, erfolgt eine Fehlermeldung.

  Das Passwort sollte niemals im Query-String einer URL verwendet werden. Zum einen erfolgt eine Übertragung im Klartext (HTTP) und zum anderen werden diese URL-Strings im Cache zwischengespeichert und wären damit zumindest theoretisch Dritten zugänglich.

- `sap-language`
  Es wird die Anmeldesprache des Systems festgelegt. Auf diese Weise lässt sich z. B. eine andere im System verfügbare Sprachversion der BSP-Applikation laden. Die Standard-Anmeldesprache wird dadurch übersteuert.

Ein Beispiel für die fast vollständige Anmeldung eines englischsprachigen Anwenders könnte z. B. wie folgt aussehen:

http://www.my-webas.de:8080/sap/bc/bsp/sap/zcode/start.htm?sap-sessioncmd=open&sap-theme=vip_customer&sap-client=100&sap-language=en&sap-user=en_george

Da das Passwort nicht Bestandteil des Query-Strings ist, fehlt diese Komponente für die vollständige Authentifizierung. Es erscheint ein entsprechendes Popup-Fenster, um die fehlenden Daten zu ergänzen.

URL-Mangling-Code

Ein weiterer Bestandteil der URL ist der so genannte *URL-Mangling-Code*. Dieser Code wird vom Server generiert und in Klammern mitgeführt. Die codierten Werte beinhalten u.a. verschiedene Anmelde-, Session- und Themeneinstellungen für die aufgerufene BSP-Applikation.

Ein Beispiel mit Mangling-Code:

http://www.my-webas.de:8080/sap(bD1kZQ==)/bc/bsp/sap/zcode/start.htm

3.3.3 Eventhandler-gesteuerte Verarbeitung

Der grundsätzliche Verarbeitungsablauf einer BSP-Applikation folgt einem fest definierten Schema. Die generelle Ablauflogik ist dabei immer dieselbe, unabhängig davon, ob es sich z.B. um eine einfache Adressverwaltung oder eine komplexe Auftragsverwaltung mit integrierter Verfügbarkeitsprüfung über ein Backend-System handelt. Eine BSP-Applikation besteht in der Regel aus mehreren BSP-Seiten (Seiten mit Ablauflogik). Der Benutzer startet die Applikation über eine Einstiegsseite und navigiert dann durch die Applikation zu verschiedenen BSP-Seiten. Irgendwann verlässt er die Applikation schließlich wieder. Die einzelnen Schritte dieses Verarbeitungsablaufs (siehe Abbildung 3.14) sind:

- Start der BSP-Applikation
- Generierung und Darstellung der angeforderten BSP-Seite
- ggf. Reaktion auf Benutzereingaben und Navigation
- Beenden der BSP-Applikation[10]

Diese verschiedenen Schritte werden im Folgenden eingehend behandelt. Dazu müssen jedoch zunächst zwei wichtige Aspekte näher beleuchtet werden – die *Eventhandler* und das verwendete *Zustandsmodell* der BSP-Applikation bzw. BSP-Seite. Diese beiden Punkte haben einen wesentlichen Einfluss auf den Verarbeitungsablauf der Applikation. Die Eventhandler wurden bereits in Abschnitt 3.3.1 kurz vorgestellt.

10 Das Beenden einer BSP-Applikation ist nur im Stateful-Zustandsmodell von Interesse. Dieses Zustandsmodell wird weiter unten erläutert.

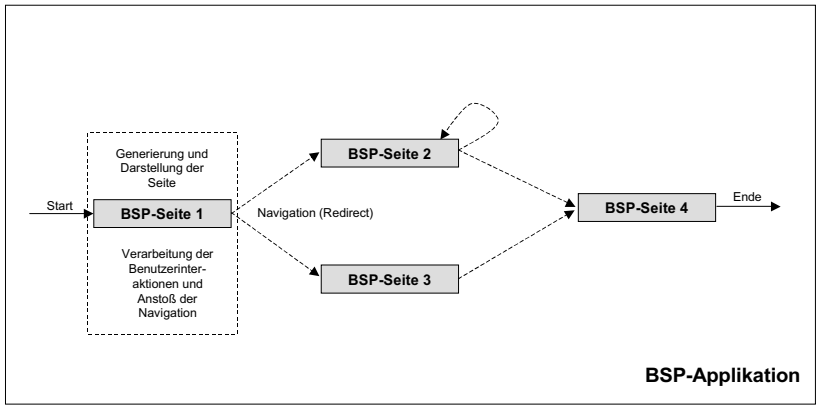

Abbildung 3.14 Der grundlegende Verarbeitungsablauf und die Navigation in BSP-Applikationen

Diese Handler stehen zu bestimmten Zeitpunkten der Verarbeitung einer BSP-Seite zur Verfügung. Während dieser Verarbeitungszeitpunkte kann eigene Programmlogik zur Ausführung gebracht werden, um ganz bestimmte Aufgaben innerhalb der Verarbeitungslogik einer Seite wahrzunehmen. Der Verarbeitungsablauf, insbesondere das Durchlaufen der verschiedenen Eventhandler, wird zudem dadurch beeinflusst, welches Zustandsmodell für die jeweilige Applikation zurzeit aktiv ist. Die im SAP Web Application Server unterstützten Zustandsmodelle sind Gegenstand der nachfolgenden Betrachtungen.

Das Zustandsmodell für BSP-Applikationen

Stateful und stateless Applikationen

Für den Einsatz in BSP-Applikationen stehen zwei Zustandsmodelle zur Verfügung: stateful (verbindungsorientiert) und stateless (zustandslos). Diese Funktionalität wird durch den ICF zur Verfügung gestellt, der in der Serverrolle beide Betriebsarten unterstützt. Eine laufende BSP-Applikation wird als *BSP-Session* bezeichnet. Wie Beginn und Ende einer Session durch den Benutzer von außen beeinflusst werden kann, wurde bereits eingehend diskutiert (siehe Abschnitt 3.3.2). Das Ende einer BSP-Session kann weiterhin durch die Applikation selbst und das Schließen des Webbrowsers initiiert werden. Wichtig hierbei ist die Trennung zwischen BSP- und Browsersession. Eine Browsersession kann nur durch das Schließen des Browsers beendet werden.

Stateful

Eine stateful BSP-Applikation wird – wie bei einer klassische SAP-Transaktion mit SAP GUI-Bildwechseln – über alle Benutzerinteraktionen hinweg in einem einzigen Kontext (Rollbereich) ausgeführt. Der Applikationskontext wird also über den Response hinaus gehalten. Bei der Wei-

terführung der Anwendung wird der entsprechende Kontext in den Workprozess hineingerollt. Damit bleiben Daten, die vom Benutzer im Lauf der Ausführung der Applikation eingegeben oder die durch die Applikation selbst ermittelt wurden, über die gesamte Ausführungsdauer der Session hinweg erhalten. Die Datenhaltung erfolgt bei stateful Applikationen über die Applikationsklasse. Abbildung 3.15 veranschaulicht das Stateful-Modell. Hierbei wird für jede Session eines Webbrowsers eine eigene Session im SAP Web Application Server bereitgestellt. Diese Session beinhaltet den Applikationskontext und steht für mehrere Request-Response-Zyklen zur Verfügung. Die dunklen, unterschiedlich langen Blöcke in der Grafik stehen für Benutzeraktivitäten. Zu diesem Zeitpunkt werden die Ressourcen des Servers also auch tatsächlich beansprucht.

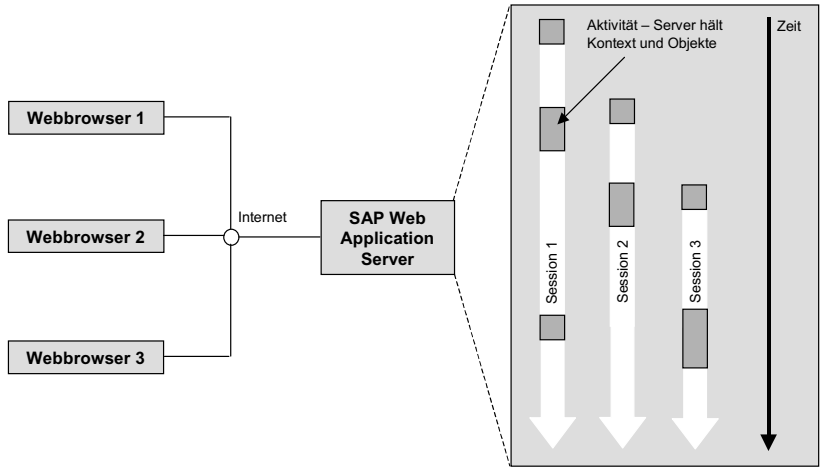

Abbildung 3.15 Das Stateful-Modell im SAP Web Application Server

Session-Cookies

Problematisch bei der Realisierung dieses Verhaltens ist die Zustandslosigkeit des HTTP-Protokolls. Es existiert kein impliziter Mechanismus, um unabhängige Requests einer gemeinsamen logischen Sitzung, d.h. einem Kontext, zuzuordnen. Die BSP-Laufzeitumgebung löst dieses Problem durch den Einsatz von Session-Cookies. Hierzu wird eine mehrstellige Session-ID generiert und jeder Request um diesen eindeutigen Stempel erweitert. So lassen sich die Requests dann als Teil einer bestimmten Session identifizieren. Der Name des auf Clientseite abgelegten Session-Cookies lautet `sap-contextid` und ist bis zum Ende einer Sitzung gültig (die Browser-Session-ID entspricht dabei der BSP-Session-ID). Die Zuordnung erfolgt dabei über die URL der BSP-Applikation. Dies bedeutet, dass eine BSP-Applikation innerhalb eines Browsers zu einem Zeitpunkt nur einmal ausgeführt werden kann. Eine andere BSP-Applikation erhält ein

eigenes Session-Cookie und lässt sich parallel im gleichen Browser betreiben. Die gleiche BSP-Applikation kann von mehreren Benutzern und Browsern beliebig häufig ausgeführt werden.

Das Stateful-Modell besitzt eine Reihe von Vor- und Nachteilen, die es zu berücksichtigen gilt:

Weniger Datenbankzugriffe

Generell gestaltet sich die Programmierung von stateful BSP-Applikationen sehr komfortabel. Wurden Daten einmal aufwändig beschafft, lassen sie sich in den Attributen der Applikationsklasse zwischenspeichern. Auf einer Folgeseite kann dann einfach auf diese Daten erneut zugegriffen werden. Aufwändige Datenbankzugriffe lassen sich damit auf ein Minimum reduzieren. Dadurch entfällt das Nachlesen vieler Daten, was zu erheblichen Performanzverbesserungen auf der Serverseite führt. Des Weiteren wird die Netzwerklast durch weniger Anfragen an Backend-Systeme minimiert.

Zu berücksichtigen ist allerdings die Tatsache, dass sich der Zustand innerhalb der Anwendung sehr leicht von dem Zustand unterscheiden kann, den ein Benutzer aufgrund seiner Benutzeroberfläche annimmt. Einfachstes Beispiel hierfür ist die Verwendung des Back-Buttons im Browser. Diese clientseitigen Aktionen werden dem Server nicht notwendigerweise mitgeteilt. Hier ist also Sorge dafür zu tragen, dass mögliche Inkonsistenzen abgefangen und Client und Server wieder »synchronisiert« werden.

Speicherressourcen

Der Vorteil der vereinfachten Programmierung resultiert allerdings in der Tatsache, dass für jede Session der jeweilige Kontext gesichert werden muss. Da die Anzahl der Sessions mindestens gleich der Anzahl der Benutzer ist (analog zum SAP GUI), wird grundsätzlich eine viel größere Last auf dem SAP Web Application Server erzeugt als bei dem Stateless-Modell. Dies stellt erhöhte Anforderungen an die Ressource Speicher, um eine Vielzahl von Sitzungen parallel ausführen zu können. Ist der verfügbare Speicherplatz erschöpft, werden keine weiteren Benutzer akzeptiert. Sie werden vom System abgewiesen! Beendet der Benutzer die BSP-Applikation nicht explizit, werden zudem unnötig lange Ressourcen im System blockiert. Navigiert der Benutzer z. B. einfach auf eine andere Seite, bleibt die Sitzung im SAP Web Application Server bestehen. In diesen Fällen meldet sich der Webbrowser nicht automatisch am System ab. Der Kontext bleibt damit erhalten und wird erst nach einer bestimmten Zeit freigegeben.

Die Zeitspanne für ein Auto-Log-Off des SAP Web Application Servers wird durch einen Timeout-Parameter im Instanzprofil festgelegt.

Stateless Im Gegensatz zum Stateful-Modell werden im Stateless-Modell Ressourcen auf dem SAP Web Application Server nicht unnötig lange blockiert. Nach Abarbeitung eines Requests werden alle belegten Ressourcen (Applikationskontext) sofort wieder freigegeben. Für jeden Request wird somit ein neuer Applikationskontext (Rollbereich) erzeugt und nach dem Response verworfen.[11] Ressourcen werden also nur unmittelbar während der Bearbeitung eines Requests beansprucht. Dieses Modell ist somit ideal für die Implementierung von Webapplikationen mit einer Vielzahl paralleler Zugriffe, da eine gute Skalierung des SAP Web Application Servers erzielt wird. Diesen Sachverhalt veranschaulicht Abbildung 3.16. Hier ist zu sehen, dass für jeden Zugriff eines Webbrowsers die Ressourcen zur Bearbeitung des Requests nur kurzzeitig auf dem Server beansprucht werden.

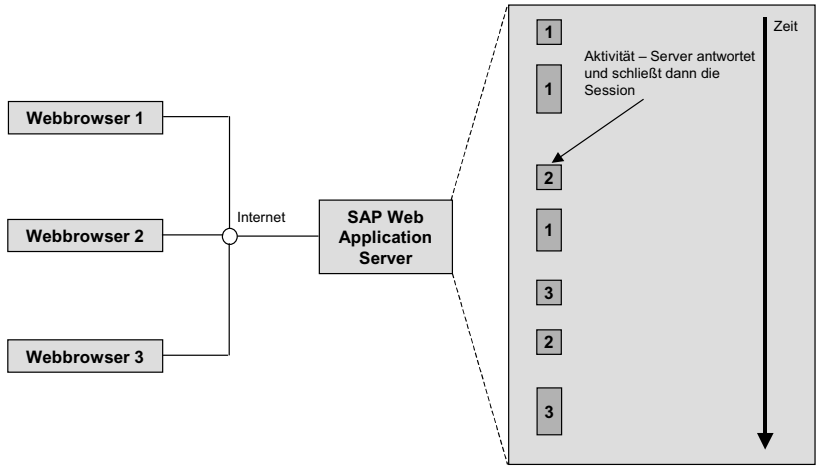

Abbildung 3.16 Das Stateless-Modell im SAP Web Application Server

Der Nachteil besteht darin, dass durch das Freigeben des Anwendungskontextes Daten mehrfach gelesen und aufbereitet werden müssen. Der Speichervorteil wird also bedauerlicherweise durch Laufzeiteinbußen kompensiert. Durch den Einsatz verschiedener Techniken lassen sich jedoch auch im Stateless-Fall Daten über Requests hinaus retten. Dazu zählen:

11 Im Stateless-Fall wird die allererste Session-ID verwendet, um die zugehörige Browsersession zu identifizieren.

- **versteckte Formularfelder**
  Diese versteckten Felder werden beim Senden eines Formulars, für den Anwender unsichtbar, mit übertragen. Es handelt sich dabei um Input-Felder mit dem Attribut `type=hidden`.
- **clientseitige Cookies**
  Hier werden die Daten auf Clientseite in kleinen Textdateien zwischengespeichert; es sind gewisse Einschränkungen zu berücksichtigen (siehe Abschnitt 3.2.5).
- **serverseitige Cookies**
  Hierbei liegen die Daten nicht auf Clientseite, sondern auf dem Server. Im Gegensatz zu clientseitigen Cookies existieren keine Größenbeschränkungen (siehe Abschnitt 3.2.5).
- **DB-Tabellen**
  Als weitere Möglichkeit bietet es sich an, die Daten in für die zu diesem Zweck erstellten DB-Tabellen abzulegen. Dies erlaubt eine eigene Typisierung der Tabellen und bringt beim Zugriff Performanzvorteile. Der Nachteil ist ein höherer Programmieraufwand.

Nachdem die beiden Zustandsmodelle vorgestellt wurden, bleibt zu klären, wie der jeweils gewünschte Modus eingestellt werden kann. Eine Einstellung kann zur Entwicklungszeit oder zur Laufzeit erfolgen. Wird nichts eingestellt, arbeitet die BSP-Applikation grundsätzlich stateless (Standardeinstellung).

Zur Entwicklungszeit kann der gewünschte Modus auf der Registerkarte **Eigenschaften** einer BSP-Applikation eingestellt werden (siehe Abbildung 3.17).

Aber auch einzelne BSP-Seiten einer BSP-Applikation können »stateful« oder »stateless« gesetzt werden. Diesen Sachverhalt veranschaulicht Abbildung 3.18. Wie dort ebenfalls zu sehen ist, können verschiedene Einstellungen bezüglich der Lebensdauer (im Stateful-Modus) vorgenommen werden. Dazu zählen:

- **bis zum Seitenwechsel**
  Die Seite wird zerstört, wenn eine andere Seite verwendet wird.
- **für die Dauer des Requests**
  Die Seite wird nach jedem einzelnen Request zerstört, d.h., sie ist jeweils nur für die Dauer eines Requests vorhanden.
- **für die Dauer der Session**
  Die Seite wird am Ende der Session zerstört.

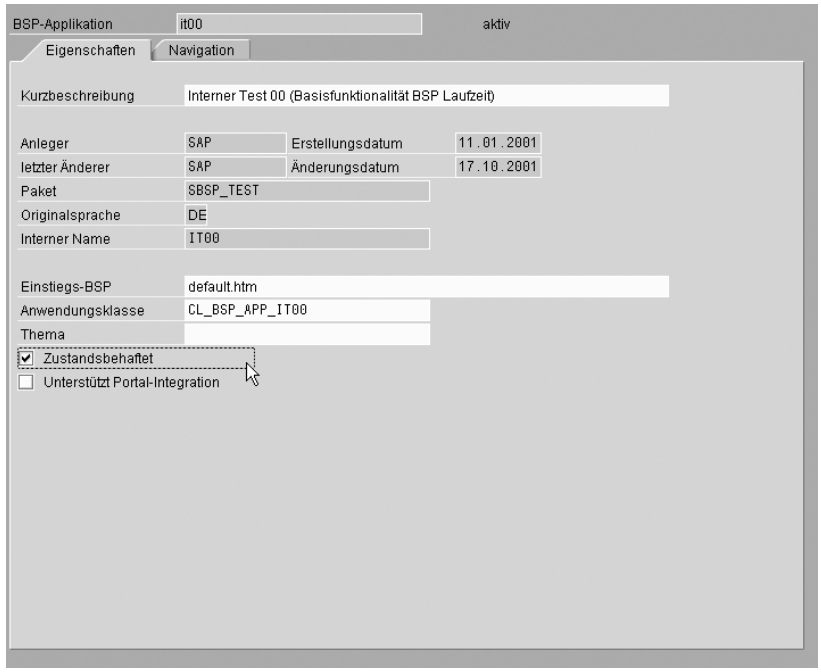

Abbildung 3.17 Eine BSP-Applikation auf »stateful« setzen

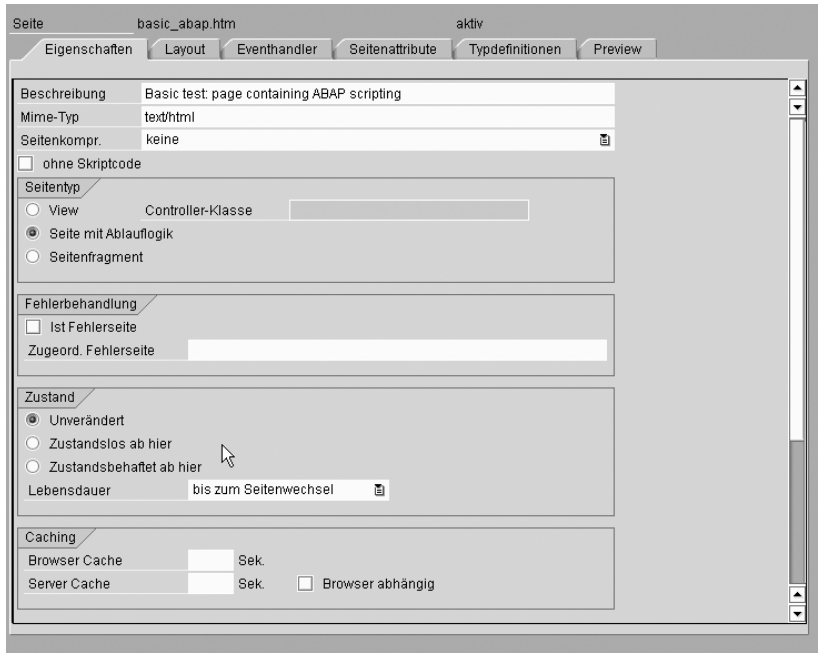

Abbildung 3.18 Das Setzen von »stateful« bzw.»stateless« in BSP-Seiten

Als zweite Möglichkeit kann zur Laufzeit (dynamisch) zwischen stateful und stateless gewechselt werden. Hierzu steht das Laufzeitobjekt `runtime`, das sich auf das Interface `IF_BSP_RUNTIME` bezieht, zur Verfügung. Die Einstellung erfolgt über das Setzen des Wertes des Attributs `KEEP_CONTEXT`. Dieses Attribut kann die Werte »0« und »1« annehmen. Nachfolgendes Beispiel setzt den stateful- bzw. stateless-Modus:

```
runtime->keep_context = 1.    " setze Anwendung auf sta-
                                teful
runtime->keep_context = 0.    " setze Anwendung auf sta-
                                teless
```

Diese Einstellungen übersteuern eventuell vorgenommene Definitionen aus der Entwicklungsumgebung.

Nachdem das Stateful- bzw. Stateless-Modell eingeführt wurde, werden nun die verfügbaren Eventhandler vorgestellt.

Die Eventhandler

Die Eventhandler repräsentieren das Ereigniskonzept der BSP-Seiten. Es existieren eine Reihe vordefinierter Handler, die beim Verarbeiten einer Seite in einer vorgegebenen Reihenfolge durchlaufen werden. Dies sind im Einzelnen:

- `onCreate`
  Dieser Handler wird beim ersten Aufruf einer Seite aufgerufen und dient der einmaligen Initialisierung von auf der Seite benötigten Objekten und Daten. Der Handler wird immer dann aufgerufen, wenn eine BSP-Page-Klasse erzeugt wird. Der Handler wird im Stateful-Modus bei der Erzeugung der Seite genau einmal aufgerufen. Im Stateless-Modus wird das Seitenobjekt jedes Mal neu initialisiert, d.h., der Handler wird immer wieder neu aufgerufen. Arbeitet man im Stateful-Modus ohne explizite Navigation und läuft einfach ein weiteres Mal durch die Seite, bleibt die Seiteninstanz bestehen. Der Handler wird dann nicht noch einmal durchlaufen.
- `onRequest`
  Dieser Handler wird bei jedem Zugriff (Request) auf eine Seite aufgerufen. Er dient der Wiederherstellung interner Datenstrukturen aus einem Request.

- `onInitialization`
  Dieser Handler dient der Datenbeschaffung. Idealerweise werden diese in Seitenattributen gespeichert und stehen dann auch im Layout zur Verfügung. Des Weiteren können beliebige Programme aufgerufen werden. Dieser Handler wird nach dem `onRequest`-Handler angestoßen.
- `onLayout`
  Hierbei handelt es sich um einen versteckten Eventhandler. Es kann kein Coding hinzugefügt werden. Das Layout der Seite wird gerendert und der HTTP-Datenstrom für den Response erzeugt.
- `onManipulation`
  Dieser Eventhandler erlaubt eine nachträgliche Manipulation des HTTP-Datenstroms. Er wird verarbeitet, nachdem die Layout-Elemente der Seite erzeugt wurden. Dieser Handler wird selten eingesetzt.
- `onInputProcessing`
  Die Verwaltung von Benutzereingaben, Eingabeprüfungen und die Navigation – auf die gleiche Seite oder zu Folgeseiten – ist Aufgabe dieses Handlers. Damit dieser Handler angestoßen wird, sind bestimmte Voraussetzungen zu erfüllen.
- `onDestroy`
  Dieser Handler wird zukünftig unmittelbar vor dem Löschen einer Seiteninstanz aufgerufen. Hier lassen sich dann abschließende Aktionen für eine Seite durchführen. Er ist das Gegenstück zum `onCreate`-Handler. Im Stateless-Modus werden `onCreate` und `onDestroy` für jeden Request-Response-Zyklus durchlaufen. Im Stateful-Modus wird dieser Handler nicht für jeden Zyklus aufgerufen, sondern nur wenn in den Stateless-Modus gewechselt wird.
  Dieser Eventhandler ist für eine zukünftige Verwendung vorgesehen, d.h., er wird im Moment bei der Verarbeitung einer Seite nicht aufgerufen.

Innerhalb jedes Eventhandlers stehen bestimmte globale Laufzeitobjekte zur Verfügung. Diese Objekte erlauben z. B. den Zugriff auf das `request`-Objekt, das `response`-Objekt oder realisieren eine Navigation zwischen BSP-Seiten über das `navigation`-Objekt. Einen Überblick über diese verfügbaren Objekte und ihre Bedeutung finden Sie in Tabelle 3.5.

Objekt	Bedeutung	Interface/Klasse/Typ
runtime	BSP-Laufzeitinformationen	IF_BSP_RUNTIME
application	Attribute und Methoden der Applikationsklasse	eigene Klasse oder Ableitung von CL_BSP_APPLICATION
page_context	Seitenkontextobjekt[1]	IF_BSP_PAGE_CONTEXT
page	BSP-Seiteninformationen	IF_BSP_PAGE
request	Zugriff auf das request-Objekt	IF_HTTP_REQUEST
response	Zugriff auf das response-Objekt	IF_HTTP_RESPONSE
navigation	Datenübergabe und Navigation	IF_BSP_NAVIGATION
event_id	Benutzerinteraktion	STRING
messages	Behandlung von Fehlermeldungen[2]	CL_BSP_MESSAGES

[1] Das page_context-Objekt stellt eine Verschalung einer BSP dar und spielt nur im Zusammenhang mit den BSP-Extensions eine Rolle.
[2] Das message-Objekt ist ein Attribut des page-Objekts.

Tabelle 3.5 Globale Laufzeitobjekte einer BSP-Applikation

Welche dieser Objekte in welchen Eventhandlern zur Verfügung stehen, wird in Tabelle 3.6 zusammengefasst.

Eventhandler	verfügbare globale Objekte
onCreate	runtime, application, page_context, page (messages)
onRequest	runtime, application, page_context, page (messages), request, navigation, event_id
onInitialization	runtime, application, page_context, page (messages), request, response, navigation,
onManipulation	runtime, application, page_context, page (messages), request, response
onInputProcessing	runtime, application, page_context, page (messages), request, navigation, event_id
onDestroy (derzeit noch nicht verfügbar)	runtime, application, page_context, page (messages)

Tabelle 3.6 Globale Laufzeitobjekte in Eventhandlern

Der Verarbeitungsablauf

Start der BSP-Applikation

Zum Start der Applikation ist die entsprechende URL in die Adresszeile des Webbrowsers einzugeben. Diese Adresse identifiziert die zu star-

tende BSP-Applikation. Das Startverhalten einer Anwendung (BSP-Session) lässt sich – wie bereits gezeigt – über verschiedene URL-Parameter konfigurieren. Das Eingeben der URL resultiert in einem HTTP-GET-Request, der an die BSP-Laufzeit gesendet wird. Die BSP-Laufzeit ermittelt daraufhin die passende BSP-Applikation und die geforderte BSP-Seite. Je nach Einstellung muss hierzu eine Anmeldung am SAP Web Application Server erfolgen (siehe Abschnitt 2.5.1).

Generierung und Darstellung

Innerhalb der BSP-Laufzeitumgebung (BSP-Engine) wird die angeforderte BSP-Seite verarbeitet, ihre Komponenten durchlaufen und die entsprechenden Verarbeitungsschritte je nach programmierter Logik angestoßen. Das Ergebnis ist eine Webseite, die als `response`-Objekt an den Aufrufer zurückgeschickt wird. Die Generierung erfolgt in unterschiedlichen Phasen. Dabei werden Ereignisse (Events) in einer bestimmten Reihenfolge durchlaufen, auf die der Programmierer bedingt Einfluss nehmen kann. Diese Ereignisse werden durch Eventhandler repräsentiert, die unterschiedliche Aufgaben wahrnehmen. Der Verarbeitungsablauf einer BSP-Applikation – insbesondere das Durchlaufen der verschiedenen Eventhandler – wird entscheidend dadurch beeinflusst, welches Zustandsmodell zurzeit aktiv ist. Einen Überblick über den schematischen Ablauf (sowohl für stateful als auch für stateless) erhalten Sie in Abbildung 3.19.

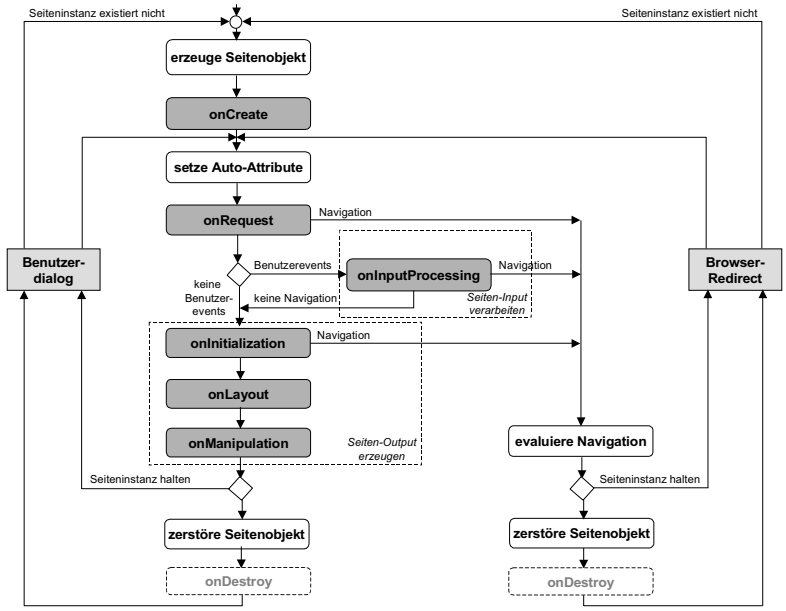

Abbildung 3.19 Der grundlegende Verarbeitungsablauf einer BSP-Seite

Verarbeitungsablauf im Stateless-Modell

Beim Aufruf der Seite wird von der BSP-Laufzeit zunächst ermittelt, ob es bereits ein Seitenobjekt für diese BSP-Seite gibt. Ist dies nicht der Fall, wird der `onCreate`-Eventhandler durchlaufen. Dadurch wird das Seitenobjekt bzw. eine Instanz davon erzeugt. Aus Entwicklersicht kann dieser Handler genutzt werden, um Daten zu initialisieren oder benötigte Objekte zu erzeugen.

Im nächsten Schritt werden eventuell vorhandene Auto-Seitenattribute übernommen.

Anschließend wird der `onRequest`-Eventhandler aufgerufen. Dieser Handler wird bei jedem Request auf eine Seite aufgerufen. Er dient der Wiederherstellung von internen Datenstrukturen aus dem Request.

Jetzt muss unterschieden werden, ob eine Benutzerinteraktion stattgefunden hat. Ist dies der Fall, wird der `onInputProcessing`-Eventhandler aufgerufen; andernfalls der `onInitialization`-Eventhandler. Zunächst wird der Fall ohne Benutzerinteraktion betrachtet.

Der `onInitialization`-Eventhandler dient zur Beschaffung notwendiger Daten. Diese Daten können aus unterschiedlichsten Quellen stammen (z. B. aus DB-Tabellen, via Funktionsbausteinen, BAPIs usw.).

Sind die notwendigen Daten beschafft, wird im Anschluss daran die `onLayout`-Phase prozessiert. Hier wird das Design und die Präsentation der Seite festgelegt, die der Anwender sieht. Es geht also um die Aufbereitung und Gestaltung der angeforderten Seiten. Eine Seite besteht aus statischen (z. B. HTML) und dynamischen (serverseitiges Scripting) Anteilen. Während das clientseitige Scripting (z. B. JavaScript) ungefiltert an den Client zurückgeschickt wird, wird das serverseitige Scripting (ABAP) verarbeitet und in ein für den Browser verständliches Coding transformiert. Das Ergebnis ist ein serialisierter HTTP-Datenstrom.

Eine Möglichkeit zur Manipulation des HTTP-Datenstroms bietet der `onManipulation`-Eventhandler, der im Anschluss an den `onLayout`-Eventhandler aufgerufen wird.

Im Stateless-Fall wird nun das erzeugte Seitenobjekt wieder zerstört. Um abschließende Arbeiten durchführen zu können (z. B. das Sichern von Daten in einem serverseitigen Cookie), steht zukünftig der `onDestroy`-Eventhandler zur Verfügung. Da dieser Eventhandler zur Zeit nicht durchlaufen wird, ist er in der Grafik entsprechend gekennzeichnet (gestrichelter Kasten).

Anschließend wird dieser HTTP-Datenstrom an den Aufrufer gesendet. Die angeforderte Webseite erscheint daraufhin im Browser des Anwenders.

Reaktion auf Benutzereingaben und Navigation

Mit der Darstellung der Seite ist es in der Regel jedoch nicht getan. Der Anwender soll Daten eingeben, Selektionen ausführen oder einfach zu einer anderen Seite navigieren können. Damit dies möglich wird, sind die entsprechenden Benutzerinteraktionen (z. B. per Maus und Tastatureingaben) entgegenzunehmen und entsprechend weiterzuverarbeiten. Das Ergebnis kann je nach programmierter Logik z. B. eine neue Webseite, die aktualisierte Ausgangsseite oder auch eine Fehlermeldung sein. Diese Benutzereingaben werden vom `onInputProcessing`-Eventhandler verarbeitet. Ist eine derartige Benutzerinteraktion erfolgt, wird nach `onRequest` statt `onInitialization` der `onInputProcessing`-Eventhandler aufgerufen.

Der `onInputProcessing`-Eventhandler dient der Prüfung fehlerhafter Eingaben, dem Weiterleiten von Attributen und der Bestimmung von Folgeseiten für die weitere Navigation. Wurde keine Folgeseite festgelegt, wird mit dem `onInitialization`-Eventhandler der Seite fortgefahren (Navigation innerhalb der BSP). Wurde dagegen eine Folgeseite definiert, erfolgt die Navigation zu dieser neuen BSP-Seite. Die Folgeseite kann dabei aus der Navigationsstruktur ermittelt oder im Programmcode vorgegeben werden.[12] Anschließend wird die angeforderte Seite nach dem bereits bekannten Verarbeitungsablauf erstellt und im Browser des Anwenders ausgegeben. Nun kann wieder eine Benutzeraktion stattfinden und der Verarbeitungsablauf beginnt erneut. Bei jeder Navigation wird das zugrunde liegende Seitenobjekt wieder zerstört. Damit der `onInputProcessing`-Eventhandler angestoßen wird, sind bei der Entwicklung eine Reihe von Vorgaben einzuhalten. Diese sind Gegenstand von Abschnitt 5.5.

Verarbeitungsablauf im Stateful-Modell

Der Verarbeitungsablauf ähnelt sehr stark dem Stateless-Fall, sodass hier nur die Unterschiede aufgeführt werden. Wichtigstes Merkmal ist der Erhalt des Seitenobjekts[13] über den Zeitraum einer Session. Beim erstmaligen Aufruf einer BSP wird der `onCreate`-Eventhandler genau ein Mal aufgerufen. Erfolgt jetzt eine Navigation innerhalb der BSP, bleibt das Seitenobjekt bestehen und `onCreate` wird nicht noch einmal aufgerufen.

12 Innerhalb dieses Eventhandlers besteht auch die Möglichkeit, aufgrund der Benutzangaben eine Folgeseite dynamisch zu bestimmen. Weitere Informationen dazu finden Sie in Kapitel 5.

13 Die Lebensdauer eines Seitenobjekts kann auf Seitenebene, Request-Ebene oder für die gesamte Session festgelegt werden.

Der `onDestroy`-Eventhandler wird nur dann aufgerufen, falls zwischenzeitlich in den Stateless-Modus gewechselt wird. Die Verarbeitung erfolgt ansonsten wie im Stateless-Fall.

Beenden

Eine stateful BSP-Applikation kann auf zweierlei Weise beendet werden. Die erste Möglichkeit besteht darin, spezielle Systemparameter an die URL anzuhängen. Der Parameter `sap-sessioncmd=close` beendet dann die Applikation. Diese speziellen URL-Systemparameter wurden in Abschnitt 3.3.2 beschrieben. Bei der zweiten Möglichkeit beendet ein Timeout-Mechanismus die Session. Ruht eine BSP-Applikation, d.h. werden keine Aktionen mehr ausgeführt, die zu einem weiteren Request-Response-Zyklus führen, greift nach einer im Instanzprofil definierten Zeit ein Timeout. Der Timeout greift auch dann, wenn einfach der Webbrowser geschlossen wird, ohne dass die BSP-Applikation zuvor explizit beendet wurde. Vom Schließen des Webbrowsers bekommt der SAP Web Application Server nämlich nichts mit. Dies ist bei der Implementierung zu berücksichtigen.

Aus einer BSP-Seite wird zur Laufzeit eine lokale ABAP-Klasse erzeugt. Das Seitenlayout und die verschiedenen Eventhandler sind Methoden dieser Klasse. Das in einer Seite enthaltende serverseitige Scripting wird übersetzt in Code der generierten Layout-Methode. Auch die Seitenattribute werden in Methodenparameter der generierten Klasse übersetzt.

3.3.4 Model-View-Controller-Design-Pattern (MVC)

Design-Pattern

Unter einem *Design-Pattern* versteht man in der Softwareentwicklung ein – im weitesten Sinne – geschriebenes Dokument, das eine allgemeine Lösung für ein Problem beschreibt, das in allgemeiner Form immer wieder in unterschiedlichsten Projekten auftaucht. Pattern beschreiben das Problem, die Lösung und weitere Faktoren formal. In der objektorientierten Entwicklung kann ein solches Pattern Beschreibungen von Objekten und Klassen einschließlich ihrer einzelnen Komponenten und Abhängigkeiten beinhalten. Eine Sammlung dieser Pattern repräsentiert ein *Pattern-Framework*.

Das MVC-Pattern beschreibt die Methodologie zur effizienten Verbindung des User-Interfaces mit dem zugrunde liegenden Datenmodell. Es ist in der Programmierung in Sprachen wie Java, Smalltalk, C und C++ weit verbreitet.[14]

14 Dieser Umstand erst berechtigt übrigens das MVC-Modell, sich *Design-Pattern* nennen zu dürfen. Eine Grundvoraussetzung ist der Verbreitungsgrad.

Trennung von Ablauf, Anwendung und Oberfläche

Das MVC-Pattern beinhaltet eine klare Trennung zwischen Ablaufsteuerung, Anwendungslogik (Datenmodell) und Präsentationslogik. Die formale Trennung dieser drei Bereiche wird über die drei Objekte Model, View und Controller realisiert. So können komplexe BSP-Applikationen in logische Einheiten getrennt werden. Das hat diverse Vorteile. Änderungen in der Benutzeroberfläche haben keine Auswirkung auf die Anwendungslogik. Umgekehrt können aber Daten mehrfach in unterschiedlichen Darstellungsformen gleichzeitig präsentiert werden. Durch geschickte Update-Mechanismen führt eine Änderung an den Daten zu einer Aktualisierung in allen Darstellungen.

Zusammenspiel der MVC-Komponenten

Die Komponenten wurden bereits in Abschnitt 3.3.1 kurz vorgestellt, werden hier zum besseren Verständnis aber noch einmal näher erläutert.

- **Model**

  Das Model repräsentiert die logische Struktur der Daten, die der Applikation zugrunde liegen. Es bietet z. B. Methoden einschließlich Backend-Services zur Datenbeschaffung und -verarbeitung an. Diese Komponente ist normalerweise fast ausschließlich für die Umsetzung der Anwendungslogik (Business-Logik) zuständig und beinhaltet daher auch keinerlei Informationen über das User Interface.

  Zu einem Model kann es verschiedene Views geben, die durch entsprechende View-Seiten realisiert sind.

- **View**

  Views bestehen in normalen Applikationen aus Sammlungen von Klassen zur Repräsentation der grafischen Elemente der Benutzeroberfläche – wie z. B. Drucktasten, Menüs und Dialogfelder – und sind für die Visualisierung von Oberflächenelementen zuständig.

  Im SAP Web Application Server werden Views als konkrete Ausprägung von BSP-Seiten realisiert. Sie beinhalten HTML-Coding und ABAP zum Rendern der Benutzeroberfläche.

  Zur Visualisierung des Zustands stellt ein View entweder Anfragen an das Model oder das Model informiert den View über mögliche Zustandsänderungen. Der beim Client zur Anzeige gebrachte View leitet Aktionen des Benutzers wie z. B. den Klick auf einen Submit-Button an einen zugeordneten Controller weiter.

  Views besitzen weder Eventhandler noch Auto-Seitenattribute. Seitenattribute werden durch den Controller gefüllt.

- **Controller**

  Controller sind die Klassen, die die Verbindung zwischen dem Model und dem View realisieren. Sie implementieren die Entscheidungsprozesse zur Reaktion auf Benutzereingaben und steuern den Ablauf. Eingabedaten des Users, die über den View entgegengenommen werden, werden hier an das Model weitergeleitet und lösen dort Änderungen an den Anwendungsdaten durch entsprechende Methodenaufrufe aus. Views werden zur Ausführung bzw. zur Änderung ihres Zustands gebracht.

  Folgende Punkte sind bei der Verwendung von Contollern zu berücksichtigen:

  - Controller werden entweder von der Basisklasse `CL_BSP_CONTROLLER` (siehe Anhang) oder von anderen Controllern abgeleitet. Dies erfolgt am einfachsten, wie in Abbildung 3.20 gezeigt, über die Vorwärtsnavigation beim Anlegen eines Views.

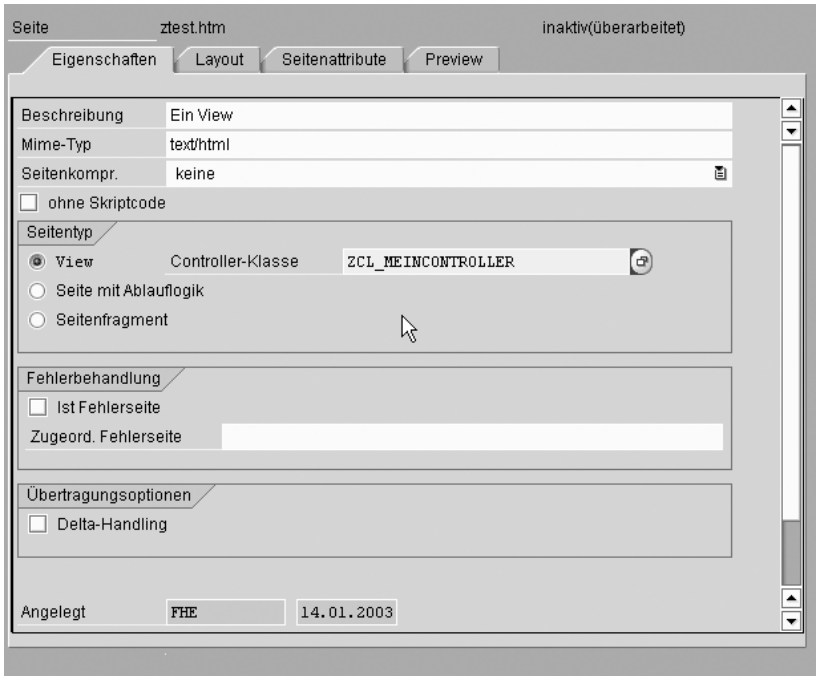

Abbildung 3.20 Das Anlegen eines Controllers per Vorwärtsnavigation

  - Controller können nur Views der eigenen BSP-Applikation steuern. Sie können allerdings die Kontrolle an Controller anderer Applikationen übergeben.

- Die Lebensdauer eines Controllers ist standardmäßig auf einen Aufruf beschränkt. Durch Verwendung der ID wird die Lebensdauer aus den Einstellungen der Controller-Eigenschaften gezogen.
- Redirects sind auch mit Controllern möglich.
- Controller sind von außen per URL erreichbar. Sie besitzen normalerweise die Endung *.do*. Ein typischer Aufruf könnte also folgendermaßen aussehen:
  http://www.my-webas.de/sap/bc/bsp/sap/zcode/start.do?sap-client=100

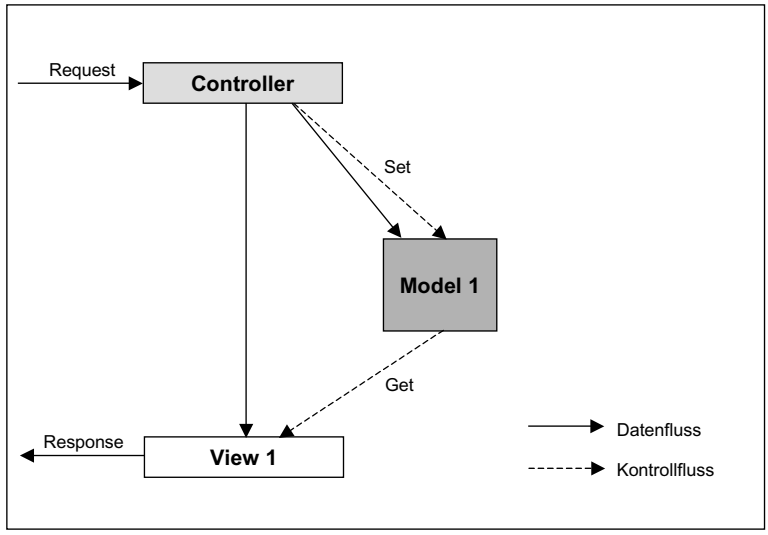

Abbildung 3.21 Schematisches Zusammenspiel der MVC-Komponenten

Abbildung 3.21 zeigt schematisch den Zusammenhang zwischen den Komponenten Model, View und Controller.

Implementierungsbeispiele

Einfache Implementierung

In Abbildung 3.14 wurde die Navigation in einer beispielhaften BSP-Applikation mit Ablauflogik dargestellt. In einer BSP-Anwendung unter Verwendung des MVC-Design-Pattern würde analog die Darstellung in Abbildung 3.22 gelten. Jede der BSP-Seiten hat alle drei Komponenten implementiert. Die Views werden jeweils direkt durch ihre zugeordneten Controller aufgerufen. Zwischen den BSP-Seiten wird per Redirect navigiert. Es wäre durchaus auch möglich, jedem Controller mehrere Models oder auch mehrere Views zuzuweisen. Hier kann man auch bereits sehen, dass der »Mischbetrieb« mit normalen BSP-Seiten durchaus möglich ist.

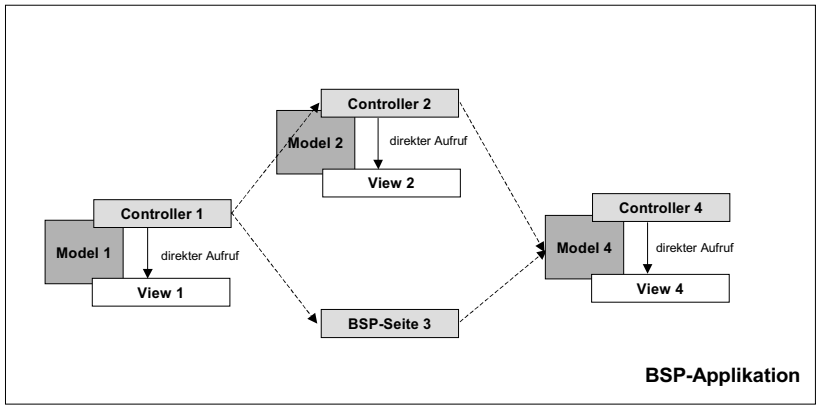

Abbildung 3.22 Einfache BSP-Applikation mit MVC

Mehrere Views

Ein Controller kann mehrere Views nacheinander steuern. Bei entsprechender Ablaufsteuerung kann er diese auch wahlweise aufrufen. Abbildung 3.23 veranschaulicht diesen Sachverhalt.

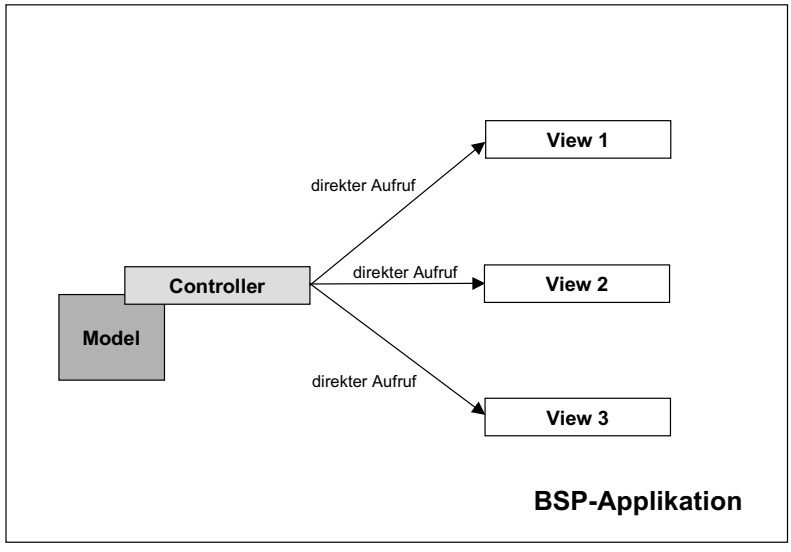

Abbildung 3.23 Mehrere Views pro Controller

Flexibilität durch zentrale Verteilung

Die Flexibilität zeigt sich insbesondere bei der Kombination der beiden vorangegangenen Möglichkeiten. Hierbei steuert ein übergeordneter zentraler Controller die Verteilung (dieser Hauptcontroller benötigt keinen eigenen View). Diese Form der Implementierung veranschaulicht Abbildung 3.24.

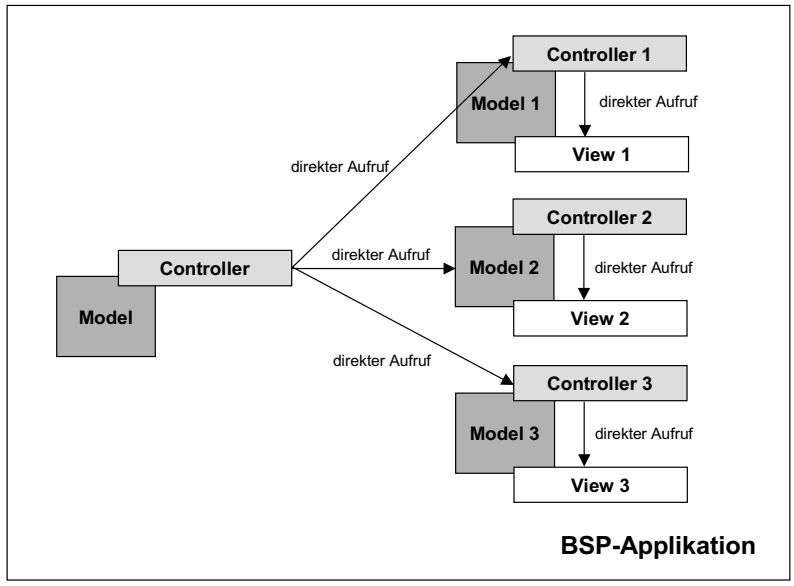

Abbildung 3.24 Ein Hauptcontroller steuert die zentrale Verteilung.

Komponentisierung

Es ist auch möglich, eine BSP-Seite dynamisch aus mehreren Views zu generieren. Diese Form der Komponentisierung ist allerdings nicht ganz einfach zu verwalten. Die folgende Abbildung 3.25 veranschaulicht diese Form der Verwendung von Controllern.

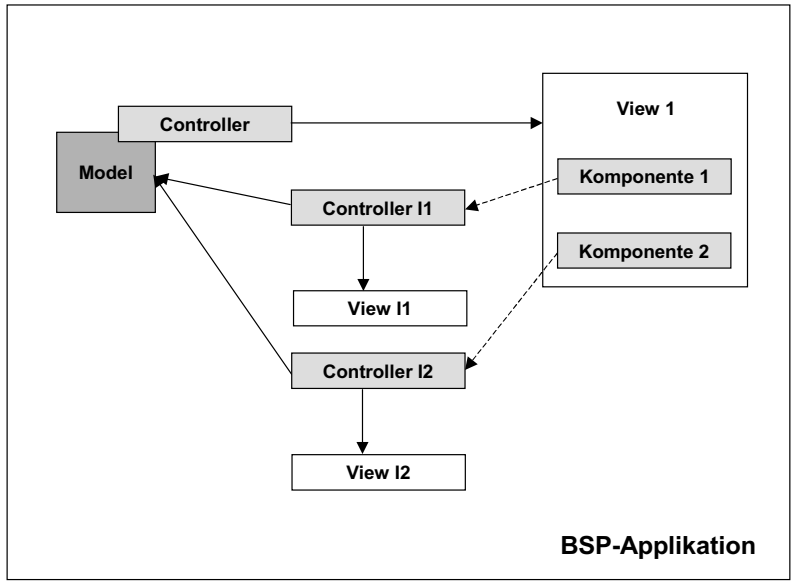

Abbildung 3.25 Komponentisierung von BSP-Seiten mit MVC

Selbstverständlich können auch Controller anderer Anwendungen aufgerufen werden. Die Beispiele zeigen, dass es eine Vielzahl an Kombinationsmöglichkeiten gibt, die fast jede erdenkliche Realisierungsform zulassen. Des Weiteren können bei all diesen Kombinationen BSP-Seiten mit Ablauflogik eingebunden werden.

Kombination von MVC mit bisherigen BSPs

Unter Berücksichtigung der folgenden Punkte können die Techniken des bisherigen Programmiermodells für BSP-Applikationen mit dem neu eingegliederten MVC-Design-Pattern kombiniert werden:

- Views können nur von Controllern aufgerufen werden. Ausnahme: Der Aufruf als Fehlerseite erlaubt die direkte Verwendung eines Views.
- Controller können unter Verwendung des `call`-Tags oder des `goto`-Tags einen Controller aufrufen. Sie können jedoch mit diesen Tags keine Seiten aufrufen.
- Übergänge von Seiten zu Controllern und zurück können mittels Redirect über die Navigationsmethoden stattfinden.

In einer BSP-Anwendung können also sowohl Seiten mit Ablauflogik als auch Controller und Views vorhanden sein.

Vorzüge des MVC-Design-Pattern

Der Einsatz des MVC-Design-Pattern erfordert natürlich einen gewissen Implementierungsaufwand. Dieser Aufwand wird aber durch entsprechende Vorteile wettgemacht:

- **Wartbarkeit**
  Durch die saubere Trennung von Präsentationslogik, Ablaufsteuerung und Anwendungslogik wird die Strukturierung vereinfacht. Dadurch erhöht sich die Wartungsfreundlichkeit. Designer können sich um das Design, Anwendungsentwickler um die Anwendung kümmern.
- **Performanz**
  Durch gezielten Einsatz der `goto`-Navigation wird die Anzahl der notwendigen Redirects reduziert und der gleiche Workprozess weiterverwendet, was häufig zu einer Ressourcenersparnis führt.

Abwägung von Aufwand und Vorteil

Es ist also durchaus zu überlegen, den erhöhten Aufwand in Kauf zu nehmen. Je komplexer ein Projekt bzw. die Anforderungen an eine BSP-Applikation werden, umso sinnvoller ist der Einsatz des Model-View-Controller-Design-Pattern.